



Unidad 1

Actividad 4



Gestión de Archivos

INTEGRANTES DEL EQUIPO:

COLIN RAMIRO JOEL
HERNÁNDEZ REYES JULIO CÉSAR
MALDONADO CERÓN CARLOS
MENDOZA GARCÍA ELIÚ EDUARDO

GRUPO: 4CM1

PROFESOR: CORTÉS GALICIA JORGE

¿Que es un archivo?

Se trata de un conjunto de información del mismo tipo como una unidad de almacenamiento.

Tienen las siguientes características

1. Deben ser capaces de contener grandes cantidades de información.
2. Su información debe permanecer y sobrevivir a los procesos que la generan ó utilizan.
3. Diferentes procesos, deben poder acceder a la información del archivo de manera concurrente.



Cada archivo debe incluir:

- **Atributos ó metadatos:**

- nombre
- fecha y hora de creación
- fecha y hora de la última actualización
- permisos(lectura,escritura,lectura y escritura)
- propietario
- contraseña de acceso
- tamaño

- **Datos**

- Información a almacenar dentro del mismo.

Estos archivos se almacenan en el dispositivo de memoria en forma de bloques(clusters).

Gestión de Archivos

Ahora bien, sabiendo lo que es un archivo, podemos comentar acerca de la gestión de estos en el Sistema Operativo.

Este Sistema Operativo posibilita que el usuario no tenga que conocer los detalles físicos de los dispositivos de E/S. Esto se logra con dos conceptos importantes:

- **Archivo.**- Se definió 2 diapositivas anteriores.
- **Directorio.**- Se define en la diapositiva siguiente

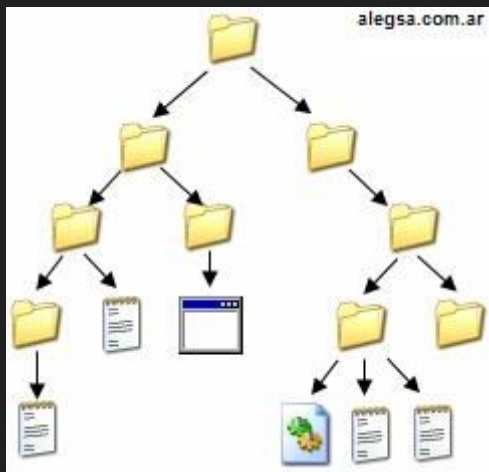


¿Qué es un Directorio?

Un directorio es un contenedor virtual donde se almacenan un conjunto de archivos, así como subdirectorios.

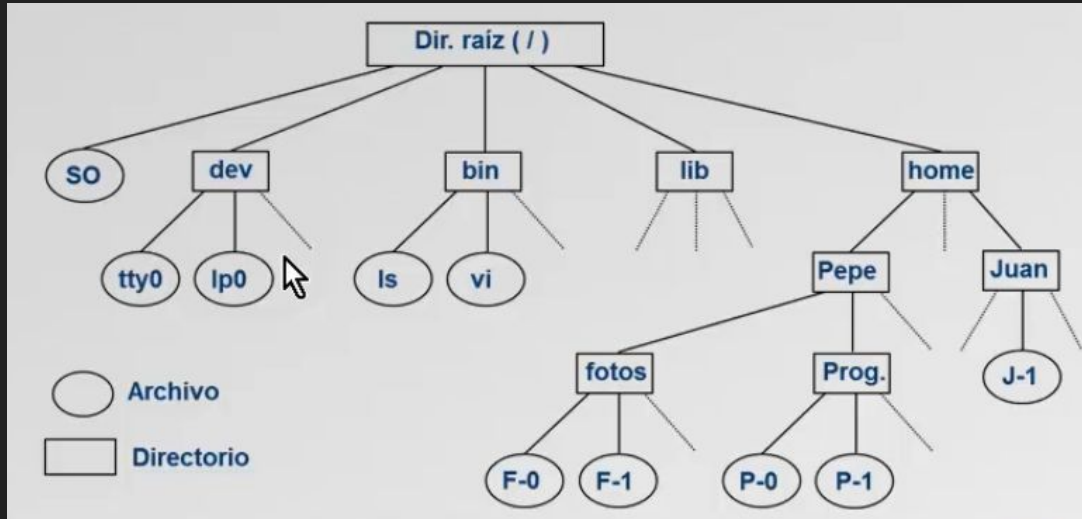
Permite al usuario organizar sus archivos

Es un archivo más, constituido por una tabla con una fila por cada archivo/subdirectorio integrado en el mismo.



Estructura Arbórea

Como pudimos observar en la imagen de la diapositiva anterior, los directorios tienen una estructura arbórea, cuyos nodos son subdirectorios y cuyas hojas son archivos.



Ruta(Path).- Es la lista de todos los directorios atravesados hasta llegar al archivo/directorio.

- **Path absoluto** = desde el directorio raíz (/)
- **Path relativo** = desde el directorio actual

Operaciones con Directorios

Para acceder a estos contenedores, se requieren de **llamadas al sistema**.

Las que están permitidas para administrarlos muestran variaciones más amplias de un sistema a otro que las **llamadas para archivos**.

A continuación se presentan algunos ejemplos de las llamadas al sistema más importantes para controlar estos directorios en UNIX:



CREATE.- Crea un directorio vacío.

DELETE.- Elimina el directorio(Solo si está vacío).

OPENDIR.- Abre el directorio para su manipulación.

CLOSEDIR.- Cierra el directorio una vez abierto, para liberar espacio de tablas internas

READDIR.- Devuelve la siguiente entrada de un directorio abierto

RENAME.- Se renombra al directorio

LINK.- Se trata de una técnica que le permite a un archivo aparecer en más de un directorio

UNLINK.- Se elimina una entrada de directorio

Sistema de Archivos

Una vez revisado lo que son los directorios, podemos avanzar a lo que es el **Sistema de Archivos**.

- El **Sistema de Archivos** es un elemento que controla cómo se almacenan y recuperan los datos.
- Sin este, los datos colocados en un medio de almacenamiento serían un gran cuerpo de datos flotando en el limbo.
- Es el encargado de administrar y facilitar el uso de las memorias periféricas ya sean secundarias o terciarias



Objetivos del Sistema de Archivos

A continuación se mencionan algunos de los objetivos principales de un Sistema de Archivos:

- ❖ Identificar y localizar un archivo
- ❖ Controlar el acceso varios usuarios a los archivos
- ❖ Bloquear el uso de archivos
- ❖ Ubicar archivos en bloques libres
- ❖ Administrar el espacio libre del soporte de información
- ❖ Garantizar que los datos introducidos son válidos
- ❖ Minimizar la pérdida de datos
- ❖ Brindar mecanismos de protección
- ❖ Ofrecer un conjunto estándar de funciones para manipularlos

Implementación de Sistema de Archivos

Componentes del Sistema de Archivos :


- **Administración de disco.**- Cómo organizar colección de bloques de discos en archivos.
- **Nombres.**- Los usuarios identifican sus archivos mediante nombres, abstrayéndose de cómo se almacenan internamente.
- **Protección.**- Cómo se protege la información de archivos en el sistema entre distintos usuarios y sistema.
- **Confiabilidad.**- Cuando el sistema “se cae”, se pierde información en Memoria(caché), pero se desea que información de archivos no se pierda


Implementación de Archivos


El aspecto más importante de la implementación del almacenamiento en archivos sea poder relacionar bloques de disco con archivos. Para esto se emplean diversos métodos en los diferentes sistemas operativos existentes, como pueden ser:

Asignación Continua.- El esquema de asignación más sencillo se puede decir que es almacenar cada archivo como un bloque contiguo de datos en el disco. Por ejemplo en un disco de 1K, a un archivo de 50k se le asignarían 50 bloques consecutivos.

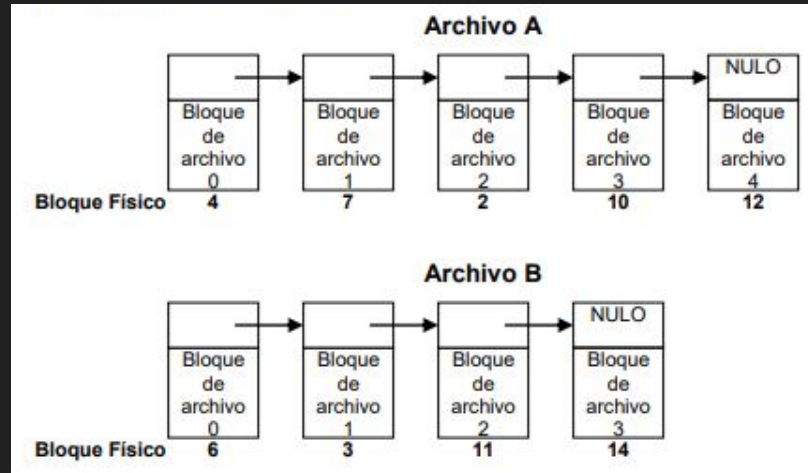
Implementación Sencilla 

Rendimiento excelente 

No es Factible 

Fragmentación del disco 

Asignación Por Lista Enlazada.- Se trata de almacenar archivos guardando cada uno como una lista enlazada bloques de disco.



Es posible utilizar todos los bloques ✓

El acceso aleatorio es lento ✗

No se pierde espacio por fragmentación ✓

Resulta menos eficiente ✗

Asignación Por Lista Enlazada empleando un índice.-

Las dos desventajas de la opción anterior, se eliminan si se toma la palabra del apuntador de cada bloque y se le coloca en una tabla o índice en la memoria.

Bloque Físico	
0	
1	
2	10
3	11
4	7
5	
6	3
7	2
8	
9	
10	12
11	14
12	NULO
13	
14	NULO
15	

Inicio del **archivo A**

Inicio del **archivo B**

Bloque no empleado

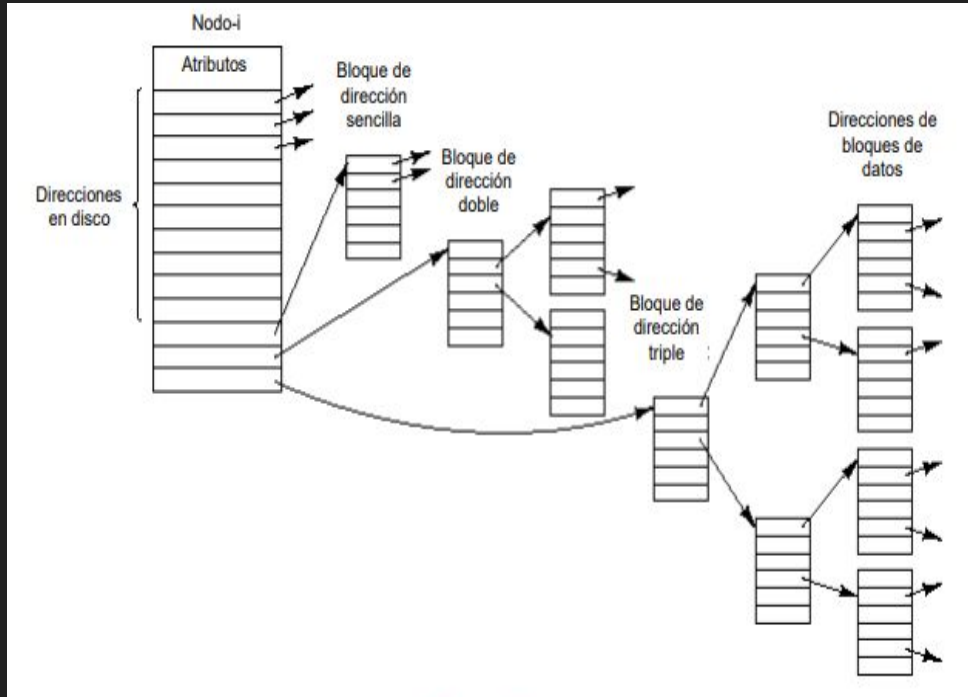
Todo el bloque queda disponible ✓

Acceso directo más sencillo ✓

Toda la tabla debe estar en la memoria

todo el tiempo ✗

Nodos-i.- Consiste en asociar a cada archivo una tabla llamada **nodo-i(nodo índice)**, la cual lista los atributos y las direcciones en disco de los bloques del archivo.



Las primeras direcciones se almacenan en el nodo-i, las cuales se refiere a los archivos pequeños, por lo que los archivos más grandes, una de las direcciones del nodo-i, es la dirección del bloque de disco llamado **bloque de indirección sencilla**. El cual contiene direcciones de disco adicionales.

Implementación de directorios



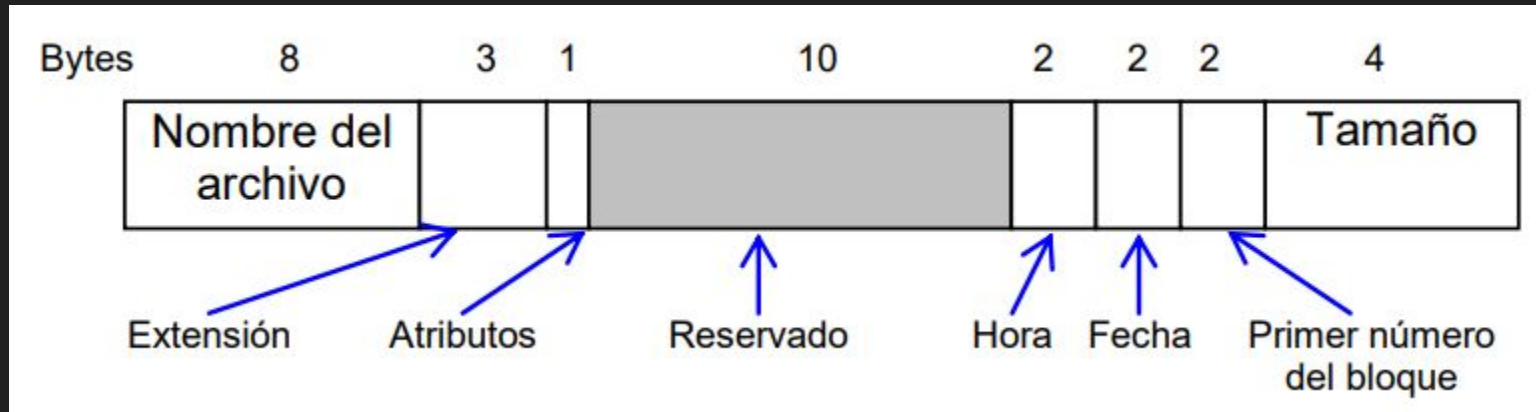
Cuando se abre un archivo, el sistema operativo usa el nombre de ruta proporcionado por el usuario para localizar la entrada de directorio. Esta entrada proporciona la información necesaria para encontrar los bloques de disco. Dependiendo del sistema, esta información puede ser la dirección en disco de todo el archivo (asignación contigua), el número del primer bloque (ambos esquemas de lista enlazada) o el número del nodo-i. En todos los casos, la función principal del sistema de directorios es transformar el nombre ASCII del archivo en la información necesaria para localizar los datos.

Un problema muy relacionado con el anterior es dónde deben almacenarse los atributos. Una posibilidad obvia es almacenarlos directamente en la entrada de directorio. Muchos sistemas hacen precisamente esto. En el caso de sistemas que usan nodos-i, otra posibilidad es almacenar los atributos en el nodo-i, en lugar de en la entrada de directorio. Este método tiene ciertas ventajas respecto a la colocación de los atributos en la entrada de directorio



Directorios en MS-DOS

Es un ejemplo de sistema con árboles de directorios jerárquicos. La Figura muestra una entrada de directorio de MS-DOS, la cual tiene 32 bytes de longitud y contiene el nombre de archivo, los atributos y el número del primer bloque de disco. El primer número de bloque se emplea como índice de una tabla. Siguiendo la cadena, se pueden encontrar todos los bloques.



En MS-DOS, los directorios pueden contener otros directorios, dando lugar a un sistema de archivos jerárquico. En este sistema operativo es común que los diferentes programas de aplicación comience por crear un directorio en el directorio raíz y pongan ahí todos sus archivos, con objeto de que no haya conflictos entre las aplicaciones.

```
Starting MS-DOS...
```

```
HIMEM is testing extended memory...done.
```

```
C:\>C:\DOS\SMARTDRV.EXE /X
```

```
MODE prepare code page function completed
```

```
MODE select code page function completed
```

```
C:\>dir
```

```
Volume in drive C is MS-DOS_6  
Volume Serial Number is 40B4-7F23  
Directory of C:\
```

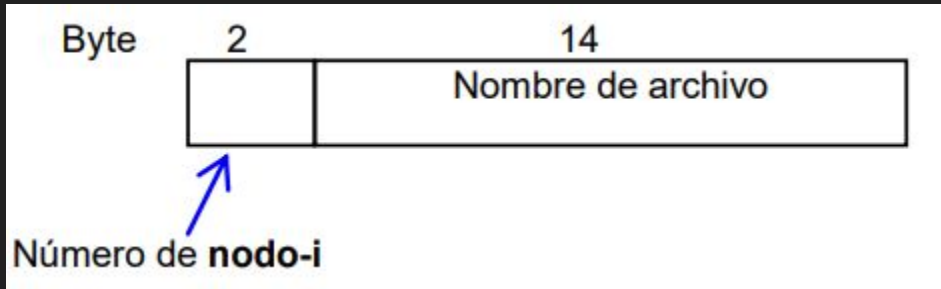
DOS	<DIR>	12.05.20	15:57
COMMAND	COM	54 645 94.05.31	6:22
WINA20	386	9 349 94.05.31	6:22
CONFIG	SYS	144 12.05.20	15:57
AUTOEXEC	BAT	188 12.05.20	15:57
5 file(s)		64 326 bytes	
		24 760 320 bytes free	

```
C:\>_
```

Directorios en UNIX

Cada entrada contiene sólo un nombre de archivo y su número de nodo-i, Toda la información acerca del tipo, tamaño, tiempos, propietario y bloques de disco está contenida en el nodo-i.

Cuando se abre un archivo, el sistema de archivos debe tomar el nombre que se le proporciona y localizar sus bloques de disco.



Supongamos que se busca el nombre de ruta /usr/ast/mbox...

Lo primero que hace el sistema de archivos es localizar el directorio raíz. En UNIX su nodo-i está situado en un lugar fijo del disco. A continuación, el sistema de archivos busca el primer componente de la ruta, usr, en el directorio raíz para encontrar el número de nodo-i del archivo /usr. La localización de un nodo.i, una vez que se tiene su número es directa, ya que cada uno tiene una posición fija en el disco. Con la ayuda de este nodo-i, el sistema localiza el directorio correspondiente a /usr y busca el siguiente componente, ast, en él. Una vez que el sistema encuentra la entrada para ast, obtiene el nodo-i del directorio /usr/ast. Con este nodo, el sistema puede encontrar el directorio mismo y buscar mbox. A continuación se trae a la memoria el nodo-i de este archivo y se mantiene ahí hasta que se cierra el archivo.

Directorio raíz

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

La búsqueda **usr**
produce **nodo-i 6**

El **nodo-i 6** es para
/usr

Modo
Tamaño
Tiempos
132

El **nodo-i 6** dice
que /usr está en el
bloque 132

El **bloque 132** es el
directorio /usr

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

usr/ast es el
nodo-i 26

El **nodo-i 26** es
para /usr/ast

Modo
Tamaño
Tiempos
406

El **nodo-i 26** dice
que /usr/ast está
en el **bloque 406**

El **bloque 406** es el
directorio /usr/ast

26	.
6	..
64	grants
92	books
60	mbox
81	scr
17	src

usr/ast/mbox es
el **nodo-i 60**

Administración del espacio en disco

Hay dos posibles estrategias para almacenar un archivo de n bytes:

- Asignar n bytes consecutivos de espacio en disco
- Dividir el archivo en varios bloques (no necesariamente) contiguos.

El almacenamiento de un archivo como secuencia contigua de bytes tiene el problema obvio de que si el archivo crece, probablemente tendrá que pasarse a otro lugar del disco. El mismo problema ocurre con los segmentos en la memoria, excepto que el traslado de un segmento en la memoria es una operación relativamente rápida comparada con el traslado de un archivo de una posición en el disco a otra. Por esta razón, casi todos los sistemas de archivos dividen los archivos en bloques de tamaño fijo que no necesitan estar adyacentes.

Tamaño de bloque

¿Qué tamaño deben tener los bloques?

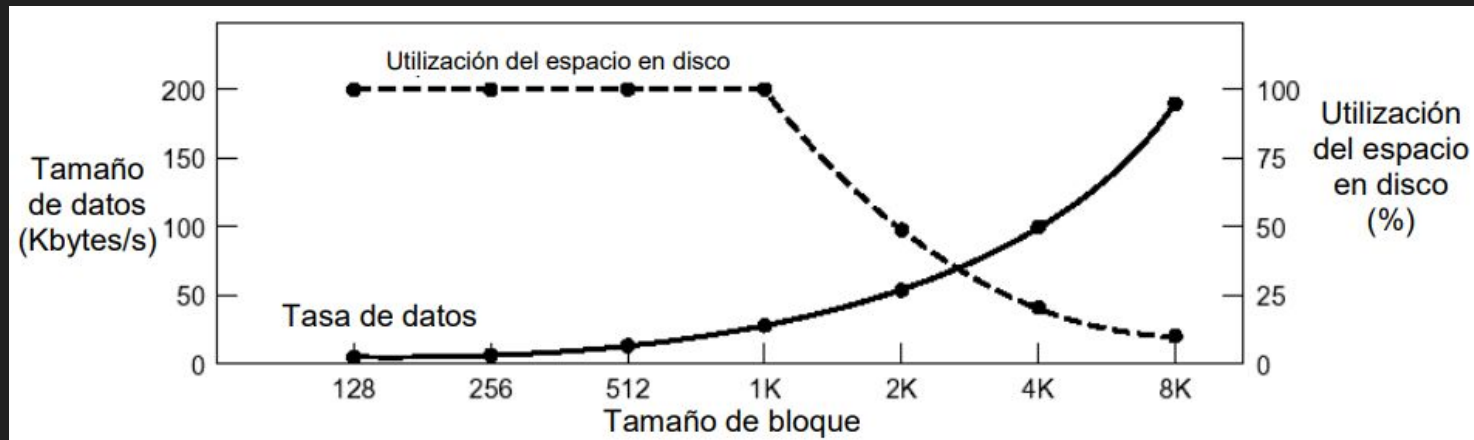
Dada la forma como están organizados los discos, el sector, la pista y el cilindro son candidatos obvios para utilizarse como unidad de asignación.

Tener una unidad de asignación grande, como un cilindro, implica que cada archivo, incluso un archivo de un byte, ocupará todo un cilindro. Estudios realizados han demostrado que la mediana del tamaño de los archivos en los entornos UNIX es de alrededor de 1K, así que asignar un cilindro de 32K a cada archivo implicaría un desperdicio de $31 / 32 = 97\%$ del espacio en disco total. Por otro lado, el empleo de una unidad de asignación pequeña implica que cada archivo consistirá en muchos bloques. La lectura de cada bloque normalmente requiere una búsqueda y un retardo rotacional, así que la lectura de un archivo que consta de muchos bloques pequeños será lenta.

La siguiente gráfica muestra la tasa de datos para un disco con las siguientes características:

- 32,768 bytes por pista
- Tiempo de rotación de 16.67 ms
- Tiempo de búsqueda medio de 30 ms

La curva indica la eficiencia de espacio del disco



Como podemos observar en la gráfica anterior, una buena utilización del espacio implica tasas de datos bajas, y viceversa. La eficiencia de tiempo y la del espacio chocan entre sí. El término medio usual es escoger un tamaño de bloques de 512, 1K o 2K bytes.

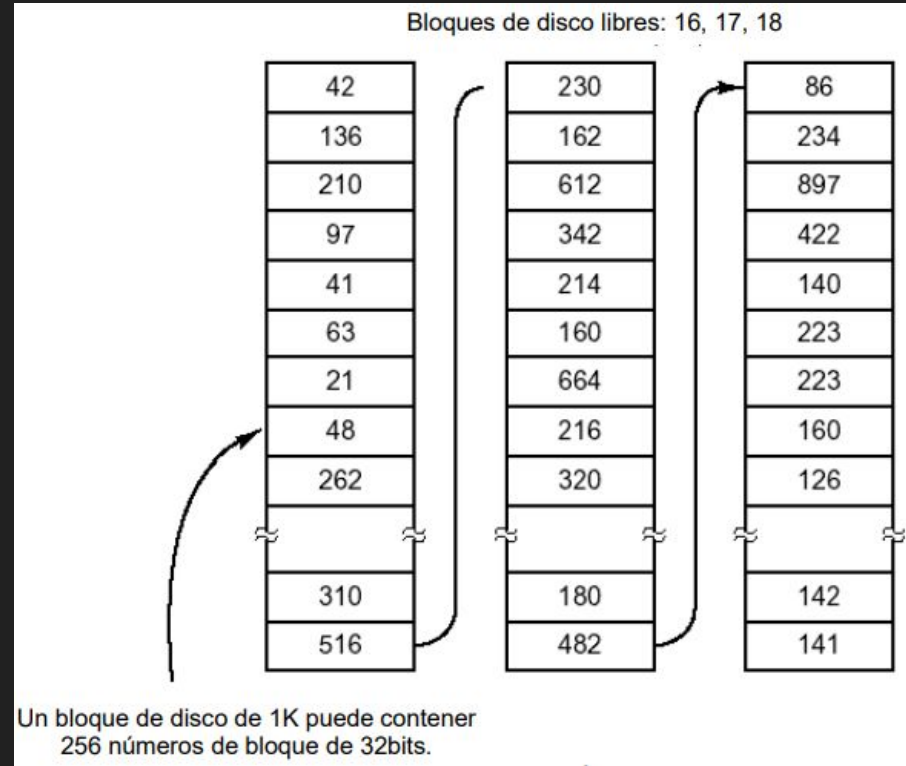
Sea cual sea la decisión que se tome, probablemente convendrá reevaluar periódicamente, ya que, al igual que sucede con todos los aspectos de la tecnología de computadoras, los usuarios aprovechan los recursos más abundantes exigiendo todavía más.

Administración de bloques libres

¿Cómo seguir la pista a los bloques libres?

Se utilizan ampliamente dos métodos.

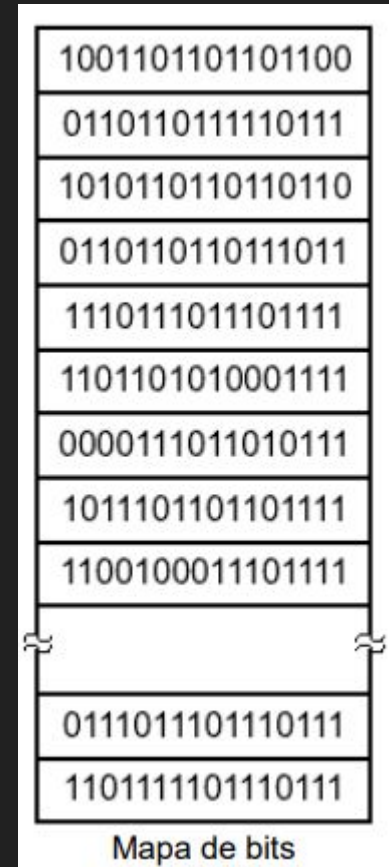
El primero consiste en usar una lista enlazada de bloques de disco, en la que cada bloque contiene tantos números de bloques de disco como quepan en él.



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 con unos en el mapa, y los bloques asignados con ceros (o viceversa).

Sólo si el disco está casi lleno el esquema de lista enlazada requerirá menos bloques que el mapa de bits.

Si hay suficiente memoria principal para contener el mapa de bits, este método generalmente es preferible. En cambio, si sólo se puede dedicar un bloque de memoria para seguir la pista a los bloques de disco libres, y el disco está casi lleno, la lista enlazada podría ser mejor.

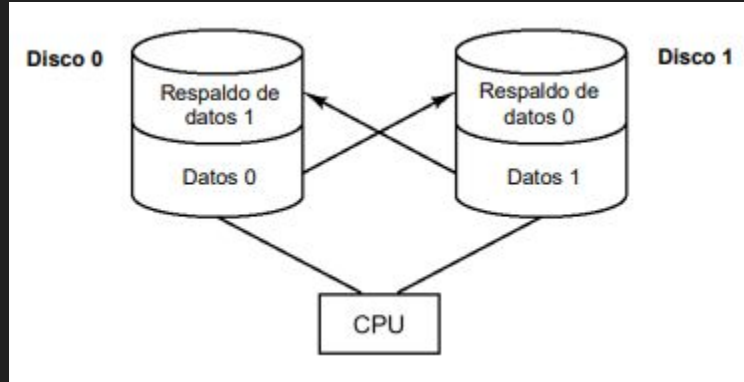


5.3.4 Confiabilidad del sistema de archivos

La destrucción de un sistema de archivos suele ser un desastre mucho peor que la destrucción de una computadora. Si el sistema de archivos de una computadora se pierde irremisiblemente, sea por causa del hardware, del software, la restauración de toda la información es difícil, tardada y, en muchos casos, imposible. Los discos pueden tener bloques defectuosos. Los discos flexibles normalmente son perfectos cuando salen de la fábrica, pero pueden adquirir defectos durante el uso.

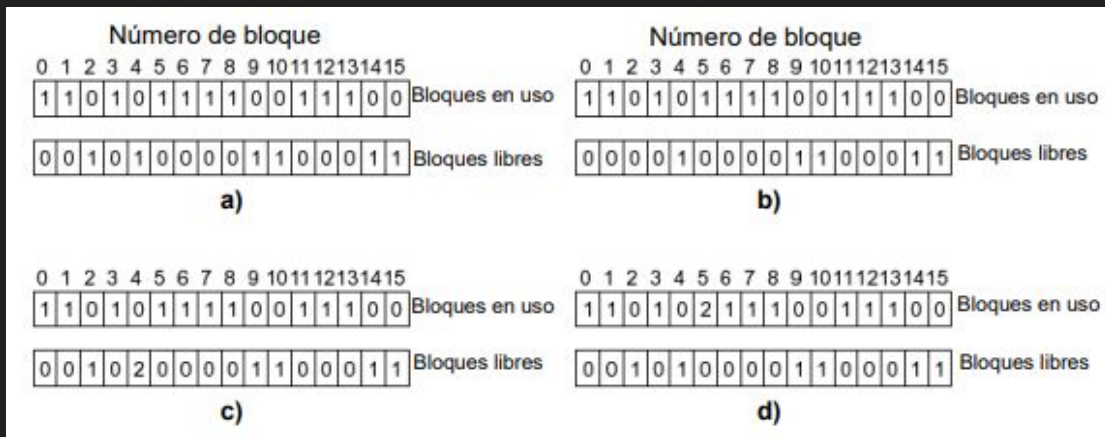
Respaldos

Los sistemas de archivos en disco flexible se pueden respaldar con sólo copiar todo el disco en uno en blanco. Los sistemas de archivos en discos winchester pequeños se pueden respaldar vaciando todo el disco en cinta magnética. Entre las tecnologías actuales están los cartuchos de cinta de 150M y las cintas Exabyte o DAT de 8G.



Consistencia del sistema de archivos

Otra área en la que la confiabilidad es importante es la consistencia del sistema de archivos. Muchos sistemas de archivos leen bloques, los modifican y los vuelven a escribir después. Si el sistema se cae antes de que todos los bloques modificados se hayan escrito en disco, el sistema de archivos puede quedar en un estado inconsistente. Este problema se vuelve crítico si algunos de los bloques que no se han escrito contienen nodos-i, entradas de directorio o la lista libre.

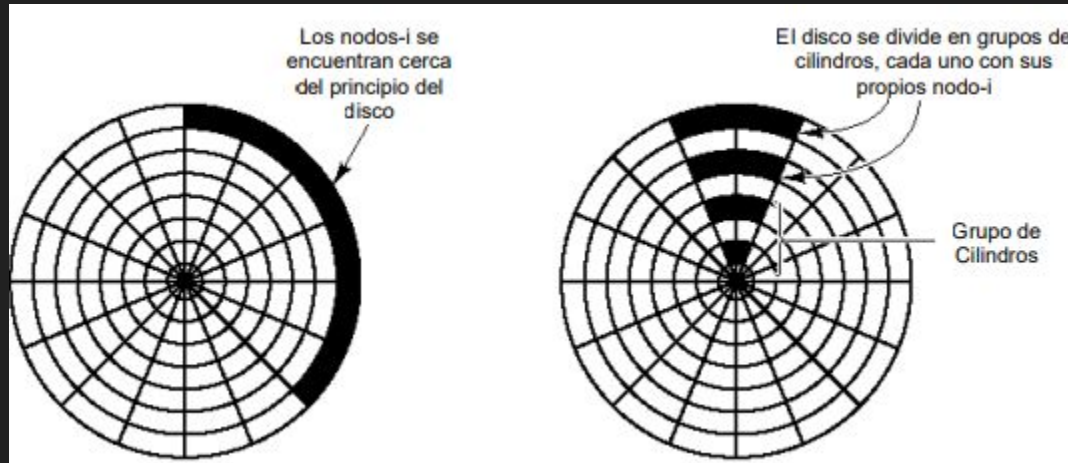


5.3.5 Rendimiento del sistema de archivos

El acceso a un disco es mucho más lento que el acceso a la memoria. La lectura de una palabra de memoria por lo regular toma decenas de nanosegundos.

La técnica más común empleada para reducir los accesos a disco es el caché de bloques o el caché de buffer (la palabra caché proviene del verbo francés cacher, que significa esconder). En este contexto, un caché es una colección de bloques que lógicamente pertenecen al disco pero que se están manteniendo en la memoria por razones de rendimiento.

Una forma fácil de mejorar el rendimiento es colocar los nodos-i en la parte media del disco, no al principio, reduciendo así a la mitad el movimiento medio del brazo para desplazarse del nodo-i al primer bloque. Otra idea es dividir el disco en grupos cilíndricos, cada uno con sus propios nodos-i, bloques y lista libre. Al crear un archivo nuevo se puede escoger cualquier nodo-i, pero se intenta encontrar un bloque que esté en el mismo grupo de cilindros que el nodo-i. Si no hay uno disponible, se usa un bloque de un cilindro cercano.



5.3.6 Sistemas de archivos estructurados por diario

Los cambios tecnológicos están ejerciendo presión sobre los sistemas de archivos actuales. En particular, las CPU cada vez son más rápidas, los discos cada vez son más grandes y económicos y el tamaño de las memorias está creciendo exponencialmente. El único parámetro que no está mejorando a pasos agigantados es el tiempo de búsqueda en disco

LFS (el sistema de archivos estructurado por diario, log-structured file system).

La idea en que se basa el diseño LFS es que conforme las CPU se hacen más rápidas y las memorias RAM se hacen más grandes, los caches de disco están aumentando aceleradamente. En consecuencia, ya es posible satisfacer una fracción sustancial de todas las solicitudes de lectura directamente del caché del sistema de archivos, sin tener que acceder al disco.