Sistemas de Archivos

Sistemas Operativos
DC - UBA - FCEN

16 de octubre de 2025

Menú para hoy

Menú para hoy

Repaso

Sistemas de archivos

El sistema de archivos es la parte del SO que nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.

Repaso

Sistemas de archivos

El sistema de archivos es la parte del SO que nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.

- El SO no suele interpretar el contenido de un archivo, sino que debe garantizar su almacenamiento y posterior recuperación.

El sistema de archivos es la parte del SO que nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.

- El SO no suele interpretar el contenido de un archivo, sino que debe garantizar su almacenamiento y posterior recuperación.
- Preguntas clave:

El sistema de archivos es la parte del SO que nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.

- El SO no suele interpretar el contenido de un archivo, sino que debe garantizar su almacenamiento y posterior recuperación.
- Preguntas clave:
  - ¿Qué estructuras de datos organizan el sistema de archivos?

El sistema de archivos es la parte del SO que nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.

- El SO no suele interpretar el contenido de un archivo, sino que debe garantizar su almacenamiento y posterior recuperación.
- Preguntas clave:
  - ¿Qué estructuras de datos organizan el sistema de archivos?
  - ¿Qué sucede internamente al abrir, leer o escribir un archivo?

El sistema de archivos es la parte del SO que nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.

- El SO no suele interpretar el contenido de un archivo, sino que debe garantizar su almacenamiento y posterior recuperación.
- Preguntas clave:
  - ¿Qué estructuras de datos organizan el sistema de archivos?
  - ¿Qué sucede internamente al abrir, leer o escribir un archivo?
  - ¿Qué estructuras se leen y/o escriben durante cada operación?

Repaso

Almacenamiento secundario

Las lecturas y escrituras a un medio de almacenamiento se hacen en unidades lógicas llamadas bloques.

Las lecturas y escrituras a un medio de almacenamiento se hacen en unidades lógicas llamadas bloques.

- Cada bloque comprende uno o más sectores.

Las lecturas y escrituras a un medio de almacenamiento se hacen en unidades lógicas llamadas bloques.

- Cada bloque comprende uno o más sectores.
- Dependiendo del disco, el sector suele ser de 512 bytes o 4096 bytes.

Las lecturas y escrituras a un medio de almacenamiento se hacen en unidades lógicas llamadas bloques.

- Cada bloque comprende uno o más sectores.
- Dependiendo del disco, el sector suele ser de 512 bytes o 4096 bytes.
- Los bloques son numerados consecutivamente a partir de 0, conformando su dirección lógica o LBA (Logical Block Address).

Las lecturas y escrituras a un medio de almacenamiento se hacen en unidades lógicas llamadas bloques.

- Cada bloque comprende uno o más sectores.
- Dependiendo del disco, el sector suele ser de 512 bytes o 4096 bytes.
- Los bloques son numerados consecutivamente a partir de 0, conformando su dirección lógica o LBA (Logical Block Address).
- El sistema de archivos puede definir un cluster de varios bloques para operar con el almacenamiento secundario de manera más eficiente.

Menú para hoy

¿Qué es un archivo?

- Una unidad de almacenamiento lógico.

¿Qué es un archivo?

- Una unidad de almacenamiento lógico.
- Un conjunto de información relacionada, con nombre (y otros metadatos), que se guarda en almacenamiento secundario.

¿Qué es un archivo?

- Una unidad de almacenamiento lógico.
- Un conjunto de información relacionada, con nombre (y otros metadatos), que se guarda en almacenamiento secundario.
- Desde una perspectiva de usuarie, es la porción más chica de almacenamiento (no puedo almacenar algo si no es en un archivo).

Archivos

Atributos de un archivo:

Archivos

Atributos de un archivo:

- Nombre
- Tipo
- Tamaño
- Permisos
- Timestamps
- ...

Archivos

Atributos de un archivo:

- Nombre
- Tipo
- Tamaño
- Permisos
- Timestamps
- ...

Operaciones sobre archivos:

Archivos

Atributos de un archivo:

- Nombre
- Tipo
- Tamaño
- Permisos
- Timestamps
- ...

Operaciones sobre archivos:

- Crear
- Abrir (syscall `open()`)
- Escribir (syscall `read()`)
- Leer (syscall `write()`)
- Borrar (syscall `unlink()`)
- ...

Menú para hoy

Asignación contigua

- Almacenar cada archivo en un conjunto de bloques contiguos.

Asignación contigua

- Almacenar cada archivo en un conjunto de bloques contiguos.
- ¿Qué datos necesito para acceder a un archivo?

Asignación contigua

- Almacenar cada archivo en un conjunto de bloques contiguos.
- ¿Qué datos necesito para acceder a un archivo?
  - La dirección del bloque inicial y el tamaño (en bloques) del archivo.

Asignación contigua

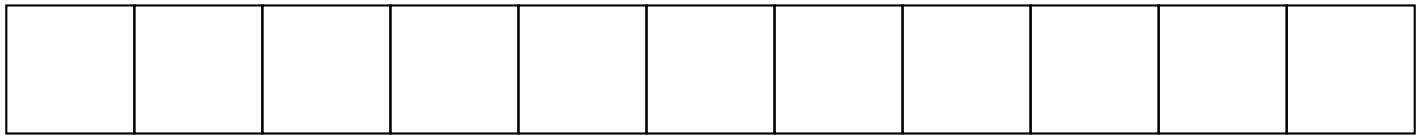
- Almacenar cada archivo en un conjunto de bloques contiguos.
- ¿Qué datos necesito para acceder a un archivo?
  - La dirección del bloque inicial y el tamaño (en bloques) del archivo.
- Es fácil de implementar pero tiene limitaciones.

Asignación contigua

- Almacenar cada archivo en un conjunto de bloques contiguos.
- ¿Qué datos necesito para acceder a un archivo?
  - La dirección del bloque inicial y el tamaño (en bloques) del archivo.
- Es fácil de implementar pero tiene limitaciones.
- Principal problema: encontrar espacio para un nuevo archivo. Como los archivos se van asignando y borrando, el espacio libre queda roto en pequeños pedacitos.

Asignación contigua

Ejemplo

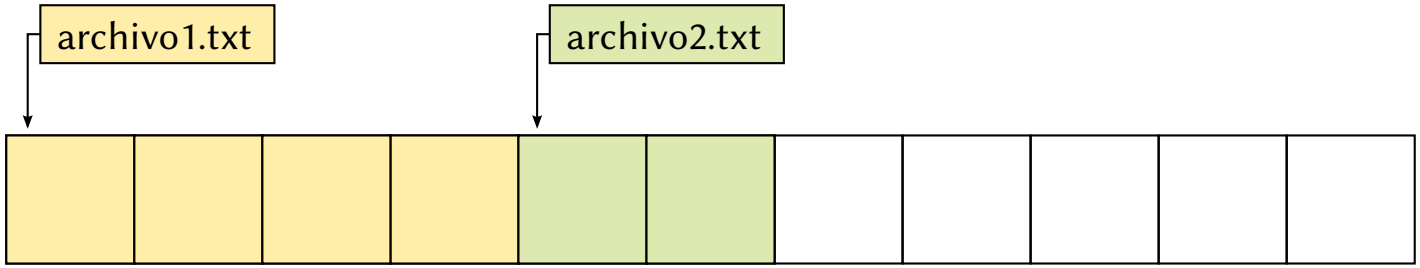


Asignación contigua

Ejemplo

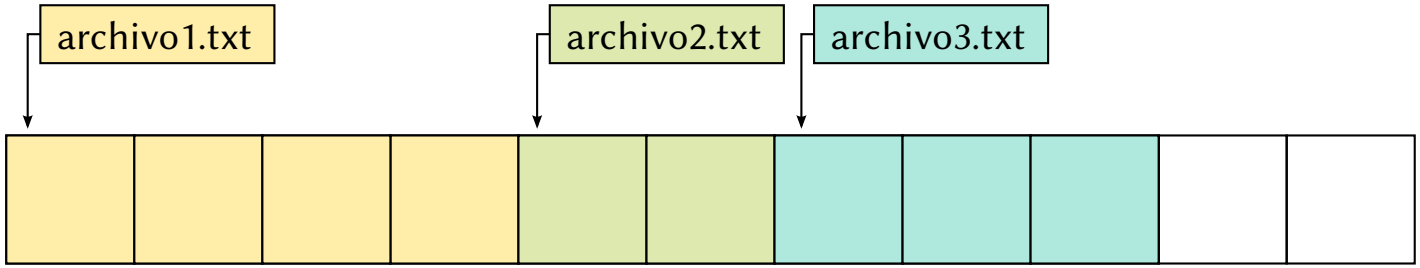
Asignación contigua

Ejemplo



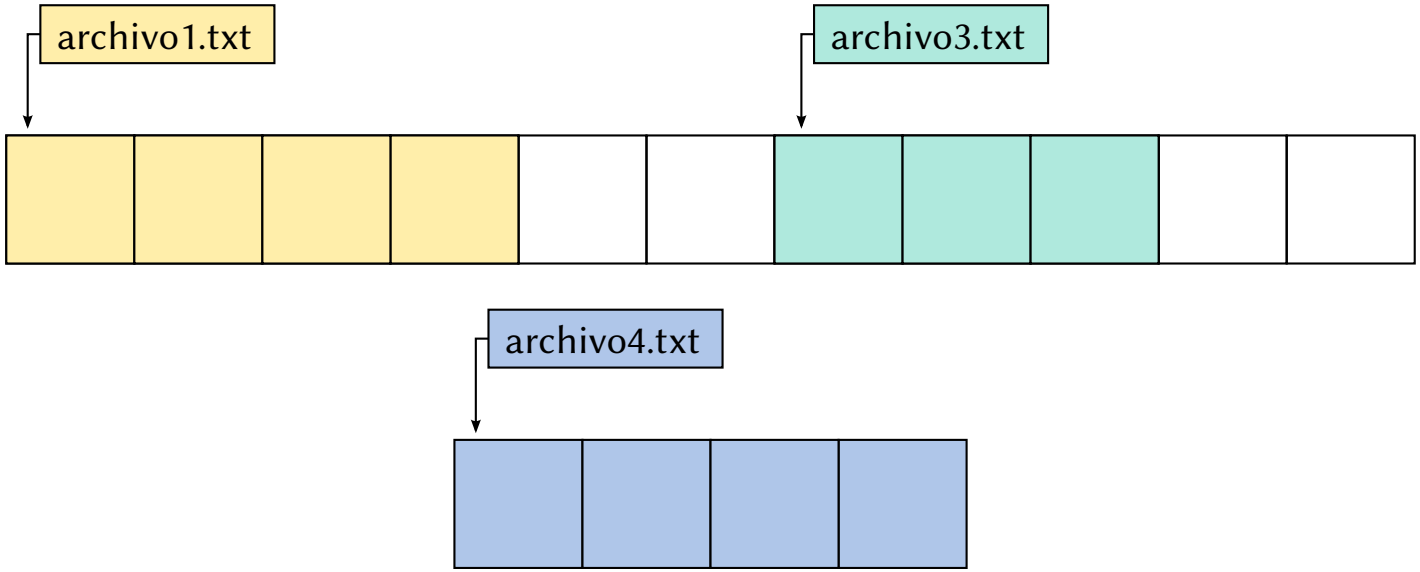
Asignación contigua

Ejemplo



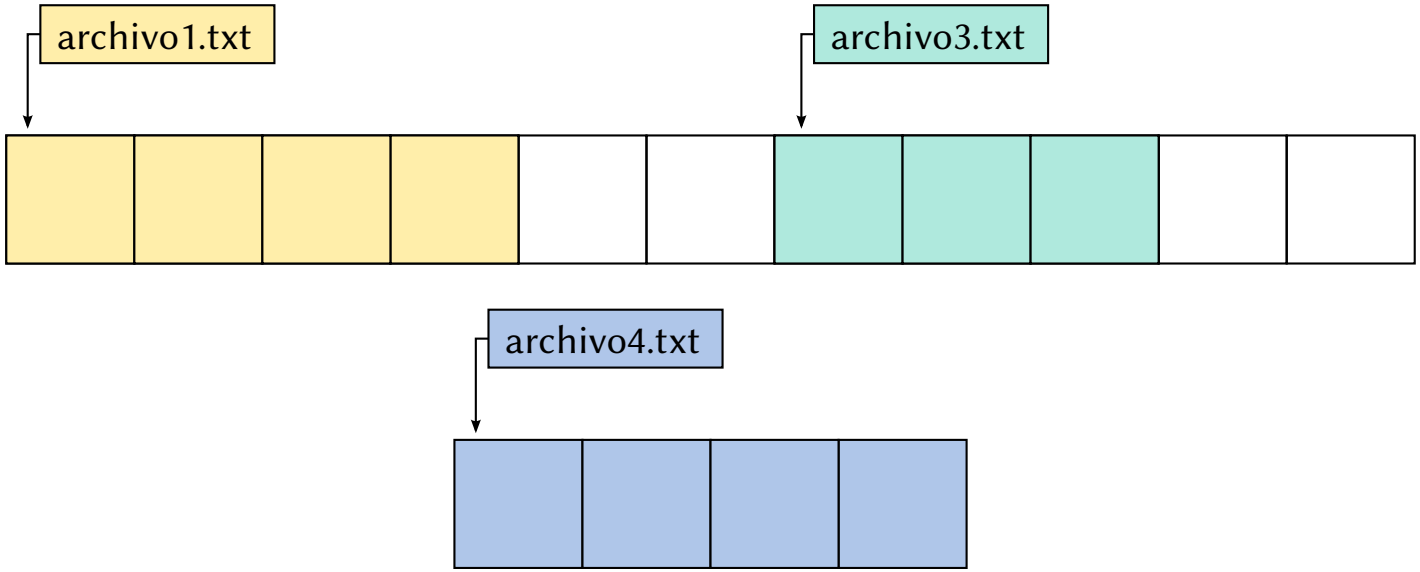
Asignación contigua

Ejemplo



Asignación contigua

Ejemplo



Problema: Aunque hay espacio para `archivo4.txt`, no se lo puede almacenar debido a la fragmentación externa.

Asignación contigua

- Compactar el disco es una tarea muy costosa.

Asignación contigua

- Compactar el disco es una tarea muy costosa.
- Para reusar los “huecos”, deberíamos saber qué “huecos” existen, y al crear un nuevo archivo deberíamos saber cuál será su tamaño final, y así asignar el “hueco” apropiado.

Asignación contigua

- Compactar el disco es una tarea muy costosa.
- Para reusar los “huecos”, deberíamos saber qué “huecos” existen, y al crear un nuevo archivo deberíamos saber cuál será su tamaño final, y así asignar el “hueco” apropiado.
- Existen situaciones que permiten esto: CD-ROMs y similares. En estos casos, los tamaños de archivo ya se conocen y se sabe que no cambiarán.

Menú para hoy

Tabla de asignación de archivos (FAT)

- Permite almacenar los archivos de forma *no secuencial*, guardando para cada bloque de un archivo, una referencia al siguiente (al estilo lista enlazada).

Tabla de asignación de archivos (FAT)

- Permite almacenar los archivos de forma *no secuencial*, guardando para cada bloque de un archivo, una referencia al siguiente (al estilo lista enlazada).
- La tabla tiene una entrada por cada bloque y se indexa por número de bloque.

Tabla de asignación de archivos (FAT)

- Permite almacenar los archivos de forma *no secuencial*, guardando para cada bloque de un archivo, una referencia al siguiente (al estilo lista enlazada).
- La tabla tiene una entrada por cada bloque y se indexa por número de bloque.
- Cada entrada de la tabla contiene el número de bloque del siguiente bloque en el archivo. El último bloque de un archivo se señala con un valor especial de EOF. Si el bloque no está en uso, se señala con otro valor especial.

Tabla de asignación de archivos (FAT)

- Permite almacenar los archivos de forma *no secuencial*, guardando para cada bloque de un archivo, una referencia al siguiente (al estilo lista enlazada).
- La tabla tiene una entrada por cada bloque y se indexa por número de bloque.
- Cada entrada de la tabla contiene el número de bloque del siguiente bloque en el archivo. El último bloque de un archivo se señala con un valor especial de EOF. Si el bloque no está en uso, se señala con otro valor especial.
- ¿Qué datos necesito para acceder a un archivo?

Tabla de asignación de archivos (FAT)

- Permite almacenar los archivos de forma *no secuencial*, guardando para cada bloque de un archivo, una referencia al siguiente (al estilo lista enlazada).
- La tabla tiene una entrada por cada bloque y se indexa por número de bloque.
- Cada entrada de la tabla contiene el número de bloque del siguiente bloque en el archivo. El último bloque de un archivo se señala con un valor especial de EOF. Si el bloque no está en uso, se señala con otro valor especial.
- ¿Qué datos necesito para acceder a un archivo?
  - El número de bloque del primer bloque del archivo.

Tabla de asignación de archivos (FAT)

Ejemplo

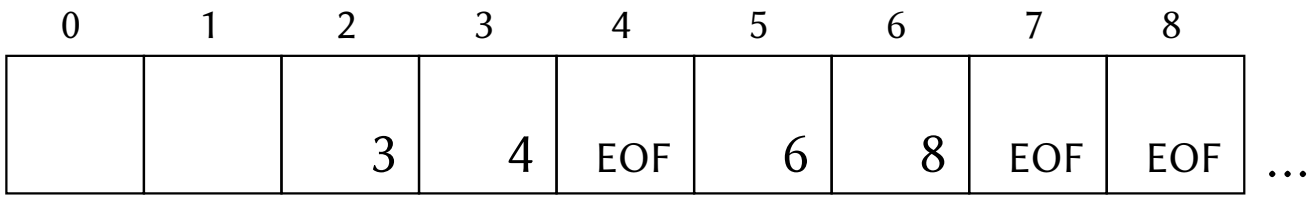


Tabla de asignación de archivos (FAT)

Ejemplo

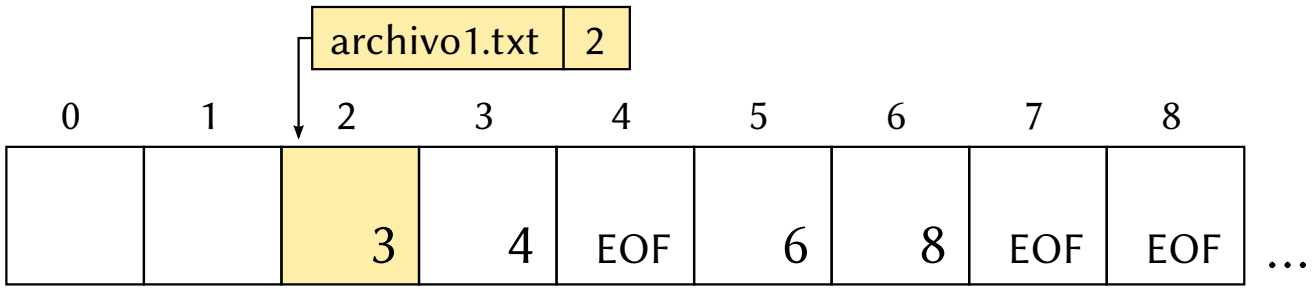


Tabla de asignación de archivos (FAT)

Ejemplo

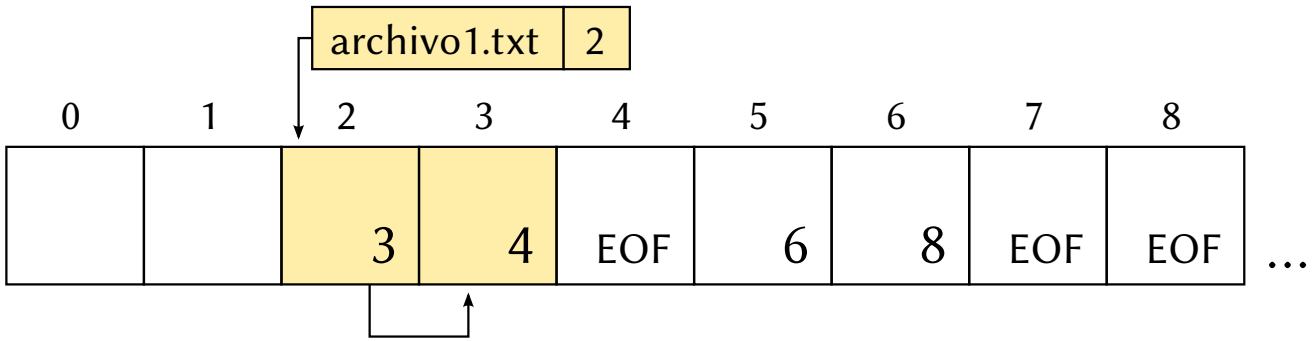


Tabla de asignación de archivos (FAT)

Ejemplo

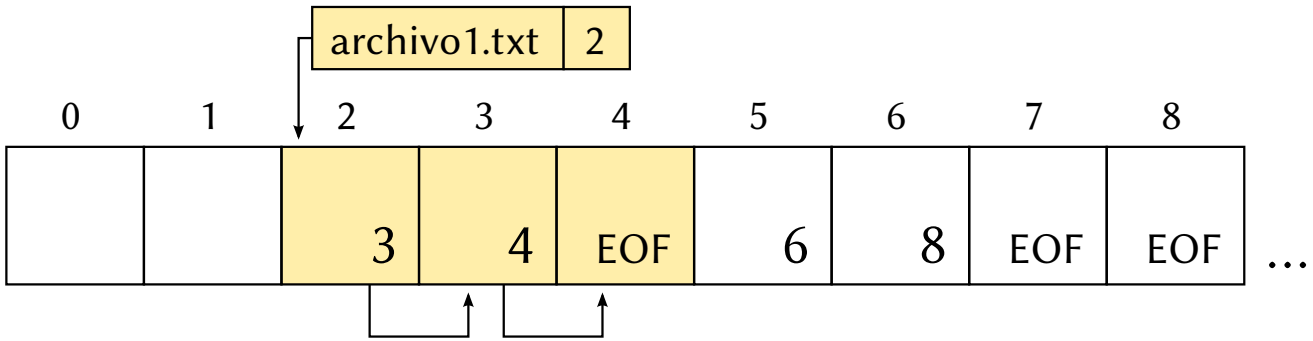


Tabla de asignación de archivos (FAT)

Ejemplo

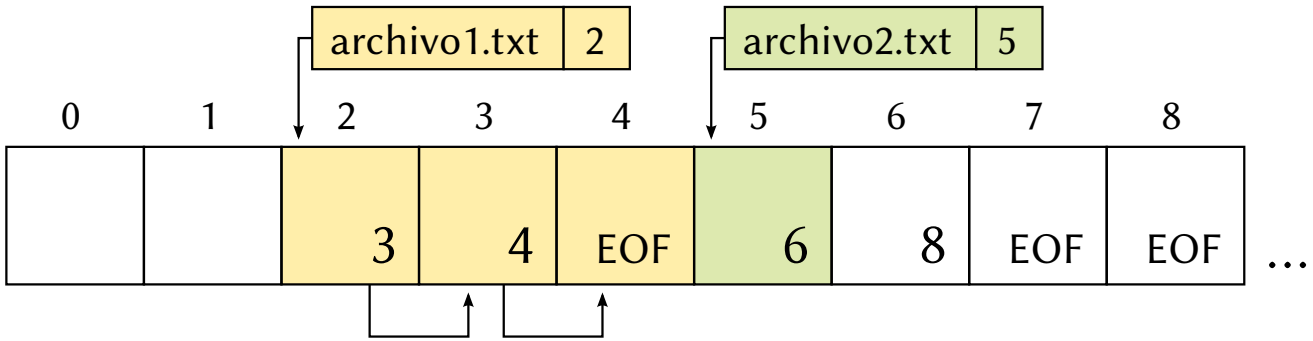


Tabla de asignación de archivos (FAT)

Ejemplo

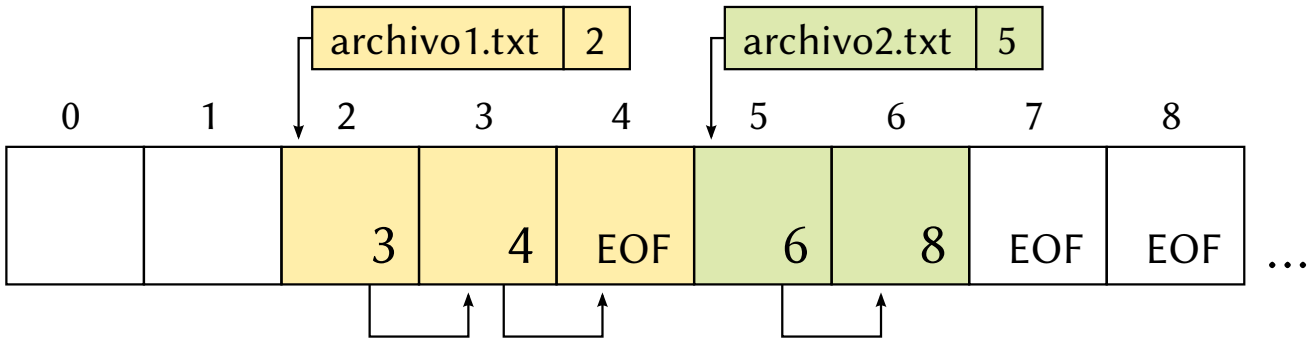


Tabla de asignación de archivos (FAT)

Ejemplo

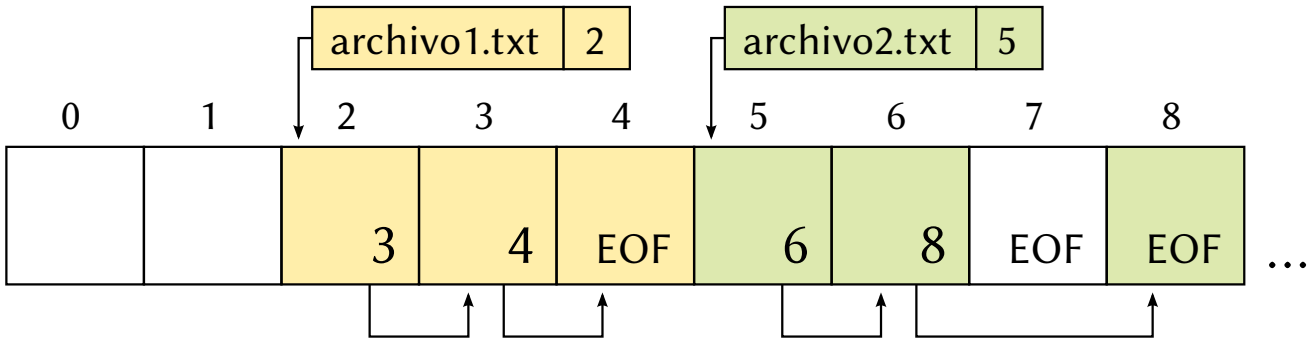


Tabla de asignación de archivos (FAT)

Ejemplo

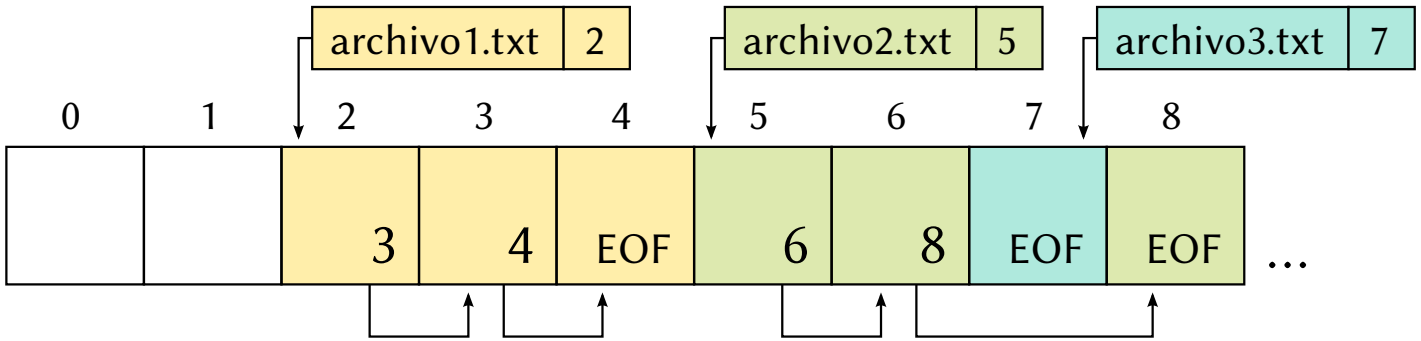


Tabla de asignación de archivos (FAT)

Ejercicio

¿Cuáles son los bloques de datos de los archivos `A` y `B`?

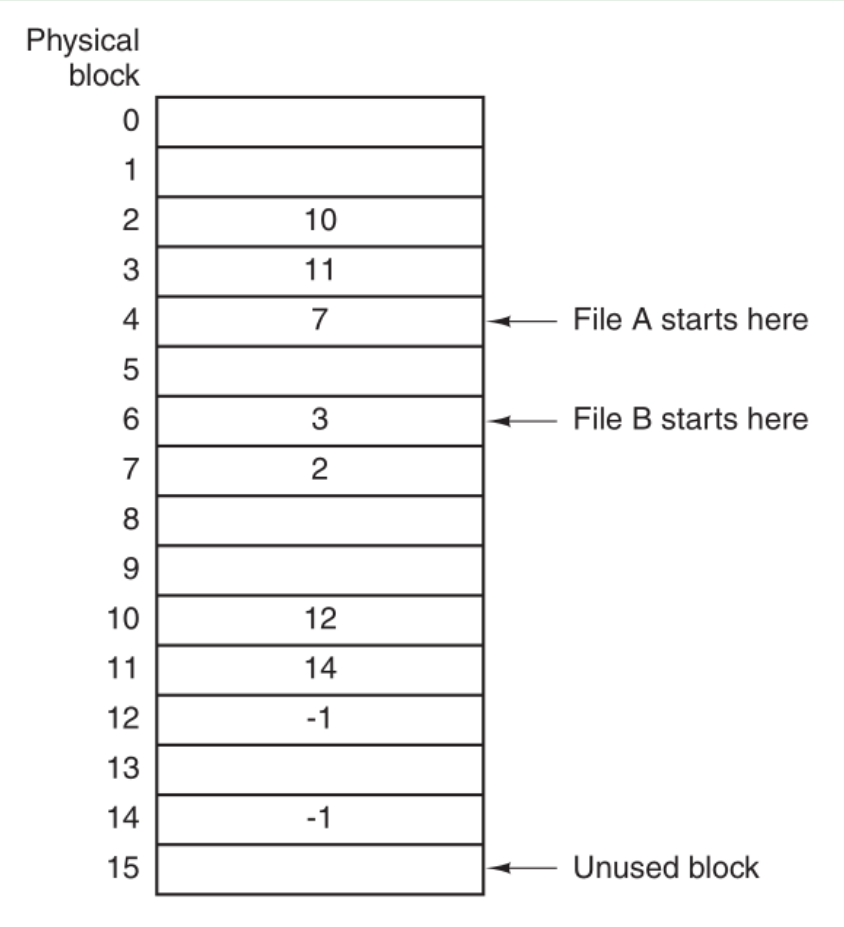
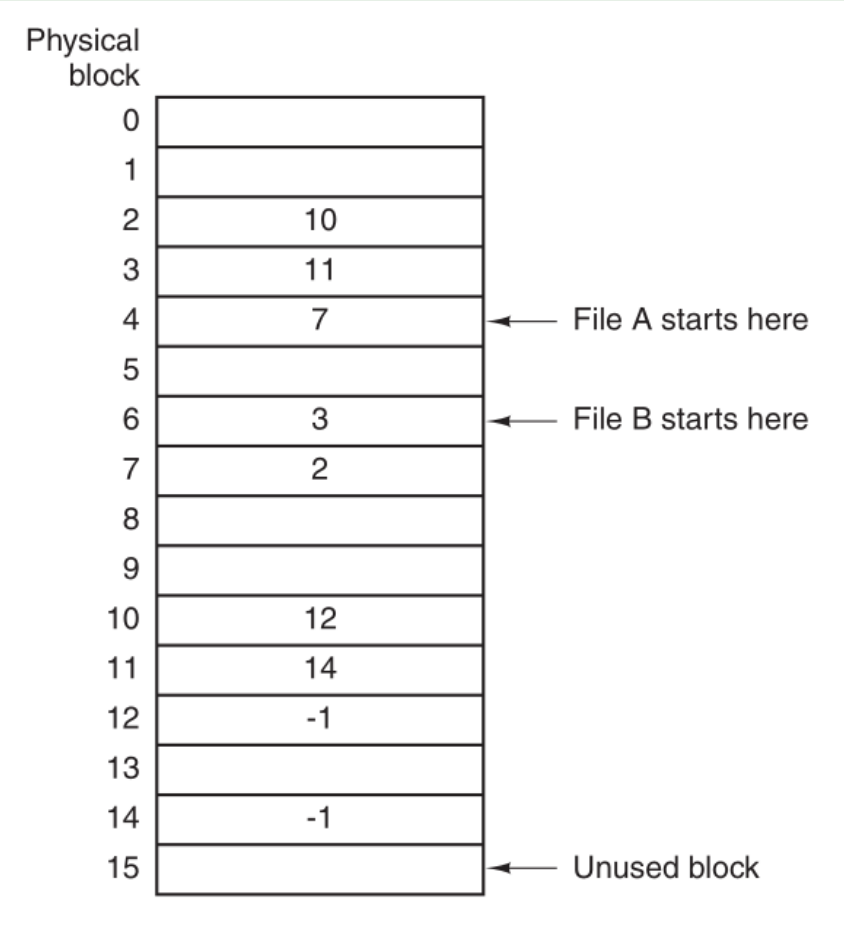


Tabla de asignación de archivos (FAT)

Ejercicio

¿Cuáles son los bloques de datos de los archivos `A` y `B`?



- `File A:` 4 - 7 - 2 - 10 - 12
- `File B:` 6 - 3 - 11 - 14

Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.

Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?

Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?
  - La tabla tendrá 1 entrada por cada bloque de disco.

Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?
  - La tabla tendrá 1 entrada por cada bloque de disco.
    Son $\frac{1TB}{1KB} = \frac{2^{40}bytes}{2^{10}bytes} = 2^{30}bytes$ entradas.

Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?
  - La tabla tendrá 1 entrada por cada bloque de disco.
    Son $\frac{1TB}{1KB} = \frac{2^{40}bytes}{2^{10}bytes} = 2^{30}bytes$ entradas.
  - Cada entrada de la tabla contendrá la dirección de un bloque.

Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?
  - La tabla tendrá 1 entrada por cada bloque de disco.
    Son $\frac{1TB}{1KB} = \frac{2^{40}bytes}{2^{10}bytes} = 2^{30}bytes$ entradas.
  - Cada entrada de la tabla contendrá la dirección de un bloque.
    Necesitamos 30 bits para direccionar esta cantidad de bloques.

Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?
  - La tabla tendrá 1 entrada por cada bloque de disco.
    Son $\frac{1TB}{1KB} = \frac{2^{40}bytes}{2^{10}bytes} = 2^{30}bytes$ entradas.
  - Cada entrada de la tabla contendrá la dirección de un bloque.
    Necesitamos 30 bits para direccionar esta cantidad de bloques.
    Como no podemos trabajar con menos de 1 byte, necesitamos $\lceil \frac{30bits}{8} \rceil = 4bytes$ para las direcciones.

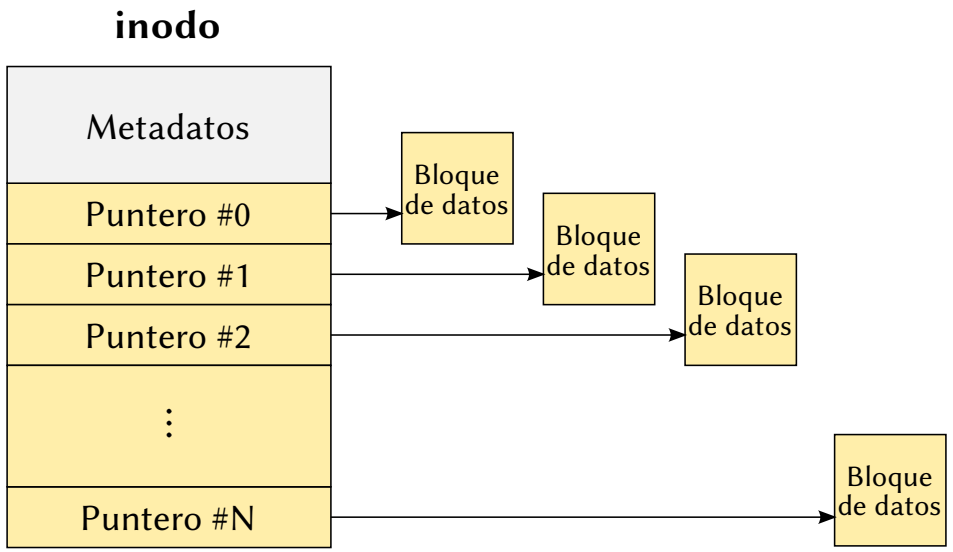
Tabla de asignación de archivos (FAT)

- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?
  - La tabla tendrá 1 entrada por cada bloque de disco.
    Son $\frac{1TB}{1KB} = \frac{2^{40}bytes}{2^{10}bytes} = 2^{30}bytes$ entradas.
  - Cada entrada de la tabla contendrá la dirección de un bloque.
    Necesitamos 30 bits para direccionar esta cantidad de bloques.
    Como no podemos trabajar con menos de 1 byte, necesitamos $\lceil \frac{30bits}{8} \rceil = 4bytes$ para las direcciones.
  - Entonces la tabla tendrá un tamaño de $2^{30} \times 2^2 bytes = 4GB$.

Menú para hoy

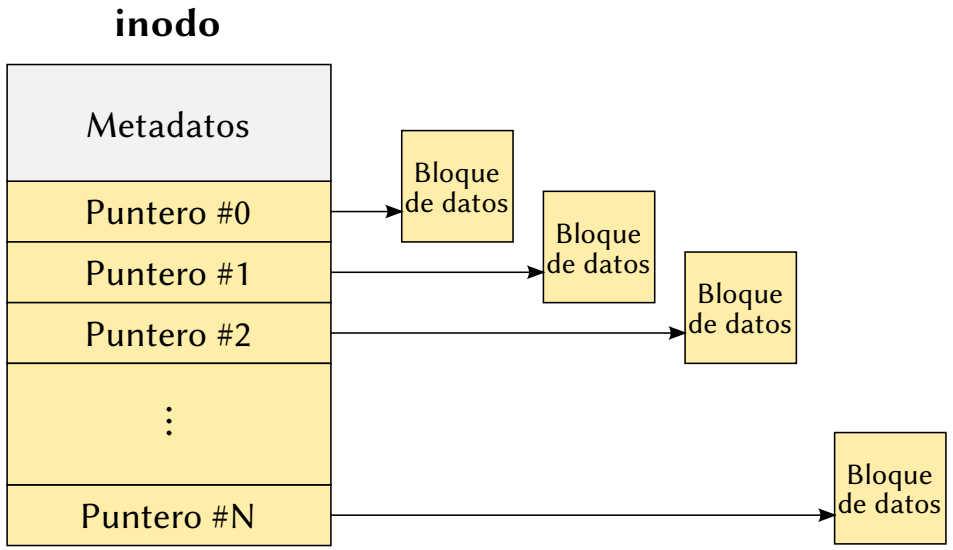
Inodos

- En un sistema con inodos, cada archivo tiene su propio índice de bloques, con punteros a los bloques de datos que conforman del archivo.



Inodos

- En un sistema con inodos, cada archivo tiene su propio índice de bloques, con punteros a los bloques de datos que conforman del archivo.
- El inodo debe cargarse en memoria sólo cuando el archivo correspondiente es abierto.



Inodos

- La i-ésima entrada en el índice apunta al iésimo bloque del archivo.

Inodos

- La i-ésima entrada en el índice apunta al iésimo bloque del archivo.
- Ojo: mantener este índice requiere espacio. ¿Qué tan grande debe ser la estructura de índices?

Inodos

- La i-ésima entrada en el índice apunta al iésimo bloque del archivo.
- Ojo: mantener este índice requiere espacio. ¿Qué tan grande debe ser la estructura de índices?
  - Queremos que sea lo más chico posible.

Inodos

- La i-ésima entrada en el índice apunta al iésimo bloque del archivo.
- Ojo: mantener este índice requiere espacio. ¿Qué tan grande debe ser la estructura de índices?
  - Queremos que sea lo más chico posible.
  - Pero si es muy pequeño, no podrá almacenar la cantidad de punteros suficientes para un archivo grande.

Inodos

Punteros con indirección

- Es deseable que los inodos tengan un tamaño fijo.

Inodos

Punteros con indirección

- Es deseable que los inodos tengan un tamaño fijo.
- Además, los primeros bloques de un archivo suelen ser accedidos con más frecuencia.

Inodos

Punteros con indirección

- Es deseable que los inodos tengan un tamaño fijo.
- Además, los primeros bloques de un archivo suelen ser accedidos con más frecuencia.
- Por eso, se utilizan **punteros con indirección**.

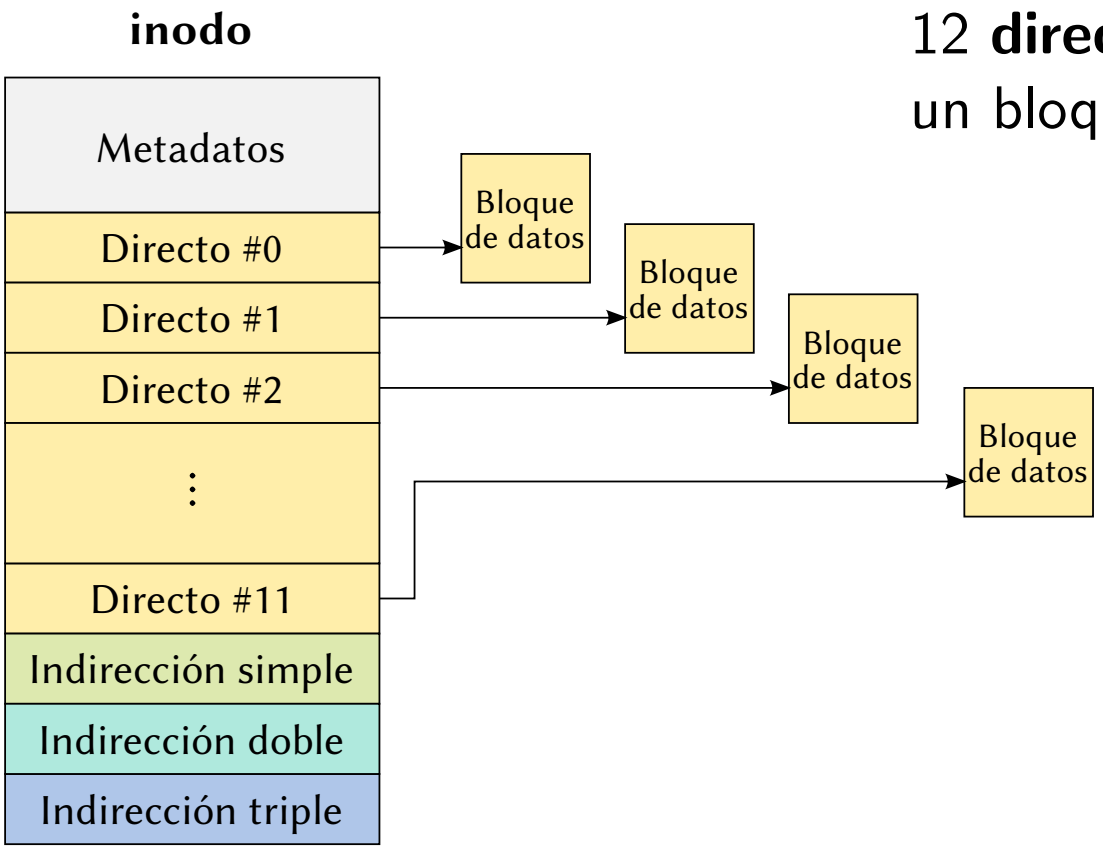
Inodos

Punteros con indirección

- Es deseable que los inodos tengan un tamaño fijo.
- Además, los primeros bloques de un archivo suelen ser accedidos con más frecuencia.
- Por eso, se utilizan **punteros con indirección**.
- Por ejemplo, en `Ext2`, todos los inodos contienen 15 punteros a bloques, de cuatro tipos distintos.

Inodos

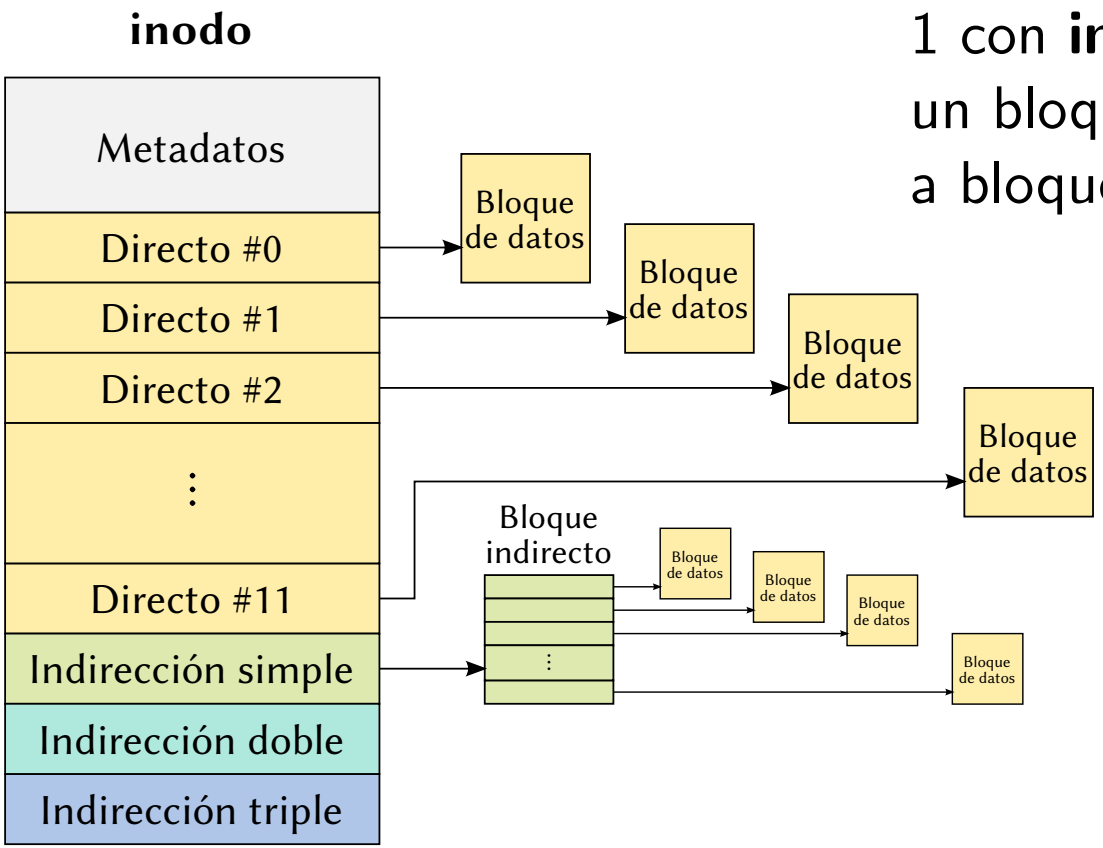
Punteros con indirección



12 **directos**: apuntan directamente a un bloque de datos.

Inodos

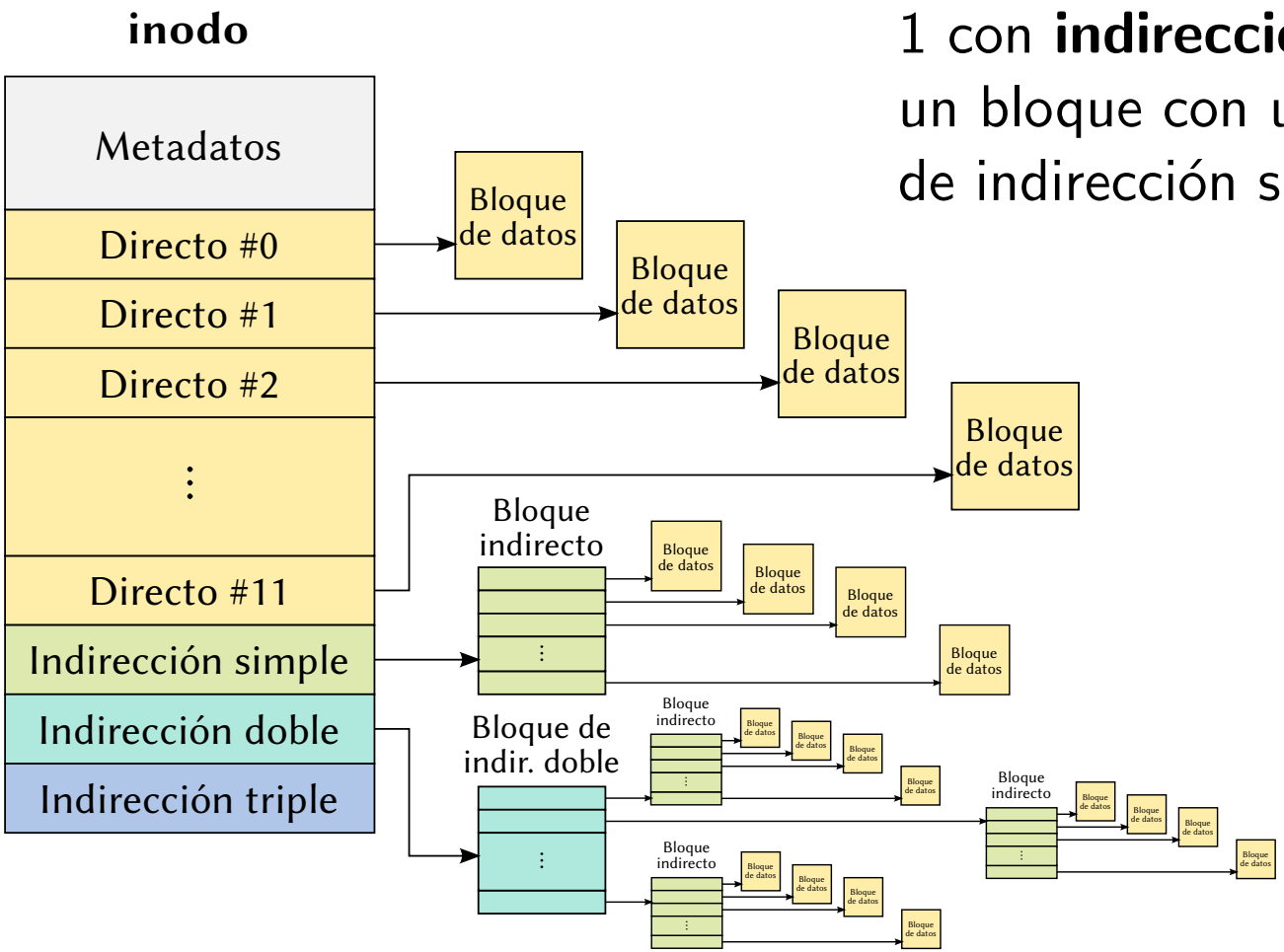
Punteros con indirección



1 con **indirección simple**: apunta a un bloque con una lista de punteros a bloques de datos.

Inodos

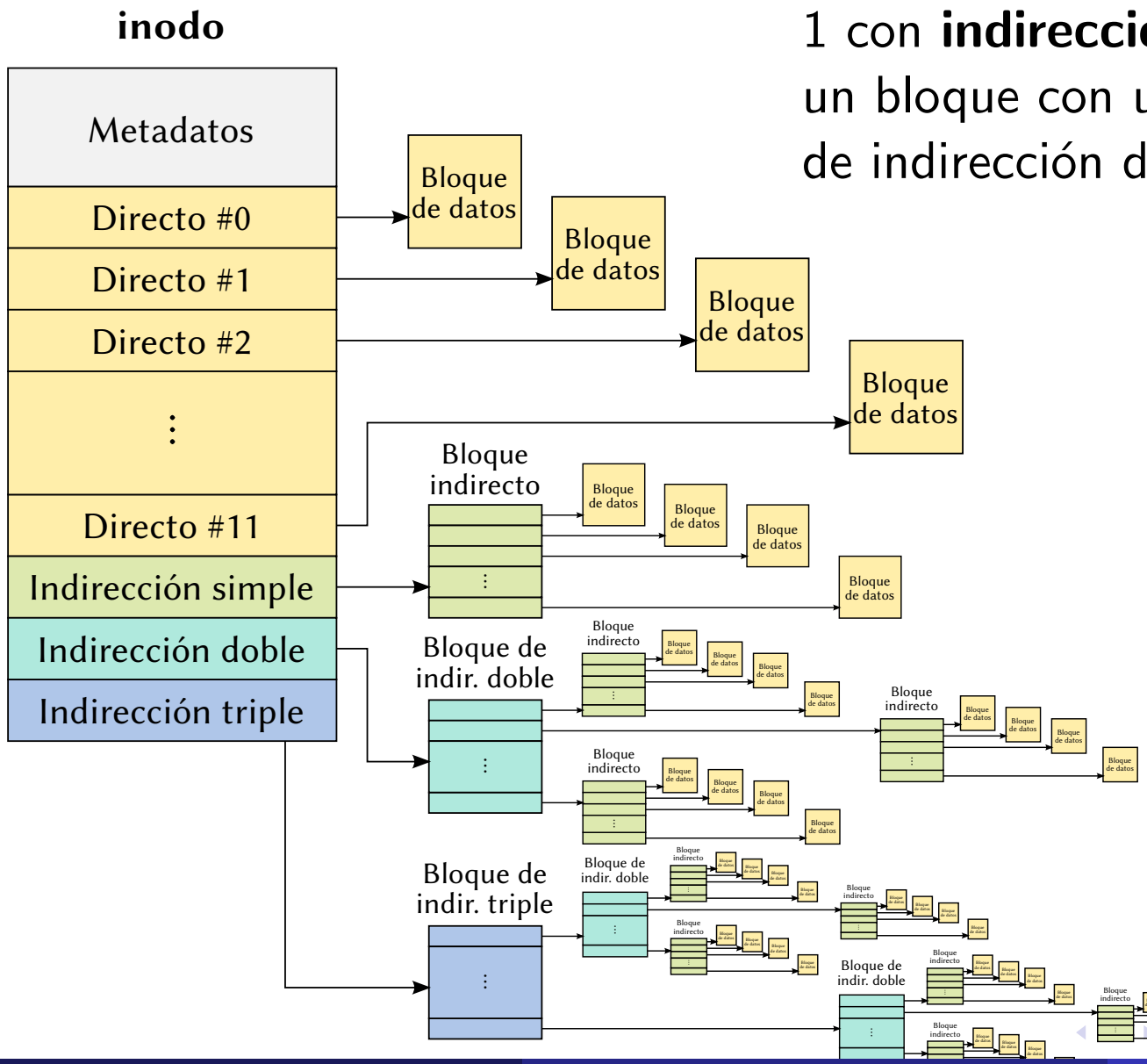
Punteros con indirección



1 con **indirección doble**: apunta a un bloque con una lista de punteros de indirección simple.

Inodos

Punteros con indirección



1 con **indirección triple**: apunta a un bloque con una lista de punteros de indirección doble.

Tabla de asignación de archivos (FAT)

Ejercicio

Consideremos un inodo con 4 entradas directas y 2 indirectas simples. ¿Qué pinta tiene el inodo de un archivo cuyos bloques de datos son 4 - 7 - 2 - 10 - 12 en ese orden?

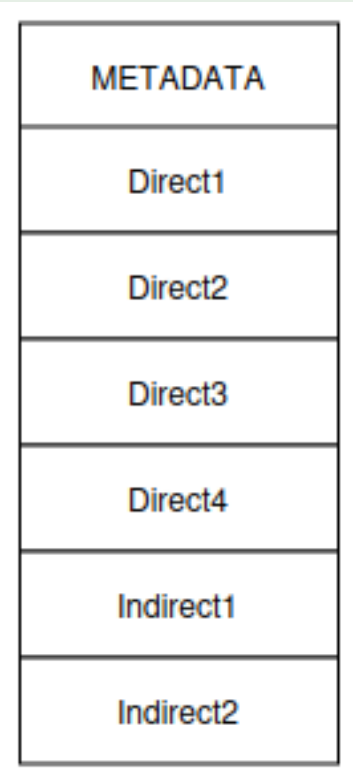
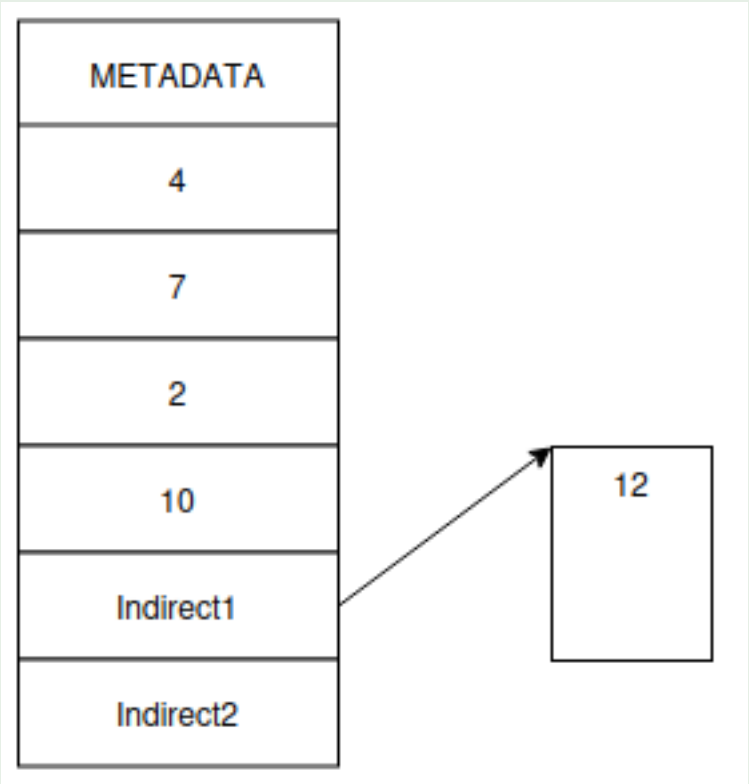


Tabla de asignación de archivos (FAT)

Ejercicio

Consideremos un inodo con 4 entradas directas y 2 indirectas simples. ¿Qué pinta tiene el inodo de un archivo cuyos bloques de datos son 4 - 7 - 2 - 10 - 12 en ese orden?



Menú para hoy

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.
- La entrada de directorio provee la información necesaria para encontrar los bloques de disco donde está almacenado el archivo:

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.
- La entrada de directorio provee la información necesaria para encontrar los bloques de disco donde está almacenado el archivo:
  - En FAT:

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.
- La entrada de directorio provee la información necesaria para encontrar los bloques de disco donde está almacenado el archivo:
  - En FAT:

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.
- La entrada de directorio provee la información necesaria para encontrar los bloques de disco donde está almacenado el archivo:
  - En FAT: el número del primer bloque.

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.
- La entrada de directorio provee la información necesaria para encontrar los bloques de disco donde está almacenado el archivo:
  - En FAT: el número del primer bloque.
  - En inodos:

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.
- La entrada de directorio provee la información necesaria para encontrar los bloques de disco donde está almacenado el archivo:
  - En FAT: el número del primer bloque.
  - En inodos:

Directorios

- Antes de poder leer un archivo (syscall `read()`), hay que abrirlo (syscall `open()`).
- Para abrir un archivo, se debe especificar la ruta (absoluta o relativa).
- El SO "recorre" el path hasta encontrar la entrada de directorio correspondiente al archivo.
- La entrada de directorio provee la información necesaria para encontrar los bloques de disco donde está almacenado el archivo:
  - En FAT: el número del primer bloque.
  - En inodos: el número de inodo.

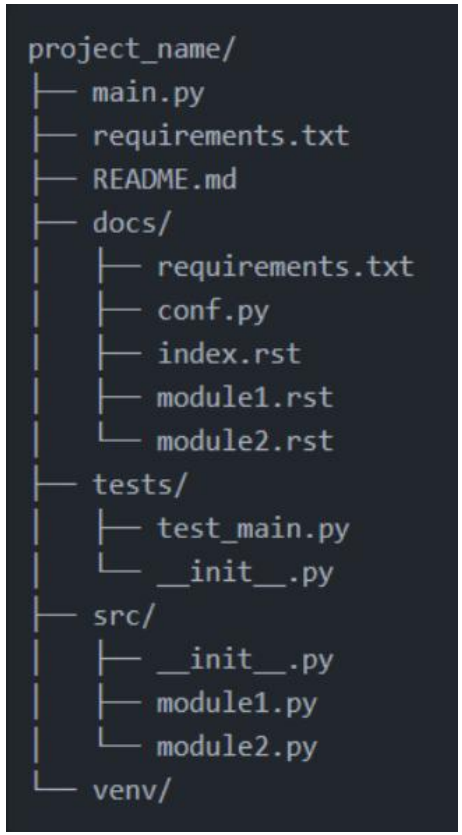
Directorios

Los directorios también son archivos. Sus bloques de datos contienen una colección de entradas, una por cada archivo dentro del directorio, indicando su nombre y el número de inodo/primer bloque.

Directorios

Los directorios también son archivos. Sus bloques de datos contienen una colección de entradas, una por cada archivo dentro del directorio, indicando su nombre y el número de inodo/primer bloque.

- Un directorio puede contener subdirectorios. Así, podemos organizar los archivos en una estructura jerárquica, mediante un árbol de directorios.
- La jerarquía de directorios comienza en el directorio raíz.
- Los directorios y archivos pueden tener el mismo nombre si se encuentran en diferentes *paths*.



Menú para hoy

Directorios en FAT

- Consiste en una lista de entradas de tamaño fijo.

Directorios en FAT

- Consiste en una lista de entradas de tamaño fijo.
- Cada entrada de directorio indica el índice del primer bloque de cada archivo.

Directorios en FAT

- Consiste en una lista de entradas de tamaño fijo.
- Cada entrada de directorio indica el índice del primer bloque de cada archivo.
- La entrada de directorio almacena también todos los metadatos: nombre, tamaño, fecha de último acceso, etc.

Directorios en FAT

- Consiste en una lista de entradas de tamaño fijo.
- Cada entrada de directorio indica el índice del primer bloque de cada archivo.
- La entrada de directorio almacena también todos los metadatos: nombre, tamaño, fecha de último acceso, etc.
- El bloque del directorio **root** es distinguido. De esta forma, podemos encontrar cualquier archivo a partir de su ruta.

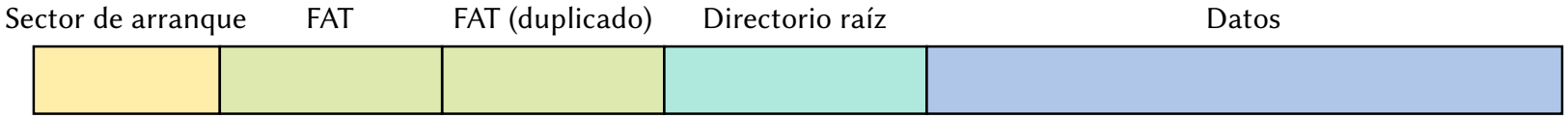
Directorios en FAT

- Consiste en una lista de entradas de tamaño fijo.
- Cada entrada de directorio indica el índice del primer bloque de cada archivo.
- La entrada de directorio almacena también todos los metadatos: nombre, tamaño, fecha de último acceso, etc.
- El bloque del directorio **root** es distinguido. De esta forma, podemos encontrar cualquier archivo a partir de su ruta.

Estructura de un sistema de archivos FAT32.

Directorios en FAT

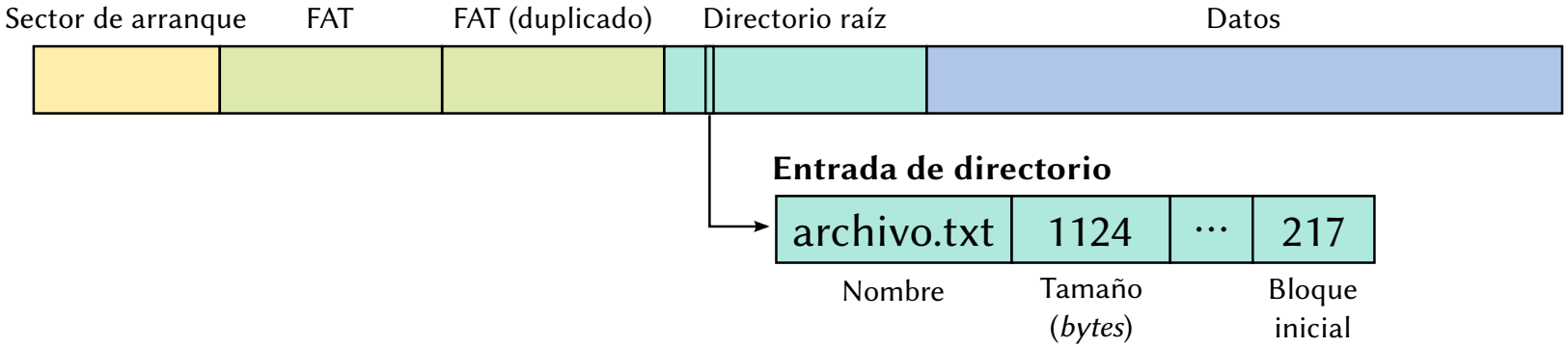
Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

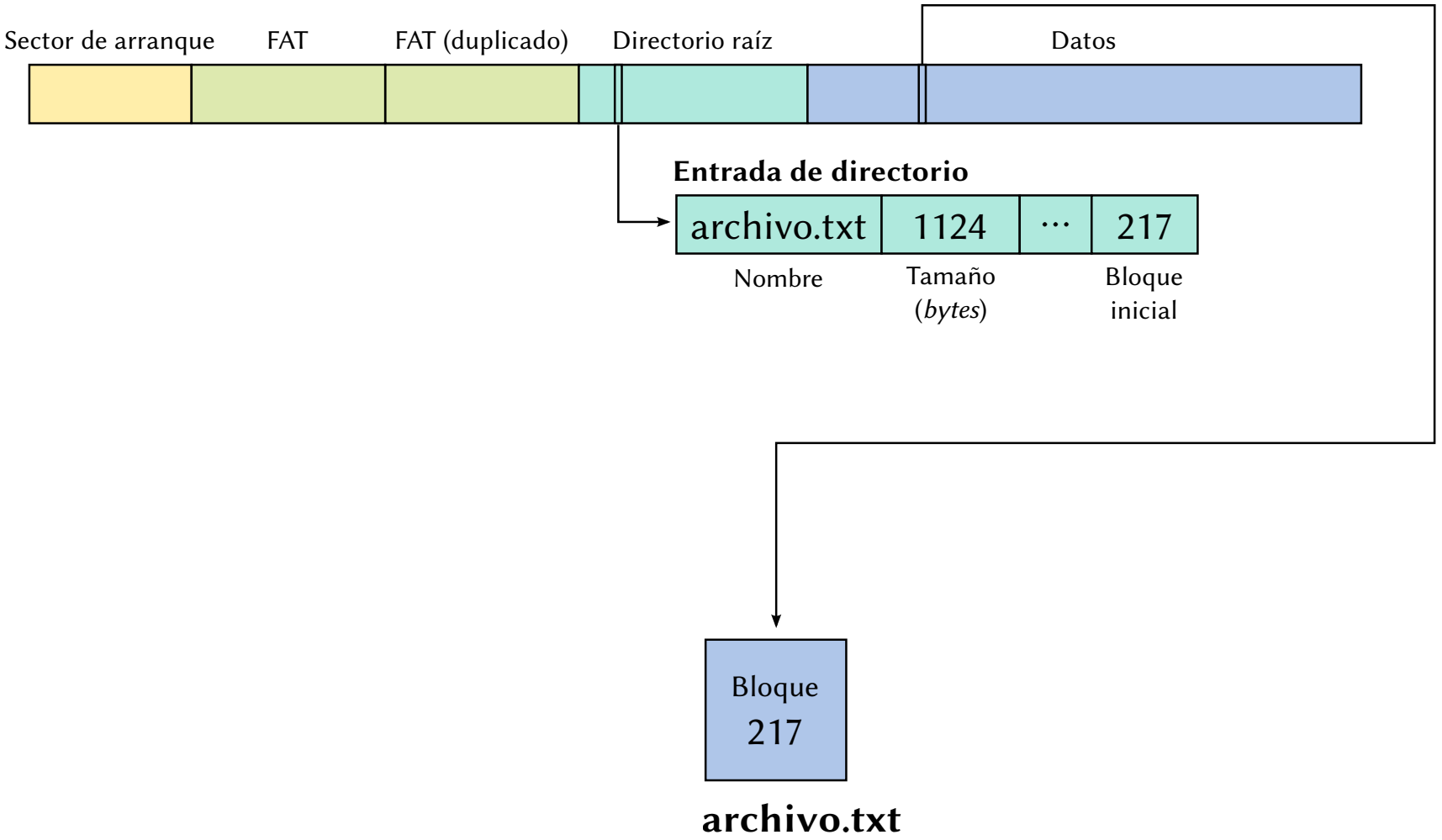
Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

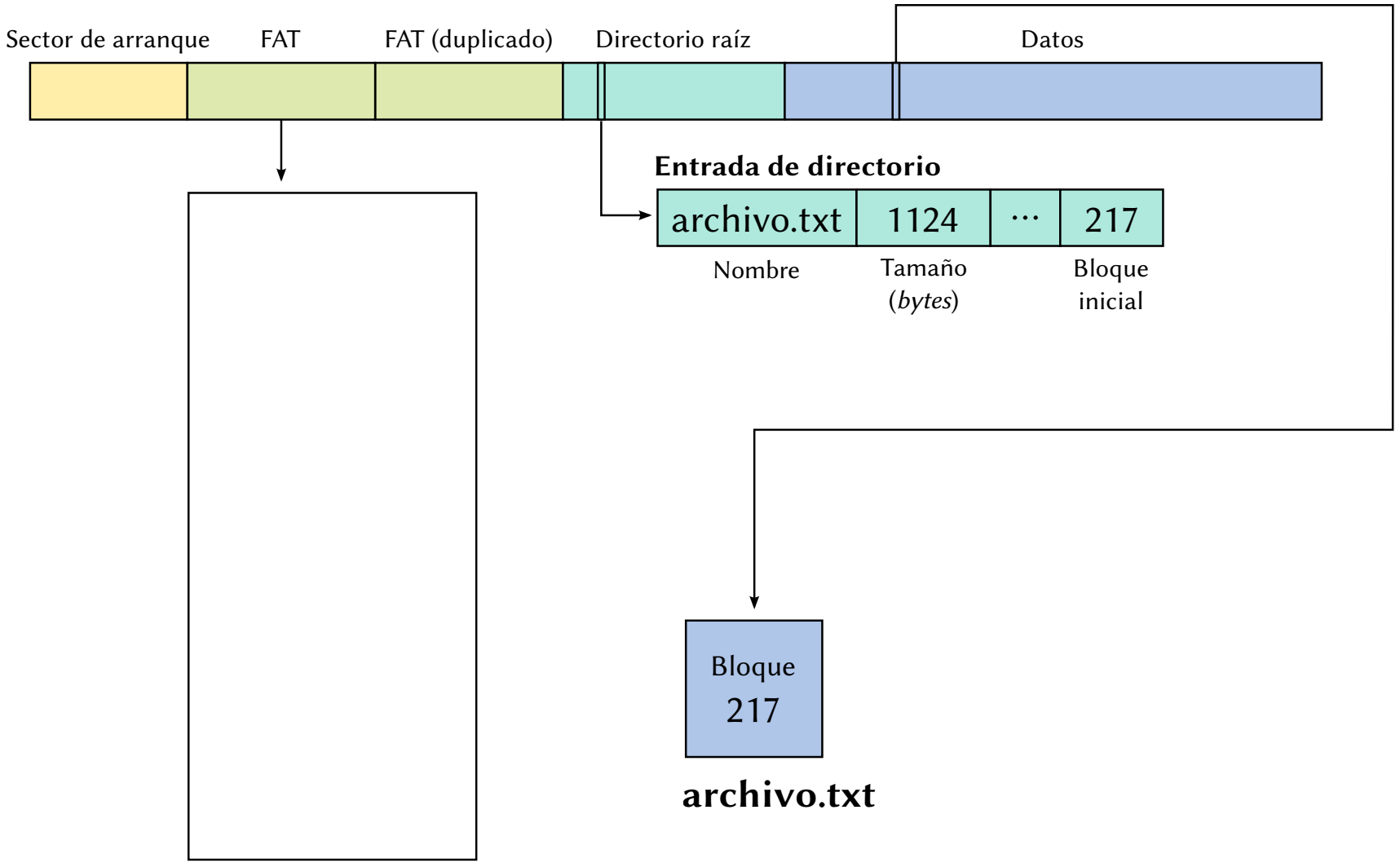
Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

Obteniendo un archivo(*)

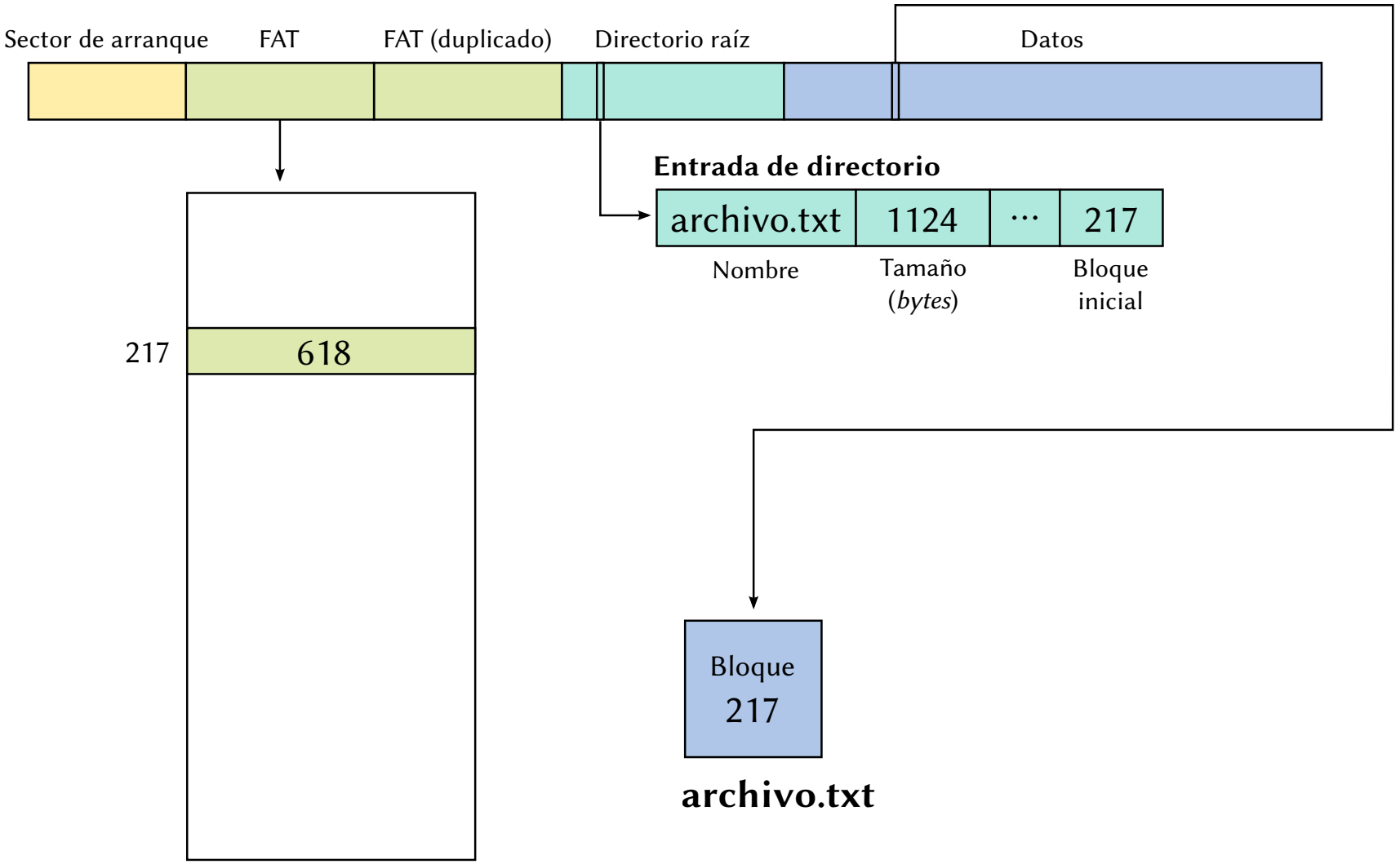
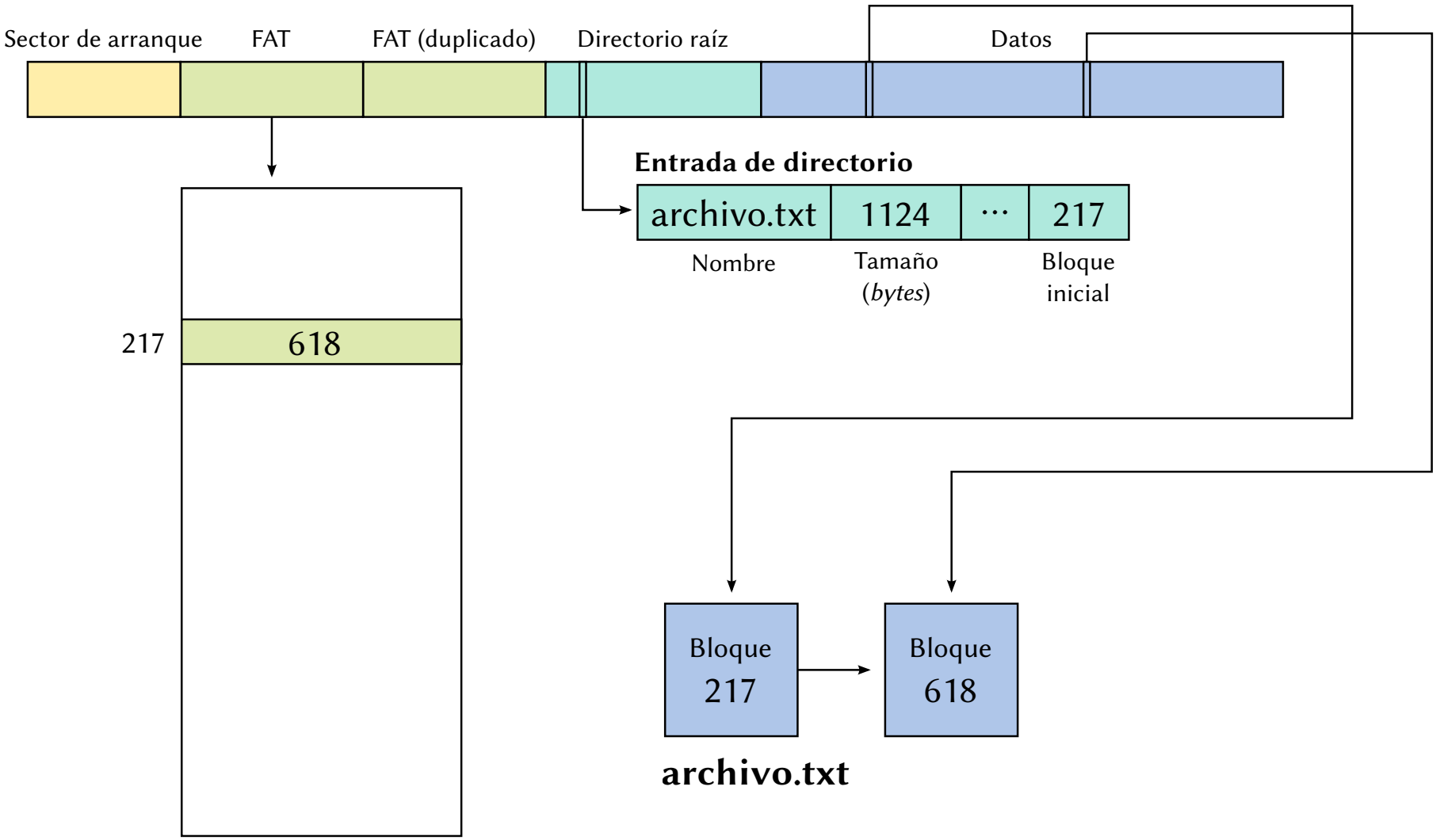


Tabla de asignación de archivos

(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

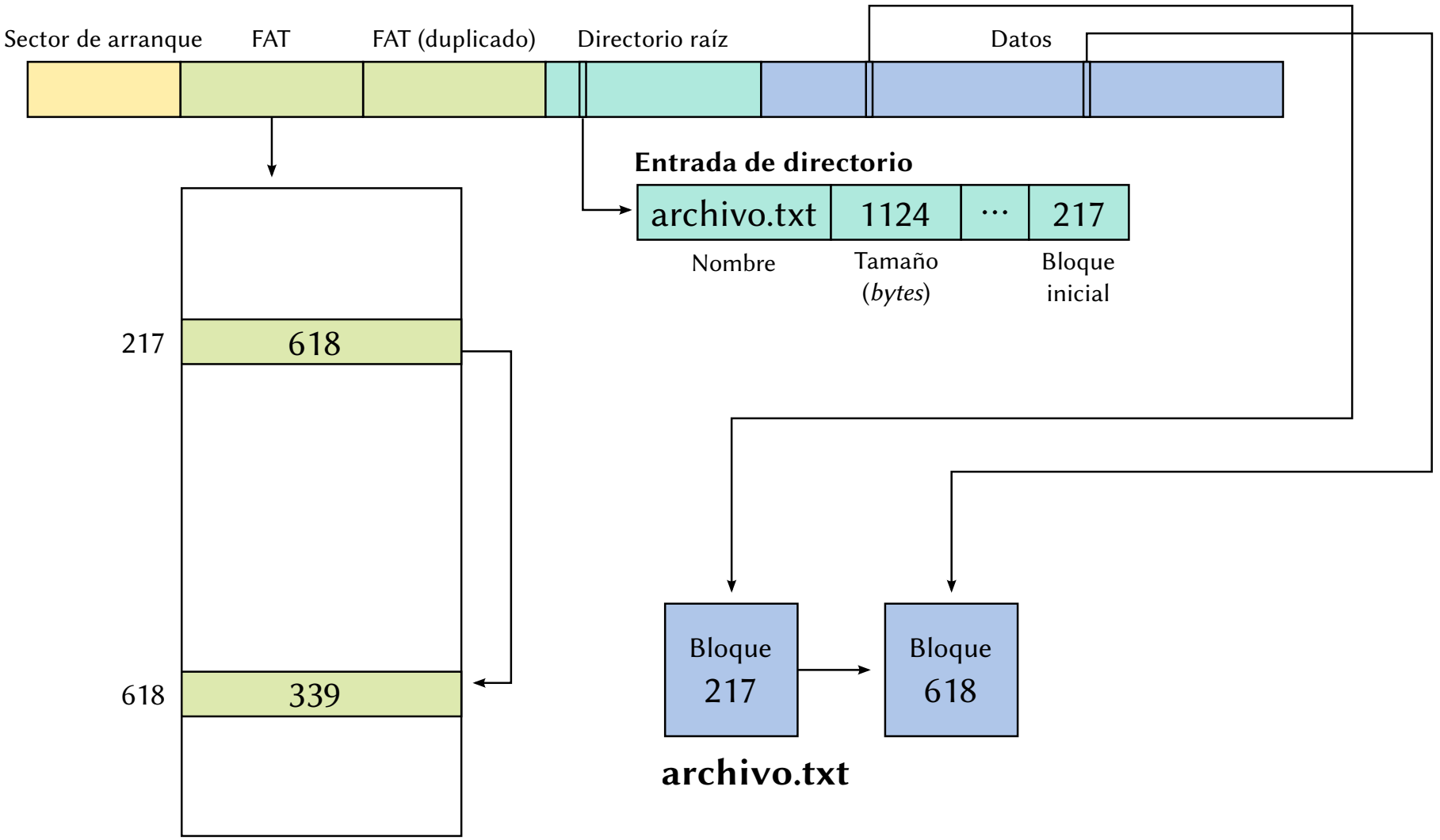
Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

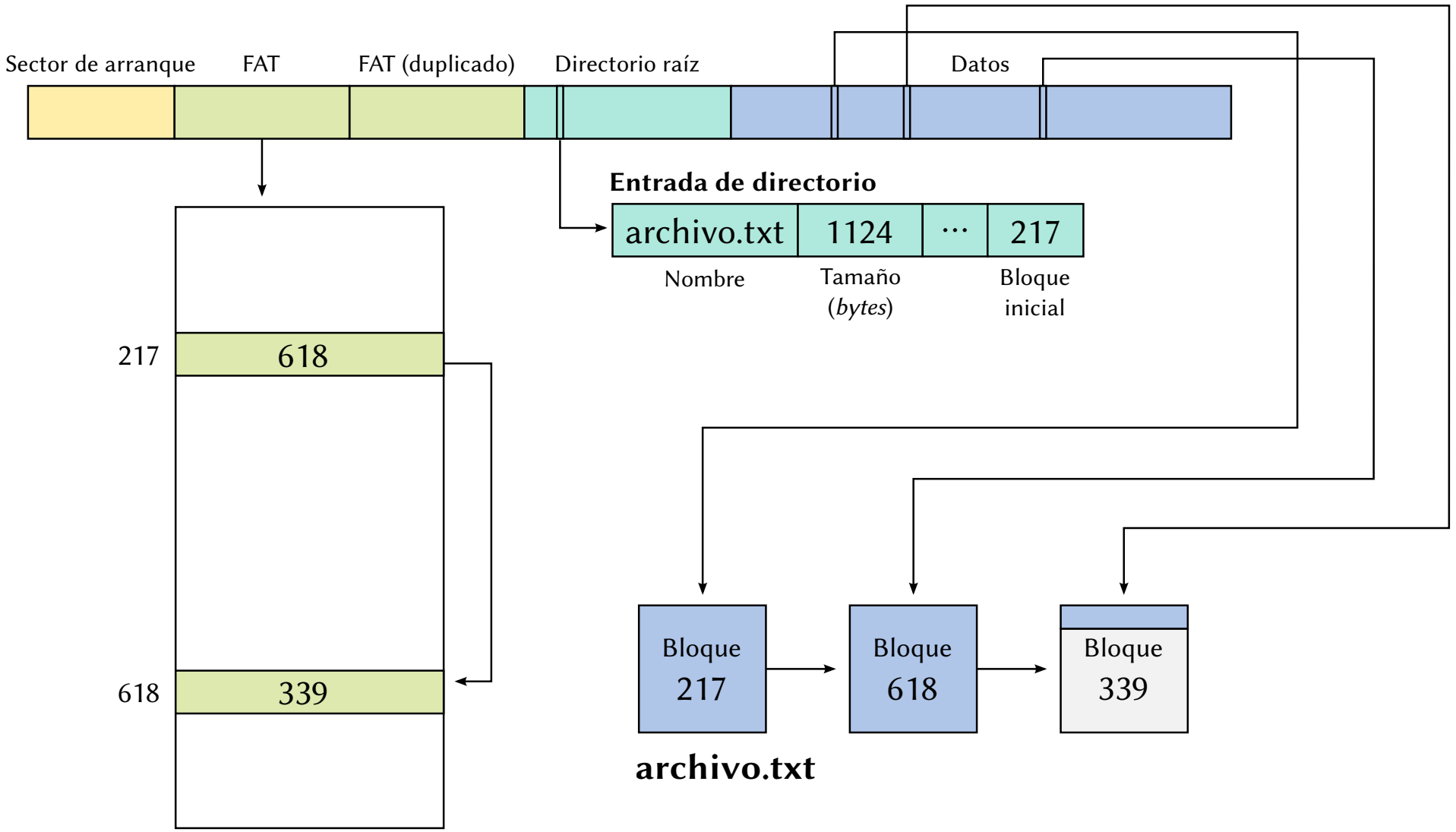
Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

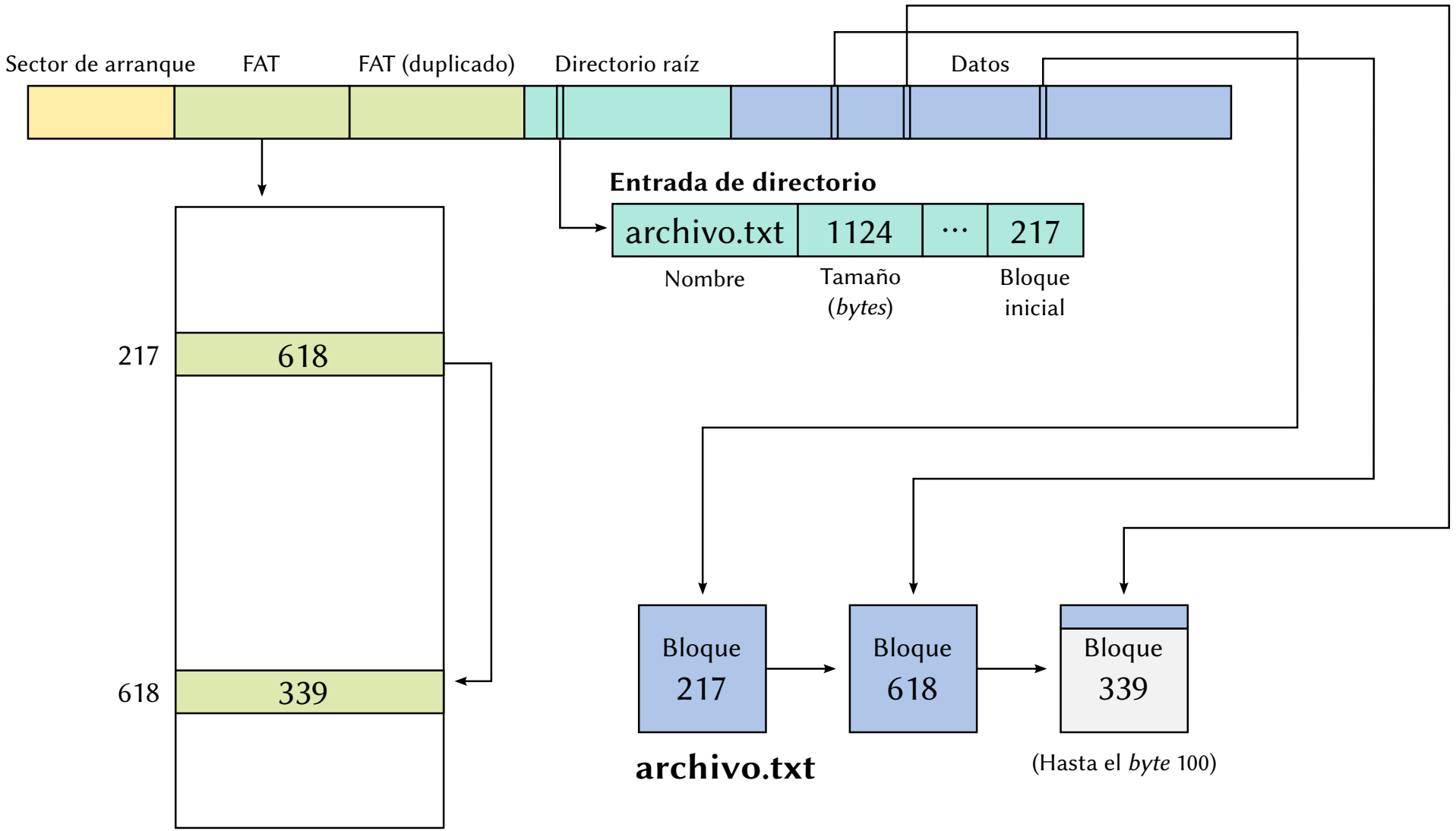
Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

Obteniendo un archivo(*)



(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

Obteniendo un archivo(*)

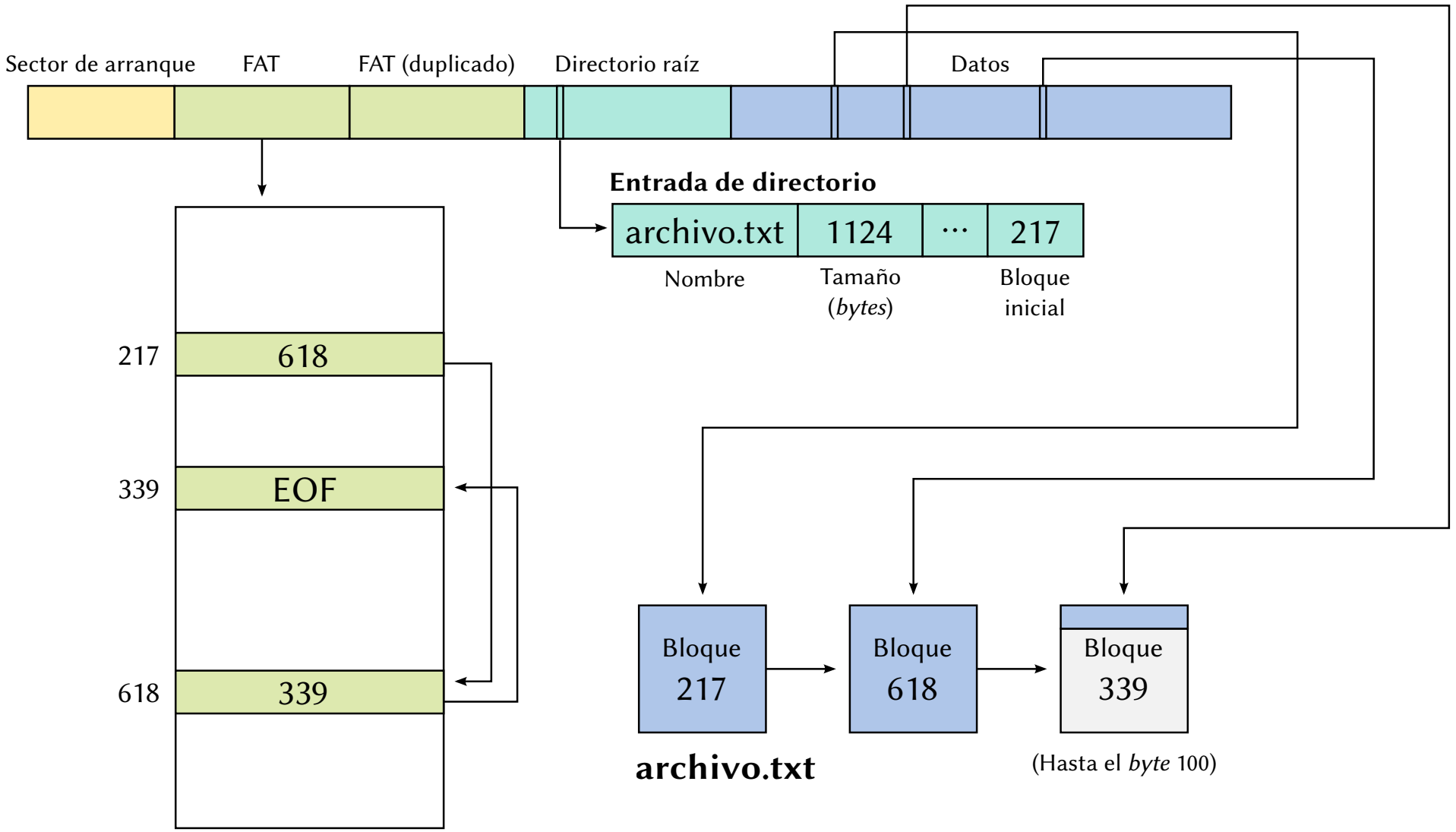


Tabla de asignación de archivos

(*)Suponiendo bloques de 512 bytes.

Directorios en FAT

Ejercicio

Enunciado

Contamos un sistema de archivos formateado con FAT12 con bloques de 2KiB (con clusters de 1 bloque). Indicar a cuántos bloques de disco hay que acceder para leer los bytes [3.000-3.083] del archivo `/home/el/parcial.avi`.
Se puede asumir que:

- Los archivos y directorios a buscar se encuentran siempre en el primer bloque de *directory entries* correspondientes al directorio padre.
- Si un bloque se lee dos veces, se va a buscar a disco sólo la primera vez.
- La FAT ya está cargada en memoria (el resto de los bloques a leer deberán ser contemplados).

Por convención, el primer byte de un archivo es el número 0. Si escribimos [0-4] significa todos los bytes del rango 0 a 4, incluyendo ambos extremos del rango. Puede dejar expresado el resultado como potencias de dos. Justificar.

Solución

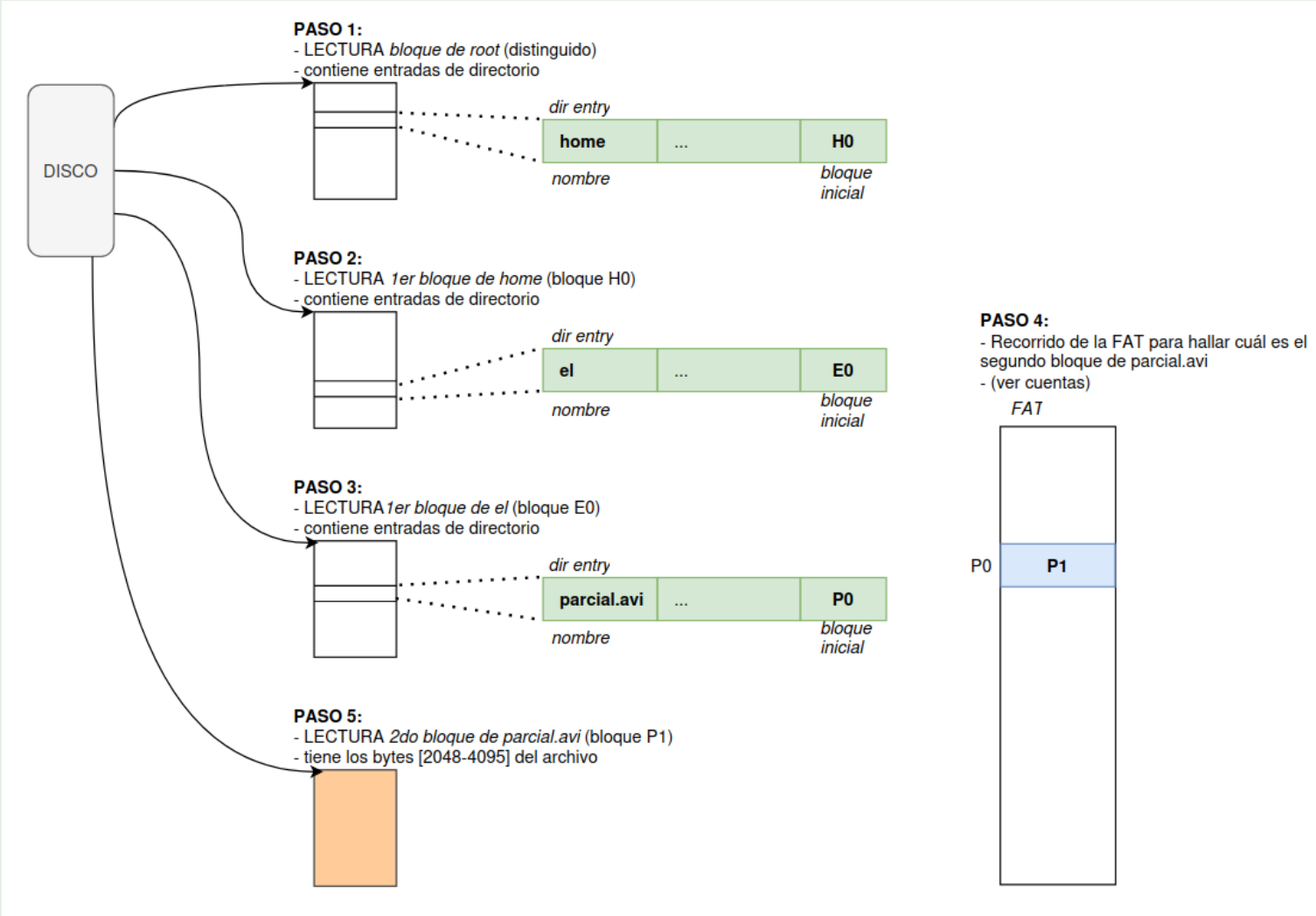
Archivo: `/home/el/parcial.avi`. Bytes: [3.000-3.083].

- Tengo la entrada de `root` (distinguido), busco en la FAT el 1er bloque de la tabla de directorios para hallar la entrada de `home`. Cuando la encuentro, busco en la FAT el 1er bloque de la tabla de directorios para hallar la entrada de `el`. Lo mismo para `parcial.avi`. **3 LECTURAS DE DISCO**.
- Ahora miro en qué bloques están los bytes pedidos. Como los bloques son de 2KiB, tengo los bytes [0-2047] en el primer bloque y [2048-4095] en el segundo, aquí están los bytes que necesito. Para llegar al segundo bloque simplemente navego la FAT (que ya está en memoria) hasta saber cuál es el segundo bloque, y lo traigo. **1 LECTURA DE DISCO**.
- **TOTAL: 4 LECTURAS DE DISCO**.

Directorios en FAT

Ejercicio

Dibujito



Menú para hoy

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.
- Las entradas de directorio son de longitud variable, permitiendo tener nombres de archivo más grandes.

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.
- Las entradas de directorio son de longitud variable, permitiendo tener nombres de archivo más grandes.
  - Ojo: una entrada puede estar repartida en más de un bloque.

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.
- Las entradas de directorio son de longitud variable, permitiendo tener nombres de archivo más grandes.
  - Ojo: una entrada puede estar repartida en más de un bloque.
- Cada entrada contiene la longitud de la entrada, el nombre del archivo, y el número de inodo al que refiere. El resto de metadatos están en cada inodo.

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.
- Las entradas de directorio son de longitud variable, permitiendo tener nombres de archivo más grandes.
  - Ojo: una entrada puede estar repartida en más de un bloque.
- Cada entrada contiene la longitud de la entrada, el nombre del archivo, y el número de inodo al que refiere. El resto de metadatos están en cada inodo.
- Entradas de directorio especiales:

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.
- Las entradas de directorio son de longitud variable, permitiendo tener nombres de archivo más grandes.
  - Ojo: una entrada puede estar repartida en más de un bloque.
- Cada entrada contiene la longitud de la entrada, el nombre del archivo, y el número de inodo al que refiere. El resto de metadatos están en cada inodo.
- Entradas de directorio especiales:
  - . $\rightarrow$ refiere al directorio actual.

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.
- Las entradas de directorio son de longitud variable, permitiendo tener nombres de archivo más grandes.
  - Ojo: una entrada puede estar repartida en más de un bloque.
- Cada entrada contiene la longitud de la entrada, el nombre del archivo, y el número de inodo al que refiere. El resto de metadatos están en cada inodo.
- Entradas de directorio especiales:
  - . $\rightarrow$ refiere al directorio actual.
  - .. $\rightarrow$ refiere al directorio padre.

Directorios en Ext2

- En ext2, a cada directorio le corresponde un inodo.
- Las entradas de directorio son de longitud variable, permitiendo tener nombres de archivo más grandes.
  - Ojo: una entrada puede estar repartida en más de un bloque.
- Cada entrada contiene la longitud de la entrada, el nombre del archivo, y el número de inodo al que refiere. El resto de metadatos están en cada inodo.
- Entradas de directorio especiales:
  - . $\rightarrow$ refiere al directorio actual.
  - .. $\rightarrow$ refiere al directorio padre.
- Al igual que en FAT, el inodo del directorio **root** es distinguido: es siempre el inodo número 2.

Enunciado

Contamos con dos sistemas de archivos basados en inodos: FSA, con bloques de 1KiB; y FSB, con bloques de 2KiB. Ambos tienen 2 entradas directas, 2 indirectas simples, 1 doble indirecta y 1 triple indirecta. Las direcciones de bloques ocupan 2 bytes.
Indicar a cuántos bloques de disco hay que acceder para leer los bytes que se indican para cada archivo:

1. FSA: archivo: `/quiero.txt`. Bytes [0-5], [12-17].
2. FSB: archivo: `/home/aprobar.txt`. Bytes [6.500-6.600], [3.500.000].

Se puede asumir que:

- Los archivos y directorios a buscar se encuentran siempre en el primer bloque de *directory entries* correspondientes al directorio padre.
- Si un bloque se lee dos veces, se va a buscar a disco sólo la primera vez.
- Los inodos necesarios ya están cargados en memoria (el resto de los bloques a leer deberán ser contemplados).

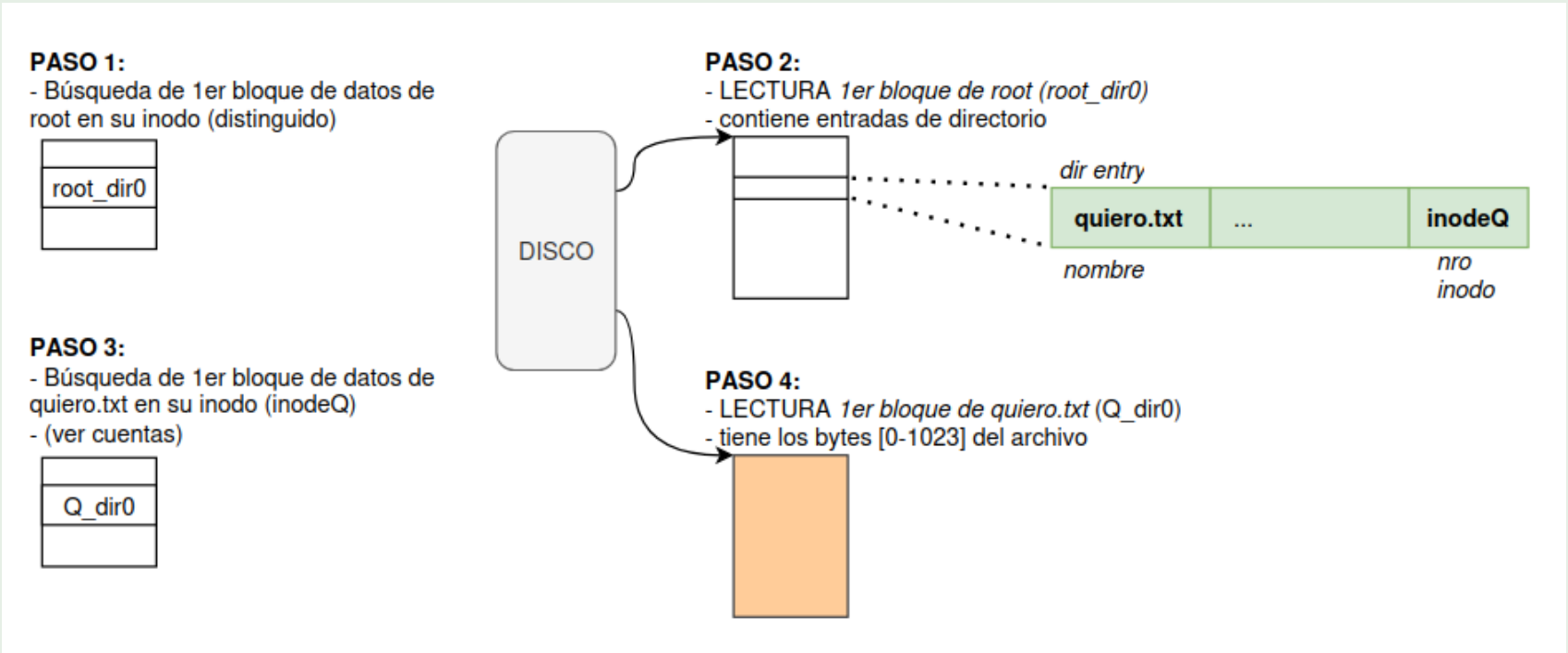
Puede dejar expresado el resultado como potencias de dos. Justificar.

Solución FSA

FSA: archivo: `/quiero.txt`. Bytes [0-5], [12-17].

- Primero tengo que saber en qué inodo buscar. Para eso miro el inodo de `root` (distinguido), traigo el primer bloque de entradas de directorio (sé que lo encuentro ahí) y busco la entrada de directorio del archivo `quiero.txt`. **1 LECTURA DE DISCO**.
- Una vez que identifico el inodo, lo miro (ya está en memoria). Quiero saber en qué bloque están los bytes que preciso. Como cada bloque es de 1KiB, la primer entrada directa apunta al bloque con los bytes [0-1023]. Ahí se encuentran los bytes solicitados. **1 LECTURA DE DISCO**.
- **TOTAL: 2 LECTURAS DE DISCO**.

FSA - Dibujito

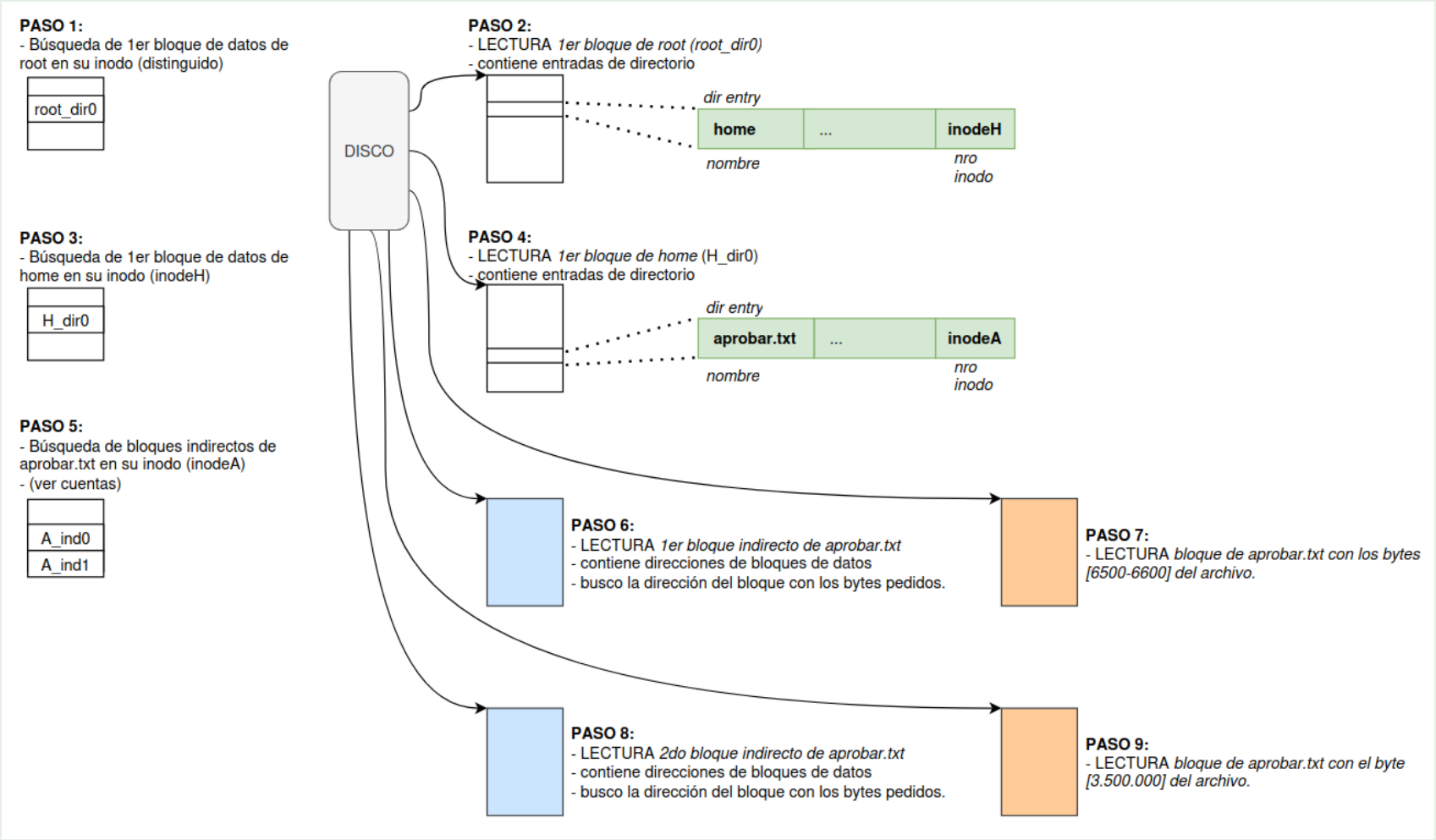


Solución FSB

FSB: archivo: `/home/aprobar.txt`. Bytes [6.500-6.600], [3.500.000].

- Para saber en qué inodo buscar,miro el inodo de `root` (distinguido), traigo el primer bloque de entradas de directorio (sé que lo encuentro ahí) y busco la entrada de directorio del archivo `home`. Luego hago lo mismo con el directorio `home` para encontrar la entrada de `aprobar.txt`. **2 LECTURAS DE DISCO**.
- Una vez que identifico el inodo (ya está en memoria), quiero saber en qué bloque están los bytes que preciso. Como cada bloque es de 2KiB, las dos primeras entradas directas apuntan a los bloques con los bytes [0-2047] y [2048-4095]. Como no me alcanza, tengo que mirar el primer indirecto.
- Como cada bloque es de 2KiB y las direcciones de bloque son de 2 Bytes, tenemos 2^{10} direcciones por bloque. Esto significa que tengo $2^{10} \cdot 2 \cdot 2^{10} = 2^{21}$ bytes direccionados en el primer indirecto. Ahí se encuentran los bytes [6.500-6.600]. **2 LECTURAS DE DISCO**
- Para el siguiente byte solicitado, el primer indirecto no me alcanza. El segundo direcciona los siguientes 2^{21} bytes y ahí está el byte [3.500.000]. **2 LECTURAS DE DISCO**.
- **TOTAL: 6 LECTURAS DE DISCO**.

FSB - Dibujito



Menú para hoy

Menú para hoy

Enlaces

Hard links

- Syscall `link()`, comando `ln`.

Enlaces

Hard links

- Syscall `link()`, comando `ln`.
- Los enlaces duros (hard links) crean otro nombre para el mismo inodo, sin duplicar los datos.

Enlaces

Hard links

- Syscall `link()`, comando `ln`.
- Los enlaces duros (hard links) crean otro nombre para el mismo inodo, sin duplicar los datos.
- Sólo es posible en los sistemas de archivos con inodos, porque el nombre de los archivos no aparece en los inodos. Así, podemos referenciar el mismo inodo con diferentes nombres, y desde más de un directorio.

Enlaces

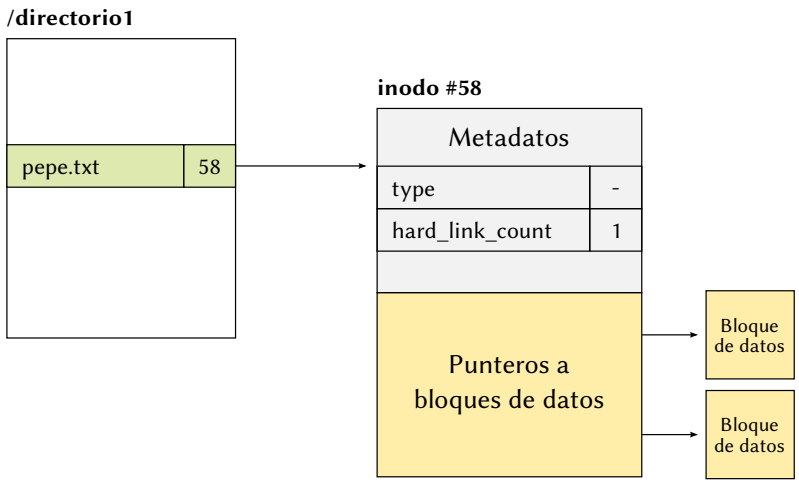
Hard links

- Syscall `link()`, comando `ln`.
- Los enlaces duros (hard links) crean otro nombre para el mismo inodo, sin duplicar los datos.
- Sólo es posible en los sistemas de archivos con inodos, porque el nombre de los archivos no aparece en los inodos. Así, podemos referenciar el mismo inodo con diferentes nombres, y desde más de un directorio.
- El nombre de archivo "." en un directorio es un enlace duro al mismo directorio. El nombre de archivo ".." es un enlace duro al directorio padre.

Enlaces

Hard links

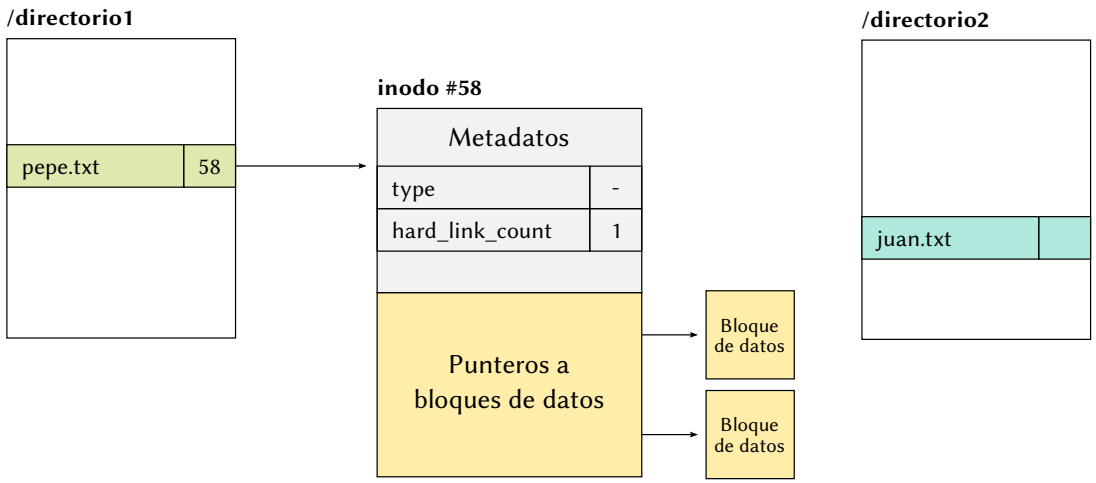
- Syscall `link()`, comando `ln`.
- Los enlaces duros (hard links) crean otro nombre para el mismo inodo, sin duplicar los datos.
- Sólo es posible en los sistemas de archivos con inodos, porque el nombre de los archivos no aparece en los inodos. Así, podemos referenciar el mismo inodo con diferentes nombres, y desde más de un directorio.
- El nombre de archivo "." en un directorio es un enlace duro al mismo directorio. El nombre de archivo ".." es un enlace duro al directorio padre.



Enlaces

Hard links

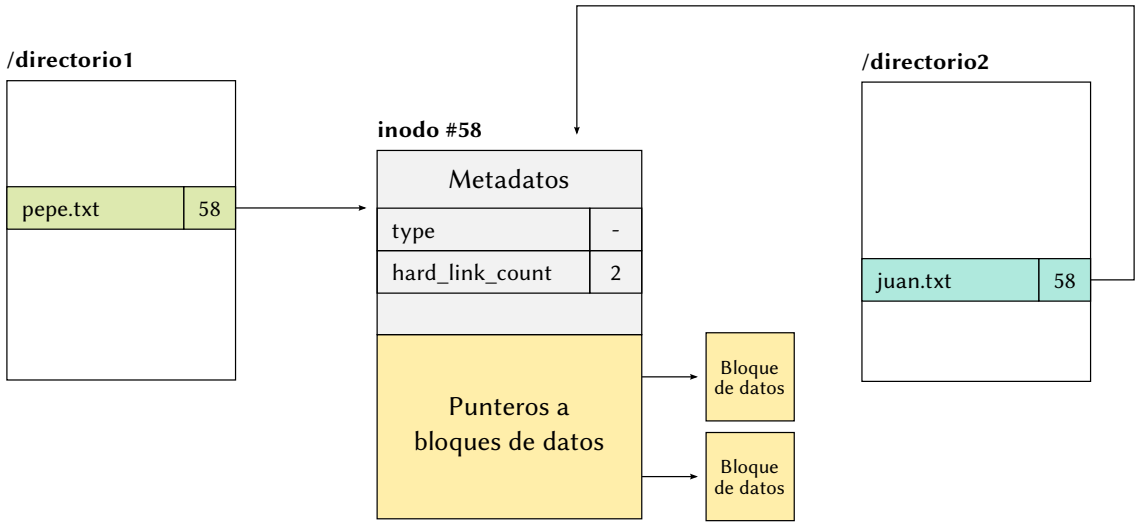
- Syscall `link()`, comando `ln`.
- Los enlaces duros (hard links) crean otro nombre para el mismo inodo, sin duplicar los datos.
- Sólo es posible en los sistemas de archivos con inodos, porque el nombre de los archivos no aparece en los inodos. Así, podemos referenciar el mismo inodo con diferentes nombres, y desde más de un directorio.
- El nombre de archivo "." en un directorio es un enlace duro al mismo directorio. El nombre de archivo ".." es un enlace duro al directorio padre.



Enlaces

Hard links

- Syscall `link()`, comando `ln`.
- Los enlaces duros (hard links) crean otro nombre para el mismo inodo, sin duplicar los datos.
- Sólo es posible en los sistemas de archivos con inodos, porque el nombre de los archivos no aparece en los inodos. Así, podemos referenciar el mismo inodo con diferentes nombres, y desde más de un directorio.
- El nombre de archivo "." en un directorio es un enlace duro al mismo directorio. El nombre de archivo ".." es un enlace duro al directorio padre.



Enlaces

Hard links

- ¿Cómo hacemos para borrar un archivo que puede tener enlaces?

Enlaces

Hard links

- ¿Cómo hacemos para borrar un archivo que puede tener enlaces?
  - Podemos preservar el archivo hasta que se borren todas las referencias.

Enlaces

Hard links

- ¿Cómo hacemos para borrar un archivo que puede tener enlaces?
  - Podemos preservar el archivo hasta que se borren todas las referencias.
- ¿Cómo sé si ya borré todas las referencias?

Enlaces

Hard links

- ¿Cómo hacemos para borrar un archivo que puede tener enlaces?
  - Podemos preservar el archivo hasta que se borren todas las referencias.
- ¿Cómo sé si ya borré todas las referencias?
  - Se mantiene la cuenta de todas las referencias al archivo en el inodo. Cuando se crea un enlace, se incrementa el contador. Cuando un enlace se borra, se decrementa el contador. El archivo se borra cuando el contador está en cero.

Menú para hoy

Enlaces

Symbolic links

- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.

Enlaces

Symbolic links

- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.
- Los enlaces simbólicos (symbolic links) crean un archivo separado (con su inodo) que almacena el *path* al archivo original.

Enlaces

Symbolic links

- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.
- Los enlaces simbólicos (symbolic links) crean un archivo separado (con su inodo) que almacena el *path* al archivo original.
- Comando `ln -s`.

Enlaces

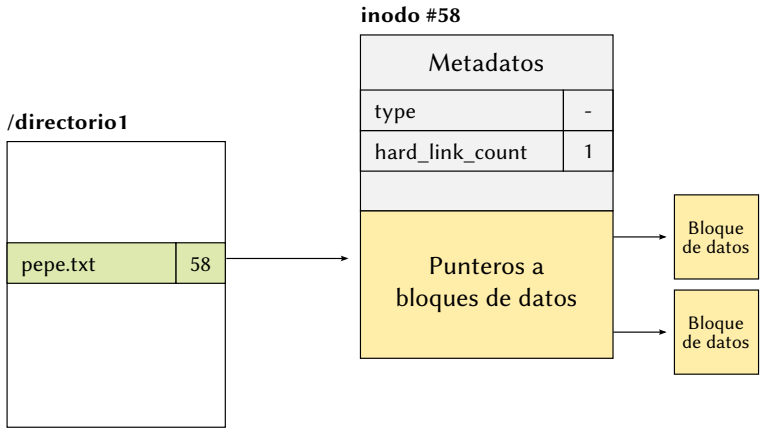
Symbolic links

- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.
- Los enlaces simbólicos (symbolic links) crean un archivo separado (con su inodo) que almacena el *path* al archivo original.
- Comando `ln -s`.
- Permiten referenciar directorios en otros sistemas de archivos.

Enlaces

Symbolic links

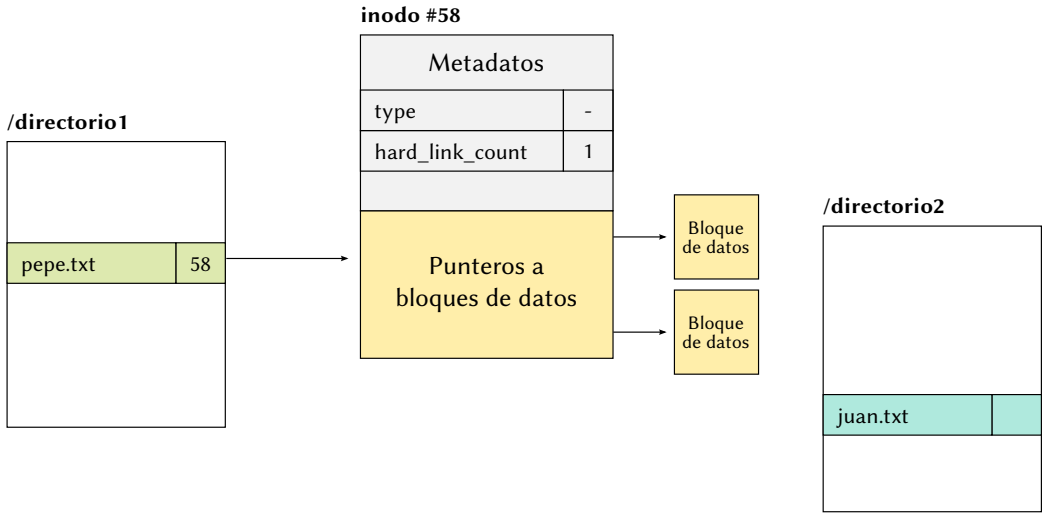
- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.
- Los enlaces simbólicos (symbolic links) crean un archivo separado (con su inodo) que almacena el *path* al archivo original.
- Comando `ln -s`.
- Permiten referenciar directorios en otros sistemas de archivos.



Enlaces

Symbolic links

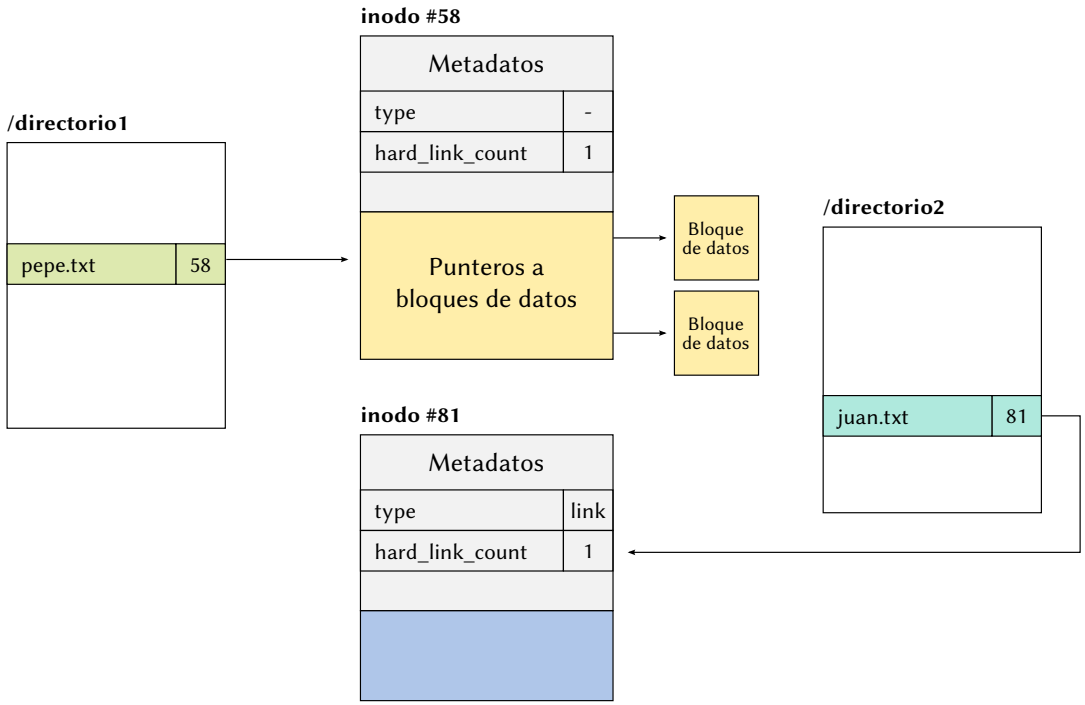
- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.
- Los enlaces simbólicos (symbolic links) crean un archivo separado (con su inodo) que almacena el *path* al archivo original.
- Comando `ln -s`.
- Permiten referenciar directorios en otros sistemas de archivos.



Enlaces

Symbolic links

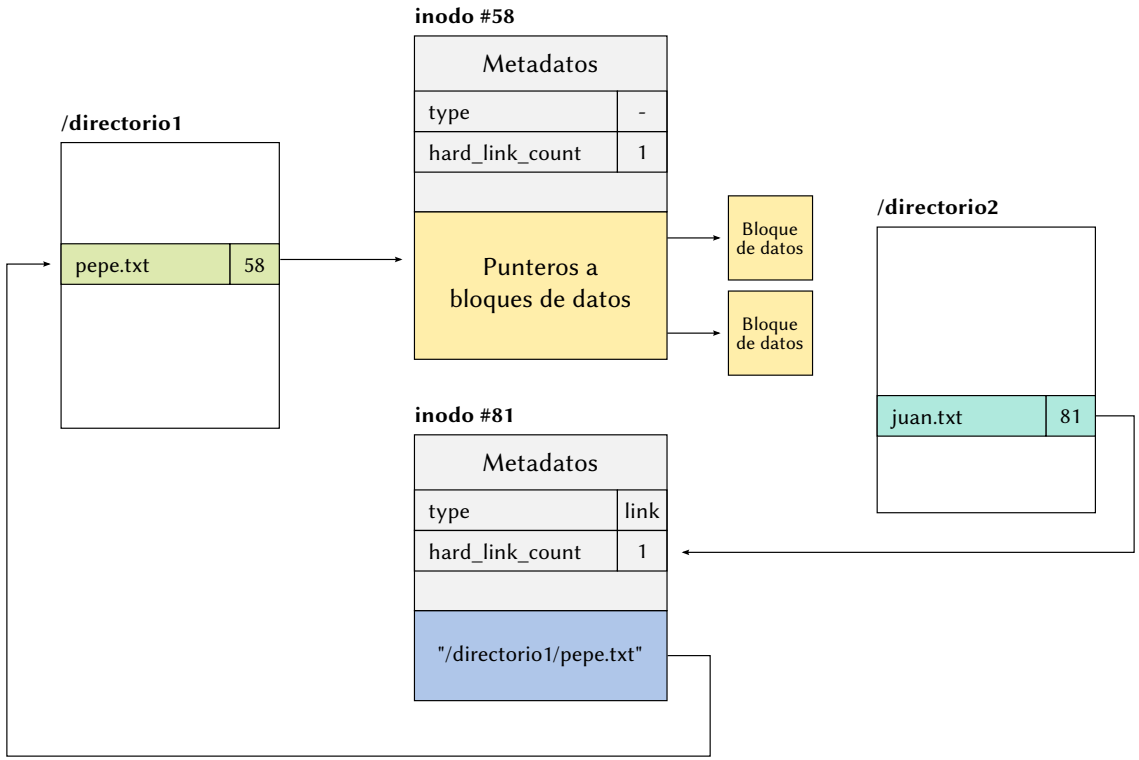
- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.
- Los enlaces simbólicos (symbolic links) crean un archivo separado (con su inodo) que almacena el *path* al archivo original.
- Comando `ln -s`.
- Permiten referenciar directorios en otros sistemas de archivos.



Enlaces

Symbolic links

- Los *hard links* no pueden hacerse sobre directorios o entre distintos sistemas de archivos.
- Los enlaces simbólicos (symbolic links) crean un archivo separado (con su inodo) que almacena el *path* al archivo original.
- Comando `ln -s`.
- Permiten referenciar directorios en otros sistemas de archivos.



Enlaces

Symbolic links

- ¿Qué pasa si borramos un enlace simbólico?

Enlaces

Symbolic links

- ¿Qué pasa si borramos un enlace simbólico?
  - No se lleva una cuenta de los enlaces simbólicos. Si se borra el enlace, el archivo original sigue igual.

Enlaces

Symbolic links

- ¿Qué pasa si borramos un enlace simbólico?
  - No se lleva una cuenta de los enlaces simbólicos. Si se borra el enlace, el archivo original sigue igual.
- ¿Qué pasa si borramos un archivo que está siendo referenciado por un enlace simbólico?

Enlaces

Symbolic links

- ¿Qué pasa si borramos un enlace simbólico?
  - No se lleva una cuenta de los enlaces simbólicos. Si se borra el enlace, el archivo original sigue igual.
- ¿Qué pasa si borramos un archivo que está siendo referenciado por un enlace simbólico?
  - El archivo no sabe que hay referencias simbólicas. Si se borra el archivo, se libera el espacio correspondiente y el enlace queda roto.

Momento para preguntas

Menú para hoy

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques
  - FAT

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques
  - FAT
  - inodos

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques
  - FAT
  - inodos
- Manejo de directorios en FAT32 y Ext2.

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques
  - FAT
  - inodos
- Manejo de directorios en FAT32 y Ext2.
- Enlaces

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques
  - FAT
  - inodos
- Manejo de directorios en FAT32 y Ext2.
- Enlaces

Resumen

Hoy vimos...

- Distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques
  - FAT
  - inodos
- Manejo de directorios en FAT32 y Ext2.
- Enlaces

Cómo seguimos...

- Con esto se puede resolver toda la guía práctica 6.
- Próxima clase: taller Ext2