



Instituto Politecnico Nacional  
Escuela Superior de Cómputo



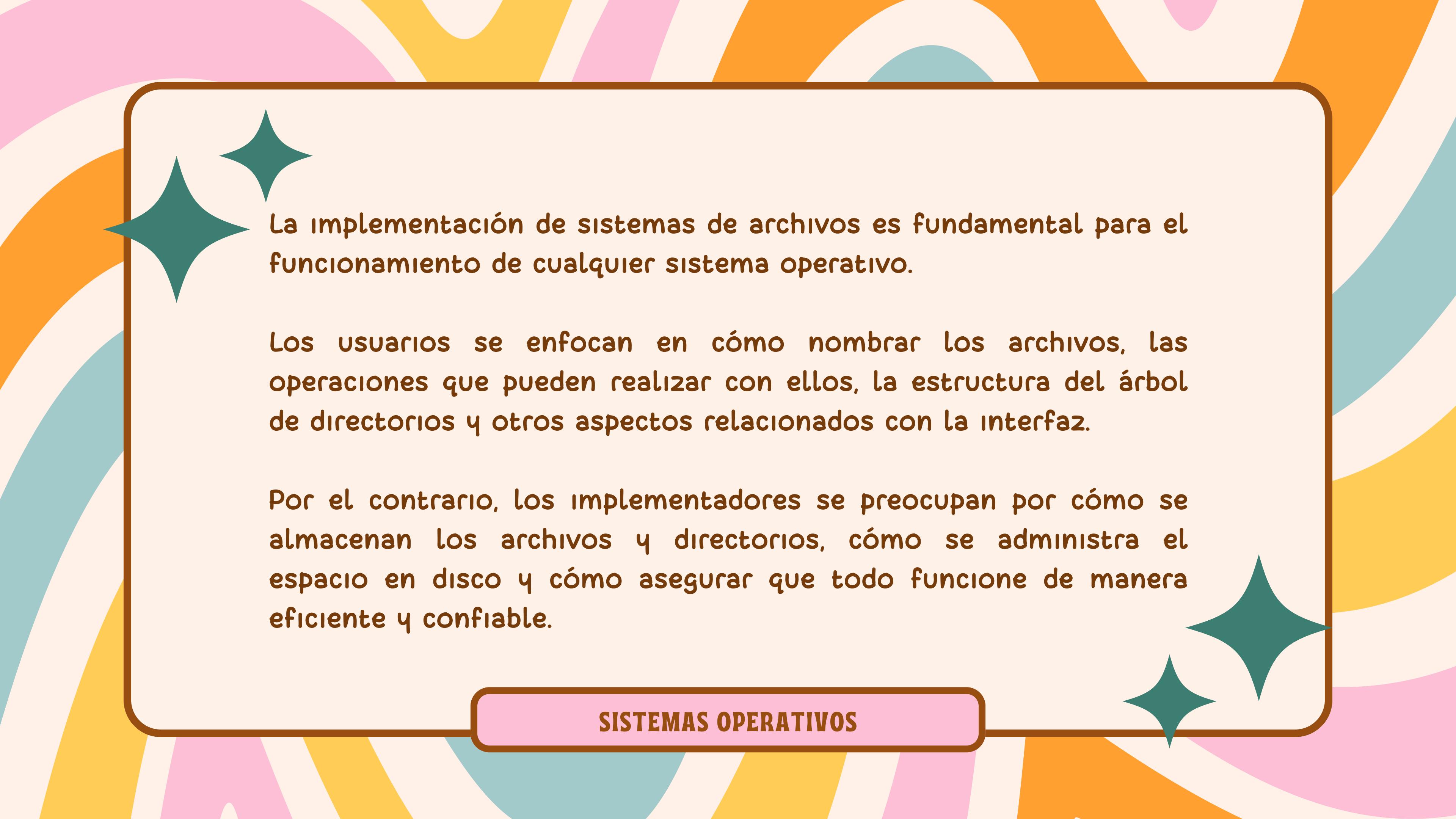
# Unidad 4

4.3 Métodos de asignación de espacio, 4.4 Sistemas de archivos estructurados y 4.5 Optimización del sistema de archivos

SISTEMAS OPERATIVOS

# Implementación de sistemas de Archivos e Implementación de archivos

SISTEMAS OPERATIVOS



**La implementación de sistemas de archivos es fundamental para el funcionamiento de cualquier sistema operativo.**

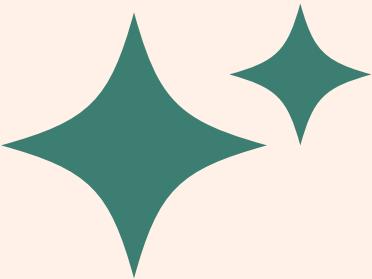
Los usuarios se enfocan en cómo nombrar los archivos, las operaciones que pueden realizar con ellos, la estructura del árbol de directorios y otros aspectos relacionados con la interfaz.

Por el contrario, los implementadores se preocupan por cómo se almacenan los archivos y directorios, cómo se administra el espacio en disco y cómo asegurar que todo funcione de manera eficiente y confiable.

**SISTEMAS OPERATIVOS**

# Asignación contigua

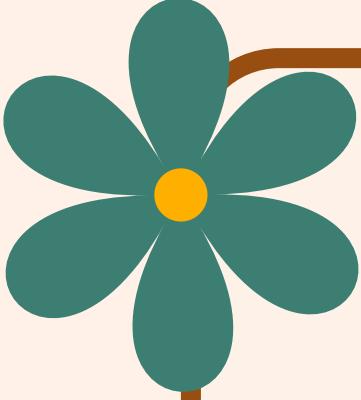
SISTEMAS OPERATIVOS



# Asignación contigua

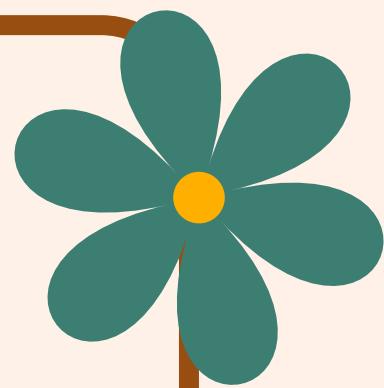
## CONCEPTO

Almacenar cada archivo como un bloque contiguo de datos en el disco, facilitando el acceso secuencial. Así, en un disco con bloques de 1K, a un archivo de 50K se le asignarían 50 bloques consecutivos.



## Ventajas

- Implementación es sencilla porque para saber dónde están los bloques de un archivo basta con recordar un número (la dirección del disco del primer bloque).
- El rendimiento es excelente porque es posible leer todo el archivo del disco en una sola operación.

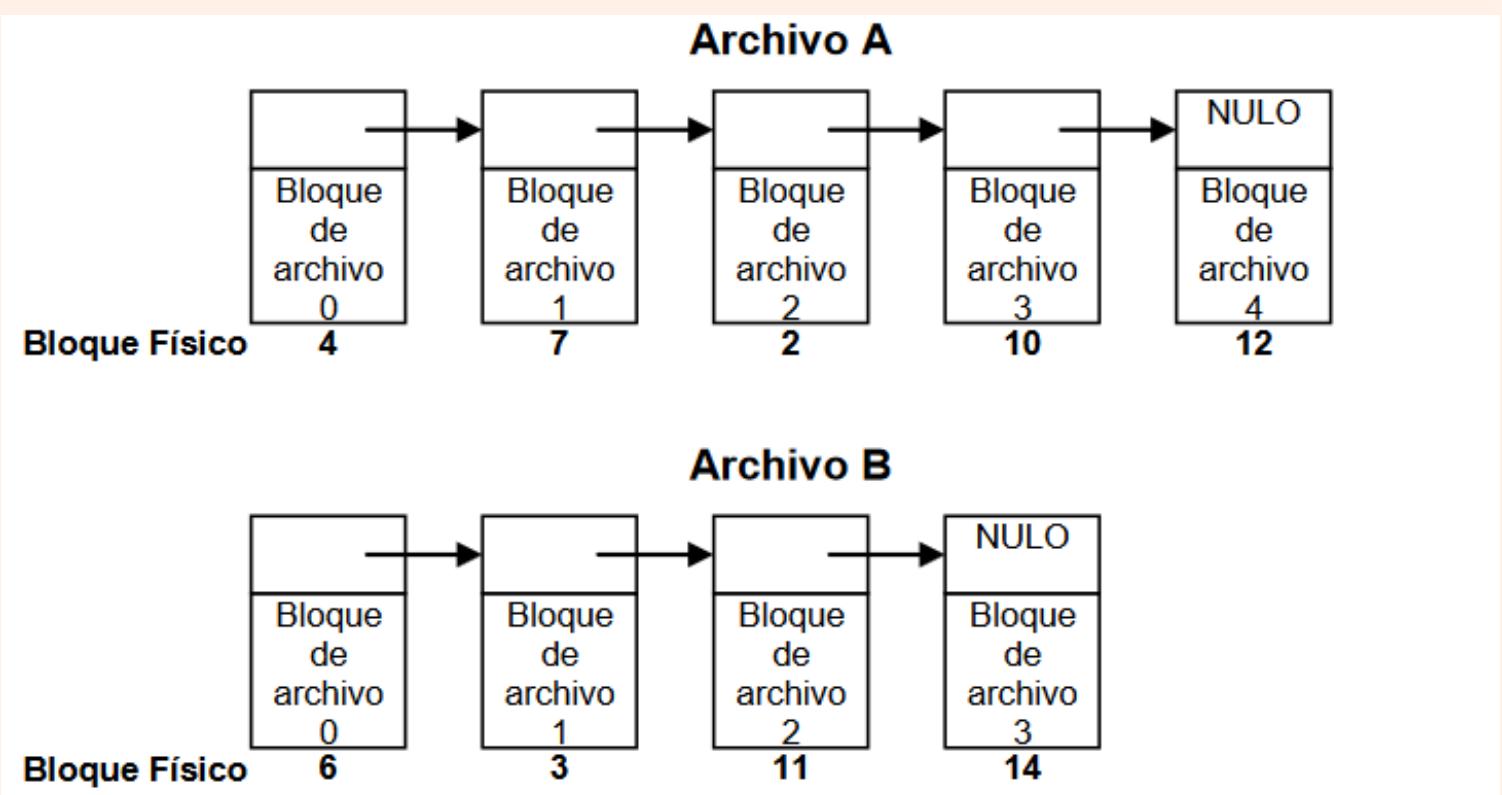


## Desventajas

- No es factible si no se conoce el tamaño máximo del archivo en el momento en que se crea el archivo.
- Fragmentación del disco que resulta de esta política de asignación. Se desperdicia espacio que de otra forma podría haberse aprovechado.

# Asignación por lista enlazada

SISTEMAS OPERATIVOS



## Listado enlazado de bloques de disco

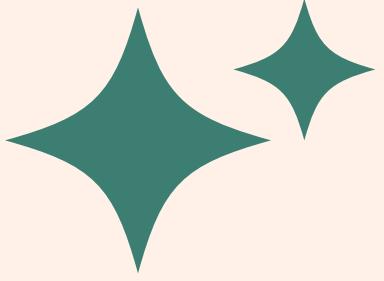
- La primera palabra de cada bloque se emplea como apuntador al siguiente.
- Con este método es posible utilizar todos los bloques.
- No se pierde espacio por fragmentación del disco.

- Aunque la lectura secuencial de un archivo es sencilla, el acceso aleatorio es extremadamente lento.
- La cantidad de almacenamiento de datos en un bloque ya no es una potencia de dos porque el apuntador ocupa unos cuantos bytes.
- No es fatal, pero resulta menos eficiente porque muchos programas leen y escriben en bloques cuyo tamaño es una potencia de dos.

SISTEMAS OPERATIVOS

# Asignación por lista enlazada empleando un índice

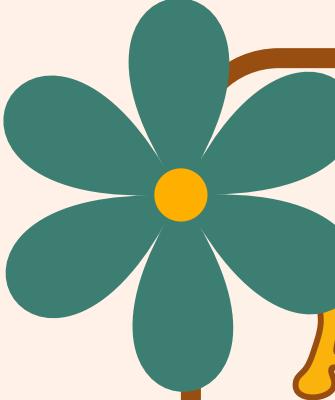
SISTEMAS OPERATIVOS



# Asignación por lista enlazada empleando un índice

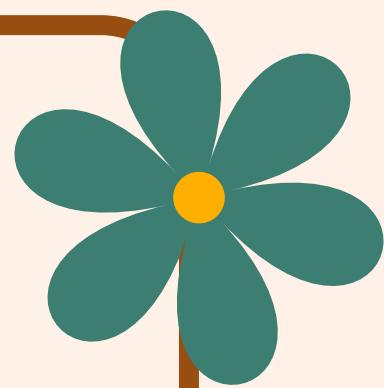
## CONCEPTO

La asignación por lista enlazada empleando un índice mejora la gestión de archivos en un sistema de archivos eliminando algunas desventajas de la asignación por lista enlazada tradicional. Se utiliza una tabla o índice en la memoria para realizar un seguimiento de los bloques de disco que componen un archivo. Esto permite un acceso más eficiente y directo a los datos.



## Desventajas de la Asignación por Lista Enlazada

- **Acceso Secuencial:** En la asignación tradicional, se necesita seguir la cadena de bloques desde el principio hasta encontrar el bloque deseado, lo que es lento.
- **Bloques con Punteros:** Parte del espacio de cada bloque se usa para almacenar punteros, reduciendo el espacio disponible para los datos.



## Solución: Uso de una Tabla en Memoria

- Se elimina la necesidad de almacenar punteros dentro de los bloques de datos, dejando todo el bloque disponible para datos.
- La tabla contiene la cadena de bloques en memoria, permitiendo un acceso directo a cualquier bloque.

# EJEMPLO

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

**Figura 5.9:** Asignación por lista enlazada empleando una tabla en la memoria principal.

## Ventajas del Método con Índice

- **Acceso Directo:** El acceso a un bloque específico es rápido, solo se necesita consultar la tabla.
- **Simplificación del Código:** La localización de todos los bloques es sencilla.
- **Uso Completo de Bloques:** Todo el bloque está disponible para datos.

## Desventaja Principal

- Para discos grandes, esto significa que la tabla puede ocupar una cantidad considerable de memoria.

# Nodos-i

SISTEMAS OPERATIVOS



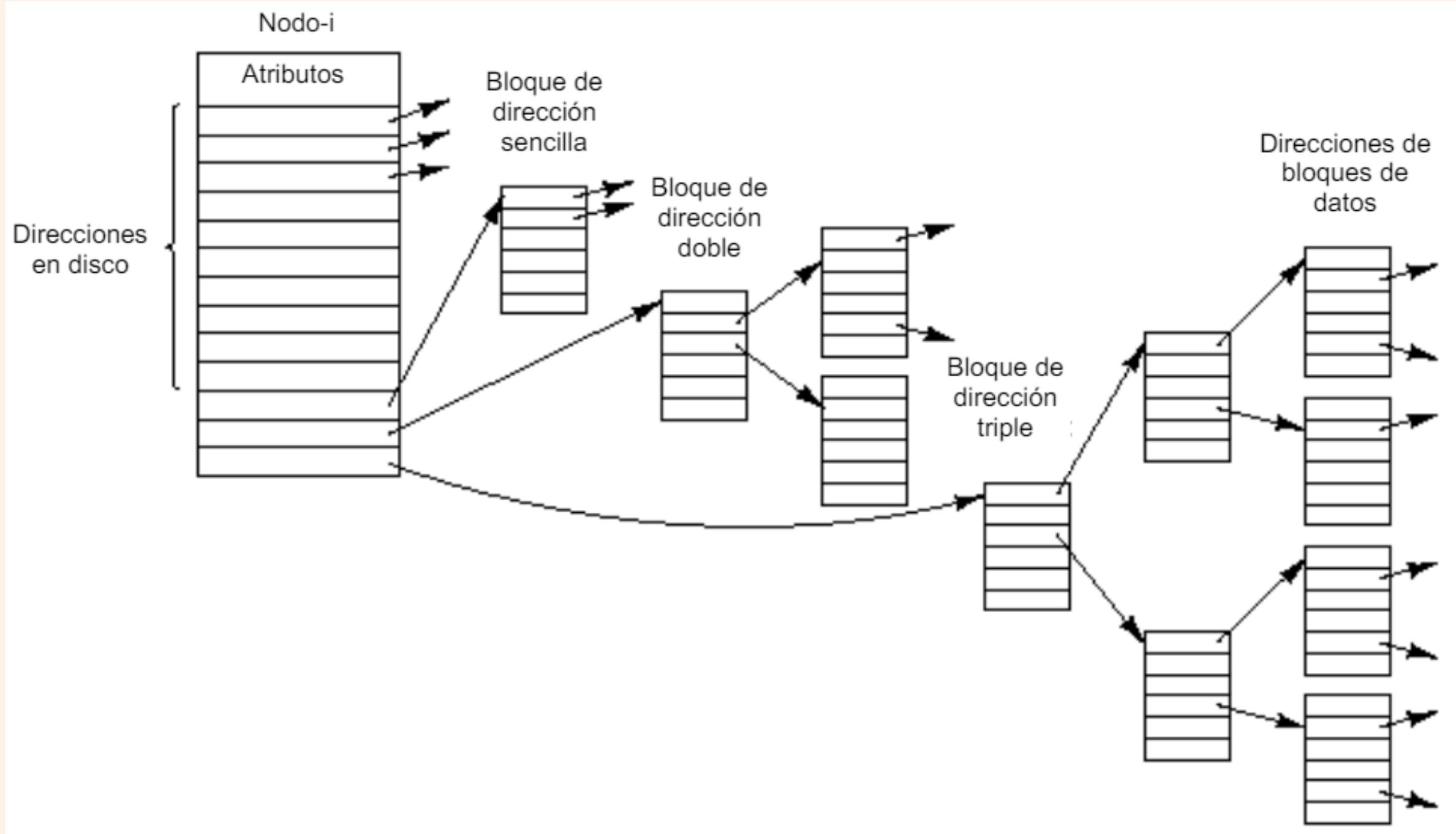
El último método para relacionar bloques de disco con archivos es asociar a cada archivo una pequeña tabla llamada nodo-i (nodo-índice), que lista los atributos y las direcciones en disco de los bloques del archivo.

Las primeras direcciones de disco se almacenan en el nodo-i mismo, lo que permite que la información de archivos pequeños se contenga completamente en el nodo-i.

Para archivos más grandes, el nodo-i utiliza bloques de indirección sencilla, doble y triple para gestionar más direcciones de disco. Este esquema es utilizado por UNIX.



# REPRESENTACIÓN



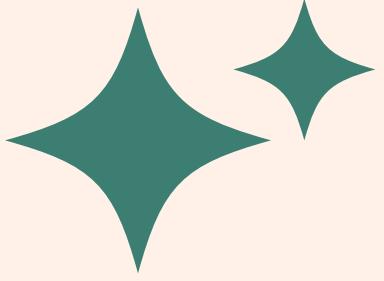
# Implementación de directorios

Cuando se abre un archivo, el sistema operativo usa el nombre de ruta proporcionado por el usuario para localizar la entrada de directorio.

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.

La función principal del sistema de directorios es transformar el nombre ASCII del archivo en la información necesaria para localizar los datos.

SISTEMAS OPERATIVOS



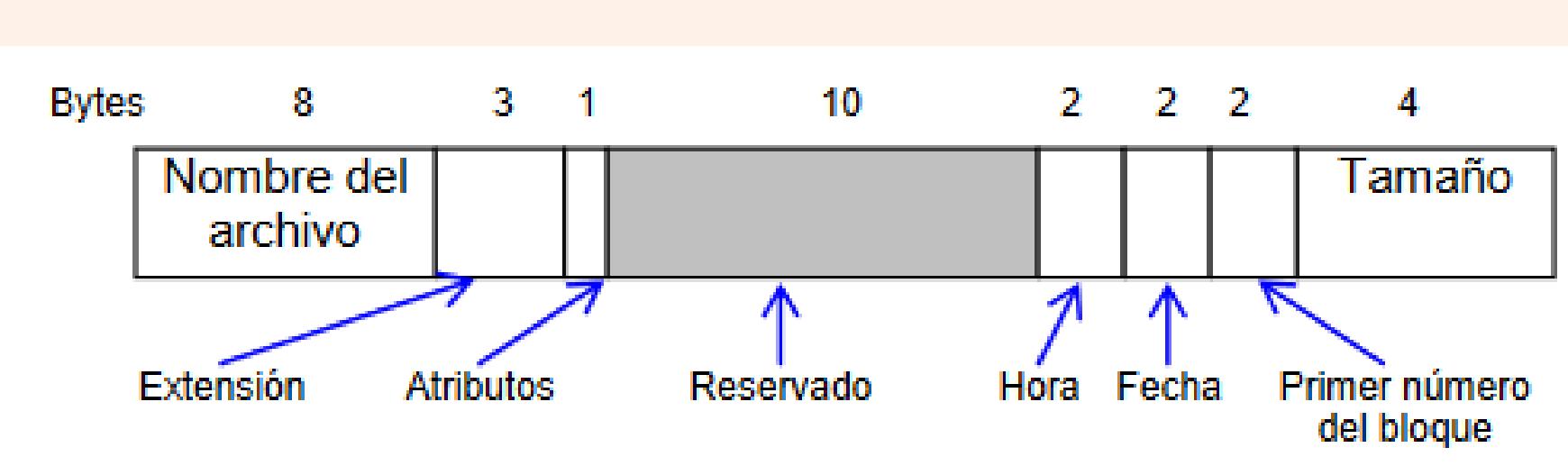
# Directorios en MS-DOS

- MS-DOS es un sistema con árboles de directorios jerárquicos.
- Los directorios pueden contener otros directorios.
- Es común que los diferentes programas de aplicación comiencen por crear un directorio en el directorio raíz y pongan ahí todos sus archivos, con objeto de que no haya conflictos entre las aplicaciones.

# Entrada de directorio

## Características

- 32 bytes de longitud.
- Nombre del archivo.
- Atributos.
- Número del primer bloque de disco (índice de tabla)



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

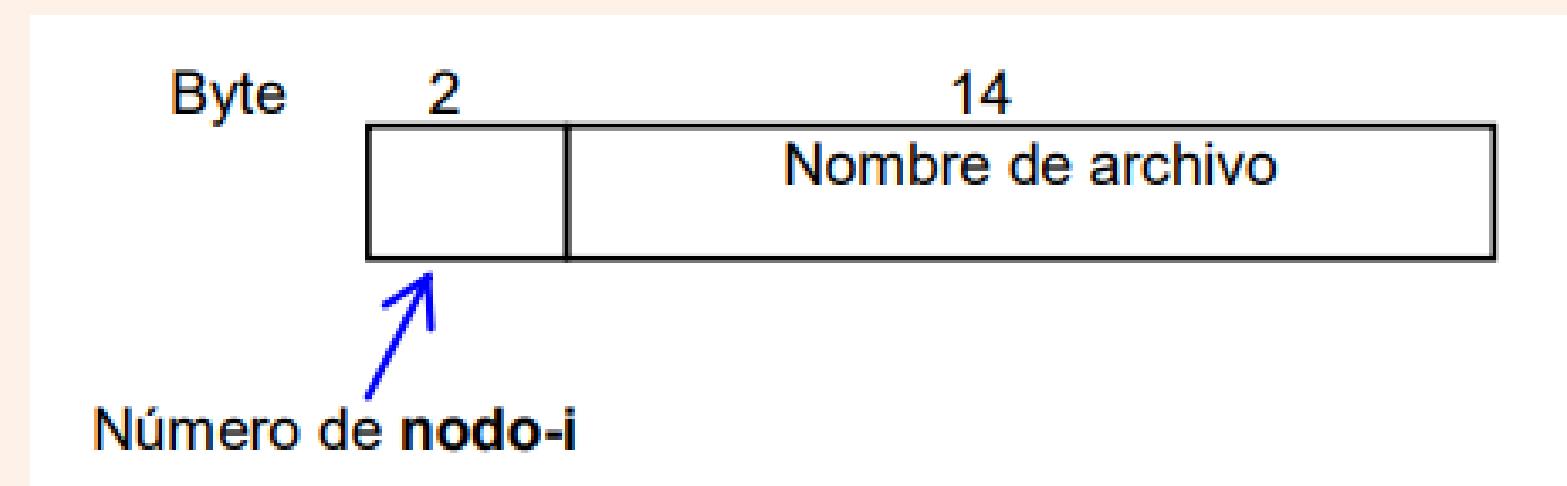
Etiquetas para los bloques:

- Inicio del archivo A (bloques 2 y 3)
- Inicio del archivo B (bloque 10)
- Bloque no empleado (bloques 12 y 13)

# Directorios en UNIX

## Estructura de Directorios en UNIX

- Cada entrada de directorio contiene solo el nombre del archivo y su número de nodo-i.
- El nodo-i almacena toda la información del archivo: tipo, tamaño, tiempos, propietario y bloques de disco.



SISTEMAS OPERATIVOS

# Directorios en UNIX

- Directorio Raíz: El sistema localiza el nodo-i del directorio raíz, que está en una posición fija del disco.
- Buscar Componentes:
- Busca el primer componente usr en el directorio raíz y obtiene su nodo-i.
- Con el nodo-i de usr, localiza el directorio correspondiente y busca el siguiente componente ast.
- Finalmente, busca mbox dentro del directorio ast.

# Directorios en UNIX

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**

# Directorios en UNIX

## Nombres de Ruta Relativos

Los nombres de ruta relativos se buscan igual que los absolutos, pero partiendo del directorio de trabajo en lugar del directorio raíz:

- Las entradas `.` y `..` representan el directorio actual y el directorio padre, respectivamente.
- Por ejemplo, `../dick/prog.c` busca `..` (directorío padre), luego `dick` y finalmente `prog.c` en ese directorio.

## Nombres de Ruta Relativos

Los nombres de ruta relativos se buscan igual que los absolutos, pero partiendo del directorio de trabajo en lugar del directorio raíz:

- Las entradas . y .. representan el directorio actual y el directorio padre, respectivamente.
- Por ejemplo, `../dick/prog.c` busca .. (directorío padre), luego dick y finalmente prog.c en ese directorio.

# Administración del espacio en disco

SISTEMAS OPERATIVOS

La administración del espacio en disco es un punto importante para los diseñadores de sistemas de archivos, ya que los archivos se almacenan en discos. Existen dos estrategias para almacenar un archivo de  $n$  bytes: asignar  $n$  bytes consecutivos de espacio en disco o dividir el archivo en varios bloques no necesariamente contiguos. Almacenar un archivo como una secuencia contigua de bytes puede ser problemático si el archivo crece, ya que probablemente tendría que ser trasladado a otro lugar del disco. Por esta razón, casi todos los sistemas de archivos dividen los archivos en bloques de tamaño fijo que no necesitan estar adyacentes.

SISTEMAS OPERATIVOS

# Tamaño del Bloque

## Sector

La unidad más pequeña que el disco puede leer o escribir.

## Pista

Una colección de sectores en una circunferencia del disco.

## Cilindro

Un conjunto de pistas alineadas verticalmente a través de varios platos.

## Página

tamaño página.  
de

**UNIDAD DE ASIGNACIÓN**

# Impacto del Tamaño de Bloque

## GRANDE

Asignar unidades grandes, como un cilindro, puede llevar a un desperdicio significativo de espacio. 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  $31 / 32 = 97\%$ .

## PEQUEÑO

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.

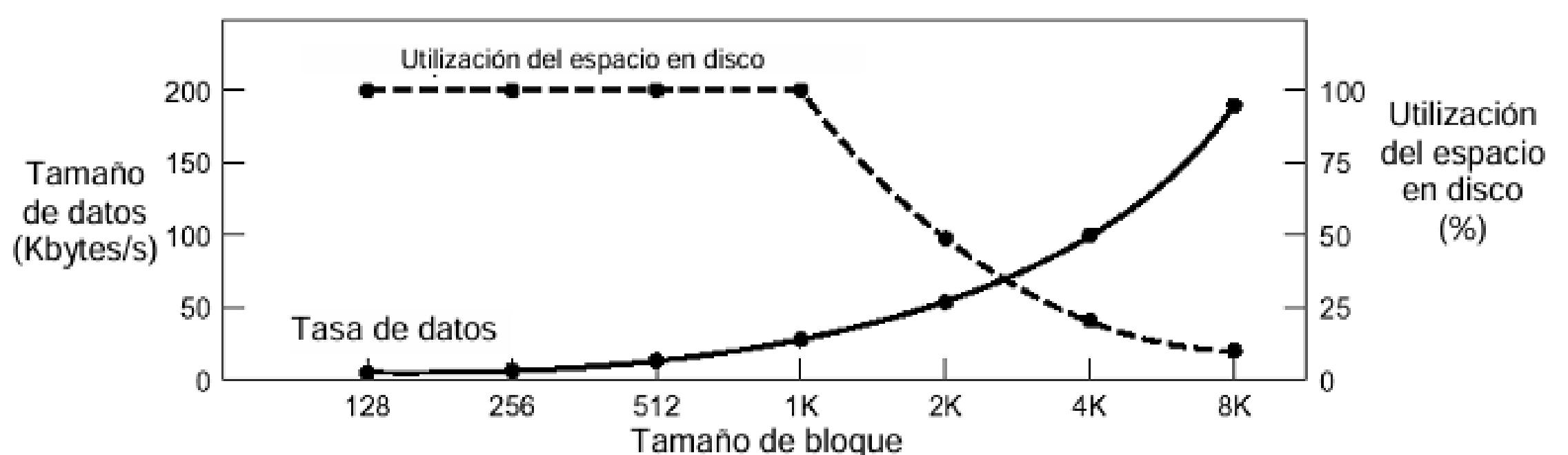
# Ejemplo

## Características

Consideremos un disco con 32,768 bytes por pista, tiempo de rotación de 16.67 ms y tiempo de búsqueda medio de 30 ms. El tiempo en milisegundos requerido para leer un bloque de  $k$  bytes es la suma de los tiempos de búsqueda, retardo rotacional y transferencia:

$$\text{Tiempo de lectura} = \text{Tiempo de búsqueda} + \text{Retardo rotacional} + \left( \frac{k}{\text{Bytes por pista}} \right) \times \text{Tiempo de rotación}$$

$$\text{Tiempo de lectura} = 30 \text{ ms} + 8.3 \text{ ms} + \left( \frac{k}{32,768} \right) \times 16.67 \text{ ms}$$



Continua: Tasa de datos en función del tamaño de bloque

Punteada: Indica la eficiencia de espacio del disco

# TÉCNICAS DE ADMINISTRACIÓN DE ESPACIO LIBRE

## **Lista enlazada**

- Cada bloque libre contiene un puntero al siguiente bloque libre, formando una cadena o lista enlazada.

## **Mapa de bits**

- Utiliza un arreglo de bits donde cada bit representa un bloque de disco; un bit en 1 indica que el bloque está ocupado, y un bit en 0 indica que el bloque está libre.

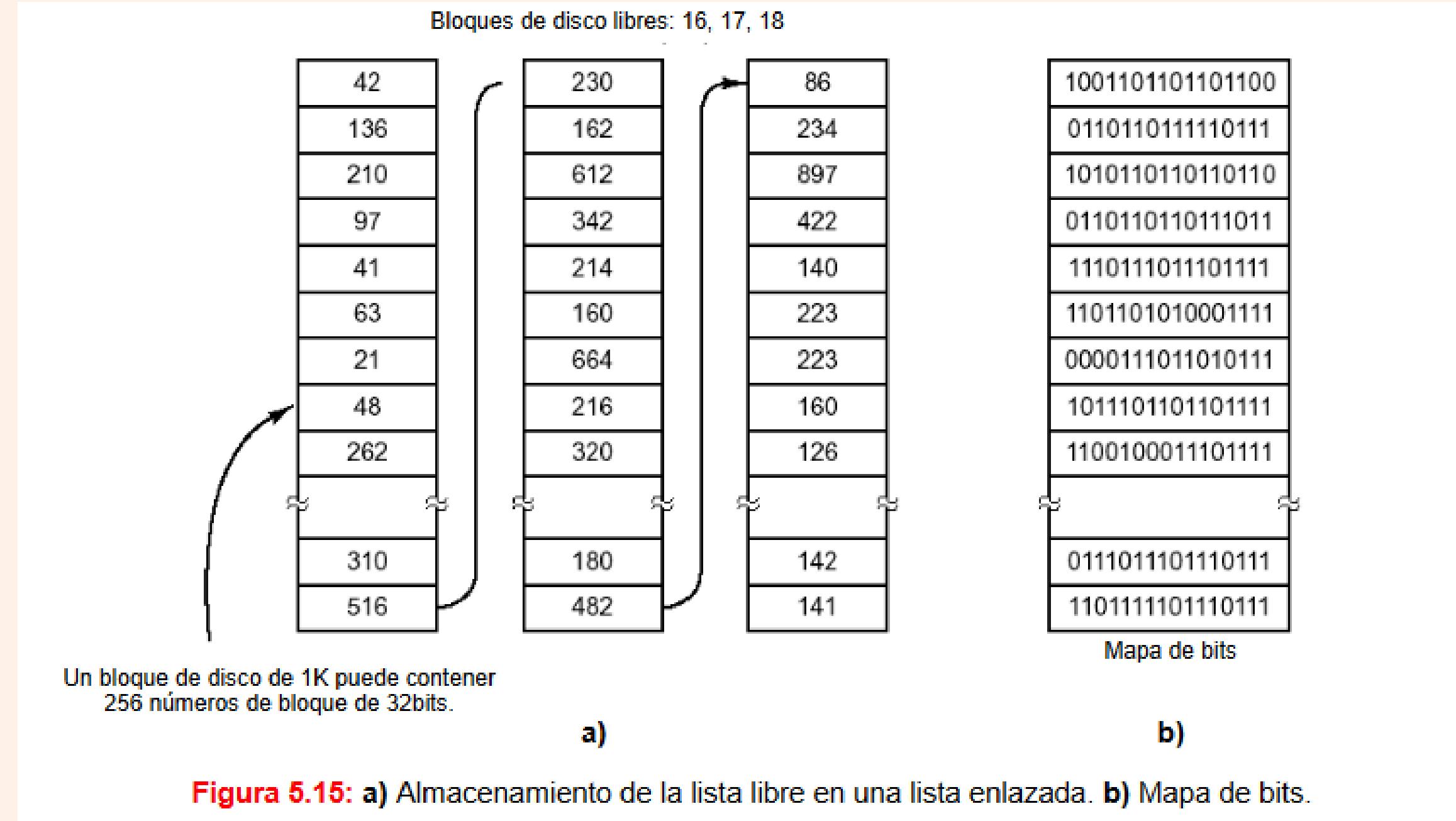
## **Tabla de índices**

- Utiliza una tabla donde cada entrada puede contener información sobre varios bloques libres, incluyendo un rango de bloques o una cantidad de bloques contiguos libres.

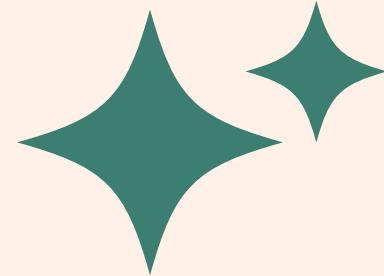
## **Grupo de bloques**

- Divide el disco en varios grupos o bloques grandes y mantiene información sobre bloques libres en cada grupo.

## SISTEMAS OPERATIVOS



# ConFiabilidad del sistema de archivos



La destrucción de un sistema de archivos es un desastre mayor que la destrucción física de una computadora. Si una computadora se destruye, puede reemplazarse fácilmente; pero si el sistema de archivos se pierde, recuperar la información es difícil y a menudo imposible, lo cual puede tener consecuencias catastróficas para empresas y usuarios.

## CONFIABILIDAD DEL SISTEMA DE ARCHIVOS

### Problemas con Bloques Defectuosos

- Discos Flexibles: Generalmente son perfectos al salir de fábrica, pero son susceptibles a defectos con el uso.
- Discos Winchester: Suelen tener bloques defectuosos debido a la complejidad y tamaño. Estos discos incluyen una lista de bloques defectuosos y reservan sectores para estos bloques.

### Manejo de Bloques Defectuosos

- Inicialización del Controlador: Al iniciarse, el controlador en hardware lee la lista de bloques defectuosos y usa bloques de repuesto para reemplazarlos.
- Actualización Continua: La lista se actualiza conforme se descubren nuevos errores, asegurando que los bloques defectuosos se sustituyan por los de repuesto.

# ConFiabilidad del sistema de archivos

## Discos modernos

Aunque las técnicas han mejorado y los bloques defectuosos son menos comunes, siguen existiendo. Los discos modernos incluyen sectores de repuesto y mecanismos para manejar bloques defectuosos sin que el usuario lo note.

## soluciones de software para discos menos modernos

Crear archivos que contengan todos los bloques defectuosos para que estos no se usen en archivos de datos.

## soluciones de software para discos menos modernos

Se debe tener cuidado durante los respaldos para evitar leer estos bloques defectuosos.

# Respaldos

SISTEMAS OPERATIVOS

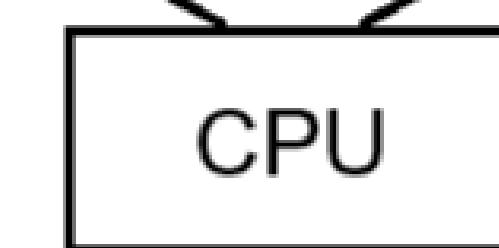
Es crucial realizar respaldos frecuentes de los archivos, incluso con estrategias ingeniosas para manejar bloques defectuosos. Los sistemas de archivos en discos flexibles pueden respaldarse copiando todo el disco en uno en blanco, mientras que los discos Winchester pequeños pueden respaldarse vaciando todo el disco en cinta magnética. Para discos Winchester grandes, respaldar toda la unidad en cinta puede ser tedioso. Una estrategia alternativa implica tener dos unidades de disco en cada computadora, divididas en mitades de datos y respaldo, lo que facilita la recuperación en caso de fallo.

# REPRESENTACIÓN

Disco 0



Disco 1



## Alternativa

Una alternativa al respaldo completo diario es el respaldo incremental, se realiza un respaldo completo periódicamente y luego respaldos diarios solo de los archivos modificados. Se puede emplear la creación de espejos con múltiples discos para permitir el funcionamiento en modo degradado si un disco falla, facilitando la recuperación de datos sin interrumpir el sistema.

# Consistencia del sistema de archivos

La consistencia de archivos es crucial para evitar errores y pérdida de datos.

Para enfrentar el problema de un sistema de archivos inconsistente, la mayor parte de las computadoras cuentan con un programa de utilería que verifica la consistencia del sistema de archivos. Este programa puede ejecutarse cuando el sistema se arranca, sobre todo después de una caída.

# Verificación de bloques

**Se basa en:**

**Construcción de Tablas:** Dos tablas con contadores para cada bloque (inicialmente en 0).

**Proceso:**

- Leer todos los nodos-1 y construir una lista de números de bloque.
- Incrementar contadores en la primera tabla (presente bloque en archivos) y en la segunda tabla (presente bloques en la lista libre).

**Si es consistente:**

Cada bloque tendrá un 1 ya sea en la primera tabla o en la segunda

# Verificación de bloques

Si es consistente:

Cada bloque tendrá un 1 ya sea en la primera tabla o en la segunda

Número de bloque	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Bloques en uso																
	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
Bloques libres																

a)

Bloques Faltantes

bloque 2 no ocurre en ninguna de las dos tablas, reduce capacidad del disco.

Número de bloque	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Bloques en uso																
	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	1
Bloques libres																

b)

Sol. El verificador del sistema de archivos simplemente los agrega a la lista libre.

# Verificación de bloques

## Bloques Duplicados

El número 4, que ocurre dos veces en la lista libre. La solución en este caso también es simple: reconstruir la lista libre.

Número de bloque	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
	0	0	1	0	1	0	0	0	1	1	0	0	0	1	1	1

a)

## Bloques de datos duplicados

Mismo bloque de datos esté presente en dos o más archivos

Número de bloque	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1

b)

Sol. Copiar el contenido a un nuevo bloque y actualizar uno de los archivos.

# Consistencia del sistema de archivos

- Cada nodo-i tiene un modo.
- Los directorios que tienen más de 1000 entradas también son sospechosos
- Encendido el bit SETUID, implican posibles problemas de seguridad

En MS-DOS y algunos otros sistemas, cuando un archivo se elimina lo único que sucede es que se enciende un bit en el directorio o nodo-i para marcar ese archivo como eliminado.

En WINDOWS, los archivos eliminados se colocan en un directorio de reciclado especial, del cual pueden recuperarse posteriormente si es necesario.

# 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 lectura de un bloque de un disco duro puede tardar 50 microsegundos.

# Rendimiento del sistema de archivos

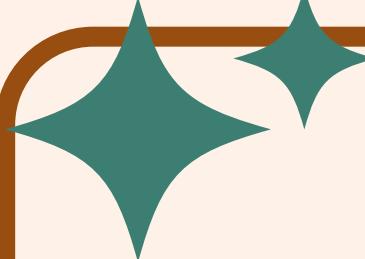
La técnica más común empleada para reducir los accesos a disco es el caché de bloques o el caché de buffer.

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.

# Rendimiento del sistema de archivos

Se pueden usar diversos algoritmos para administrar el caché, uno de los más comunes es inspeccionar todas las solicitudes de lectura para ver si el bloque requerido está en el caché

Si es necesario cargar un bloque en el caché y éste está lleno, es preciso deshacerse de un bloque y escribirlo en el disco si ha sido modificado desde que se trajo del disco.



# Rendimiento del sistema de archivos

Se presentan problemas con las caídas y la consistencia del sistema de archivos. Si un bloque crítico, digamos un bloque de nodo-1, se coloca en el caché y se modifica, pero no se reescriben en el disco, una caída dejaría al sistema de archivos en un estado inconsistente. Si el bloque de nodo-1 se coloca al final de la cadena de LRU, puede pasar un buen rato antes de que llegue al principio y se reescriba en el disco.

# Rendimiento del sistema de archivos

Algunos bloques, como los de doble indirección, casi nunca se hace referencia dos veces dentro de un tiempo corto. Estas consideraciones dan lugar a un esquema LRU modificado, teniendo en cuenta dos factores:

1. ¿Es probable que el bloque se necesite pronto otra vez?
2. ¿Es indispensable el bloque para la consistencia del sistema de archivos?

# Rendimiento del sistema de archivos

los bloques pueden dividirse en categorías, como:

- bloques de nodo-1
- bloques de dirección
- bloques de directorio
- bloques de datos llenos y bloques de datos parcialmente llenos.

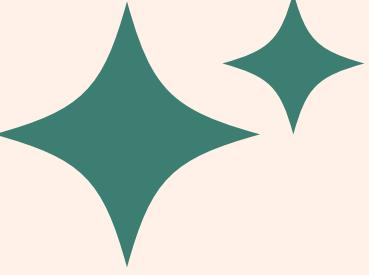
Los bloques que probablemente no se necesitarán pronto se colocan al frente, no al final, de la lista de LRU, con objeto de que sus buffers se reutilicen rápidamente. Los bloques que podrían necesitarse pronto otra vez, como un bloque parcialmente lleno que se está escribiendo, se colocan al final de la lista, para que estén en ella un buen tiempo.

# Rendimiento del sistema de archivos

Si el bloque es indispensable para la consistencia del sistema de archivos (básicamente, todo excepto los bloques de datos) y ha sido modificado, se deberá escribir de inmediato en el disco, sin importar en qué extremo de la lista LRU se coloque.

Incluso con esta medida para mantener intacta la integridad del sistema de archivos, no es deseable mantener los bloques de datos demasiado tiempo en el caché antes de escribirlos en el disco.

- UNIX tiene la llamada al sistema, SYNC, que obliga la escritura inmediata en disco de todos los bloques modificados.
- MS-DOS graba todos los bloques modificados en el disco tan pronto como se modifican



# Rendimiento del sistema de archivos

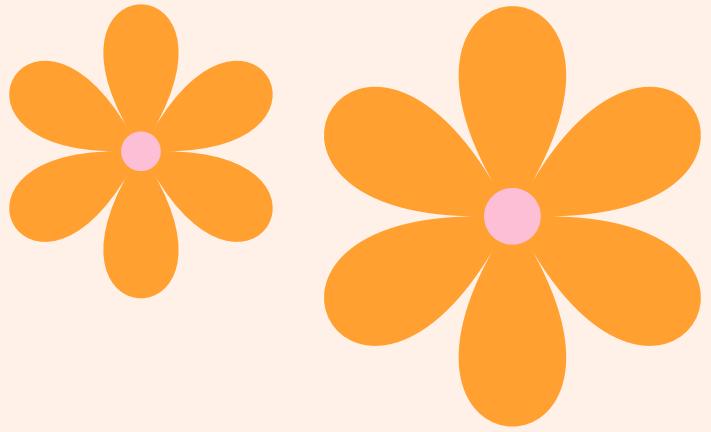
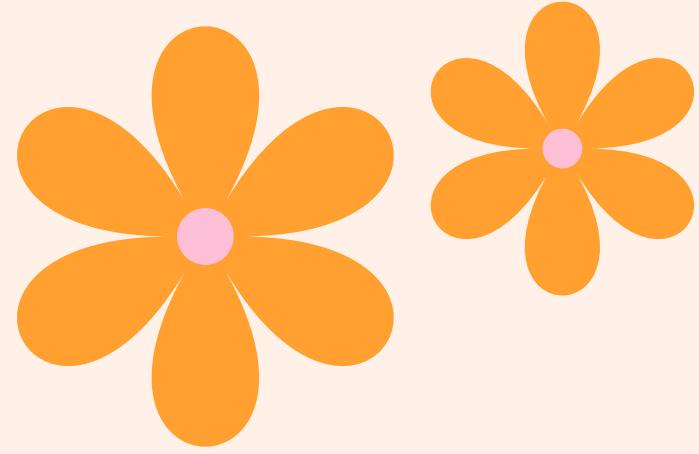
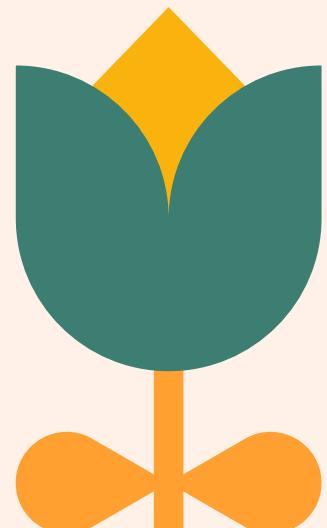
Si retiramos un disco (flexible) de un sistema UNIX sin efectuar SYNC casi siempre perderemos datos, y en algunos casos corromperemos también el sistema de archivos. En MS-DOS no hay problema.

## ADMINISTRACIÓN DEL ESPACIO DE ALMACENAMIENTO EN DISCO

Agrupamiento de bloques consecutivos

Optimización del acceso a nodos-i

Posicionamiento rotacional



# Rendimiento del sistema de archivos

- Una forma fácil de mejorar el rendimiento es colocar los nodos-i en la parte media del disco, reduciendo así a la mitad el movimiento medio del brazo para desplazarse del nodo-i al primer bloque.
- 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.

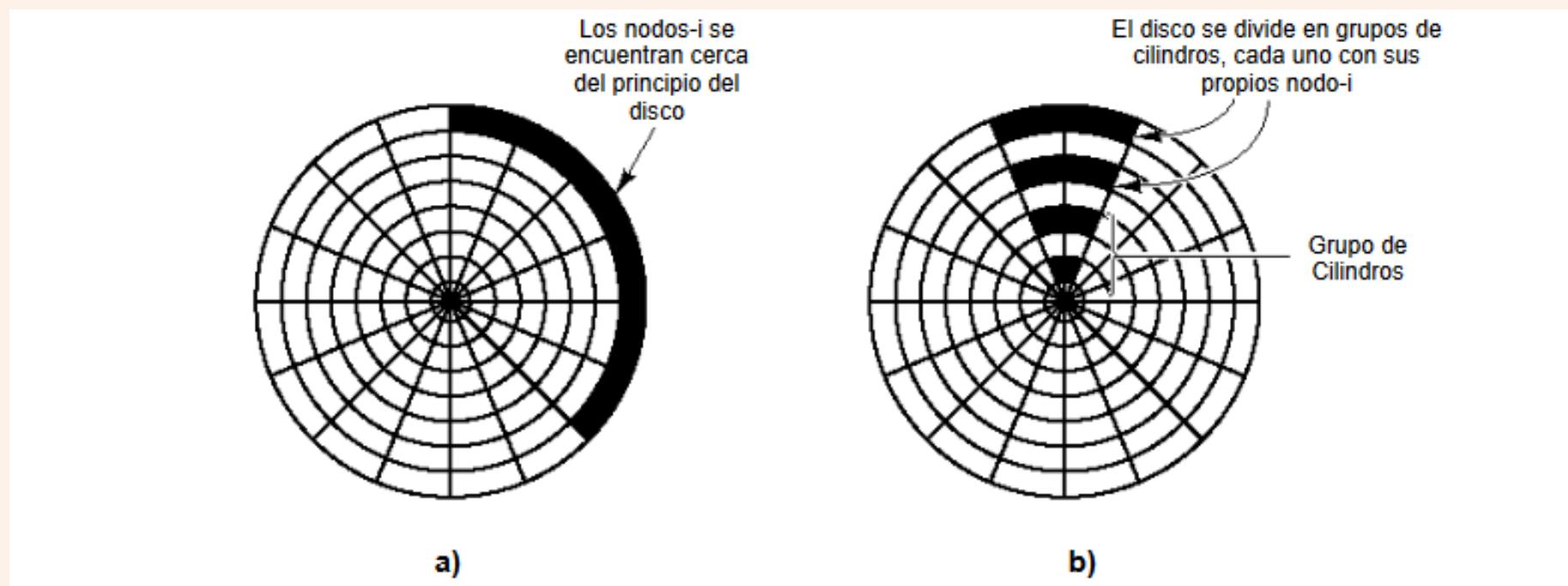
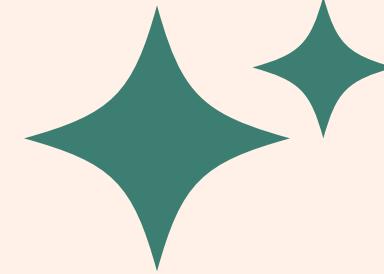


Figura 5.18: a) Nodos-i colocados al principio del disco. b) Disco dividido en grupos de cilindros, cada uno con sus propios bloques y nodos-i.



# Sistemas de Archivos Estructurado s por Diario (LFS)

Los avances tecnológicos, como CPU más rápidas, discos más grandes y memorias crecientes, ejercen presión sobre los sistemas de archivos actuales. Sin embargo, el tiempo de búsqueda en disco no ha mejorado al mismo ritmo, creando un cuello de botella en el rendimiento de los sistemas de archivos.



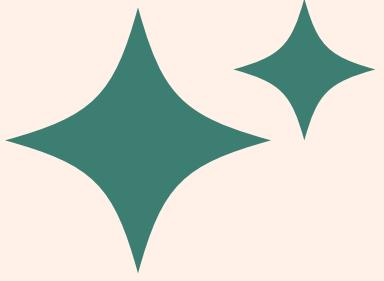
# Gestión de archivos por el sistema operativo

## CONCEPTO

Cuando el usuario deseé referirse a un conjunto de información del mismo tipo como una unidad de almacenamiento, no tiene nada más que crear un archivo dándole el nombre que considere oportuno, dentro de las reglas establecidas por el sistema operativo que se esté utilizando.

## CARACTERÍSTICAS

- Deben ser capaces de contener grandes cantidades de información,
- Su información debe permanecer y sobrevivir a los procesos que la generan o utilizan, y
- Distintos procesos deben poder acceder a la información del archivo concurrentemente.



# Gestión de archivos

Para almacenar información, Anteriormente, para llegar a un sector específico de un disco, era necesario especificar el número de unidad, superficie, pista y sector.

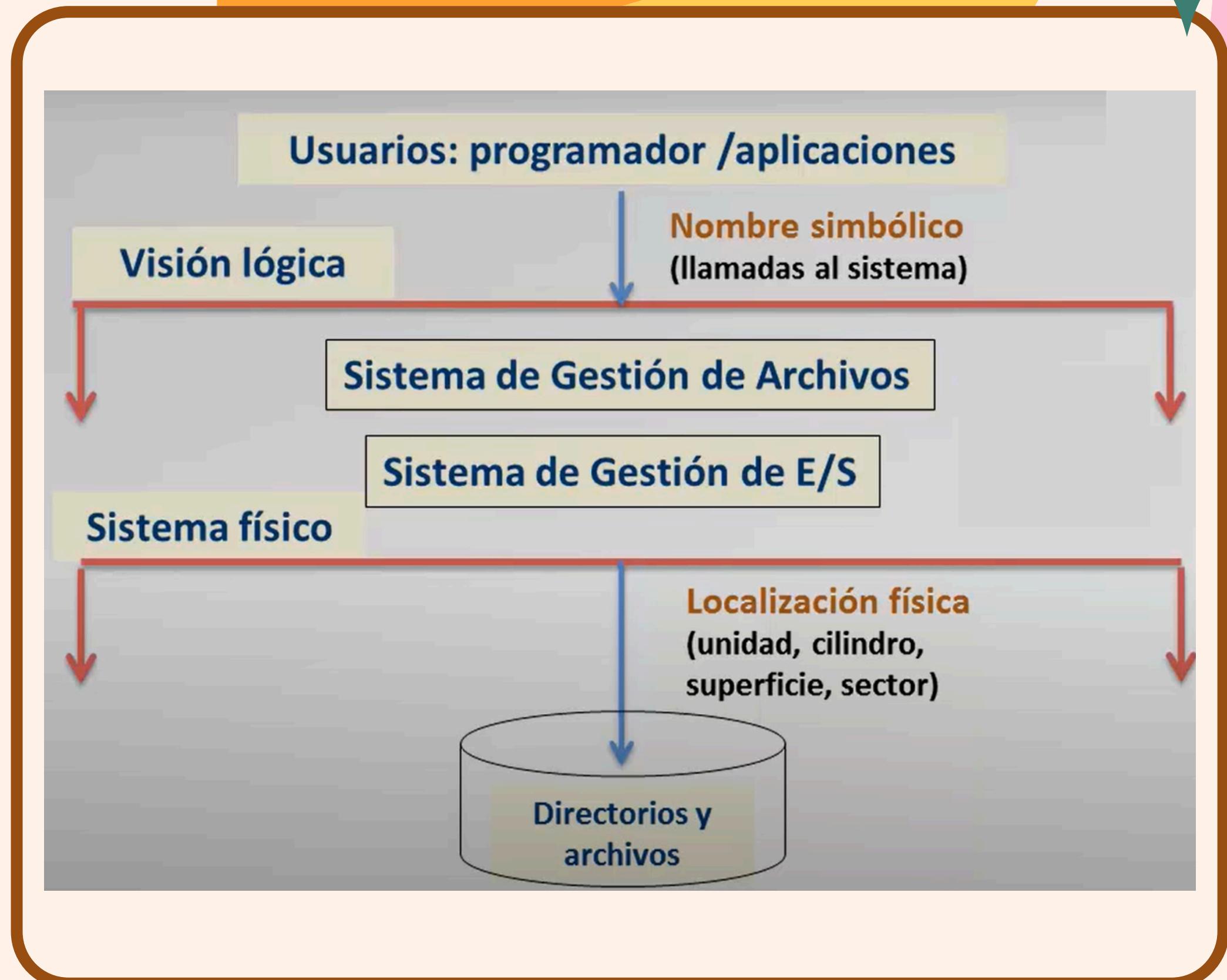
El sistema operativo, con sus conceptos de archivo y directorio, simplifica esto al usuario, evitando que tenga que conocer los detalles técnicos del hardware.

Dos visiones:

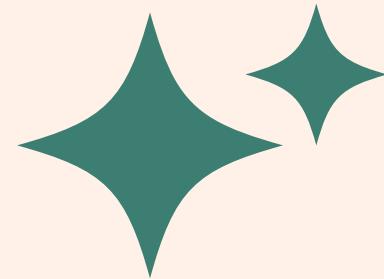
- Lógica: Relacionada con cómo la aplicación utiliza la información.
- Física: Relacionada con cómo la información se guarda en el hardware.

# Gestión de archivos

- **Visión Lógica:** Se centra en cómo los usuarios y las aplicaciones interactúan con los archivos de manera sencilla y abstracta.
- **Sistema Físico:** Se enfoca en cómo y dónde se almacenan realmente los datos en el hardware, manejando la distribución física de la información.



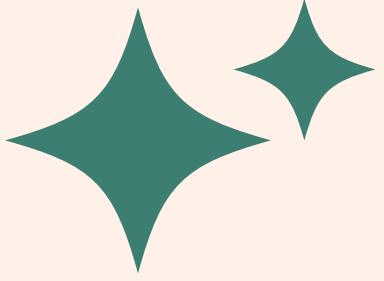
# Concepto de archivo



Cada archivo incluye:

- Atributos o metadatos: nombre, fecha y hora de creación y actualización, bits de protección, propietario, contraseña de acceso, número de bytes por registro, capacidad máxima, capacidad ocupada y ubicación.
- Datos propiamente dichos.

Los archivos se almacenan en el dispositivo de memoria masiva en forma de "contenedores" o bloques (clústeres).



# Concepto de directorio o carpeta

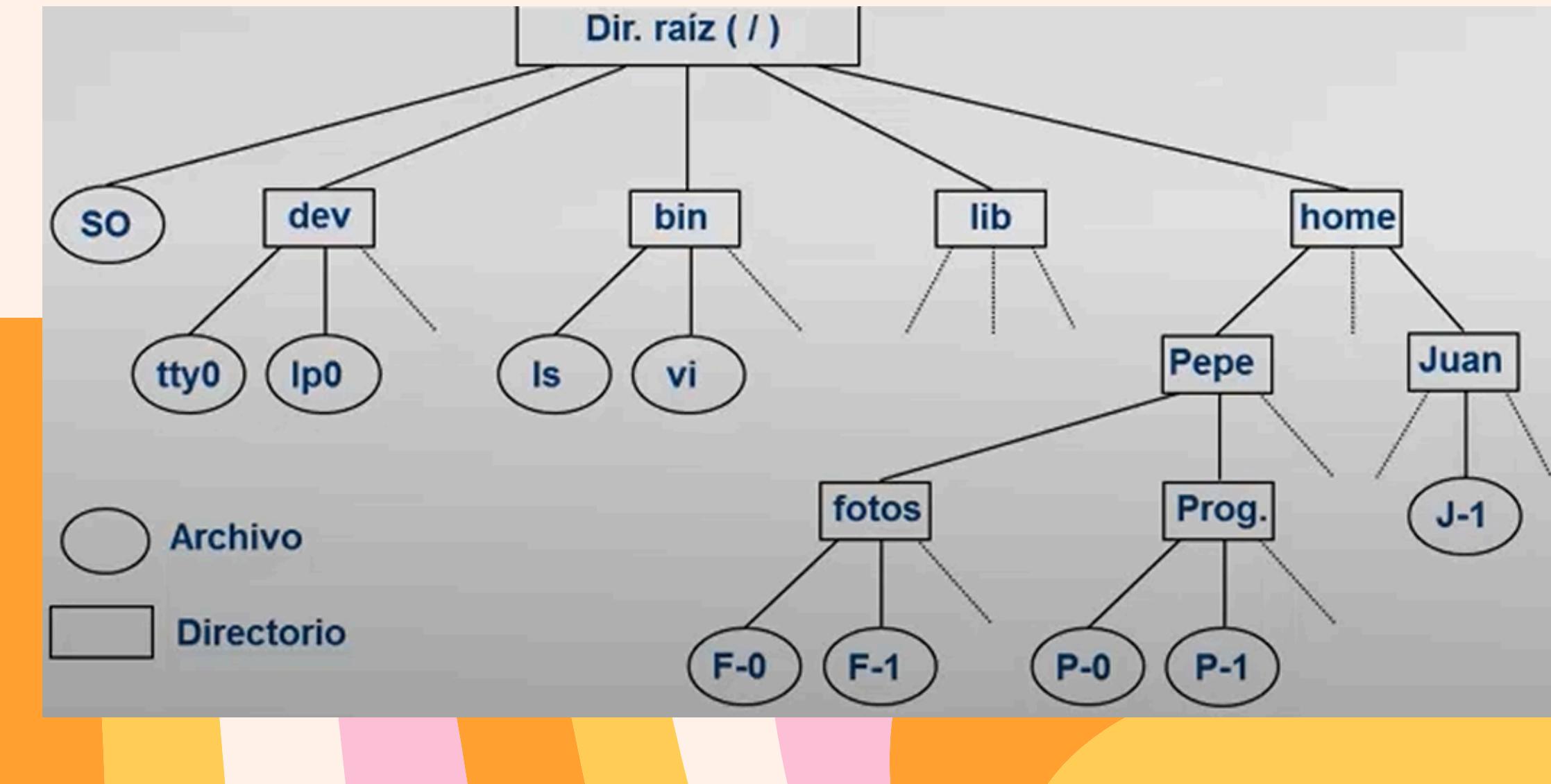
## Concepto de Directorio o Carpeta

- Permite al usuario organizar sus archivos.
- Está constituido por un grupo de archivos y/o otros directorios (subdirectorios).
- Un directorio en realidad es un archivo más, constituido por una tabla con una fila por cada archivo/subdirectorio integrado en el directorio, con metadatos sobre el mismo:
  - Nombre simbólico, propietario, grupo, enlaces a posiciones físicas, número de bloques que ocupa.

**Los directorios o carpetas tienen una estructura arbórea, incluyendo en sus nodos subdirectorios y en sus hojas archivos**

### Estructura Arbórea de Directorios

Los directorios o carpetas tienen una estructura arbórea, con subdirectorios en sus nodos y archivos en sus hojas.



### Ruta (Path) de un Archivo

- Lista de todos los directorios atravesados hasta llegar al archivo/directorio.
  - Path absoluto: desde el directorio raíz (/).
  - Path relativo: desde el directorio actual

# Registros lógicos y físicos

Visión lógica:

- Archivo: sucesión de registros lógicos.
- Registro lógico: estructurado en campos.
- Campo: conjunto de caracteres.

Sistema físico:

- Archivo: sucesión de registros físicos (disco: bloque o clúster).
- Bloque: conjunto de bytes que se transfieren en una operación de E/S.



# Tamaño de los registros Fisicos (bloques)

## Bloque pequeño (4Kb)

Poca eficiencia (mas interrupciones, tablas de enlaces mayores)

## Bloque grande (64Kb)

Fragmentación interna:

- A no ser que la capacidad del archivo sea un múltiplo del tamaño del bloque, se desaprovecha el último "contenedor", y cuanto mayor sea el tamaño del bloque, mayor sera el desperdicio.
- Los ficheros muy pequeños utilizaran mas de lo necesario

## EL SO y el dispositivo fisico:

Definen el tamaño de los bloques

## Factor de bloqueje

nº registros logicos por registro fisico

# Tabla descriptora de archivos

**Cuando se requiere el uso de un archivo se utiliza OPEN o CREATE**

Con ella se "abre" el archivo.

Pasa al SO el nombre simbólico del archivo

El sistema de gestión de archivos, lleva a MP toda la información referente a dicho archivo(metadatos) y asigna al archivo un nº(descriptor de archivo).

**El SO mantiene una tabla descriptora de archivos**

Contiene una lista de todos los archivos abiertos en un momento dado.

Cada vez que se abre un archivo se añade una fila con el nombre simbólico y el descriptor del nuevo archivo y sus atributos (metadatos), tamaño enlace al primer bloque, etc.

Cuando se cierra el archivo se elimina la información que éste en la tabla

# Funciones del SO para gestión de archivos

Ademas de la función "open" hay otras del SO que permiten utilizar el archivo.

Por ejemplo si una aplicación va a escribir unos datos en el fichero utiliza la función (llamada al sistema) "write".

`write(descrip_archivo,pzona,numB)`

Parámetros: "descrip\_archivo":descriptor del archivo; "pzona": puntero a la zona de memoria (buffer) donde se encuentran los datos a escribir en el archivo y "numB": n<sup>a</sup> de bytes a escribir.

El sistema de gestión de archivos lanza a ejecutar el programa que implementa la función "write" y que contiene una serie de ordenes y parametros para el controlador DMA, considerando el n<sup>a</sup> de bloque y del puntero.

Funciones	UNIX	Windows
Crear	open	Createfile
Borrar	unlink	Deletefile
Cerrar	close	CloseHandle
Leer	read	Readfile
Escribir	write	Writefile
Mover puntero	lseek	SetfilePointer
Información	stat	GetfileAtributes

# Asignación de espacio en disco

## Sistema de archivo en discos

Es la estructura u organización del almacenamiento de archivos de forma que estos pueden ser leídos o almacenados directamente por el computador.

## Pueden ser:

FAT, NTFS, HFS, HFS+, ext2, ext3, ISO 9660, ODS-5 y UDF

## Cluster

Es un grupo de sectores contiguos.

## Se pueden almacenar como

- Lista de enlaces
- Fichero de índices o í-nodos
- Árboles B+

# Lista de enlaces

- Es una estructura de datos utilizada para organizar y gestionar los archivos y directorios en un sistema de archivos. Cada entrada en la lista de enlaces apunta a un archivo o directorio específico.

# Tipos de enlaces

- Enlaces duros (Hard Links): Son referencias directas al inode de un archivo.
- Enlaces simbólicos (Soft Links o Symlinks): Son archivos que contienen una referencia a otro archivo o directorio en forma de ruta.

# Fichero de índices o i-nodos en UNIX

El sistema de archivos UFS (Unix File System) utiliza una estructura de datos llamada i-nodo para gestionar los archivos.

Cada archivo o directorio tiene un i-nodo único, y estos i-nodos están indexados por número. El sistema de archivos utiliza estos números de i-nodo para rastrear los archivos.

# Árboles B+

Son una estructura de datos utilizada en bases de datos y sistemas de archivos para organizar y acceder eficientemente a grandes cantidades de datos.

**Todos los datos están en las hojas**

A diferencia del árbol B, donde los datos pueden estar en nodos internos, en un árbol B+ todos los datos se almacenan en los nodos hoja.

**Nodos internos contienen solo claves**

Los nodos internos del árbol B+ contienen solo las claves que sirven como índices para facilitar la búsqueda,

**Enlaces secuenciales**

Las hojas del árbol B+ están enlazadas secuencialmente mediante punteros, lo que permite un acceso eficiente a los datos en orden secuencial.

# Sistema de archivos NTFS

Es el sistema de archivos estándar utilizado en las versiones modernas de Windows, comenzando con Windows NT y utilizado en todas las versiones posteriores desde Windows 2000.

## Seguridad y permisos

NTFS permite configurar permisos detallados para archivos y carpetas, lo que proporciona un control granular sobre quién puede acceder y modificar los archivos.

## Compatibilidad con archivos grandes

A diferencia de sistemas de archivos más antiguos como FAT32, NTFS puede manejar archivos de tamaño muy grande, así como volúmenes de disco grandes.

# Sistema de archivos NTFS

## Compresión y cifrado

NTFS soporta la compresión de archivos y carpetas para ahorrar espacio en disco y también permite el cifrado a nivel de archivo para mejorar la seguridad de los datos.

## Registros y recuperación de fallos

NTFS utiliza un sistema de registro que permite la recuperación automática de errores del sistema de archivos, minimizando la pérdida de datos en caso de fallos del sistema.

## Administración de espacio eficiente

Utiliza una estructura basada en árboles B+ para gestionar los archivos y directorios, lo que mejora la eficiencia en la búsqueda y acceso a los datos.

# Conclusión

En resumen, la implementación de sistemas de archivos es esencial para el funcionamiento eficiente y confiable de cualquier sistema operativo. Este proceso abarca la gestión del almacenamiento de archivos y directorios, la administración del espacio en disco y la optimización de la estructura de archivos. Existen varios métodos de asignación de espacio, como la asignación contigua, la asignación por lista enlazada y la asignación mediante nodos-1, cada uno con sus propias ventajas y desventajas en términos de rendimiento y eficiencia. La fiabilidad y consistencia del sistema de archivos también son aspectos críticos, abordados mediante técnicas de manejo de bloques defectuosos, estrategias de respaldo y verificación de la consistencia del sistema de archivos. Además, la optimización del rendimiento mediante el uso de cachés de bloques y la administración eficiente del espacio de almacenamiento contribuyen a mejorar la velocidad y la confiabilidad del acceso a los datos.

**Gracias por  
su atención**