

A large, semi-transparent circular graphic on the left side of the slide features a complex network of interconnected nodes and edges. The nodes are small circles of varying sizes, colored in a gradient from white to dark purple. The edges are thin, light-colored lines connecting the nodes. The entire graphic is set against a dark, solid background.

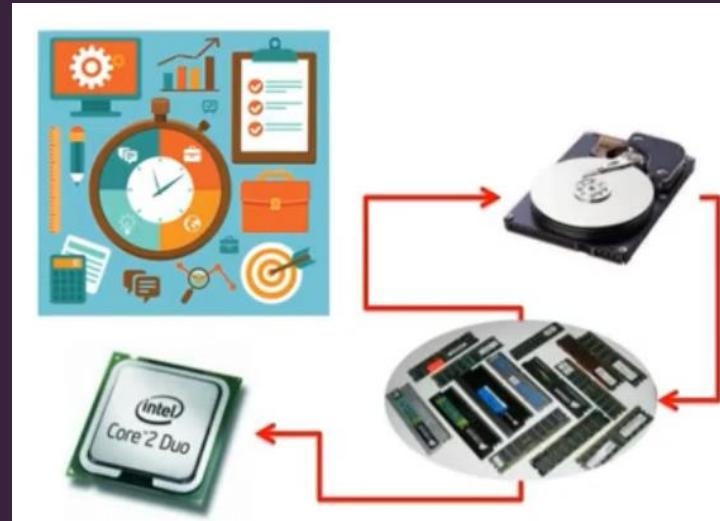
Sistemas operativos

Gonzalo Carreño

gonzalocarrenob@gmail.com

Gestión de memoria

- Se refiere a los distintos métodos y operaciones que se encargan de obtener la máxima utilidad de la memoria



Organizando los procesos y programas que se ejecutan de manera tal, que se aproveche de la mejor manera posible la memoria disponible

Gestión de memoria

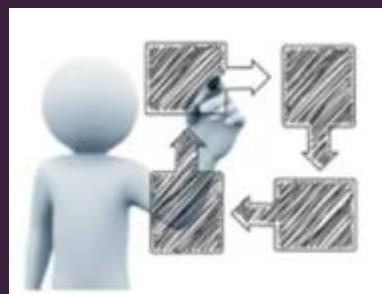
- La parte del sistema operativo que administra la memoria se le conoce como administrador de memoria



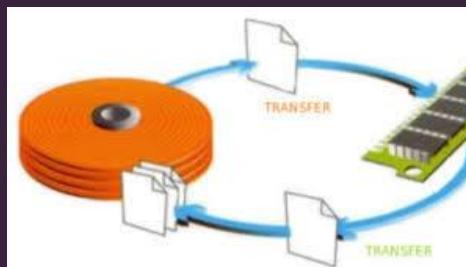
Administrador de memoria

- Funciones:

- Llevar un registro de la memoria libre y de las zonas que se están usando.



Liberar espacio en memoria
de los procesos terminados



Reservar espacio en memoria para
nuevos procesos



Gestionar el intercambio de datos
entre memoria y disco

Memoria secundaria

- Es el conjunto de dispositivos y soportes de almacenamiento que datos que conforman el subsistema de memoria del computador, junto con la memoria primaria principal.



Sistemas de archivo

- Componente del sistema operativo encargado de administrar y facilitar el uso de los dispositivos de almacenamiento.
- Los usuarios deben poder crear, modificar y borrar archivos.



Sistemas de archivos

- Estructuran la información guardada en un dispositivo de almacenamiento de datos o unidad de almacenamiento.
 - Normalmente un disco duro
 - Puede ser una memoria flash (pendrive)
 - Memorias micro SD
 - Unidades de estado sólido



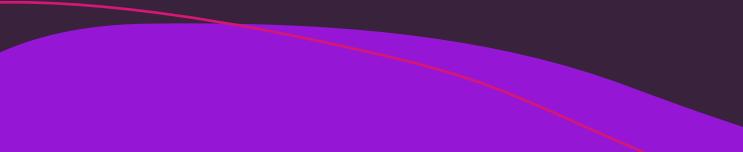
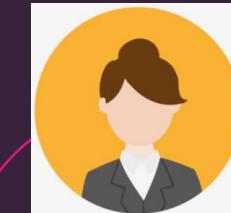
Sistema de archivos

- Sus principales funciones son:
 - Asignación de espacio a los archivos
 - Administración del espacio libre
 - Acceso a datos resguardados



Sistema de archivos

- Sus principales funciones son:
 - Poder compartir los archivos de una manera cuidadosamente controlada
 - Controla accesos
 - Acceso de lectura, Acceso de escritura, acceso de ejecución, varias combinaciones de estos, etc.



Sistema de archivos: FAT

- Fat = File allocation table
- Desarrollado para MS-DOS.
- Popular para diskettes.
- Esquema de asignación de nombres 8.3
 - Nombarch.ext

Relativamente sencillo



Sistema de archivos: FAT

- FAT12
 - Las direcciones de bloque solamente contiene 12 bits.
- FAT16
 - Utilizado en MS-DOS y en las primeras versiones de Windows.
 - Particiones con una capacidad máxima de 2GB.

Sistema de archivos: FAT32

- Evolución de FAT16
- El tamaño máximo de un archivo con FAT32 es de 4GB.
- Fue lanzado con el segundo gran lanzamiento de Windows 95
- Soporta volúmenes de hasta 32GB.
- Admite nombre de archivos largos.
- Utilizado en tarjetas de memoria y dispositivos similares.



Sistema de archivos: exFAT

- exFAT = Extended File Allocation Table
- Patentado y propiedad de Microsoft
- Especialmente adaptado para memorias flash
- Presentado con Windows CE (Windows Embedded CE 6.0).
- Se utiliza cuando el NTFS no es factible debido a la sobrecarga de las estructuras de datos.
- Límite teórico para el tamaño de archivo de 2^{64} bytes
 - 16 hexabytes

Sistema de archivos: NTFS

- NTFS = New Technology File System
- Windows 2000, XP, Server 2003, Server 2008, Vista, 7, 8, 10.
- El tamaño mínimo recomendado para las particiones es de 10GB
- Volumen máximo de datos: 256 TB
 - El tamaño límite de una partición es de 2^{64} bytes = 16 Hexabytes.

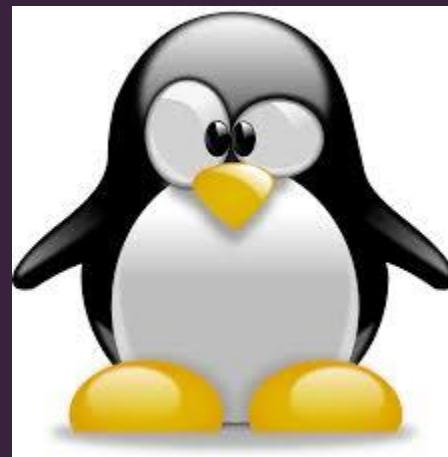


Sistema de archivos: NTFS

- Mucho más rápido en el acceso a los archivos que FAT.
 - Utiliza un árbol binario de alto rendimiento para localizar los archivos.
 - Mas estable que FAT.
 - Mas tolerante a fallos
- Mayor seguridad
- Controla que usuarios y grupos puede tener acceso a archivos y carpetas en un volumen NTFS

Sistema de archivos: EXT3

- EXT3=Third Extended Filesystem
- Principalmente utilizado en distribuciones Linux.
- Principales objetivos: disponibilidad y confiabilidad.



Sistema de archivos: EXT3

- Implementa Journaling
 - Registro diario para restablecer datos del sistema de archivos en caso de falla.
 - Si apagamos nuestro equipo incorrectamente se encargara de que al encender lo tengamos igual que como lo dejamos antes de apagarlo.

Esta siendo reemplazado por su sucesor, ext4, aunque todavía es usado.

Sistema de archivos: EXT4

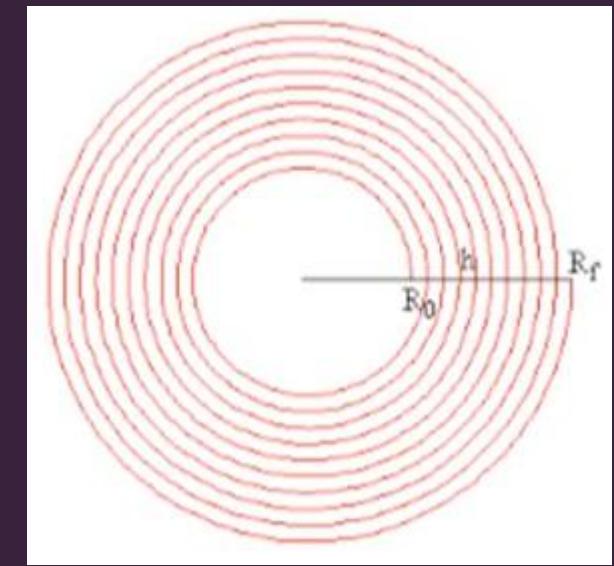
- EXT4=Fourth Extended Filesystem
- Mejora compatible de ext3
 - Utiliza menos CPU
 - Mejora la velocidad de lectura y escritura.
- Soporta volúmenes de hasta 1024 PiB (PebiByte) ($1 \text{ PiB} = 2^{50}$ bytes)



Sistema de archivos: EXT4

- Es mas lento en la eliminación de archivos.
- Se introducen los exents
 - Conjunto de bloques físicos contiguos
 - Reemplazan al tradicional esquema de bloques utilizado por ext2 y ext3.
 - Mejoran el rendimiento al trabajar con archivos grandes.
 - Reducen la fragmentación.

Sistemas de archivos/Discos ópticos



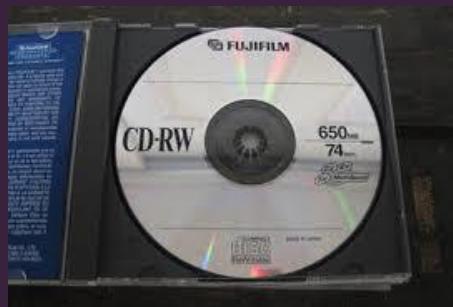
Sistema de archivos/Discos ópticos

- ISO 9660
 - Define un sistema de archivos para CD-ROM.
 - Su propósito es que tales medios sean legibles por diferentes sistemas operativos, de diferentes proveedores y en diferentes plataformas.



Sistema de archivos/Discos ópticos

- UDF (Universal Disc Format)
 - Permite leer, escribir o modificar los archivos contenidos en discos CD/DVD reescribibles (RW)
 - Utiliza la tecnología de grabación por paquetes (Packet Writing) soportado por grabadoras CD-RW, DVD-RAM/RW, HD DVD y BLU-ray.

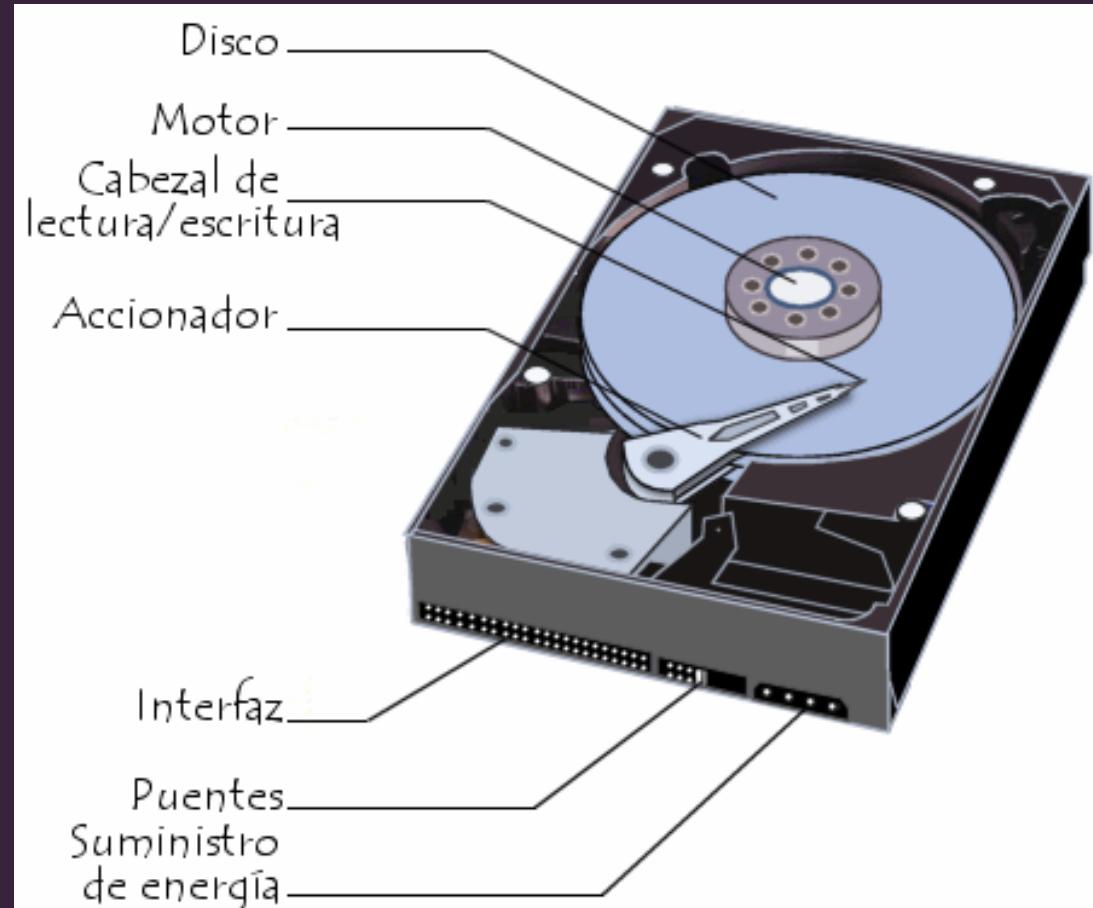


Sistema de archivos/Discos ópticos

- Mount Rainer
 - Añade la posibilidad de utilizar escritura de paquetes al UDF (Universal Disk Format).
 - Su propósito es el de sustituir al disco flexible.
 - Recibe el nombre en honor a una montaña cercana a Seattle, Washington, Estados Unidos.



Disco duro (HDD)



Disco duro (HDD)

- Sirve para almacenar de forma permanente tus datos.
- Están compuestos de piezas mecánicas.
- Utilizan el magnetismo para grabar tus datos y archivos.
- Se compone de uno o varios discos rígidos unidos por un mismo eje.
 - Estos giran a gran velocidad dentro de una caja metálica.

Disco duro (HDD)

- Cada plato y en cada una de sus caras, una cabeza de lectura/escritura lee o graba tus datos sobre los discos.



Cabezas
8 cabezas,
4 platos

Disco duro (HDD)

- Sus tamaños pueden ser de 1,8" 2,5" o de 3,5".
- La gran ventaja de estos discos con respecto a los SSD es que **son bastantes mas económicos.**



Unidades de estado sólido (SSD)

- Son una alternativa a los discos duros.
- Almacenan los archivos en microchips con memorias flash interconectadas entre sí.
 - Podríamos considerarlos como una evolución de las memorias USB.

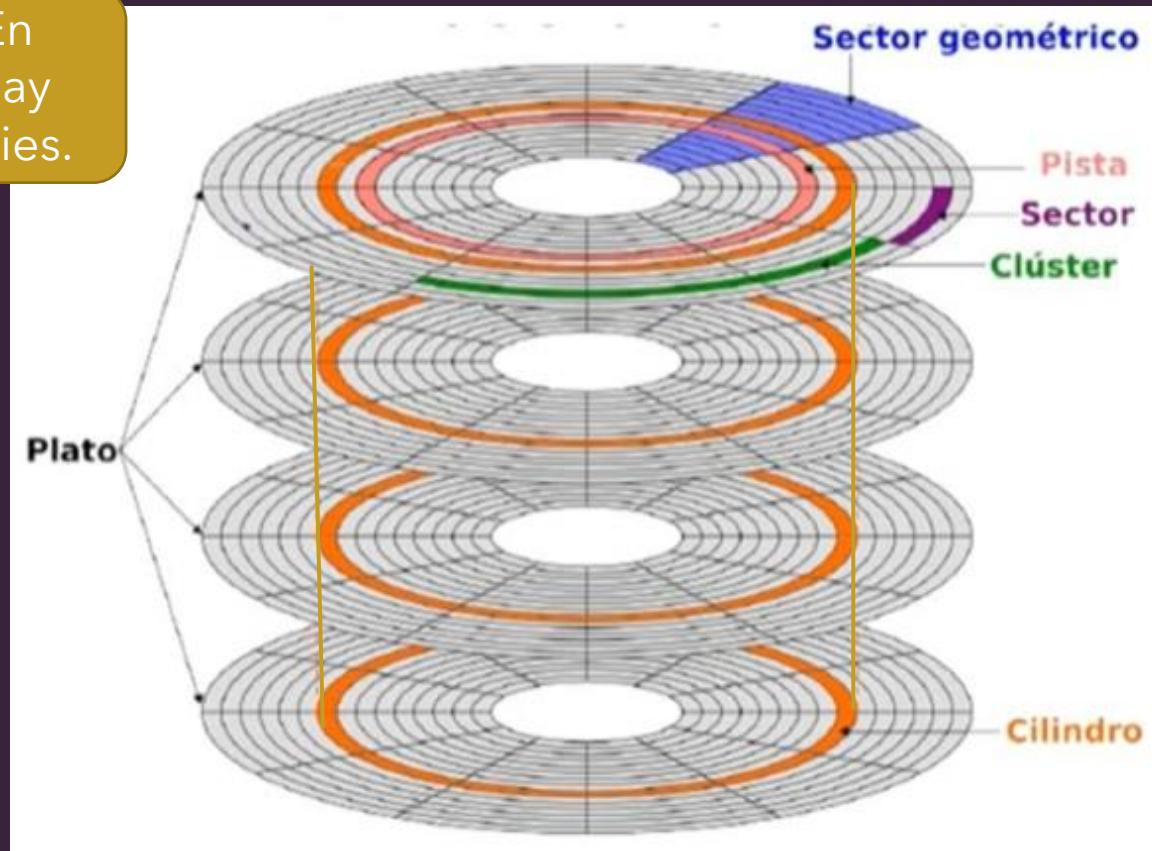


Unidades de estado sólido (SSD)

- Incluyen un procesador integrado para realizar operaciones relacionadas con la lectura y escritura de datos.
- Son los que toman las decisiones sobre como almacenar, recuperar, almacenar en cache y limpiar los datos del disco.
- Al no depender del giro de un componente físico, también se logra una unidad mas silenciosa que los discos mecánicos.
- Suelen ser de 2,5"
- Diseño casi idéntico al de los discos duros mecánicos.
 - Compatibles con las mismas carcasa y ranuras donde van montados los discos duros convencionales.

Sectores físicos, cilindros y superficies

Superficie: En cada plato hay dos superficies.



Sector físico: sección de la superficie del mismo que corresponde al área encerrada entre dos líneas radiales de una pista

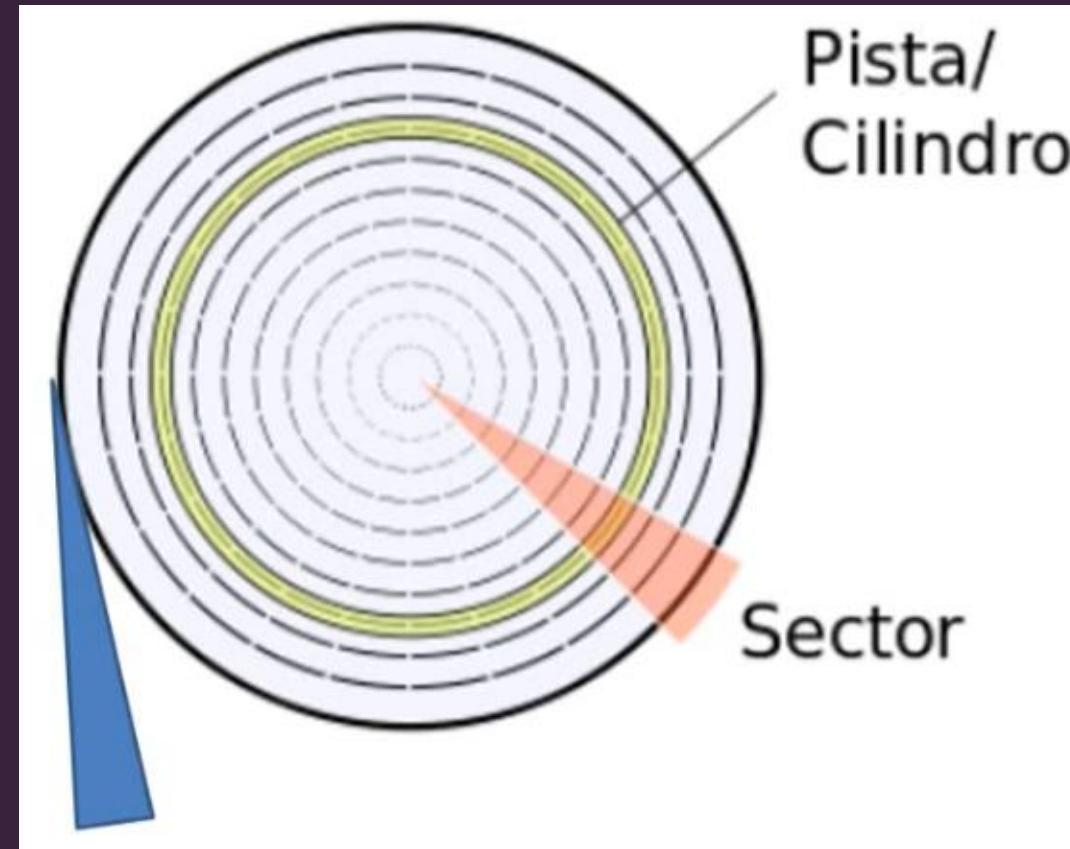
Cilindro: son todas las pistas de todas las superficies con las misma numeración

Sectores físicos, cilindros y superficies

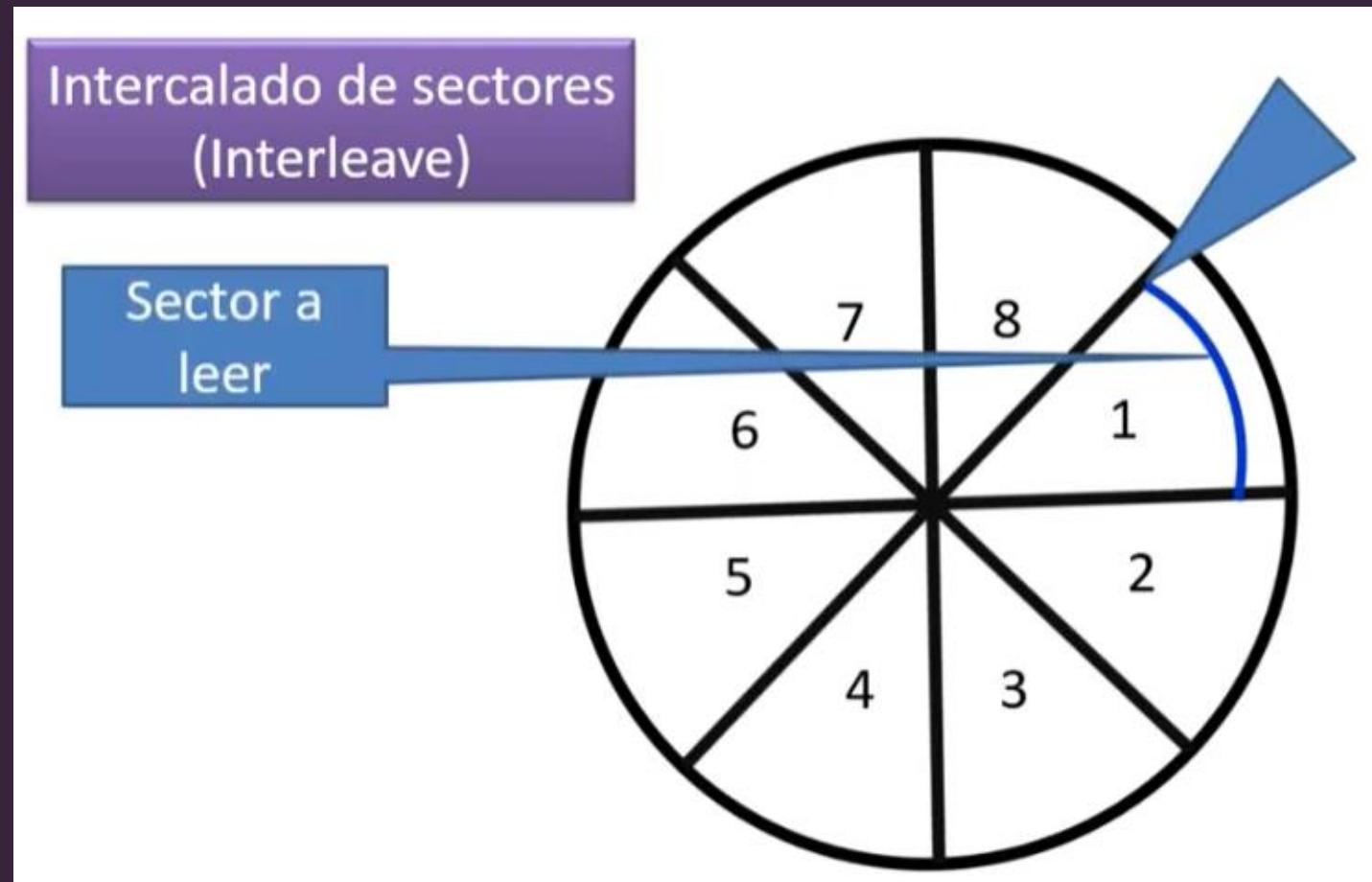
Tiempo de búsqueda:
cantidad de tiempo
requerida por la
cabeza para
posicionar su brazo
en la pista

Tiempo=Distancia/Velocidad

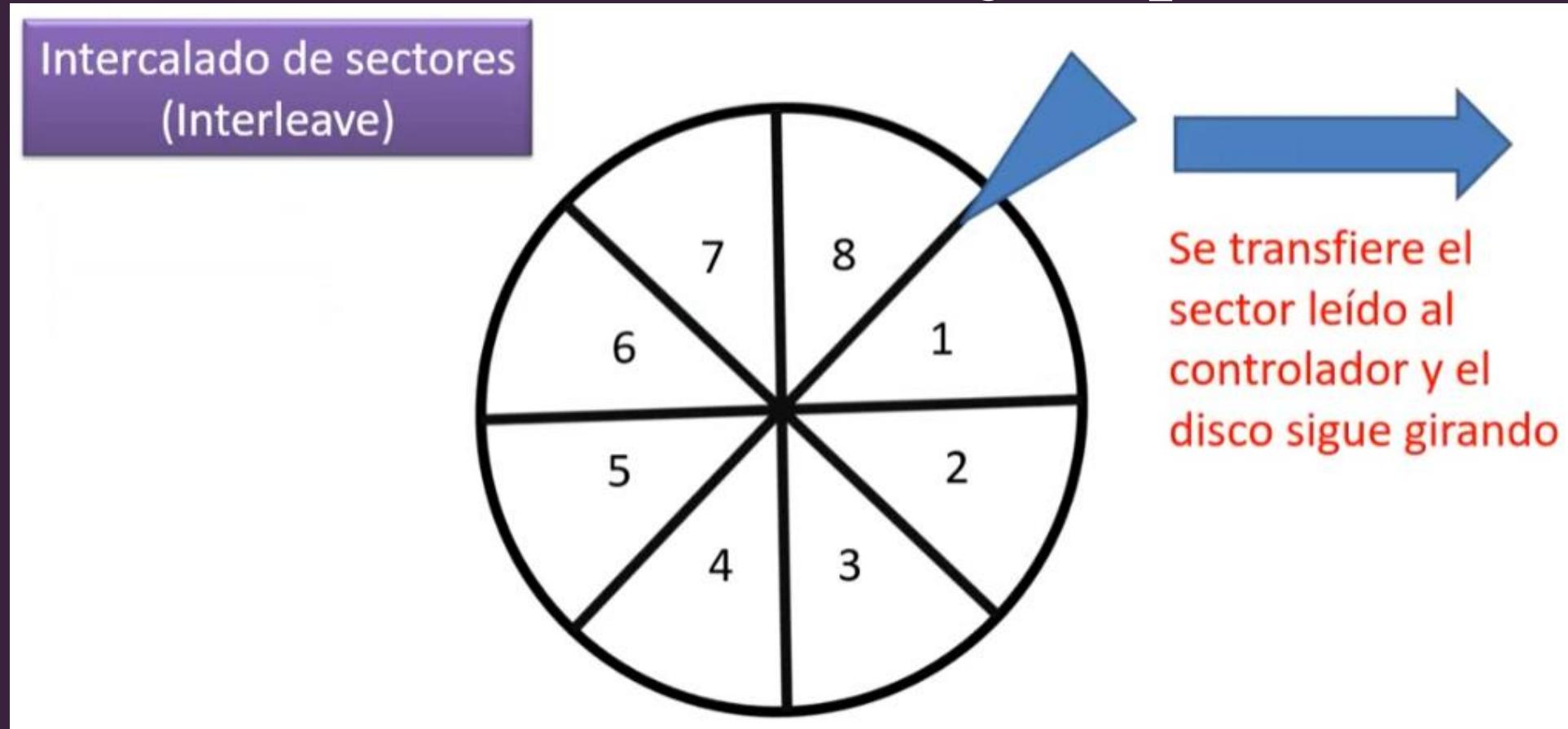
Fragmentación



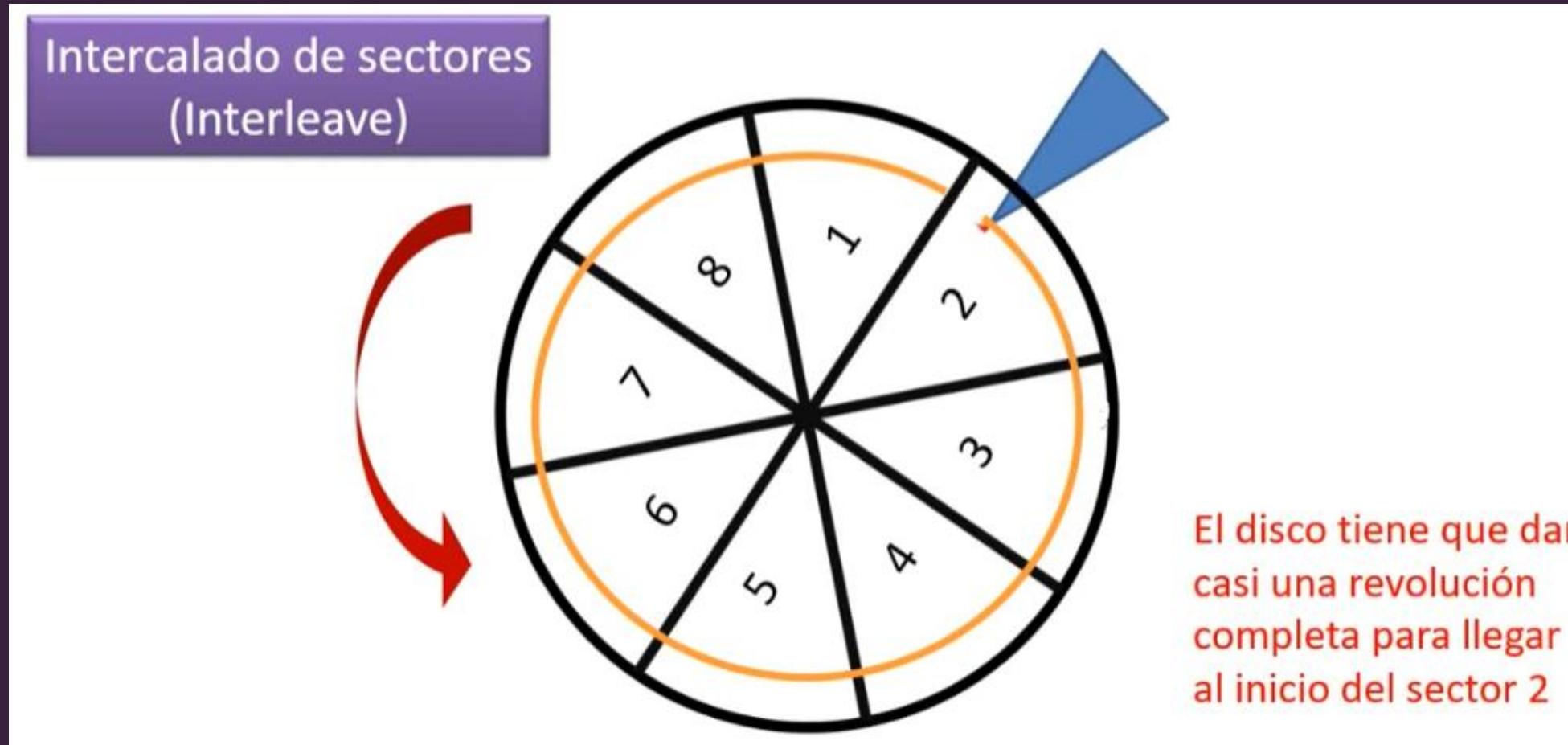
Sectores físicos, cilindros y superficies



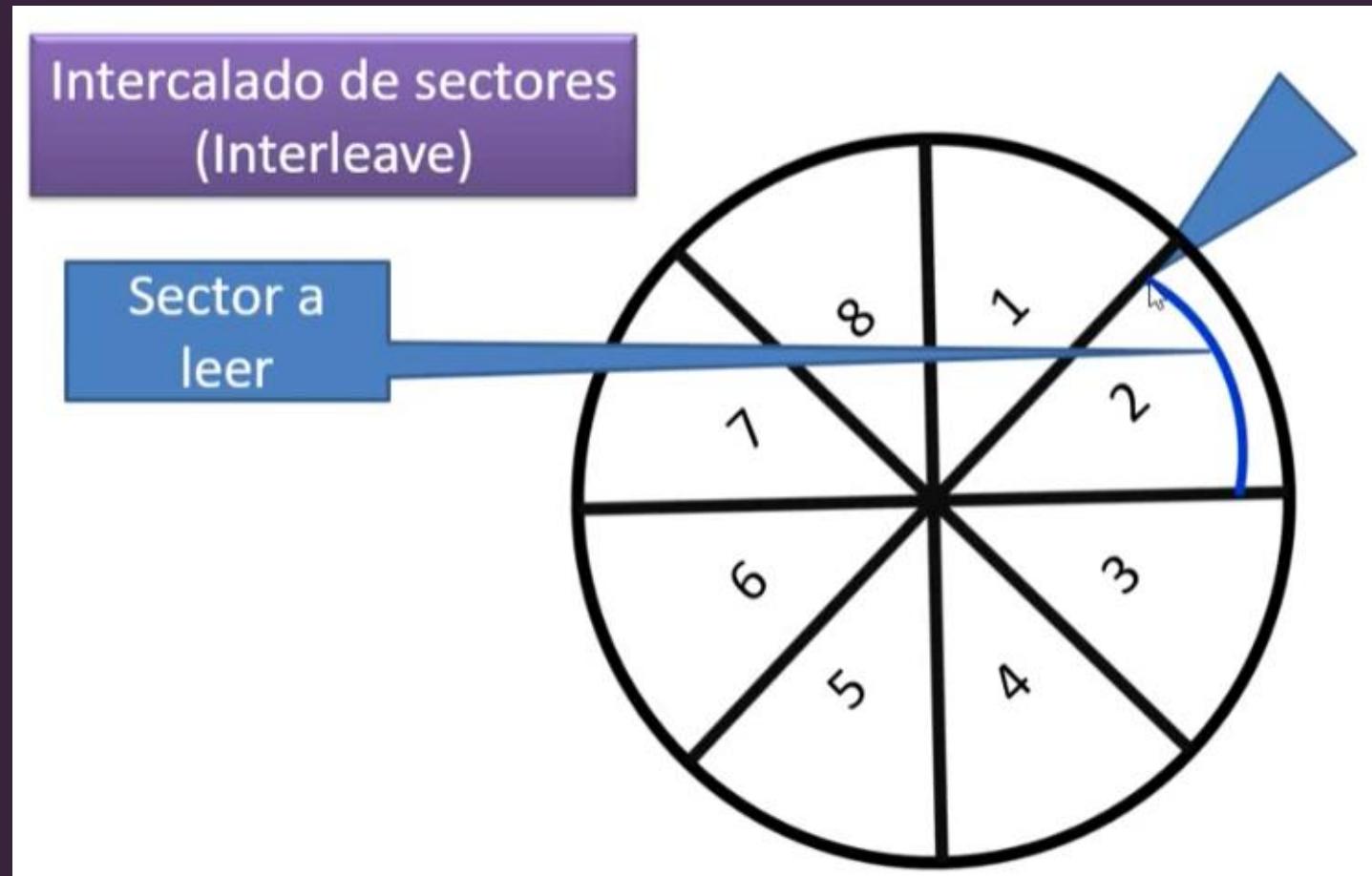
Sectores físicos, cilindros y superficies



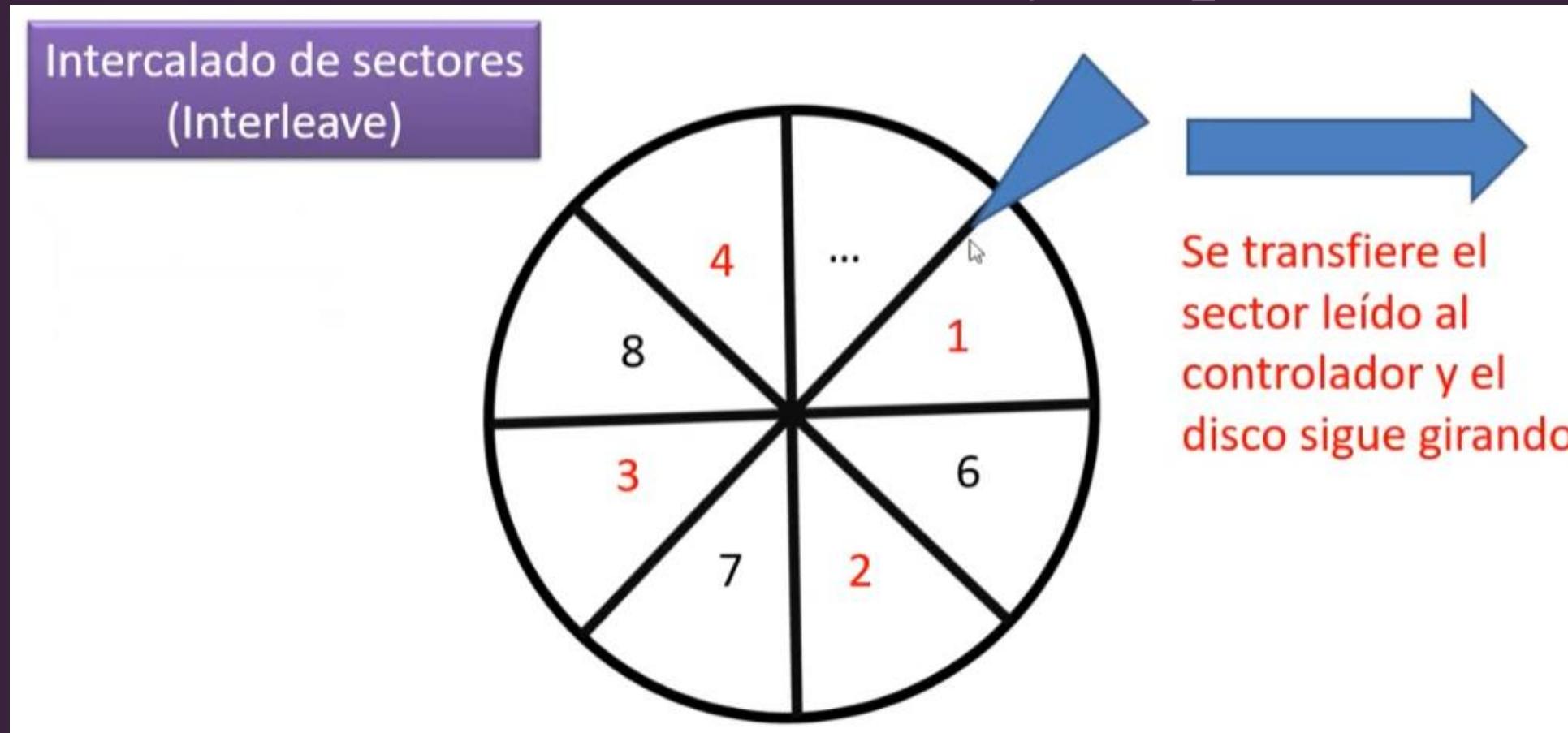
Sectores físicos, cilindros y superficies



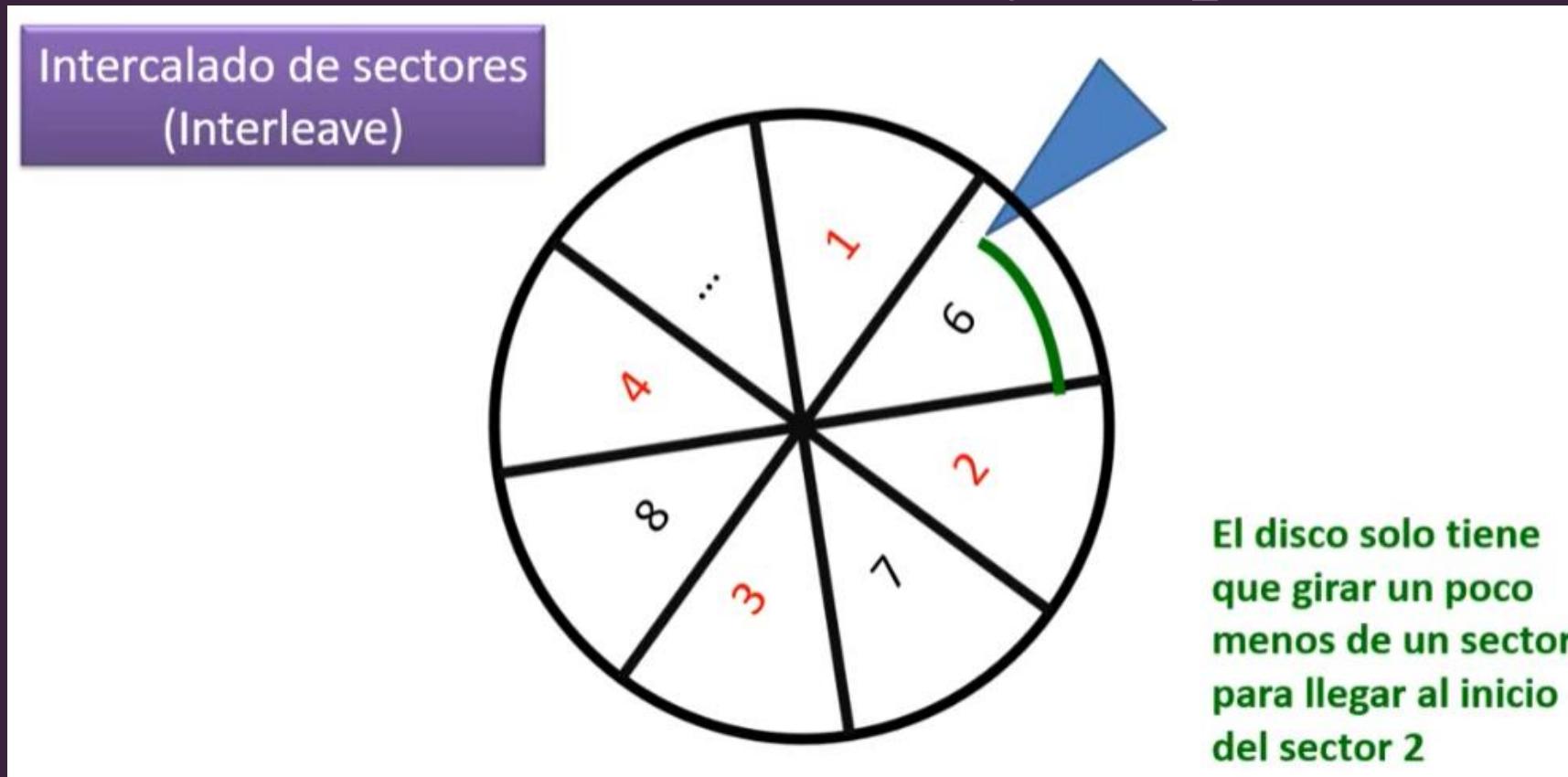
Sectores físicos, cilindros y superficies



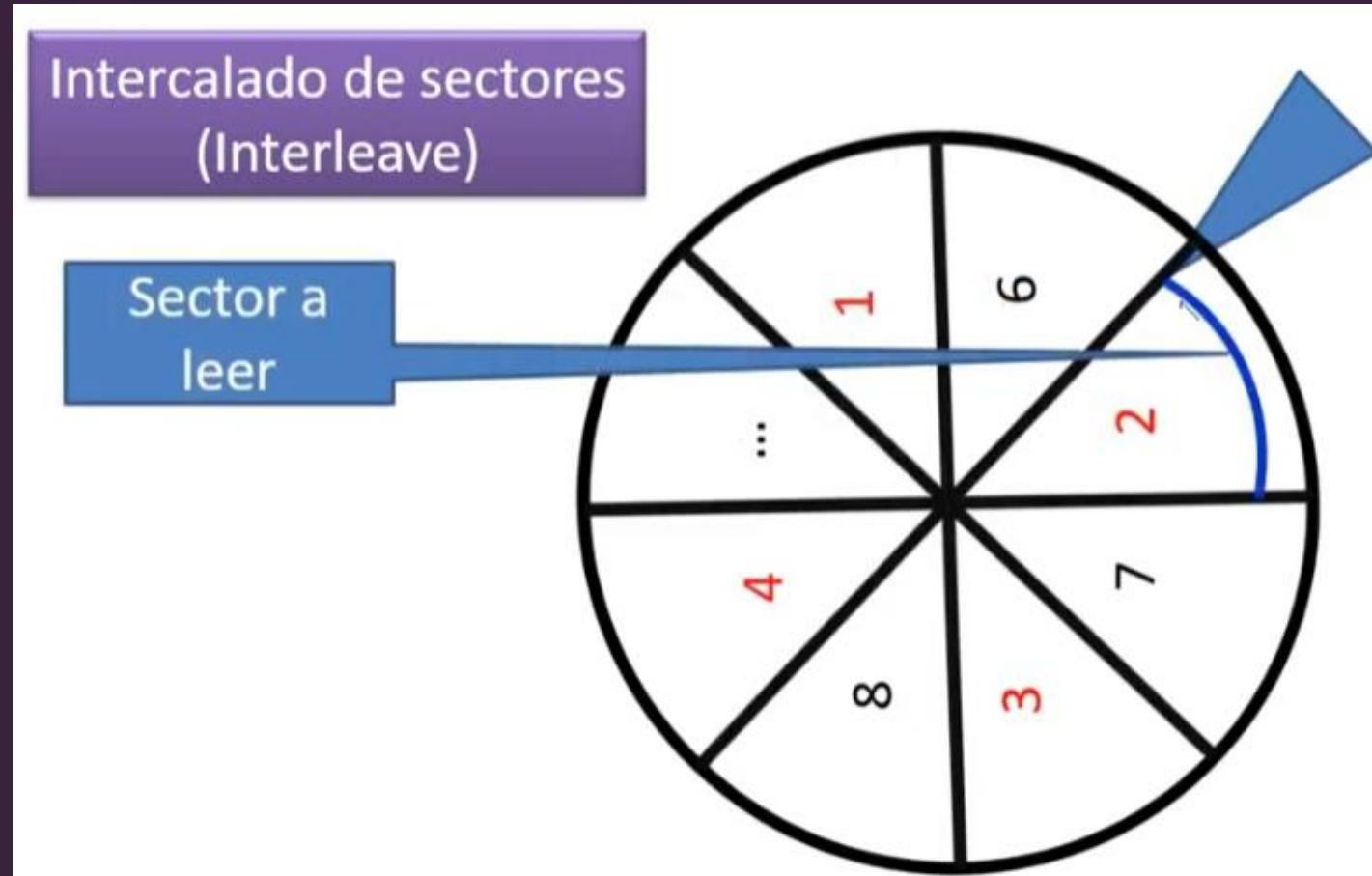
Sectores físicos, cilindros y superficies



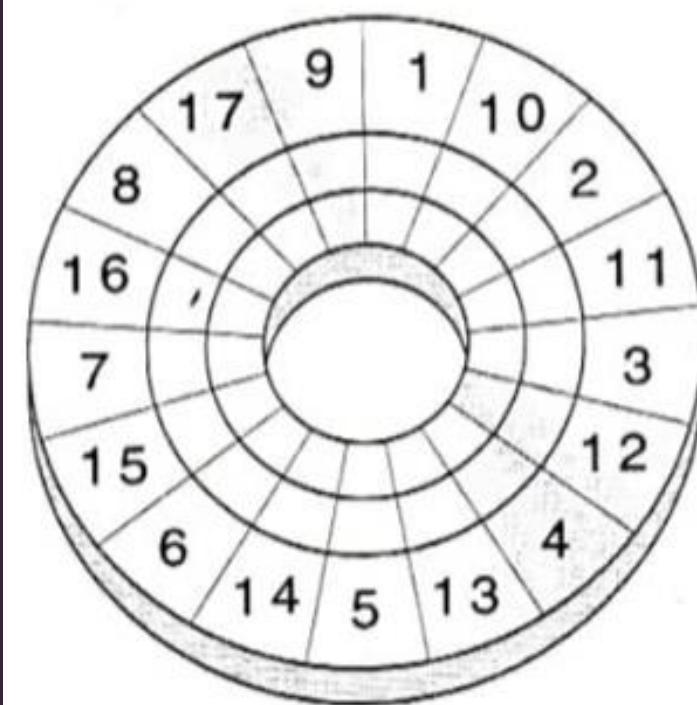
Sectores físicos, cilindros y superficies



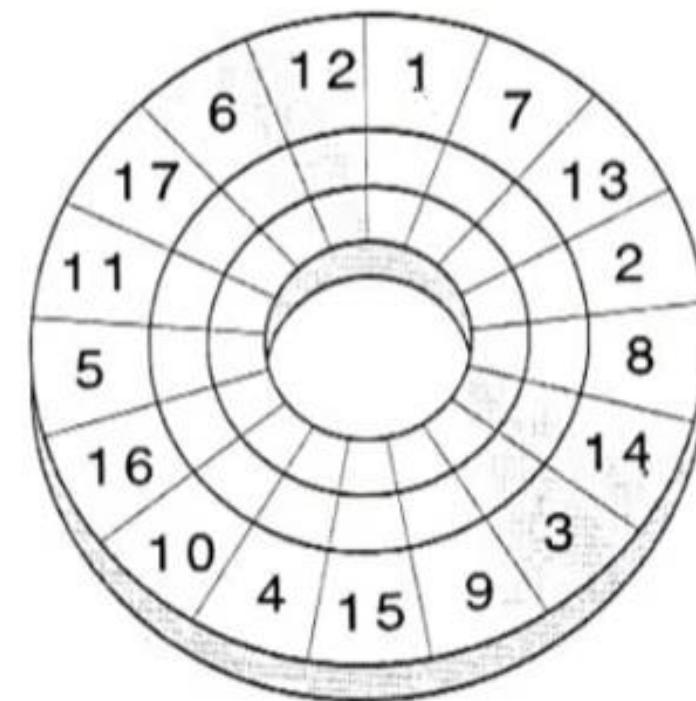
Sectores físicos, cilindros y superficies



Sectores físicos, cilindros y superficies



Interleave 1:2



Interleave 1:3

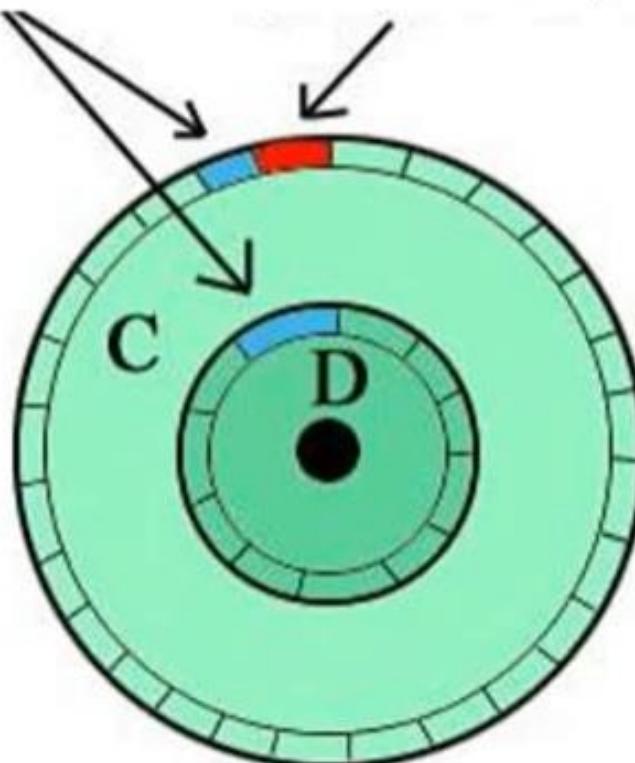
Ejemplo

- Geometría del disco
 - Cilindros = 200
 - Desde el 0 hasta el 199
 - Superficies = 8
 - Desde la 0 hasta la 7
 - Sectores físicos por track = 27
 - Desde el 1 hasta el 27

Total de sectores de la unidad = $200 * 8 * 27 = 43200$

Particiones de disco

Sector de arranque o superbloque de cada partición



MBR (Sector de arranque maestro)
Cilindro=0, Superficie=0,Sector físico=1

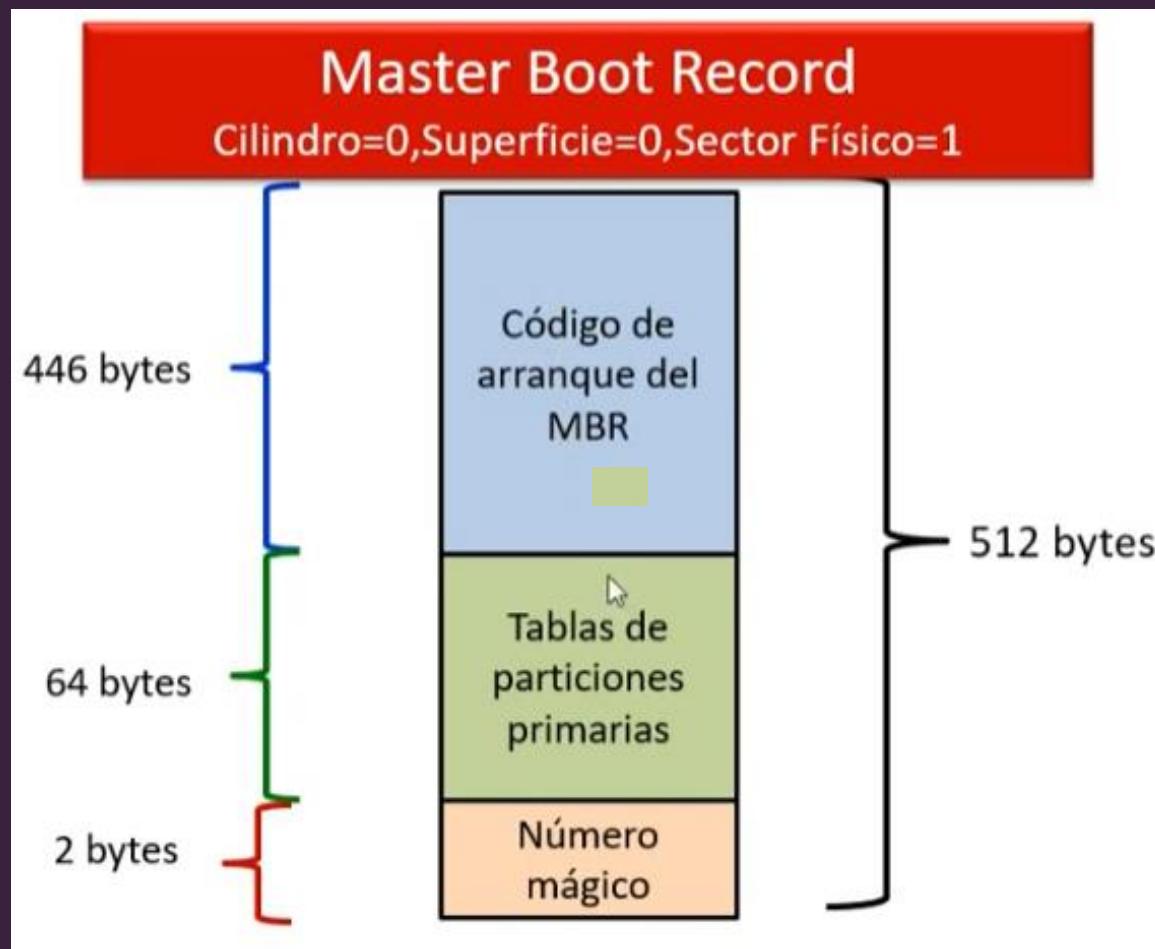
Con las particiones podemos tener múltiples unidades de almacenamiento lógicas en una unidad física

Cada partición puede estar formateada con un sistema de archivos diferente

Las particiones permiten administrar el control sobre el uso del disco.

- Una partición para intercambio de la memoria
- Una partición para instalar aplicaciones
- Una partición para archivos de usuarios

Particiones de disco



Particiones de disco

Tabla de particiones primarias			
Partición 0		Partición 2	
1 byte	Marca de arranque (bit7)	1 byte	Marca de arranque (bit7)
3 bytes	CHS de inicio	3 bytes	CHS de inicio
1 byte	<u>Tipo de partición</u>	1 byte	<u>Tipo de partición</u>
3 bytes	CHS final	3 bytes	CHS final
4 bytes	LBA	4 bytes	LBA
4 bytes	Tamaño en sectores	4 bytes	Tamaño en sectores
Partición 1		Partición 3	
1 byte	Marca de arranque (bit7)	1 byte	Marca de arranque (bit7)
3 bytes	CHS de inicio	3 bytes	CHS de inicio
1 byte	<u>Tipo de partición</u>	1 byte	<u>Tipo de partición</u>
3 bytes	CHS final	3 bytes	CHS final
4 bytes	LBA	4 bytes	LBA
4 bytes	Tamaño en sectores	4 bytes	Tamaño en sectores

16 bytes [] 16 bytes [] 16 bytes []

Particiones de disco

CHS = Cylinder, Head, Sector

1 byte	h ₇₋₀		superficie ^[el]
	x x x x x x x x x		
1 byte	c ₉₋₈	s ₅₋₀	Sector en los bits 5–0; bits 7–6 son los bits de la parte alta del cilindro ^[el]
	x x x x x x x x x		
1 byte	c ₇₋₀		bits 7–0 del cilindro
	x x x x x x x x x		

Ejemplo

Si la partición inicia en

Cilindro: 0

Superficie: 0

Sector físico: 2

00000000	00000010	00000000
----------	----------	----------

00 02 00

Si la partición termina en

Cilindro: 199

Superficie: 7

Sector físico: 27

00000111	00011011	11000111
----------	----------	----------

07 1B C7

Sectores lógicos



Es mucho más fácil manejar los sectores numerados de forma **lineal** que de una forma tridimensional

Sectores lógicos

```
sectorLógicoInicioPartición = cilindroInicial * SEC_X_TRACK * SUPERFICIES +  
superficieInicial * SEC_X_TRACK +  
sectorInicial -  
1
```

Ejemplo

cilindroInicial = 0

superficieInicial = 0

sectorInicial = 2

SEC_X_TRACK = 27

SUPERFICIES = 8

$$\begin{aligned}\text{sectorLógicoInicioPartición} &= 0 * 27 * 8 + 0 * 27 + 2 - 1 \\ &= 1\end{aligned}$$

Sectores lógicos

SEC_X_TRACK = 27

SUPERFICIES = 8

sectorLógicoInicioPartición = 1

sectorFísico = (sectorLógico + sectorLógicoInicioPartición) % SEC_X_TRACK + 1

superficie = (sectorLógico + sectorLógicoInicioPartición) / SEC_X_TRACK % SUPERFICIES

cilindro = (sectorLógico + sectorLógicoInicioPartición) / (SEC_X_TRACK * SUPERFICIES)

Ejemplo: sectorLógico = 1000

sectorFísico = (1000 + 1) % 27 + 1 = 3

superficie = (1000 + 1) / 27 % 8 = 5

cilindro = (1000 + 1) / (27 * 8) = 4

Formateo

- Dos tipos de formateo
 - De bajo nivel o formateo físico
 - Alto nivel o lógico

Formateo

- De bajo nivel o formateo físico
 - Se caracteriza por ser un proceso de gran lentitud, por la rigurosidad con la que se tiene que realizar.
 - Se hace la selección del interleave.
 - Actualmente suele ser aplicado, por el fabricante.
 - Esta tarea actualmente no la tiene que hacer el usuario.

Formateo

- Alto nivel o lógico
 - Para crear un sistema de archivos en una partición.
 - Realizado de manera rápida.
 - Escribe o reescribe para inicializar las áreas del sistema de archivos utilizadas para:
 - Llevar el control de los bloques libres
 - Bloques ocupados
 - Directorio raíz

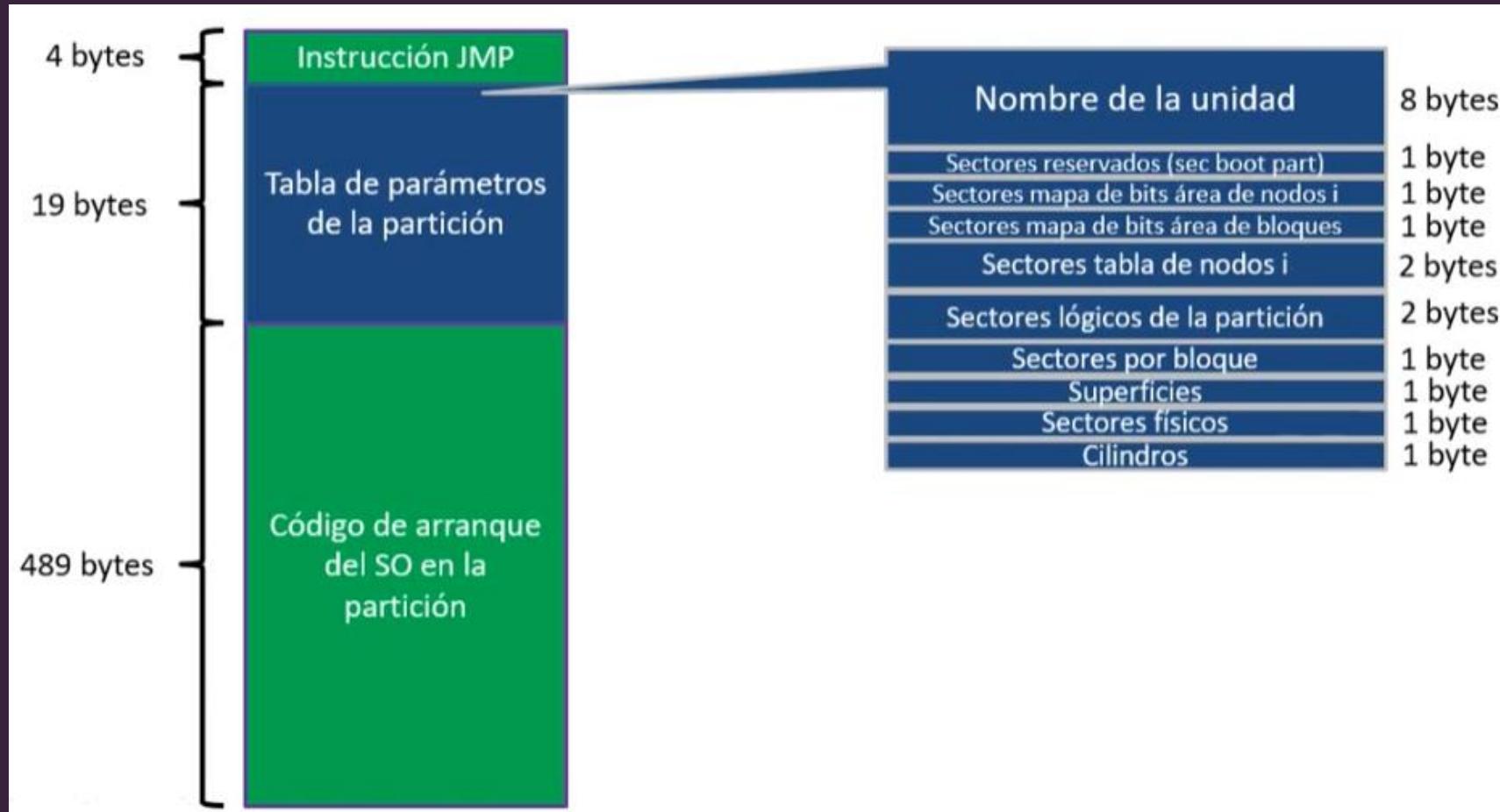
Formateo

- Alto nivel o lógico
 - No requiere reescribir todos los demás sectores del disco
 - Al menos que haga un formateo completo que puede durar horas
 - Aunque los archivos aun existan; después de un tiempo, y con el almacenamiento de nuevos datos, se reescribirán los anteriores, haciéndolos irrecuperables.
 - Escribe el sector de arranque de la partición o superbloque
 - Inicializa los parámetros de la partición.

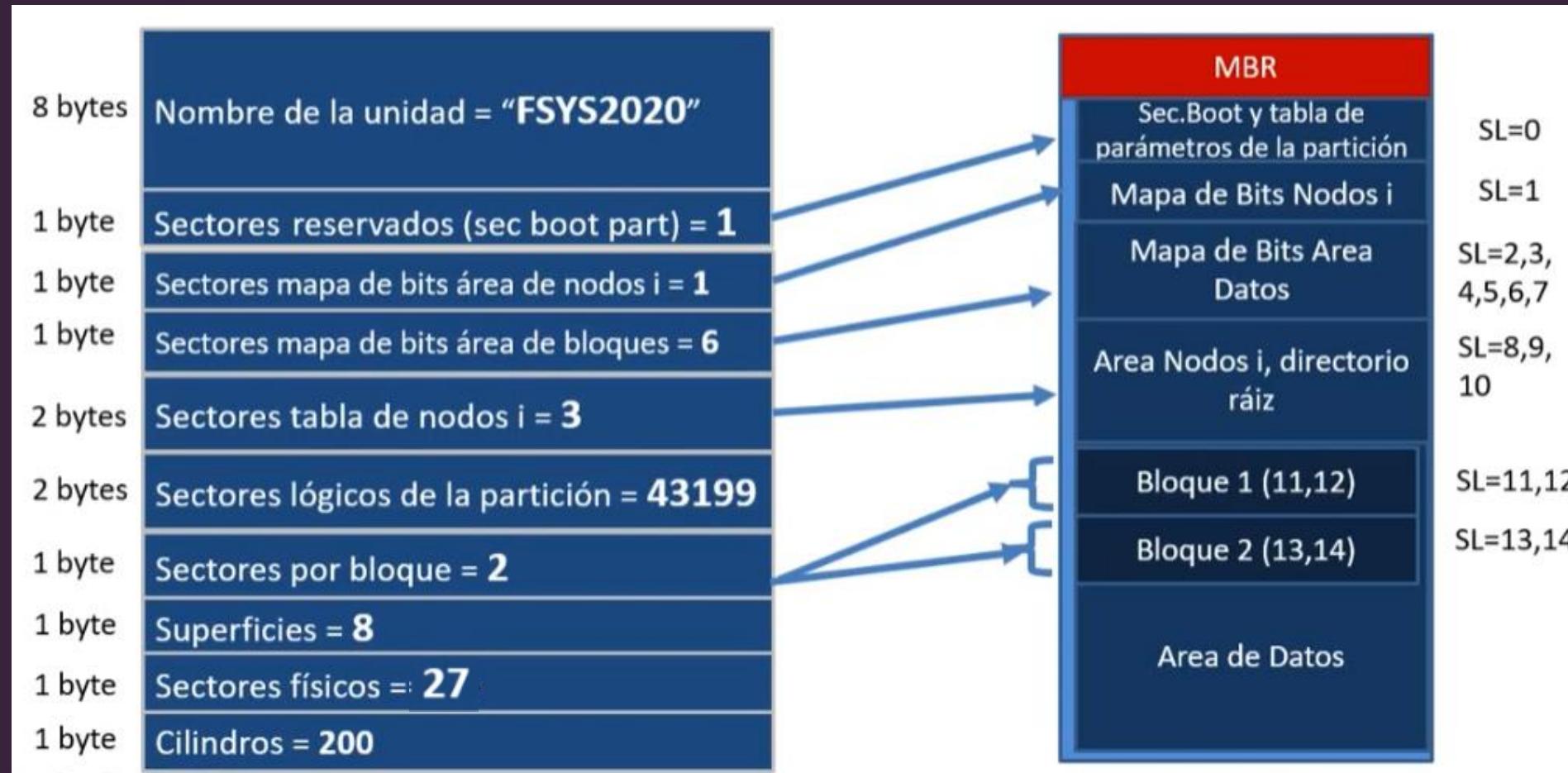
Sector de arranque de la partición o superbloque

- Por cada partición debe de haber un sector de arranque o superbloque
- Mide 512 bytes y contiene:
 - Código de arranque del SO instalado en la partición.
 - Tabla de parámetros de la partición
 - Información que describe al sistema de archivos como el tamaño de las distintas partes que la conforman.

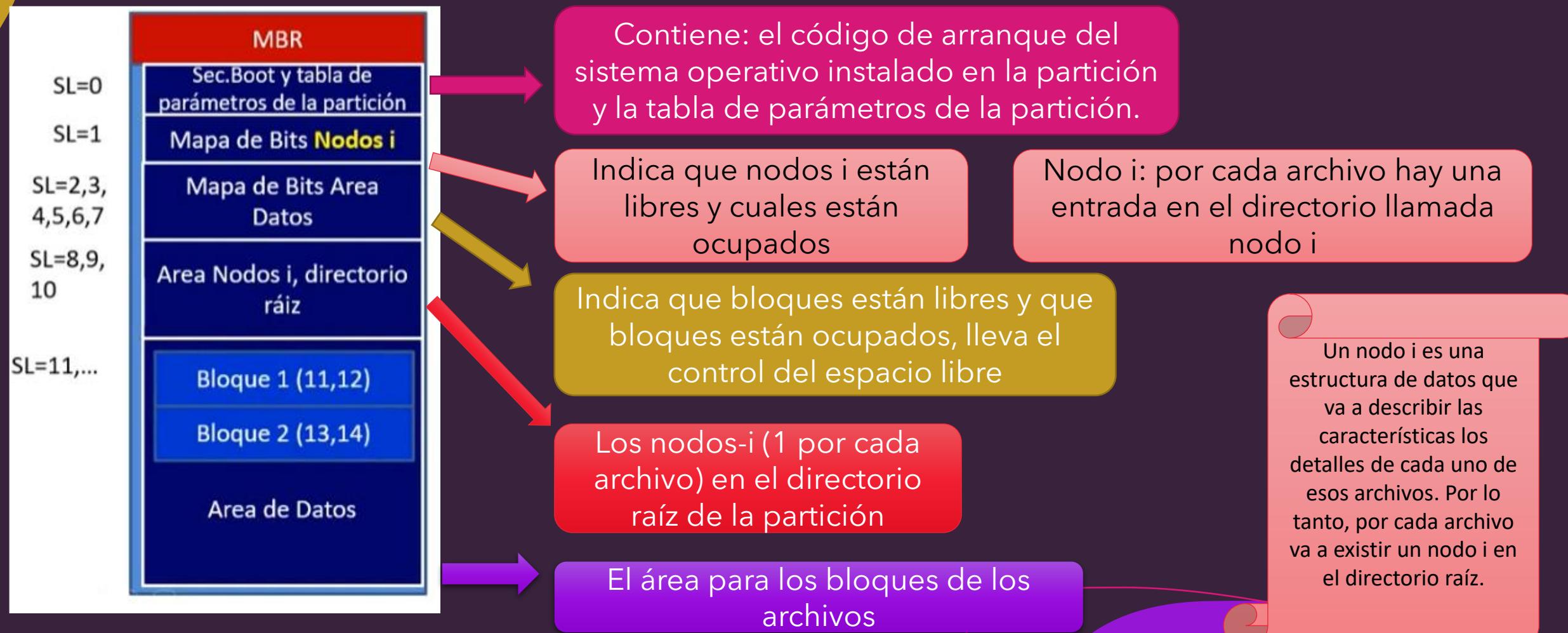
Sector de arranque de la partición o superbloque



Sector de arranque de la partición o superbloque



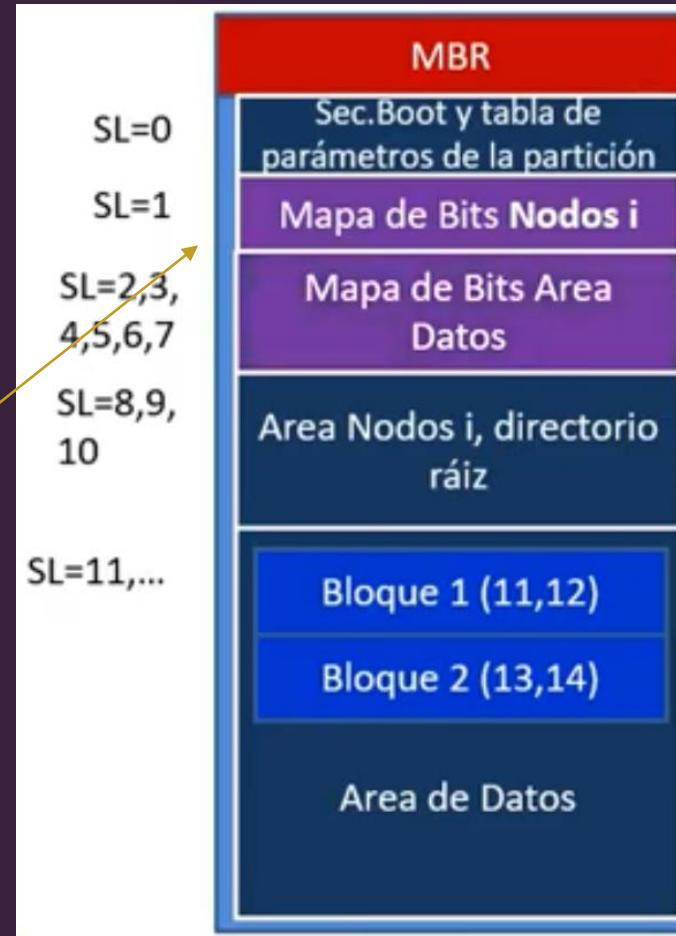
Áreas del sistema de archivos



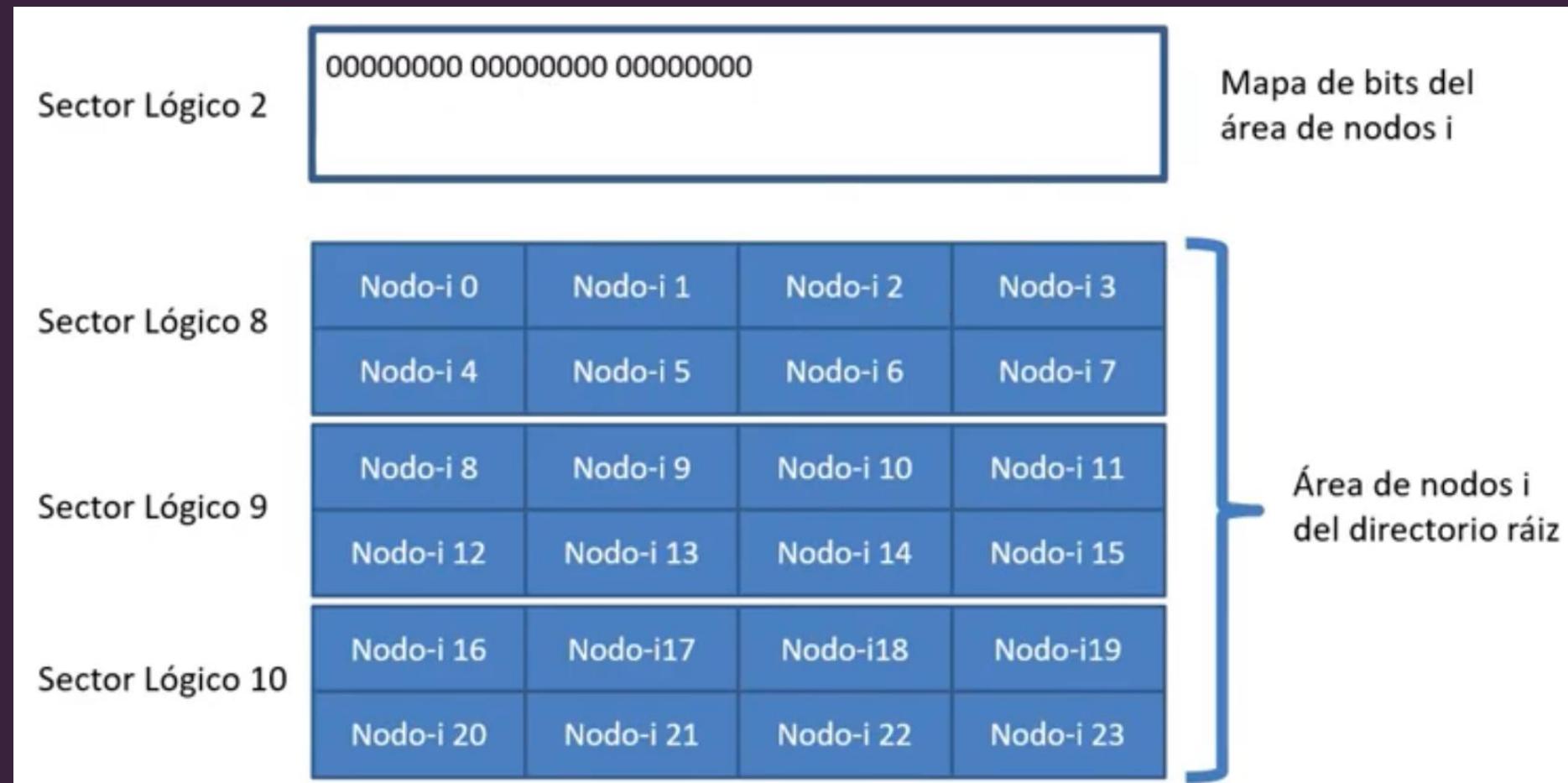
Mapas de bits

Nuestro sistema de archivos tendrá 2 mapas de bits

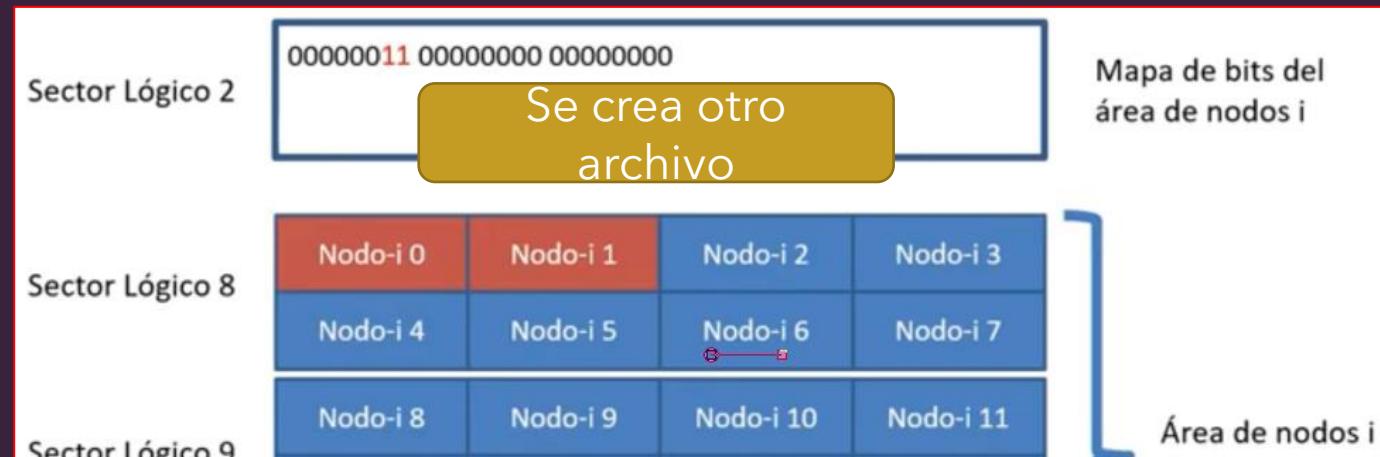
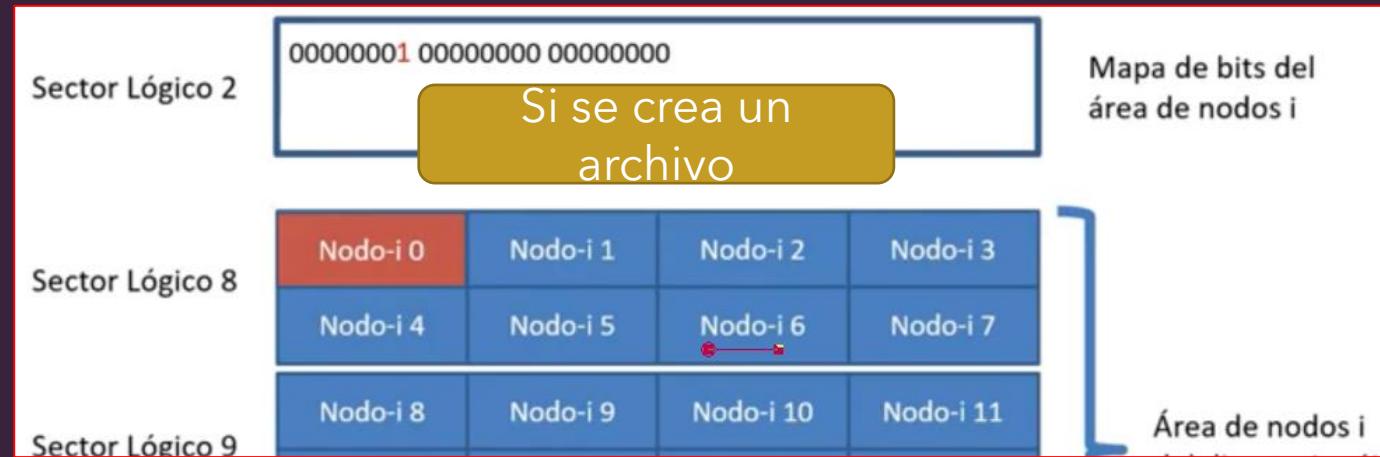
- Se utilizará para llevar el control de los nodos-i libres y ocupados.
- Por cada nodo-i existe un bit que indica su estado.
- Si el bit es 0 significa que el nodo-i está libre, y si es 1 está ocupado.



Mapa de bits del área de nodos i



Mapa de bits del área de nodos i



Mapa de bits del área de nodos i

Sector Lógico 2

00000010 00000000 00000000

Si se borra el primer
archivo

Mapa de bits del
área de nodos i

Sector Lógico 8

Nodo-i 0	Nodo-i 1	Nodo-i 2	Nodo-i 3
Nodo-i 4	Nodo-i 5	Nodo-i 6	Nodo-i 7
Nodo-i 8	Nodo-i 9	Nodo-i 10	Nodo-i 11

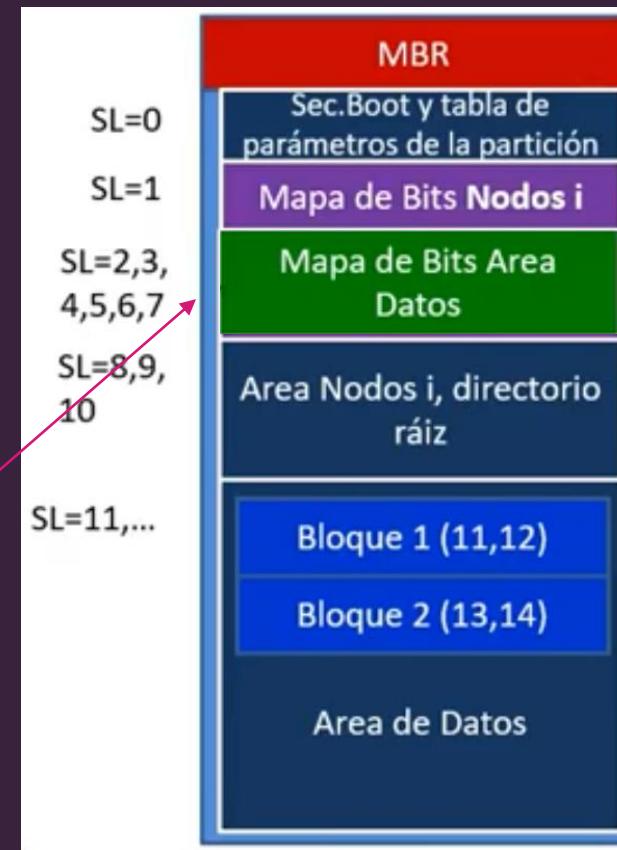
Sector Lógico 9

Área de nodos i

Mapa de bits del área de datos

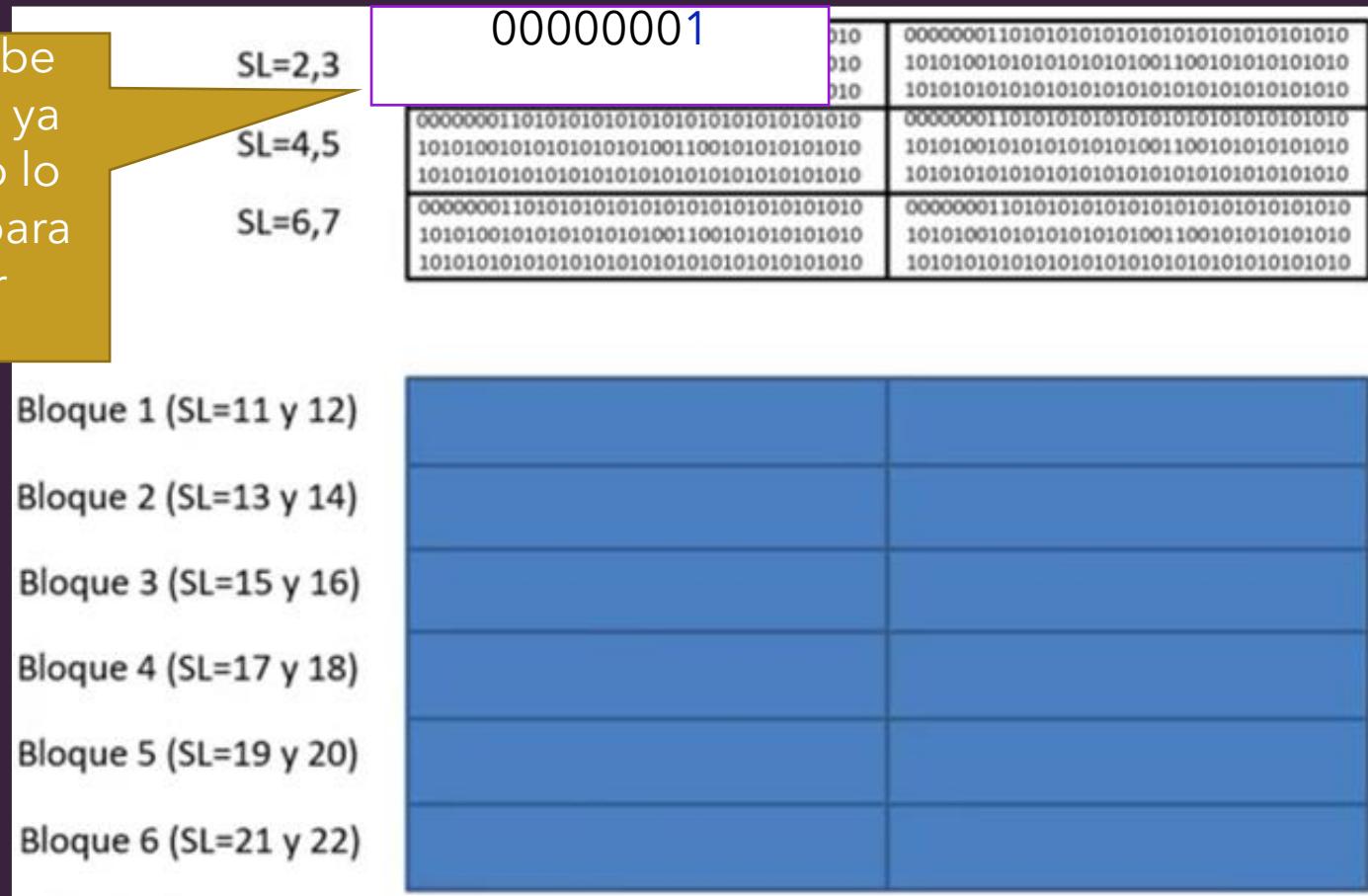
Nuestro sistema de archivos tendrá dos mapas de bits.

- Se utiliza para llevar el control de los bloques del área de datos. (necesitamos saber que bloques están libres y cuales están ocupados)
- La información contenida en los archivos se almacena en los bloques en el área de datos. (El primer bloque no parte del 0 ya que es un valor reservado utilizado para indicar que no se apunta ningún bloque)
- Su funcionamiento es idéntico al del mapa de bits de los nodos-i. (Si el bit correspondiente a un nombre esta en 0 quiere decir que ese bloque esta libre y puede ser utilizado para escribir un archivo. Si el bit en cambio esta en 1 quiere decir que ese bloque esta ocupado y no debe ser utilizado)



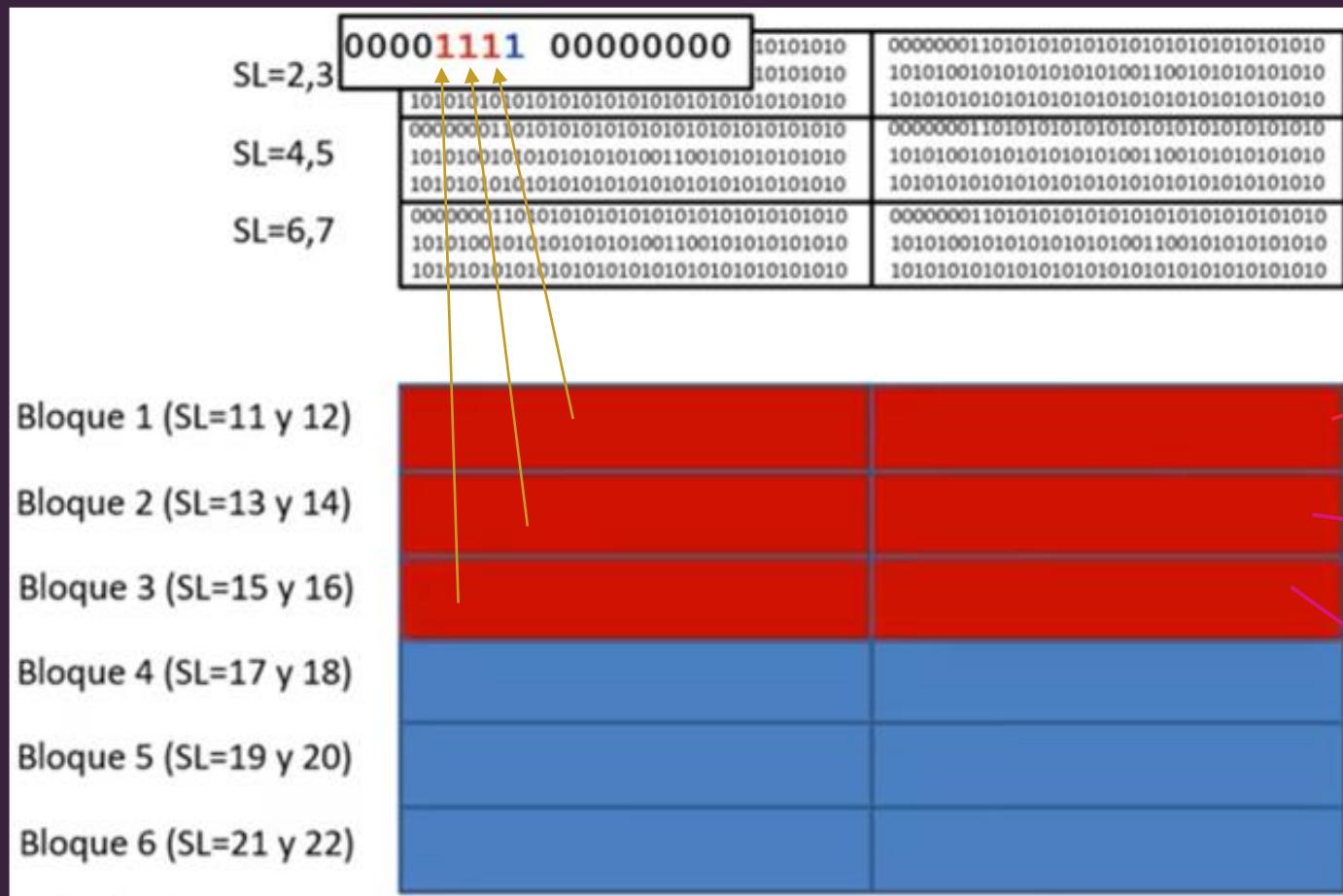
Mapa de bits del área de datos

Este bit debe iniciar en 1 ya que el 0 no lo usaremos para nombrar bloques



- Cada bit mapea un bloque
- 6 sectores para el mapa de bits
- $=512 \times 6 = 3072$ bytes
- $=3072 \times 8 = 24576$ bits
- Se pueden mapear 24576 bloques

Mapa de bits del área de datos

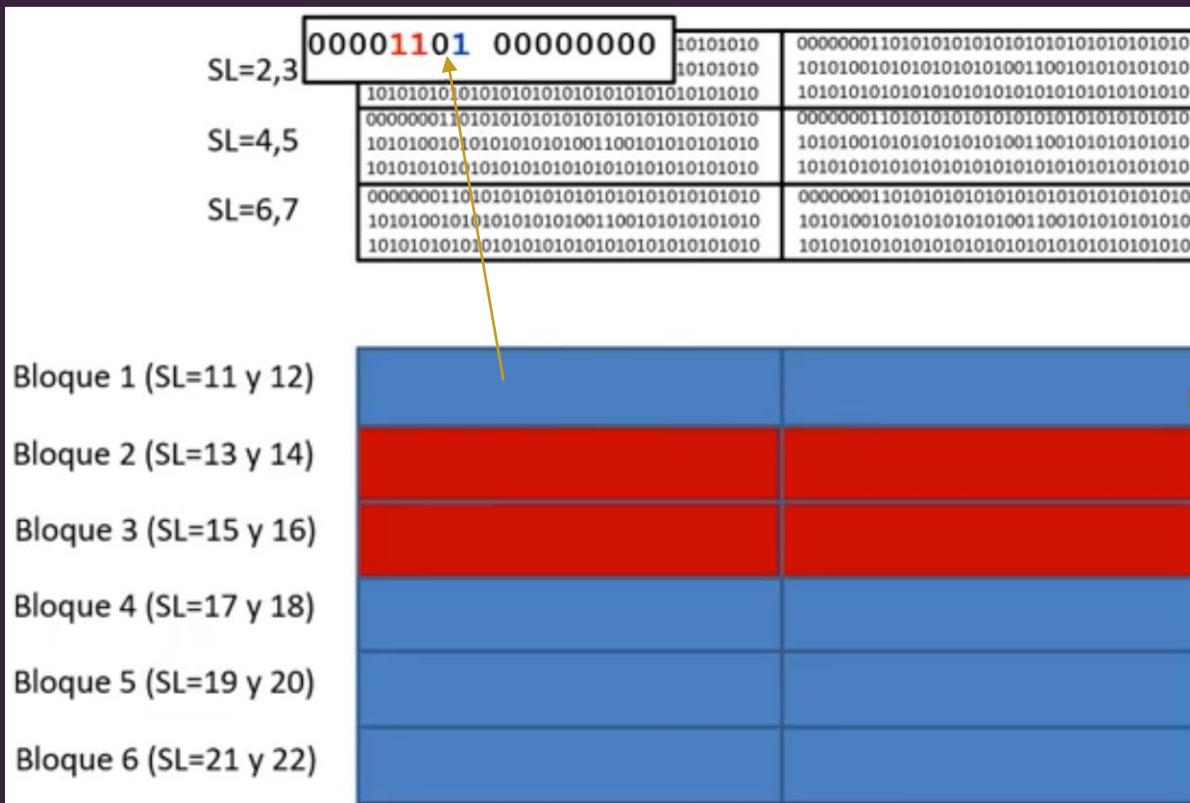


Se asigna el bloque 1 al archivo A

Se asigna el bloque 2 al archivo B

Se asigna el bloque 3 al archivo B

Mapa de bits del área de datos



Si se elimina el archivo A

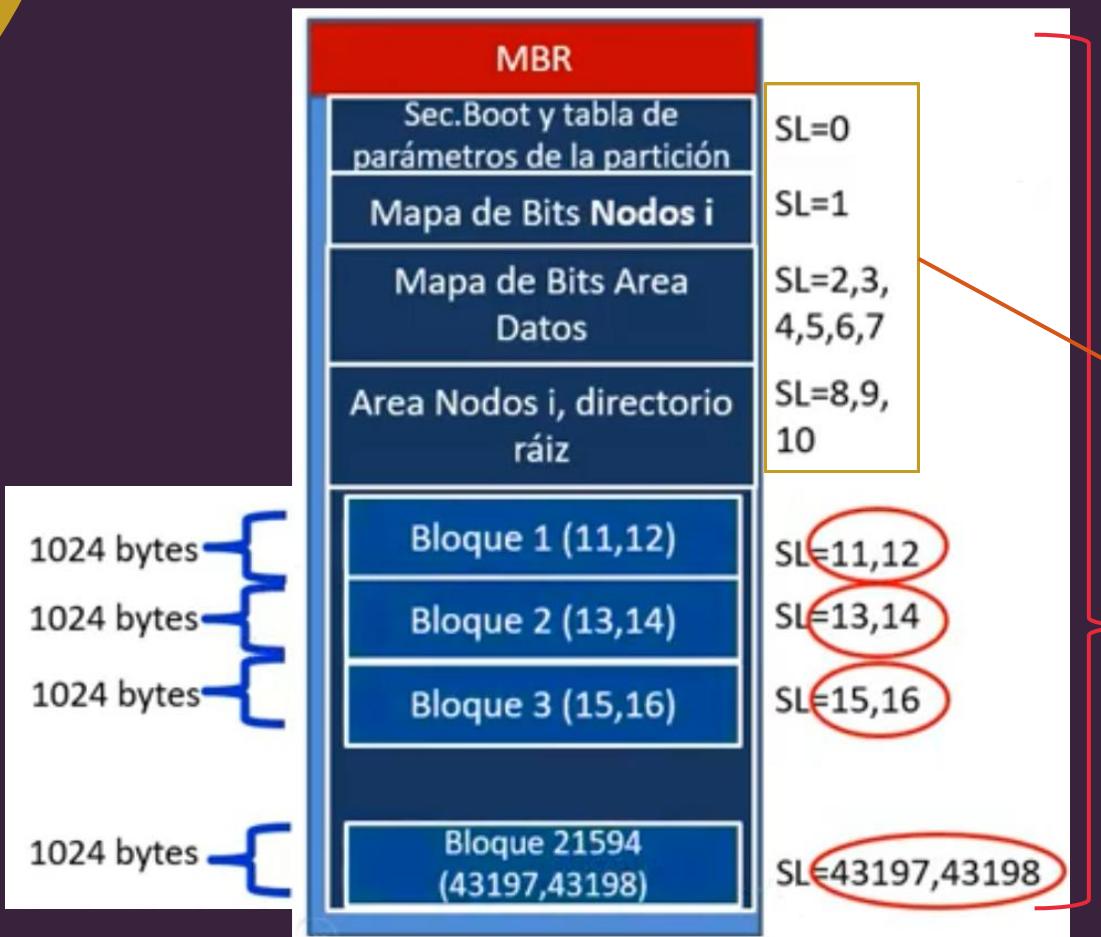
Bloques

- Se forma de uno o mas sectores
- Si es mas de uno, usualmente el tamaño es potencia de 2
 - 2 sectores x bloque
 - 4 sectores x bloque
 - 8 sectores x bloque
 - 16 sectores x bloque
 - 32 sectores x bloque
 - 64 sectores x bloque
 - 128 sectores x bloque
 - ...

Bloques

- En los bloques se almacena el contenido de los archivos o apuntadores a otros bloques
- Es la unidad mínima de asignación para los archivos
 - Un archivo por mas pequeño que sea va a ocupar un bloque completo
- El primer bloque del sistema de archivos será el bloque 1
 - El numero 0 es un valor reservado.

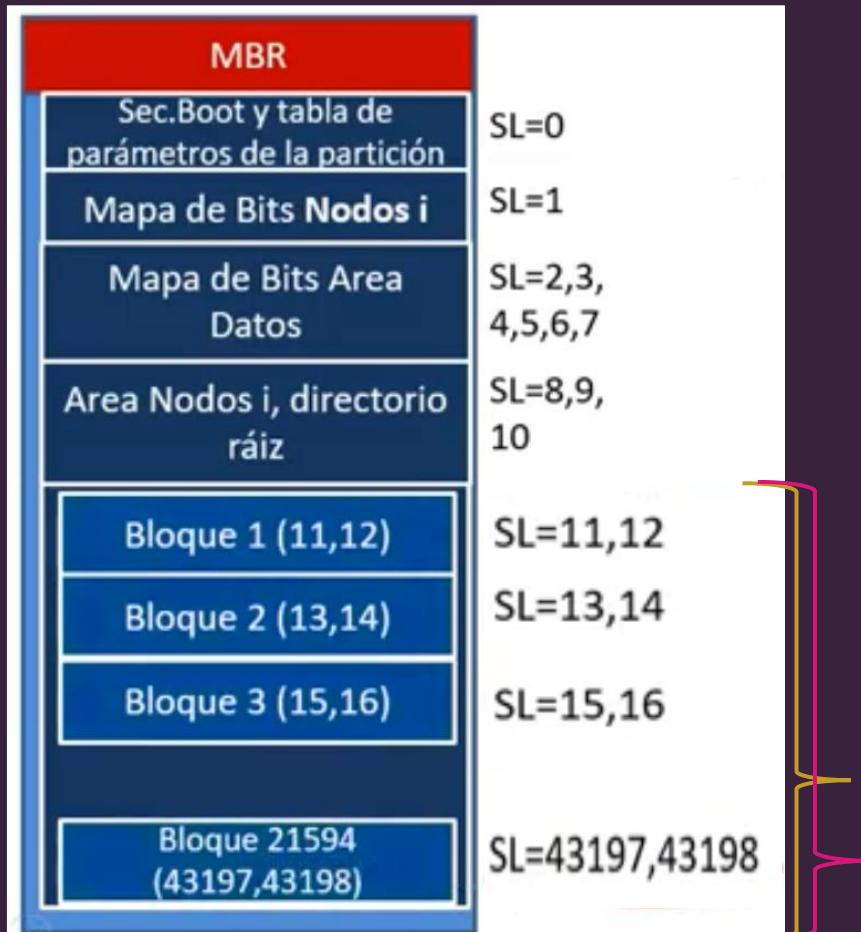
Bloques



- En nuestro sistema de archivos un bloque será conformado por dos sectores
- 1 bloque = 2×512 bytes = 1024 bytes
- La partición tiene 43199 sectores, menos 11 sectores que usamos para las áreas criticas = 43188 sectores para los bloques.

$$43199 - 11 = 43188 \text{ sectores}$$

Bloques



- En nuestro sistema de archivos un bloque será conformado por dos sectores
 - 1 bloque = 2×512 bytes = 1024 bytes
 - La partición tiene 43199 sectores, menos 11 sectores que usamos para las áreas criticas = 43188 sectores para los bloques.
 - 43188 sectores en el área de archivos
 - 43188 entre 2 sectores x bloque = 21594 bloques.

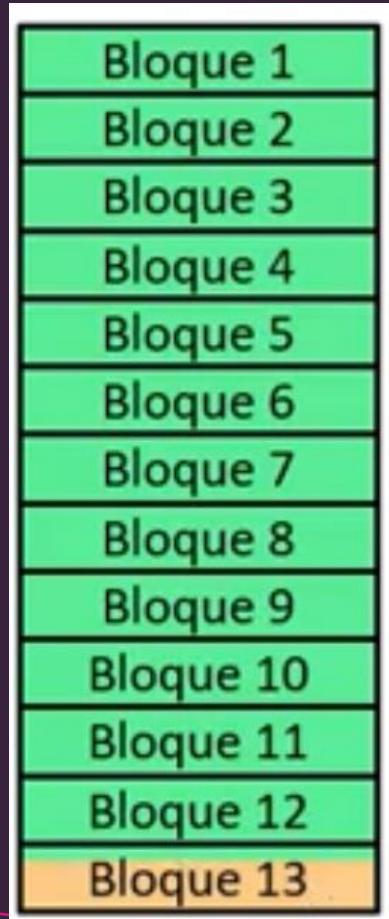
43188 sectores / 2 sectores x bloque
= 21594 bloques

Bloques grandes vs bloques pequeños

- ¿Cuántos sectores por bloque?
- Bloques pequeños
 - Pocos sectores por bloque
 - Muchos bloques por archivo
 - Se desperdicia menos espacio en el ultimo bloque de los archivos.
 - Muchos bloques por partición -> Muchos sectores para el mapa de bits.

1 bloque = 1
sector

00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010

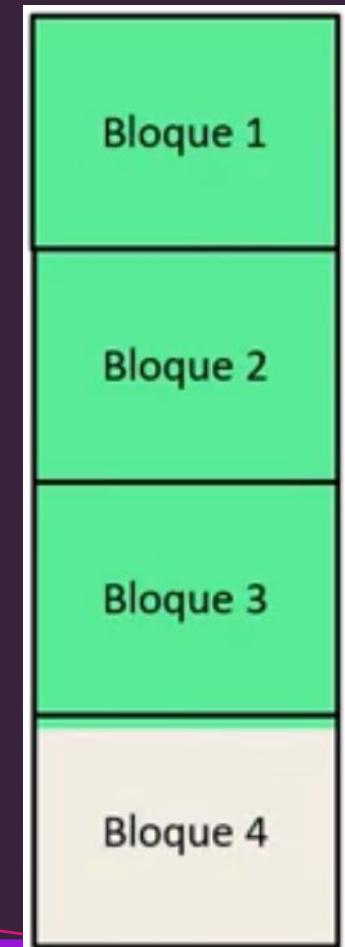


Bloques grandes vs bloques pequeños

- ¿Cuántos sectores por bloque?
- Bloques grande
 - Muchos sectores por bloques
 - Pocos bloques por archivo
 - Se desperdicia mucho espacio en el ultimo bloque de los archivos
 - Pocos bloques por partición -> Pocos sectores para el mapa de bits

1 bloque = 4
sectores

00001110101010101010101010
01010101001000011100010010
00001110101010101010101010
01010101001000011100010010



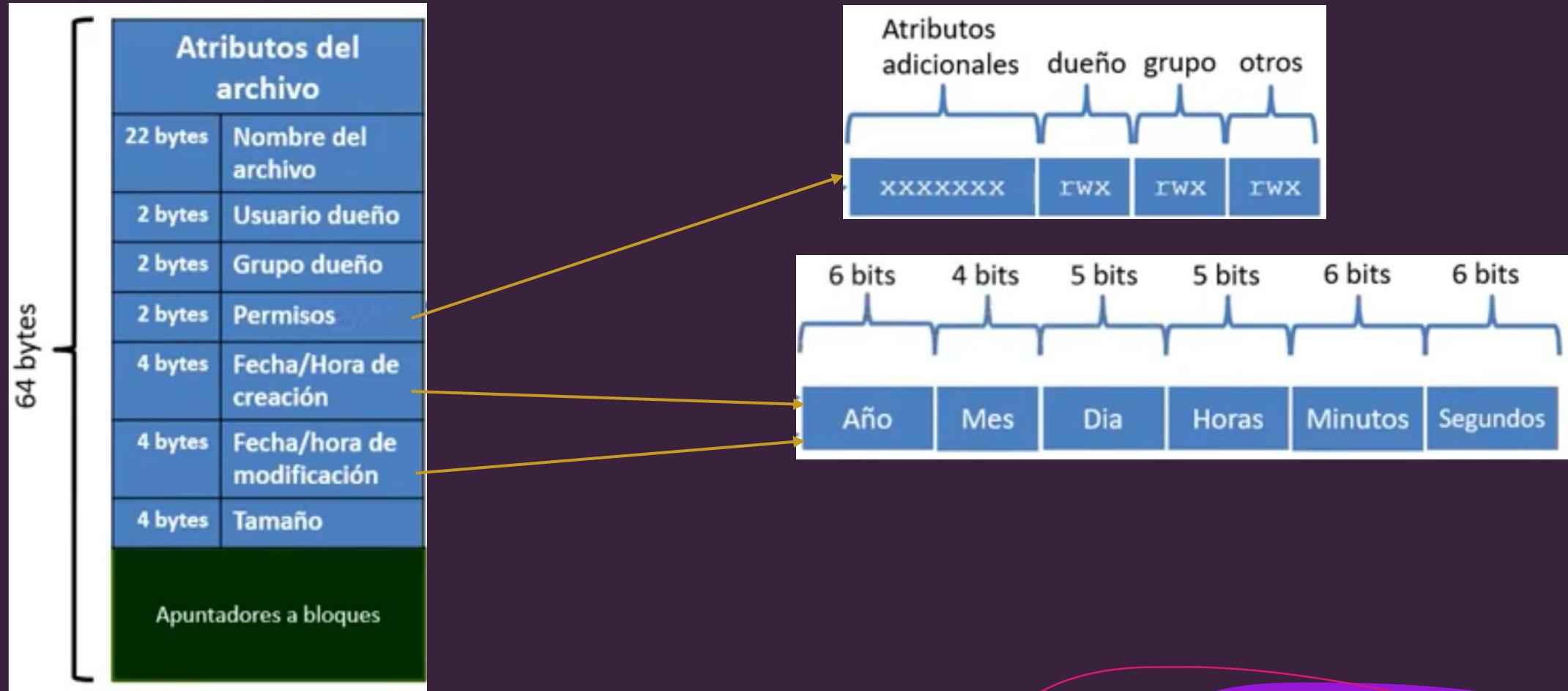
Nodos i

- Nodo índice
- Estructura de datos propia de los sistemas de archivos.
 - Común en los sistemas operativos tipo UNIX, por ejemplo Linux

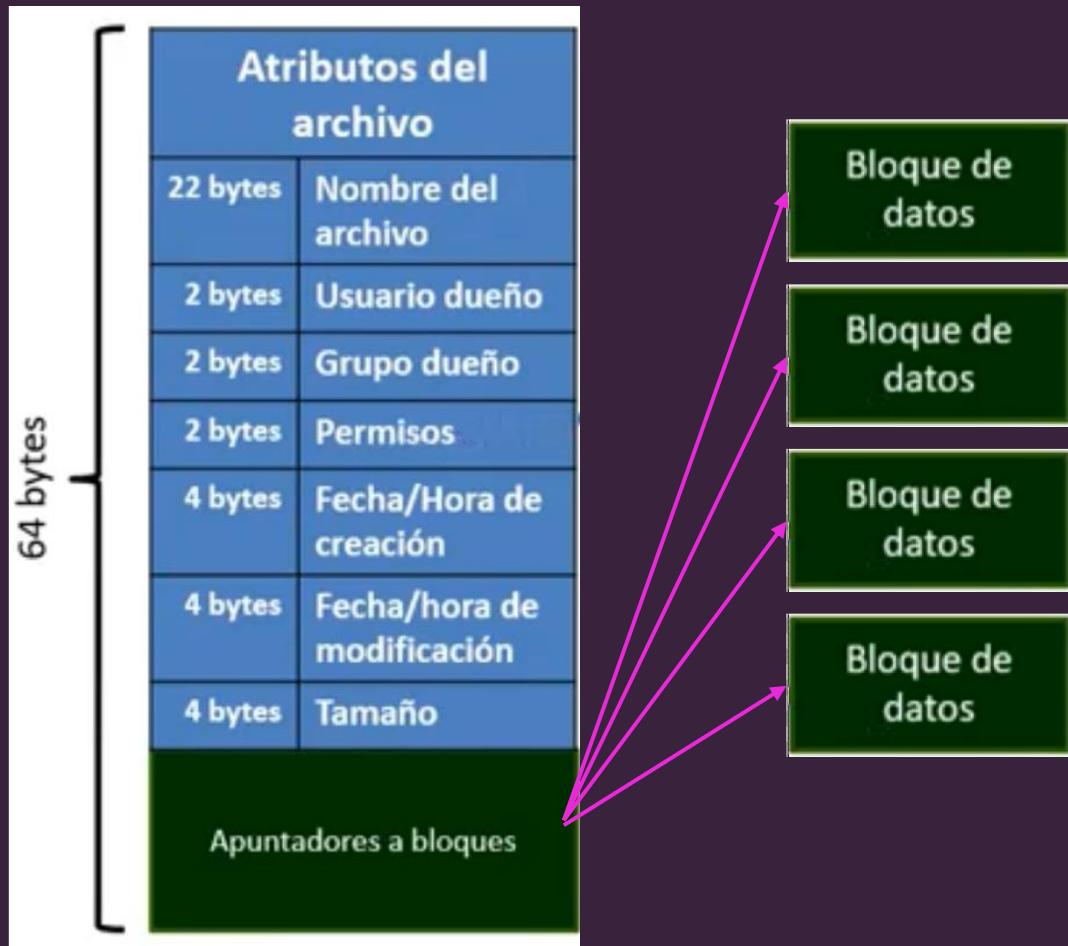
Un nodo-i contiene la descripción de:

- Un archivo regular
- Un directorio
- Enlaces simbólicos
 - Puede tener más de un nombre en distintos o incluso en el mismo directorio

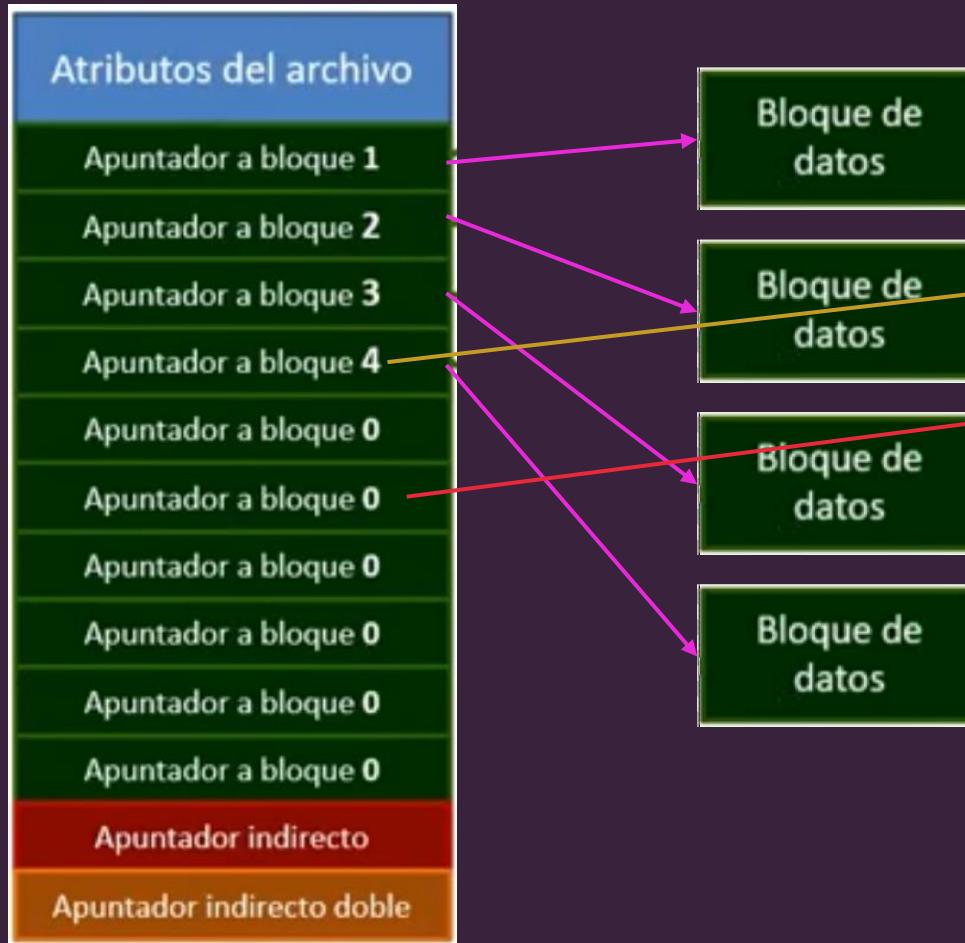
Nodos i



Nodos i



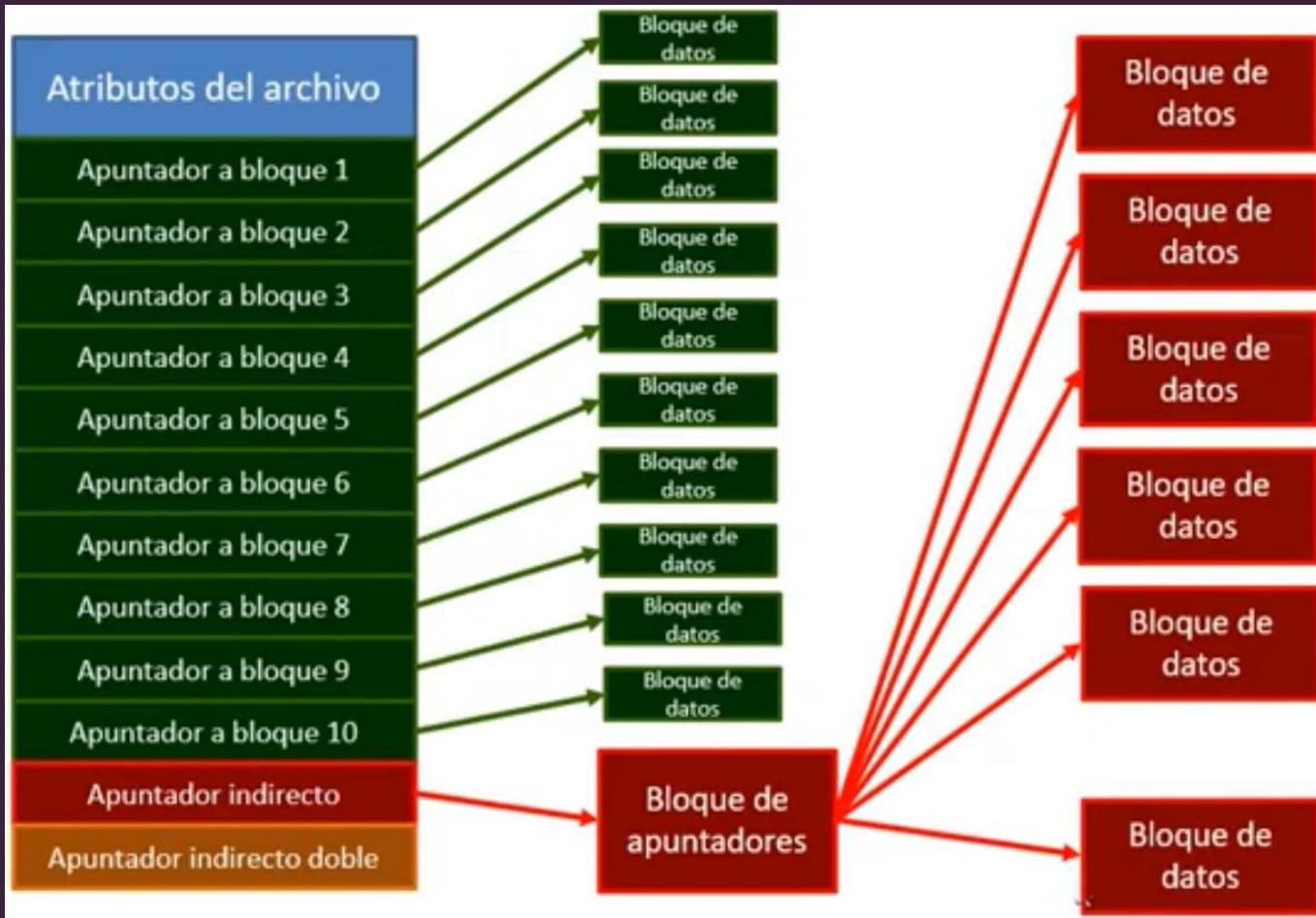
Nodos i



- Los apuntadores a bloques son de 16bits
 - Máximo 65535 bloques
- Cada apuntador indica un bloque del archivo
- Si el apuntador es 0 entonces no apunta a ningún bloque

Con 10 apuntadores un archivo podría medir 10 Kbytes
Podría tener un pequeño archivo de texto solamente

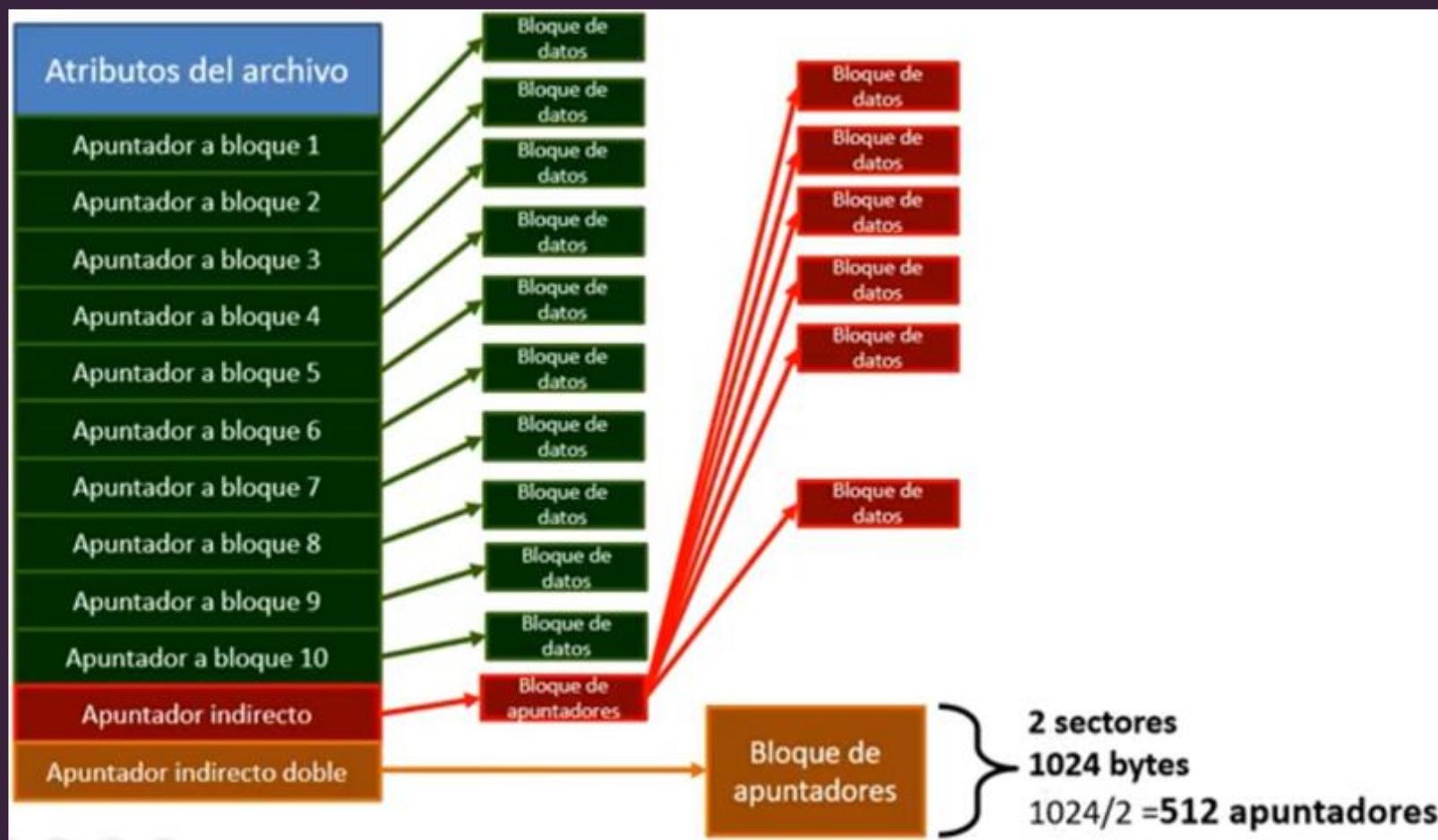
Nodos i



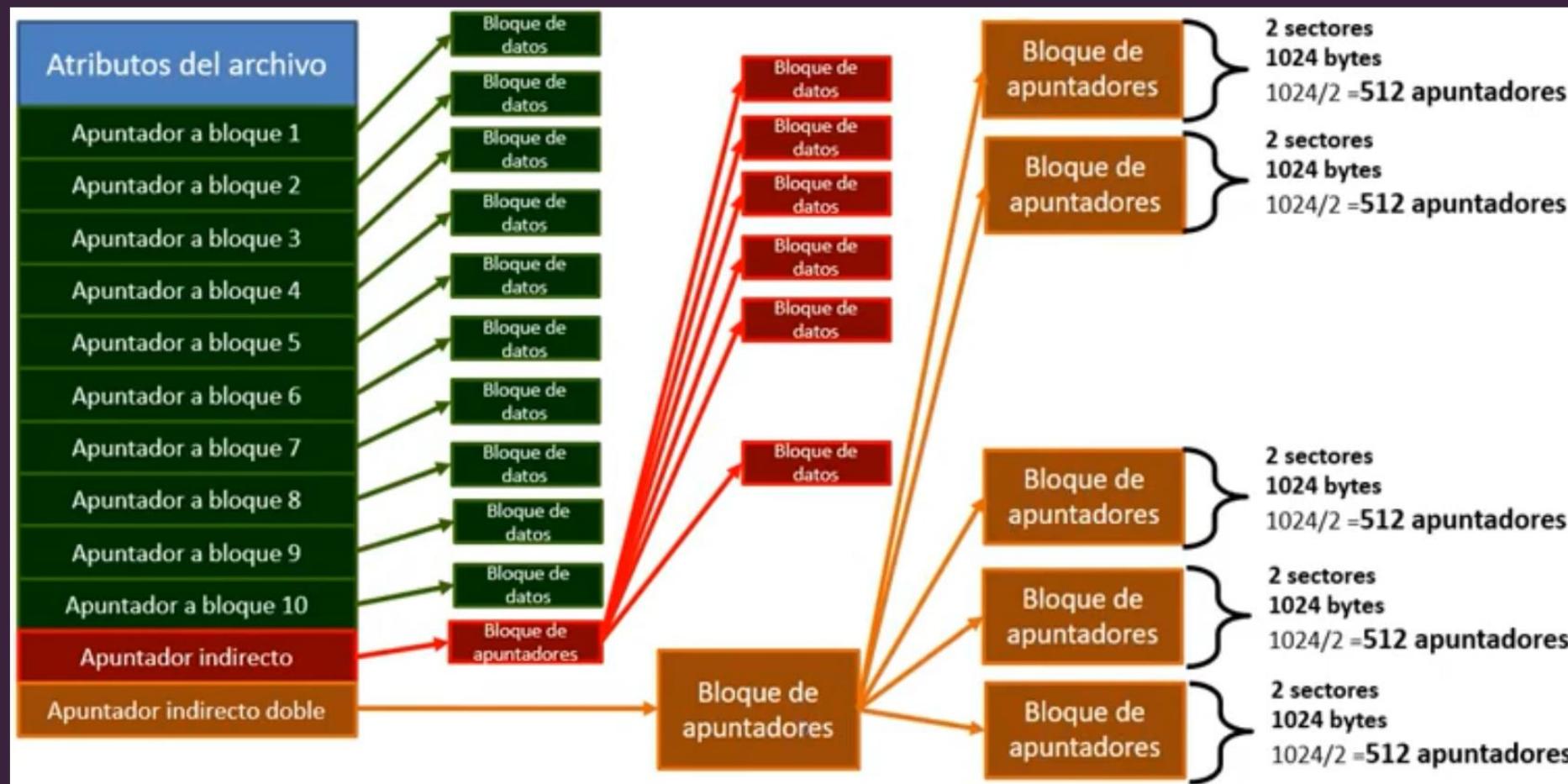
Ahora un archivo puede tener
 $10+512=522$ bloques

- 522 Kbytes máximo por archivo
- Medio minuto de audio MP3 a 128 Kbps (Calidad media)

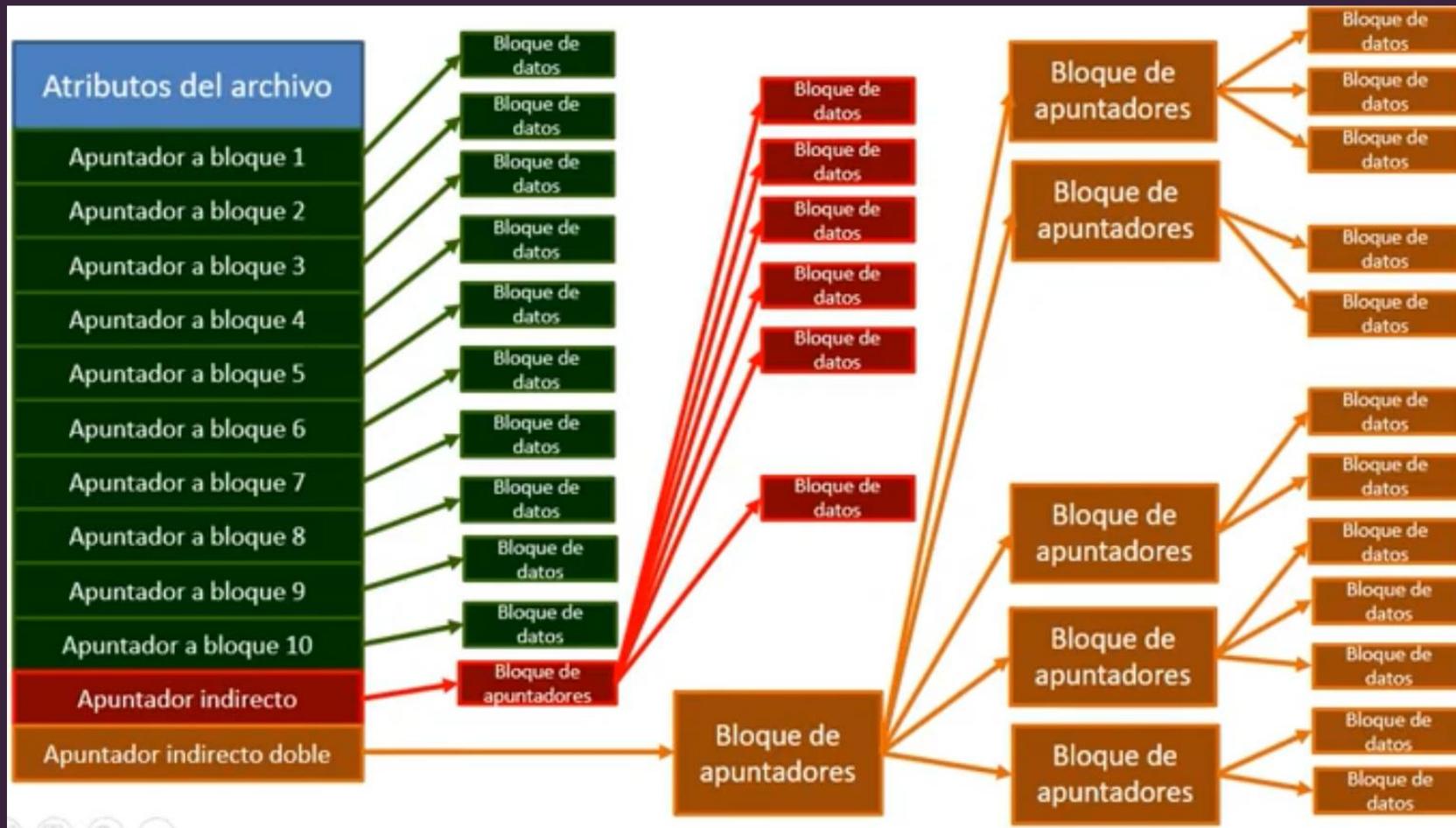
Nodos i



Nodos i



Nodos i



$$10 + 512 + 512^2 = 262666 \text{ bloques por archivo}$$

- Mis archivos podrían medir hasta 256mb.

Operaciones con archivos

- Crear archivos
- Eliminar archivos
- Abrir archivos
- Lectura
- Posicionamiento dentro de un archivo
- Escritura dentro de un archivo
- Cerrar archivos
- Directorios

Operaciones con los archivos/Crear archivos

Nombre del archivo
UID
GID
Perms
Fecha/Hora de creación
Fecha/Hora de modificación
Tamaño = 0
Apuntador a bloque = 0
Apuntador a bloque = 0
Apuntador a bloque = 0
Apuntador a bloque = 0
Apuntador a bloque = 0
Apuntador a bloque = 0

Int vdcreat(char *filename, unsigned short perms)

Mapa de bits del área de nodos-i

11111111 11000011 11111111

- Buscar en el mapa de bits de nodo-i un nodo-i libre
- Poner el nombre del archivo en el nodo-i encontrado
- Establecer el id del usuario y grupo que crea el archivo
- Establecer los permisos indicados en el segundo argumento
- Establecer fecha y hora de creación y modificación con la hora del sistema
- Establecer el tamaño del archivo en 0
- Inicializar todos los apuntadores a bloques directos, indirectos en 0s
- El archivo ya está abierto

Operaciones con los archivos/Eliminar archivos

Tabla de nodos i
Nodo-i 0
Nodo-i 1
Nodo-i 2
Nodo-i 3
Nodo-i 4
Nodo-i 5
Nodo-i 6
Nodo-i 7
Nodo-i 8

Int vdunlink (char ***filename**)

- Buscar el nodo-i donde esté el nombre del archivo del argumento

Operaciones con los archivos/Eliminar archivos

Nodo i 3
Nombre del archivo
UID
GID
Perms
Fecha/Hora de creación
Fecha/Hora de modificación
Tamaño = 0
Apuntador a bloque = 1
Apuntador a bloque = 2
Apuntador a bloque = 3
Apuntador a bloque = 4
Apuntador a bloque = 0

Int vdunlink (char *filename)

Mapa de bits del área de datos
00000001

- Buscar el nodo-i donde esté el nombre del archivo del argumento
- Recorrer los apuntadores a bloques directos para poner esos bloques como libres en el mapa de bits.
- Recorrer los apuntadores a bloques indirectos para poner esos bloques como libres en el mapa de bits.
- En el mapa de bits de nodos-i establecer el bit del nodo-i en 0

Operaciones con los archivos/Eliminar archivos

Nodo i 3	
Nombre del archivo	
UID	
GID	
Perms	
Fecha/Hora de creación	
Fecha/Hora de modificación	
Tamaño = 0	
Apuntador a bloque = 1	
Apuntador a bloque = 2	
Apuntador a bloque = 3	
Apuntador a bloque = 4	
Apuntador a bloque = 0	

Int vdunlink (char *filename)

Mapa de bits del área de nodos i
00000111

- Buscar el nodo-i donde esté el nombre del archivo del argumento
- Recorrer los apuntadores a bloques directos para poner esos bloques como libres en el mapa de bits.
- Recorrer los apuntadores a bloques indirectos para poner esos bloques como libres en el mapa de bits.
- En el mapa de bits de nodos-i establecer el bit del nodo-i en 0

Operaciones con los archivos/Abrir archivos

Tabla de nodos i
Nodo-i 0
Nodo-i 1
Nodo-i 2
Nodo-i 3
Nodo-i 4
Nodo-i 5
Nodo-i 6
Nodo-i 7
Nodo-i 8

Int vdopen (char *filename, unsigned short mode)

- Buscar el nodo-i donde esté el nombre del archivo del argumento

Operaciones con los archivos/Abrir archivos

Nodo i 3
Nombre del archivo
UID
GID
Perms
Fecha/Hora de creación
Fecha/Hora de modificación
Tamaño = 0
Apuntador a bloque = 1
Apuntador a bloque = 2
Apuntador a bloque = 3
Apuntador a bloque = 4
Apuntador a bloque = 0

Int vdopen (char *filename, unsigned short mode)

	En Uso	Nombre
0	1	STDIN
1	1	STDOUT
2	1	STDERR
3	0	
n	0	

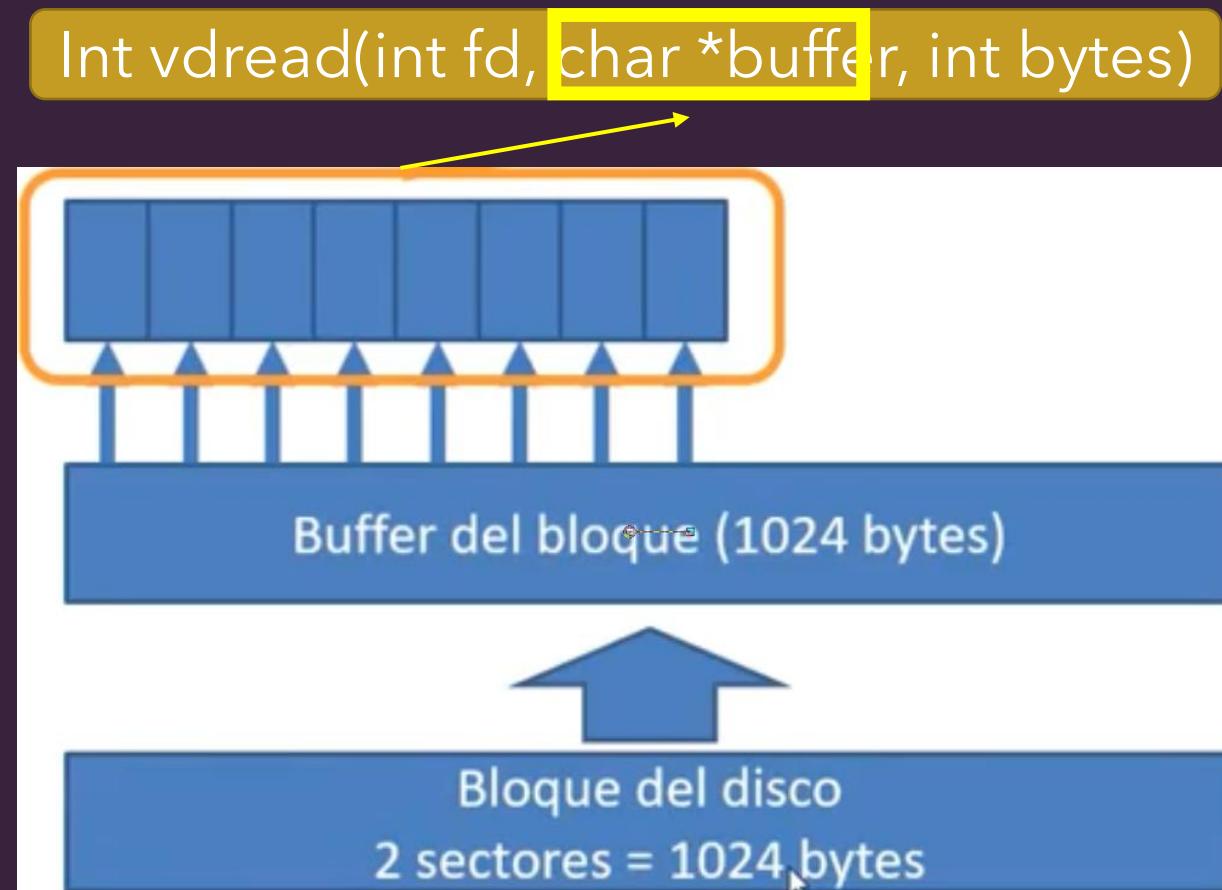


	En Uso	Nombre
0	1	STDIN
1	1	STDOUT
2	1	STDERR
3	1	Nombre del archivo
n	0	

- Buscar el nodo-i donde esté el nombre del archivo del argumento
- Abrir el nodo-i
- Transferir información del nodo-i a la tabla de archivos abiertos
- Establecer el archivo “en uso” en la tabla de archivos abiertos
- La función regresa el numero de entrada de la tabla donde está el archivo (Descriptor de archivo)

Operación con los archivos/Lectura

- Leer un bloque al buffer del bloque
- Traer los bytes solicitados al char *buffer en la función
- Cuando se terminen de leer todos los datos del buffer del bloque, leer el siguiente bloque

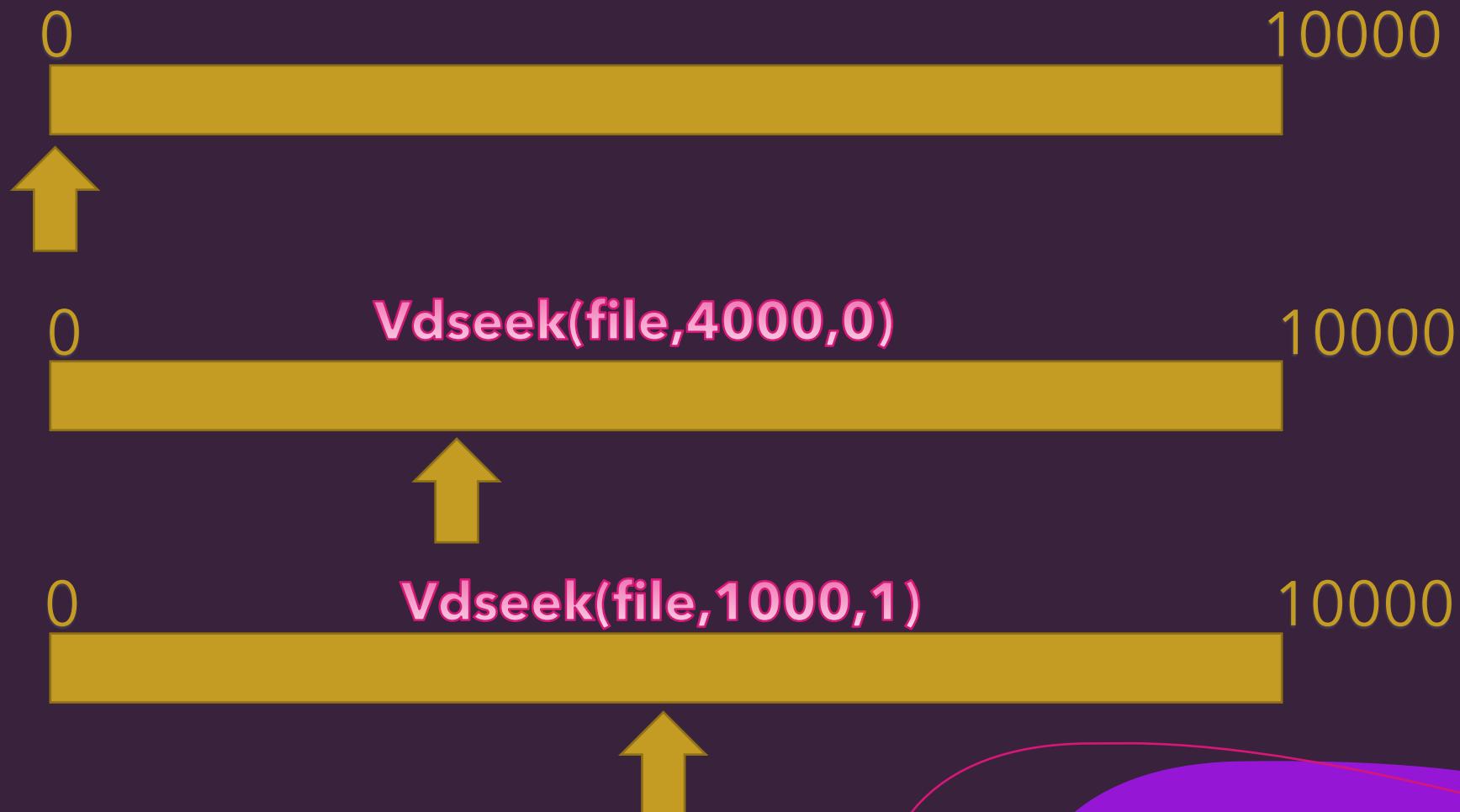


Operación con los archivos/Posicionamiento

Int vdseek(int fd, int offset, int whence)

- Para mover el apuntador del disco
- El apuntador indica a partir de que posición del archivo se va a leer o escribe
- El parámetro offset indica la cantidad de bytes a moverse
- El parámetro whence indica a partir de donde se cuenta el offset
 - Si es 0, será a partir del inicio
 - Si es 1, será a partir de la posición actual del puntero, offset podría ser negativo
 - Si es 2. será a partir del final, offset debe ser negativo

Ejemplo

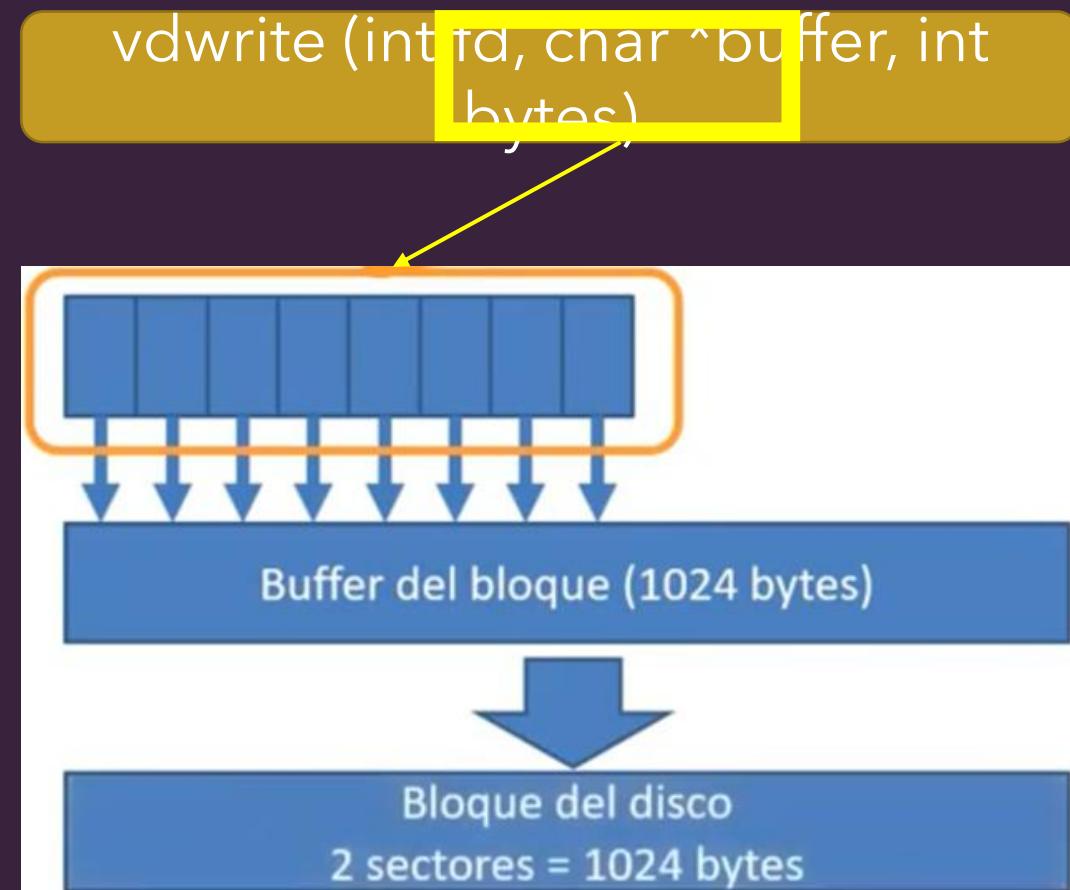


Ejemplo



Operación con los archivos/Escritura

- Copiar los bytes de `char *buffer` al buffer del bloque
- Cuando se llena el buffer del bloque, escribir en bloque del disco



Operación con los archivos/Cerrar archivos



Int vdclose(int fd)

	En Uso	Nombre
0	1	STDIN
1	1	STDOUT
2	1	STDERR
3	1	Nombre del archivo
n	0	



	En Uso	Nombre
0	1	STDIN
1	1	STDOUT
2	1	STDERR
3	0	
n	0	

- Usar el descriptor del archivo para localizarlo en la tabla de archivos abiertos
- Vaciar buffers si hay escrituras pendientes
- Poner el bit de uso en 0
- Eliminar el nombre del archivo en la tabla

Directorios/Abrir directorio

VDDIR *vdopendir (char *dirname)

- Un directorio es una tabla de nodos-i
- Abre el directorio correspondiente a dirname indicado en el argumento y pone en una tabla de directorios abiertos.
- El valor que regresa VDDIR * es un descriptor(dirdesc) que apunta a una entrada en la tabla de directorios abiertos
- El apuntador a las entradas del directorio se posiciona en la primera entrada

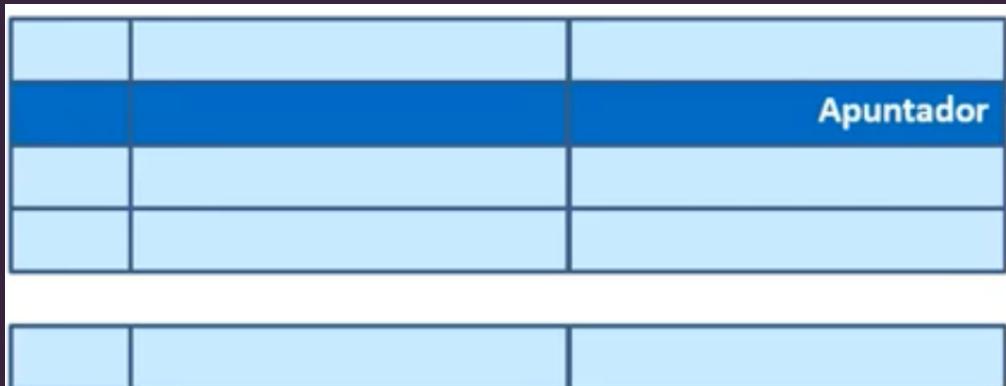


Tabla de nodos i

Nodo-i 0

Nodo-i 1

Nodo-i 2

Nodo-i 3

Nodo-i 4

Nodo-i 5

Nodo-i 6

Nodo-i 7

Nodo-i 8

Directorios/Obtener una entrada del directorio

Struct vddirent *vdreaddir (VDDIR *dirdesc)

- Lee a memoria una estructura vddirent del directorio apuntado por dirdesc
- Cuando no hay mas entradas que leer devuelve NULL

Tabla de nodos i
Nodo-i 0
Nodo-i 1
Nodo-i 2
Nodo-i 3
Nodo-i 4
Nodo-i 5
Nodo-i 6
Nodo-i 7
Nodo-i 8

Directorios/Obtener una entrada del directorio

Struct vddirent *vdreaddir (VDDIR *dirdesc)

```
while((entry=vdreaddir(dd))!=NULL)  
    printf("%s\n",entry->d_name);
```

```
    NULL  
while((entry=vdreaddir(dd))!=NULL)  
    printf("%s\n",entry->d_name);
```

Tabla de nodos i
Nodo-i 0
Nodo-i 1
Nodo-i 2
Nodo-i 3
Nodo-i 4
Nodo-i 5
Nodo-i 6
Nodo-i 7
Nodo-i 8

Directorios/Cerrar directorio

```
Int vdclosedir(VDDIR  
*dirdesc)
```

- Cierra el directorio asociado con el descriptor dirdesc
- El descriptor dirdesc ya no estará disponible después de esta llamada

De aquí en adelante no se
considera en la prueba

ALGORITMOS DE PLANIFICACIÓN

- La literatura de sistemas operativos está repleta de interesantes problemas que se han descrito y analizado ampliamente, mediante el uso de un sin numero de métodos e ideas de sincronización y planificación.
- Cuando una computadora se multiprograma, con frecuencia tiene varios procesos o hilos que compiten por la CPU al mismo tiempo. Esta situación ocurre cada vez que dos o más de estos procesos se encuentran al mismo tiempo en el estado "listo". Si sólo se dispone de una CPU, hay que decidir cuál proceso se va a ejecutar.

ALGORITMOS DE PLANIFICACIÓN

- La decisión de que proceso se ejecuta (hace uso de la CPU) la realiza el planificador de procesos que implementa un algoritmo de planificación.
- Muchas de las soluciones implementadas para planificar procesos, también son usadas para planificar hilos.
- En los inicios de los S.O. en los sistemas de procesamiento por lotes, el algoritmo de planificación era simple, se ejecutaba el siguiente trabajo en la cinta. Con la llegada de los sistemas multiprogramados, el algoritmo se tuvo que modificar y ser más complejo.

PLANIFICACIÓN DE PROCESOS

- Se define un conjunto de políticas y mecanismos que se incorporan por medio de un módulo: Planificador
- Las decisiones del planificador deben realizarse sin perder de vista su principal objetivo que consiste en el máximo aprovechamiento del sistema, lo que implica proveer un buen servicio a los procesos existentes en un momento dado

OBJETIVOS DE LA PLANIFICACIÓN

- La Planificación de procesos tiene como principales objetivos:
 - Equidad: Todos los procesos deben ser atendidos.
 - Eficacia: La CPU debe estar ocupado el 100% del tiempo.
 - Tiempo de respuesta: El tiempo empleado en dar respuesta a las solicitudes del usuario debe ser el menor posible.
 - Tiempo de regreso: Reducir al mínimo el tiempo de espera de los resultados esperados por los usuarios por lotes.
 - Rendimiento: Maximizar el número de tareas que se procesan por cada hora.

ALGORITMOS DE PLANIFICACIÓN

- Existen diversos algoritmos que buscan cumplir los objetivos antes mencionados. Cada uno de estos algoritmos presenta uno o más problemas (llamados clásicos) que representan la situación a resolver.

Trabajo

Enunciado del problema:

- Cinco filósofos están sentados alrededor de una mesa y pasan su vida cenando y pensando. Cada filósofo tiene un plato de fideos y un palillo a la izquierda de su plato. Para comer los fideos son necesarios dos palillos y cada filósofo sólo puede tomar el palillo que está a su izquierda y el de su derecha. Si cualquier filósofo toma un palillo y el otro está ocupado, se quedará esperando, con el palillo en la mano, hasta que pueda tomar el otro palillo, para luego empezar a comer. El resto de filósofos que no está ni comiendo ni con un palillo en la mano está pensando.
- El problema consiste en inventar un algoritmo que permita comer a los filósofos.

Información sobre el problema:

- El problema de la cena de los filósofos o problema de los filósofos cenando (dining philosophers problem) es un problema clásico de las ciencias de la computación propuesto por Edsger Dijkstra en 1965 para representar el problema de la sincronización de procesos en un sistema operativo. Se trata de lanzar cinco procesos (filósofos) y ponerles a competir por obtener unos recursos. La solución consiste en configurar para que estos procesos (filósofos) puedan acceder a los recursos (dos palillos) y desarrollar el trabajo (puedan comer). El problema es que el algoritmo de solución debe ser justo y no debe permitir que uno de ellos ocupe todo el sistema y no deje comer a los demás o que entre ellos se bloqueen y ninguno pueda tener acceso paralizando todo el trabajo.

Información adicional al problema:

- Lógicamente si son cinco filósofos y hay cinco palillos el algoritmo debe permitir que la mayor parte del tiempo haya dos filósofos comiendo y uno pensando. Pero además se deben tomar en cuenta diferentes factores:
- Si dos filósofos adyacentes intentan tomar el mismo palillo a la vez, se produce una condición de carrera: ambos compiten por tomar el mismo palillo, y uno de ellos se queda sin comer.
- Si todos los filósofos toman el palillo que está a su derecha al mismo tiempo, entonces todos se quedarán esperando eternamente, porque alguien debe liberar el palillo que les falta. Nadie lo hará porque todos se encuentran en la misma situación (esperando que alguno deje sus palillos). Entonces los filósofos se morirán de hambre. Este bloqueo mutuo se denomina interbloqueo o deadlock.
- Resumiendo: El problema consiste en encontrar un algoritmo que permita que los filósofos nunca se mueran de hambre y que el comedor esté dando de comer constantemente y de una manera lo más eficiente posible.

Cena de los filósofos

