# Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
*Universitat Politècnica de València*

## Bloque Temático 3: Gestión de Archivos
## Unidad Temática 7

# Implementación de Archivos o Ficheros ("Implementing file systems")

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

- **Objetivos**
  - Introducir el concepto de **archivo** como **una abstracción** de la memoria secundaria
  - Explicar la **función** de los sistemas de archivos
  - Describir los niveles de abstracción del sistema de archivos
  - Analizar las **técnicas de asignación** de bloques de disco a archivos

- **Contenido**
  - **Arquitectura del sistema de archivos**
  - Concepto de archivo
  - Asignación de bloques a archivos

- **Bibliografía**
  - A. Silberschatz, P.B. Galvin: “Fundamentos de Sistemas Operativos”, McGraw-Hill, 7ª ed. 2006 (Capítulos 10 y 11)

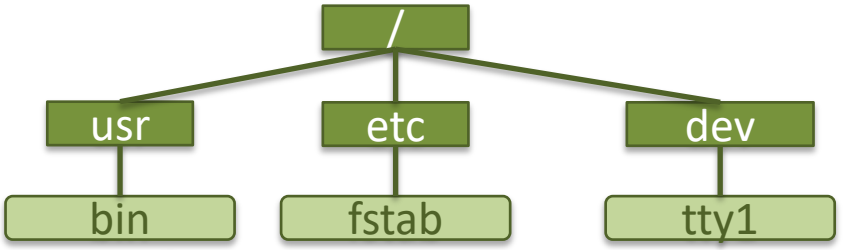
# Arquitectura del sistema de archivos

- La estructura del sistema de archivos tiene tres aspectos:
    1. La forma en que están **organizados los archivos** desde el punto de vista del usuario
    2. La forma en que se **almacenan los archivos** en el sistema de almacenamiento secundario (normalmente el disco)
    3. La forma en que se **manipulan los archivos**. Es decir, como realizar lecturas, escrituras, etc.

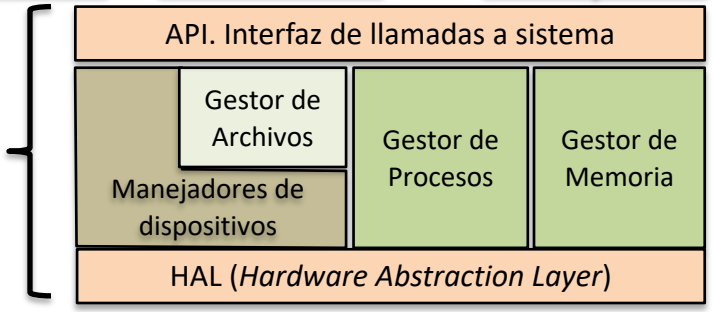
# Arquitectura del sistema de archivos

**Archivo**
Abstracción proporcionada por el S.O.

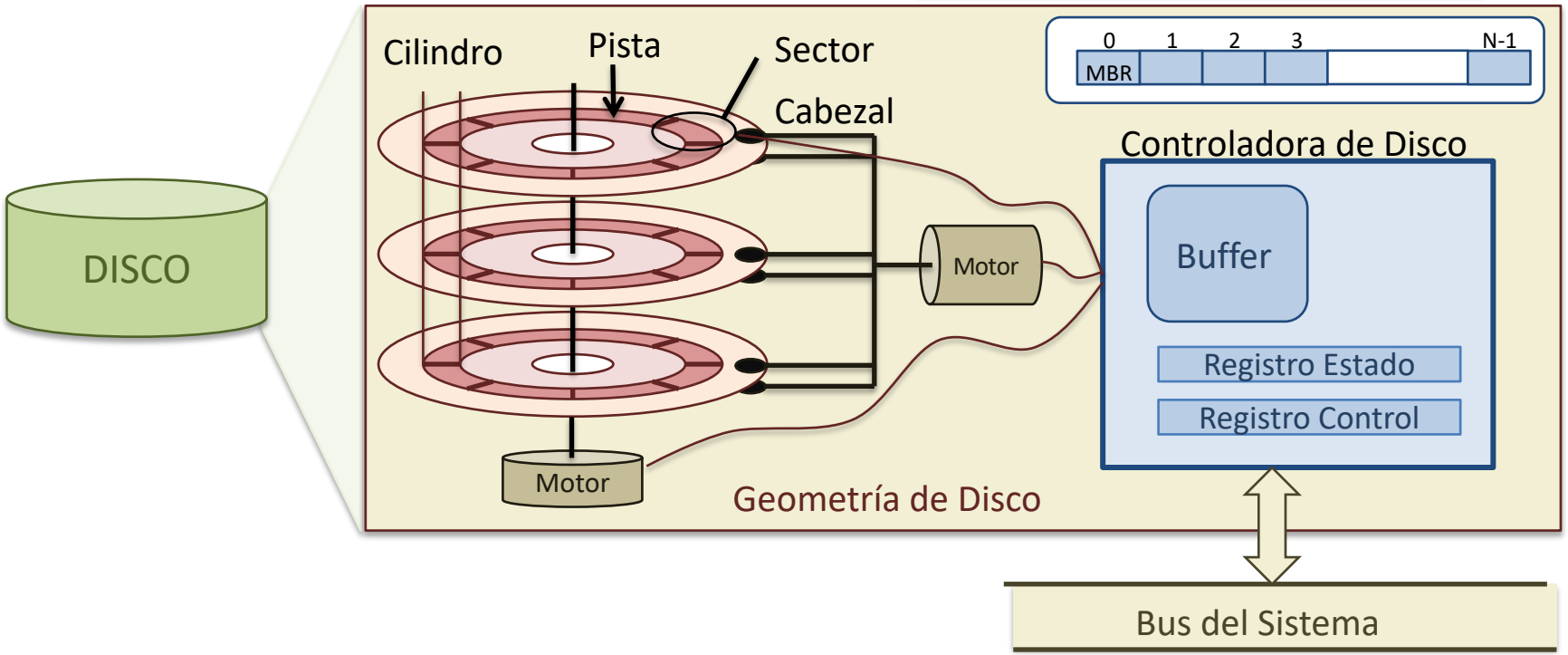
El S.O. **abstrae las propiedades físicas del sistema de almacenamiento** y proporciona **una visión uniforme de los dispositivos de almacenamiento**




Nivel Usuario




S.O.




Nivel Físico

# Arquitectura del sistema de archivos

Sistema de archivos proporciona mecanismos para:

- **Almacenamiento persistente** de la información
  - la información permanece aunque el sistema informático no este en marcha, el dispositivo más habitual es el **disco**
- **Acceder a la información**
  - Ofrece una **interfaz al usuario** compuesta básicamente por dos tipos de elementos:
    - **Archivo**: unidad lógica de almacenamiento persistente
    - **Estructura de directorios:** mecanismo para organizar todos los archivos

- Consideraciones importantes
  - Mantiene datos críticos para el sistema
  - Condiciona las prestaciones globales
  - Es el aspecto más visible y utilizado de un SO

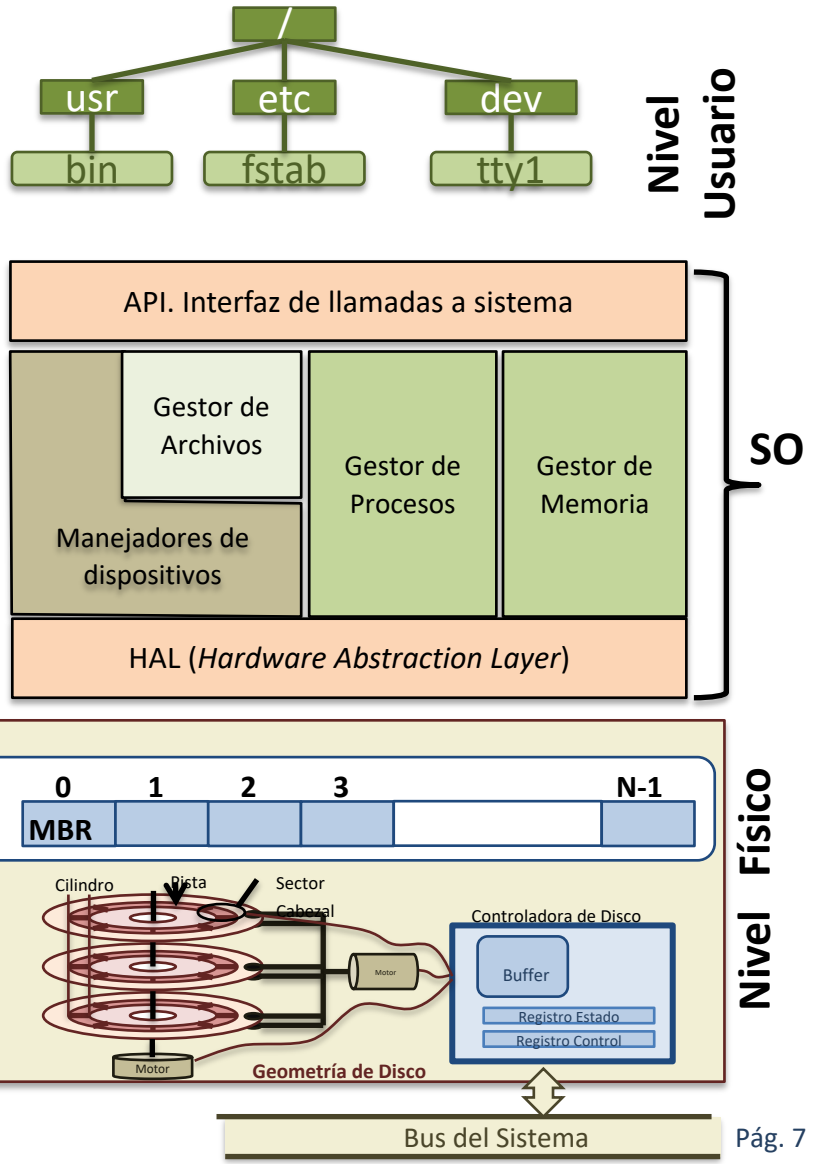
# Arquitectura del sistema de archivo

**Llamadas al sistema** (Bibliotecas de usuario)
Facilitan la gestión de ficheros y directorios desde el punto de vista del programador

**Gestor de Archivos:**
- Utiliza manejador de dispositivo para realizar transferencias de información entre disco y memoria
- Implementa mecanismos para ofrecer **coherencia**, **seguridad** y **protección**
- **Optimiza** prestaciones
- Crea los elementos básicos de la interfaz con los usuarios: **archivos** y **directorios**

**Manejador de dispositivo (driver)**
- Dialoga con el controlador de dispositivo
- Inicia las operaciones físicas y procesa el fin de la E/S

**Nivel físico:**
**Dispositivo + controlador**
- Dispositivo de bloques
- Geometría de disco:
  **Disco = Vector de bloques**



- **Arquitectura del sistema de archivos: Visión Usuario**

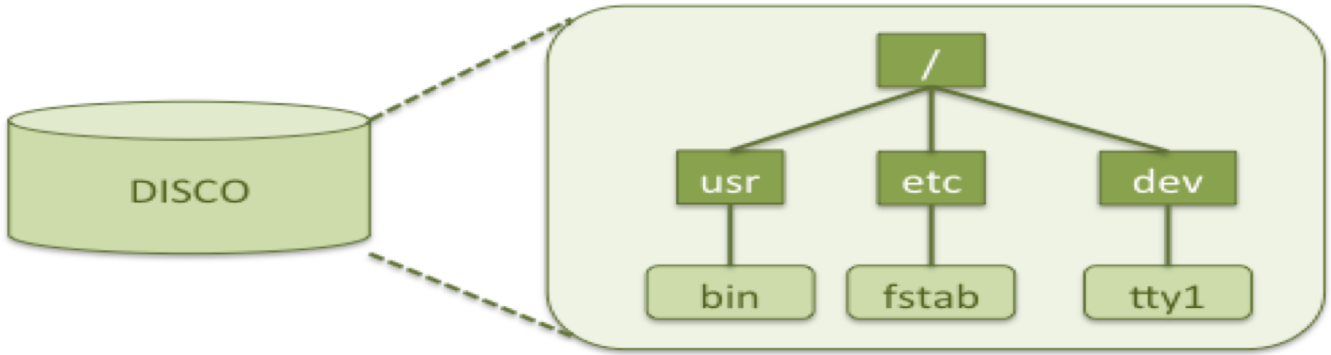
**Nivel Usuario**
Abstracciones de Archivo y Directorio

**Bibliotecas Usuario** (para operar con archivos)
Interfaz con las llamadas al sistema sobre ficheros y directorios

**Operaciones sobre archivos:**
- **Abrir y cerrar** archivos
- **Leer/Escribir** sobre archivo
- **Posicionarse** dentro de un archivo

**Operaciones sobre directorios:**
- **Crear/Borrar entradas** a directorio
- **Renombrar** archivos
- Buscar por nombre
- Recorrer el sistema de archivos

**Visión jerárquica**
Organización jerárquica en archivos y directorios



- **Apertura de un archivo**
  - Llamada al sistema que permite acceder al contenido del archivo
    - Los procesos tienen que abrir un archivo para leer o escribir en él
    - El proceso obtiene un identificador de archivo necesario para referenciar al archivo en sus operaciones de lectura, escritua y accesos.
  - Define los métodos de acceso (leer, escribir o ambos) y fija una posición inicial del puntero de lectura/escritura
    - Ejemplo: un proceso puede abrir un archivo para escritura
      - posicionando el puntero de escritura al principio del archivo se reescribir el contenido
      - posicionando el puntero de escritura al final se añade nuevo contenido al ya existente
    - Las transferencias posteriores irán referidas al puntero y lo desplazarán
  - Los permisos de acceso del proceso se verifican en la apertura
    - Sólo permitirá la apertura si el proceso tiene los permisos apropiados
    - La llamada open() falla si el modo especificado es inconsistente con los permisos del archivo

- **Operaciones sobre archivos**

  El SO ofrece un conjunto de llamadas para trabajar con archivos

  - **Crear archivo**
    - Requiere espacio libre en disco y crear una entrada nueva en un directorio
  - **Abrir archivo**
    - Operación necesaria antes de leer o escribir
  - **Leer archivo**
    - Requiere un identificador de archivo y puntero de posición de lectura
  - **Escribir archivo**
    - Requiere un identificador de archivo, datos a escribir y puntero de escritura
  - **Posicionamiento del puntero de acceso** (seek)
    - Operación que permite avanzar o retroceder punteros de lectura/escritura
  - **Cerrar archivo**
    - Operación inversa a abrir, libera las estructuras del SO que soportan el acceso a archivo acceder
  - **Borrado de archivo**
    - Libera el espacio en disco asociado al archivo y borra la entrada de directorio asociada

- **Contenido**
  - Arquitectura del sistema de archivos
  - **Concepto de archivo**
  - Asignación de bloques a archivos

# Concepto de archivo

- **Un archivo es:**
  - Un tipo abstracto de datos
  - Una colección de información relacionada por su creador
  - El elemento necesario para escribir/almacenar información en memoria secundaria

```c
#include <stdio.h>

main() {
  int x; /*variable entera*/
  int y;/*variable entera*/
  int *px; /* puntero a entero*/
  x=5;
  px=&x;/*px=direccion de x*/
  y=*px;/* y contiene el dato apuntado por px*/

  printf("x=%d\n",x);
  printf("y=%d\n",y);
  printf("*px=%d\n",*px);
 printf("px = %p\n", px);

}
```

Contenido de un archivo

# Concepto de archivo

- **Archivo** = Atributos + Datos
  - **Atributos de un archivo**
    - Varían de un sistema operativo a otro
      - Tipo
      - Tamaño
      - Información de protección
      - Propietario
      - Fecha y hora de creación
  - **Datos del archivo**
    - El SO ve el contenido del archivo como un vector de bytes, y son las aplicaciones las encargadas de interpretarlos
    - Un archivo puede almacenar distintos tipos de información: programa fuente, texto, código, gráficos, grabaciones sonoras, etc.
    - Los datos de un archivo, pueden tener una estructura determinada que depende del tipo de archivo
    - Un archivo ejecutable es una serie de secciones de código que el sistema es capaz de cargar y ejecutar

**METADATOS**
**Atributos** necesarios para gestionar el sistema de archivos

**DATOS**
**Contenido** del archivo. Ejemplo: caracteres, contenido binario, etc.

Listado de algunos atributos de los archivos contenidos en un directorio

```
gandreu@shell-labs:~/sisop/FSO/ejemplosC$ ls -lhi
total 136K
11928522 -rw-r--r-- 1 gandreu disca-upvnet  470 sep 20  2013 aritmetica_punteros.c
11930469 -rw-r--r-- 1 gandreu disca-upvnet  453 sep 26  2011 aritmetica_punteros.c~
11930470 -rwxr-xr-x 1 gandreu disca-upvnet 8,8K sep 26  2011 cir
11928236 -rw-r--r-- 1 gandreu disca-upvnet  193 sep 22  2011 circulo.c
11930472 -rwxr-xr-x 1 gandreu disca-upvnet 8,9K sep 26  2011 cua
11928246 -rwxr-xr-x 1 gandreu disca-upvnet 8,3K sep 16 16:41 cuad
11928433 -rwxr-xr-x 1 gandreu disca-upvnet 8,9K sep 20  2013 cuadrado
11928435 -rw-r--r-- 1 gandreu disca-upvnet  214 sep 22  2011 cuadrado.c
11928418 -rw-r--r-- 1 gandreu disca-upvnet  193 sep 22  2011 cuadrado.c~
11928437 -rwxr-xr-x 1 gandreu disca-upvnet 8,9K sep 22  2011 cuadro
11933192 -rw-r--r-- 1 gandreu disca-upvnet   80 sep 10  2013 error
11930471 -rw-r--r-- 1 gandreu disca-upvnet  579 sep 26  2011 hipotenusa.c
11930468 -rw-r--r-- 1 gandreu disca-upvnet  453 sep 26  2011 hipotenusa.c~
11927803 -rwxr-xr-x 1 gandreu disca-upvnet  424 jun 27  2014 hola.c
11928409 -rwxr-xr-x 1 gandreu disca-upvnet  241 jun 20  2014 hola.c~
11930473 -rwxr-xr-x 1 gandreu disca-upvnet 8,8K sep 26  2011 punt
11928436 -rwxr-xr-x 1 gandreu disca-upvnet 8,8K sep 22  2011 puntero
11928438 -rw-r--r-- 1 gandreu disca-upvnet  315 sep 22  2011 punteros.c
11928434 -rw-r--r-- 1 gandreu disca-upvnet  214 sep 22  2011 punteros.c~
11928243 -rw-r--r-- 1 gandreu disca-upvnet  525 sep 22  2011 variables.c
```

```c
#include <stdio.h>

main() {
  int x; /*variable entera*/
  int y;/*variable entera*/
  int *px; /* puntero a entero*/
  x=5;
  px=&x;/*px=direccion de x*/
  y=*px;/* y contiene el dato apuntado por px*/

  printf("x=%d\n",x);
  printf("y=%d\n",y);
  printf("*px=%d\n",*px);
 printf("px = %p\n", px);

}
```

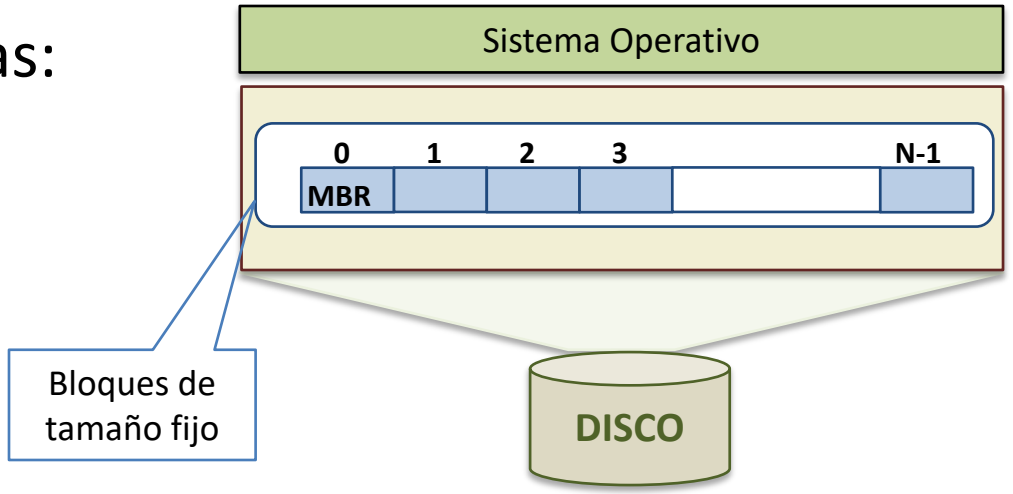
Contenido de un archivo

- **Métodos de acceso a los datos (contenido) del archivo**
  - Existen tres modos de acceso a la información de un archivo:
    - **Secuencial**
      - La información se procesa (ya sea para leer o escribir) en orden
      - En cada operación de lectura/escritura se utiliza y actualiza implícitamente un puntero de posición
    - **Directo**
      - Se considera al archivo compuesto de registros lógicos
      - En cada operación se indica como argumento el registro sobre el que se trabaja
    - **Mapeado en memoria**
      - Se asocia al archivo un rango de direcciones de memoria lógica de uno (o varios procesos)
      - De esta forma las lecturas/escrituras sobre el archivo se transforman en lecturas/escrituras sobre memoria principal
      - El SO es el encargado de actualizar la información en disco

- **Contenido**
  - Arquitectura del sistema de archivos
  - Concepto de archivo
  - **Asignación de bloques a archivos**

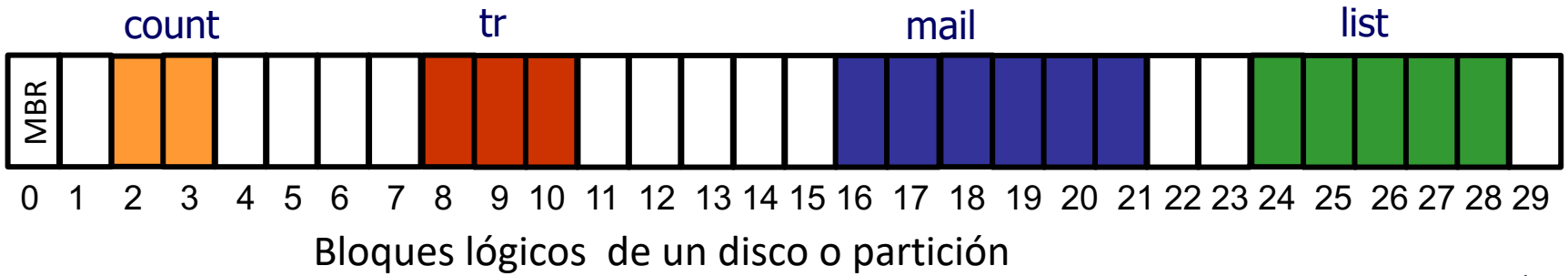
# Asignación de bloques a archivos

- ¿Cómo asignar espacio de disco a los archivos?
  - Los sistemas operativos modernos perciben los discos como un conjunto numerado de **bloques** de tamaño fijo
    - Tamaños de bloque habituales entre 512 Bytes y 4 KB
  - Se requiere
    - utilización eficiente del espacio de disco
    - acceso a los archivos de forma rápida
  - Existen tres alternativas:
    - Asignación Contigua
    - Asignación Enlazada
    - Asignación Indexada



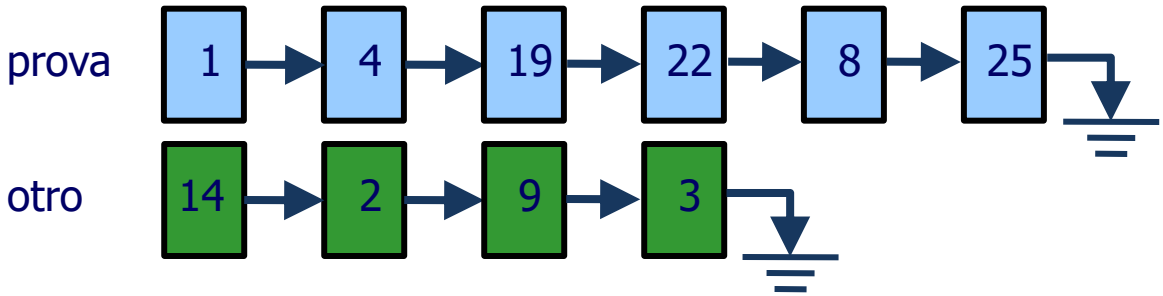
- **Asignación Contigua**
  - A cada archivo se le asignan un conjunto de **bloques** del disco **consecutivos**
  - Definida para cada archivo por el primer bloque del archivo y la longitud en bloques

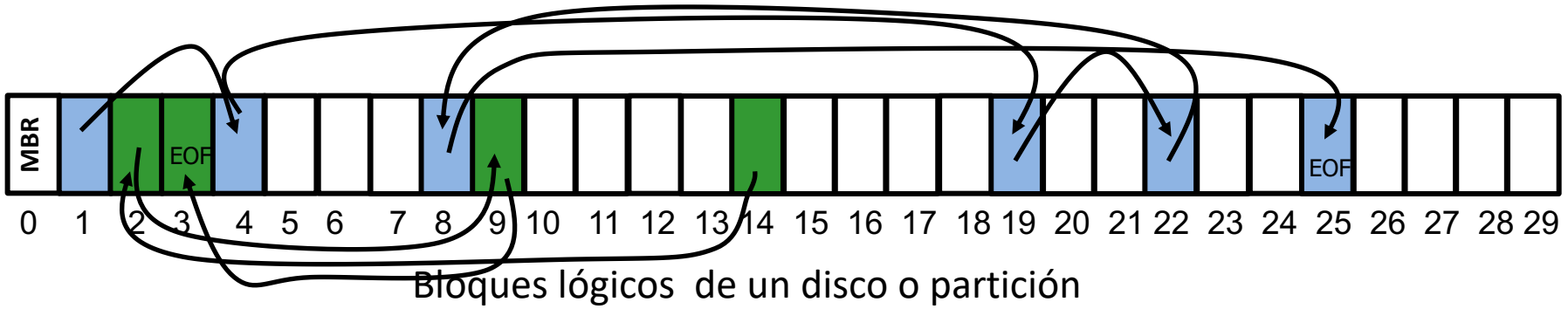
| Directorio | | |
|---|---|---|
| **Archivo** | **Inicio** | **longitud** |
| count | 2 | 2 |
| tr | 8 | 3 |
| mail | 16 | 6 |
| list | 24 | 5 |



- **Asignación Enlazada**
  - Los datos se almacenan en una **lista enlazada** de bloques, donde cada bloque apunta al siguiente

| Directorio | |
|---|---|
| **Archivo** | **Inicio** |
| prova | 1 |
| otro | 14 |



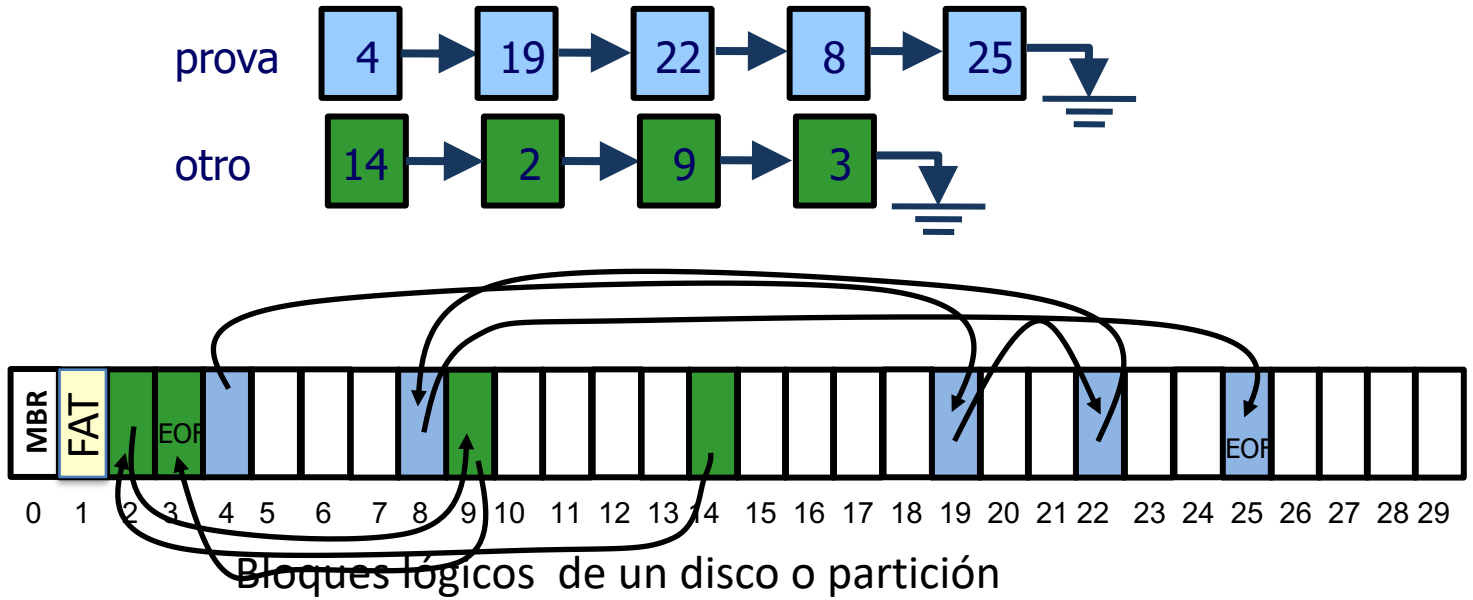



- **FAT** - variante de asignación **enlazada**

| | |
|---|---|
| 0 | reservado |
| 1 | reservado |
| 2 | 9 |
| 3 | Eof |
| 4 | 19 |
| 5 | - |
| 6 | - |
| 7 | 4 |
| 8 | 25 |
| 9 | 3 |
| 10 | - |
| 11 | - |
| 12 | - |
| 13 | - |
| 14 | 2 |
| 15 | - |
| 16 | - |
| 17 | - |
| 18 | - |
| 19 | 22 |
| 20 | - |
| 21 | - |
| 22 | 8 |
| 23 | - |
| 24 | - |
| 25 | Eof |
| 26 | - |
| 27 | - |
| 28 | - |
| 29 | - |

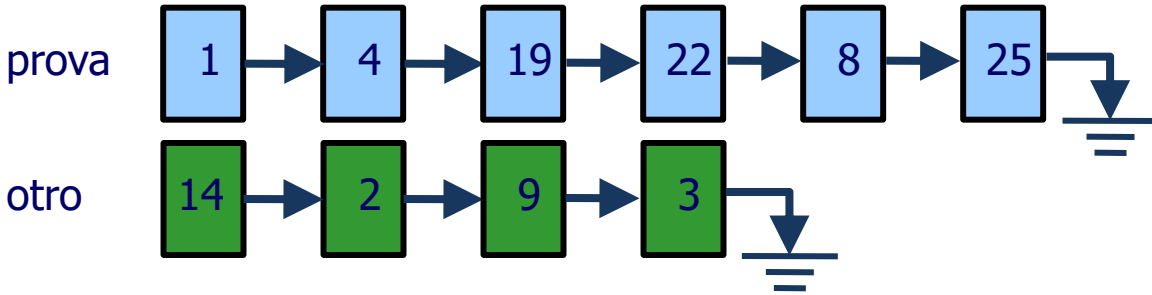
- Las referencias no se guardan en los bloques, sino en un área de disco dedicada
- EOF marca el fin de cada lista

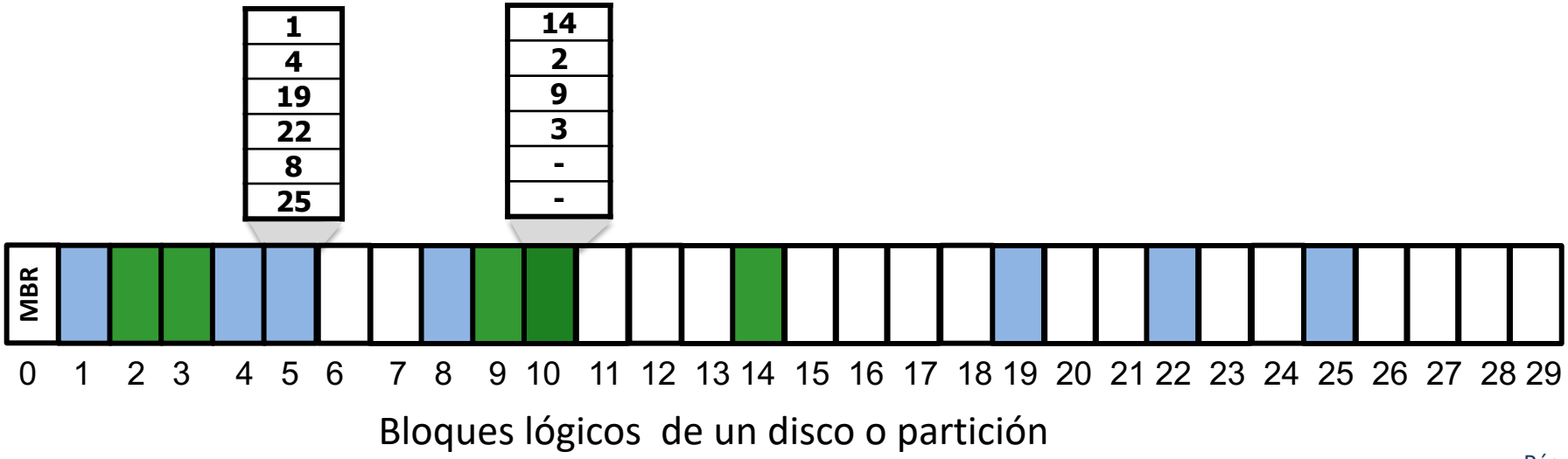
| Directorio | |
|---|---|
| Archivo | Inicio |
| prova | 4 |
| otro | 14 |



- **Asignación Indexada**
  - Un **bloque índice** contiene punteros a los bloques que contienen los datos

| Directorio | |
|---|---|
| **Archivo** | **Inicio** |
| prova | 5 |
| otro | 10 |

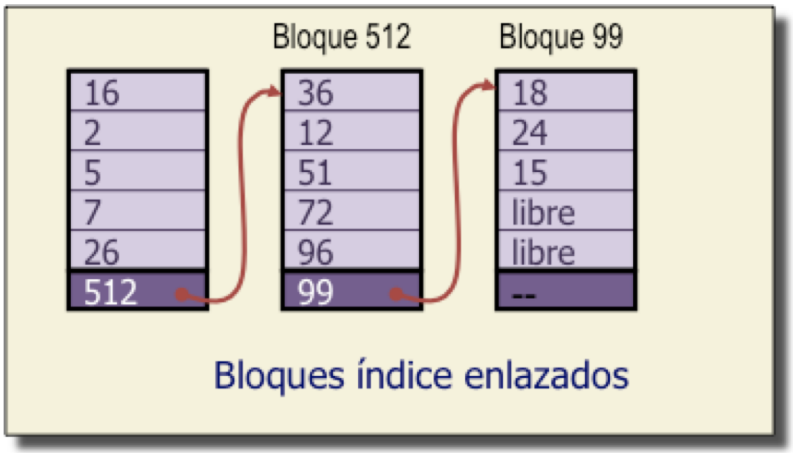


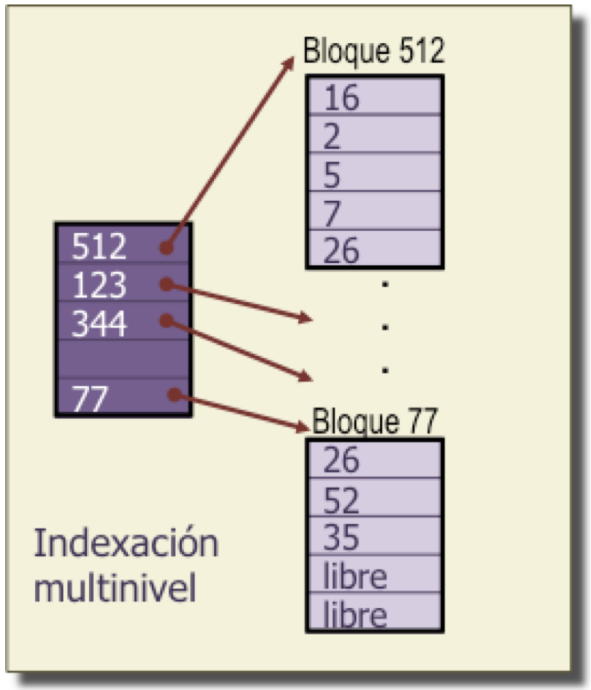




Bloques lógicos de un disco o partición

# Asignación de bloques a archivos

- **Indexación multinivel**
  - Es una variante de asignación indexada
  - Motivación
    - Implementar archivos grandes requiere **varios bloques índice**
  - Solución
    - Una referencia puede apuntar a un bloque de datos o a otro bloque de índices



- Análisis de tipos de asignación

| | Ventajas | Inconvenientes |
|---|---|---|
| **Contigua** | Es el más eficiente<br>Soporta acceso secuencial y directo<br>Velocidad de acceso estable<br>Ideal con dispositivos de sólo lectura (CD, DVD, etc) | Gestión de espacio compleja (ej.- determinación del mejor hueco, problemas cuando crece el archivo,.. )<br>problemas de fragmentación externa (necesidad de compactación) |
| **Enlazada** | No limita el crecimiento de los archivos | No soporta acceso directo<br>Poco robusta ante fallos |
| **Fat** | Si se copia la FAT a memoria, soporta acceso directo<br>Facilita la gestión de espacio libre | Si la FAT no cabe en RAM, no presenta ninguna ventaja -> sólo útil para dispositivos de baja capacidad<br>Poco robusta ante fallos |
| **Indexada** | Soporta acceso secuencial y directo | Limita el crecimiento (índice de talla limitada) |
| **Indexada multinivel** | No limita el crecimiento de los archivos | Para localizar un bloque pueden ser necesarios varios accesos a disco |

NOTA.- en todos los casos aparece **fragmentación interna,** desperdiciando por término medio la mitad del último bloque