### Sistemas de Archivos

Sistemas Operativos DC - UBA - FCEN

22 de mayo de 2025

# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- 6 Cierre

# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- 6 Cierre

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.

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.

• Partes de un sistema de archivos:

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.

- Partes de un sistema de archivos:
  - Archivos que almacenan datos.

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.

- Partes de un sistema de archivos:
  - Archivos que almacenan datos.
  - Estructura de directorios para organizar los archivos.

Almacenamiento secundario

#### Almacenamiento secundario

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

• Cada bloque comprende uno o más sectores.

#### Almacenamiento secundario

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

#### Almacenamiento secundario

- 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).

#### Almacenamiento secundario

- 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

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- 6 Cierre

## ¿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.

(Sistemas Operativos) FS 1C 2025 7/48

# ¿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).

Atributos de un archivo:

(Sistemas Operativos) FS 1C 2025 8/48

#### Atributos de un archivo:

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

Atributos de un archivo:

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

Operaciones sobre archivos:

#### Atributos de un archivo:

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

#### Operaciones sobre archivos:

- Crear
- Abrir
- Escribir
- Leer
- Borrar
- ...

# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- Cierre

• Almacenar cada archivo en un conjunto de bloques contiguos.

10 / 48

(Sistemas Operativos) FS 1C 2025

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

(Sistemas Operativos) FS 1C 2025 10 / 48

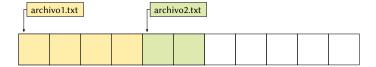
- Almacenar cada archivo en un conjunto de bloques contiguos.
- ¿Qué datos necesito para acceder a un archivo?
  - La dirección del bloque inicial y el tamaño (en bloques) del archivo.

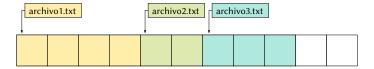
- Almacenar cada archivo en un conjunto de bloques contiguos.
- ¿Qué datos necesito para acceder a un archivo?
  - La dirección del bloque inicial y el tamaño (en bloques) del archivo.
- Es fácil de implementar pero tiene limitaciones.

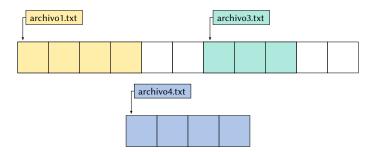
- Almacenar cada archivo en un conjunto de bloques contiguos.
- ¿Qué datos necesito para acceder a un archivo?
  - La dirección del bloque inicial y el tamaño (en bloques) del archivo.
- Es fácil de implementar pero tiene limitaciones.
- Principal problema: encontrar espacio para un nuevo archivo. Como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



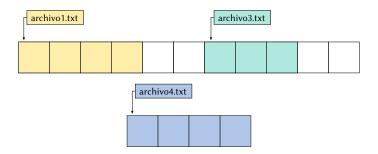








Ejemplo



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

• Compactar el disco es una tarea muy costosa.

(Sistemas Operativos) FS 1C 2025 12 / 48

- 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.

- 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

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- Cierre

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

14 / 48

- 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.

- 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.

- 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?

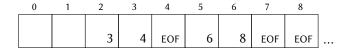
14 / 48

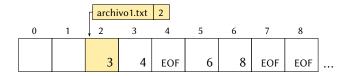
(Sistemas Operativos) FS 1C 2025

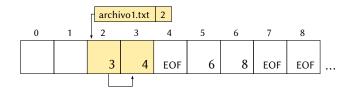
- Permite almacenar los archivos de forma no secuencial, guardando para cada bloque de un archivo, una referencia al siguiente (al estilo lista enlazada).
- La tabla tiene una entrada por cada bloque y se indexa por número de bloque.
- Cada entrada de la tabla contiene el número de bloque del siguiente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

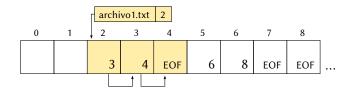
14 / 48

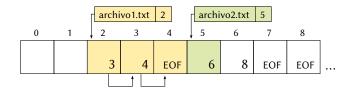
(Sistemas Operativos) FS 1C 2025

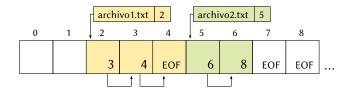


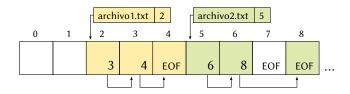


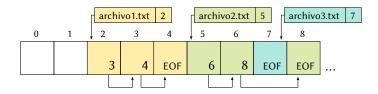






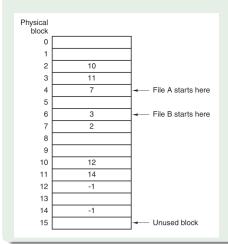






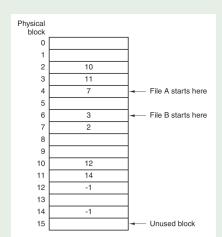
### Ejercicio

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



#### 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

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

- 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 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.

- 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.

- 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.

- 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.

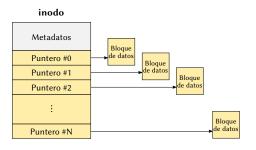
- La principal desventaja de FAT es que la tabla completa debe estar en memoria.
- ¿Cuál es el tamaño de la tabla si consideramos un disco de 1 TB y bloques de 1 KB?
  - La tabla tendrá 1 entrada por cada bloque de disco. Son  $\frac{1TB}{1KB} = \frac{2^{40} bytes}{2^{10} bytes} = 2^{30} bytes$  entradas.
  - 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  $\left\lceil \frac{30bits}{8} \right\rceil = 4bytes$  para las direcciones.

- 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 \[ \frac{30bits}{8} \] = 4bytes para las direcciones.
  - Entonces la tabla tendrá un tamaño de  $2^{30} \times 2^2$  bytes = 4 GB.

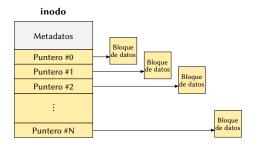
# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- 6 Cierre

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



- 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.



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

(Sistemas Operativos)

- 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?

- 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.

- 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.

#### Punteros con indirección

• Es deseable que los inodos tengan un tamaño fijo.

#### 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.

21 / 48

#### 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.

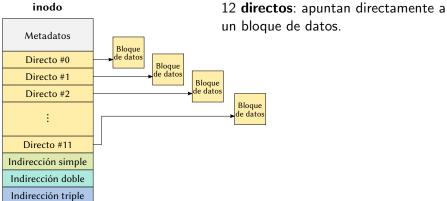
#### Punteros con indirección

#### inodo

Metadatos Directo #0 Directo #1 Directo #2 Directo #11 Indirección simple Indirección doble Indirección triple

- 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.

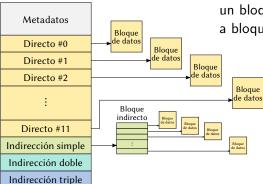
#### Punteros con indirección



un bloque de datos.

#### Punteros con indirección

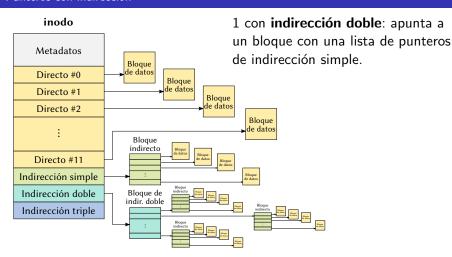
#### inodo



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

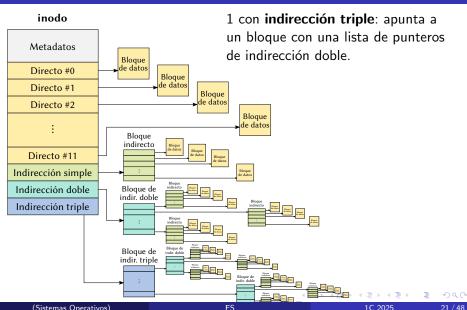
#### Inodos

#### Punteros con indirección



#### Inodos

#### Punteros con indirección



# Tabla de asignación de archivos (FAT)

## Ejercicio

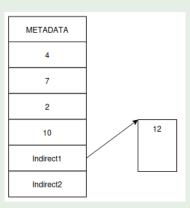
Consideremos un inodo con 4 entradas directas y 2 indirectas simples. ¿Qué pinta tiene el inodo de un archivo cuyos bloques de datos son 4 - 7 - 2 - 10 - 12 en ese orden?

METADATA Direct1 Direct2 Direct3 Direct4 Indirect1 Indirect2

# 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

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- Cierre

• Antes de poder leer un archivo (syscall read()), hay que abrirlo.

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

24 / 48

(Sistemas Operativos) FS 1C 2025

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

24 / 48

(Sistemas Operativos) FS 1C 2025

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

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

24 / 48

(Sistemas Operativos) FS 1C 2025

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

24 / 48

(Sistemas Operativos) FS 1C 2025

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

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

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

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

24 / 48

(Sistemas Operativos) FS 1C 2025

Los directorios también son archivos. Sus bloques de datos representan una tabla con una entrada por cada archivo que contienen, indicando su nombre y el número de inodo/primer bloque.

Los directorios también son archivos. Sus bloques de datos representan una tabla con una entrada por cada archivo que contienen, 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.

# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- Cierre

• Consiste en una lista de entradas de tamaño fijo.

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

- 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.

- 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.

- 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.

Sector de arranque	FAT	FAT (duplicado)	Directorio raíz	Datos (Otros directorios y todos los archivos)
--------------------	-----	--------------------	--------------------	---

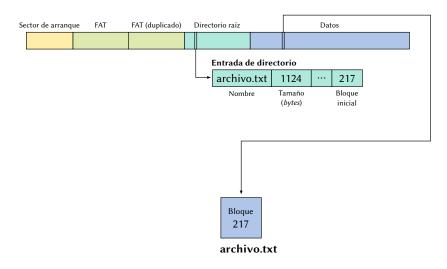
Estructura de un sistema de archivos FAT32.

Sector de arranqu	ie FAT	FAT (duplicado)	Directorio raíz	Datos

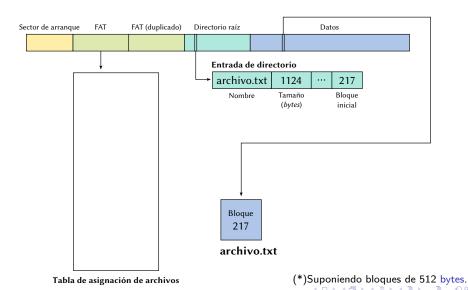
Obteniendo un archivo(\*)



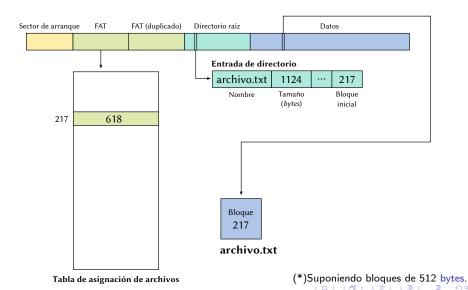
Obteniendo un archivo(\*)

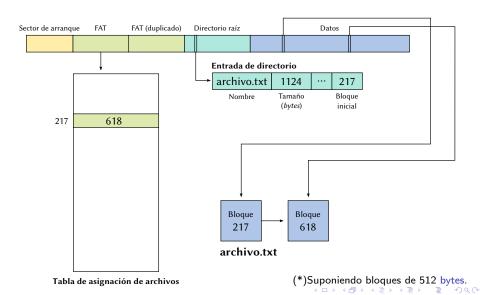


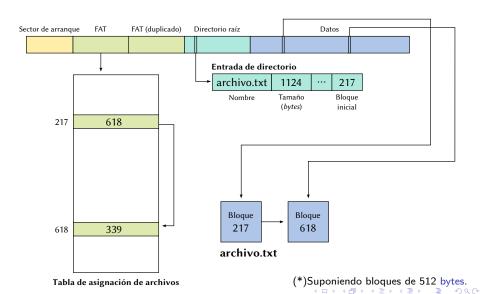
(\*)Suponiendo bloques de 512 bytes.

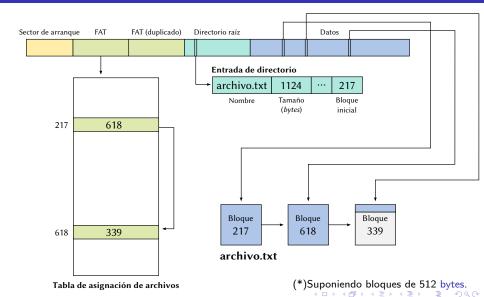


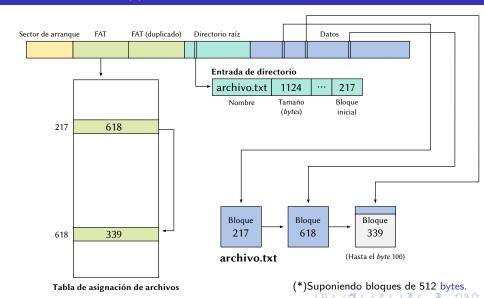
Obteniendo un archivo(\*)

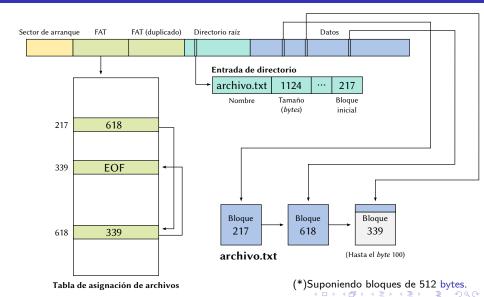












#### Enunciado

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].

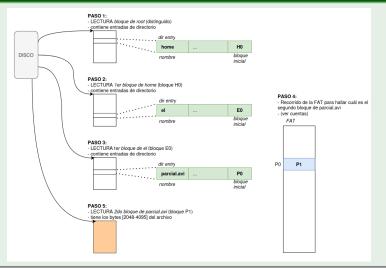
- Tengo la entrada de root (distinguido), busco en la FAT el 1er bloque de la tabla de directorios para hallar la entrada de home. Cuando la encuentro, busco en la FAT el 1er bloque de la tabla de directorios para hallar la entrada de el. Lo mismo para parcial.avi. 3 LECTURAS DE DISCO.
- Ahora miro en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

30 / 48

(Sistemas Operativos) FS 1C 2025

#### Ejercicio

## Dibujito



# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- Cierre

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

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

- 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.

- 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.

- 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.
- Las primeras dos entradas en todos los directorios son ''.'' y

- 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.
- 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.

#### 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:

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

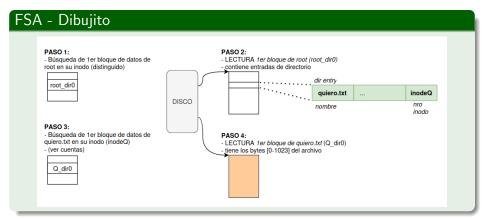
Puede dejar expresado el resultado como potencias de dos. Justificar.

### 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.

35 / 48



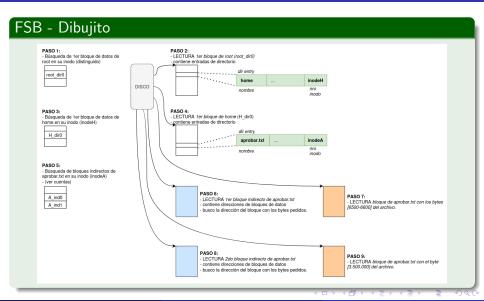
36 / 48

## 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<sup>10</sup> direcciones por bloque. Esto significa que tengo 2<sup>10</sup> · 2 · 2<sup>10</sup> = 2<sup>21</sup> 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<sup>21</sup> bytes y ahí está el byte [3.500.000]. 2 LECTURAS DE DISCO.
- TOTAL: 6 LECTURAS DE DISCO.

#### **Ejercicio**



# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- 6 Cierre

# Menú para hoy

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- 6 Cierre

#### Hard links

 En los sistemas de archivos con inodos, 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.

41 / 48

(Sistemas Operativos) FS 1C 2025

#### Hard links

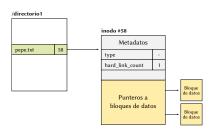
- En los sistemas de archivos con inodos, el nombre de los archivos no aparece en los inodos. Así, podemos referenciar el mismo inodo con diferentes nombres, y desde más de un directorio.
- Esto se conoce como enlace duro o físico (hard link).

41 / 48

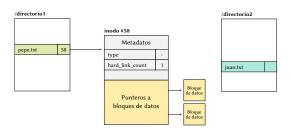
(Sistemas Operativos) FS 1C 2025

- En los sistemas de archivos con inodos, 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.
- Esto se conoce como enlace duro o físico (hard link).
- 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.

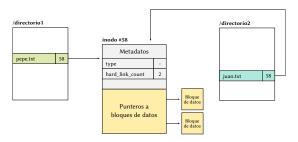
- En los sistemas de archivos con inodos, el nombre de los archivos no aparece en los inodos. Así, podemos referenciar el mismo inodo con diferentes nombres, y desde más de un directorio.
- Esto se conoce como enlace duro o físico (hard link).
- El nombre de archivo "." en un directorio es un enlace duro al mismo directorio. El nombre de archivo ".." es un enlace duro al directorio padre.



- En los sistemas de archivos con inodos, 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.
- Esto se conoce como enlace duro o físico (hard link).
- 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.



- En los sistemas de archivos con inodos, 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.
- Esto se conoce como enlace duro o físico (hard link).
- 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.



Hard links

• ¿Cómo hacemos para borrar un archivo que puede tener 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 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?

- ¿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

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- Cierre

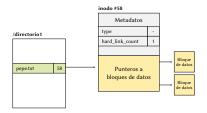
#### Symbolic links

 También podemos crear enlaces simbólicos (symbolic links). Estos se implementan mediante un inodo adicional, donde se almacena el destino del enlace.

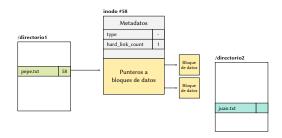
44 / 48

- También podemos crear enlaces simbólicos (symbolic links). Estos se implementan mediante un inodo adicional, donde se almacena el destino del enlace.
- Permiten referenciar directorios en otros sistemas de archivos.

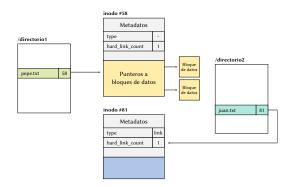
- También podemos crear enlaces simbólicos (symbolic links). Estos se implementan mediante un inodo adicional, donde se almacena el destino del enlace.
- Permiten referenciar directorios en otros sistemas de archivos.



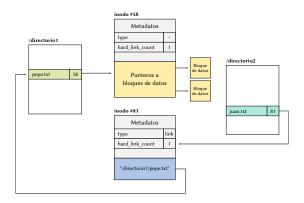
- También podemos crear enlaces simbólicos (symbolic links). Estos se implementan mediante un inodo adicional, donde se almacena el destino del enlace.
- Permiten referenciar directorios en otros sistemas de archivos.



- También podemos crear enlaces simbólicos (symbolic links). Estos se implementan mediante un inodo adicional, donde se almacena el destino del enlace.
- Permiten referenciar directorios en otros sistemas de archivos.



- También podemos crear enlaces simbólicos (symbolic links). Estos se implementan mediante un inodo adicional, donde se almacena el destino del enlace.
- Permiten referenciar directorios en otros sistemas de archivos.



Symbolic links

• ¿Qué pasa si borramos un enlace simbólico?

#### 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.

45 / 48

- ¿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?

- ¿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

- Introducción
- 2 Archivos
  - Asignación contigua
  - Tabla de asignación de archivos (FAT)
  - Inodos
- 3 Directorios
  - Directorios en FAT
  - Directorios en Ext2
- 4 Enlaces
  - Hard links
  - Symbolic links
- 6 Cierre

# Hoy vimos...

• Distintos enfoques de sistemas de archivos.

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

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

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

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

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

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

# 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