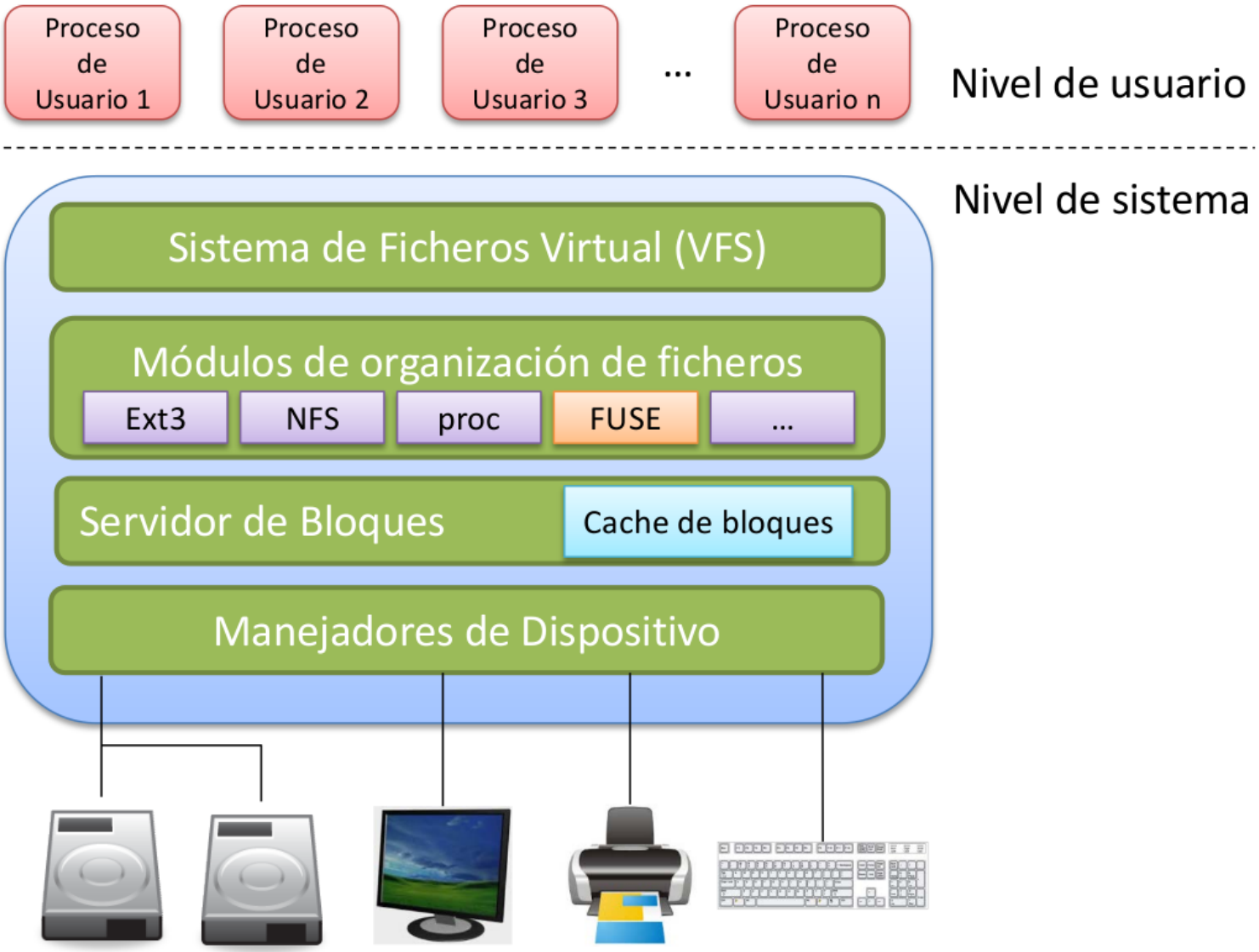
Sistemas Operativos

Gestión de Ficheros

Sistemas de Ficheros

Agenda

Servidor de ficheros



Sistema de Ficheros Virtual

- Capa de abstracción que se encarga de comprobaciones de errores comunes a todos los SSFF
  - Permite tratar de forma homogenea a todos los SSFF soportados
  - Ejemplo: permisos de acceso
- Maneja descriptores virtuales (nodos-i virtuales)
  - Contienen una referencia a la información que necesita el módulo de organización de ficheros correspondiente.
  - Permite completar información no disponible en algunos SSFF
  - Ejemplo: referencia al nodo-i de un SF ext3 montado
- Redirige la llamada al módulo de organización de ficheros correspondiente

Módulos de organización de ficheros

- Proporcionan el modelo de fichero del sistema operativo e implementan las operaciones básicas
  - Un módulo por cada SF soportado (UNIX, AFS, Windows NT, MS-DOS, EFS, MINIX, etc.).
  - Relacionan el modelo lógico con su almacenamiento real, transladando offsets lógicos a números de bloques físicos
  - Gestionan el espacio, la asignación de bloques, el manejo de los descriptores internos, etc.
- También hay módulos para pseudoficheros (`dev`, `proc`).
  - Son SF virtuales que el SO utiliza para ofrecer información interna y/o dar algún servicio

Servidor de bloques

- Aisla de los detalles de los drivers de disco a los SSFF
- API simple: leer/escribir bloque de un disco
- Implementa una Cache de bloques:
  - Los bloques más recientemente accedidos se dejan copiados en memoria, mejorando el rendimiento del sistema
  - En cada acceso se comprueba si el bloque está en la cache.
  - Si no está se copia del disco a la cache.
  - Si la cache está llena, hay que quitar un bloque para hacer hueco: políticas de reemplazo.
  - Si el bloque ha sido escrito (sucio): política de escritura.

Cache: Políticas de reemplazo

- FIFO (First In First Out)
- MRU (Most Recently Used)
- LRU (Least Recently Used)
  - Política más común, se reemplaza el bloque que lleva más tiempo sin ser referenciado. Los bloques más usados se encuentran en RAM.

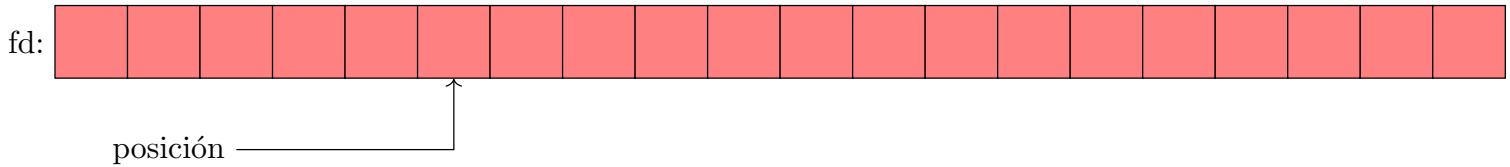
Cache: Políticas de escritura

- Escritura inmediata (write-through): cada actualización en cache implica escritura en disco. Rendimiento malo.
- Escritura diferida (write-back): un bloque sólo se escribe a disco cuando se elige para su reemplazo en la cache.
  - Optimiza el rendimiento, pero genera problemas de fiabilidad
- Escritura retrasada (delayed-write), periódicamente se escriben a disco los bloques modificados (30s en UNIX).
  - Compromiso entre rendimiento y fiabilidad.
  - Los bloques especiales se escriben inmediatamente al disco.
  - Hay que volcar los datos de la cache antes de quitar un disco
- Escritura al cierre (write-on-close): cuando se cierra un archivo se escriben en disco los bloques modificados

Agenda

Programador vs SO

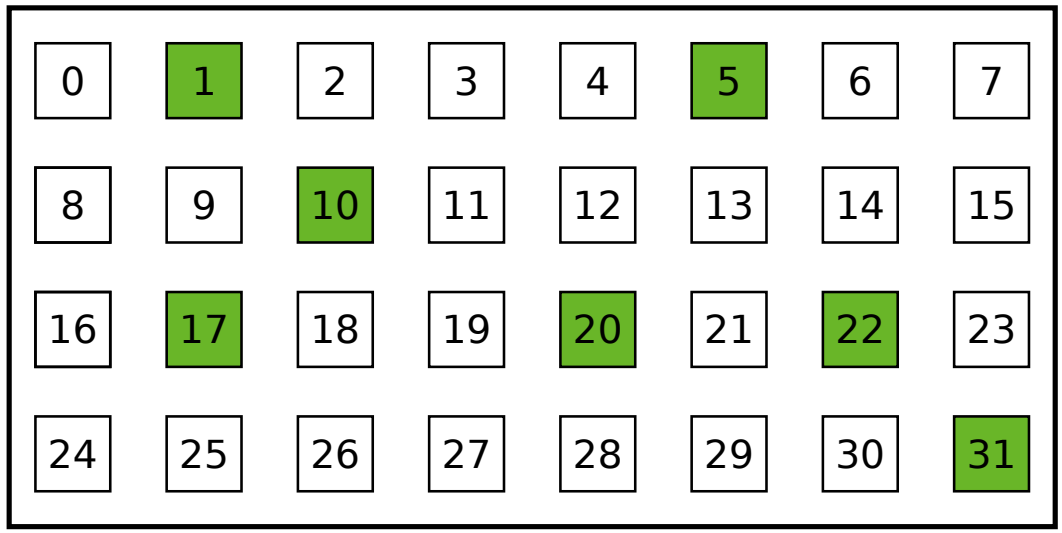
- Programador: array de bytes con un puntero de posición



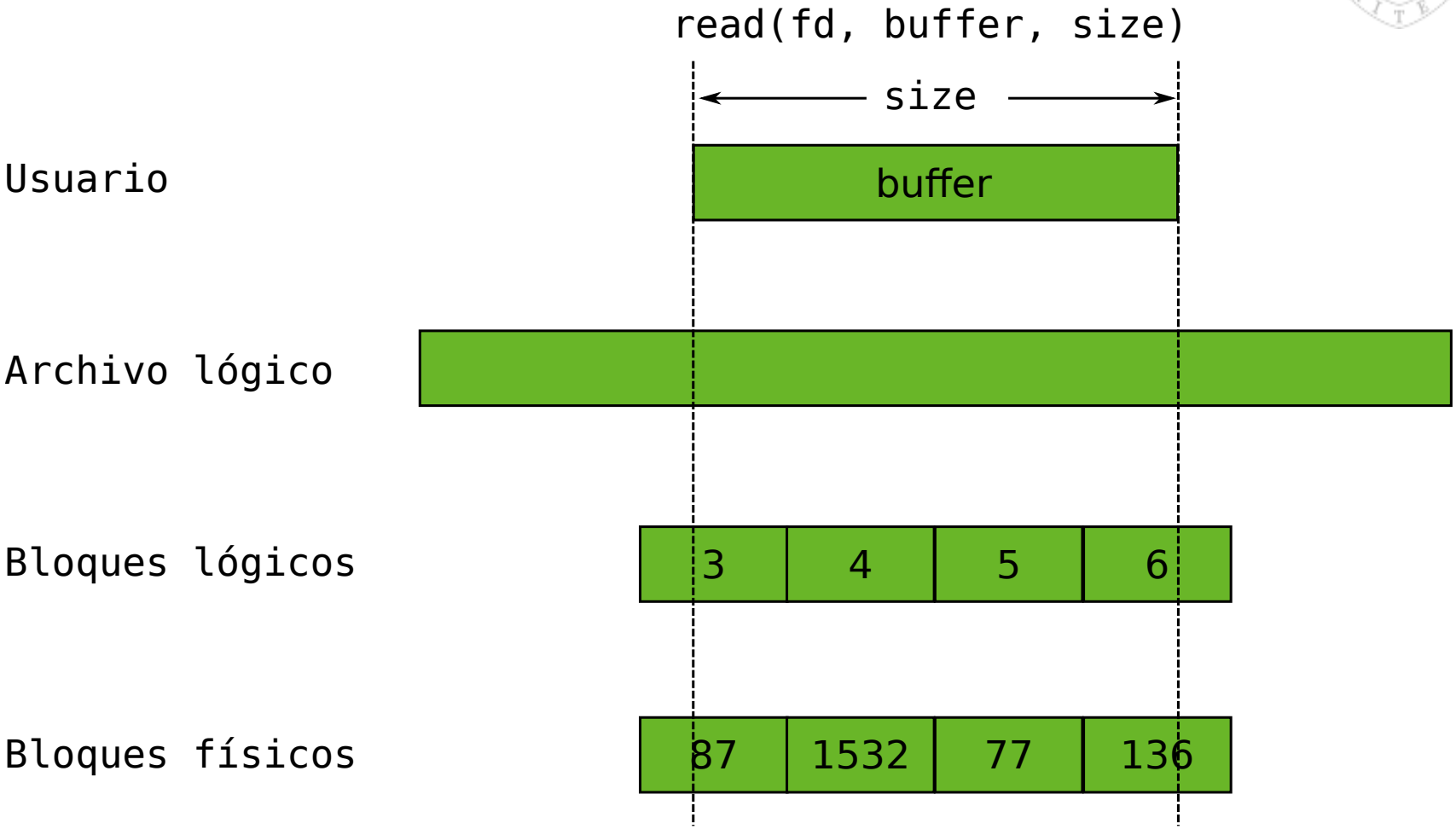
- SO: un conjunto de bloques del disco

Bloques del Archivo X

0	17
1	22
2	1
3	5
4	20
5	31
6	10



Operaciones en ficheros

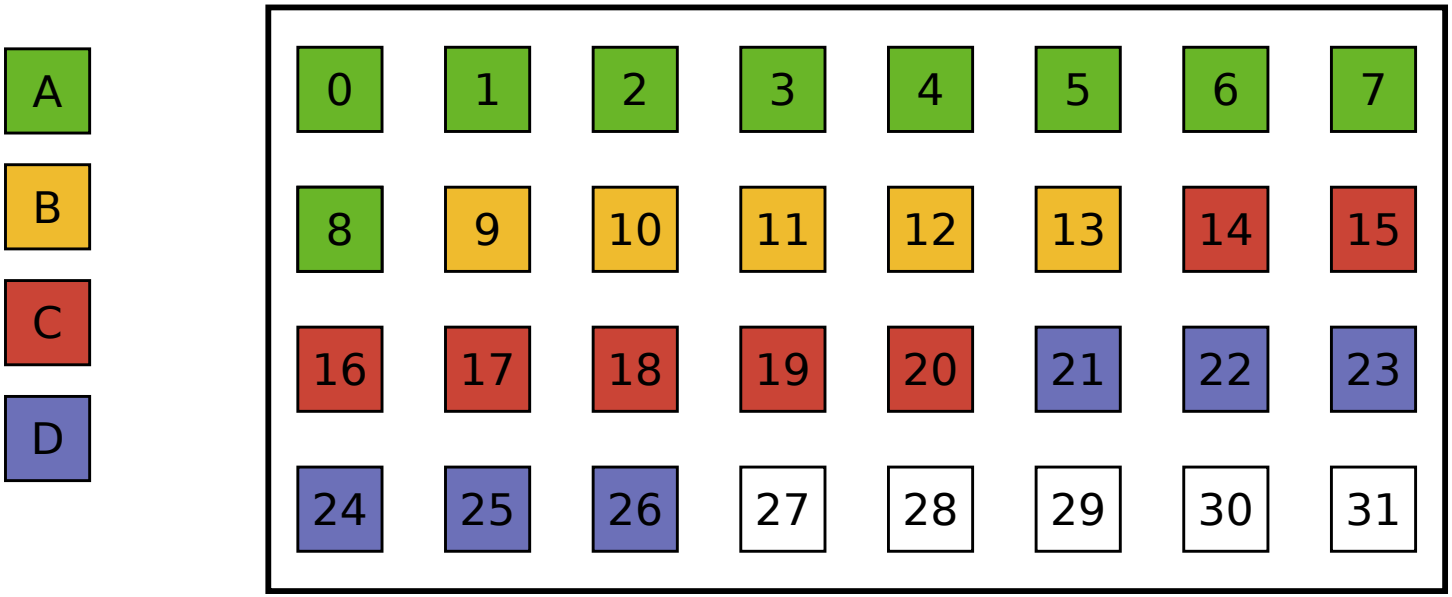


Asignación de bloques

- Bloques contiguos:
  - Usado en CD-ROM y cintas
- Bloques Enlazados:
  - Usado en sistemas FAT (File Allocation Table)
- Bloques Indexados:
  - unix-sv, fff (fast file system), ext2, ext3
- Extents (grupos de bloques contiguos):height
  - NTFS, JFS, Reiser4, HFS, XFS, ...

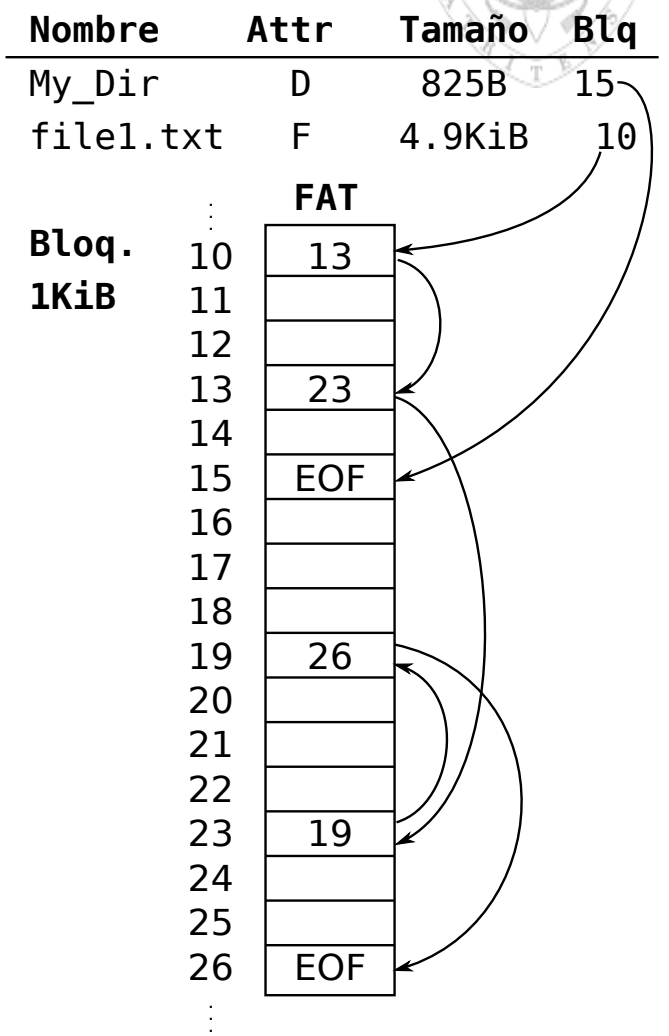
Bloques Contiguos

- Ventajas:
  - Acceso secuencial óptimo, permite lecturas anticipadas, fácil acceso aleatorio
- Desventajas:
  - Fragmentación externa, pre-declaración de tamaño, necesidad de compactación
- Tamaño máximo de fichero:
  - Num. Bloques dispositivo x Tam. bloque



Bloques enlazados (FAT)

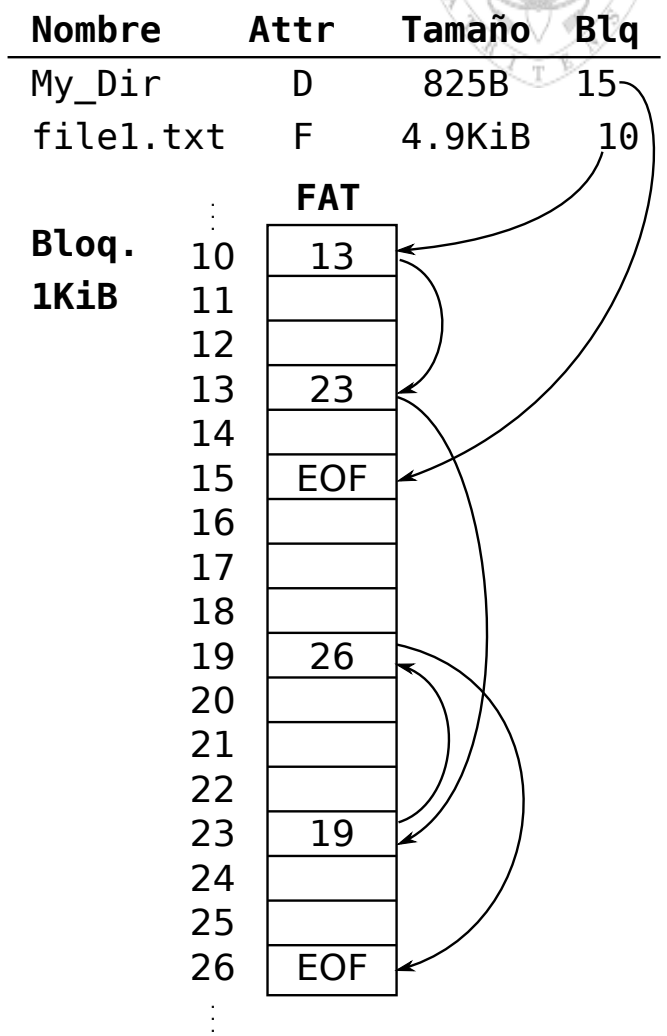
- Lista de bloques enlazados
  - En el directorio tenemos guardado el número de bloque físico que se utiliza para almancenar el primer bloque lógico del fichero
  - Se utiliza una tabla (FAT) para almacenar los enlaces a los siguientes bloques
  - Se usa una marca especial en la tabla FAT para indicar que es el último bloque del fichero (EOF)
- Distintas versiones de FAT indican el número de bits (12, 16, 32) usados para identificar un bloque



Acceso al n-ésimo byte

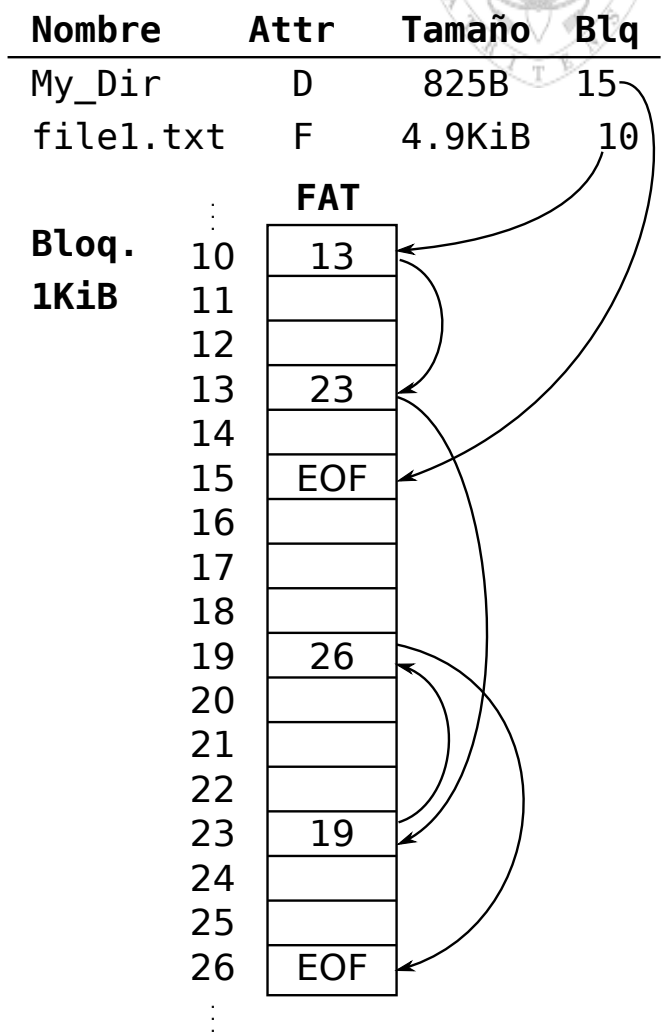
Para encontrar el n-ésimo byte del fichero tenemos que:

1. Calcular el bloque lógico en el que se encuentra el byte: $B = n/TB$
2. Acceder al directorio para obtener el id del primer bloque
3. Seguir los enlaces en la tabla FAT hasta llegar al bloque lógico B (B-1 enlaces)
4. Acceder al disco, en el bloque físico (indicado en la última entrada de la FAT consultada)



Ventajas y desventajas

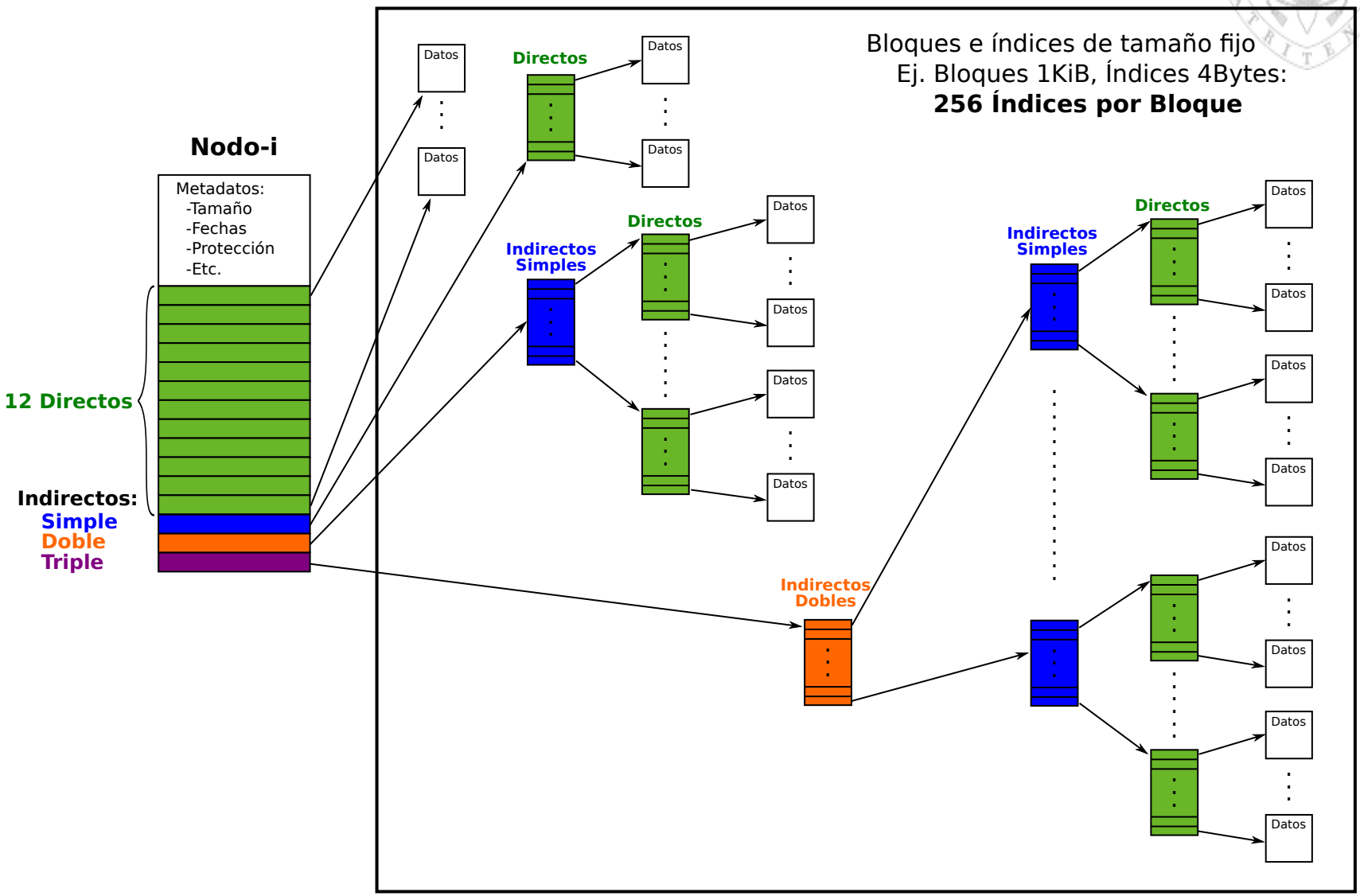
- Ventajas:
  - No produce fragmentación externa
  - Asignación dinámica simple: Cualquier clúster libre puede ser añadido a la cadena
  - Acceso secuencial fácil
- Desventajas:
  - No toma en cuenta el principio de localidad, falta de contigüidad
  - Acceso aleatorio ineficiente e irregular
    - Mejora si la FAT está en memoria (sólo FAT16)
  - Es conveniente realizar compactaciones periódicas



Bloques indexados

- Cada fichero tiene una estructura llamada índice
  - Entrada $n \rightarrow$ id del bloque físico que contiene el bloque lógico n
- Puede usar varios niveles de indirección
  - Ejemplo: Unix, ext2, ext3
  - Acceso eficiente a ficheros pequeños o grandes y dispersos
  - Operaciones de borrado y truncado ineficientes en ficheros grandes
- Accesible desde una estructura asociada al fichero
  - Nodo-i en Unix
  - Representa al fichero y almacena toda la meta información (atributos) del mismo
  - El Sistema de ficheros guarda una tabla con estos nodos
  - El directorio almacena el id del nodo asociado a cada fichero

Bloques indexados tipo unix



Acceso al n-ésimo byte

```
#define DP 12                                   // #Direct pointers
#define BLOCK_SIZE 1024                         // Block size
// n: byte position
block_id get_bid(int n, struct inode *p_inode) {
    int log_b = n/BLOCK_SIZE, m = BLOCK_SIZE/sizeof(block_id);
    int b_id, ind;
    block_id *p_block;
    if (log_b < DP) {                           // Direct
        b_id = p_inode->direct[log_b];
    } else if (log_b < (DP + m)) {              // Simple
        p_block = get_disk_block(p_inode->ind_simple);
        b_id = p_block[log_b - DP];
    } else if (log_b < (DP + m + m*m)) {        // Double
        log_b = log_b - (DP + m);
        p_block = get_disk_block(p_inode->ind_double);
        ind = log_b / m;
        p_block = get_disk_block(p_block[ind]);
        ind = log_b % m;
        b_id = p_block[ind];
    } ...
```

Acceso al n-ésimo byte

```
    ... else {                                  // Triple
        log_b = log_b - (DP + m + m*m);
        p_block = get_disk_block(p_inode->ind_triple);
        ind = log_b / (m*m);
        p_block = get_disk_block(p_block[ind]);
        ind = (log_b % (m*m)) / m;
        p_block = get_disk_block(p_block[ind]);
        ind = ( log_b % (m*m)) % m;
        b_id = p_block[ind];
    }

    return bid;
}
```

Tamaño máximo de fichero

- Tamaño máximo de fichero según organización:

$$T_B \times (E_d + T_B/B_{id} + (T_B/B_{id})^2 + (T_B/B_{id})^3)$$

  - T_B: tamaño de bloque
  - E_d: número de enlaces diréctos en el nodo-i
  - B_{id}: número de bytes usados para el identificador de bloque físico.
- Tamaño máximo de fichero según dirección:

$$T_B \times 2^{8B_{id}}$$

  - Es una aproximación que desprecia el espacio usado para almacenar el propio índice (usa todos los bloques para los datos del fichero)
- ¿Influye el tamaño del puntero de Lectura/Escritura?
- ¿Cómo influyen estos datos en el tamaño de la partición?

Ejemplo: ext2

Sistema de ficheros básico usado en Linux (antecesor de ext3 y ext4)

- Tamaño de bloque típico 1KiB, 4 bytes para id bloque.
- Tamaño máximo de fichero:

$$min(2^{10} \times (12 + 2^8 + 2^{16} + 2^{24}), 2^{10} \times 2^{32}) \simeq 2^{34} = 16GiB$$

- ¿Cuántos bytes se emplean para almacenar únicamente los índices de un fichero que ocupa 16GiB? (Sin contar el nodo-i)

$$2^{10} \times (1 + (1 + 2^8) + (1 + 2^8 + 2^{16})) \simeq 64MiB$$

- ¿Y para tamaño de bloque 2KiB?

Ejemplo: ext2

fichero con formato ext2

```
$ dd if=/dev/zero of=/tmp/disk.img bs=1024 count=100K
102400+0 records in
102400+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0,298336 s, 351 MB/s
$ mkfs -t ext2 -b 1024 /tmp/disk.img
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 102400 1k blocks and 25688 inodes
Filesystem UUID: 1da29113-7c62-4758-8aa1-e0d2b767e3f8
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729
Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Ejemplo: ext2

montaje y uso

```
$ mkdir /tmp/disk
$ sudo mount -t ext2 -o defaults,loop /tmp/disk.img /tmp/disk
$ cd /tmp/disk
$ sudo dd if=/dev/urandom of=file.bin bs=1024 count=1 seek=0
1+0 records in
1+0 records out
1024 bytes (1,0 kB, 1,0 KiB) copied, 0,0002593 s, 3,9 MB/s
$ ls -la file.bin
-rw-r--r-- 1 root root 1024 Feb 21 23:36 file.bin
$ stat file.bin
  File: 'file.bin'
  Size: 1024            Blocks: 2          IO Block: 1024   regular file
Device: 700h/1792d      Inode: 12          Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-02-21 23:36:29.000000000 +0100
Modify: 2018-02-21 23:36:29.000000000 +0100
Change: 2018-02-21 23:36:29.000000000 +0100
```

Ejemplo: ext2

- ¿Y si en vez de usar dd con seek=0 ponemos seek=11?

fichero disperso (*sparse*) en ext2

```
$ rm file.bin
$ sudo dd if=/dev/urandom of=file.bin bs=1024 count=1 seek=11
1+0 records in
1+0 records out
1024 bytes (1,0 kB, 1,0 KiB) copied, 0,000197212 s, 5,2 MB/s
$ ls -la file.bin
-rw-r--r--  1 root root 12288 Feb 21 23:41 file.bin
$ stat file.bin
  File: 'file.bin'
  Size: 12288           Blocks: 2          IO Block: 1024   regular file
Device: 700h/1792d      Inode: 13          Links: 1
...
```

- ¿Qué veremos si hacemos $ hexdump -v file.bin?
- ¿Y con seek=12? ¿Y seek=268?

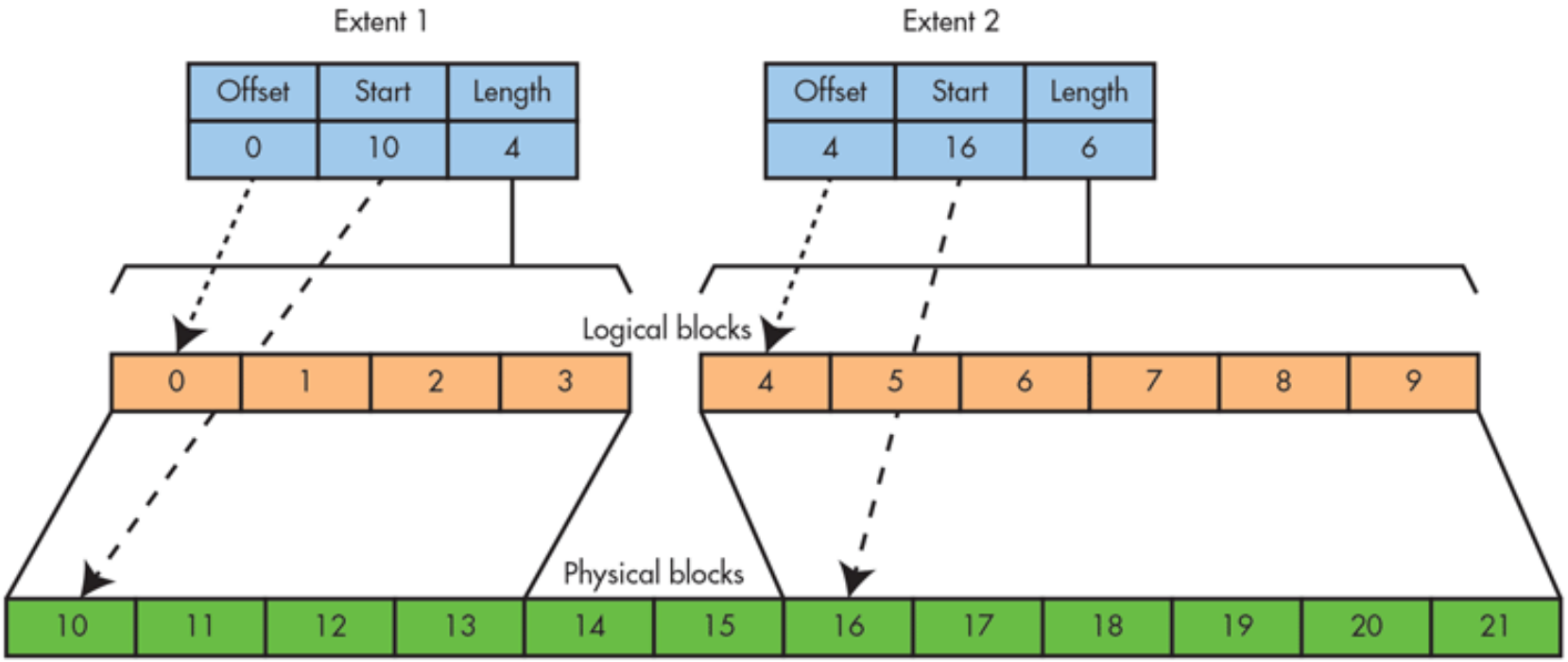
Ejemplo: ext2

fichero disperso (*sparse*) en FAT

```
$ dd if=/dev/zero of=/tmp/disk.img bs=1024 count=10K
10240+0 records in
10240+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0,0315767 s, 332 MB/s
$ mkfs -t vfat -s 2 disk.img
mkfs.fat 3.0.28 (2015-05-16)
$ sudo mount -t vfat -o defaults,loop /tmp/disk.img /tmp/disk
$ cd disk/
$ sudo dd if=/dev/urandom of=file.bin bs=1024 count=1 seek=12
1+0 records in
1+0 records out
1024 bytes (1,0 kB, 1,0 KiB) copied, 0,000205747 s, 5,0 MB/s
$ ls -la file.bin ; du -h file.bin ; stat file.bin
-rwxr-xr-x 1 root root 13312 Feb 22 16:32 file.bin
13K        file.bin
  File: 'file.bin'
  Size: 13312           Blocks: 26         IO Block: 1024   regular file
Device: 700h/1792d      Inode: 6           Links: 1
...
```

Extents

- Un extent es un grupo de bloques contiguos:
  - Bloque lógico del primero (offset)
  - Primer bloque físico (start)
  - Número de bloques del extent (length)



Organización del fichero

- Árbol balanceado de extents
  - Acceso homogéneo a todo el fichro
  - Ej: B Trees, B+ Trees, . . . (Video 12 min)
- El SF tiene un nodo por fichero (como nodo-i en unix)
  - Contiene metadatos y la raiz del árbol
- Permite tamaños de fichero muy grandes
  - En ext4 de haste 16 TiB con bloques de 4KiB
- Requiere algoritmos eficientes de asignación de bloques
  - La asignación a disco se retrasa lo más posible
  - Buscan grupos de bloques consecutivos
  - Permiten pre-asignación de espacio (Ej: máquinas virtuales)
- Los SSFF modernos usan extents (NTFS, ext4, HFS, . . .)
  - Ejemplo: ext4 (Video 54min, ver desde 15:40 hasta 40:35)

Gestión de Espacio Libre

- Mapa de bits
  - Tamaño mapa = Tamaño_de_disco / (8 * Tamaño_de_bloque)
  - Ej.: 16GiB / (8 * 1KiB) = 2MiB
- Bloques libres encadenados
- Indexación de bloques libres
  - El espacio libre como fichero con indexación sobre bloques de tamaño fijo
  - Variante: Indexación con zonas de tamaño variable
- Lista de bloques libres implementada como pila o cola con parte en memoria

Agenda

Estructura de los directorios

- Los directorios se suelen implementar como ficheros con formato conocido para el SO
- Hay estructuras de directorio muy distintas. La información contenida en el directorio depende de esa estructura. Dos alternativas principales:
  - Almacenar atributos de fichero en entrada directorio.
    - FAT: Nombre, tipo, atributos, fechas, tamaño, primer bloque
  - Almacenar únicamente [nombre, identificador]. El resto de los datos fichero en una estructura distinta (i-nodo de UNIX)

Interpretación de nombres en LINUX I

Bloques de datos del inodo 2

Nombre	i-nodo
.	2
..	2
tmp	43
home	342
info	27
⋮	⋮

Bloques de datos del inodo 342

Nombre	i-nodo
.	342
..	2
mary	430
miguel	256
elvira	78
⋮	⋮

Bloques de datos del inodo 256

Nombre	i-nodo
.	256
..	342
claves	758
texto	3265

Interpretación de nombres en LINUX II

- Interpretar `/home/miguel/claves`

- Traer a memoria i-nodo 2 (conocido) y su[s] datos[s]
- Buscar la cadena `home` para obtener el i-nodo 342
- Traer a memoria i-nodo 342 (debe ser un directorio) y su[s] dato[s]
- Buscar la cadena `miguel` para obtener el i-nodo 256
- Traer a memoria i-nodo 256 (debe ser un directorio) y su[s] dato[s]
- Buscar la cadena `claves` para obtener el i-nodo 758
- Se lee el nodo-i 758 y ya se tienen los datos del fichero

- ¿Cuándo parar?
  - Se ha encontrado el i-nodo del fichero
  - No se ha encontrado y no hay más subdirectorios
  - Estamos en un directorio y no contiene la siguiente componente del nombre (por ejemplo, miguel)

Agenda

Sistemas de Ficheros

- El sistema de ficheros permite organizar la información dentro de los dispositivos de almacenamiento secundario en un formato inteligible para el SO.
- Se crean sobre particiones o volúmenes:
  - Una partición es una porción de un disco a la que se la dota de una identidad propia y que puede ser manipulada por el SO como una entidad lógica independiente.
  - Un volumen es un conjunto particiones de discos que pueden ser tratado por el SO como una única unidad lógica
- Una vez creadas las particiones, el SO debe crear las estructuras de los SSFF dentro de esas particiones.
  - Se proporcionan comandos como `format` o `mkfs` al usuario.

Sector y Cluster

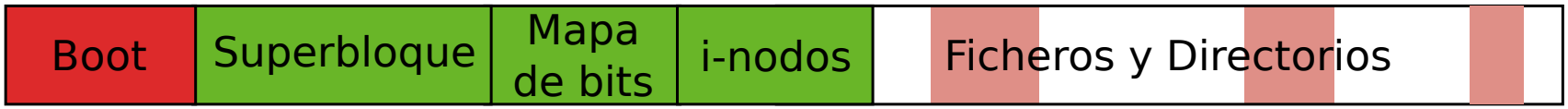
- Sector:
  - Unidad mínima de transferencia que puede manejar el controlador de disco, 2^mBytes (normalmente 512Bytes)
- Cluster (o bloque del SF):
  - Agrupación lógica de sectores de disco que supone la unidad de transferencia mínima que usa el sistema de ficheros. Por lo tanto, un fichero ocupará, como mínimo, un cluster.
  - Los SSFF tienen un tamaño de bloque por defecto, pero se puede especificar otro al usar `mkfs`
  - Sirve para optimizar la eficiencia de la entrada/salida de los dispositivos secundarios de almacenamiento.
  - El problema que introducen las agrupaciones grandes es la existencia de fragmentación interna. El tamaño que ocupa un fichero en disco es múltiplo del tamaño del cluster.

Estructura en disco

- Ejemplos de Sistemas de Ficheros:

FAT

UNIX



Agenda

Semántica de coutilización

- Especifica el efecto de varios procesos accediendo de forma simultánea al mismo fichero y cuando se hacen efectivas las modificaciones
  - Cualquier forma de acceso tiene problemas cuando varios usuarios trabajan con el fichero simultáneamente
- Tipos de semánticas:
  - Semántica UNIX (POSIX)
  - Semántica de sesión
  - Semántica de versiones
  - Semántica de ficheros inmutables
- Semántica UNIX (POSIX)
  - Las escrituras son inmediatamente visibles para todos los procesos con el fichero abierto
  - Los procesos pueden compartir ficheros.
    - Si existe relación de parentesco pueden compartir el marcador de posición.
    - La coutilización afecta también a los metadatos

Semántica de coutilización

- Semántica de sesión:
  - Las escrituras que hace un proceso no son inmediatamente visibles para los demás procesos con el fichero abierto
  - Cuando se cierra el fichero los cambios se hacen visibles para las futuras sesiones
  - Un fichero puede asociarse temporalmente a varias imágenes
- Semántica de versiones
  - Las actualizaciones se hacen sobre copias con nº versión
  - Sólo son visibles cuando se consolidan versiones
  - Sincronización explícita si se requiere actualización inmediata
- Semántica de ficheros inmutables
  - Una vez creado el fichero sólo puede ser compartido para lectura y no cambia nunca

Tablas del Servidor de Ficheros

- Locales, una por proceso:
  - Tabla de descriptores de ficheros abiertos (TFA)
- Globales, una para todo el sistema:
  - Tabla intermedia de posiciones (TIP).
  - Tabla intermedia de nodos-i (TIN).