Sistemas de Archivos

Departamento de Computación - FCEyN

21 de mayo de 2024

Estructura de la clase

Repaso

- Un **sistema de archivos** nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.

Repaso

- Un **sistema de archivos** nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.
- Partes de un sistema de archivos:

Repaso

- Un **sistema de archivos** nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.
- Partes de un sistema de archivos:
  - Archivos que almacenan datos.

Repaso

- Un **sistema de archivos** nos permite administrar y ordenar los archivos dentro de un medio de almacenamiento.
- Partes de un sistema de archivos:
  - Archivos que almacenan datos.
  - Estructura de directorios para organizar los archivos.

Almacenamiento secundario

- Las lecturas y escrituras a un medio de almacenamiento se hacen en unidades llamadas **bloques**.
- Uno de los problemas que debe resolver un sistema de archivos es cómo **asignar** los bloques a los diferentes archivos.
- Cada archivo ocupa una cantidad entera de bloques. ¿Qué problema ocasiona esto?

Almacenamiento secundario

- Las lecturas y escrituras a un medio de almacenamiento se hacen en unidades llamadas **bloques**.
- Uno de los problemas que debe resolver un sistema de archivos es cómo **asignar** los bloques a los diferentes archivos.
- Cada archivo ocupa una cantidad entera de bloques. ¿Qué problema ocasiona esto? $\rightarrow$ *fragmentación interna*.

¿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
- Escribir
- Leer
- Borrar
- ...

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?

Asignación contigua

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

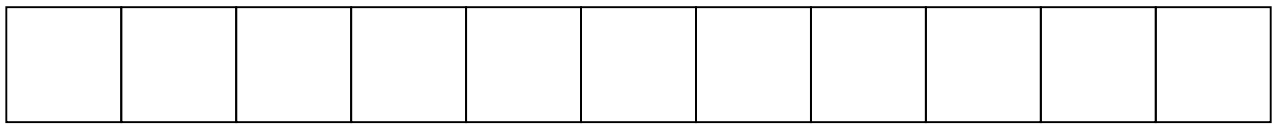
Asignación contigua

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

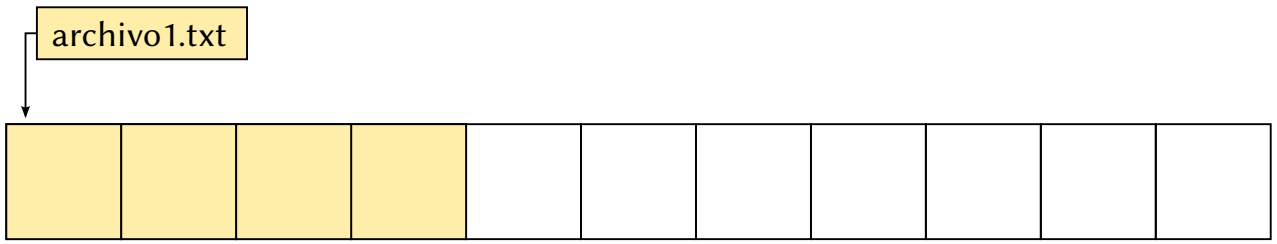
Asignación contigua

- 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 → *fragmentación externa*.

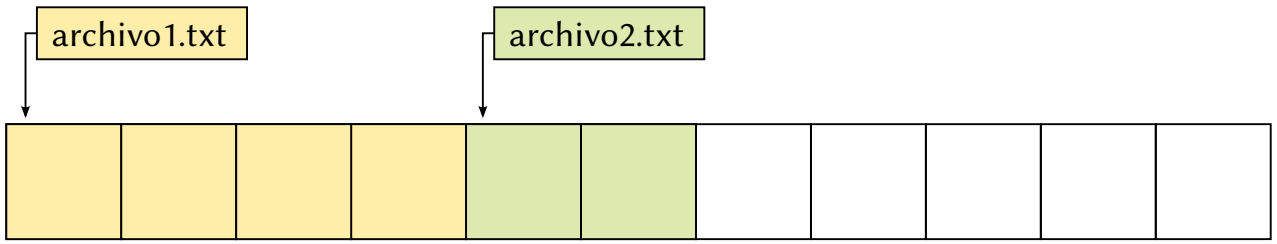
Asignación contigua: ejemplo



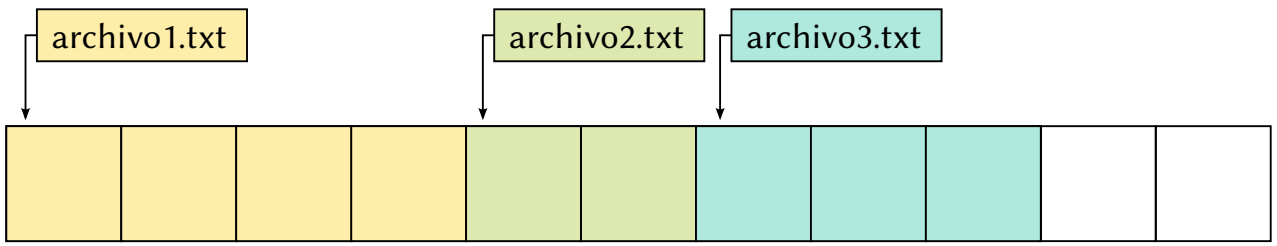
Asignación contigua: ejemplo



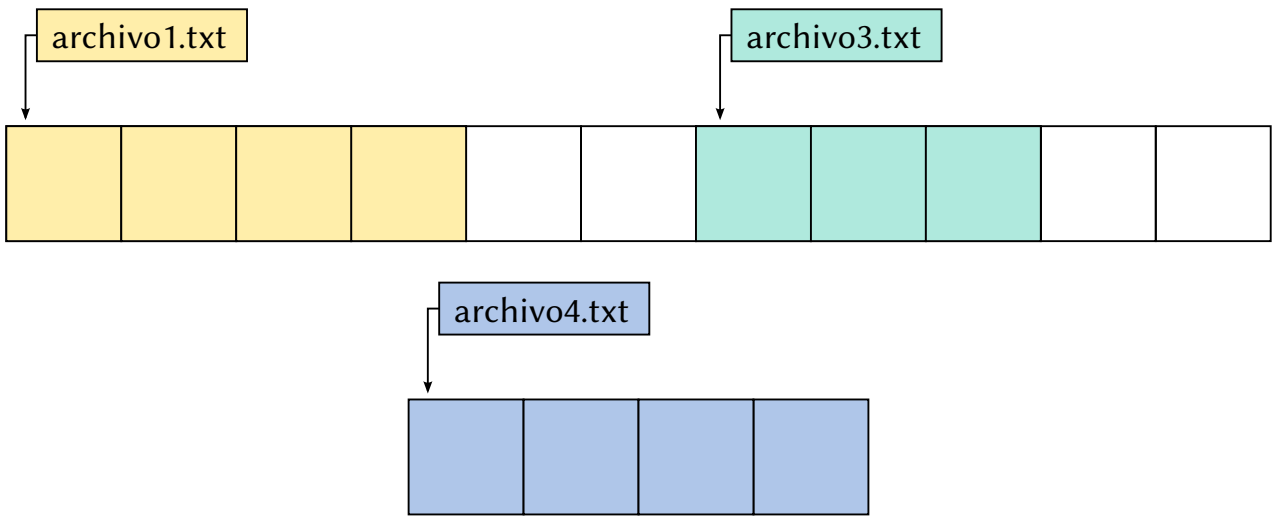
Asignación contigua: ejemplo



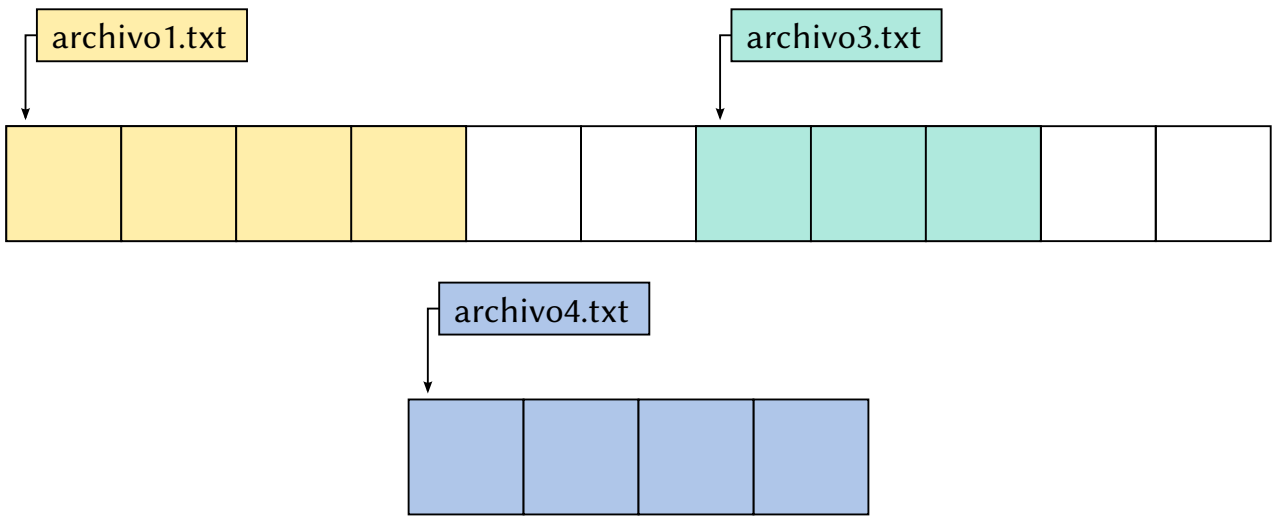
Asignación contigua: ejemplo



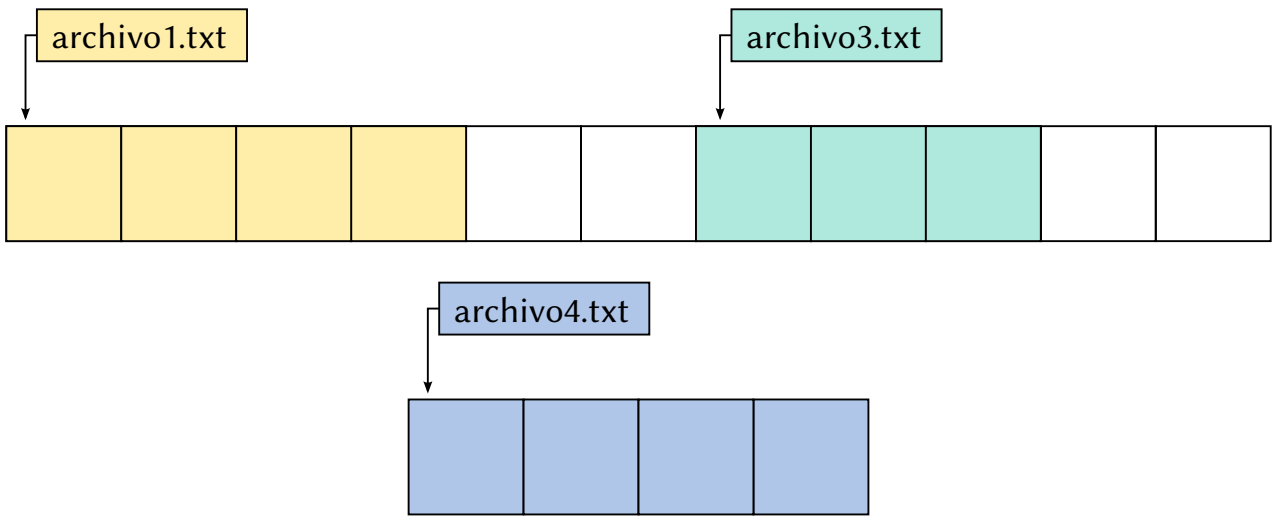
Asignación contigua: ejemplo



Asignación contigua: ejemplo



Asignación contigua: ejemplo



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

Ejercicio 1 - Contiguo

1. Se tiene un disco con bloques de 8 KB, y un sistema de archivos donde los bloques de un archivo se almacenan en forma contigua. Considerar un archivo de 10 MB, para el que se conoce en qué posición del disco se encuentra su primer bloque. Suponer que hay bloques libres luego del final del archivo, pero no antes del comienzo.

Ejercicio 1 - Contiguo

1. Se tiene un disco con bloques de 8 KB, y un sistema de archivos donde los bloques de un archivo se almacenan en forma contigua. Considerar un archivo de 10 MB, para el que se conoce en qué posición del disco se encuentra su primer bloque. Suponer que hay bloques libres luego del final del archivo, pero no antes del comienzo.
   - ¿Cuántas operaciones de disco son necesarias para...
     - **(a)** agregar un bloque al principio del archivo?
     - **(b)** agregar un bloque en la mitad del archivo?
     - **(c)** agregar un bloque al final del archivo?
     - **(d)** eliminar el primer bloque del archivo?
     - **(e)** eliminar el bloque en la mitad del archivo?
     - **(f)** eliminar el último bloque del archivo?

Ejercicio 1 - Contiguo

1. Se tiene un disco con bloques de 8 KB, y un sistema de archivos donde los bloques de un archivo se almacenan en forma contigua. Considerar un archivo de 10 MB, para el que se conoce en qué posición del disco se encuentra su primer bloque. Suponer que hay bloques libres luego del final del archivo, pero no antes del comienzo.

	Contiguo	FAT			inodos		
	Disco	Disco	Lect	Esc	Disco	Lect	Esc
add_0	2561	X	X	X	X	X	X
$add_{\frac{n}{2}}$	1281	X	X	X	X	X	X
add_n	1	X	X	X	X	X	X
del_0	2558	X	X	X	X	X	X
$del_{\frac{n}{2}}$	1278	X	X	X	X	X	X
del_n	0	X	X	X	X	X	X

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?

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? $\rightarrow$ El número de bloque del primer bloque del archivo.

Tabla de asignación de archivos (FAT): ejemplo

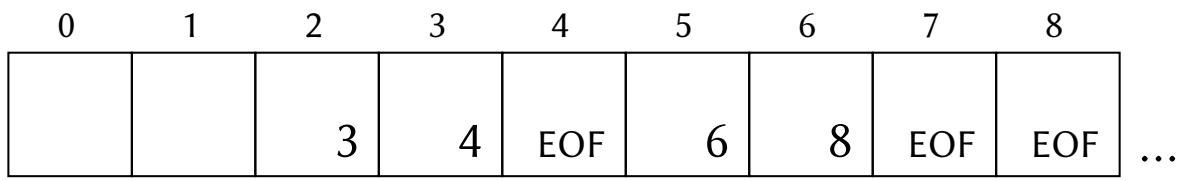


Tabla de asignación de archivos (FAT): ejemplo

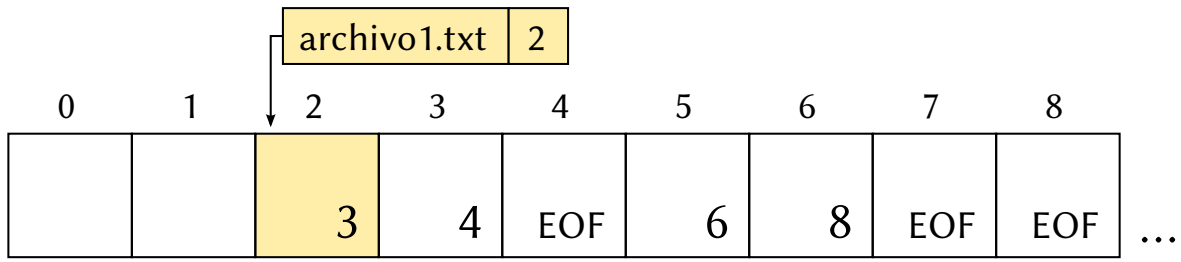


Tabla de asignación de archivos (FAT): ejemplo

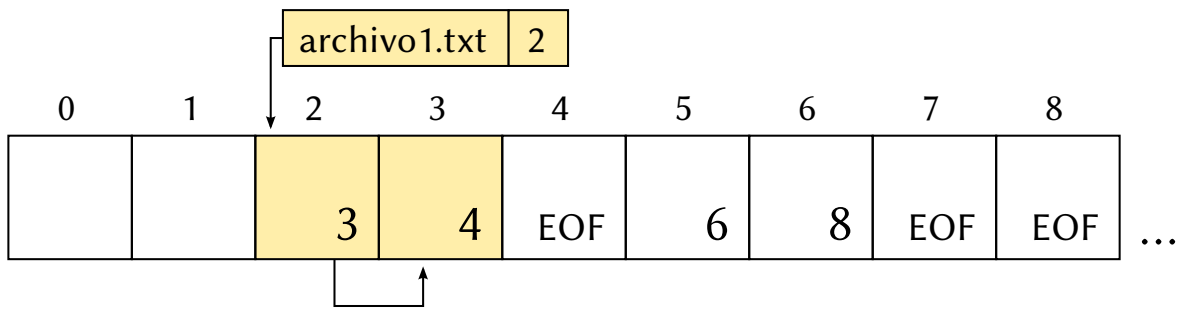


Tabla de asignación de archivos (FAT): ejemplo

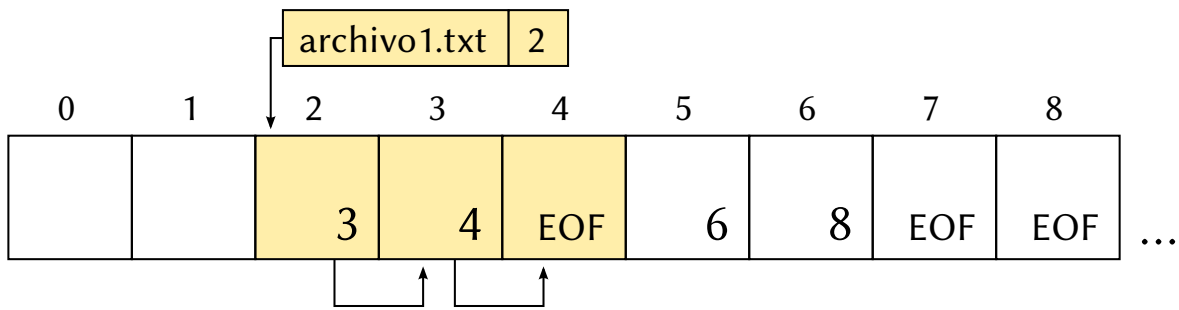


Tabla de asignación de archivos (FAT): ejemplo

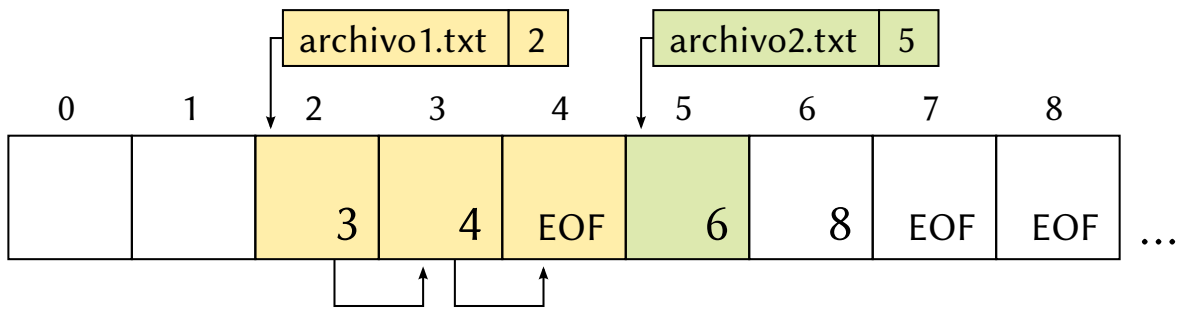


Tabla de asignación de archivos (FAT): ejemplo

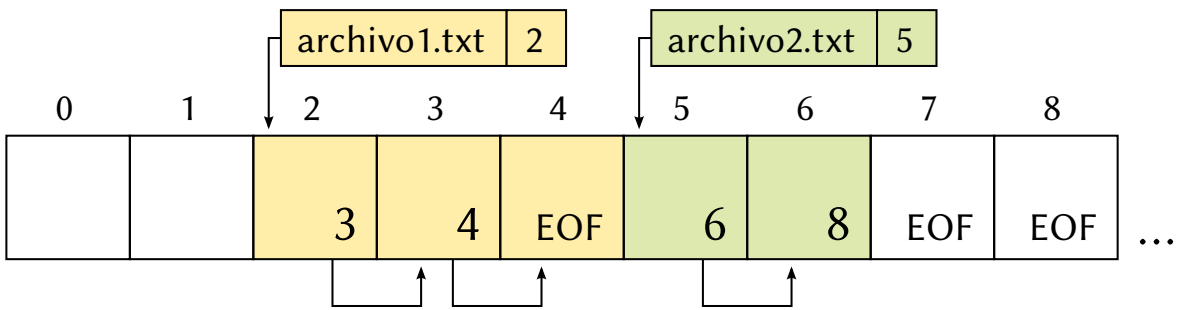


Tabla de asignación de archivos (FAT): ejemplo

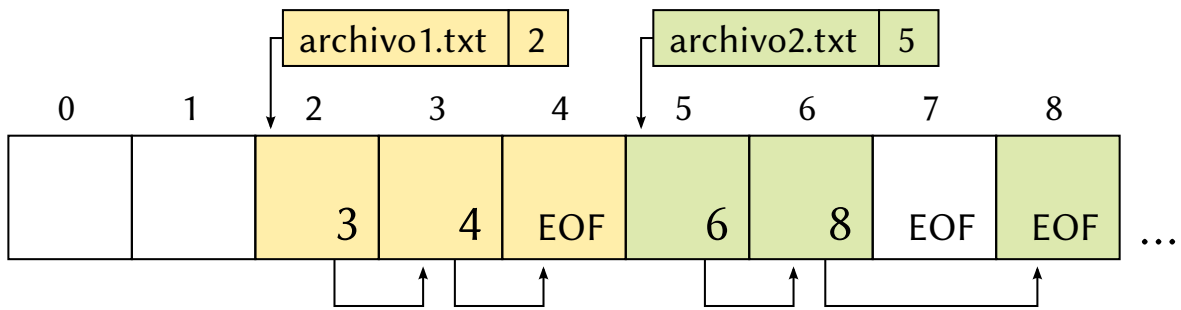
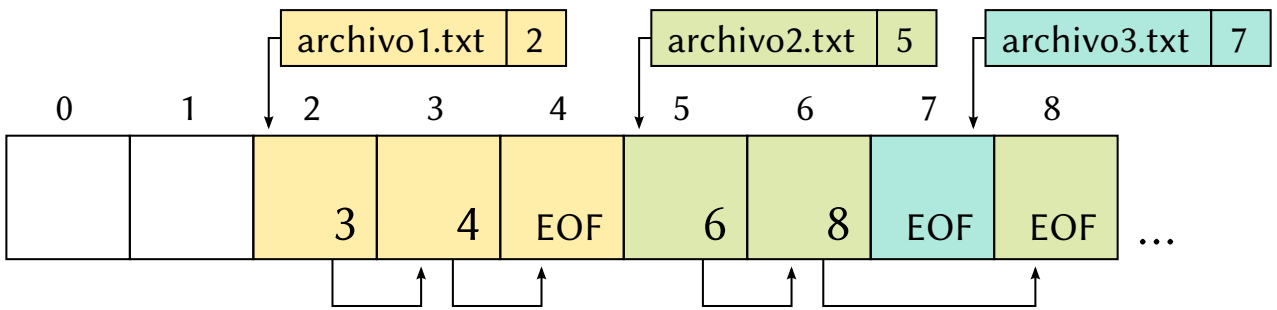


Tabla de asignación de archivos (FAT): ejemplo



Ejercicio 1 - FAT

2. Se tiene un disco con bloques de 8 KB y un sistema de archivos con una FAT, que está cargada en memoria. Considerar un archivo de 10 MB, para el que se conoce la dirección de su primer bloque.

Ejercicio 1 - FAT

2. Se tiene un disco con bloques de 8 KB y un sistema de archivos con una FAT, que está cargada en memoria. Considerar un archivo de 10 MB, para el que se conoce la dirección de su primer bloque.
   - ¿Cuántas operaciones de disco son necesarias para...
     (a) agregar un bloque al principio del archivo?
     (b) agregar un bloque en la mitad del archivo?
     (c) agregar un bloque al final del archivo?
     (d) eliminar el primer bloque del archivo?
     (e) eliminar el bloque en la mitad del archivo?
     (f) eliminar el último bloque del archivo?

     ¿Cuántas posiciones de la FAT deben consultarse y cuántas deben modificarse en cada caso?
   - Comparar con los resultados obtenidos para la asignación contigua.

Ejercicio 1 - FAT

2. Se tiene un disco con bloques de 8 KB y un sistema de archivos con una FAT, que está cargada en memoria. Considerar un archivo de 10 MB, para el que se conoce la dirección de su primer bloque.

	Contiguo	FAT			inodos		
	Disco	Disco	Lect	Esc	Disco	Lect	Esc
add_0	2561	1	0	1	X	X	X
$add_{\frac{n}{2}}$	1281	1	640	2	X	X	X
add_n	1	1	1279	2	X	X	X
del_0	2558	0	1	0	X	X	X
$del_{\frac{n}{2}}$	1278	0	641	1	X	X	X
del_n	0	0	1278	1	X	X	X

Tabla de asignación de archivos (FAT): Problemas

- Es poco eficiente cuando el acceso a los bloques no es secuencial.
- Tener que cargar toda la tabla en memoria es problemático.

Directorios

- ¿Cómo sabemos, a partir de su nombre, dónde está almacenado un archivo?

Directorios

- ¿Cómo sabemos, a partir de su nombre, dónde está almacenado un archivo? Gracias a los **directorios**.

Directorios

- ¿Cómo sabemos, a partir de su nombre, dónde está almacenado un archivo? Gracias a los **directorios**.

Los **directorios** también son archivos. Consisten en una tabla con una entrada por cada archivo que contienen, indicando su nombre y su posición física en el disco.

Directorios

- ¿Cómo sabemos, a partir de su nombre, dónde está almacenado un archivo? Gracias a los **directorios**.

Los **directorios** también son archivos. Consisten en una tabla con una entrada por cada archivo que contienen, indicando su nombre y su posición física en el disco.

- Un directorio puede contener subdirectorios. Así, podemos organizar los archivos en una *estructura jerárquica*, mediante un árbol de directorios.

Directorios en FAT

- Indican el índice del **primer bloque** de cada archivo.

Directorios en FAT

- Indican el índice del **primer bloque** de cada archivo.
- Contienen todos los **metadatos**: nombre, tamaño, fecha de último acceso, etc.

Directorios en FAT

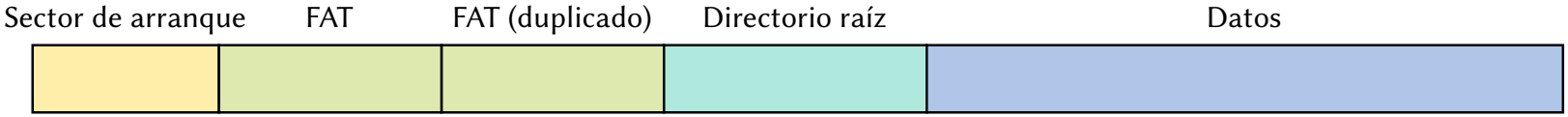
- Indican el índice del **primer bloque** de cada archivo.
- Contienen todos los **metadatos**: nombre, tamaño, fecha de último acceso, etc.
- El bloque del directorio **root** es distinguido. De esta forma, podemos encontrar cualquier archivo a partir de su **ruta**.

Directorios en FAT

- Indican el índice del **primer bloque** de cada archivo.
- Contienen 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**.

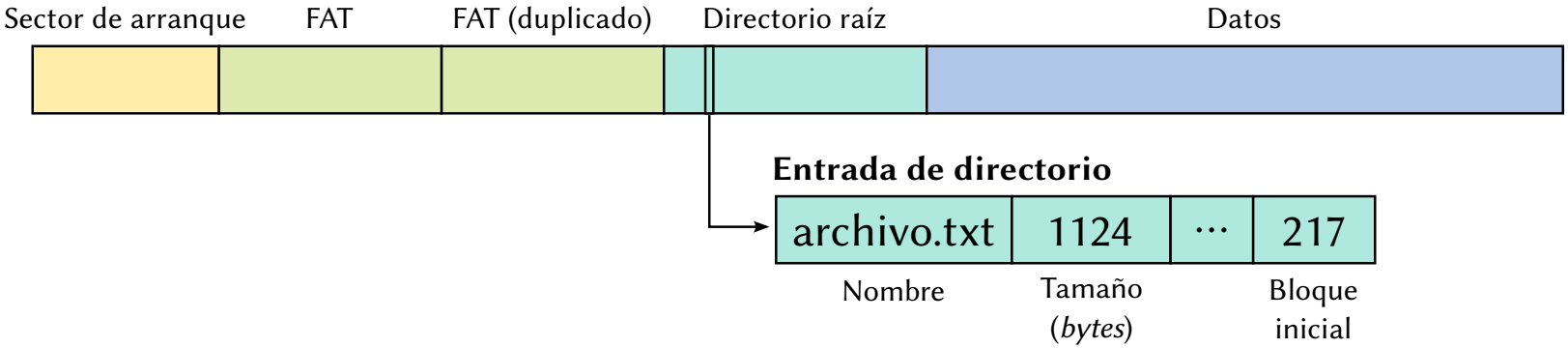
Estructura de un sistema de archivos FAT32.

Directorios en FAT: obteniendo un archivo(*)



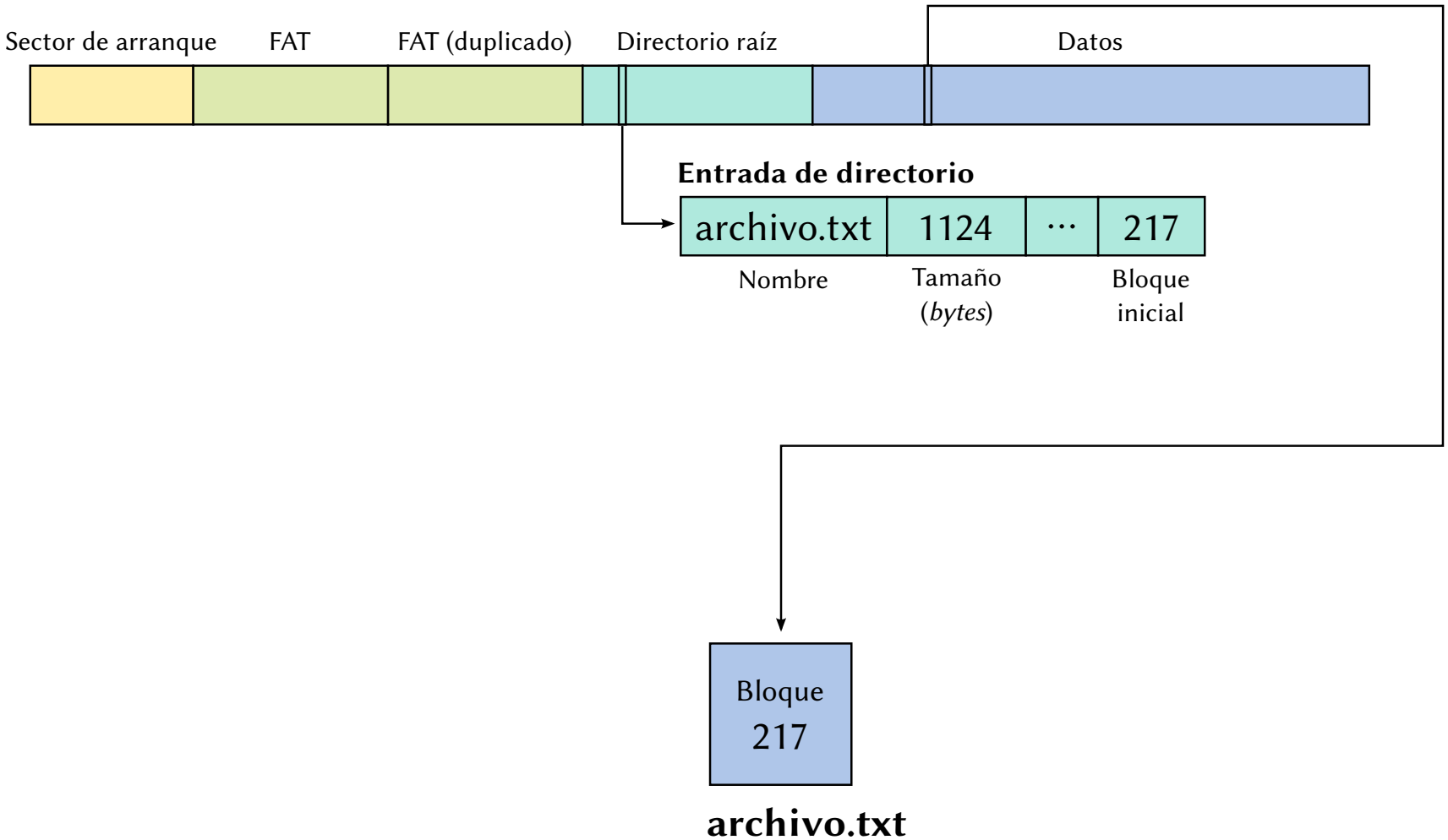
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



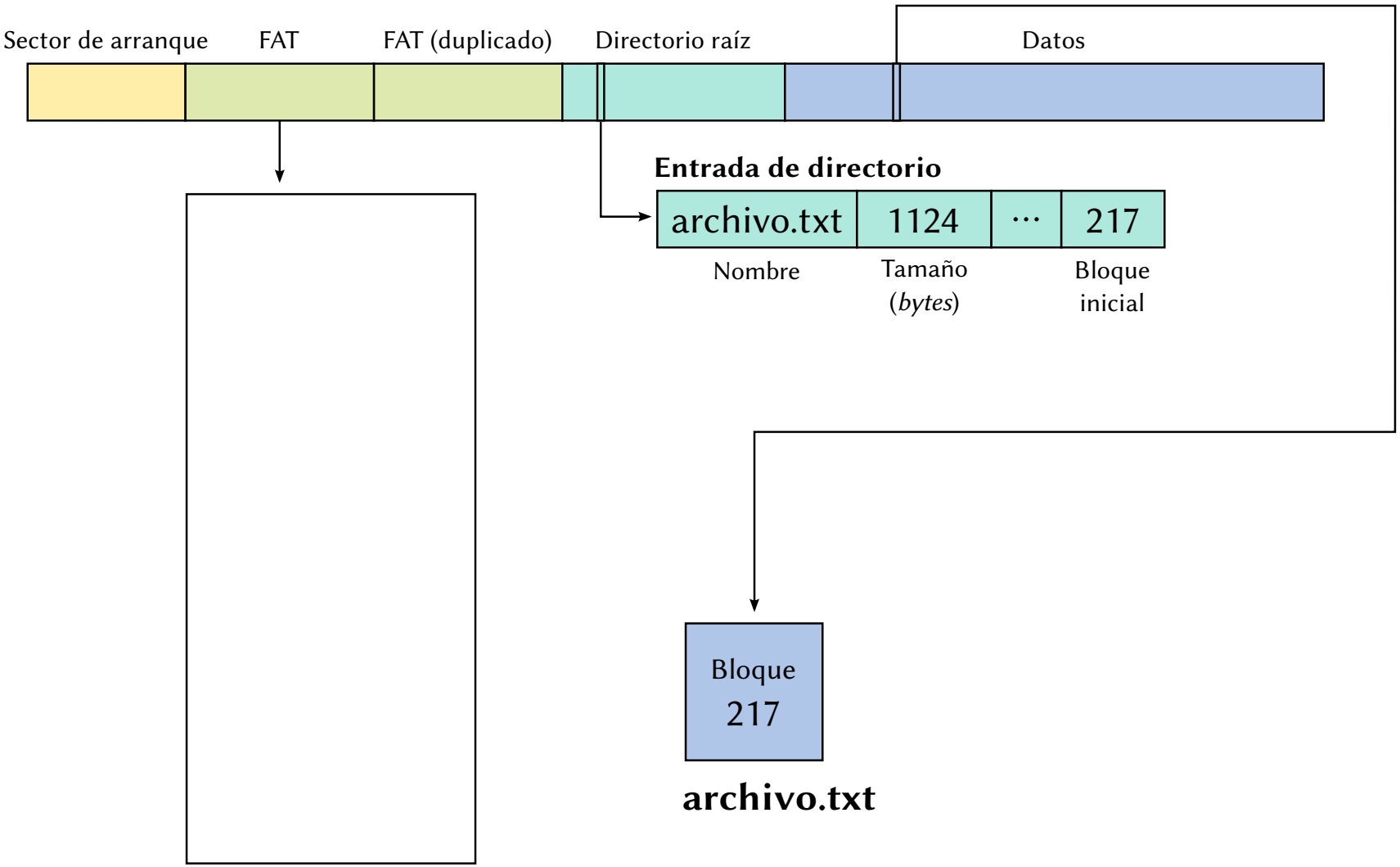
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



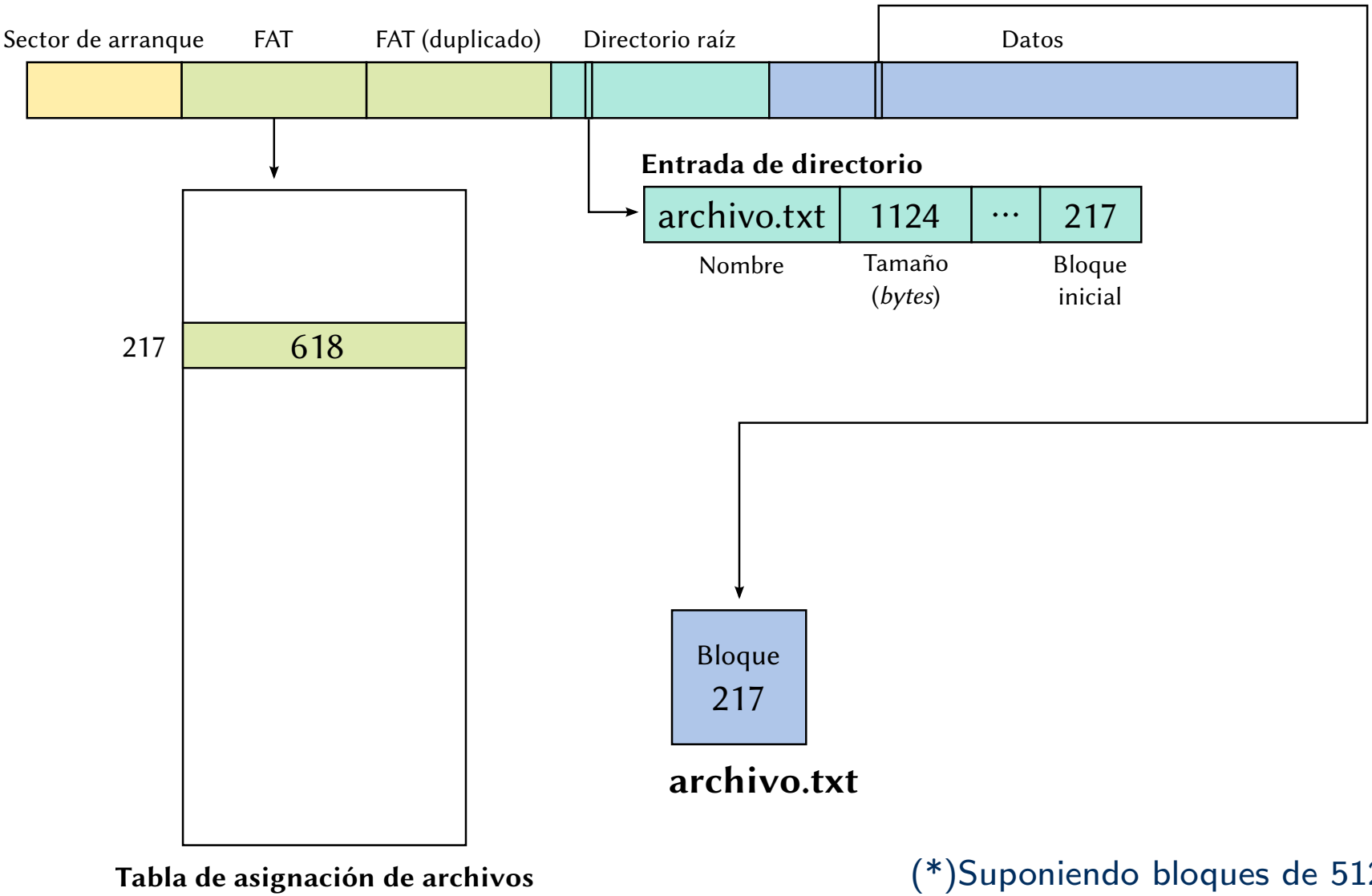
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



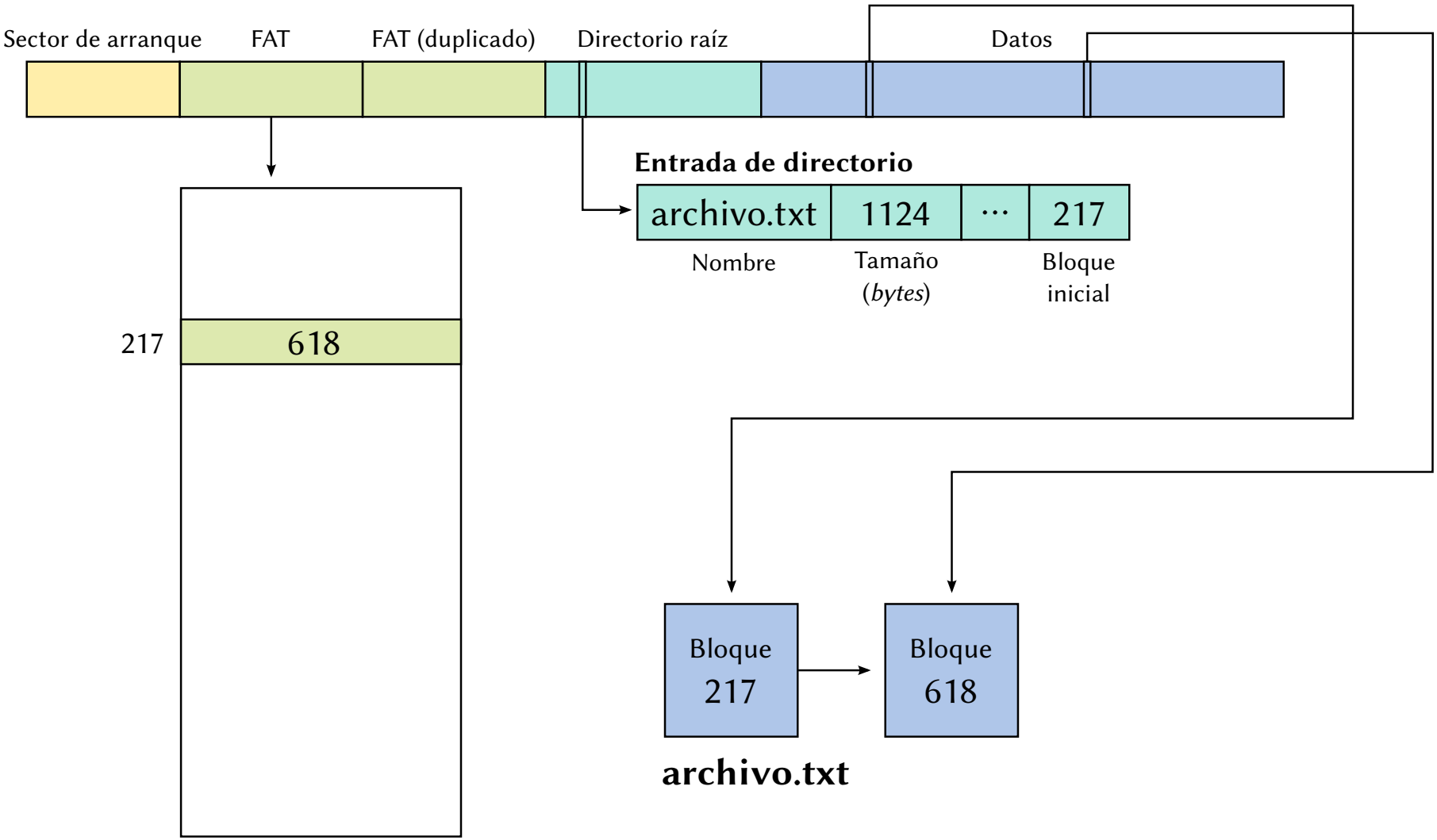
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



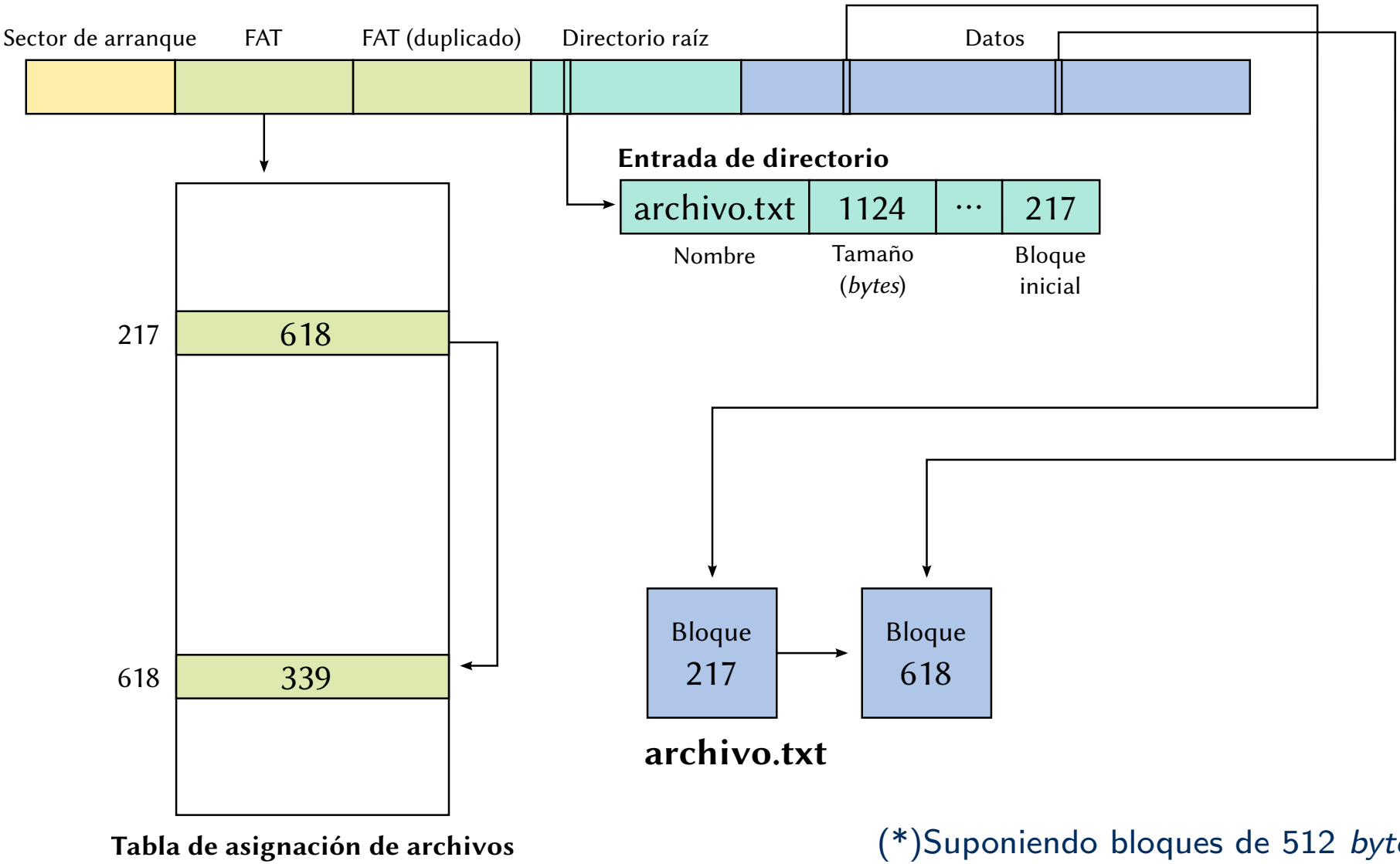
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



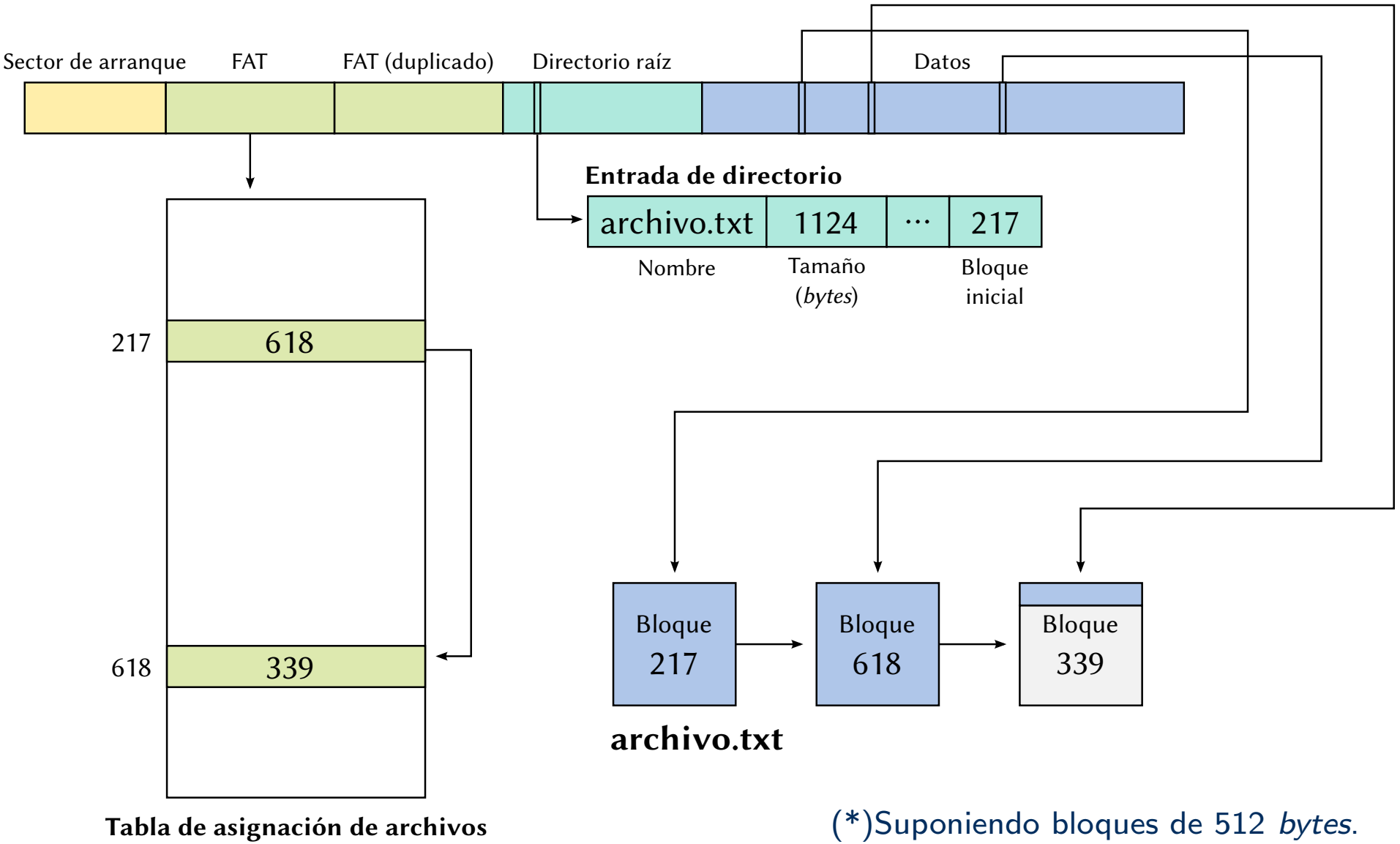
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



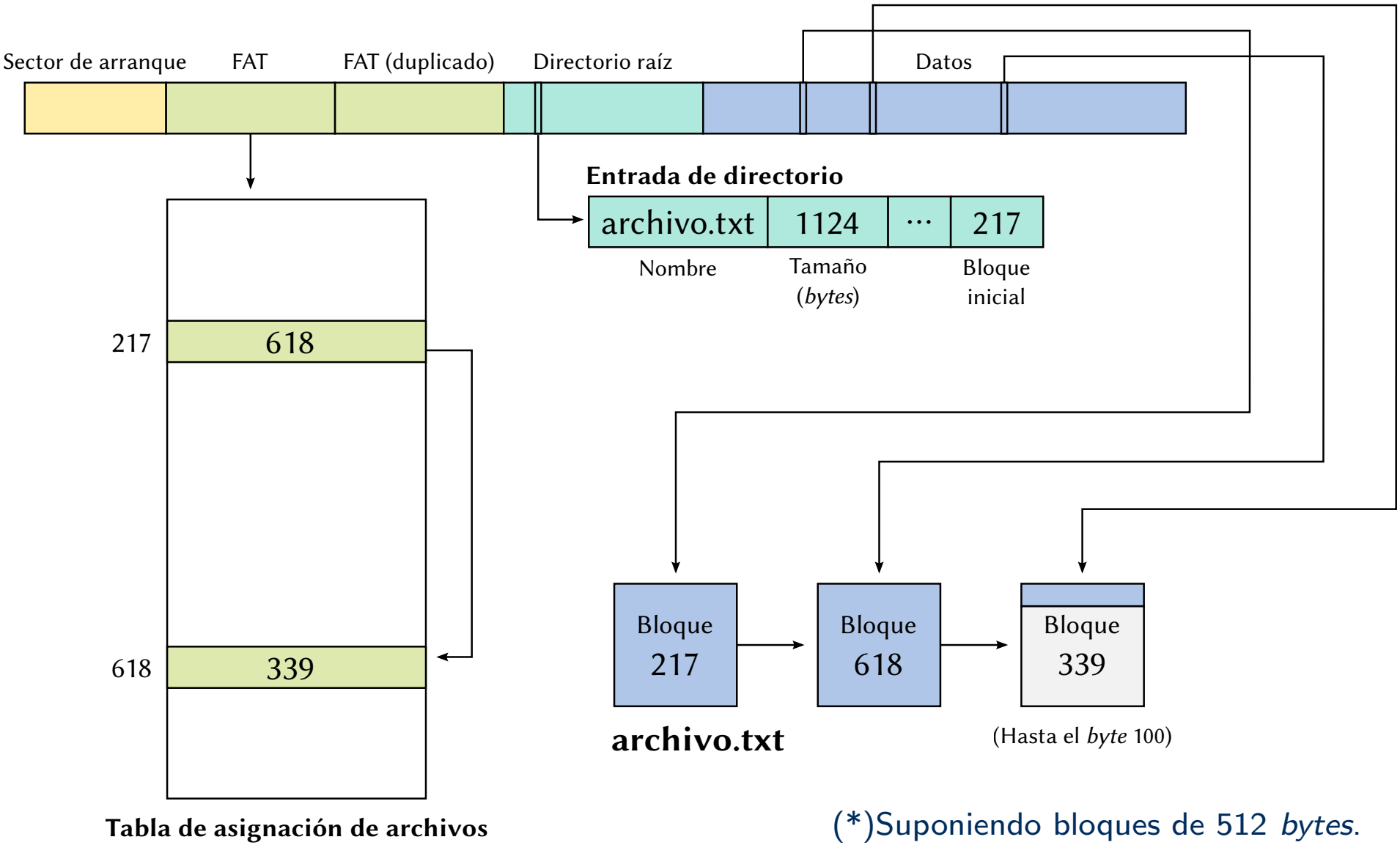
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



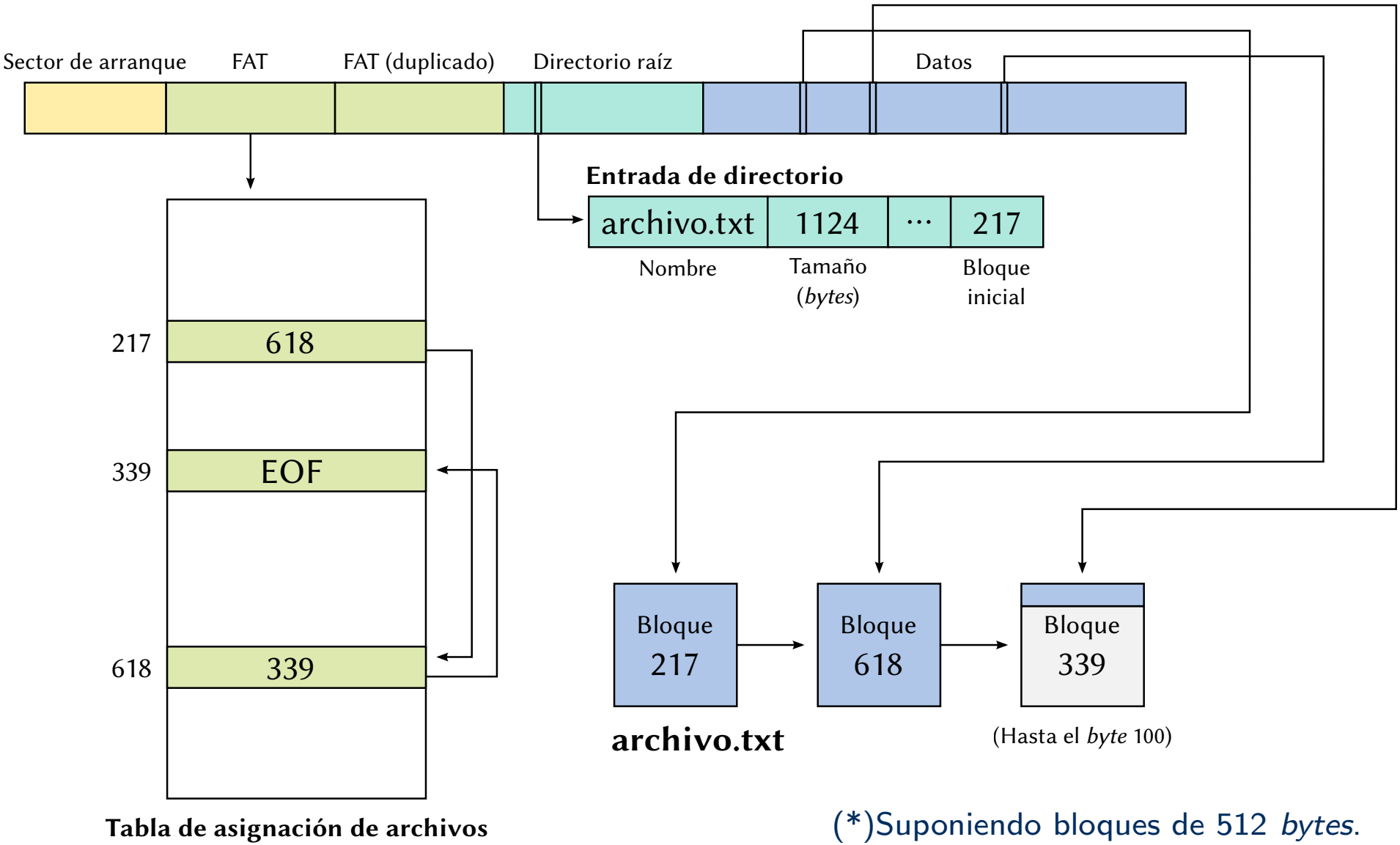
(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



(*)Suponiendo bloques de 512 *bytes*.

Directorios en FAT: obteniendo un archivo(*)



(*)Suponiendo bloques de 512 *bytes*.

Ejercicio 2

Contamos un sistema de archivos:

- FSC: formateado con FAT12 con bloques de 2KiB (con clusters de 1 bloque)

Indicar, para cada el sistemas de archivos mencionados (FSC), a cuántos bloques tengo que acceder en disco para leer, del archivo mencionado, los bytes de cada listado mencionado debajo.
Nos indican que los archivos y directorios a buscar se encuentran siempre en el primer bloque de directories entries correspondientes al directorio padre.
Asumir que si un bloque se lee dos veces en el mismo filesystem se va a buscar una sola vez al disco y que, la FAT del archivo que estaremos leyendo y el boot sector ya están cargados en memoria (el resto de los bloques a leer deberán ser contemplados). Justificar claramente como llega a esa conclusión.

Ejercicio 2

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.

1. FSC: archivo: '/home/el/parcial.avi'. Bytes [3.000-3.083].

Puede dejar expresado el resultado como cuentas de potencias de dos, pero deberá explicar claramente dichas cuentas.

Solucion - Ejercicio 2

FSC: archivo: '/home/el/parcial.avi'. Bytes [3.000-3.083].

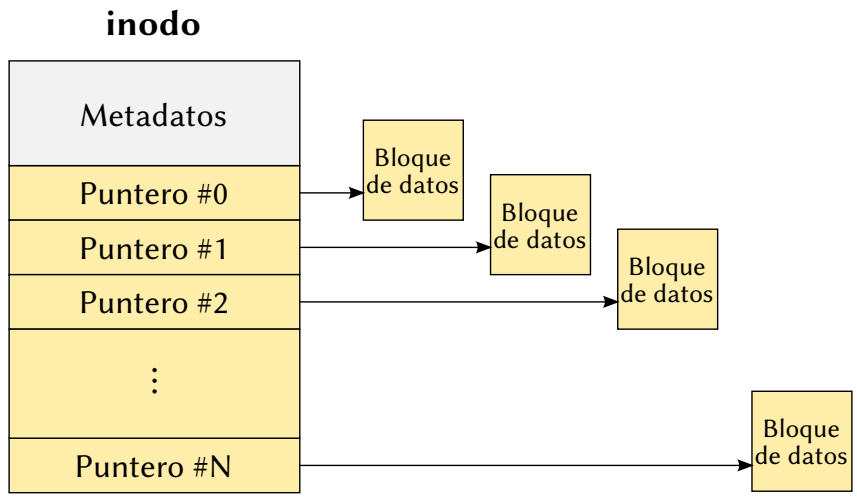
- Tengo la entrada de root (distinguido), busco en la FAT el 1er bloque de la tabla de directorios para hallar la entrada de home. Cuando la encuentro, busco en la FAT el 1er bloque de la tabla de directorios para hallar la entrada de el. Lo mismo para parcial.avi. **3 LECTURAS DE DISCO**.
- Ahora miro 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**.

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.
- La iésima 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

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

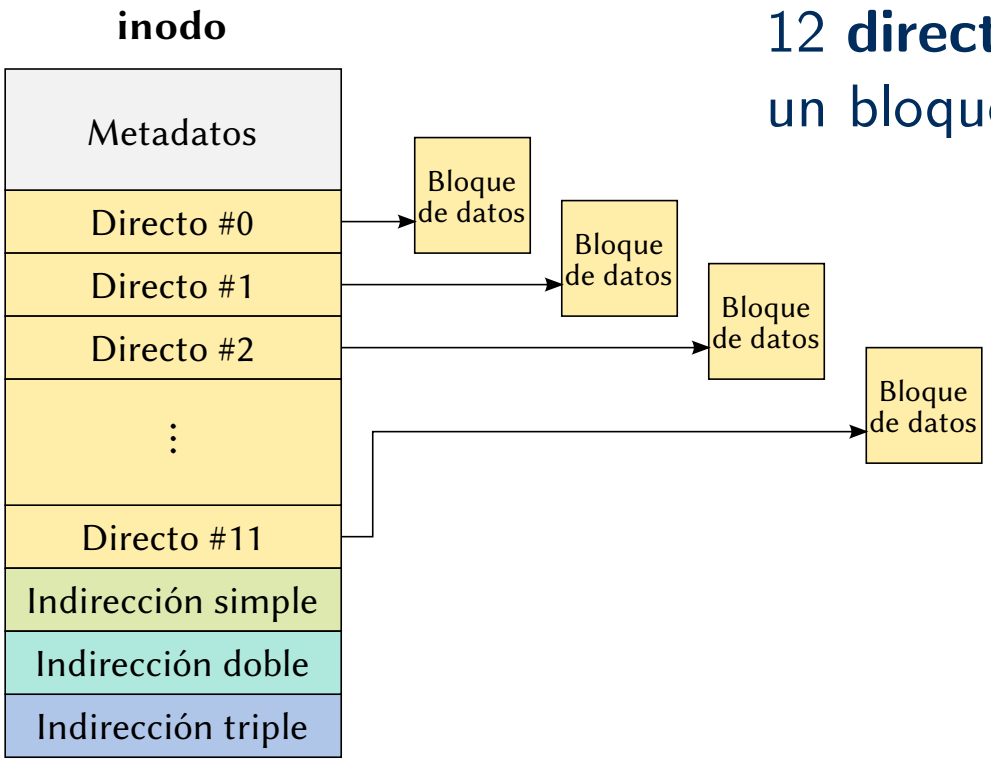
Inodos: Punteros con indirección

- 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

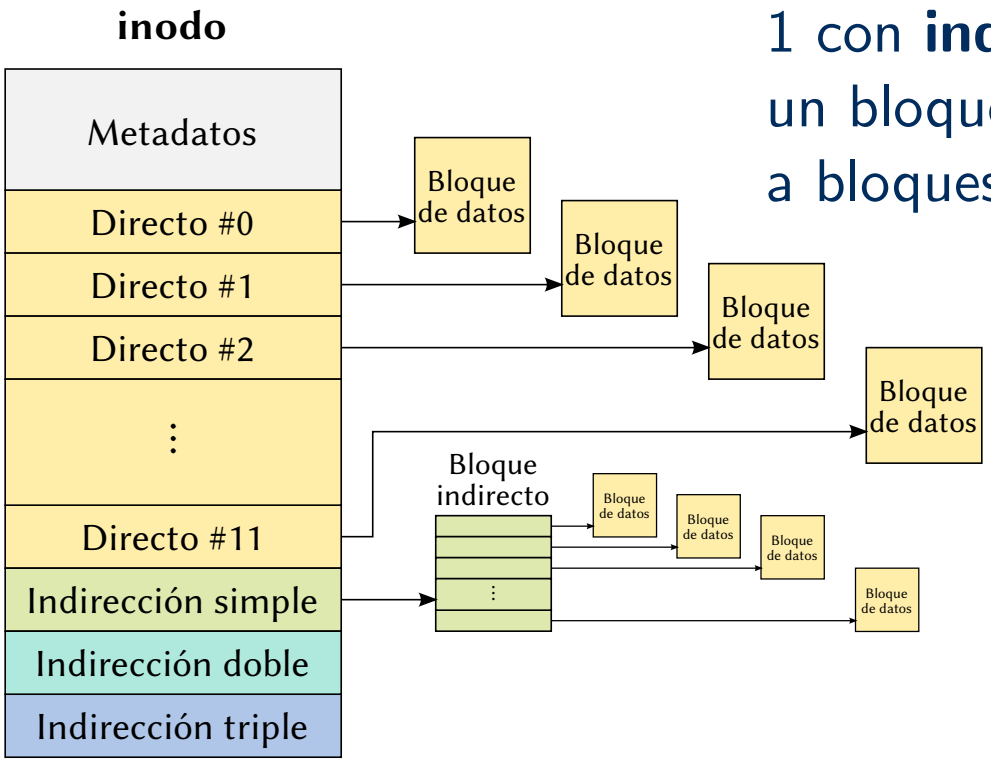
- Es deseable que los inodos tengan un tamaño fijo.
- Además, los primeros bloques de un archivo suelen ser accedidos con más frecuencia.
- Por eso, se utilizan **punteros con indirección**.
- Por ejemplo, en ext2, todos los inodos contienen 15 punteros a bloques, de cuatro tipos distintos.

Inodos: Punteros con indirección



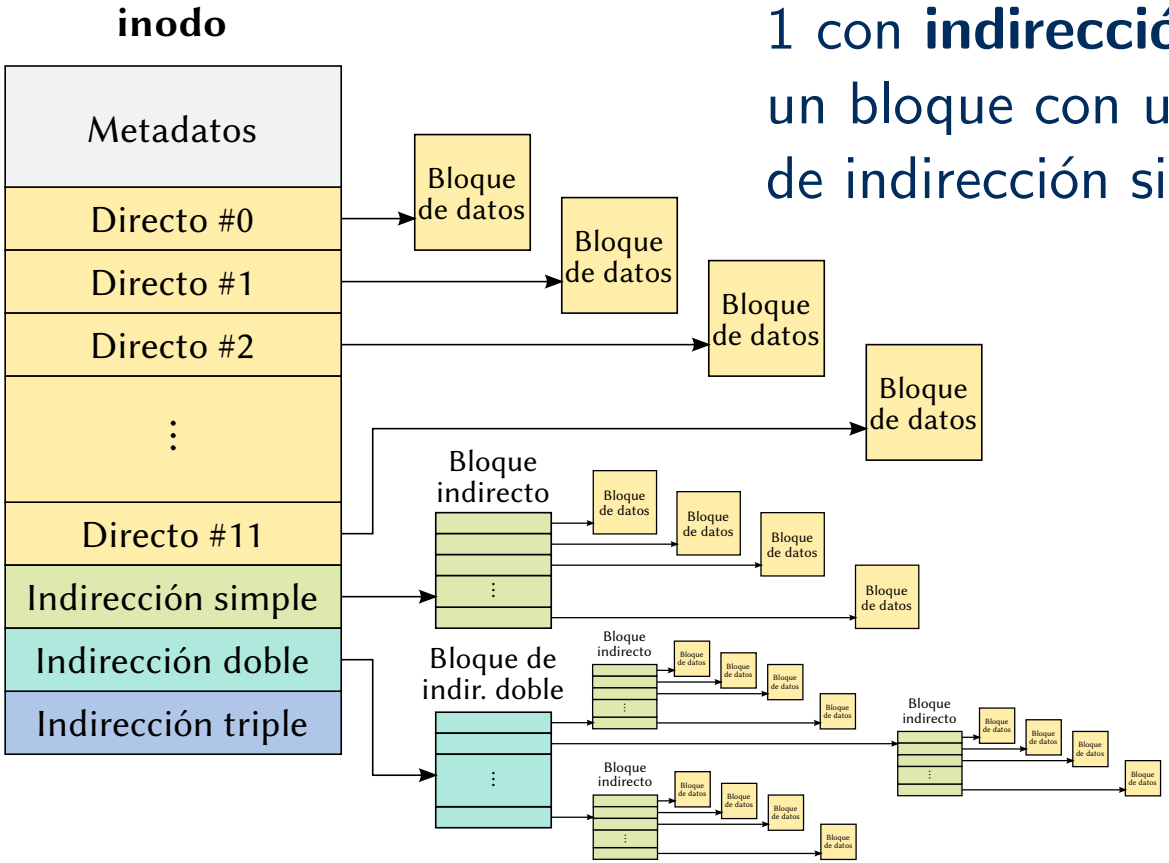
12 **directos**: apuntan directamente a un bloque de datos.

Inodos: Punteros con indirección



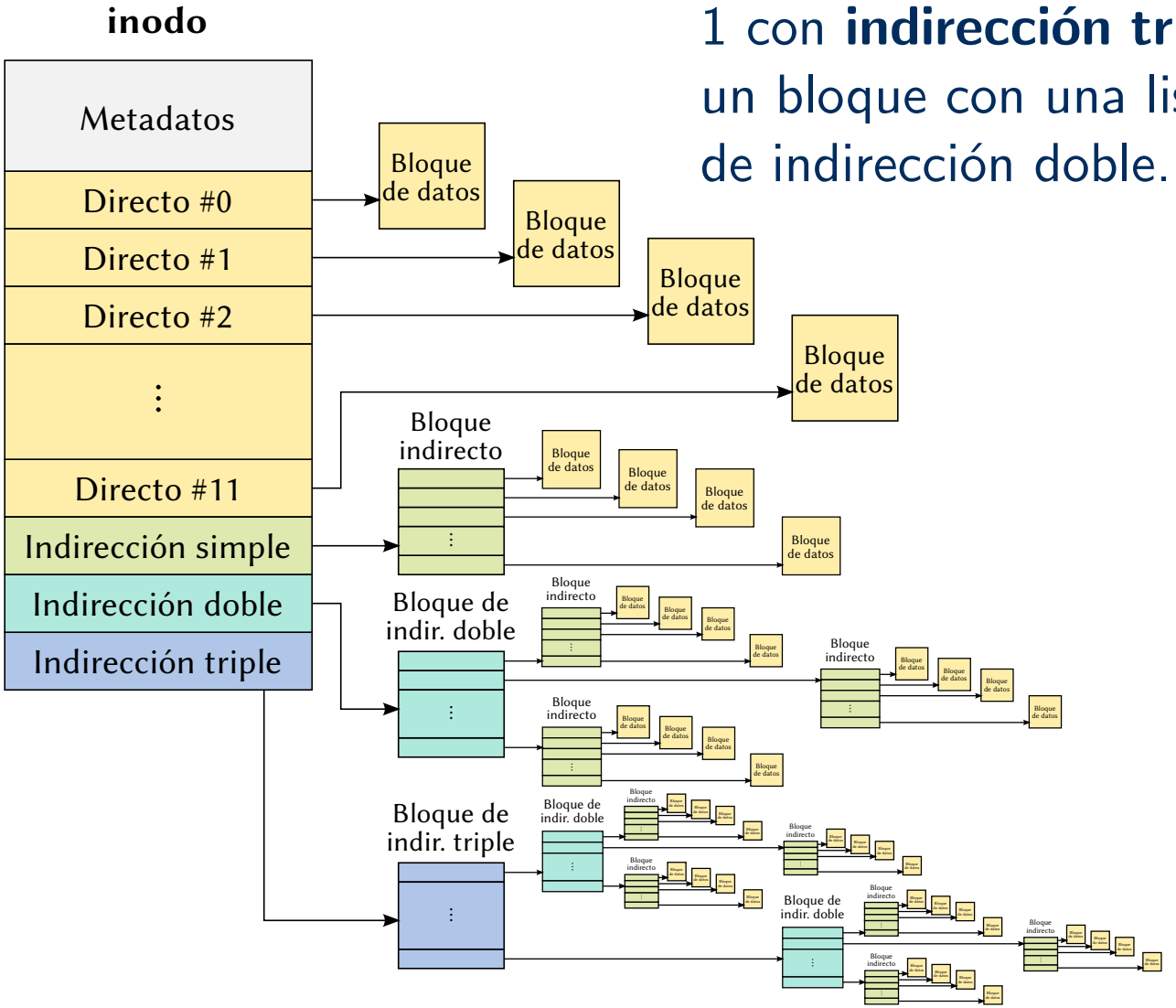
1 con **indirección simple**: apunta a un bloque con una lista de punteros a bloques de datos.

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.

Ejercicio 1 - inodos

3. Se tiene un disco con bloques de 8 KB y un sistema de archivos con inodos, sin indirecciones. Considerar un archivo de 10 MB, cuyo inodo está cargado en memoria.

Ejercicio 1 - inodos

3. Se tiene un disco con bloques de 8 KB y un sistema de archivos con inodos, sin indirecciones. Considerar un archivo de 10 MB, cuyo inodo está cargado en memoria.
   - ¿Cuántas operaciones de disco son necesarias para...
     - **(a)** agregar un bloque al principio del archivo?
     - **(b)** agregar un bloque en la mitad del archivo?
     - **(c)** agregar un bloque al final del archivo?
     - **(d)** eliminar el primer bloque del archivo?
     - **(e)** eliminar el bloque en la mitad del archivo?
     - **(f)** eliminar el último bloque del archivo?

     ¿Cuántas posiciones del inodo deben consultarse y cuántas deben modificarse en cada caso?
   - Comparar con los resultados obtenidos para la asignación contigua y con FAT.

Ejercicio 1 - inodos

3. Se tiene un disco con bloques de 8 KB y un sistema de archivos con inodos, sin indirecciones. Considerar un archivo de 10 MB, cuyo inodo está cargado en memoria.

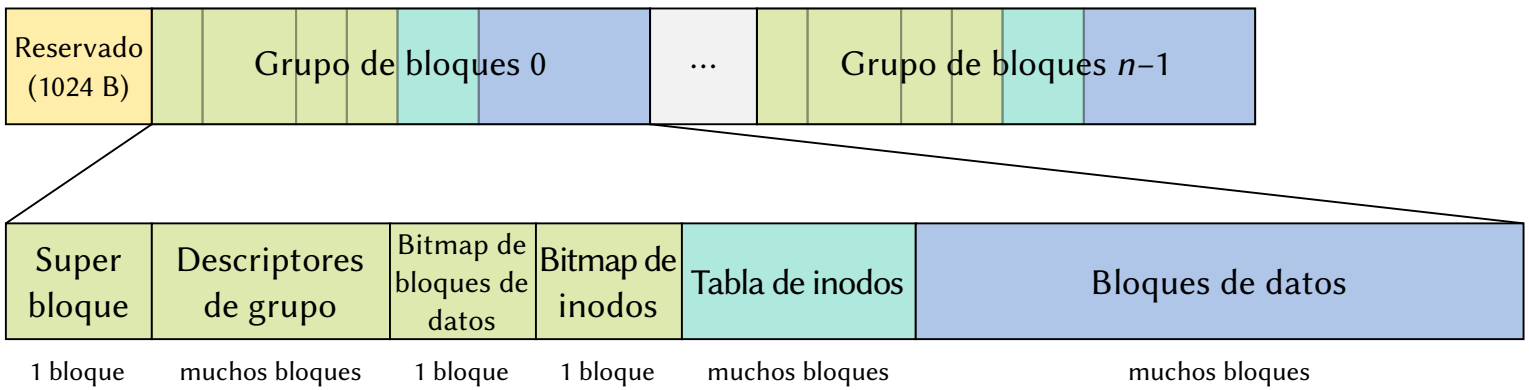
	Contiguo	FAT			inodos		
	Disco	Disco	Lect	Esc	Disco	Lect	Esc
add_0	2561	1	0	1	1	1280	1281
$add_{\frac{n}{2}}$	1281	1	640	2	1	640	641
add_n	1	1	1279	2	1	0	1
del_0	2558	0	1	0	0	1279	1279
$del_{\frac{n}{2}}$	1278	0	641	1	0	639	639
del_n	0	0	1278	1	0	0	0

¿Y dónde están los inodos?

- En un sistema de archivos ext2, los bloques del disco están particionados en **grupos de bloques** contiguos.

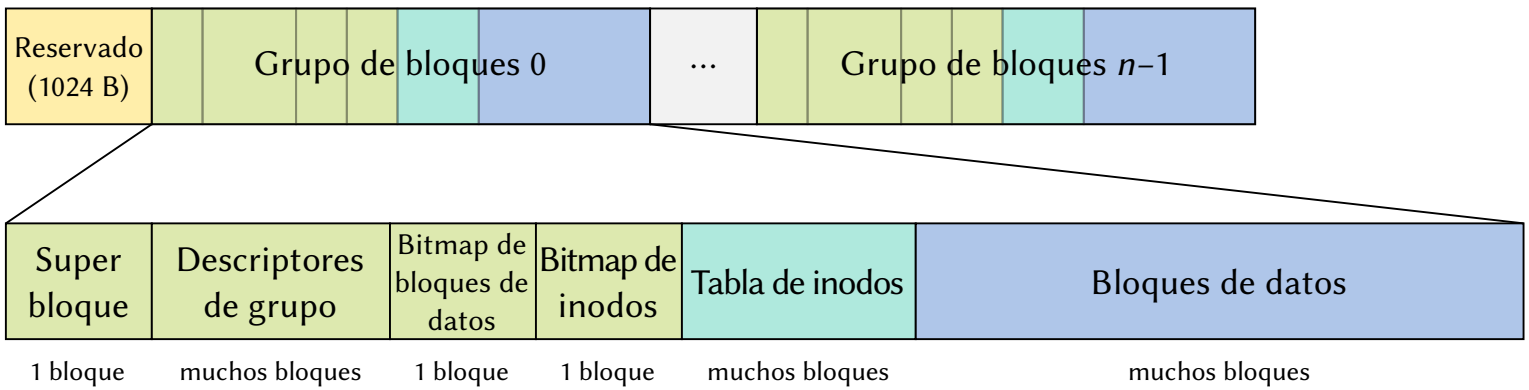
¿Y dónde están los inodos?

- En un sistema de archivos ext2, los bloques del disco están particionados en **grupos de bloques** contiguos.
- Cada grupo contiene bloques de **datos** y bloques de **inodos**.



¿Y dónde están los inodos?

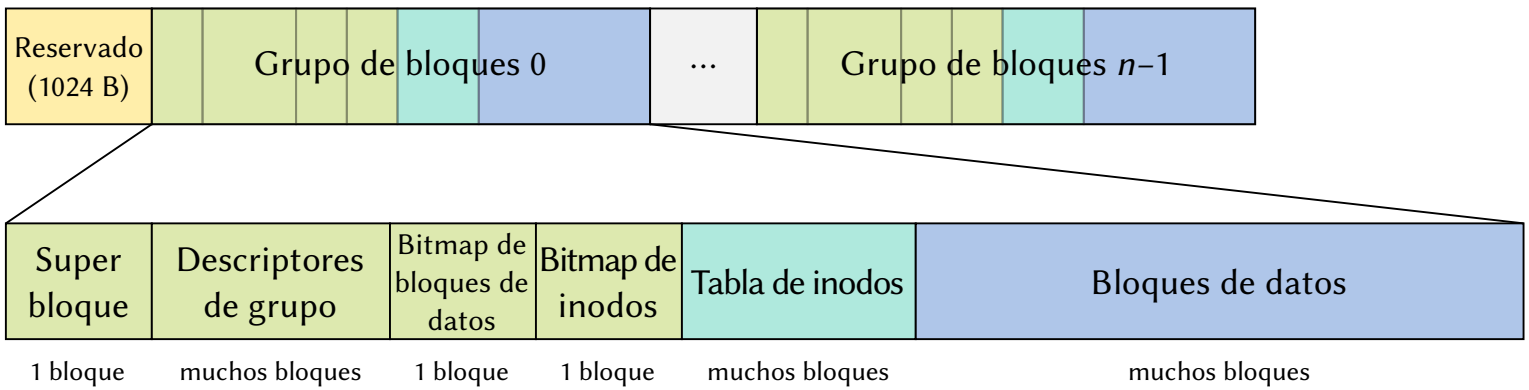
- En un sistema de archivos ext2, los bloques del disco están particionados en **grupos de bloques** contiguos.
- Cada grupo contiene bloques de **datos** y bloques de **inodos**.



- Esto quiere decir que los inodos están *repartidos* a lo largo de todo el disco.

¿Y dónde están los inodos?

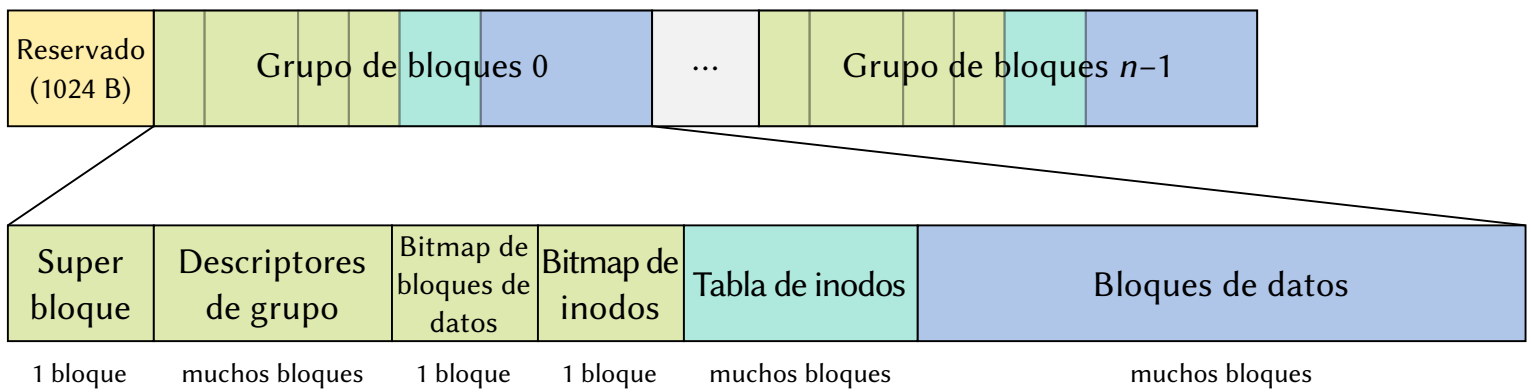
- En un sistema de archivos ext2, los bloques del disco están particionados en **grupos de bloques** contiguos.
- Cada grupo contiene bloques de **datos** y bloques de **inodos**.



- Esto quiere decir que los inodos están *repartidos* a lo largo de todo el disco.
- En un único bloque puede haber varios inodos.

¿Y dónde están los inodos?

- En un sistema de archivos ext2, los bloques del disco están particionados en **grupos de bloques** contiguos.
- Cada grupo contiene bloques de **datos** y bloques de **inodos**.



- Esto quiere decir que los inodos están *repartidos* a lo largo de todo el disco.
- En un único bloque puede haber varios inodos.
- Más detalles, en el **taller de ext2**.

Directorios en ext2

En ext2, los directorios:

- Los directorios se almacenan en disco como archivos comunes, aunque su contenido se interpreta diferente. Se los distingue por el tipo de archivo.

Directorios en ext2

En ext2, los directorios:

- Los directorios se almacenan en disco como archivos comunes, aunque su contenido se interpreta diferente. Se los distingue por el tipo de archivo.
- **A cada directorio le corresponde un inodo**, y cada bloque de un directorio es una lista de entradas de directorio (*dentry*). Cada entrada a su vez 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, los directorios:

- Los directorios se almacenan en disco como archivos comunes, aunque su contenido se interpreta diferente. Se los distingue por el tipo de archivo.
- **A cada directorio le corresponde un inodo**, y cada bloque de un directorio es una lista de entradas de directorio (*dentry*). Cada entrada a su vez 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.
- Las primeras dos entradas en todos los directorios son ‘‘.’’ y ‘‘..’’.

Directorios en ext2

En ext2, los directorios:

- Los directorios se almacenan en disco como archivos comunes, aunque su contenido se interpreta diferente. Se los distingue por el tipo de archivo.
- **A cada directorio le corresponde un inodo**, y cada bloque de un directorio es una lista de entradas de directorio (*dentry*). Cada entrada a su vez 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.
- Las primeras dos entradas en todos los directorios son ‘‘.’’ y ‘‘..’’.
- Al igual que en FAT, el inodo del directorio **root** es distinguido: es siempre el inodo número 2.

Resumen de la clase:

- Analizamos distintos enfoques de sistemas de archivos.
  - Asignación contigua de bloques.
  - FAT
  - inodos
- Vimos cómo se manejan los directorios en FAT32 y Ext2.

Próxima clase:

- Ejercicio tipo parcial
- Presentación Taller ext2

Con esto se puede comenzar la guía práctica 6.


SO ASK ME
QUESTIONS
quickmeme.com