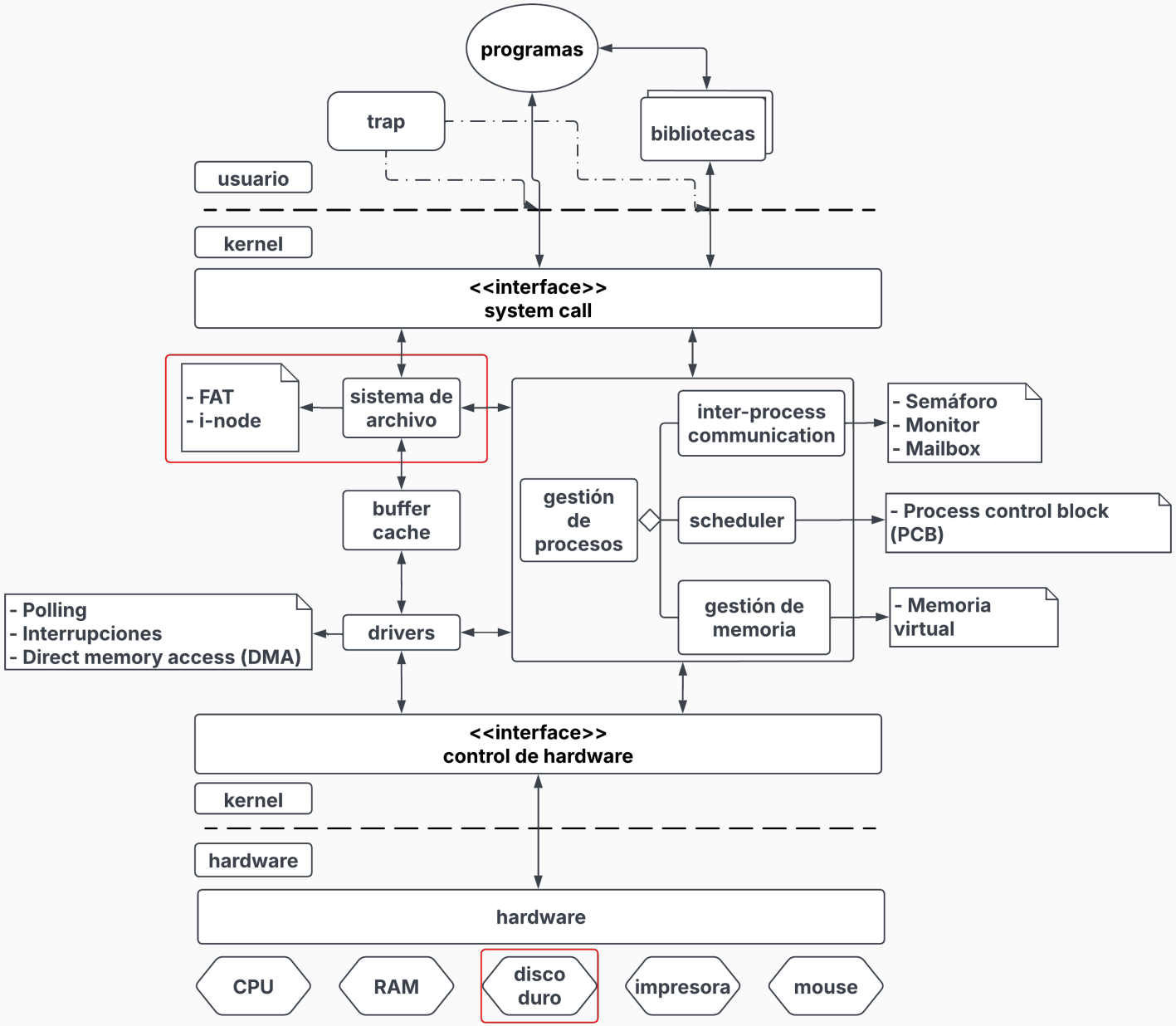
Sistemas Operativos

Sistema de Archivos I

Curso 2025

Facultad de Ingeniería, UDELAR

Arquitectura del sistema operativo



Agenda

1. Interfaz del sistema de archivos
   - Archivos
   - Directorios
   - Seguridad
2. Implementación del sistema de archivos
   - Diseño
   - Organización
   - Estructura
   - Métodos de asignación
   - Administración del espacio libre
3. Ejemplo de implementación

Interfaz del sistema de archivos

Archivos

- Los dispositivos masivos (discos duros, cintas, etc.) permiten guardar información de forma no volátil.
- El sistema operativo se abstrae de las propiedades físicas de los dispositivos para definir una unidad lógica de almacenamiento: el archivo.
- Los archivos son *mapeados* por el sistema a los distintos dispositivos.
- Un archivo es una agrupación de información que es guardada en algún dispositivo no volátil.
- Desde la perspectiva del usuario, es la unidad mínima de almacenamiento que el sistema le provee.

Archivos

- Los archivos poseen atributos:
  - Nombre: un nombre simbólico que permite identificar el archivo.
  - Identificador: símbolo que lo identifica de forma única a nivel global del sistema de archivos. Usualmente es un número.
  - Tipo: programa ejecutable, archivo de datos, etc.

Archivos

- Los archivos poseen atributos:
  - Ubicación: puntero al dispositivo y lugar donde reside el archivo.
  - Tamaño: tamaño actual del archivo (en bytes, palabras o bloques).
  - Protección: información de control para el acceso al archivo. Ej.: usuarios que pueden acceder, en que forma, etc.
  - Información de conteo: fecha de creación, último acceso, etc.

Operaciones sobre los archivos

- El sistema operativo brinda servicios para la manipulación de archivos:
  - Crear y abrir: creación de un archivo en el sistema de archivos. Se debe especificar un nombre para el nuevo archivo. Además, se provee la apertura de una archivo ya existente para acceder o modificar su información.
  - Escribir: escribir información en un archivo previamente abierto.
  - Leer: leer información en un archivo previamente abierto.
  - Reposicionar dentro de un archivo: acceder a cualquier parte del archivo.
  - Eliminar: destruir el archivo a nivel del sistema de archivo.
  - Truncar: eliminar parte de la información del archivo.

Operaciones sobre los archivos

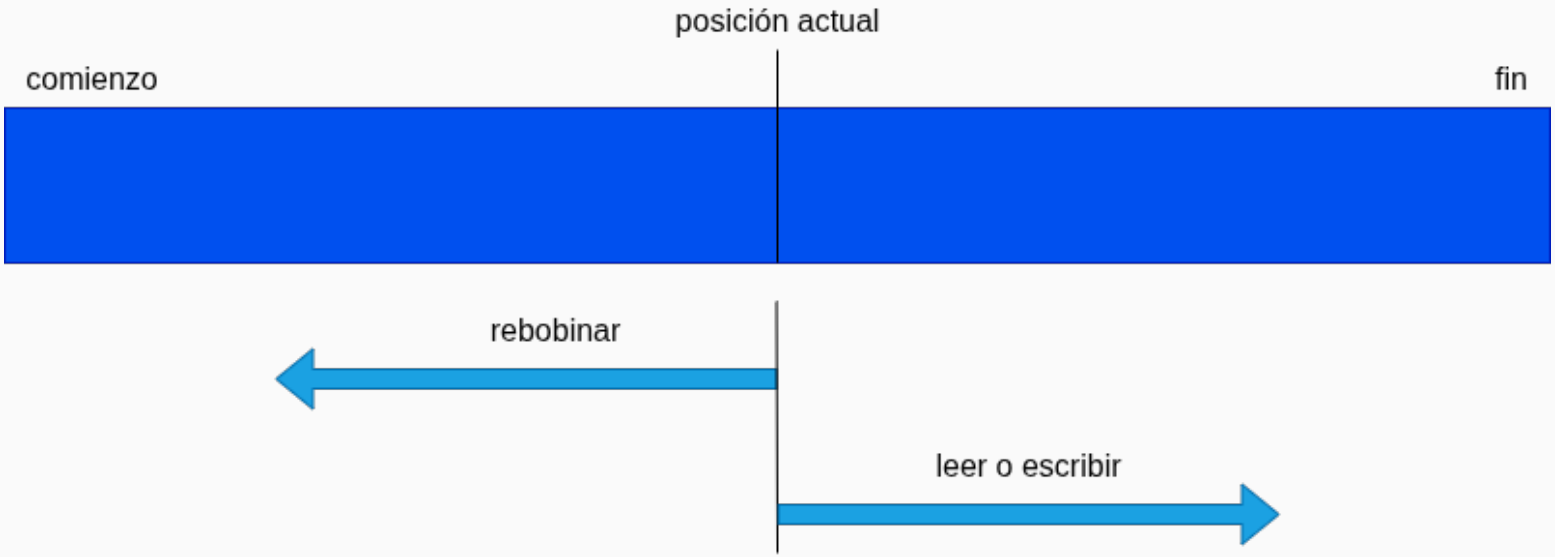
- Por lo general, los sistemas mantienen una tabla de archivos abiertos por proceso.
- Estos archivos se abren a través de un llamado al sistema y, de esa forma, se puede operar con ellos (leer, escribir, etc.).
- Finalmente, el archivo es cerrado antes que finalice la ejecución del proceso.
- Tener un archivo abierto para el sistema implica mantener una estructura que tenga por lo menos:
    - puntero de archivo (*file pointer*) para operaciones de lectura y escritura
    - ubicación del archivo en el dispositivo
    - permisos de acceso

Operaciones sobre los archivos

- Algunos sistemas proveen acceso exclusivo a un archivo (lock) por parte de los procesos.
- A su vez, algunos sistemas implementan el mapeo de archivos al espacio de memoria virtual de un proceso. De esta forma, no es necesario realizar read y write para operar sobre el archivo, sino que se puede acceder directamente a memoria principal. Esto trae el beneficio de no hacer el llamado a sistema para operar sobre el archivo.

Métodos de acceso: Secuencial

- La información es accedida en orden, registro a registro.
- El concepto de registro depende del tipo de archivo (en texto sería un caracter).
- Las operaciones de lectura acceden a la información en forma secuencial e incrementando el puntero de archivo.
- Basado en el modelo de cinta, y funciona en dispositivos secuenciales y de acceso directo.

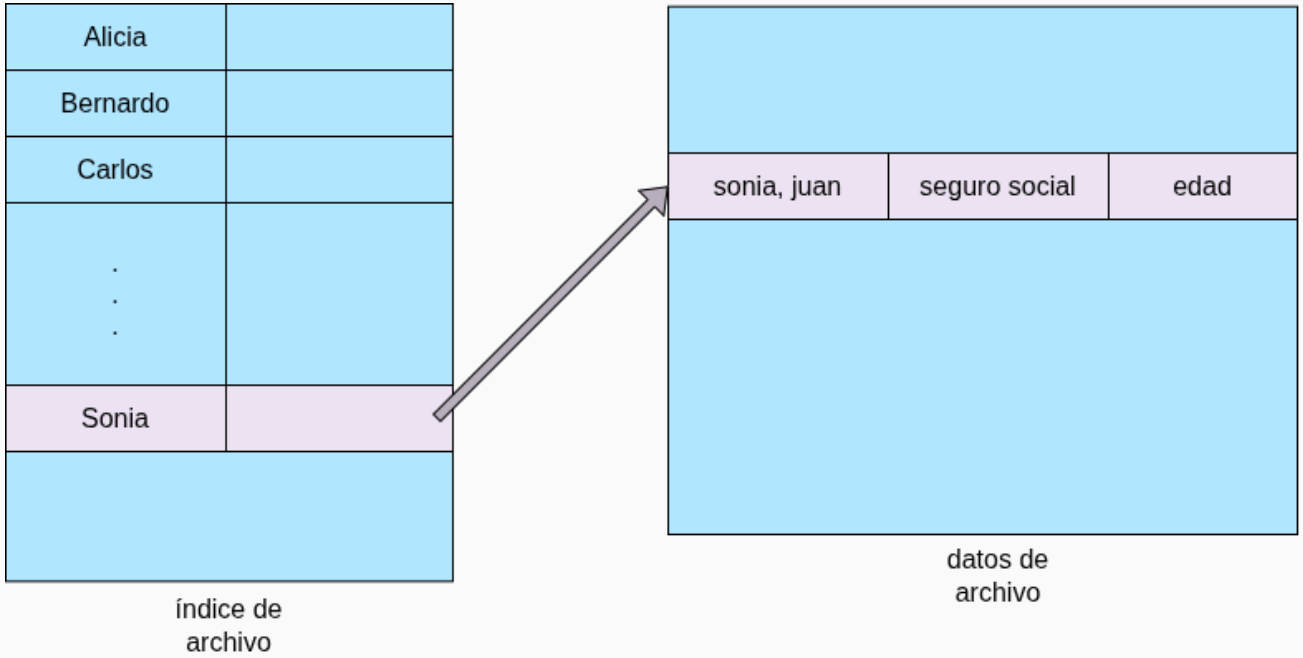


Métodos de acceso: Directo

- El archivo es presentado como una secuencia numerada de registros.
- La información es accedida libremente. No existe restricciones sobre el orden de escritura y lectura del archivo.
- Es basado en el modelo de disco duro, en donde los archivos están estructurados en bloques.

Directorios

- El sistema de archivos se encuentra, por lo general, estructurado en directorios que contienen archivos.
- Los directorios permiten a los usuarios del sistema tener una organización lógica del sistema de archivo.
- Un directorio puede verse como una tabla de símbolos que mapea nombres a bloques que forman un archivo.

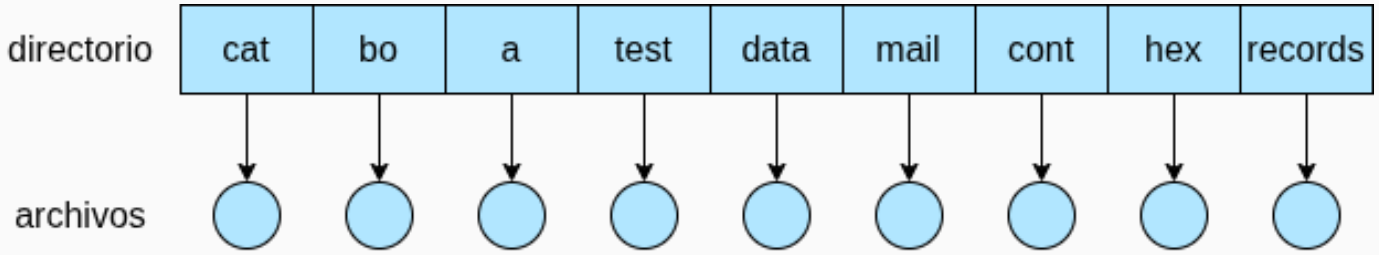


Operaciones sobre los directorios

- Búsqueda: es necesario poder buscar un archivo en un directorio
- Crear un archivo: archivos nuevos deben ser creados e incorporados al directorio.
- Eliminar: borrar un archivo del directorio.
- Listar: visualizar los archivos que están en un directorio.
- Renombrar un archivo: cambiar el nombre de un archivo dentro del directorio.
- Permitir la navegación: acceder a todos los directorios del sistema de archivos.

Estructura de directorios – Nivel único

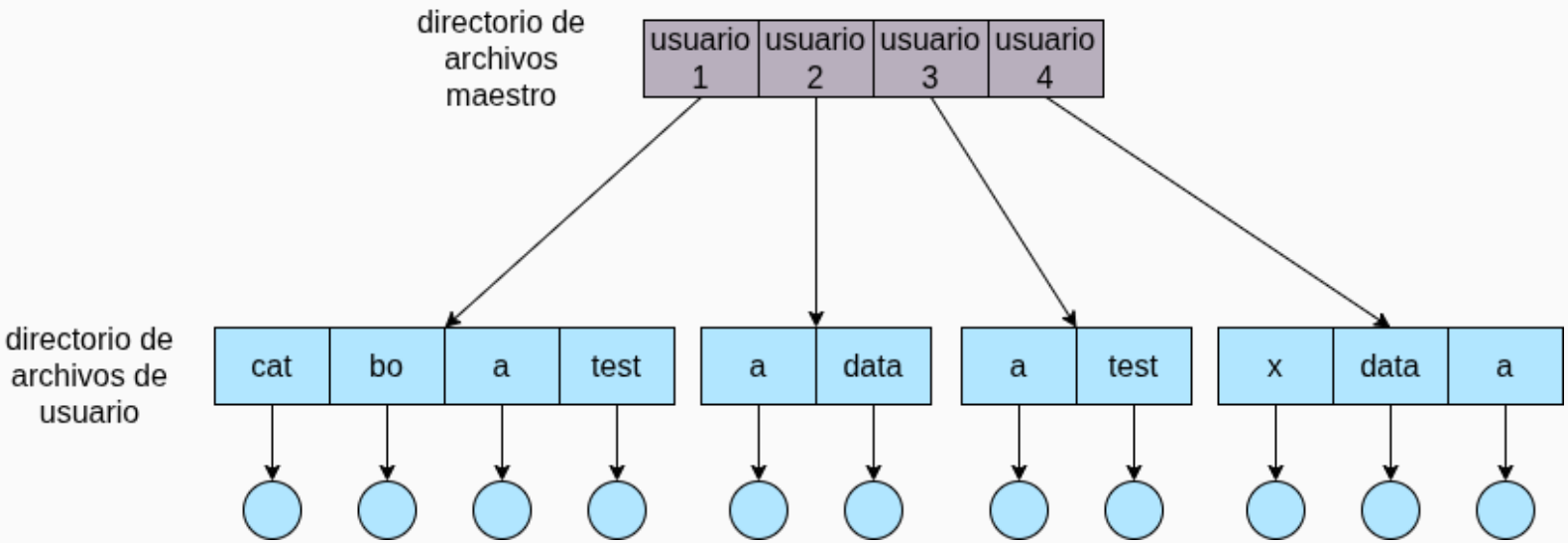
- El esquema más sencillo es tener un único nivel de directorios en el sistema de archivos.
- Todos los archivos estan contenidos en el mismo directorio.



- A medida que el sistema crece trae limitaciones. Por ejemplo, los archivos deben tener nombre único.

Estructura de directorios – Dos niveles

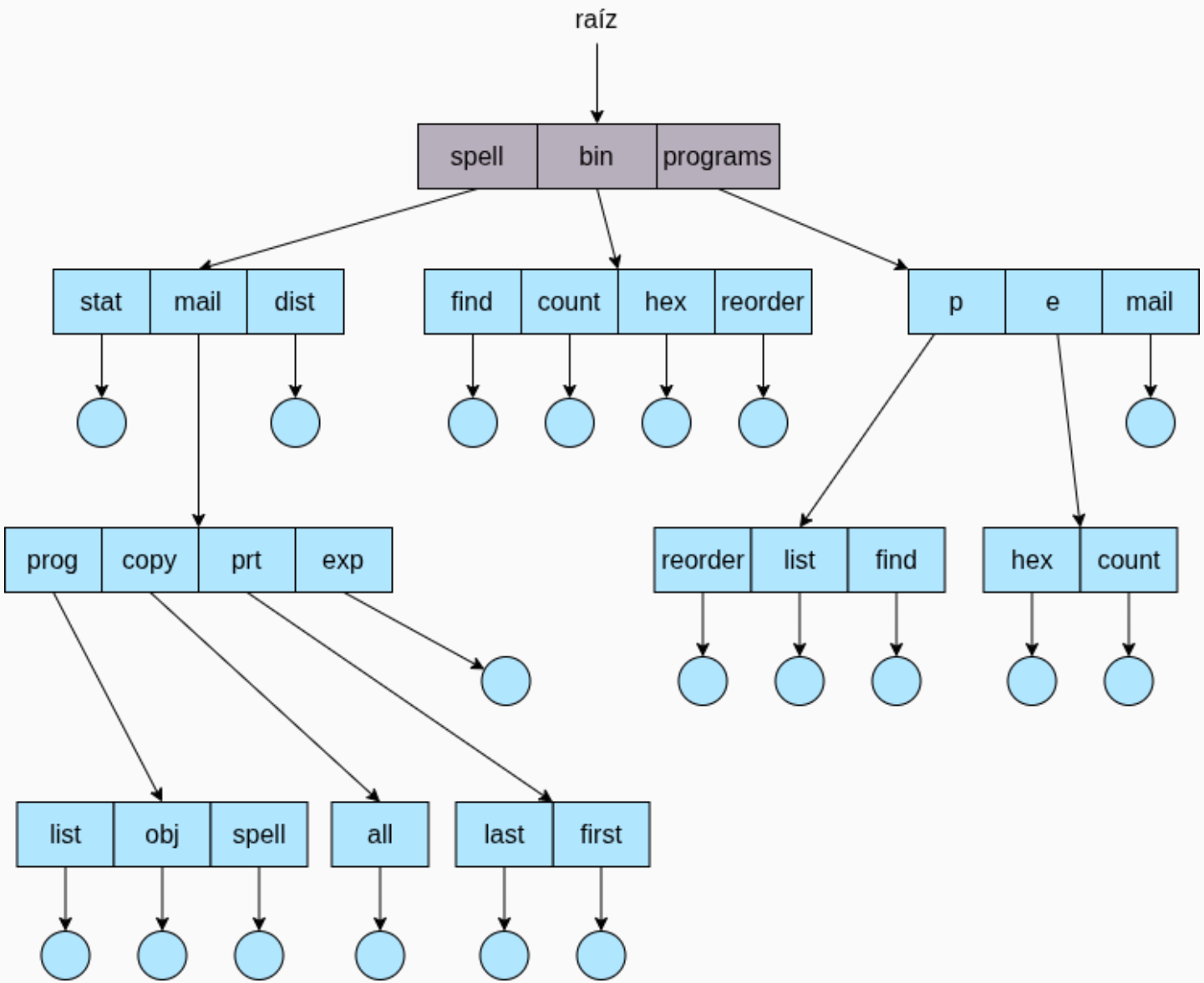
- Tener un único directorio genera confusión entre diferentes usaurios por los nombres de archivos.
- La solución estándar es tener un directorio independiente para cada usuario.



Estructura de directorios – Árbol

- Un mejor enfoque es permitir varios niveles de directorios.
- Se logra teniendo directorios (o subdirectorio) dentro de un directorio.
- Se genera una estructura jerárquica de directorios en forma de árbol.
- Se denomina ruta absoluta de un archivo, al camino desde la raíz hasta el archivo.

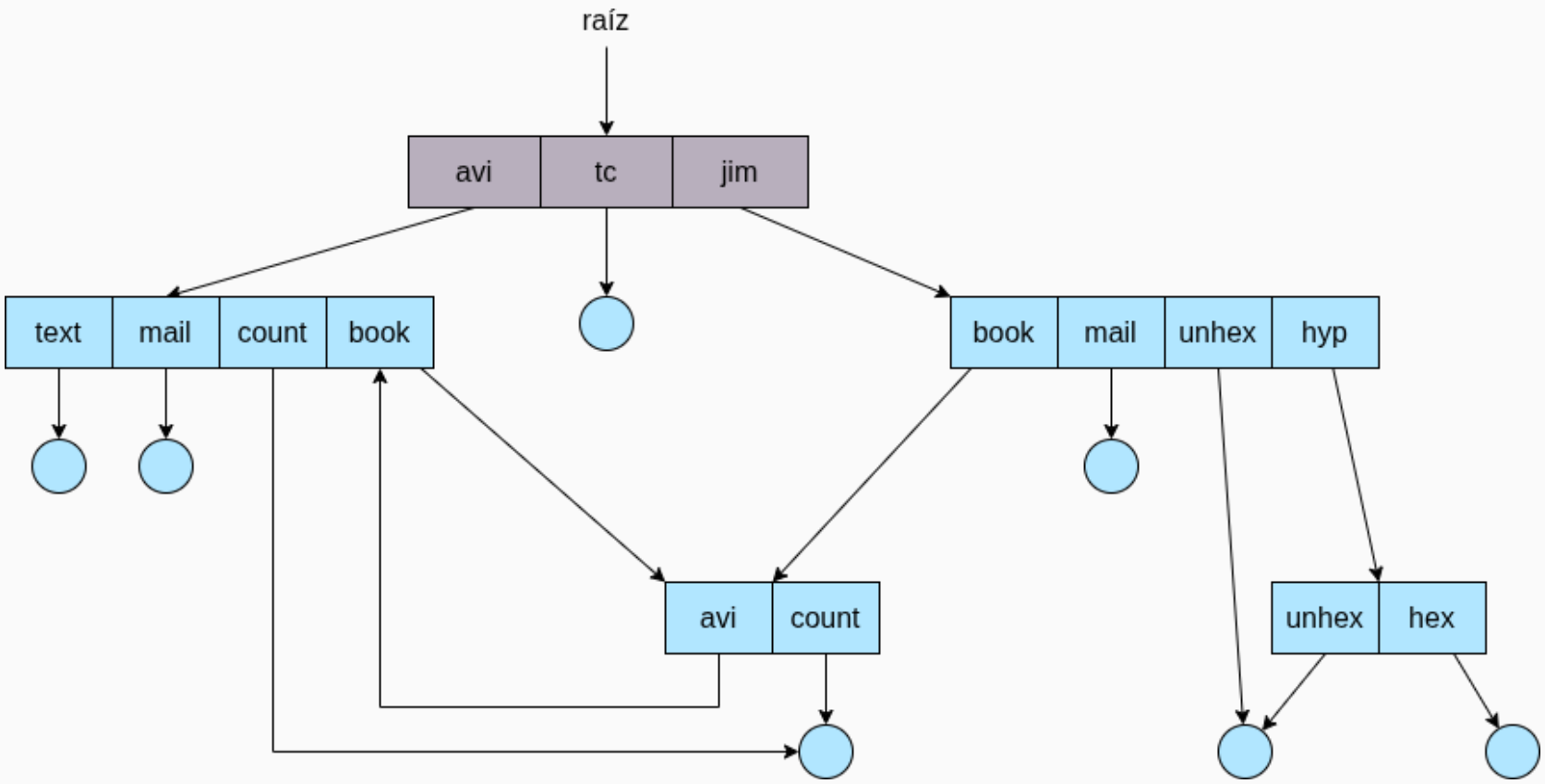
Estructura de directorios – Árbol



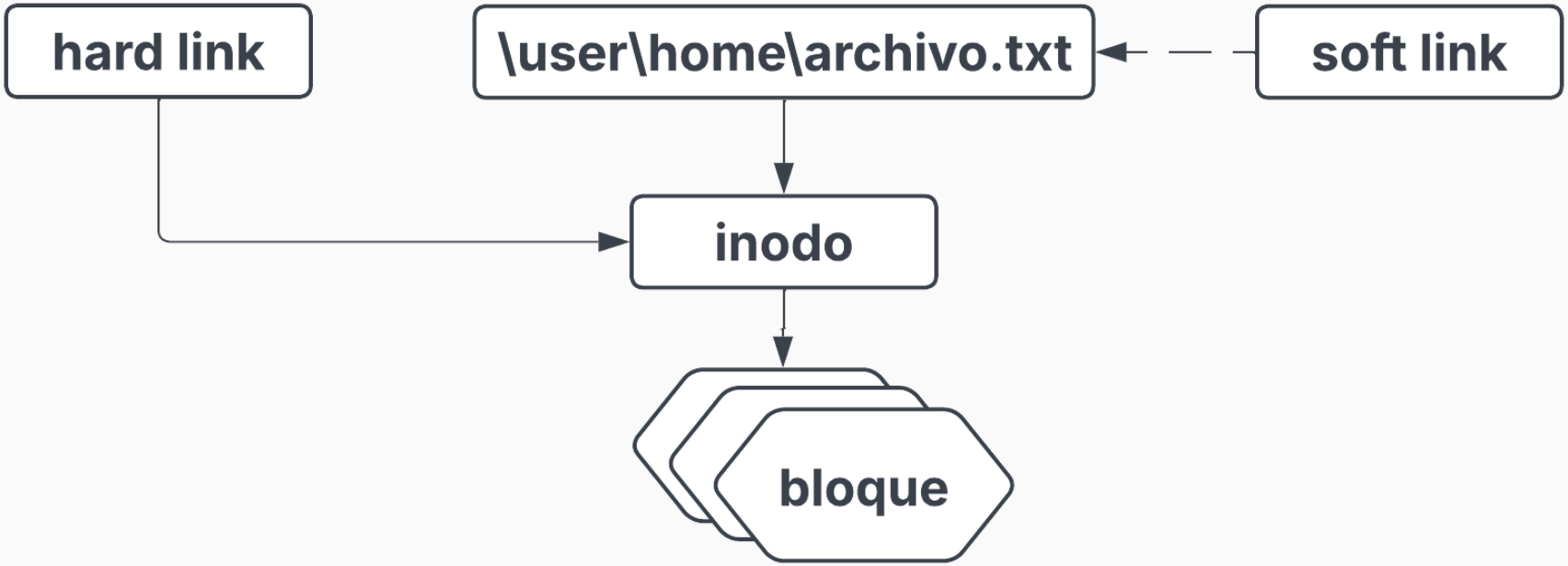
Estructura de directorios – Grafo

- La estructura anterior se potencia con un esquema de grafo mediante enlaces (*links*).
- Dos estrategias:
  - Enlace simbólico a un archivo o directorio mediante archivos de enlace (soft link).
  - Duplicar la información de un archivo en diferentes directorios (hard link).

Estructura de directorios – Grafo



Estructura de directorios – Grafo



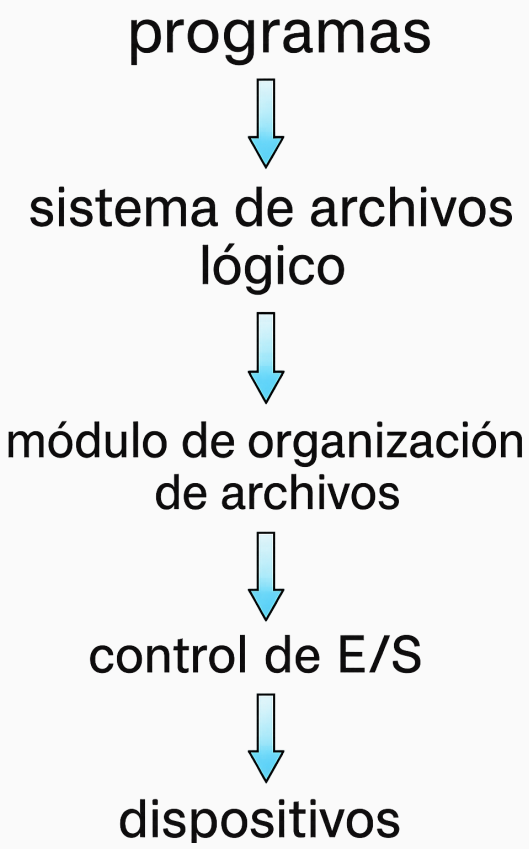
Seguridad

- En un sistema multiusuario es necesario proteger la información de cada usuario.
- En muchos casos los usuarios se agrupan según el uso que tienen sobre un sistema.
- Se definen permisos sobre los archivos tanto a nivel de usuario como de grupo.
- Los permisos más comunes son de:
    - Escritura
    - Lectura
    - Ejecución
    - Eliminar
    - Listar

Implementación del sistema de archivos

Diseño

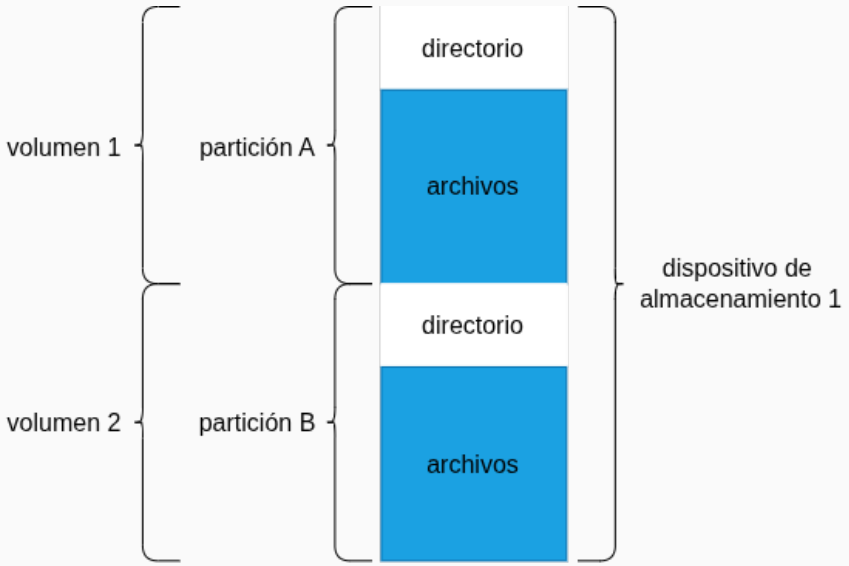
- El diseño de un sistema de archivos enfrenta dos problemas:
  - Visión del usuario: implica definir los atributos y las operaciones válidas de los archivos, y la estructura de directorios para la organización de los archivos.
  - La creación de algoritmos y estructura de datos para hacer corresponder el sistemas de archivos lógico con los dispositivos físicos de datos.
- El sistema de archivos está compuesto de varias capas. Cada una de las cuales utiliza la funcionalidad de la capa inferior.

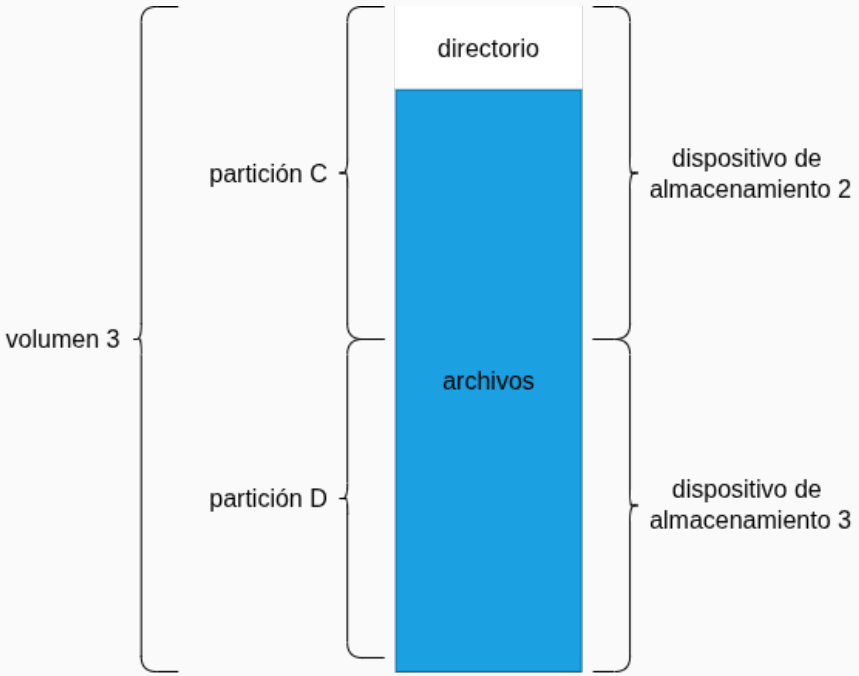


Organización

- Un sistema puede tener múltiples dispositivos físicos de almacenamiento.
- Cada dispositivo puede estar divididos en particiones.
- La particiones contienen volúmenes que pueden extenderse por varias particiones.
- Los volúmenes contienen los directorios y archivos.

Organización





Montado del sistemas de archivos

- Un sistema de archivos debe ser montado para poder ser usado.
- La estructura de directorio del sistema puede estar compuesta por múltiples volúmenes.
- La operación de montado es simple: recibe un volumen y un directorio (típicamente vacío) como argumentos.
- El volumen raíz es seleccionado montado al momento de inicio del sistema y típicamente contiene el kernel.

Estructura: en dispositivo

- Los dispositivos físicos contienen las estructuras:
  - Bloque de control para el arranque por volumen (*boot control block*): es necesario para iniciar el sistema operativo en el volumen. Este bloque es típicamente el primer bloque del volumen y puede estar vacío.
  - Bloque de control de volumen por volumen (*volume control block*): contiene información del volumen como cantidad de bloques, tamaño del bloque, bloques usados y libres, etc.
  - Estructura de directorios por sistema de archivos: para la organización de los archivos, incluye información de directorios, nombres de archivos, punteros al archivo, etc.
  - Bloque de control del archivo por archivo (*file control block (FCB)*): contiene los detalles del archivo, en particular debe indicar que bloques del disco contienen la información del archivo.

Estructura: en memoria

- El sistema operativo mantiene en memoria las estructuras:
  - La tabla de montado con información de los volúmenes montados.
  - La caché de estructura de directorio con la información de los directorios accedidos recientemente.
  - La tabla de descriptores de archivos abiertos a nivel global del sistema: contiene una copia del FCB de cada archivo abierto.
  - La tabla de descriptores de archivos abiertos por cada proceso del sistema: contiene punteros a la tabla global de archivos abiertos.

Estructura: por archivo

- Cada archivo en el sistema tiene un bloque de control.
- El bloque de control contiene varios atributos de conteo, permisos y donde están los datos del mismo:
  - Fechas (creación, acceso, modificación).
  - Usuario propietario y grupo propietario.
  - Permisos del archivo.
  - Tamaño del archivo.
  - Bloques de datos del archivo.

Estructura: por directorio

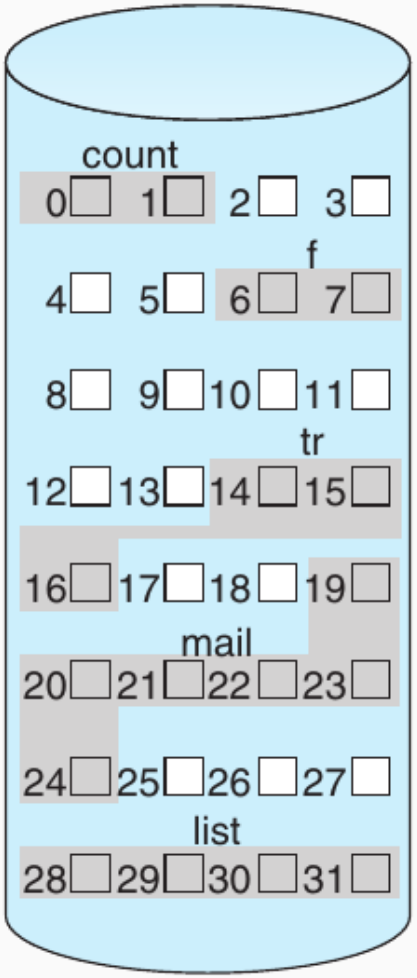
- Los directorios contienen la información de los archivos que pertenecen a él (como mínimo: nombre y referencia de ubicación).
- Para organizar la información existen varias alternativas. Por ejemplo:
  - Lista lineal: el nombre de cada archivo y un puntero sus bloques de datos son dispuestos en una lista (casi) lineal. En la búsqueda, inserción, borrado, etc., es necesario un acceso lineal.
  - Tabla de hash abierto: con el nombre del archivo se genera una clave que ayuda a identificar en que bloque se encuentra la entrada buscada. Luego la búsqueda se resuelve de forma lineal.

Métodos de asignación

- Para la organización de los datos de un archivo en disco se tienen, en general, tres métodos:
  - Asignación contigua: Los datos son dispuestos en forma contigua. Para mantener la información es necesario saber en que bloque comienzan los datos y la cantidad de bloques de datos.
  - Asignación en forma de lista: Los bloques de datos forman una lista encadenada. Es necesario una referencia al primer y último bloque de datos.
  - Asignación indexada: Se mantiene una tabla en donde cada entrada referencia a un bloque de datos.

Asignación contigua

- Sufre de fragmentación externa.
- Es necesario reubicar continuamente los archivos si crecen en tamaño.
- Se utilizan técnicas de asignación de tamaños más grandes para prever el crecimiento futuro de los archivos.

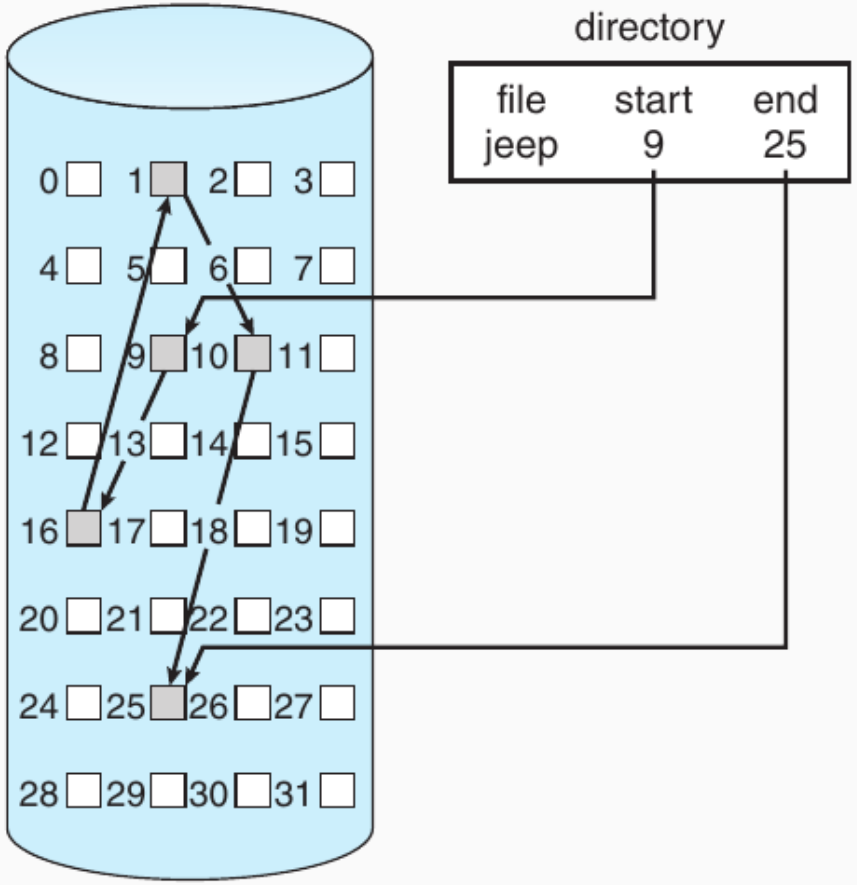


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

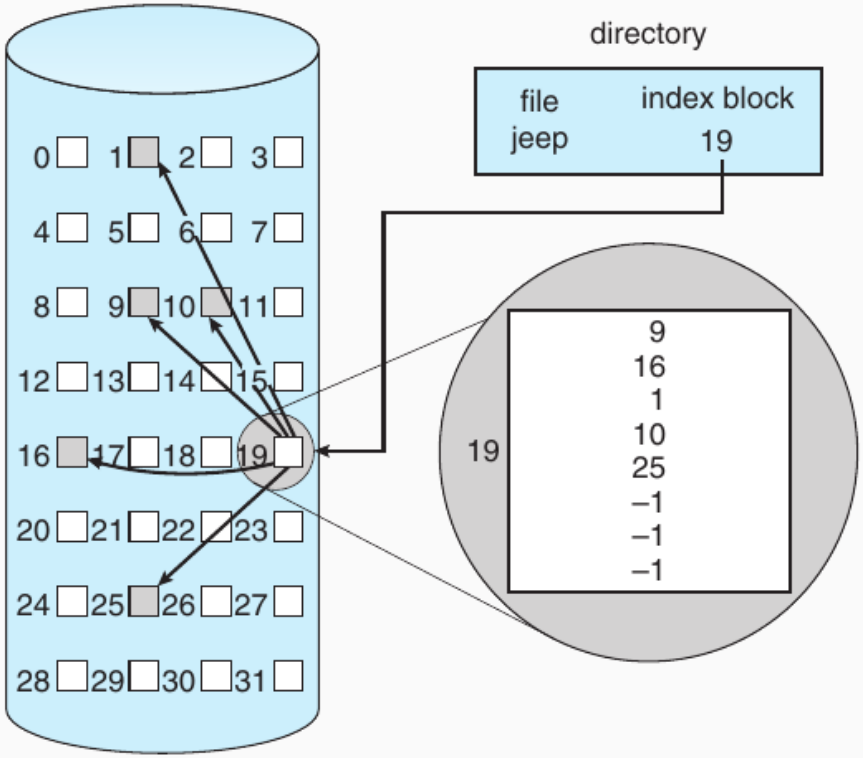
Asignación en forma de lista

- Soluciona el problema de la fragmentación externa.
- El acceso a los bloques es de orden lineal.
- Los punteros ocupan espacio en los bloques.
- La pérdida de una referencia genera la pérdida de gran parte de información del archivo.



Asignación indexada

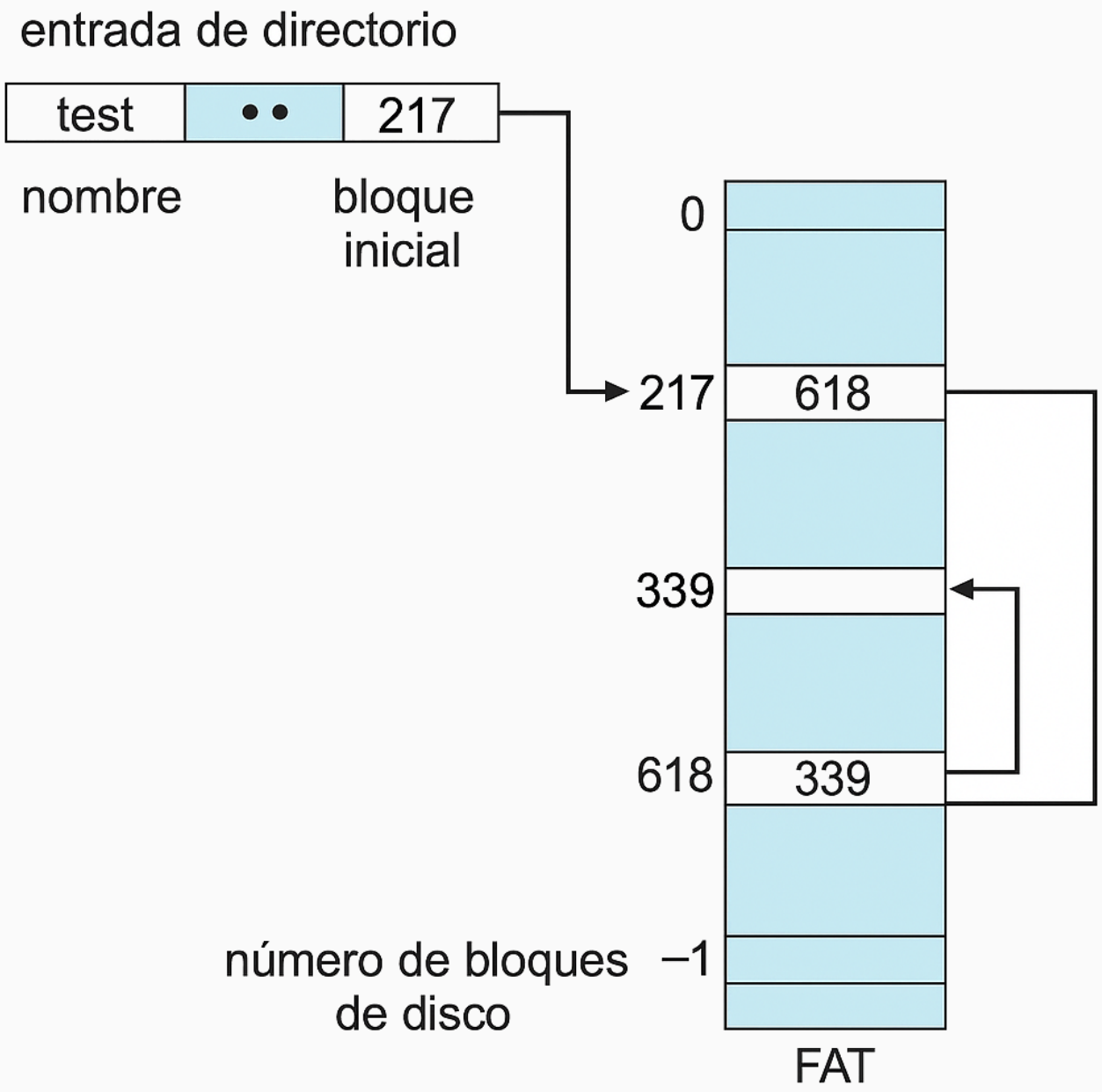
- Los bloques son accedidos directamente a través del bloque de indexación (*index block*).
- El bloque de indexación ocupa lugar por lo que se trata de que sea lo más pequeño posible.
- Una estrategia utilizada es la indexación en varios niveles. Algunos índices hacen referencia a bloques de datos (directos) y otros a bloques con referencias a bloques de datos (indirectos).



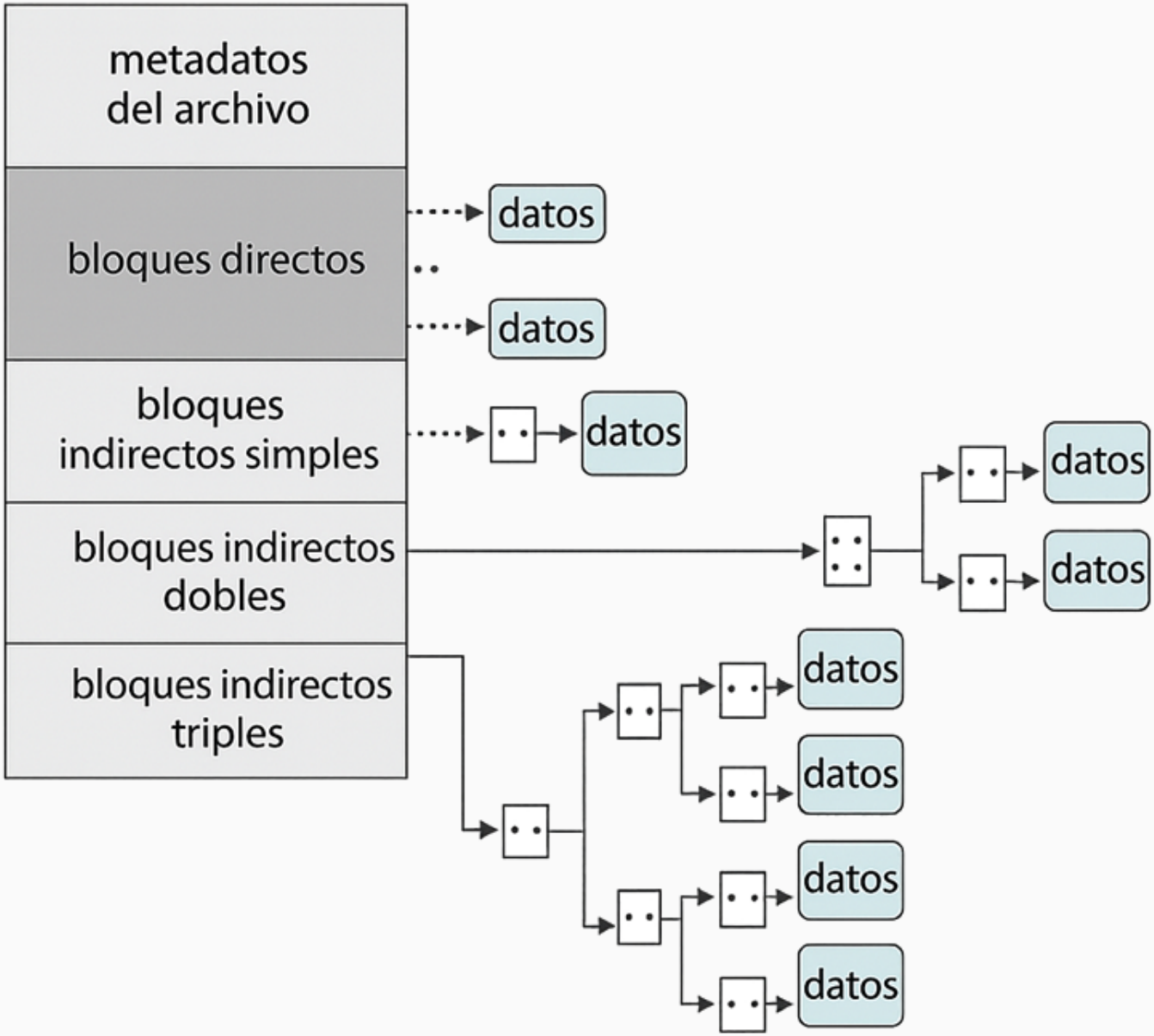
Asignación en forma de lista: Ejemplo FAT

- Tiene una tabla de asignación de archivos (*File Allocation Table*) al comienzo de cada volumen.
- Esta tabla contiene la lista de bloques de cada archivo.
- Tiene una entrada por cada bloque de disco, y es indexada por el número de bloque.

Asignación en forma de lista: Ejemplo FAT



Asignación indexada: Ejemplo UNIX



Administración del espacio libre

- En el sistema de archivos es necesario mantener que bloques están ocupados y cuales están libres.
- Alternativas posibles para la administración de los bloques:
  - Vector de bits, mapa de bits: se dispone de un bit para cada bloque del dispositivo, que representa si está ocupado o libre.
  - Lista de bloques libres: Se mantiene una lista encadenada con los bloques libres a través de los bloques. Es necesario una referencia al primer bloque.
  - Agrupación: es una variación de la lista encadenada. En cada bloque de la lista se contiene un grupo de bloques libres.
  - Conteo: se mantiene una lista en donde cada bloque contiene información de cuantos bloques contiguos, a partir de él, están libres.

Ejemplo de implementación

Problema

Un sistema de archivos usa una estrategia indexada de dos niveles con 7 bloques directos en el primer nivel y 1 bloque de indirección simple en el segundo nivel, y un mapa de bits para administrar el espacio libre del disco.

Las variables IT, MB, y D son globales, el inodo número 0 es el directorio raíz y las entradas de los directorios son almacenadas utilizando un array de entradas.

Estructura 1/2

```
const MAX_BLOQUES = 65536; // 2^16
const TAM_BLOQUE = 4096;   // 2^12
const MAX_INODOS = 8192;   // 2^13

type bloque = array [0..TAM_BLOQUE-1] of byte;
type mapa_bits = array [0..MAX_BLOQUES-1] of bit;

type entrada_dir = Record
  usado: bool;                    // 1 bit
  nombre: array [0..123] of char; // 124 bytes
  es_dir: bool;                   // 1 bit
  inodo_num: int16;               // 2 bytes
  permisos: array [0..13] of bit; // 14 bits
end; // 128 bytes
```

Estructura 2/2

```
type inodo = Record
  usado: bool;                      // 1 bit
  inodo_num: int16;                 // 2 bytes
  es_dir: bool;                     // 1 bit
  tamano: int32;                    // 4 bytes
  directo: array [0..7] of int16;   // 16 bytes
  directo_tope: int16;              // 2 bytes
  indirecto: int16;                 // 2 bytes
  indirecto_tope: int16;            // 2 bytes
  reservado: array [0..29] of bit; // 30 bits
end; // 32 bytes

type inodos_tabla = array [0..MAX_INODOS-1] of inodo;
type disco = array [0..MAX_BLOQUES-1] of bloque;

var IT: inodos_tabla;
MB: mapa_bits;
D: disco;
```

Indique el tamaño máximo que puede tener un archivo

- Tamaño máximo según el campo “tamano” del inodo: $2^{32} - 1$ bytes = 4 GB -1 bytes
- Tamaño máximo según la cantidad de bloques del disco: $65536 - 1 \times 4096 = 2^{16} \times 2^{12} = 2^{28}$ bytes
- Tamaño máximo según la cantidad de bloques asignables a un archivo:
  $(8 + (2^{12}/2^{1})) \times 2^{12} = (8 + 2^{11}) \times 2^{12} < 2^{12} \times 2^{12} = 2^{24}$ bytes
  El tamaño máximo está determinado por la menor de estas cotas. Por lo tanto es: $(8 + 2^{11}) \times 2^{12}$ bytes

Indique la cantidad máxima de archivos que puede almacenar el sistema

Para encontrar la cantidad máxima de archivos se estudia cuántos archivos de tamaño 0 es posible almacenar. Tenemos dos cotas para esto: (a) la cantidad máxima de inodos disponibles y (b) la cantidad máxima de archivos almacenables en la estructura de directorios.

(a) La cantidad máxima de inodos es $2^{13} = 8192$ y descontando el inodo raíz tenemos 8191 disponibles. Como cada archivo necesita un inodo no podemos tener más de 8191 archivos de tamaño 0.

(b) La cantidad máxima de archivos que puede contener el directorio raíz es $(8 + 2^{11}) \times 2^5$ archivos, es decir, $2^{12}/2^7 = 2^5$ `entrada_dir` por cada bloque asignado al directorio. Esto, sin siquiera contar subdirectorios, ya es significativamente mayor que la cota (a).

Entonces, la cantidad máxima de archivos de tamaño 0 es 8191 y se da cuando el directorio raíz los contiene a todos.