Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Índice

- El dispositivo lógico de almacenamiento
- Asignación de espacio de almacenamiento para un fichero
- Gestión del espacio libre
- Directorios

Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Resultados de aprendizaje

- Explicar los mecanismos de gestión de sistemas de ficheros y resolver casos de uso y diseño
    - Definir el concepto de sistema de ficheros lógico y explicar su motivación
    - Analizar y comparar las alternativas básicas que existen para implementar ficheros y directorios
    - Explicar diferentes mecanismos para gestionar el espacio libre de un dispositivo por bloques

Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Bibliografía

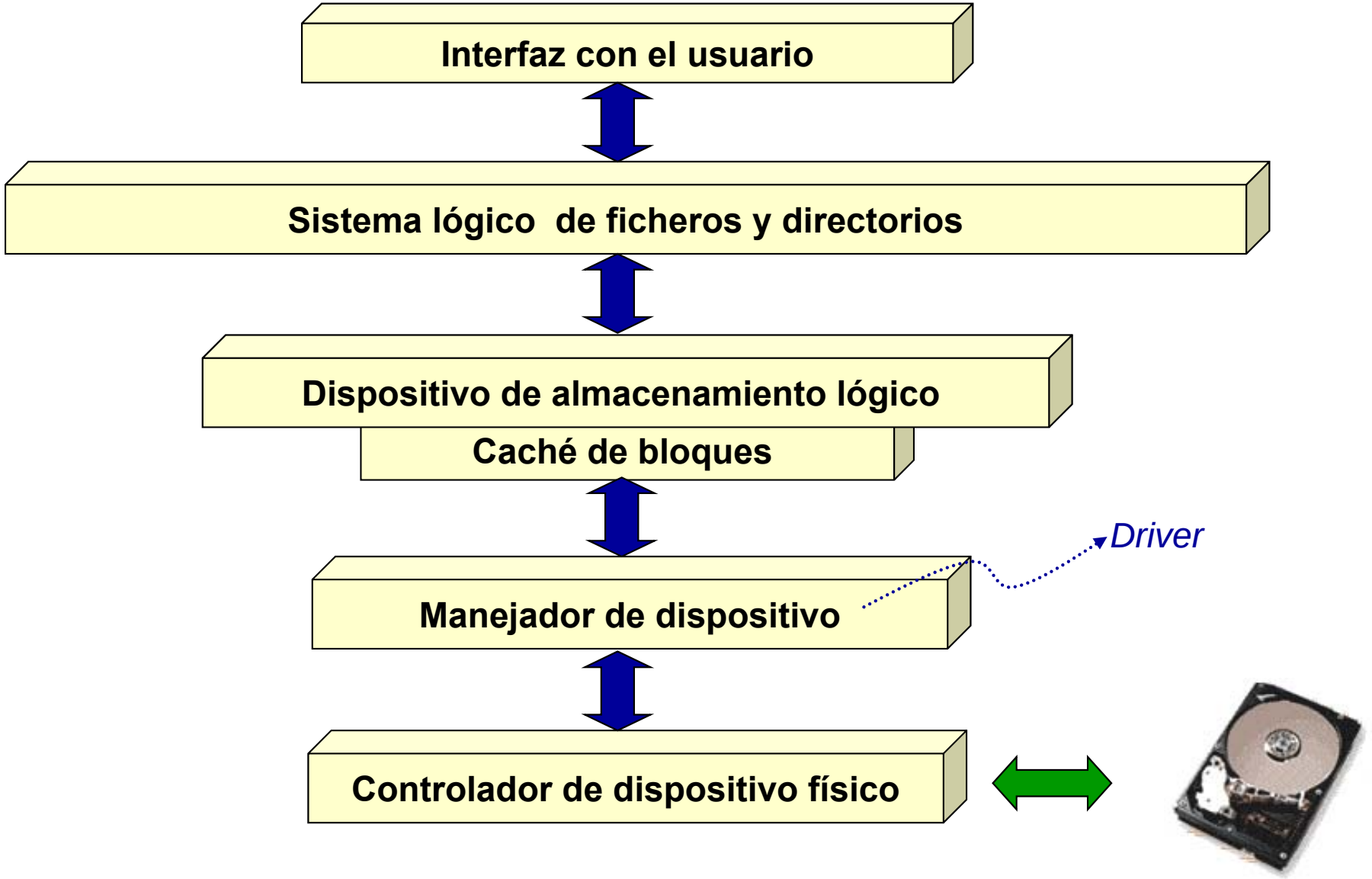
- M.A. Castaño et al. *Problemas de Sistemas Operativos.* Colección "Materials", nº 109 del Servicio de Publicaciones de la UJI. 2000. Capítulo 5.
- J. Carretero et al. *Sistemas Operativos: Una Visión Aplicada.* McGraw-Hill. 2007. Capítulo 9

Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Índice

Implementación del sistema de ficheros



Dispositivo físico vs. dispositivo lógico

- **Bloques funcionales del sistema de ficheros:**
  - **Subsistema de ficheros lógico:**
    - Parte del SO con el que interactúan los usuarios
    - Trabaja con bloques lógicos
  - **Subsistema de ficheros físico:**
    - Parte del SO que interactúa con el dispositivo físico de almacenamiento masivo
    - Conoce la estructura HW del dispositivo físico
    - Oculta las características del dispositivo físico al resto de niveles del SO → Esquema de ficheros independiente del dispositivo
    - Transformación de número de bloque en parámetros físicos (cara, pista, sector)

Dispositivo físico vs. dispositivo lógico

- **Dispositivo físico** (disco de almacenamiento):
  - Número de caras
  - Número de cilindros
  - Número sectores por cilindro
  - Capacidad de cada sector

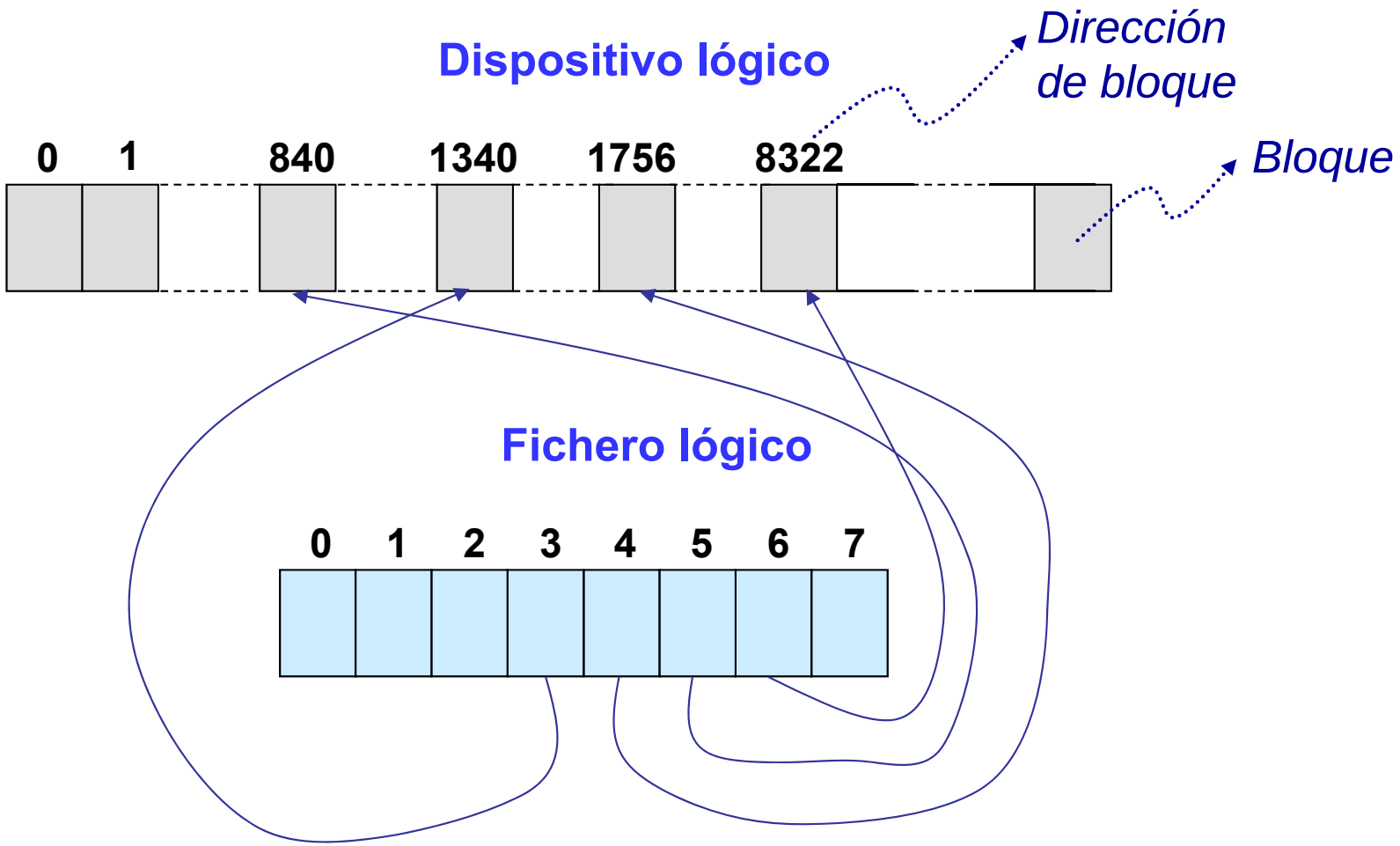
Secuencia o vector de bloques

- **Dispositivo lógico**:
  - Número de bloques
  - Capacidad de cada bloque
    - Tamaños habituales: 512 bytes, 1024 bytes

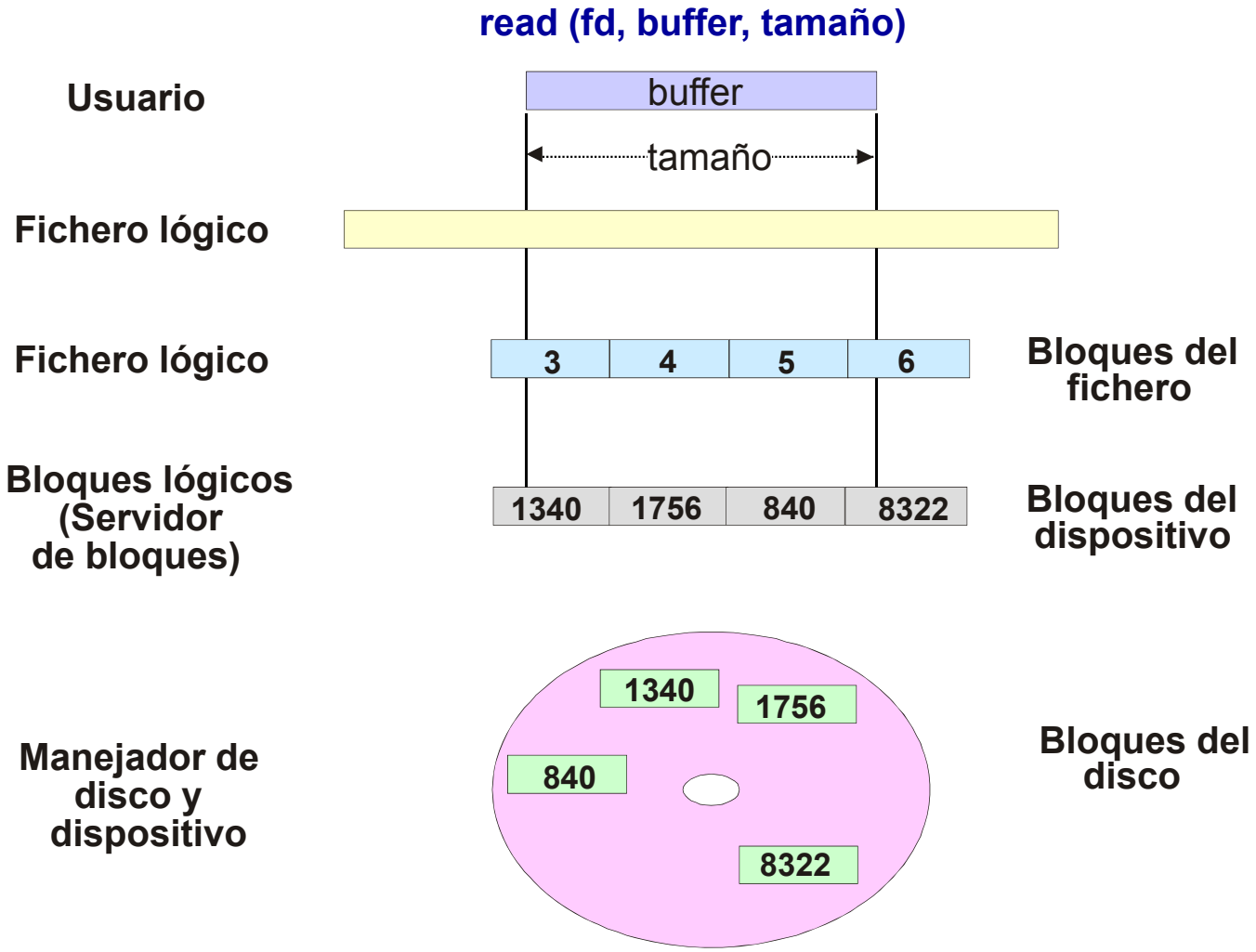
Dispositivo físico vs. dispositivo lógico

- El dispositivo lógico se compone de un conjunto de bloques
- Bloque:
  - Agrupación lógica de sectores de disco
  - Unidad mínima de transferencia del SF
  - Permite optimizar la eficiencia de la E/S de los dispositivos secundarios de almacenamiento
- Cada bloque tiene asignada una dirección que lo identifica: **referencia** o **dirección de bloque**
- Esta dirección lógica es traducida en dirección física de dispositivo: cara, pista, sector
- A cada bloque lógico del fichero se le asigna un bloque de dispositivo

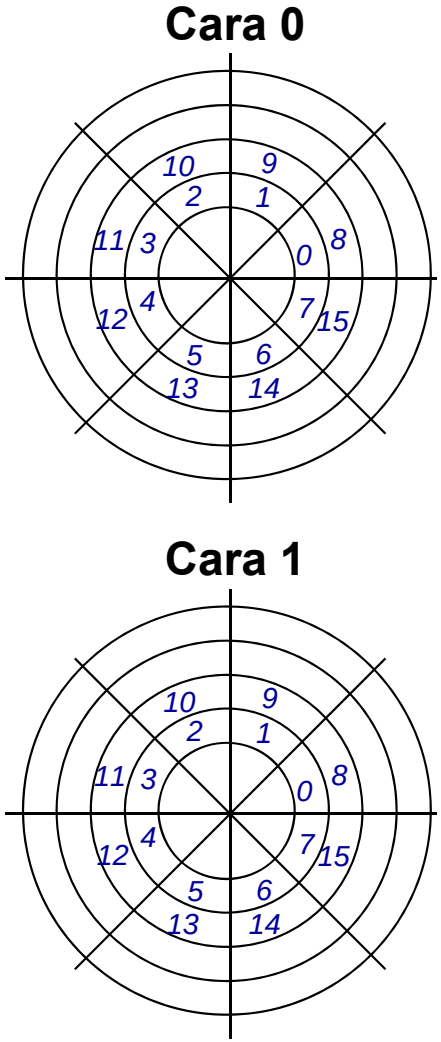
Dispositivo físico vs. dispositivo lógico



Dispositivo físico vs. dispositivo lógico

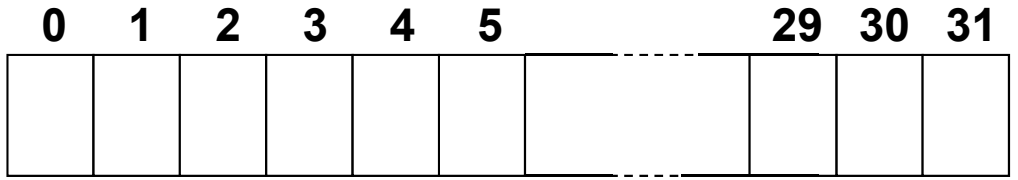


Dispositivo físico vs. dispositivo lógico



Supongamos que el disco tiene **8 sectores por cilindro**

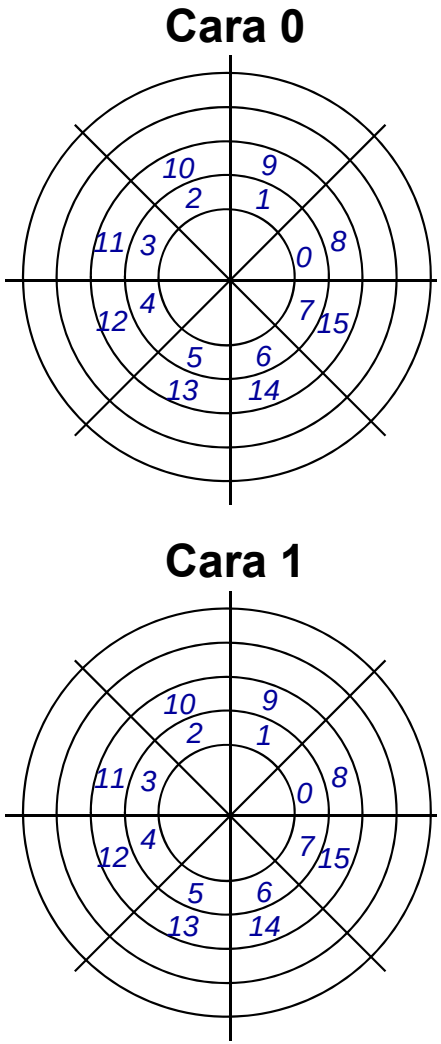
Dispositivo lógico: **32 bloques (1 bloque = 2 sectores)**



Función de transformación del dispositivo lógico al físico:

	cara	cilindro	sector
Bloque lógico **J**:	**0 ,**	**J DIV 8 ,**	**J MOD 8**
	1 ,	**J DIV 8 ,**	**J MOD 8**

Dispositivo físico vs. dispositivo lógico



Bloque lógico **J**:

cara	cilindro	sector
0 ,	**J DIV 8 ,**	**J MOD 8**
1 ,	**J DIV 8 ,**	**J MOD 8**

Bloque	Cilindro	Sector
0	0	0
1	0	1
	…	
7	0	7
8	1	0
9	1	1
	…	

Nota

De aquí en adelante vamos a suponer que trabajamos con un dispositivo lógico que tiene un tamaño de bloque definido por la constante

TAMANYO_BLOQUE.

También supondremos que para identificar (referenciar) un bloque se utiliza un número entero, representado por **TR** (tamaño referencia) bytes.

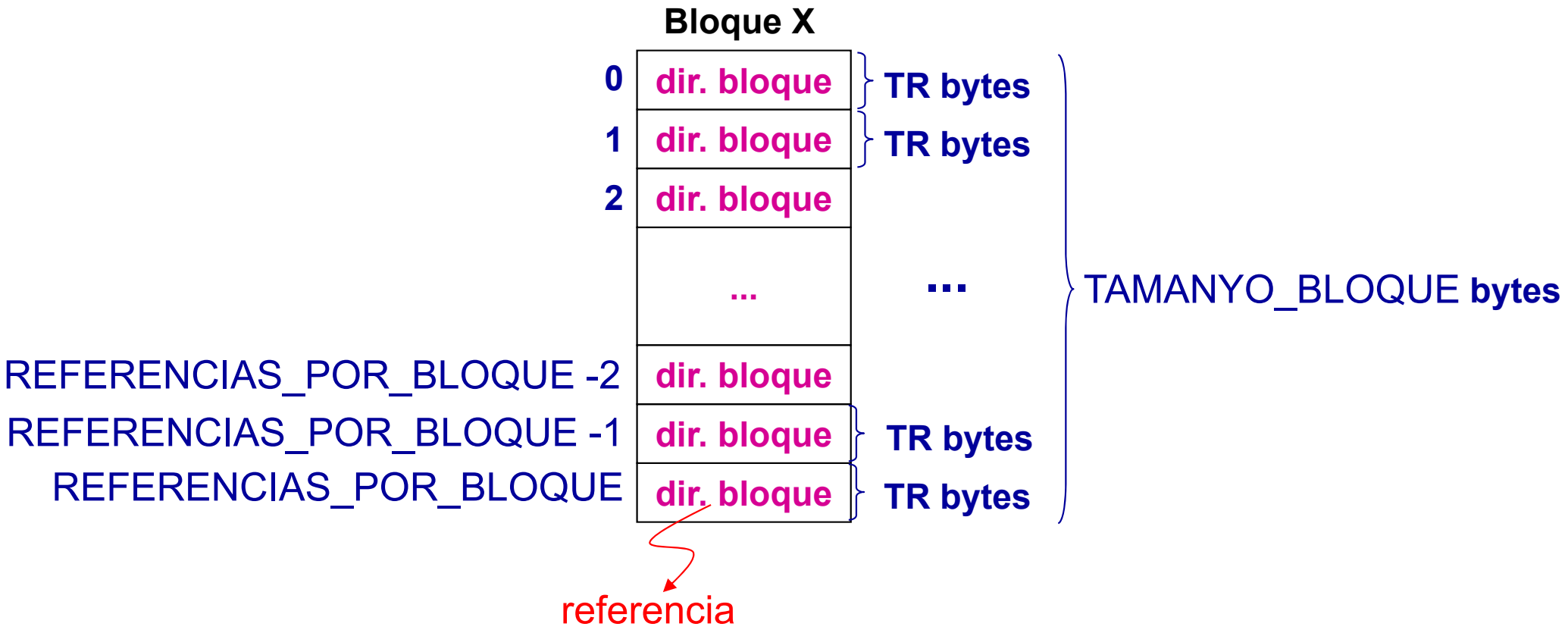
Referencia ≡ Dirección de bloque

¿Cuántas direcciones de bloque (referencias) caben en un bloque?

TAMANYO_BLOQUE / TR = **REFERENCIAS_POR_BLOQUE.**

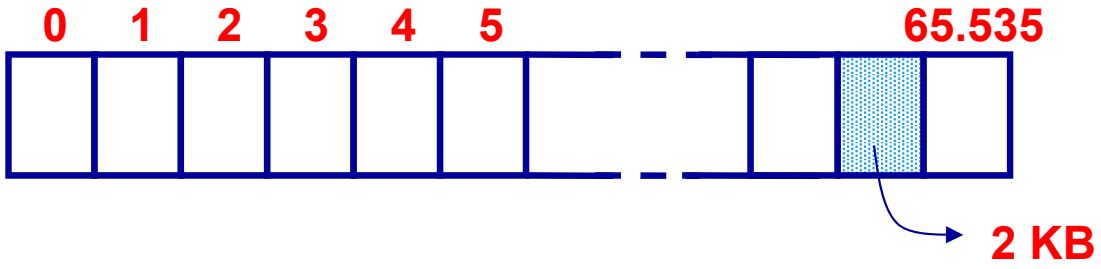
Nota

$$\text{REFERENCIAS\_POR\_BLOQUE} = \frac{\text{TAMANYO\_BLOQUE}}{\text{TR}}$$



Nota

Sea un dispositivo lógico con 65.536 bloques, cada uno de ellos de capacidad 2 KB.



Entonces:

- 65.536 bloques = 2^{16} bloques 16 bits = 2 bytes → TR = 2 bytes
- REFERENCIAS_POR_BLOQUE = $\frac{\text{TAMANYO\_BLOQUE}}{\text{TR}}$ =

$$= \frac{2^{11}\ \frac{\text{by}}{\text{bloque}}}{2\ \frac{\text{by}}{\text{referencia}}} = 1.024\ \frac{\text{referencias}}{\text{bloque}}$$

Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Índice

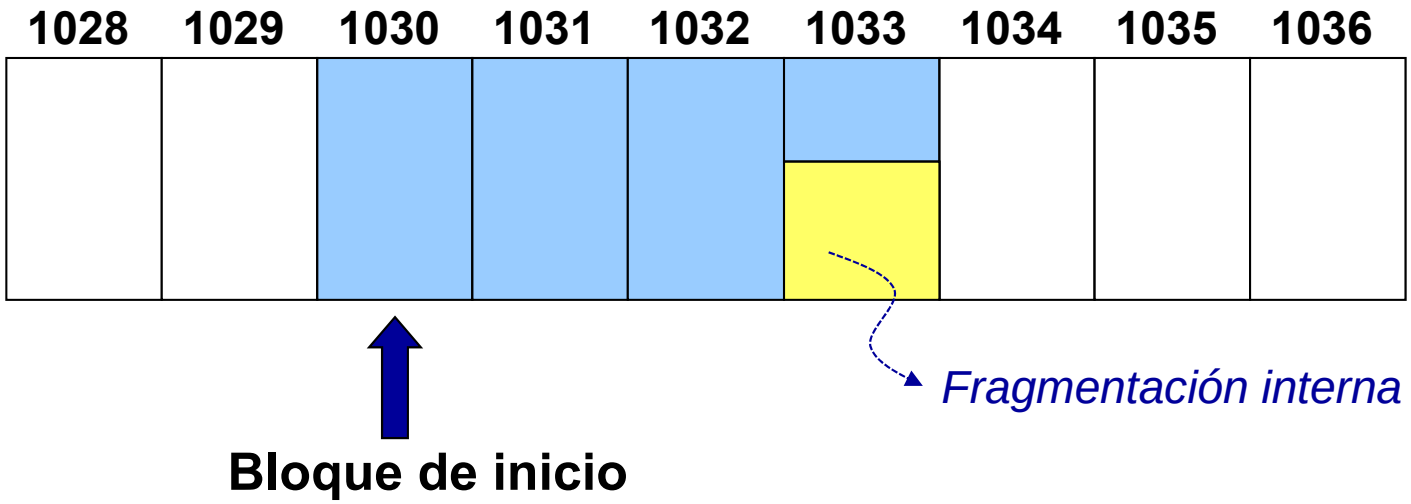
- El dispositivo lógico de almacenamiento
- Asignación de espacio de almacenamiento para un fichero
- Gestión del espacio libre
- Directorios

Asignación de espacio a ficheros

- **Asignación CONTIGUA**
  - Los bloques de un fichero constituyen un **conjunto consecutivo** de bloques del dispositivo de almacenamiento lógico
- **Asignación ENLAZADA**
  - Los bloques de un fichero son nodos de una **lista enlazada** de bloques del dispositivo de almacenamiento lógico
- **Asignación INDEXADA**
  - Por cada fichero existe una **tabla de índices** que indica los bloques asignados a este fichero

Asignación de espacio a ficheros

- **Asignación CONTIGUA**



Suponiendo un tambaño de bloque de 1024 y un tamaño del fichero de 3742 bytes, en el bloque 1033 se están desaprovechando 354 bytes (**fragmentación interna**)

Asignación de espacio a ficheros

- **Asignación CONTIGUA**
  - Ventajas:
    - Representación sencilla
    - Facilidad de acceso secuencial
    - Facilidad de acceso directo

      ¿Cómo localizar el byte pos de un fichero?

      bloque_lógico = pos DIV TAMANYO_BLOQUE

      posición_bloque_lógico = pos MOD TAMANYO_BLOQUE

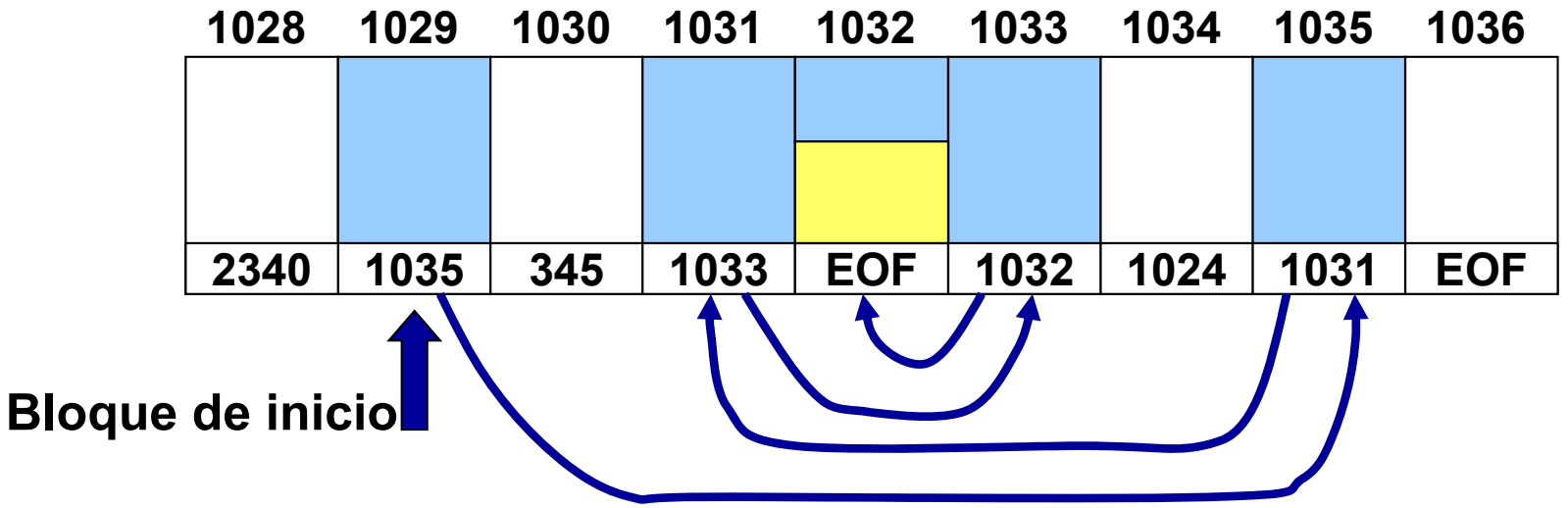
      N° de byte relativo al inicio del fichero
  - Desventajas:
    - Genera **fragmentación externa** → Necesaria compactación
    - Problemas con el crecimiento de los ficheros
    - Útil para sistemas de ficheros de sólo lectura

      Muchos huecos no utilizables

Asignación de espacio a ficheros

- **Asignación ENLAZADA**

  En cada bloque, el SO tiene reservado un espacio para almacenar el indicador del siguiente bloque que contiene datos del fichero



Bloques de un fichero: 1029 → 1035 → 1031 → 1033 → 1032

Asignación de espacio a ficheros

- **Asignación ENLAZADA**
  - Ventajas:
    - No problemas con el crecimiento de los ficheros
    - Facilidad de acceso secuencial
  - Desventajas:
    - Ineficiencia con el acceso directo
    - Problemas de consistencia: Si un bloque pasa a ser defectuoso, se pierden las referencias al resto de bloques del fichero
    - En cada bloque el espacio de almacenamiento no es potencia de 2

Hay que leer todos los bloques previso al bloque destino

Complica el cálculo del número de bloque en el que está un determinado byte del fichero

Asignación de espacio a ficheros

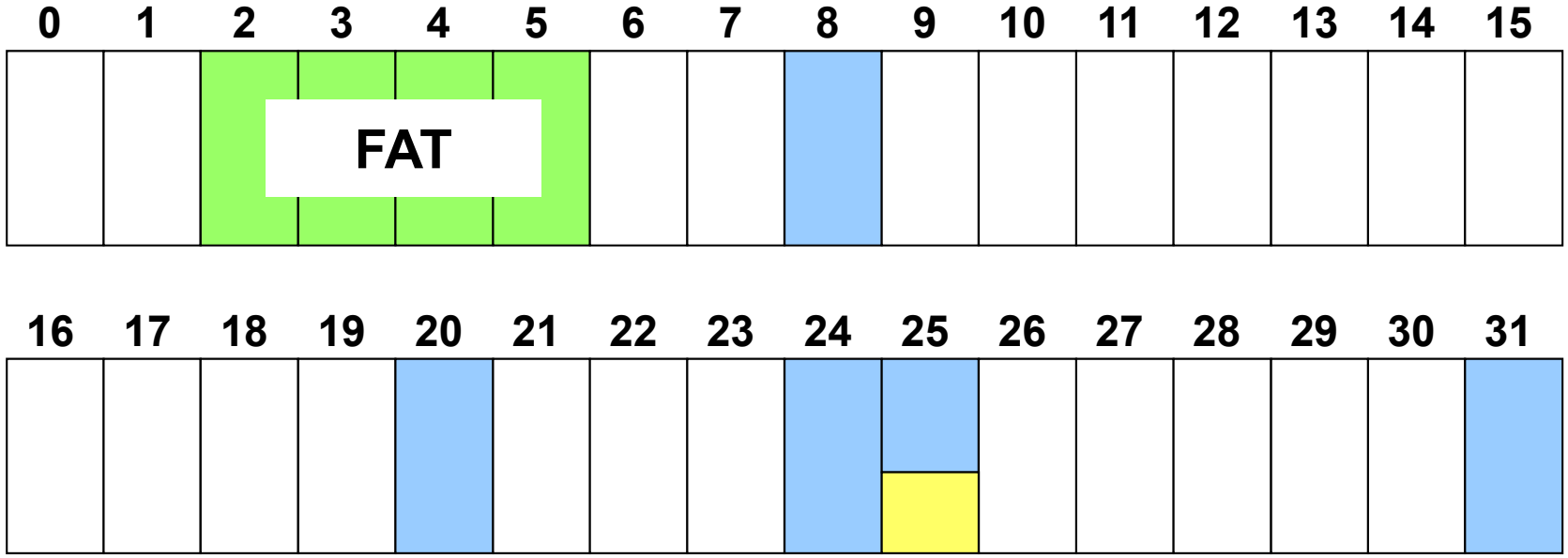
- **Asignación ENLAZADA**
  - Modificacón: Tabla de Localización de Ficheros ("File Allocation Table", FAT)

    Todos los enlaces se almacenan en un conjunto específico de bloques, denominado FAT

    (en los sistemas de ficheros de casi todos los SO de Microsoft)

  - Ventajas:
    - Facilidad de acceso directo
    - Posibilidad de duplicar los índices (la FAT) en prevención de pérdidas de bloques

Asignación de espacio a ficheros



bloque 2	
0	X
1	X
2	X
3	X
4	X
5	X
6	29
7	BAD

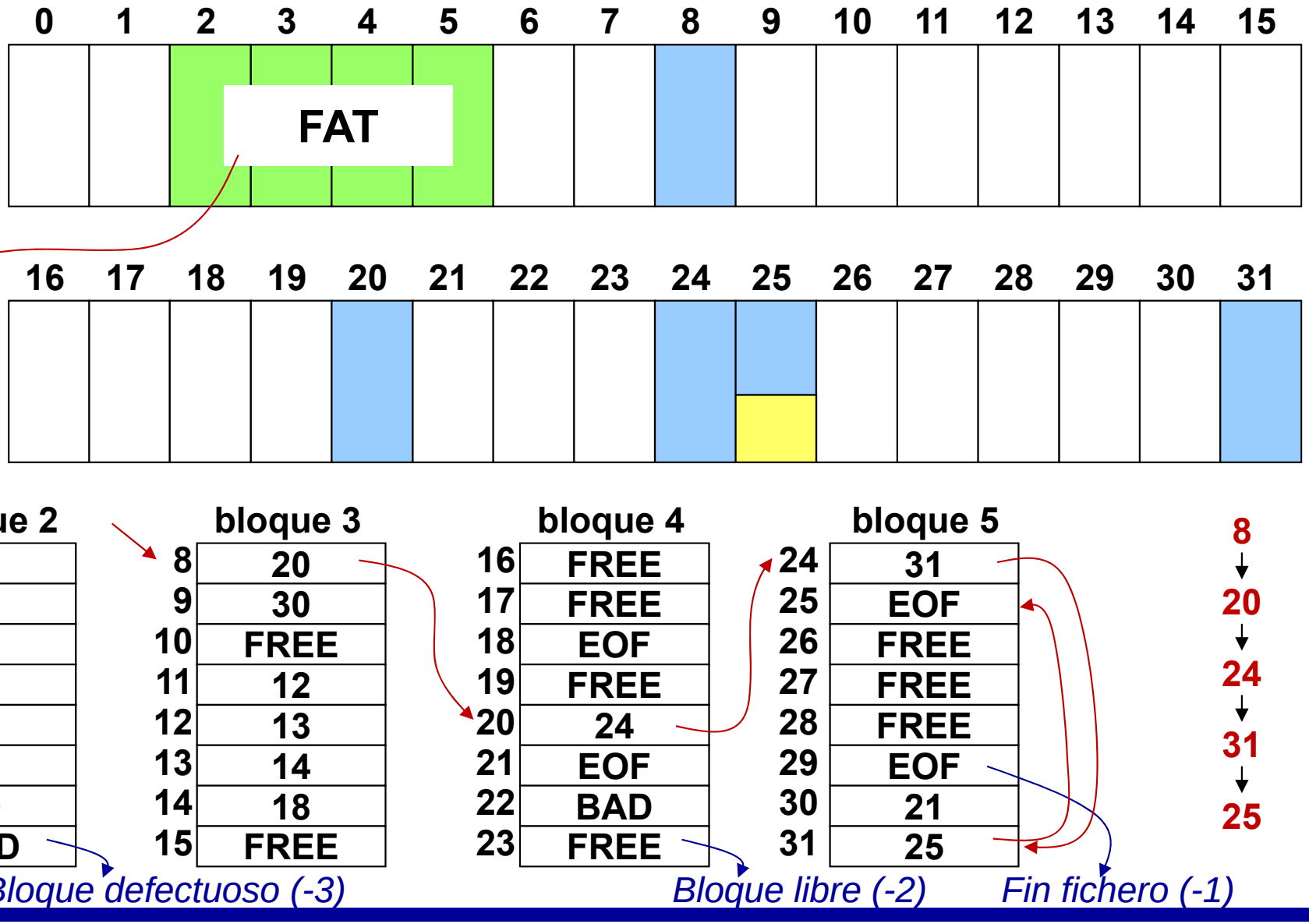
bloque 3	
8	20
9	30
10	FREE
11	12
12	13
13	14
14	18
15	FREE

bloque 4	
16	FREE
17	FREE
18	EOF
19	FREE
20	24
21	EOF
22	BAD
23	FREE

bloque 5	
24	31
25	EOF
26	FREE
27	FREE
28	FREE
29	EOF
30	21
31	25

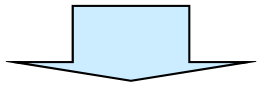
Bloque defectuoso (-3) *Bloque libre (-2)* *Fin fichero (-1)*

Asignación de espacio a ficheros



Asignación de espacio a ficheros

- **Asignación enlazada (FAT)**
  - Problemas:
    - Aunque sólo haya un fichero abierto en el sistema, potencialmente hace falta toda la FAT en memoria
    - Discos muy grandes implican ocupación de la FAT grande en memoria
    - Sistema lento para accesos directos en ficheros grandes
  - Solución:
    - Mantener la lista de bloques para cada fichero separada



Asignación indexada

Asignación de espacio a ficheros

- **Asignación INDEXADA**

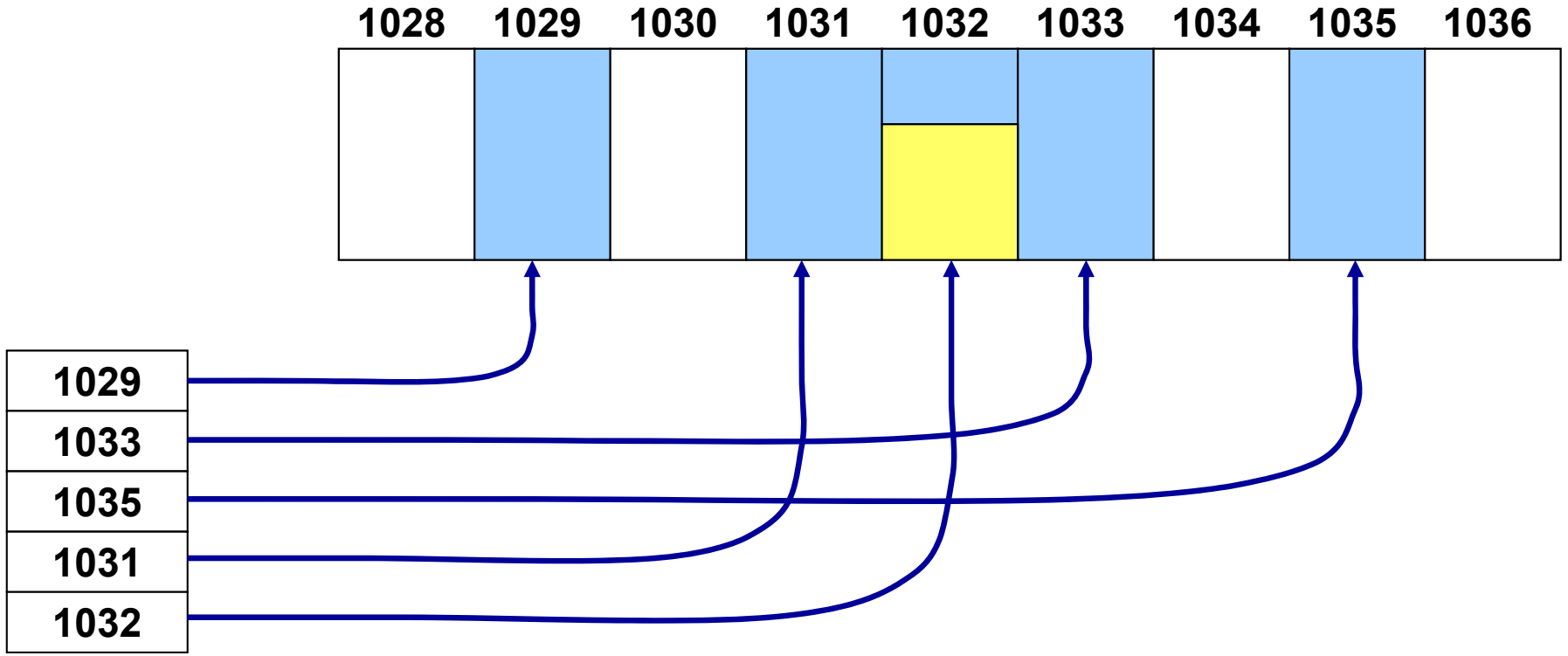


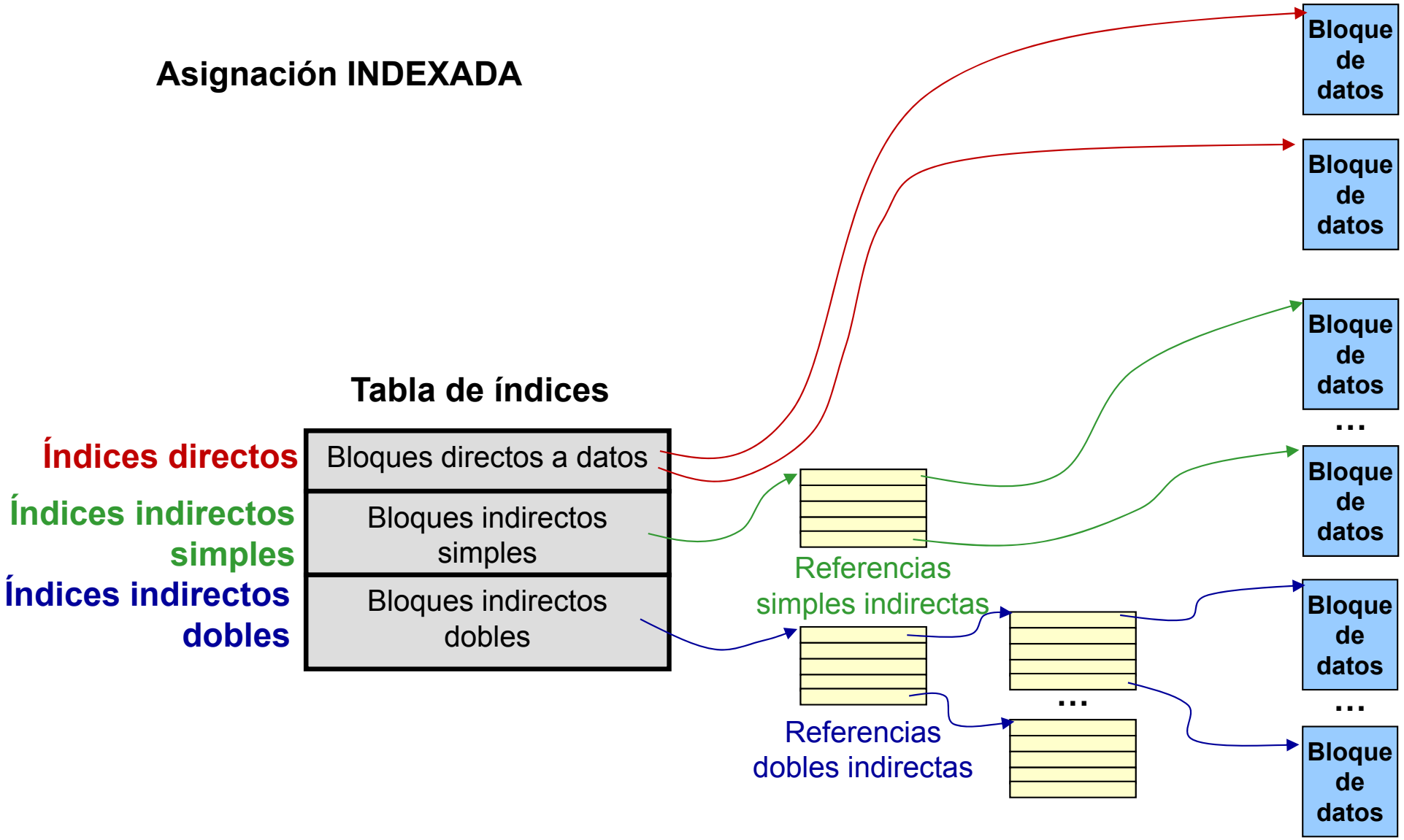
Tabla de índices
(de un fichero)

Asignación de espacio a ficheros

- **Asignación INDEXADA**
  - Ventajas:
    - Basta con traer a memoria el bloque de índices donde están las referencias a los bloques de datos para tener acceso al bloque de datos.

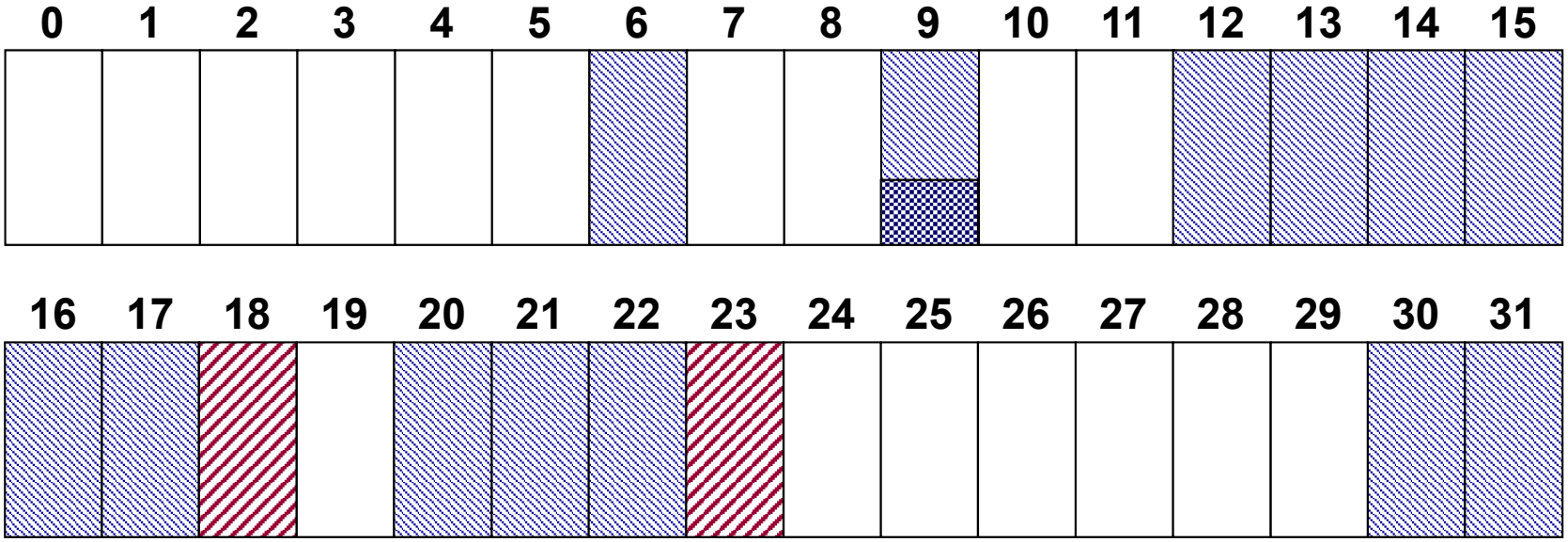
      Si una referencia de bloque ocupa 4 bytes y el bloque es de 4 Kbytes, con un único acceso a disco tendremos 1024 referencias a bloques del archivo
    - Facilidad de acceso secuencial y directo
  - Desventajas:
    - El crecimiento del fichero está limitado al tamaño de la tabla de índices. Para solucionar este problema se puede considerar una estructura con **varios niveles de tablas de índices**

Asignación de espacio a ficheros



Asignación de espacio a ficheros

Asignación INDEXADA



0	16
1	18
2	23

Tabla de índices

1 índice directo
2 índices indirectos simples

bloque 18

20
30
31
12
13
14
15
17

bloque 23

21
22
6
9
FREE
FREE
FREE
FREE

Los bloques 18 y 23 son bloques de referencias indirectas simples

Bloques de datos del fichero:

16 → 20 → 30 → 31 → 12 → 13 →

→ 14 → 15 → 17 → 21 → 22 → 6 → 9

Asignación de espacio a ficheros

- **Asignación INDEXADA**
  - Suponiendo que RPB (REFERENCIAS__POR_BLOQUE) es el número màximo de referencias en un bloque:
    - Índice simple indirecto:
      - Referencia a RPB bloques de datos
    - Índice doble indirecto:
      - Referencia a RPB^2 bloques de datos
    - Índice triple indirecto:
      - Referencia a RPB^3 bloques de datos

Asignación de espacio a ficheros

- **Asignación INDEXADA**
  - Si RPB = 1024 y el tamaño de bloque es 4096,
    - Con 1 índice triple indirecto se pueden acceder hasta:

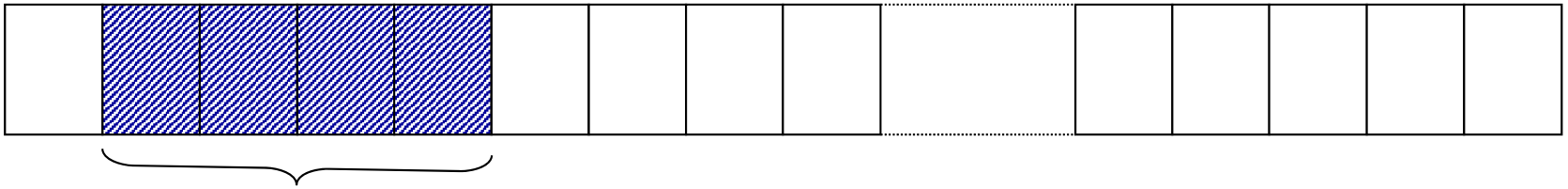
      $(2^{10})^3 \times 2^{12}$ bytes = 2^{42} bytes = 4 TB
    - Tamaño máximo de un fichero:

      $[\ 2 + 2^{10} + (2^{10})^2 + (2^{10})^3\ ] \times 2^{12}$ bytes = O (TB)

2^{10} =1KB
2^{20} =1MB
2^{30} =1GB
2^{40} =1TB

Asignación de espacio a ficheros

- **Asignación INDEXADA**
  - Las tablas de índices son almacenadas por el SO en un conjunto de bloques del dispositivo reservados a tal efecto
  - Esto limita el número máximo de ficheros que pueden existir en un sistema de ficheros → *Tantos ficheros como tablas de índices*



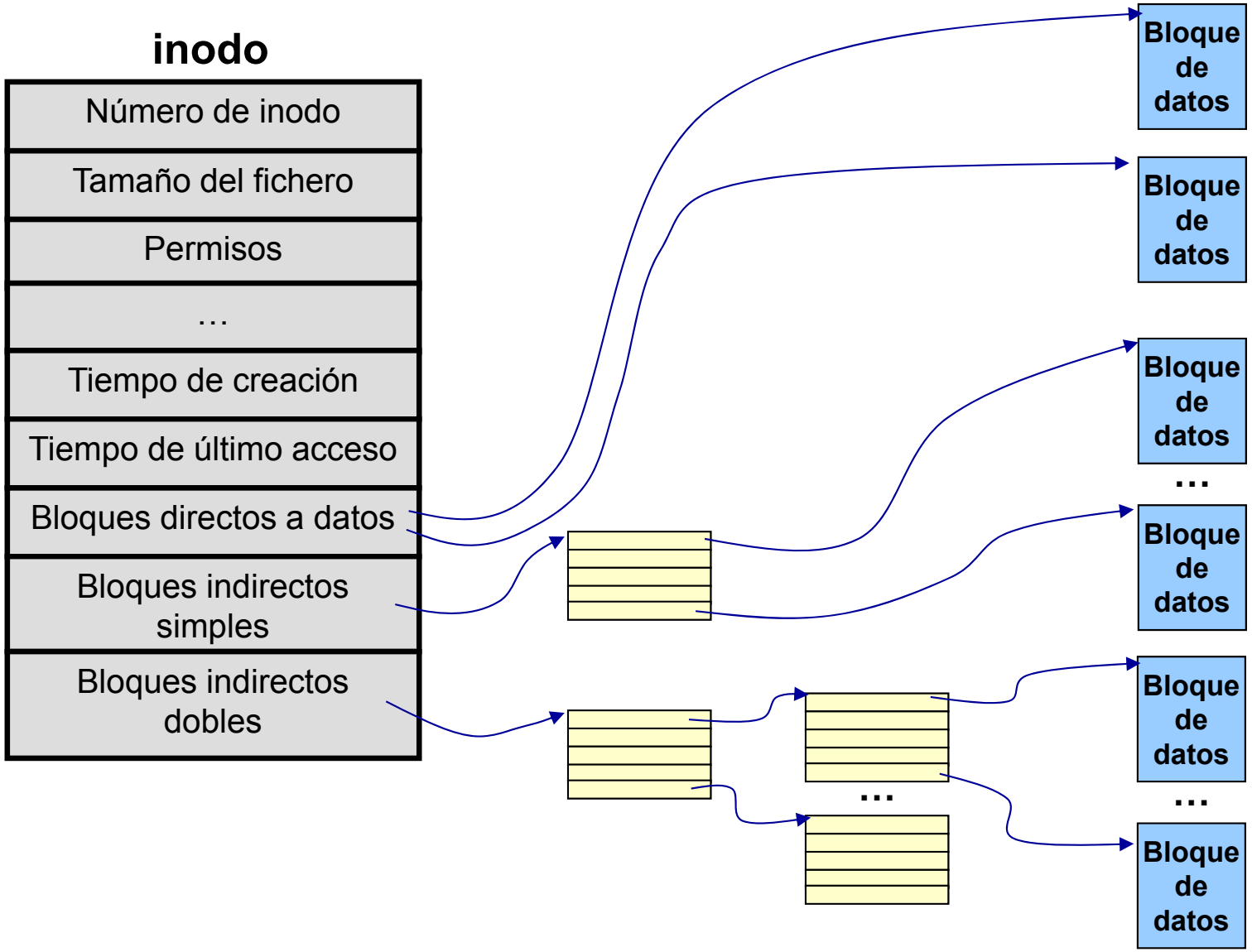
Bloques reservados para las tablas de índices

Si en cada bloque caben 128 tablas de índices, el número máximo de ficheros que pueden existir en este dispositivo es de 512

Asignación de espacio a ficheros

- **Asignación INDEXADA**
  - En la tabla de índices, junto con las referencias, también se almacenan atributos del fichero
  - Sistema de inodos de UNIX:
    - Cada fichero tiene assignado un INODO, que no es más que una estructura de datos almacenada en disco donde el SO mantiene los índices del fichero junto con sus atributos

Asignación de espacio a ficheros



Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Índice

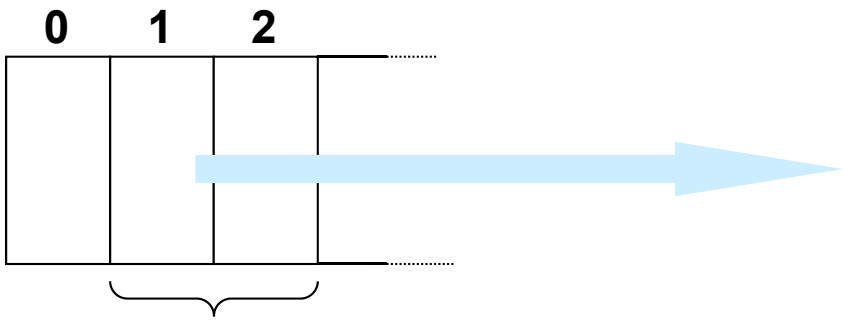
- El dispositivo lógico de almacenamiento
- Asignación de espacio de almacenamiento para un fichero
- Gestión del espacio libre
- Directorios

Gestión del espacio libre

- **Identificación de los bloques libres en el disco lógico:**
  - **Mapa de bits**: Cada bloque del disco queda representado por un bit: Si el bit tiene valor 0 quiere decir que está libre y si tiene valor 1, que está ocupado
  - **Lista de bloques libres**: Se enlazan todos los bloques libres en una lista

Gestión del espacio libre

- **Mapa de bits**

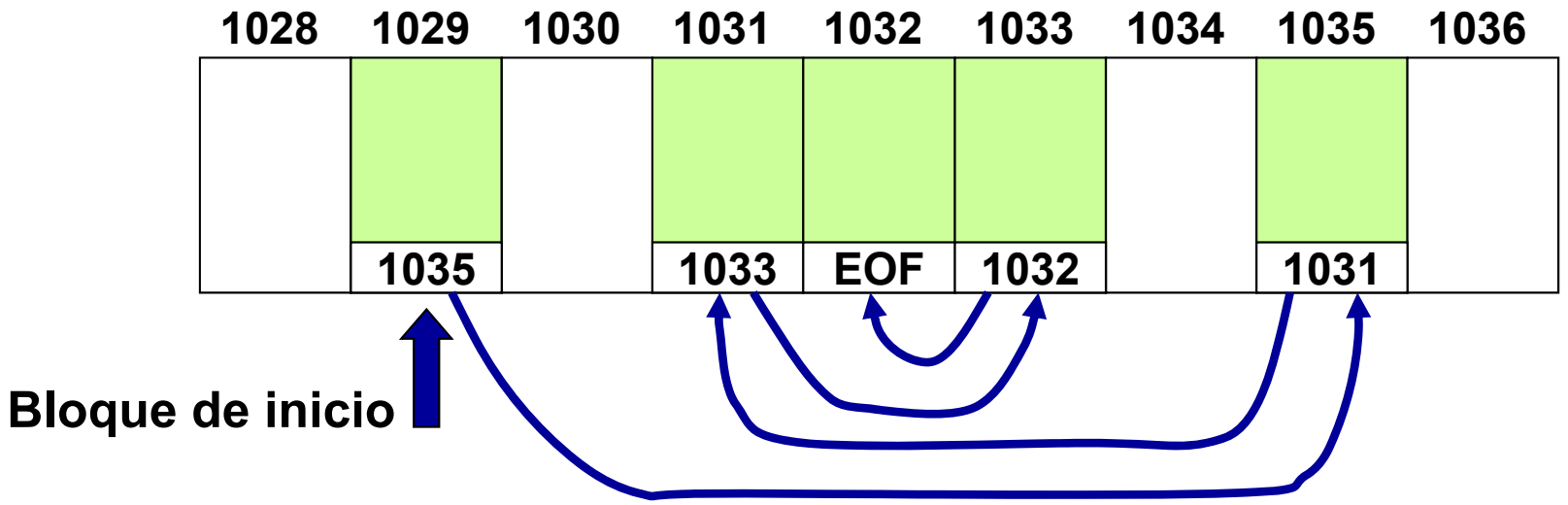



```
11111111111111111111111111111111
01111111101110000000000111101101
01100111101111101111111011100111
00000001111111111100000011100000
11111111110000001110111111111101
00000000000000000001111111000000
00001111101111111111011111111011
................................
                                
                                
                                
                                
                                
                                
................................
00010100000000000000100000000000
00000000000000000000000000000000
```

Bloques libres: 32, 42, 46-54, 59, 62, 64, 67-68, 73, etc.

Gestión del espacio libre

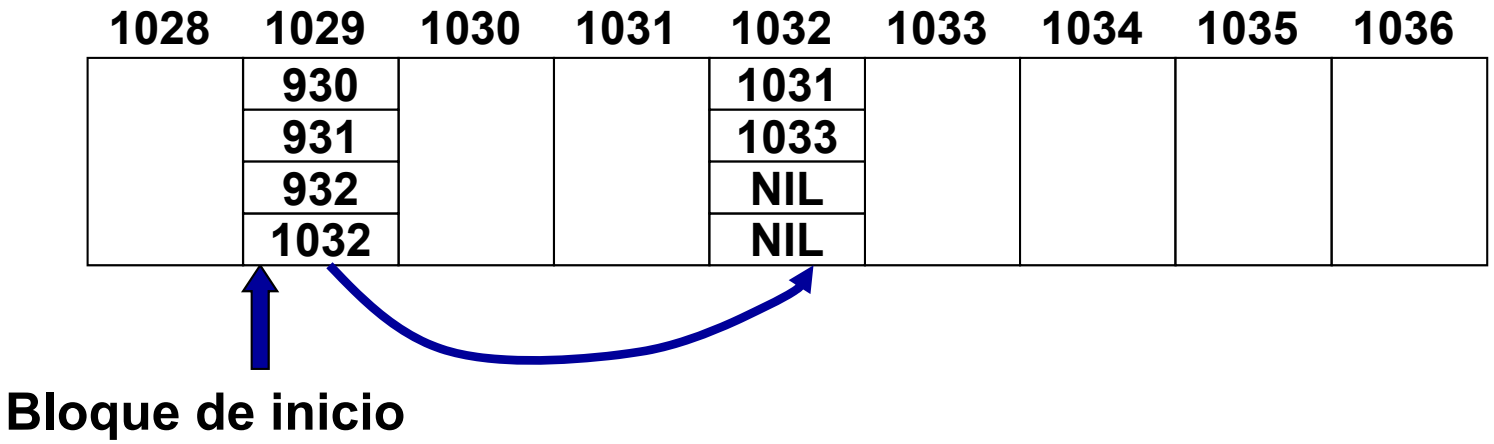
- **Lista de bloques libres:**
  - Cada bloque contiene la referencia del siguiente bloque libre



Bloques libres: 1029, 1035,1031,1033,1032

Gestión del espacio libre

- **Lista de bloques libres:**
  - Cada bloque tiene referencias a otros bloques libres



Bloques libres: 1029, 930-932, 1032, 1031, 1033

Gestión del espacio libre

- **Mapa de bits:**
  - Ventajas:
    - Gestión mediante operaciones lógicas y de manipulación de bits
    - Poco espacio en disco
  - Desventajas:
    - Problemas si el bloque del mapa de bits pasa a ser defectuoso (Solución: Replicar el mapa en varias partes del dispositivo)

Gestión del espacio libre

- **Lista de bloques:**
  - Ventajas:
    - Facilidad en la obtención de identificativos de bloques libres
  - Desventajas:
    - Interacción con el disco para gestionar los bloques libres

Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Índice

- El dispositivo lógico de almacenamiento
- Asignación de espacio de almacenamiento para un fichero
- Gestión del espacio libre
- Directorios

El sistema de directorios

- **Directorio:**
  - Fichero especial que es utilitado por el SO para mantener referencias a los ficheros y subdirectorios que contiene
- **Finalidad doble de un directorio:**
  - Cara al usuario: Permite organizar los ficheros de éste
  - Cara al SO: Permite transformar un nombre de fichero en las estructuras de datos necesarias para acceder a los datos del fichero

El sistema de directorios

- **La estructura interna del directorio varía según el esquema utilizado para identificar los bloques asignados a un fichero**
  - **Lista enlazada**: En el directorio aparece el identificador del primer bloque asignado al fichero
  - **Tabla de índices**: En el directorio aparece el identificador de la tabla de índices del fichero

Tema 5. Estructura e implementación del sistema de ficheros

UNIVERSITAT JAUME·I

Índice

- El dispositivo lógico de almacenamiento
- Asignación de espacio de almacenamiento para un fichero
- Gestión del espacio libre
- Directorios

El sistema de directorios

- **Directorio:**
  - Fichero especial que es utilitado por el SO para mantener referencias a los ficheros y subdirectorios que contiene
- **Finalidad doble de un directorio:**
  - Cara al usuario: Permite organizar los ficheros de éste
  - Cara al SO: Permite transformar un nombre de fichero en las estructuras de datos necesarias para acceder a los datos del fichero

El sistema de directorios

- **La estructura interna del directorio varía según el esquema utilizado para identificar los bloques asignados a un fichero**
  - **Lista enlazada**: En el directorio aparece el identificador del primer bloque asignado al fichero
  - **Tabla de índices**: En el directorio aparece el identificador de la tabla de índices del fichero

Directorio para FAT

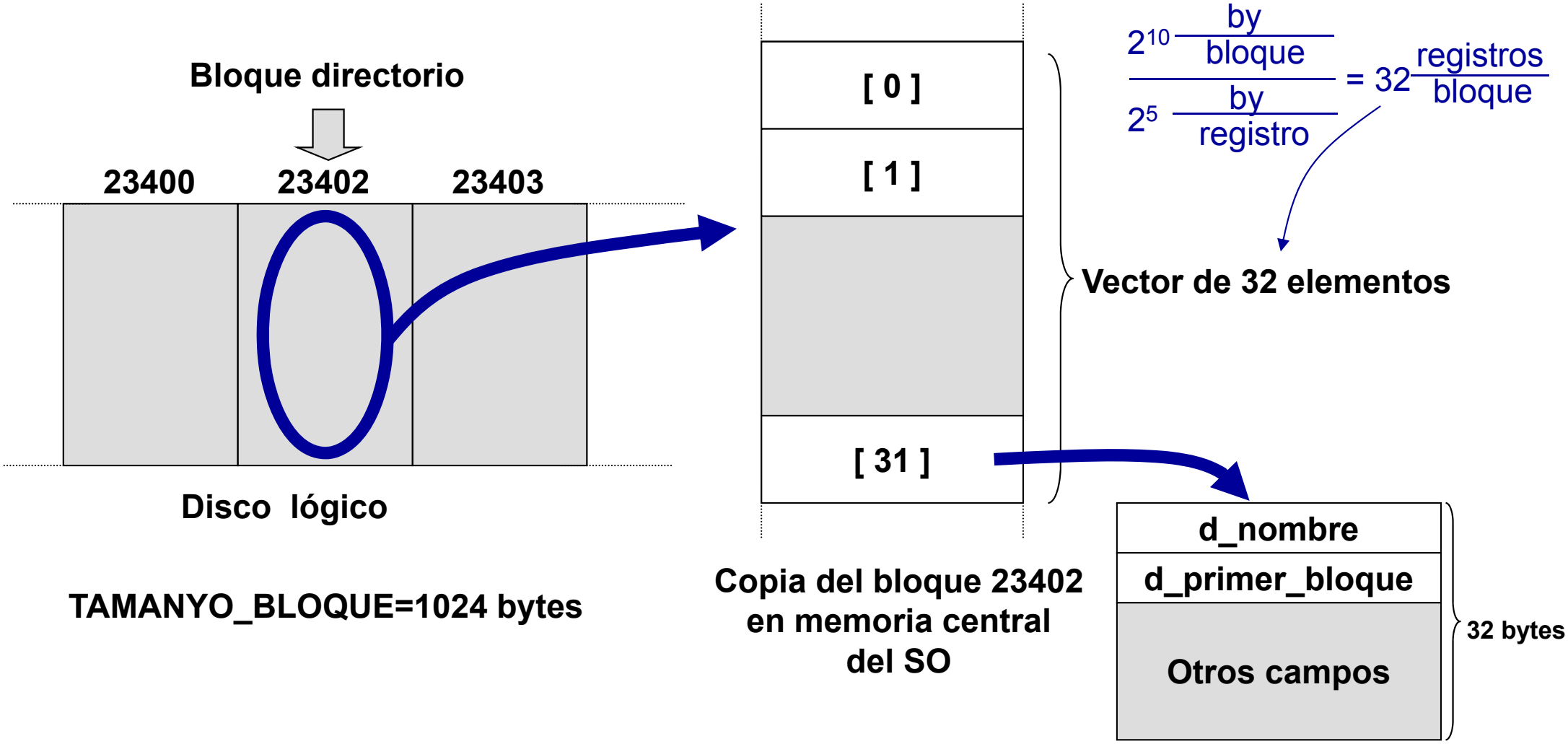
- **Especificación de una entrada de un directorio para FAT:**

```
typedef struct {
                char        d_tipo;
                int         d_tamanyo;
                int         d_permisos;
                Nombre      d_nombre;
                int         d_primer_bloque;
                Byte        d_relleno[ NRELLENO ];
    } T_Reg_Direct;

    typedef char Nombre [16 ];

    typedef char Byte;
```

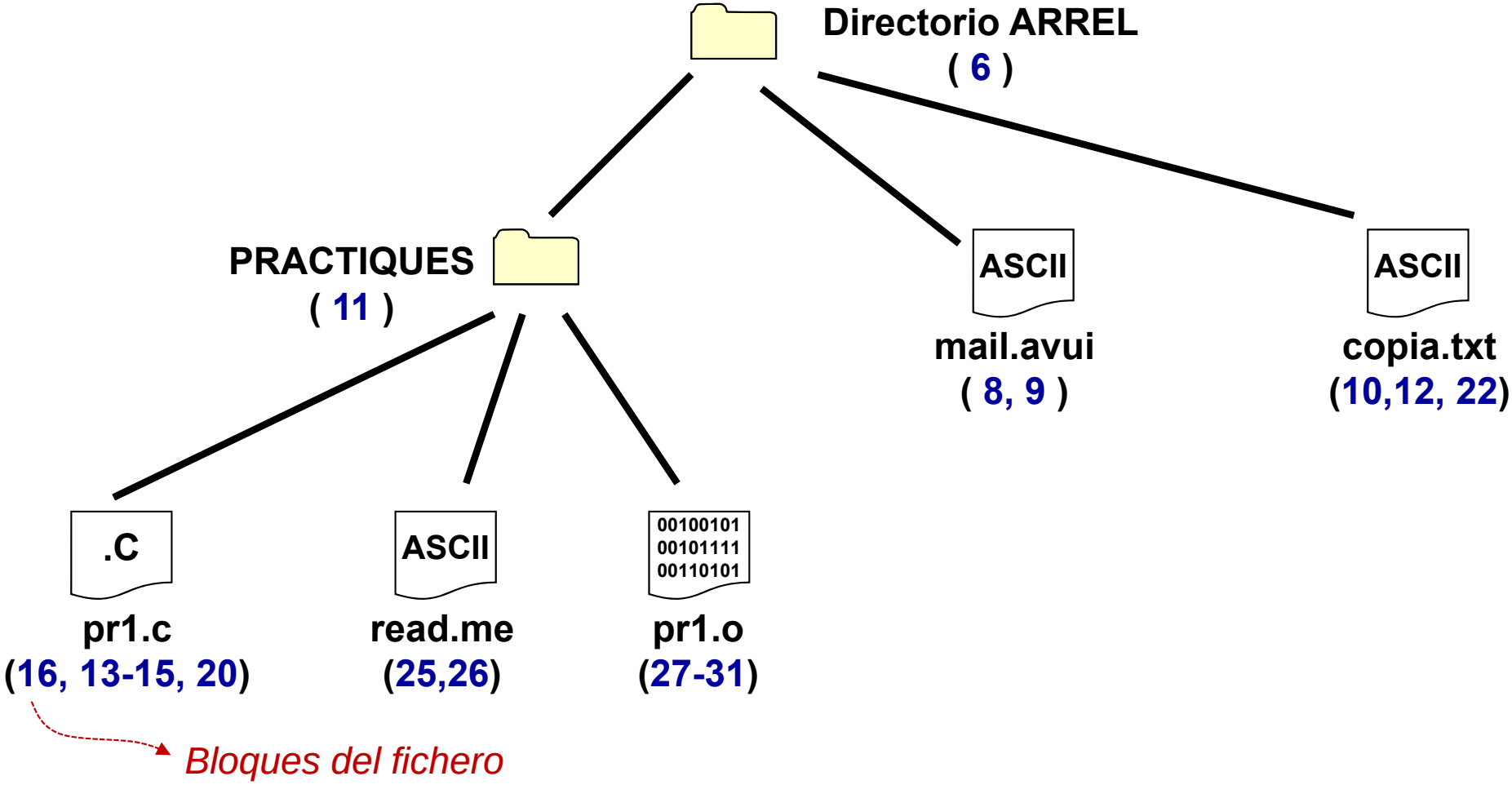
3 bytes de relleno para que la entrada de directorio sea una potencia exacta de 2

Directorio para FAT

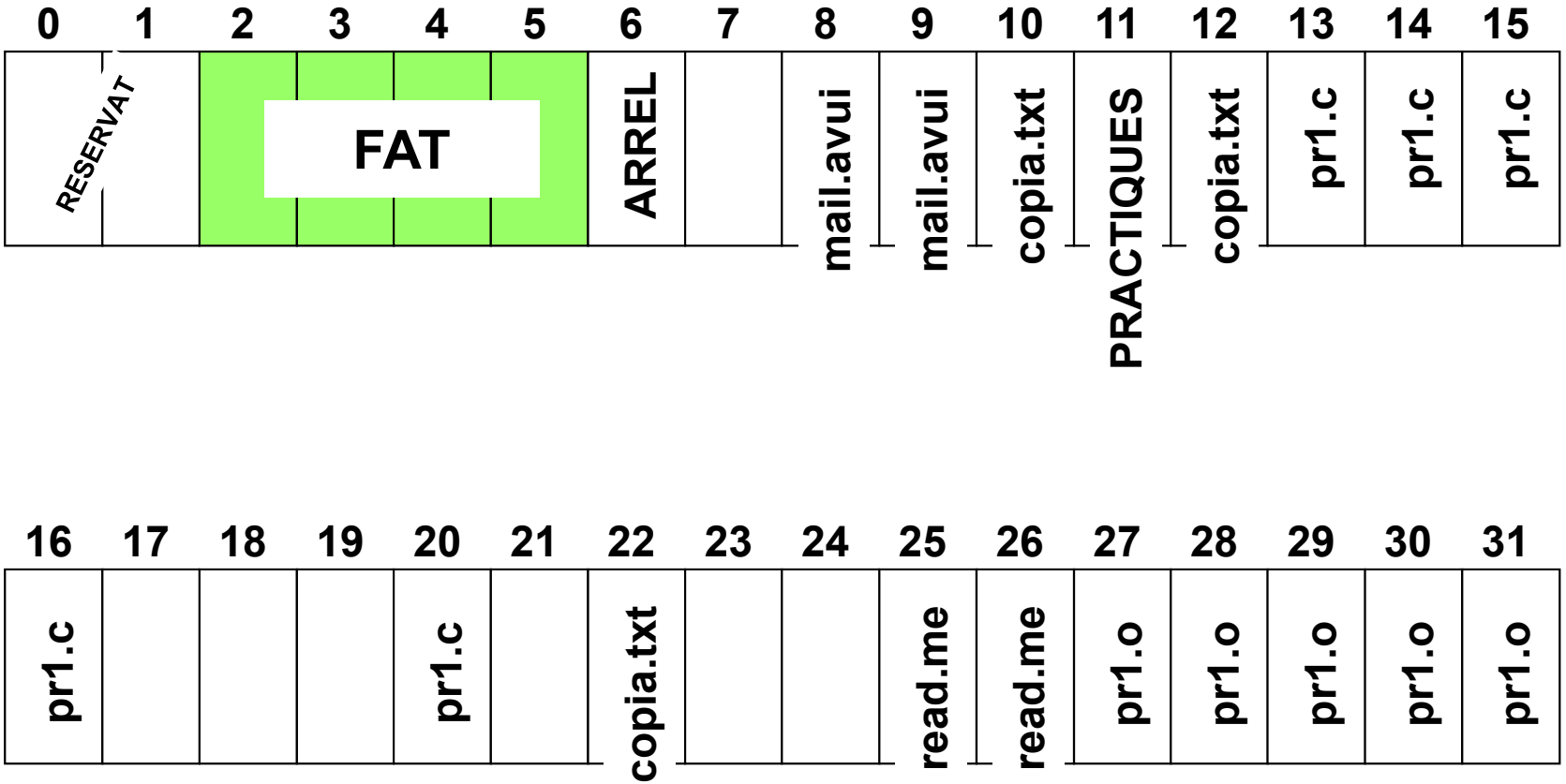
- ¿Cuántos registros de directorio caben en un bloque?



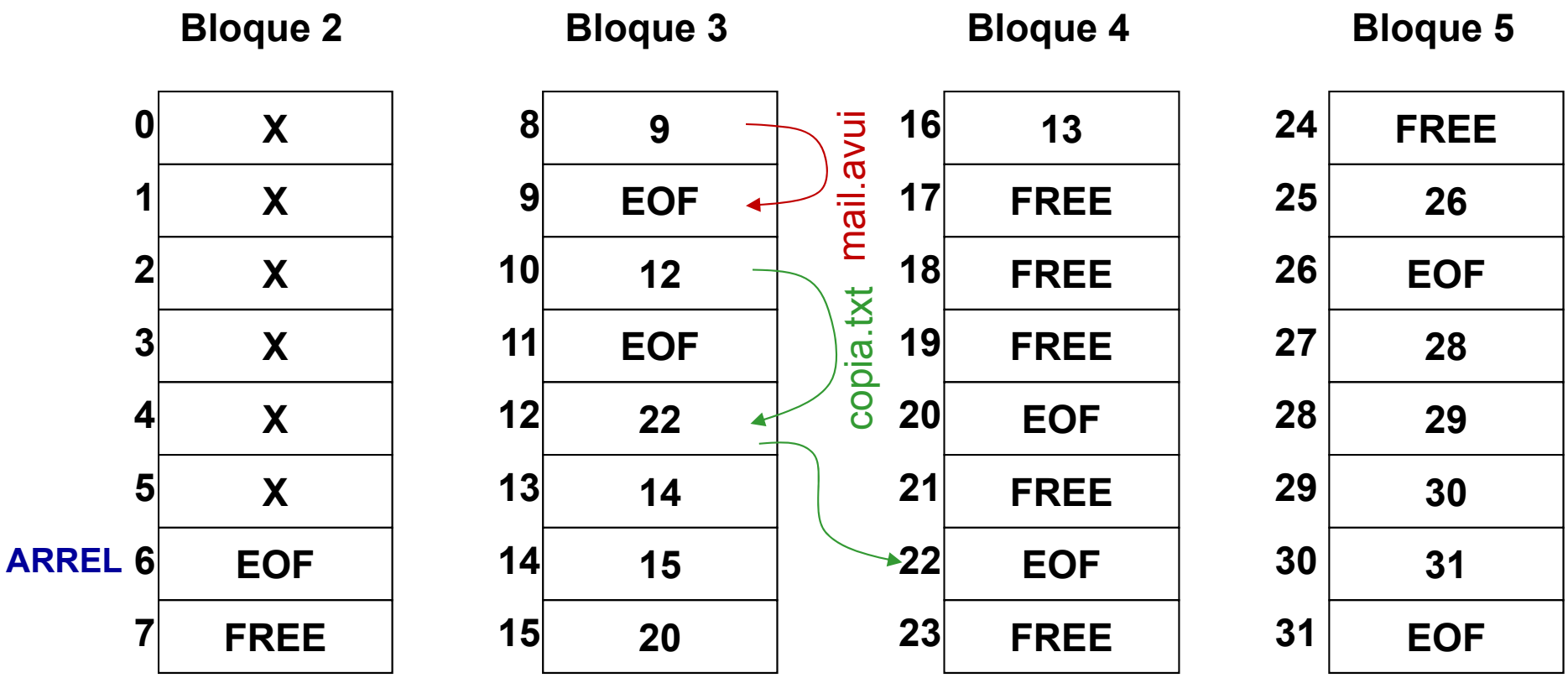
Sistema de ficheros FAT



Sistema de ficheros FAT



Sistema de ficheros FAT



Sistema de ficheros FAT

Bloque 6

Directorio ARREL

.	6
Otros campos	
. .	6
Otros campos	
PRACTIQUES	11
Otros campos	
mail.avui	8
Otros campos	
copia.txt	10
Otros campos	
...	
Basura	FREE
Otros campos	

ARREL: 6

PRACTIQUES: 11

mail.avui: 8 → 9

copia.txt: 10 → 12 → 22

Sistema de ficheros FAT

Bloque 11

Directorio PRACTIQUES

.	11
Otros campos	
. .	6
Otros campos	
pr1.c	16
Otros campos	
read.me	25
Otros campos	
pr1.o	27
Otros campos	
...	
Basura	FREE
Otros campos	

pr1.c: 16 → 13 → 14 → 15 → 20

read.me: 25 → 26

pr1.o: 27 → 28 → 29 → 30 → 31

Sistema de ficheros FAT

- Atendiendo al contenido de los bloques 6 y 11, ¿cuál es el primer bloque asignado al directorio raíz?
- Si en memoria central se encuentra el único bloque del directorio de trabajo (`PRACTIQUES`),
  - ¿qué acciones implica realizar la operación

    `open ("../copia.txt",O_WRONLY)`?

    - Localizar ".." en bloque del directorio `PRACTIQUES` y su primer bloque (6)
    - Traer bloque 6 a MC
    - Localizar `copia.txt` en bloque 6 y obtener sus atributos

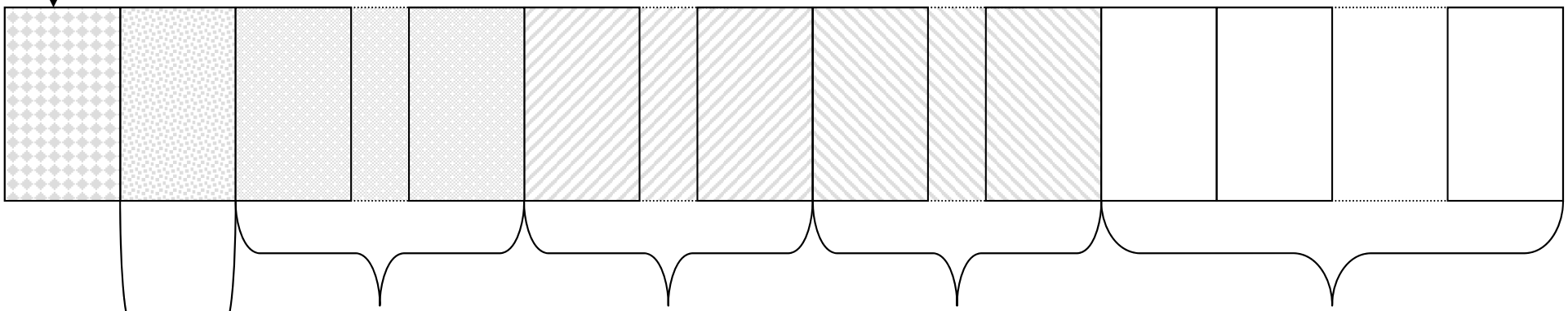
Sistema de ficheros FAT

- Si en memoria central se encuentra el único bloque del directorio de trabajo (`PRACTIQUES`),
  - ¿cuántos accesos a disco se realizarán como máximo para leer el fichero `../copia.txt`?
    - 1 acceso para leer bloque 6 de ..
    - 1 acceso para leer bloque 10 de `copia.txt`
    - 1 acceso para leer bloque 3 de la FAT
    - 1 acceso para leer bloque 12 de `copia.txt`
    - En bloque 3 de la FAT vemos que el bloque siguiente al 12 es el 22
    - 1 acceso para leer bloque 22 de `copia.txt`
    - 1 acceso para leer bloque 4 de la FAT

    Total: 6 accesos como máximo

`copia.txt`: 10 → 12 → 22

Sistema de ficheros FAT



Directorio para inodos

- **Especificación de una entrada de un directorio:**

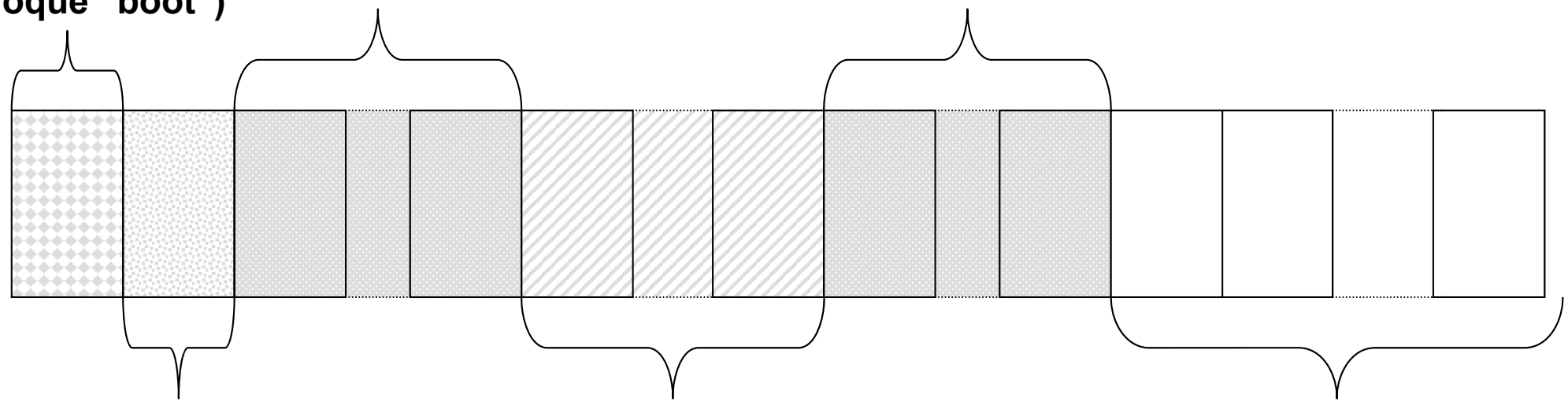
```
typedef struct {
        int             d_inodo;
        Nombre          d_nombre;
} T_Reg_Direct;

typedef char Nombre [28 ];
```

- **Especificación de un INODO:**

```
typedef struct {
    char      i_tipo;
    int       i_tamanyo;
    int       i_permiso;
    int       i_directos[IND_DIRECTOS];
    int       i_simple_ind[IND_ISIMPLE];
    int       i_doble_ind[IND_IDOBLE];
    T_Byte    i_relleno[NRELLENO];
    } T_Inodo;

typedef char Byte;
```

bytes de relleno para que T_Inodo sea una potencia exacta de 2

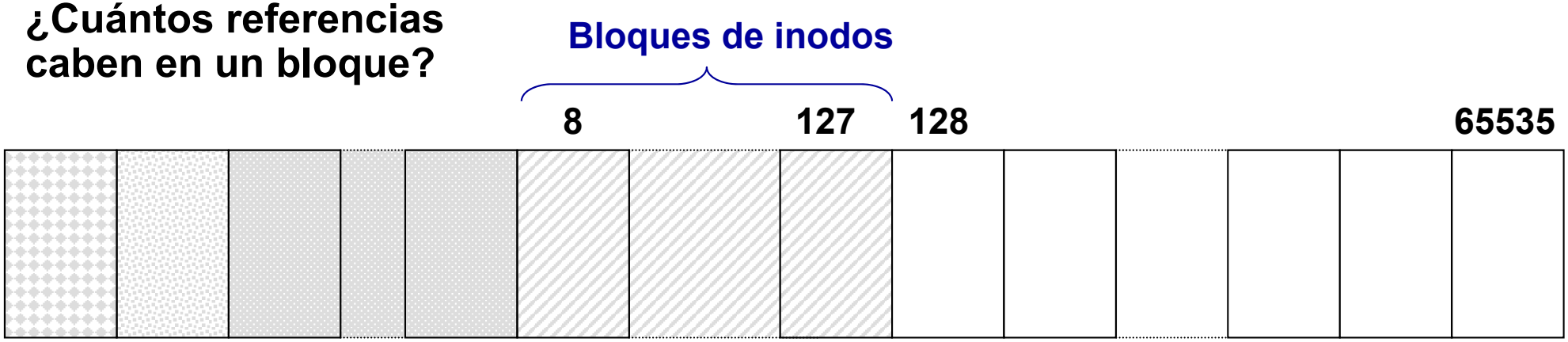
Sistema de ficheros con inodos

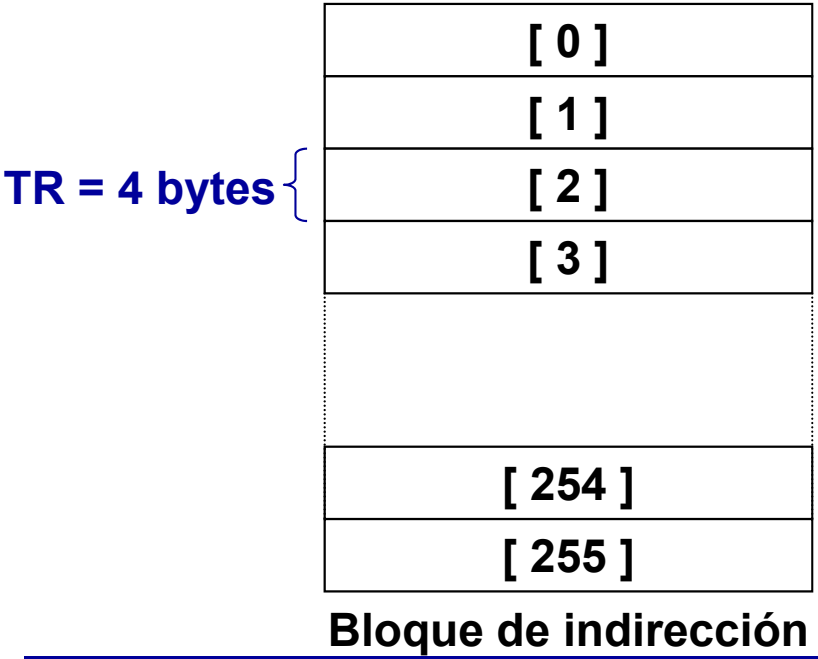


- *N° bloques del SF*
- *N° inodos del SF*
- *Tamaño bloque*
- *N° inodo raíz*
- *Primer bloque datos*
- *...*

Sistema de ficheros con inodos

- ¿Cuántos referencias caben en un bloque?



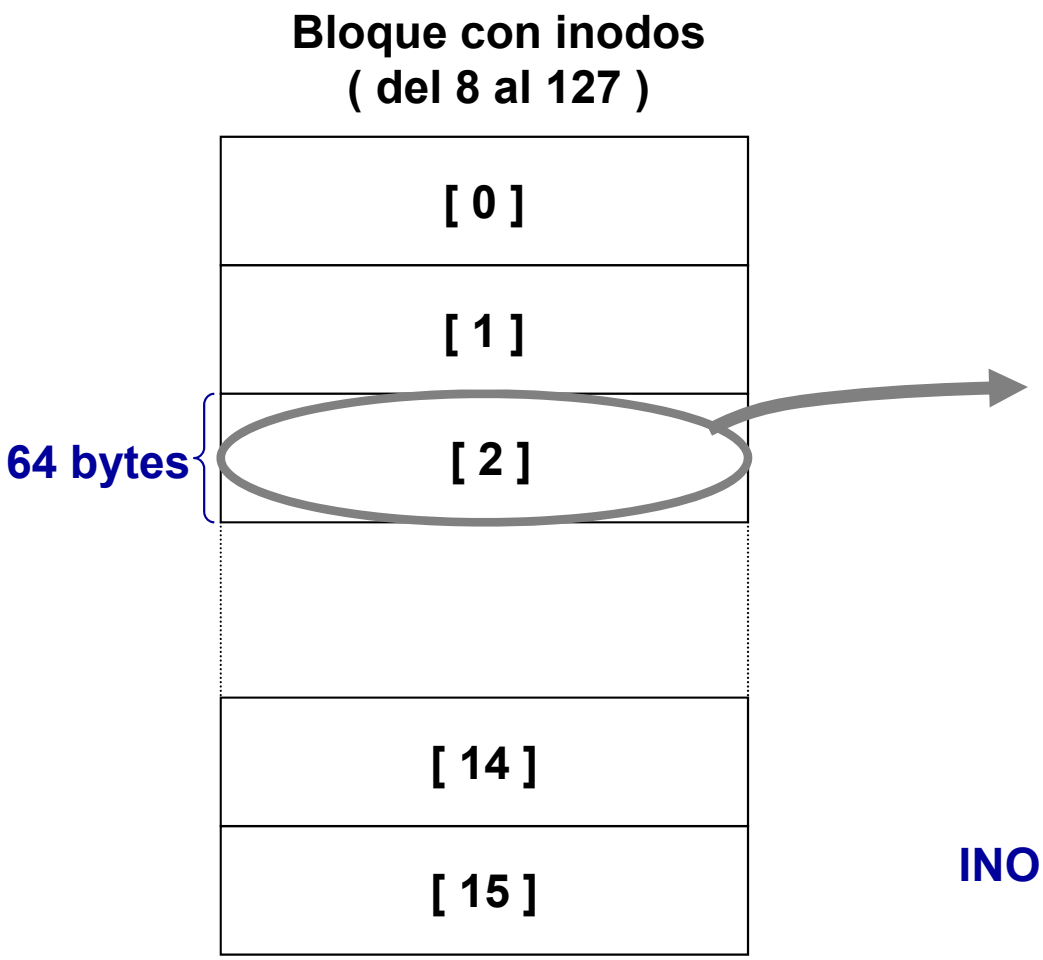


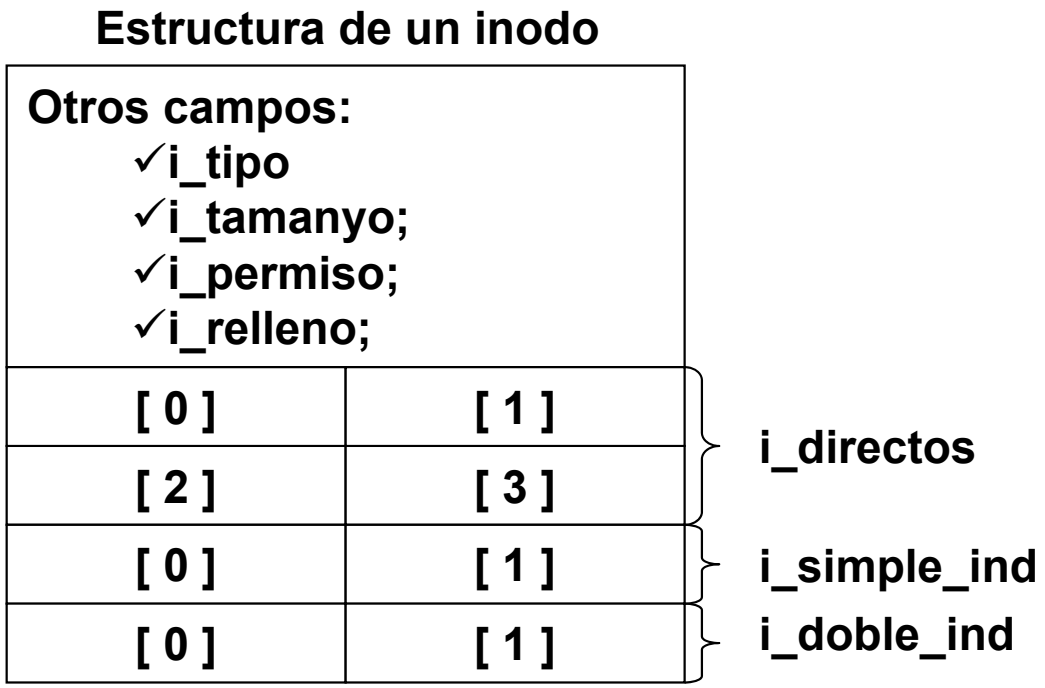
TAMANYO_BLOQUE=1024 bytes

$$\text{REFS\_POR\_BLOQUE} = \frac{2^{10}\ \frac{\text{by}}{\text{bloque}}}{2^{2}\ \frac{\text{by}}{\text{referencia}}} = 256\ \frac{\text{referencias}}{\text{bloque}}$$

Sistema de ficheros con inodos

- ¿Cuántos inodos caben en un bloque?





$$\text{INODOS\_POR\_BLOQUE} = \frac{2^{10}\ \frac{\text{by}}{\text{bloque}}}{2^{6}\ \frac{\text{by}}{\text{inodo}}} = 16\ \frac{\text{inodos}}{\text{bloque}}$$

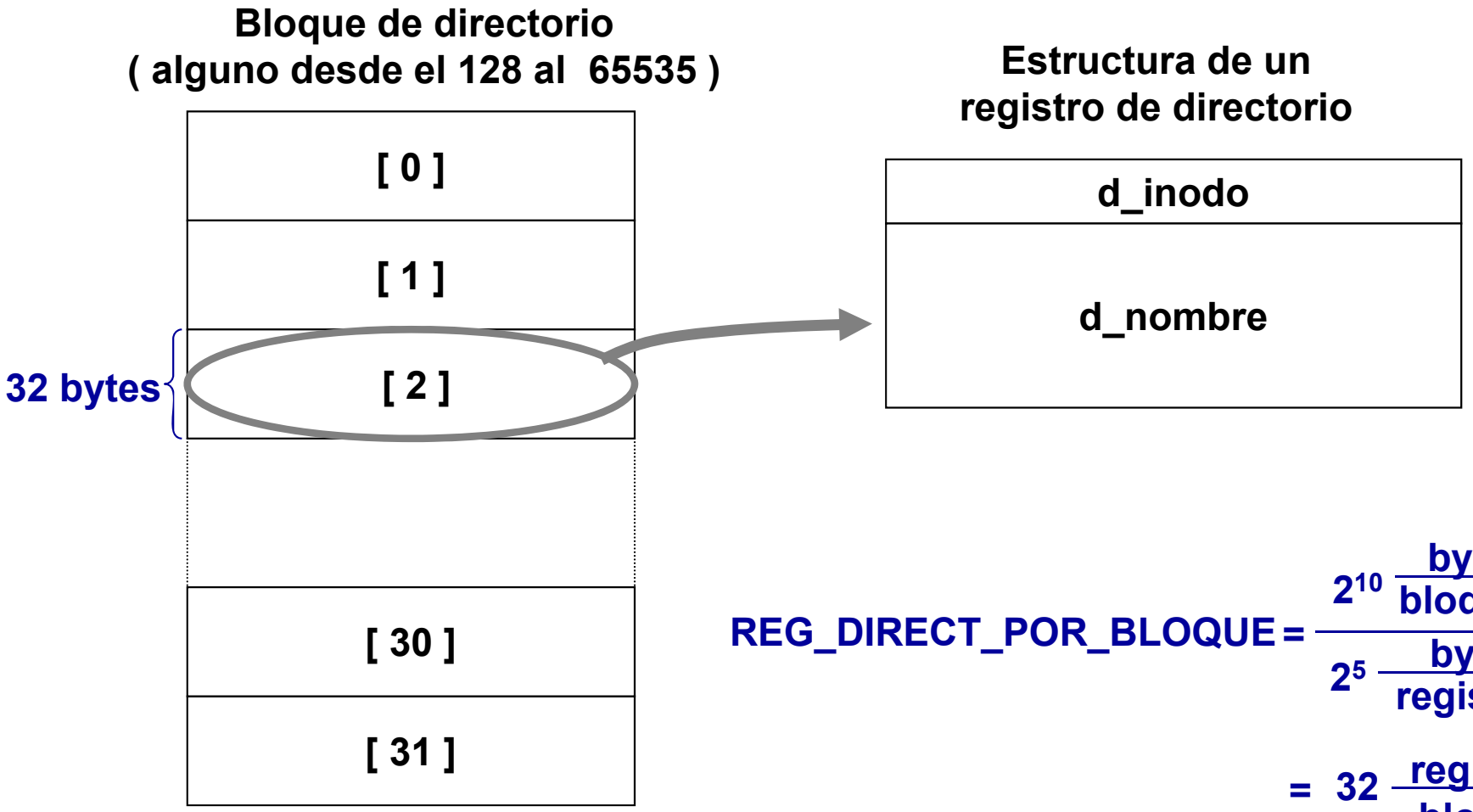
Sistema de ficheros con inodos

- **Localización de un inodo:**

  El inodo con número **X** estará en la posición **X MOD 16** del bloque **(X DIV 16) + 8**

Sistema de ficheros con inodos

- ¿Cuántos registros de dircetorio caben en un bloque?



$$REG\_DIRECT\_POR\_BLOQUE = \frac{2^{10}\ \frac{by}{bloque}}{2^{5}\ \frac{by}{registro}} = 32\ \frac{registros}{bloque}$$

Sistema de ficheros con inodos

- ¿Cómo localizar /home/iif/al20110/prb_vi.txt?

INODO nº 0 (/)

Otros campos: ✓i_tipo ✓i_tamanyo; ✓i_permiso; ✓i_relleno;		
128	NIL	i_directos
NIL	NIL	
NIL	NIL	i_simple_ind
NIL	NIL	i_doble_ind
NIL	NIL	

/ home / iif / al20110 / prb_vi.txt

Bloque 128

0	[0]
.	
0	[1]
. .	
4	[2]
bin	
6	[3]
dev	
14	[4]
lib	
20	[5]
home	
...	
FREE	[31]
basura	

Sistema de ficheros con inodos

INODO nº 20 (/home)

Otros campos: ✓i_tipo ✓i_tamanyo; ✓i_permiso; ✓i_relleno;	
200	201
230	NIL
NIL	NIL
NIL	NIL

i_directos

i_simple_ind

i_doble_ind

/ home / iif / al20110 / prb_vi.txt

Bloque 200

20	[0]
.	
0	[1]
. .	
FREE	[2]
iif.back	
70	[3]
itif	
FREE	[4]
prb	
101	[5]
edu	
...	
115	[31]
iif	

Sistema de ficheros con inodos

INODO nº 115 (.../iif)

Otros campos: ✓i_tipo ✓i_tamanyo; ✓i_permiso; ✓i_relleno;	
800	801
830	831
832	NIL
NIL	NIL

... / iif / al20110 / ...

Bloque 832

840	[0]
841	[1]
1001	[2]
850	[3]
890	[4]
891	[5]
1023	[6]
3021	[7]
911	[8]
950	[9]
925	[10]
2100	[11]
...	
12067	[254]
842	[255]

Bloque 891

200	[0]
al11111	
201	[1]
al11112	
FREE	[2]
al11114	
210	[3]
al20110	
FREE	[4]
al02010	
126	[5]
al30111	
...	
209	[31]
al32011	

Sistema de ficheros con inodos

INODO nº 210 (.../al20110)

Otros campos:
- ✓i_tipo
- ✓i_tamanyo;
- ✓i_permiso;
- ✓i_relleno;

43333	43334	i_directos
43335	25060	
NIL	NIL	i_simple_ind
NIL	NIL	i_doble_ind

/ home / iif / al20110 / prb_vi.txt

Bloque 43333

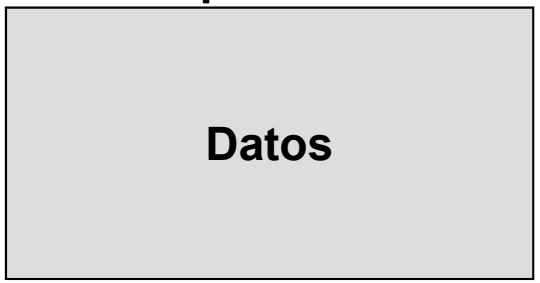
210	[0]
.	
115	[1]
. .	
FREE	[2]
foto.jpg	
300	[3]
.profile	
301	[4]
.kshrc	
302	[5]
prb_vi.txt	
...	
FREE	[31]
nsmail	

Sistema de ficheros con inodos

INODO nº 302 (.../prb_vi.txt)

Otros campos: ✓i_tipo ✓i_tamanyo; ✓i_permiso; ✓i_relleno;		
64000	64001	i_directos
NIL	NIL	
NIL	NIL	i_simple_ind
NIL	NIL	i_doble_ind

Bloque 64000



Bloque 64001

/ home / iif / al20110 / prb_vi.txt

Ejercicios

- **Ejercicio 1:**

  ¿Cuál es el tamaño mínimo de bloque con el que se podría trabajar en un dispositivo de 8 GB si cada bloque se identifica por un entero sin signo de 2 bytes?

Solución:

TR = 2 bytes = 16 bits $\rightarrow 2^{16}$ posibles direcciones $\rightarrow 2^{16}$ posibles bloques

8 GB = 2^{33} bytes

$$\text{TAMANYO\_BLOQUE} = \frac{2^{33}\ \frac{\text{by}}{\text{dispositivo}}}{2^{16}\ \frac{\text{bloques}}{\text{dispositivo}}} = 2^{17}\ \frac{\text{by}}{\text{bloque}} = 128\ \text{K}\ \frac{\text{by}}{\text{bloque}}$$

Ejercicios

- **Ejercicio 2:**

  ¿Se podría usar un mapa de bits como método de identificación de los bloques asignados a un fichero?

Ejercicios

- **Ejercicio 3:**

  En un sistema de ficheros en el que las entradas de directorio tienen todas un tamaño fijo de 32 bytes y los bloques tienen un tamaño de 2 KB,

  ¿cuál es el número máximo de entradas libres que podría haber en un directorio que ocupase únicamente un bloque?

Ejercicios

- **Ejercicio 4:**

  ¿Cuántos ficheros de datos de usuario se podrán tener almacenados como máximo en un dispositivo de almacenamiento virtual donde se mantiene un sistema de ficheros basado en inodos con las siguientes características:

  - Tamaño de bloque: 4 KB
  - Tamaño de inodo: 128 bytes
  - Cada dirección de bloque se especifica mediante 4 bytes
  - Número de bloques reservados para inodos: 64

Ejercicios

- **Ejercicio 4 (solución):**

$|\text{bloque}| = 2^{12}\ \frac{\text{by}}{\text{bloque}}$ $\quad |\text{inodo}| = 2^{7}\ \frac{\text{by}}{\text{inodo}}$ $\quad$ TR = 4 by $\quad 2^{6}$ bloques de inodos

Número máximo de ficheros ← Número máximo de inodos

¿Número de inodos por bloque?

$$\frac{2^{12}\ \frac{\text{by}}{\text{bloque}}}{2^{7}\ \frac{\text{by}}{\text{inodo}}} = 2^{5}\ \frac{\text{inodos}}{\text{bloque}}$$

¿ Número máximo de inodos?

$$2^{6}\ \text{bloques} \times 2^{5}\ \frac{\text{inodos}}{\text{bloque}} = 2\ \text{K inodos} \rightarrow 2048\ \text{ficheros}$$

¿ Número máximo de ficheros de usuario?

2048 – (1) = 2047 ficheros → ***1 fichero para el directorio raíz***

Ejercicios

- **Ejercicio 5:**

  En un dispositivo virtual de almacenamiento masivo dividido en bloques de 1KB y donde cada número de bloque se representa mediante 4 bytes, se implementa un sistema de ficheros basado en una organización indexada.

  Cada inodo tiene 3 referencias directas, 2 simples indirectas y 2 dobles indirectas.

  ¿Cuál de los siguientes tamaños de fichero generará mayor desperdicio en el dispositivo de almacenamiento virtual?

  a) 1027 * $\mathbf{2^{10}}$ bytes

  b) 253 * $\mathbf{2^{10}}$ bytes

  c) 258 * $\mathbf{2^{10}}$ bytes

  d) 2050 bytes

Ejercicios

- **Ejercicio 5 (solución):**

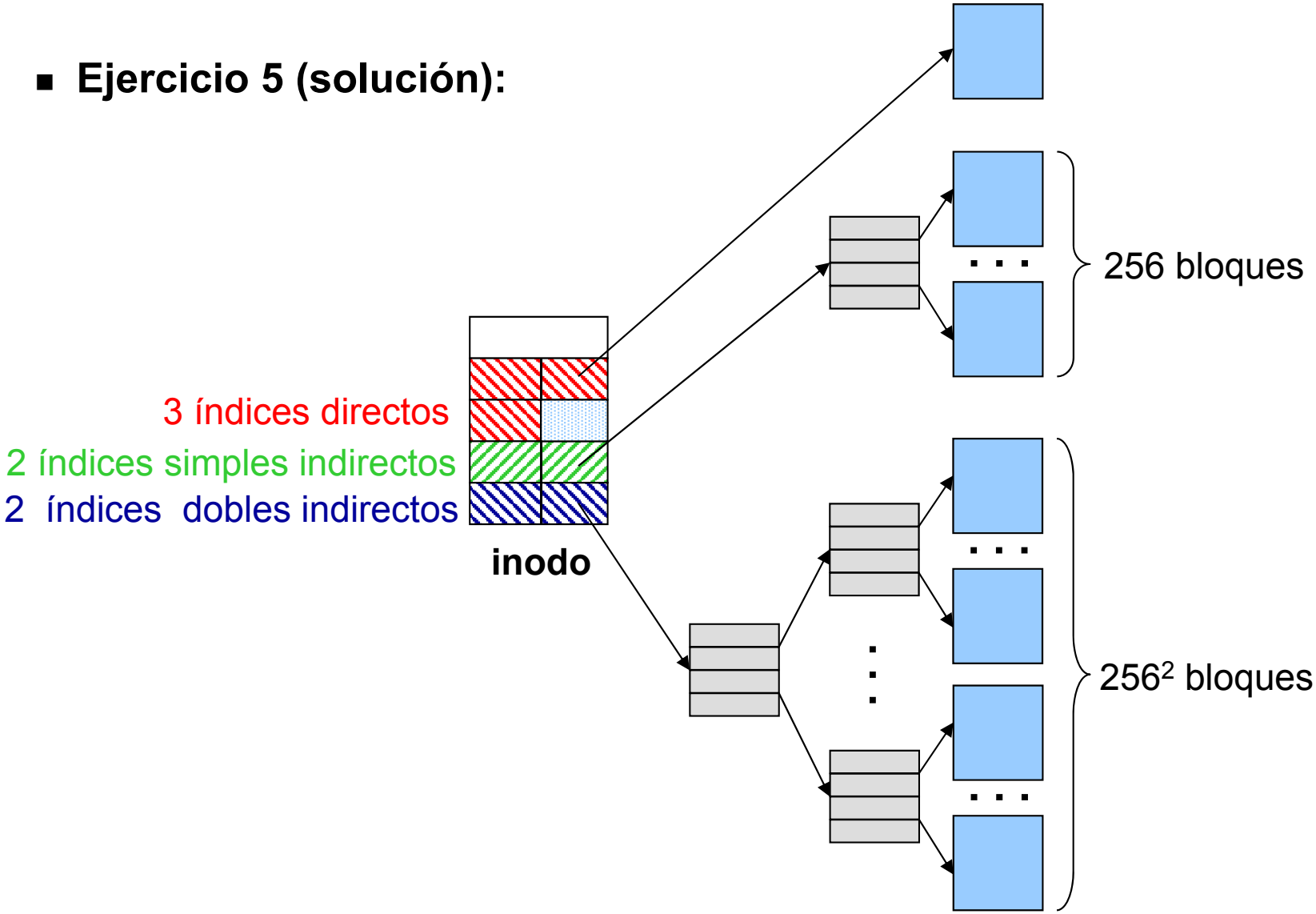
$$|\text{bloque}| = 1\ \text{KB} = 2^{10}\ \frac{\text{by}}{\text{bloque}} \qquad \text{TR} = 4\ \text{by}$$

¿Referencias por bloque?

$$\frac{2^{10}\ \frac{\text{by}}{\text{bloque}}}{2^{2}\ \frac{\text{by}}{\text{dirección}}} = 2^{8}\ \frac{\text{direcciones}}{\text{bloque}} = 256\ \text{referencias\_por\_bloque}$$

Ejercicios

- **Ejercicio 5 (solución):**



Ejercicios

- **Ejercicio 5 (solución):**

  Capacidad de direccionamiento con los diferentes tipos de índices:

  - Con 3 índices directos (ID) hasta 3 bloques de datos
  - Con 1 índice indirecto simple (IIS) hasta 259 bloques de datos (3 + 256)
  - Con 2 IIS hasta 515 bloques de datos (3 + (2 x 256))
  - Con 1 índice indirecto doble (IID) hasta 66.051 bloques de datos
    (3 + (2 x 256) + 256^2)

  Los apartados a), b) y c) sólo desperdician espacio en bloques de indirección.
  El apartado d) sólo desperdicia espacio en bloque de datos.

  a) 1027 * $\mathbf{2^{10}}$ bytes → Ocupa 1027 bloques completos de datos

  b) 253 * $\mathbf{2^{10}}$ bytes → Ocupa 253 bloques completos de datos

  c) 258 * $\mathbf{2^{10}}$ bytes → Ocupa 258 bloques completos de datos

  d) 2050 bytes → Ocupa 2 bloques completos de datos y 1 incompleto

Ejercicios

- **Ejercicio 5 (solución):**

  Espacio desaprovechado:

  a) 1027 bloques → 3 ID + 2 IIS + 1 IID

  3 ID → 3 bloques

  2 IIS → 2 x 256 bloques

  1 IID → 1027 – (2 x 256 + 3)=512 bloques → 2 entradas del bloque de indirección doble

  Desperdicio:

  - En bloque del primer nivel del IID: 1024 – 4 = **1020 bytes**
  - En bloque del segundo nivel del IID: 1024 – (2 x 4) = **1016 bytes**

  Total: 2036 by

  b) 253 bloques → 3 ID + 1 IIS

  3 ID → 3 bloques

  1 IIS → 250 entradas del bloque de indirección simple

  Desperdicio: 1024 – (250 x 4) = **24 bytes**

Ejercicios

- **Ejercicio 5 (solución):**

  Espacio desaprovechado:

  c) 258 bloques → 3 ID + 1 IIS
  ↓ ↓
  3 bloques 255 bloques
  ↓
  255 entradas del bloque de indirección simple

  Desperdicio: 1024 – (255 x 4) = **4 bytes**

  d) 2050 bytes → 3 ID
  ↓
  3 bloques

  Desperdicio: (3 x 1024) – 2050 = **1022 bytes** → *Fragmentación interna*

Ejercicios

- **Ejercicio 6:**

  Un dispositivo virtual de almacenamiento masivo de 16 MB se divide en bloques de 4 KB.

  Sobre dicho dispositivo se implementa un sistema de ficheros basado en FAT en el que cada bloque se identifica mediante un número especificado con 4 bytes.

  ¿Cuántos bloques del dispositivo serán necesarios para mantener la FAT?

  ¿Cuál es el tamaño de la FAT en bytes?

  ¿Cuál sería el tamaño de la FAT si el dispositivo ocupase 16 GB?

Ejercicios

- **Ejercicio 6 (solución):**

$|\text{disco}| = 16\ \text{MB} = 2^{24}\ \frac{\text{by}}{\text{disco}}$ $\quad |\text{bloque}| = 4\ \text{KB} = 2^{12}\ \frac{\text{by}}{\text{bloque}}$ $\quad \text{TR} = 4\ \text{by}$

¿Número de bloques del disco?

$$\frac{2^{24}\ \frac{\text{by}}{\text{disco}}}{2^{12}\ \frac{\text{by}}{\text{bloque}}} = 2^{12}\ \frac{\text{bloques}}{\text{disco}} \rightarrow 2^{12}\ \text{posibles direcciones de bloque}$$

Entradas de la FAT

¿Referencias por bloque?

$$\frac{2^{12}\ \frac{\text{by}}{\text{bloque}}}{2^{2}\ \frac{\text{by}}{\text{dirección}}} = 2^{10}\ \frac{\text{direcciones}}{\text{bloque}}$$

Ejercicios

- **Ejercicio 6 (cont.):**

¿Número de bloques de la FAT?

Entradas de la FAT

$$\frac{2^{12}\ \frac{\text{direcciones}}{\text{disco}}}{2^{10}\ \frac{\text{direcciones}}{\text{bloque}}} = 2^{2}\ \frac{\text{bloques FAT}}{\text{disco}}$$

¿Tamaño de la FAT?

$$2^{2}\ \frac{\text{bloques FAT}}{\text{disco}} \times 2^{12}\ \frac{\text{by}}{\text{bloque}} = 16\ \text{KB}$$

Si |disco|=16 GB → 4 K bloques de FAT → 16 MB de FAT

Ejercicios

- **Ejercicio 7:**

  Sea un sistema de ficheros que utiliza asignación indexada y en el que:

  - Cada inodo tiene 3 índices directos, 2 indirectos simples y 1 indirecto doble
  - Cada bloque se identifica con un número natural representado mediante 32 bits
  - El tamaño de bloque es de 4 KB

  ¿Cuánto espacio de disco (sin contar el del propio inodo) se necesitará para almacenar un fichero de tamaño 12 MB?

Ejercicios

- **Ejercicio 7 (solución):**

TR = 32 bits = 4 by |bloque| = 2^{12} by |fichero| = 3 x 2^{22} by

¿Número de bloques del fichero?

$$\frac{3 \times 2^{22} \text{by}}{2^{12} \frac{\text{by}}{\text{bloque}}} = 3 \times 2^{10} \text{ bloques} = 3072 \text{ bloques}$$

¿Referencias por bloque?

$$\frac{2^{12} \frac{\text{by}}{\text{bloque}}}{2^{2} \frac{\text{by}}{\text{dirección}}} = 2^{10} \frac{\text{direcciones}}{\text{bloque}} = 1 \text{ K} \frac{\text{direcciones}}{\text{bloque}}$$

Ejercicios

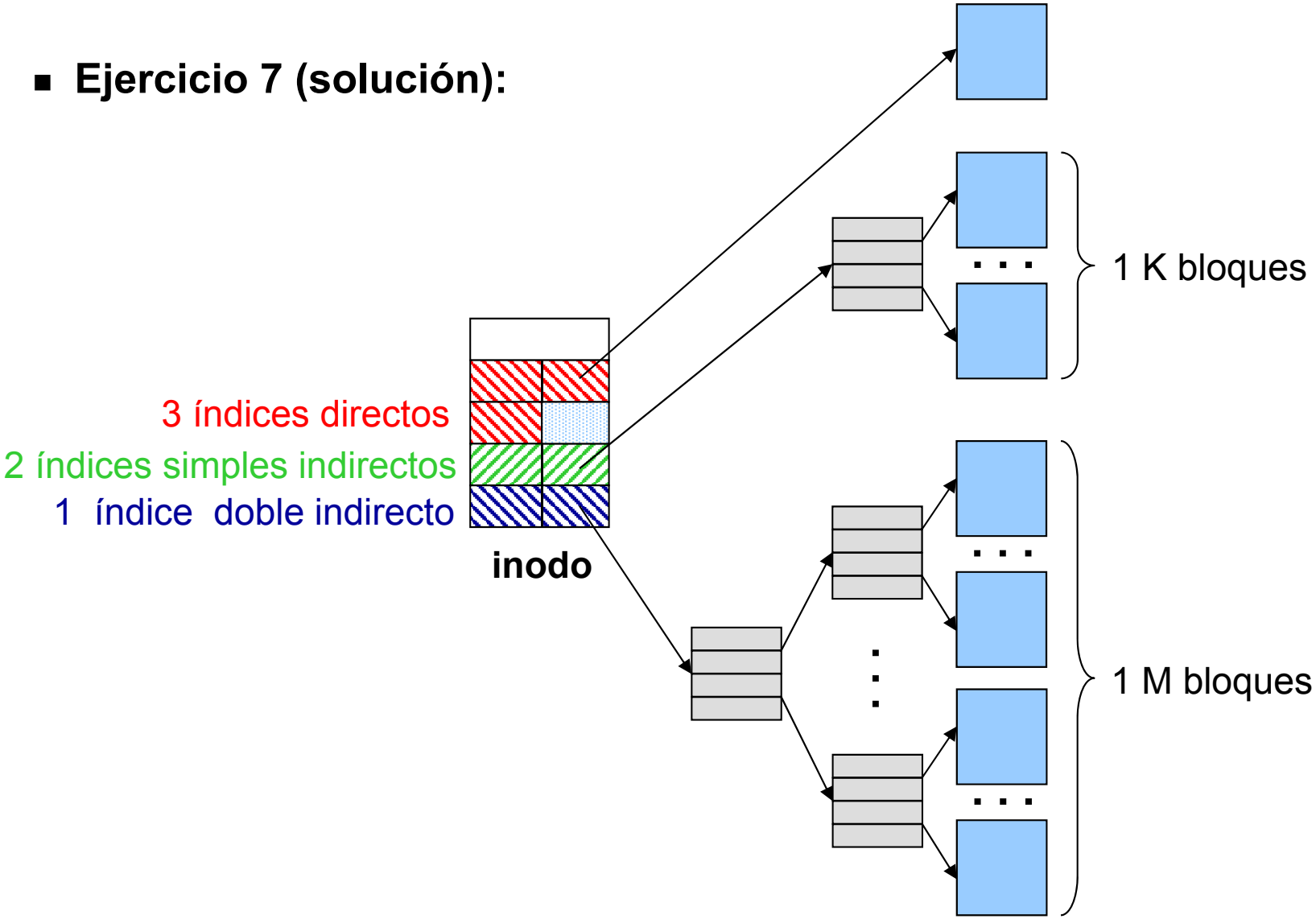
- **Ejercicio 7 (solución):**

  Capacidad de direccionamiento con los diferentes tipos de índices:

  - Con 3 índices directos (ID) hasta 3 bloques de datos
  - Con 2 índices indirectos simple (IIS) hasta 2051 bloques de datos
  - Con 1 índice indirecto doble (IID) hasta 1 M + 2051 bloques de datos

Ejercicios

- **Ejercicio 7 (solución):**



Ejercicios

- **Ejercicio 7 (solución):**

Espacio de datos del usuario ocupado por el fichero:

- Con 3 ID: 3 bloques
- Con 2 IIS: 2 K = 2048 bloques
- Con 1 IID: 3072 – 3 – 2048 = 1021 bloques

Se usa sólo la primera entrada del bloque de IID

Espacio de datos del SO ocupado por el fichero:

- Con 2 IIS: 2 bloques
- Con 1 IID: 2 bloques

Espacio total ocupado por el fichero:

- 3072 bloques de datos del usuario + 4 bloques de indirección del SO = 3076 bloques

Ejercicios

- **Ejercicio 8:**

  En un dispositivo de almacenamiento masivo, dividido en bloques de 1 KB, se implementa un sistema de ficheros basado en una organización enlazada.

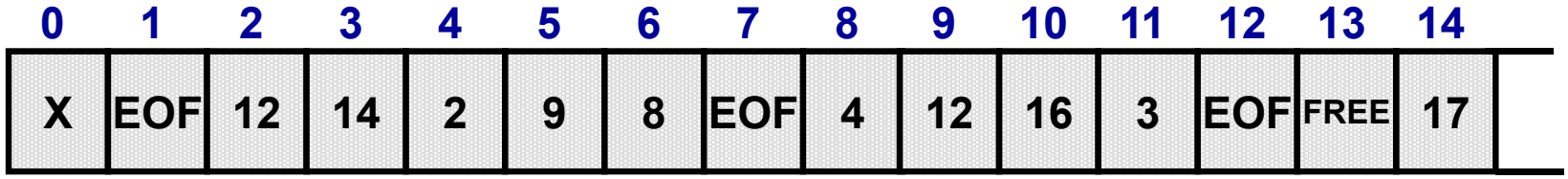
  Si en dicho dispositivo, después de restar el total de bloques utilizado por el SO para sus datos (superbloque, mapas de bits y FAT) quedan libres 64 Kbloques,

  ¿cuál será el máximo tamaño que puede alcanzar un fichero de datos de un usuario en este sistema de ficheros?

Ejercicios

- **Ejercicio 9:**

  Sea la siguiente tabla de asignación de bloques a ficheros:



  Obténgase el intervalo (en bytes) en que se encontrará el tamaño de un fichero cuyo primer bloque es el 6, suponiendo que:

  - El tamaño del sistema de ficheros es de 512 MB
  - Un número de bloque lógico se expresa mediante 16 bits
  - El dispositivo se ha dividido en el máximo número posible de bloques

Ejercicios

- **Ejercicio 9 (solución):**

  $|\text{disco}| = 2^{29}$ by TR = 2 by $\rightarrow 2^{16}$ posibles bloques

  ¿Número bloques del fichero?

  $6 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 12$ (5 bloques)

  ¿Tamaño bloque?

  $$\frac{2^{29}\ \frac{\text{by}}{\text{disco}}}{2^{16}\ \frac{\text{bloques}}{\text{disco}}} = 8\ \text{K}\ \frac{\text{by}}{\text{bloque}}$$

  ¿Rango tamaño fichero?

  [(4 x 8 K) + 1 .. 5 x 8 K] bytes

Ejercicios

- **Ejercicio 10:**

  Contenido de un directorio de un SF basado en FAT:

Nombre (22)	Tipo (1)	Tamaño (4)	Primer Bloque (4)	Permisos (1)
EJEMPLO1.C	F	12.688	6080	LE-
ESTRUCTURA.H	F	406	2100	LE-
ALMACEN.C	F	74.105	0300	L--
EJECUTABLES	D	8.192	2102	LEX
SALIDA	F	204	2200	LE-
ALMACEN.X	F	4.106.404	2208	LEX

Tamaño en bytes del campo

F: fichero
D: directorio

L: lectura
E: escritura
X: ejecución

Ejercicios

- **Ejercicio 10 (cont.):**

  Disposición de los índices en la FAT para el fichero EJEMPLO1.C:

  $$6080 \rightarrow 6140 \rightarrow 6141 \rightarrow 6142$$

  Un directorio ocupa un número entero de bloques.

  Obtener:

  - Tamaño de la FAT si la capacidad del disco es 40 MB y el número de bloque ocupa 4 bytes.
  - Número máximo y mínimo de entradas en el directorio EJECUTABLES asumiendo que las entradas libres están al final del último bloque.

Ejercicios

- **Ejercicio 10 (solución):**

  Obtener (cont.):

  Asumir también que toda la FAT está en MC

  - Número máximo y mínimo de accesos a disco para ejecutar ALMACEN.X asumiendo que:
    - Se tiene en memoria la entrada que define el directorio que se muestra en la tabla.
    - Se debe cargar en memoria todo el fichero que contiene el código.
  - Pregunta anterior con una estructura tipo inodo con 512 índices directos, 64 indirectos simples y 32 indirectos dobles.
  - Tamaño máximo de un fichero con la anterior estructura de inodos.

Ejercicios

- **Ejercicio 10 (cont.):**

  Obtener tamaño de la FAT si la capacidad del disco es 40 MB y el número de bloque ocupa 4 bytes.

  Tamaño_FAT = n_bloques * n_bytes_de_un_numero_de_bloque

  n_bloques = Capacidad_disco / Tamaño_bloque

  ¿Tamaño_bloque (TB)?

  Según tabla:

  - EJEMPLO1.C ocupa 12688 bytes y 4 bloques.

    3 x TB < 12688

    4 x TB ≥ 12688

  - EJECUTABLES ocupa 8192 bytes

    Un directorio ocupa un número entero de bloques:

    8192 MOD TB = 0

  Luego TB=4 KB.

Ejercicios

- **Ejercicio 10 (cont.):**

  Obtener tamaño de la FAT si la capacidad del disco es 40 MB y el número de bloque ocupa 4 bytes (cont.).

  n_bloques = Capacidad_disco / Tamaño_bloque =

  = (40 * 2^{20} by/disco) / (4 * 2^{10} by/bloque) = 10 K bloques/disco

  Tamaño_FAT = n_bloques * n_bytes_de_un_numero_de_bloque =

  = 10 K bloques * (4 by/bloque) = 40 KB = 10 bloques

Ejercicios

- **Ejercicio 10 (cont.):**

  Obtener número máximo y mínimo de entradas en el directorio EJECUTABLES asumiendo que las entradas libres están al final del último bloque.

  Tamaño_reg_tipo_dir = 22 + 1 +4 +4 +1 = 32 by

  Regs_tipo_dir_por_bloque = (4 * 2^{10} by/bloque) / (2^5 by/reg) = 128 regs/bloque

  El directorio EJECUTABLES ocupa 2 bloques completos. Por tanto:

  - Número mínimo entradas en el directorio = 128 + 1 = 129
  - Número máximo entradas en el directorio = 128 * 2 = 256

Ejercicios

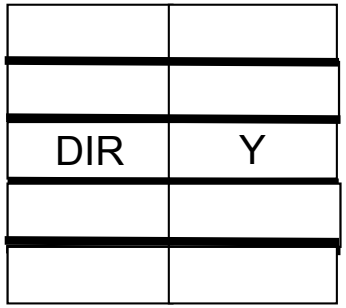
- **Ejercicio 10 (cont.):**

  Obtener número mínimo de accesos a disco para ejecutar ALMACEN.X.

  Si denotamos por DIR al directorio cuyo contenido se muestra en el enunciado, en memoria se tiene:

  El directorio DIR cabe en un bloque (6 * 32 by < 4 Kb y).

  Bloque X

  Lectura de todo el fichero ALMACEN.X:

  - Lectura del bloque Y asociado al directorio DIR que contiene al fichero.
  - Lectura de los $\lceil 4106404 / (4 * 210) \rceil = 1003$ bloques del fichero.
  - Se asume que toda la FAT está en memoria.

  Total = 1004 accesos como mínimo.

Ejercicios

- **Ejercicio 10 (cont.):**

  Obtener número máximo de accesos a disco para ejecutar ALMACEN.X.

  Si no cabe toda la FAT en memoria, a lo sumo se hacen:

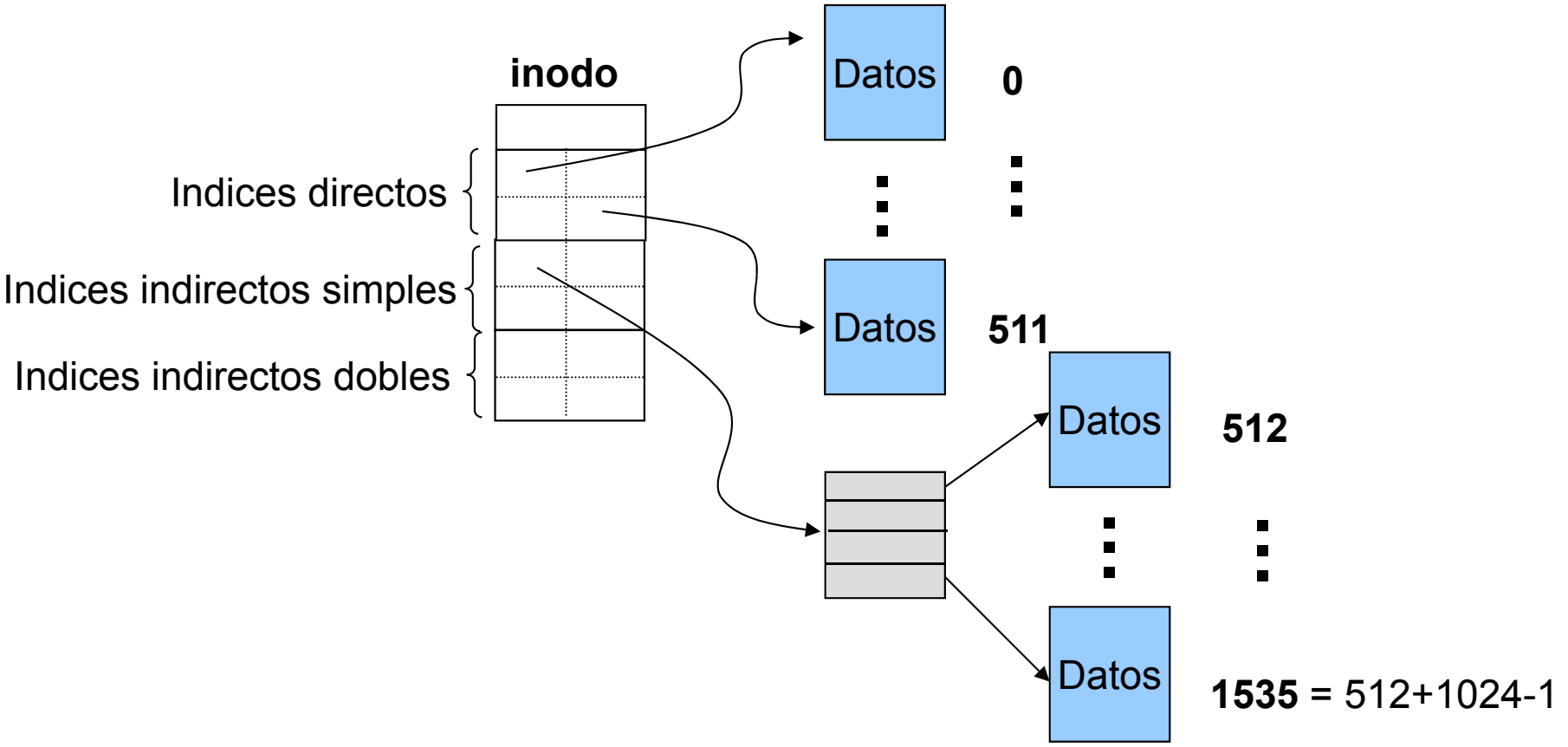
  - Lectura del bloque Y asociado al directorio DIR que contiene al fichero.
  - Lectura de los $\lceil 4106404 / (4 * 210) \rceil = 1003$ bloques del fichero.
  - 1003 accesos a bloques de la FAT para obtener la secuencia de números de bloques asignados al fichero.

  Total = 2008 accesos como máximo.

Ejercicios

- **Ejercicio 10 (cont.):**

  Pregunta anterior con una estructura tipo inodo con 512 índices directos, 64 indirectos simples y 32 indirectos dobles.



Ejercicios

- **Ejercicio 10 (cont.):**

  Pregunta anterior con una estructura tipo inodo con 512 índices directos, 64 indirectos simples y 32 indirectos dobles (cont.).

  Referencias_por_bloque = (4 * 2^{10} by/bloque) / (4 by/direcc_bloque) =

  = 2^{10} direcc_bloque /bloque.

  Lectura de todo el fichero ALMACEN.X:

  - Lectura del inodo del directorio.
  - Lectura del bloque del directorio.
  - Lectura del bloque con el inodo del fichero.
  - Lectura de los 1003 bloques de datos del fichero.
  - Lectura de un bloque de indirecciones.

  Total = 1 + 1 + 1 + 1003 + 1 = 1007 accesos.

Ejercicios

- **Ejercicio 10 (cont.):**

  Tamaño máximo de un fichero con la anterior estructura de inodos.

  Capacidad direccionamiento con un inodo:

  - Indices directos: 512 bloques.
  - Indices indirectos simples: $64 * 2^{10}$ bloques.
  - Indices indirectos dobles: $32 * (2^{10})^2$ bloques.

  Total = $[512 + 64 * 2^{10} + 32 * (2^{10})^2]$ bloques x $(4 * 2^{10}$ by/bloque) = 128 GB.