El sistema de entrada/salida

Organización en capas

- Decodificación
- Sistema de archivos
- Cache de disco
- Scheduler de accesos a disco
- Driver
- Disco o SSD

Capa: Decodificación

- La API de Unix uniformiza la E/S para archivos, dispositivos y otras abstracciones por medio de *open*, *read*, *write, close*, etc.
- El procesamiento interno es muy distinto según la naturaleza de la abstracción
- Todo parte a nivel del proceso del usuario con la llamada a open:

```
int fd= open(nombre, modo);
```

- La implementación de esta función está en el núcleo del sistema operativo
- Open retorna un entero *fd* denominado *file descriptor* que será usado por el proceso en operaciones como *read*, *write* y *close*
- Para cada proceso el núcleo mantiene un arreglo con los descriptores abiertos
- *fd* se usa para indexar el arreglo recuperando una estructura de tipo *struct file* con el estado de la abstracción
- El puntero *fops* en esta estructura es la dirección de otra estructura de tipo *file_operations* con punteros a las funciones que implementan las operaciones *read*, *write*, *close* y otras para esa abstracción
- Si el proceso invoca:

```
int rc= read(fd, buf, count);
```

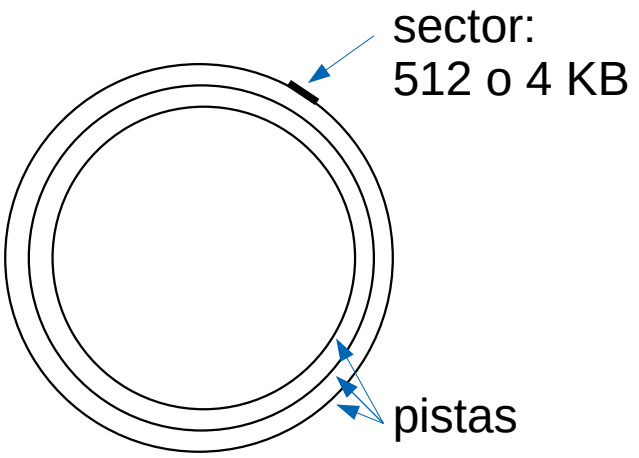
- El núcleo decodifica qué función se encarga de esta lectura con:

```
struct file *fil=
process→fd_table[fd];

(*fil→fops→read)(fil,
buf, count,
&fil→f_pos);
```

Capa: El disco

- Un drive contiene 2 o más platos que almacenan los bits en pistas que se graban magnéticamente
- Cada plato se graba por ambos lados
- El brazo contiene cabezales para cada lado de todos los platos
- El cabezal es microscópico, mientras más pequeño mayor es la densidad de bits del disco
- El drive está sellado porque cualquier grano de polvo dañaría el cabezal
- Cuando el disco gira, el cabezal vuela sobre la superficie sin tocarlo
- Un típico drive almacena 2 TB en 2 platos que se pueden leer a unos 100 MB/segundo
- El problema es el acceso directo: mover el cabezal a otra pista del disco toma en promedio 10 milisegundos (se alcanzan apenas 100 operaciones por segundo)



Capa: Sistema de archivos

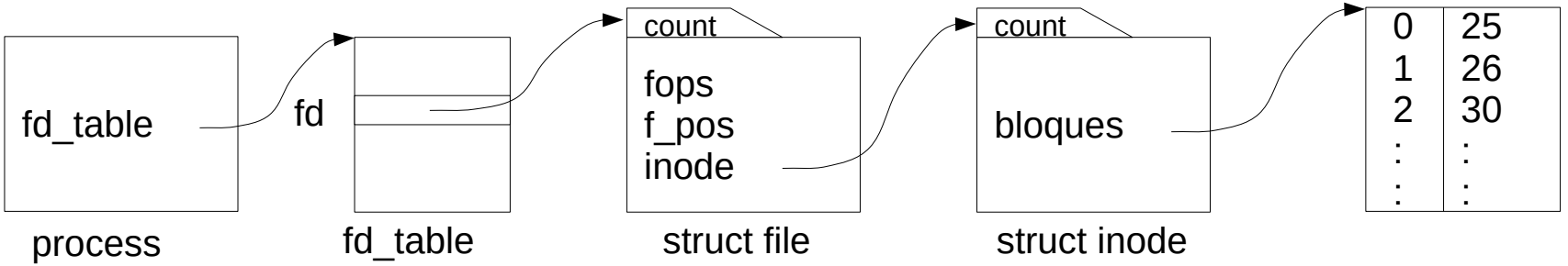
- Para el caso de apertura de dispositivos como /dev/mouse, todo el proceso lo hace el driver
- Para el caso de apertura de un archivo el acceso se descompone en capas adicionales:
  - sistema de archivos
  - cache de disco
  - scheduler de disco
  - driver de disco
  - disco (o ssd)

Sistema de Archivos

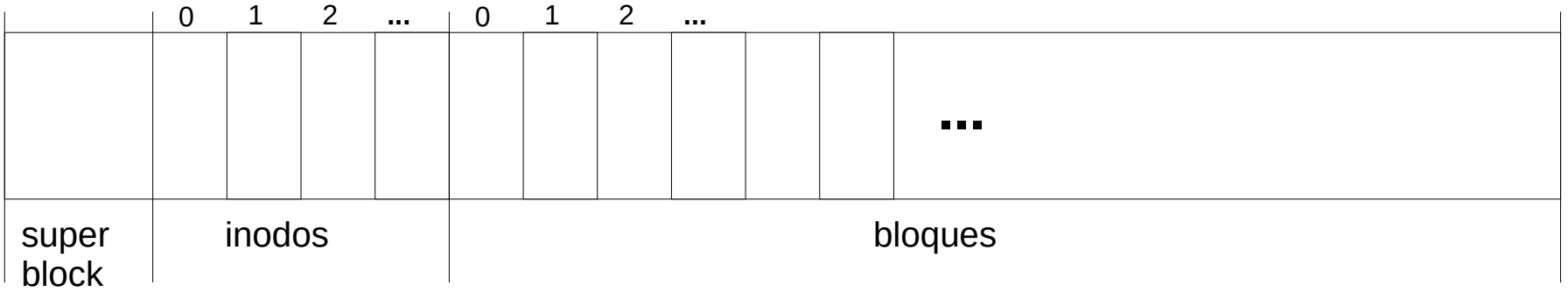
- Ve una partición del disco como un arreglo de bloques de datos
- Se encarga de darle una estructura jerárquica de directorios y archivos
- Su función más importante es determinar en qué bloques de la partición se encuentra un archivo para pedírselos al cache de disco
- Se esfuerza en atribuir a los archivos sectores consecutivos del disco para que el acceso secuencial sea rápido, reduciendo el movimiento del cabezal del disco
- Un bloque se almacena en 1 o más sectores del disco

Estructura del sistema de archivos

- Diagrama de acceso a un archivo:



- Formato de una partición:



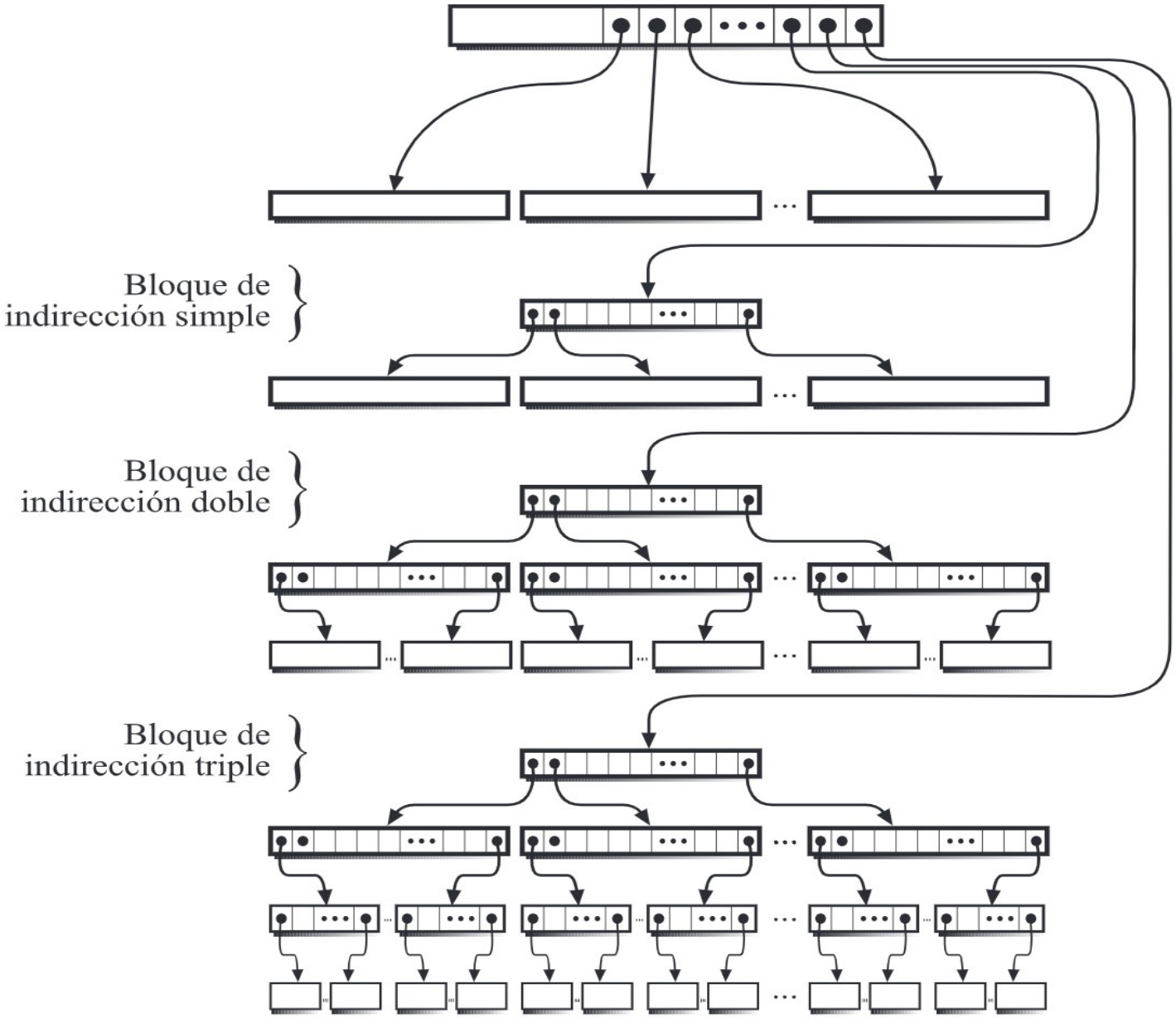
El super block contiene:

- tamaño de los bloques
- nº de inodos
- nº de bloques
- etc.

Un inodo representa un archivo y contiene:

- largo del archivo, permisos
- nº de usuario, nº de grupo
- tiempos de creación, modificación y consulta (miliseg. desde el 1/1/1970)
- nº de links duros
- tipo: normal, dir., link simb., dispositivo, etc.
- 12 punteros a bloques de datos
- 1 puntero a un bloque de indirección simple
- 1 puntero a un bloque de indirección doble
- 1 puntero a un bloque de indirección triple

Diagrama de bloques de un archivo



Formato de directorios

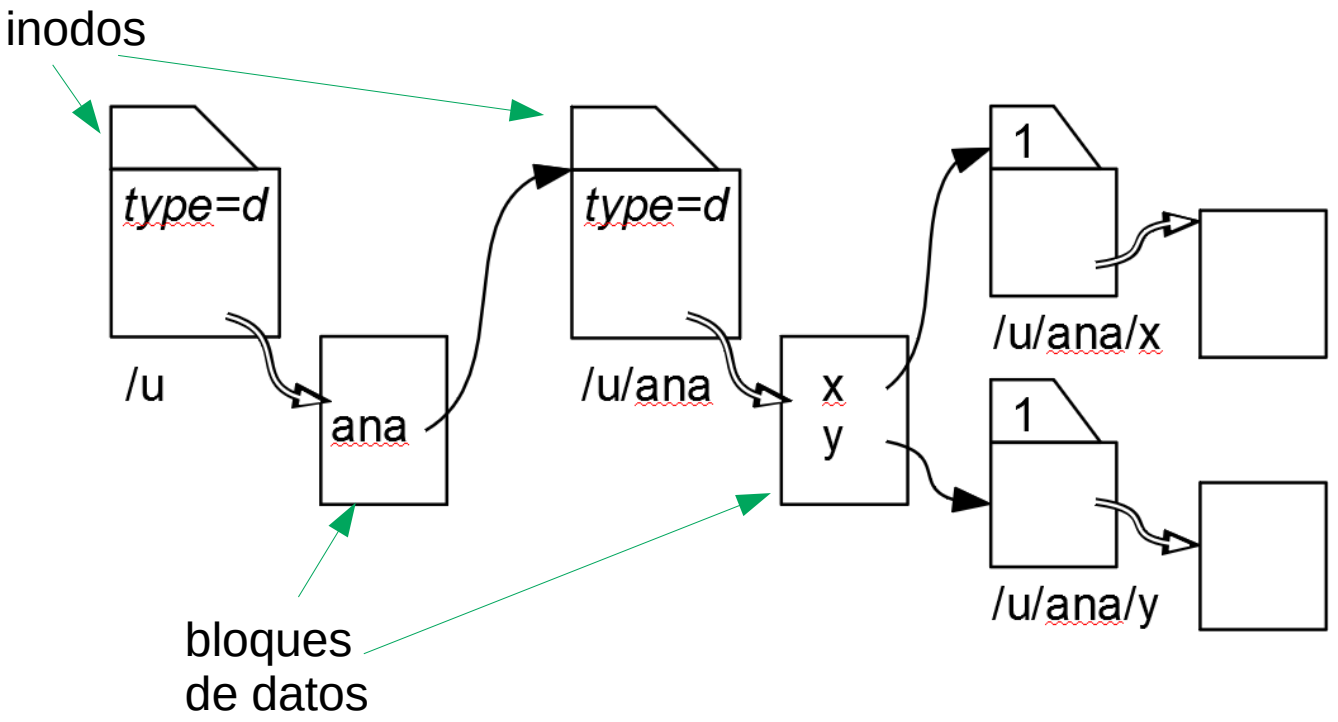
- Un directorio ocupa un inodo y bloques de datos como los archivos normales
- Por cada archivo perteneciente a ese directorio se almacena una fila de largo variable
- El primer byte indica el largo del nombre del archivo en bytes
- Luego viene el nombre
- Finalmente 2 bytes que indican el número del inodo que representa ese archivo
- El nombre se almacena en el directorio, no en el inodo
- Por ejemplo si un directorio contiene los archivos *dat.txt* y *hello*:

1	.	inodo
2	. .	inodo’
7	d a t . t x t	inodo’’
5	h e l l o	inodo’’’

- Un directorio siempre contiene un link duro a sí mismo (.) y un link duro del directorio padre (..)

Ejemplo de diagrama de inodos, directorios y archivos

- La partición */u* contiene el directorio */u/ana* que a su vez contiene los archivos *x* e *y*
- El siguiente diagrama muestra inodos, archivos y directorios



Capa: caché de disco

- Almacena los bloques recientemente usados de una partición
- Ocupa típicamente el 30% de la RAM del computador
- Si se lee un bloque y no se encuentra en el caché se le pide al scheduler de disco que lo lea y se almacena en el caché reemplazando algún otro bloque
- Lee más bloques que los solicitados porque es probable que sí se soliciten pronto (*read-ahead*)
- Cuando se escribe, se escribe primero en el caché y se lleva a disco más tarde (*write-after*)
- Un proceso *daemon* llamado update lleva a disco cada 30 segundas todas las escrituras pendientes
- También se puede hacer explícitamente invocando la llamada a sistema *sync* o el comando *sync*

Capa: driver del disco

- Implementa una API estándar para acceder a todos los tipos de disco: M.2, SATA, ATA, SCSI, etc.
- Ve el disco como un arreglo de sectores de 512 bytes (o 4096 bytes en discos de más de 2 TB)
- Accede directamente a los puertos de entrada/salida de la interfaz del disco para ejecutar comandos de lectura o escritura de *n* sectores a partir del *k*-ésimo sector