

3.4 Organización del Sistema de ficheros. Espacios de nombres, directorios.

El porqué del sistema de ficheros (X03)

- Necesidades de un sistema de almacenamiento
 - Permitir **organizar de forma ordenada la información** almacenada los dispositivos de almacenamiento del SO.
 - Poder **cumplir con las necesidades de gestión** de datos de los usuarios
 - Permitir **guardar gran cantidad de información** y durante mucho tiempo.
 - **Garantizar la validez de la información** (en la medida de lo posible)
 - **Soportar diferentes tipos de dispositivos**: Discos, cintas, IDE/SCSI, NFS
 - Proporcionar un **acceso rápido a la información** almacenada
 - **Ser fiable** (minimizar o eliminar la posibilidad de pérdida de datos)
 - Proporcionar **mecanismos de seguridad** (permitir/limitar el acceso a la información)
 - Permitir **acceso concurrente**
 - Proporcionar una **interfaz estándar**.

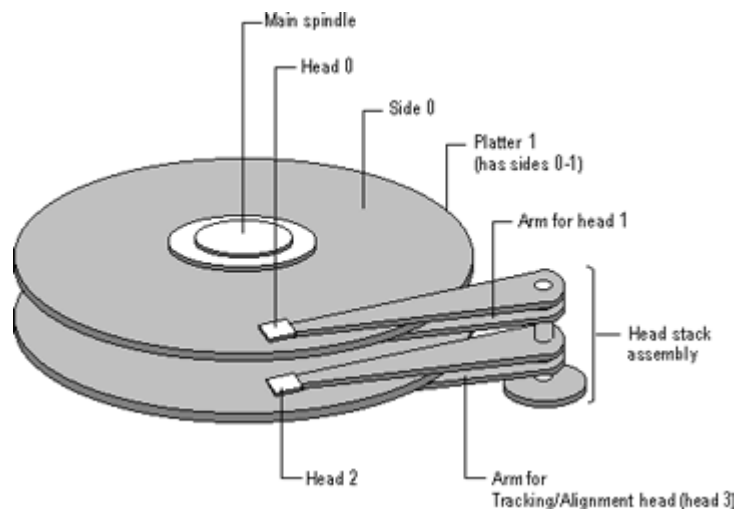
Definición

- El Sistema de ficheros (FS) consiste en una *abstracción de los dispositivos de almacenamiento* (discos, cintas, CDs,...)
- Un Sistema de Ficheros (FS) es un *método para guardar y organizar información* de un computador con el objetivo de *facilitar la búsqueda y acceso a la información*.
- Visión del **usuario**:
 - Lo importante para el usuario es **la interfaz** (“como se puede acceder”): **funciones** de crear, borrar, leer, escribir,...
- Visión del **diseñador** del Sistema Operativo:
 - Lo importante para el diseñador del SO es que el FS sea **eficiente, estable, seguro**,...

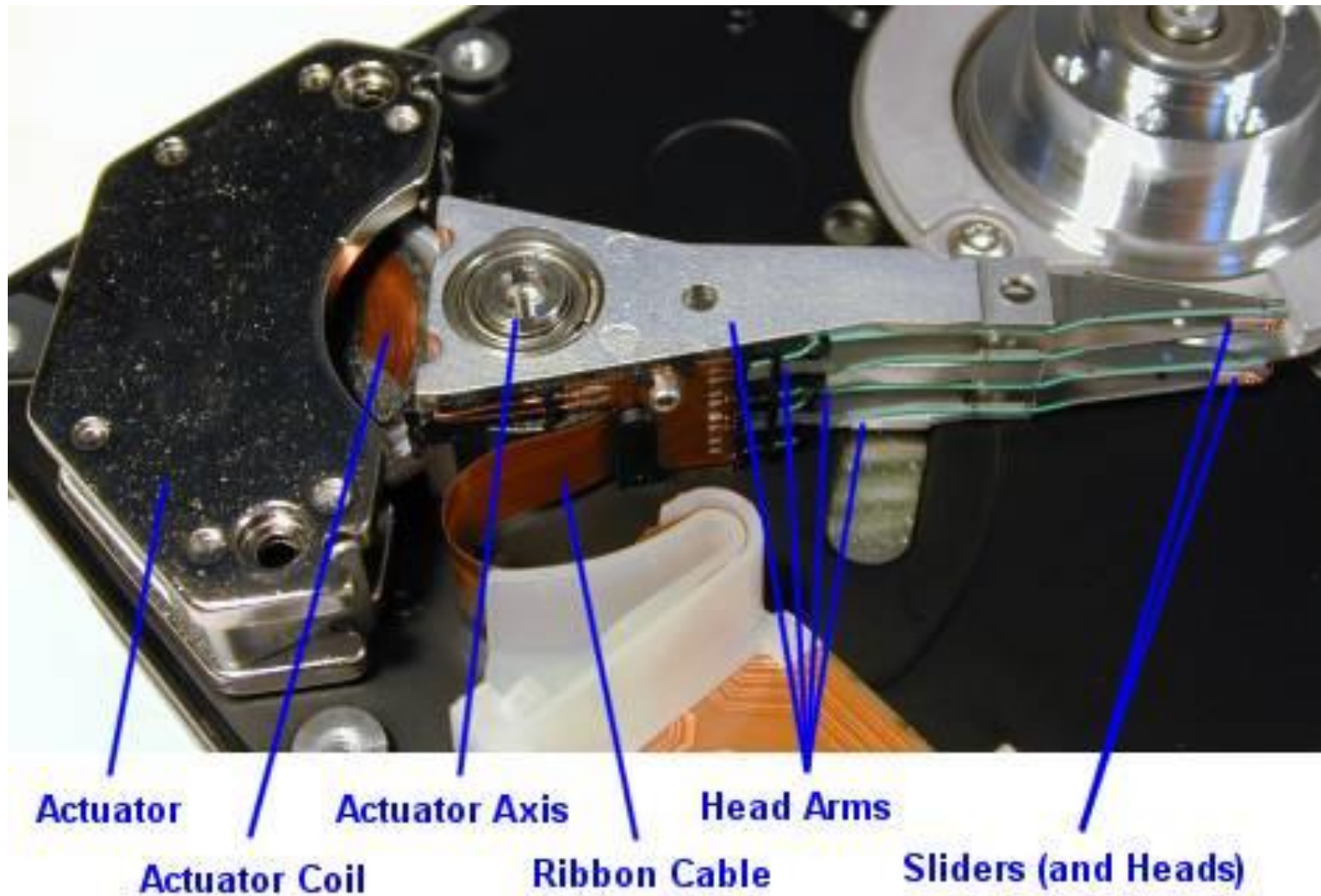
Organización del Sistema de ficheros. Espacios de nombres, directorios.

1. Estructura Física de Disp. Almacenamiento
2. Estructura Lógica de Disp. Almacenamiento
3. Organización del Sistema de ficheros
 - Ficheros, Directorios

Estructura Física de un Disco Duro (HDD)



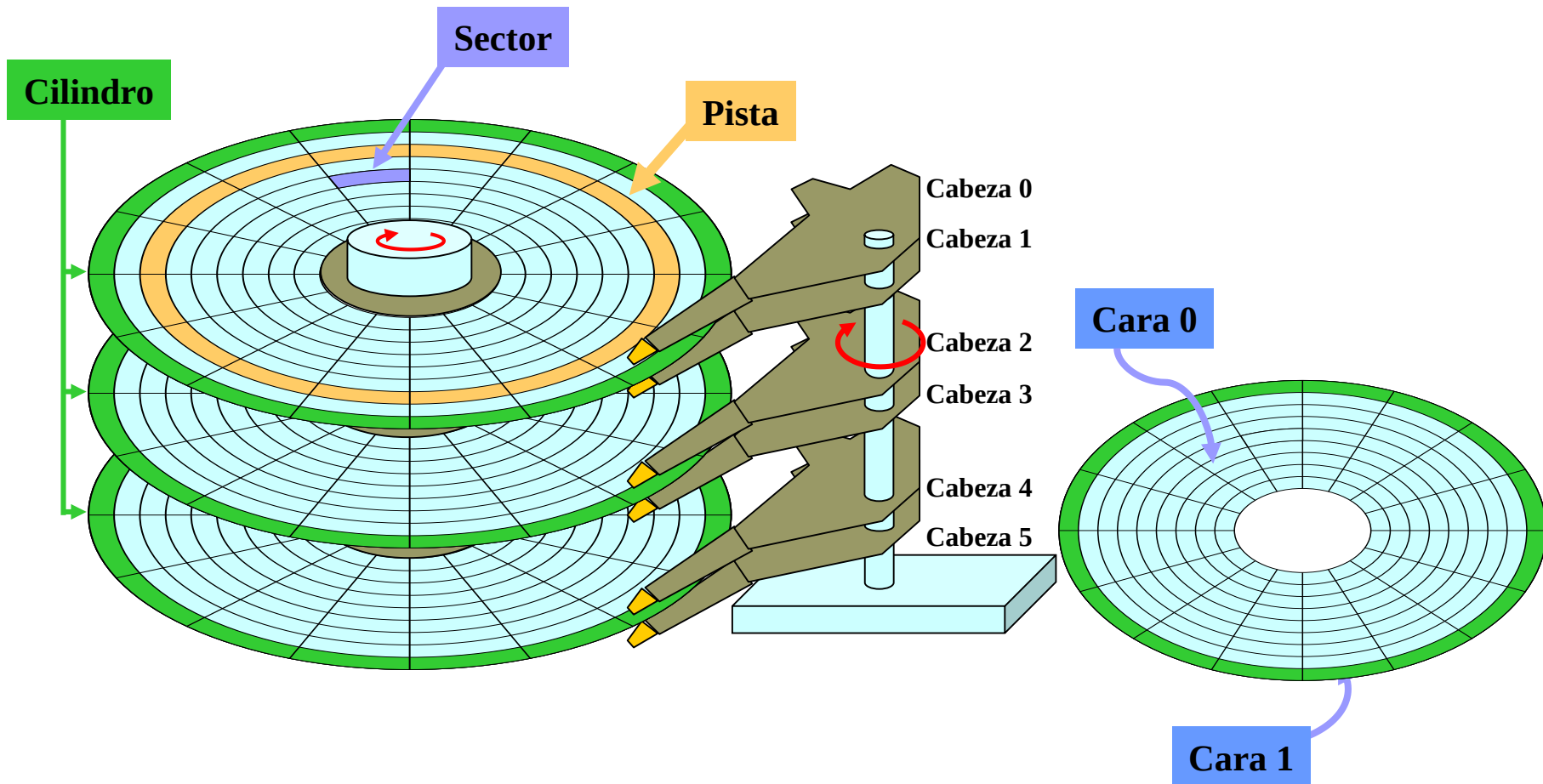
Estructura Física de un Disco Duro (cont I)



Estructura Física de un Disco Duro (cont II)



Estructura Física de un Disco Duro (cont III)



Estructura Física de un Disco de estado sólido SDD



Source : <https://www.crucial.es/esp/es/storage-ssd-mx500>

Estructura Lógica de los Dispositivos de Almacenamiento

□ Partición

- División lógica de un disco duro.
- El Sistema Operativo puede formatear por separado cada partición.

https://en.wikipedia.org/wiki/GUID_Partition_Table

https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface

https://en.wikipedia.org/wiki/EFI_system_partition#Linux

□ Volumen

- Método de ubicar espacio en dispositivos de almacenamiento más flexible que las particiones.
- Los dispositivos se trocean en partes de pequeño tamaño (unos 4 MB) se agrupan en volúmenes.
- Un volumen puede estar formado por parte de una partición, una o varias particiones o incluso varias partes de particiones de diferentes dispositivos.

https://en.wikipedia.org/wiki/Logical_volume_management#EXTENT

□ Formateo: Proceso de preparar un dispositivo de almacenamiento para su uso en un sistema de ficheros.

- Formateo de bajo nivel:
 - División de las caras de un disco en pistas, sectores y cilindros.
 - Estas divisiones permitirán acceder a los datos guardados en el disco.
- Formateo de alto nivel
 - Propio del Sistema de Ficheros (FS), inicializa el mismo para su uso en un FS concreto
 - P.E. EXT2 de UNIX: Crea el superbloque e inicializa la Lista de Inodes,...

□ La **unidad mínima** de acceso es el **sector**

□ Modo de direccionamiento:

- **LBA** (Logical Block Addressing) la dirección es un **número secuencial** que se le asigna a un sector del dispositivo https://en.wikipedia.org/wiki/Logical_block_addressing
- **CHS** (Cylinder-Head-Sector) la dirección se especifica mediante el **cilindro** (radio), la **cabeza** (cara del plato), y el **sector** (posición angular) <https://en.wikipedia.org/wiki/Cylinder-head-sector>

Organización del Sistema de ficheros.

- *Fichero* o archivo para guardar datos o código
- *Directorios* o carpetas – para organizar ficheros
- Todos ellos en una *estructura en Árbol*

Analogía del sistema de ficheros

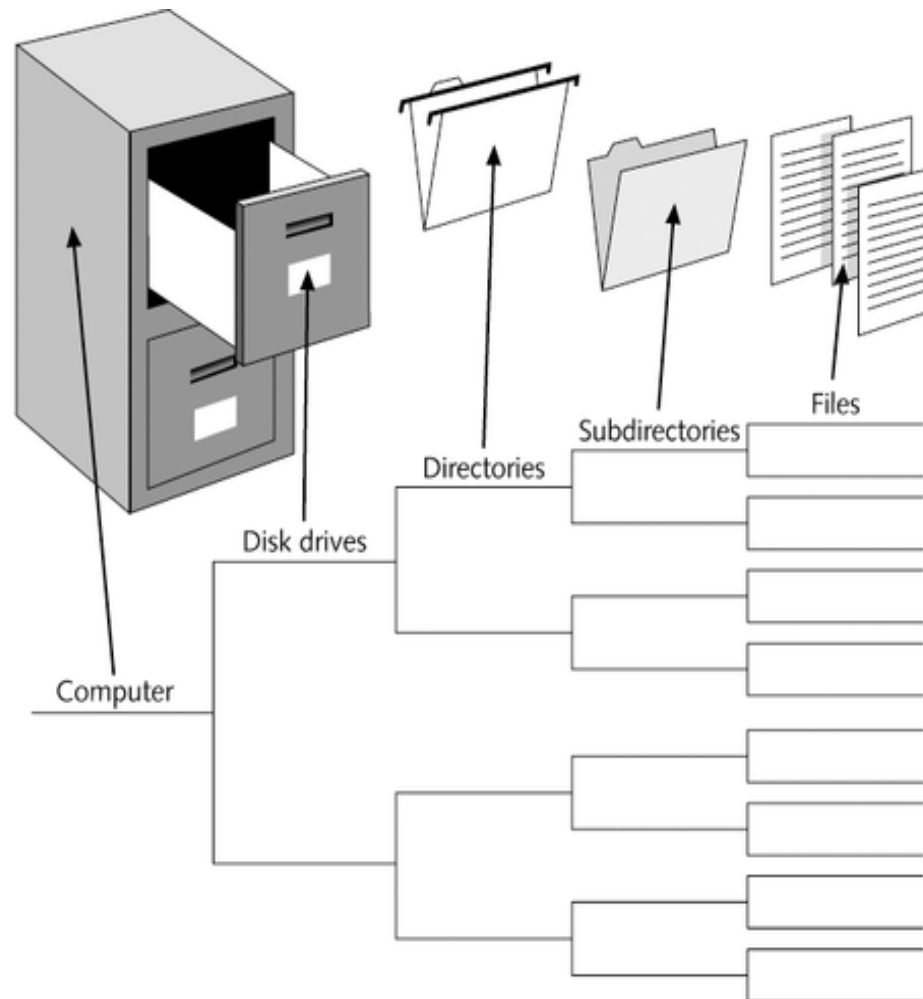


Figure 3-1 A file system

(Estructura en forma de árbol)



Proyecto P06. Cómo gestionar los directorios



¿Cómo implementamos entonces un comando como ls?

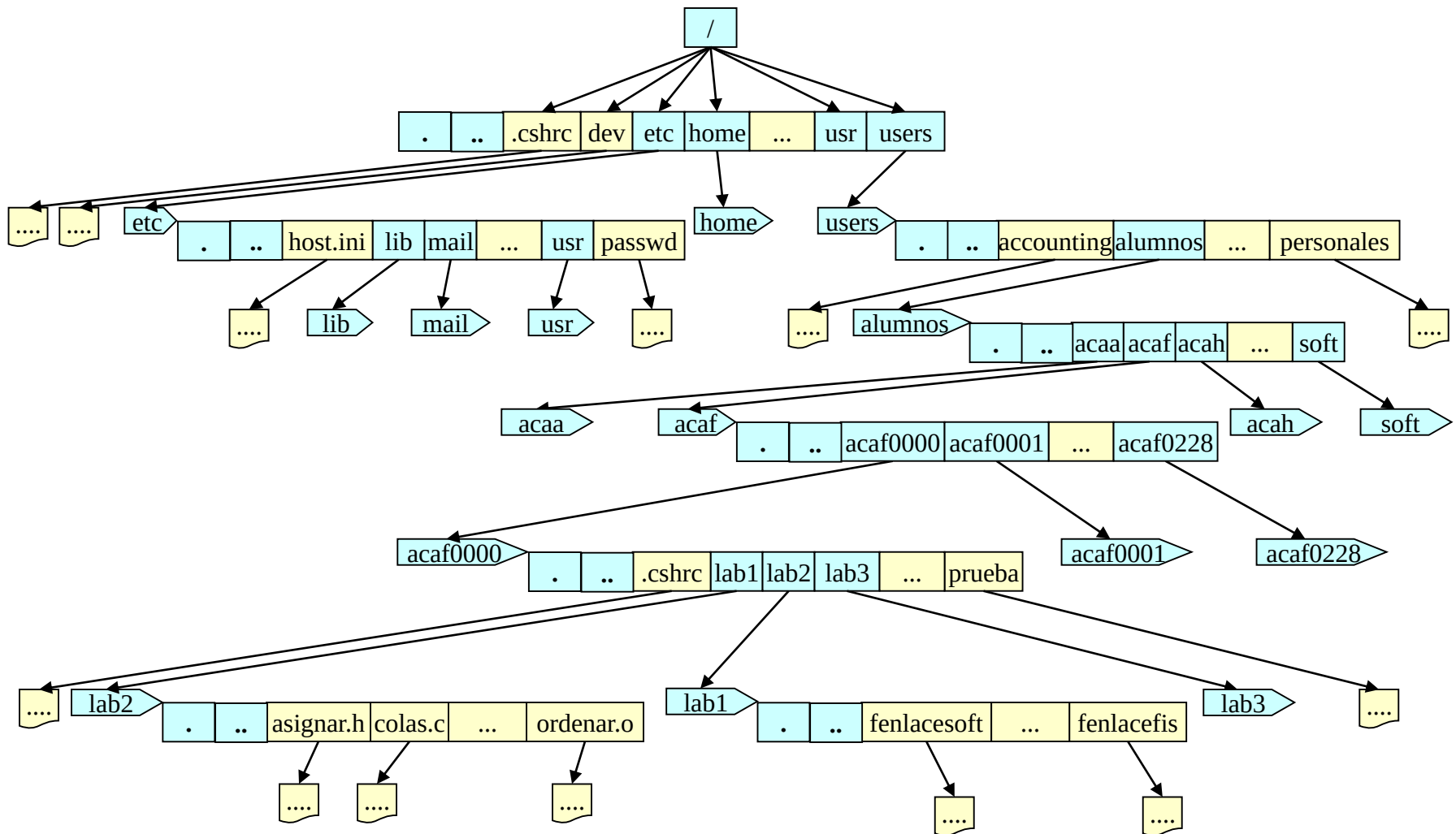
¿Cómo podemos crear un directorio?

Las funciones para gestión de directorios

Las funciones para obtener las propiedades de los ficheros

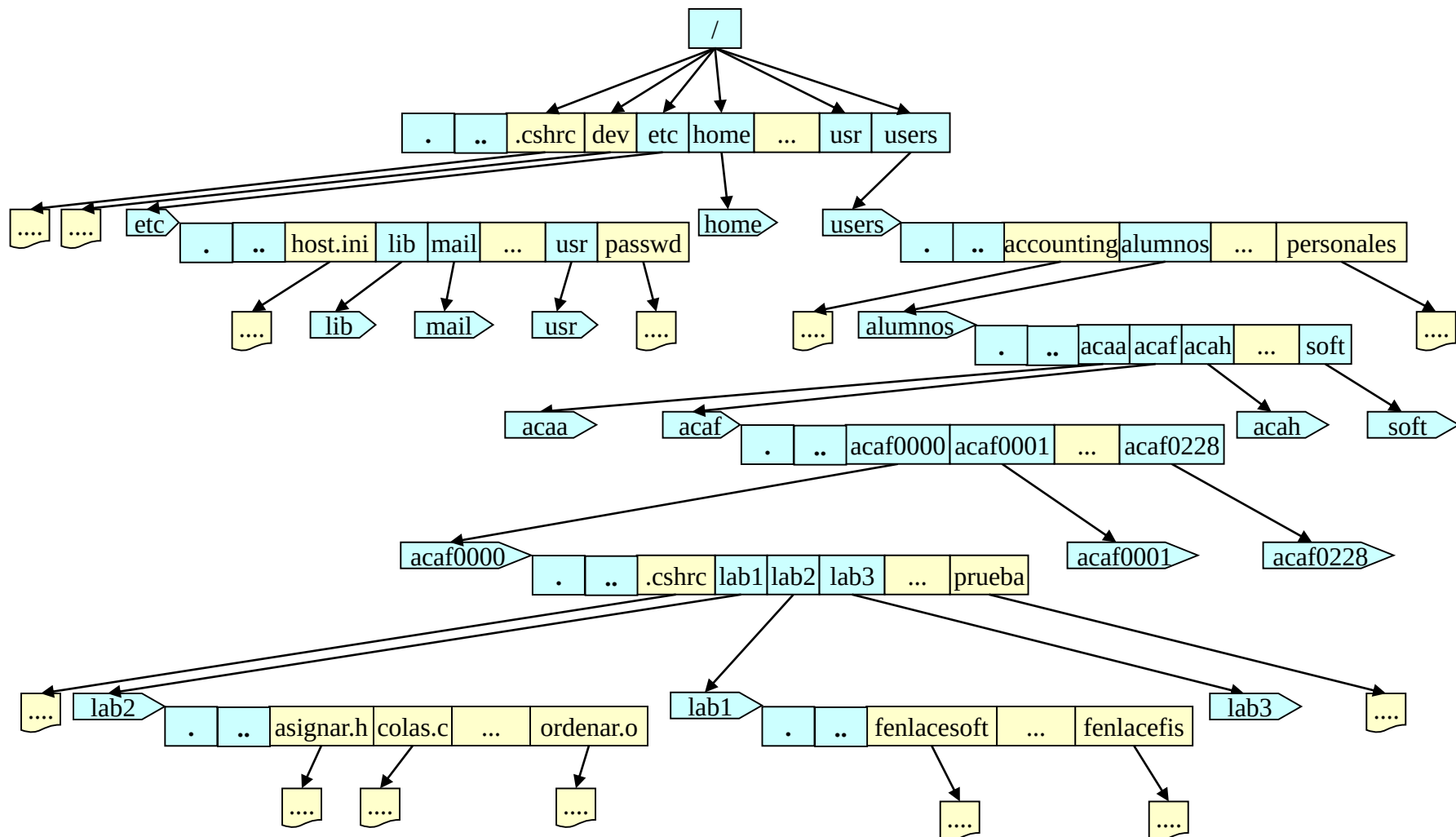
Visión del Sistema de Ficheros

(Estructura en forma de árbol)



Visión del Sistema de Ficheros

(Estructura en forma de árbol)



Organización de la estructura de los directorios

- A la hora de organizar la estructura de los directorios es necesario considerar lo siguiente:
 - Ficheros del Sistema Operativo
 - Aplicaciones software
 - Ficheros de trabajo, es decir, documentos, gráficos, hojas de cálculo y bases de datos
 - Los ficheros que se comparten en red
 - Los ficheros de las utilidades
 - Los ficheros temporales

Directorios

- / – directorio raíz en Unix
- ./ – directorio actual
- ../ – directorio padre

Referencia a un fichero (*nombre*)

[**Camino**]**Nombre**

- ❑ **Camino**: ruta que permite localizar el fichero en el árbol del sistema de ficheros.
 - **Absoluto**: ruta desde la raíz del árbol (comienza por **/**).
 - **Relativo**: ruta desde el directorio en curso. (comienza por distinto de **/**).
(directorio en el que encuentra el proceso que lo referencia)

Si el camino es vacío quiere decir que el fichero se encuentra en el directorio en curso.

- ❑ **Nombre**: nombre de la entrada correspondiente al fichero en el directorio.

Ejemplos:

```
lab1/ejemplo
../lab1/ejemplo
../acaf0003/lab1/ejemplo
/users/alumnos/acaf/acaf0003/lab1/ejemplo
```

Información complementaria : Ayuda de Linux *man PATH_RESOLUTION* (Sección 7)

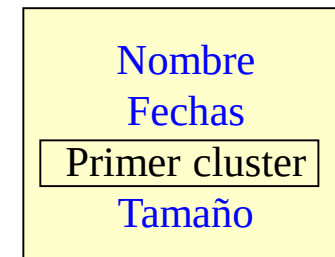
Información de un fichero

Un fichero está compuesto por Información de control y por Datos

❑ **Información de control: ¿Ubicación?**

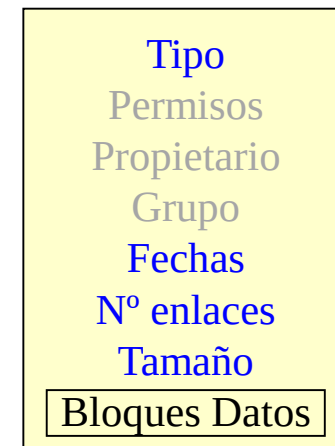
- En el **directorio** (FAT-16)

*Entrada del
directorio*



- En propio **fichero/i-node** (UNIX/Linux)

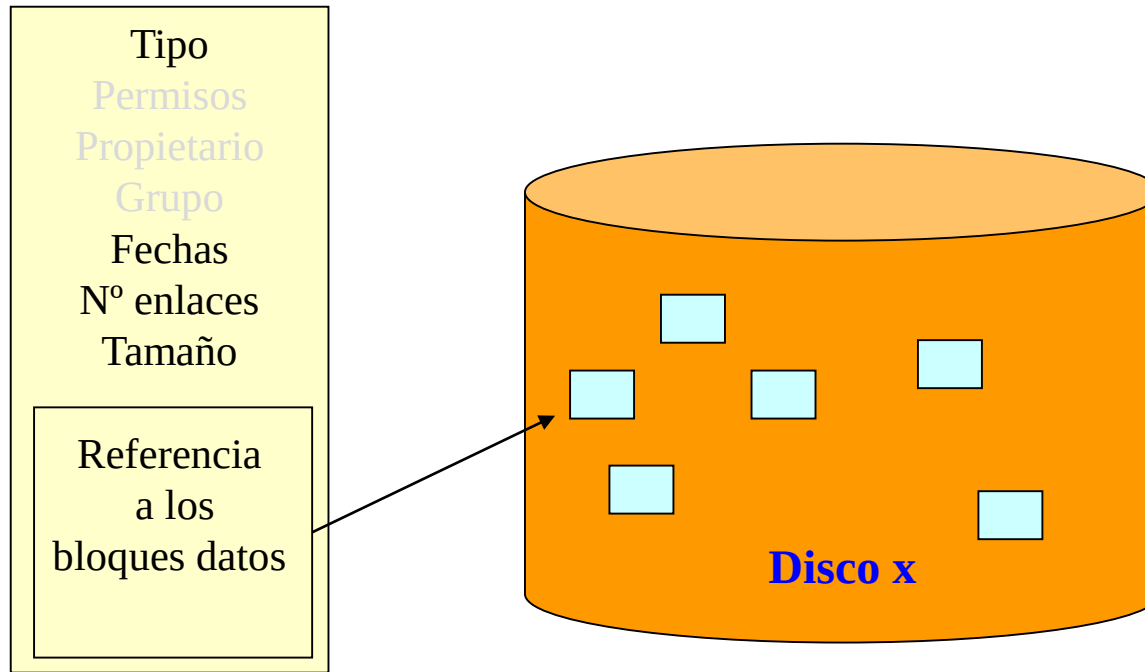
Inode



❑ **Datos o contenido**

Información de control de un fichero

Inode



Indica de alguna forma los bloques del dispositivo de almacenamiento donde se encuentran los datos del fichero

Información de un fichero (Datos o contenido)

❑ Datos o contenido

➤ Lista encadenada.

P.E. FAT16 : lista de nº de clusters

0	1	2	3	4	5	6		K
1	2	FF	5	FF	6	4	...	

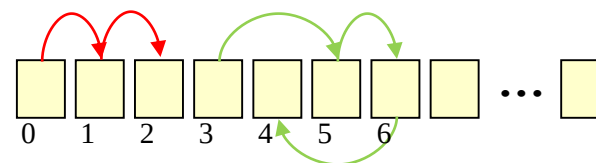
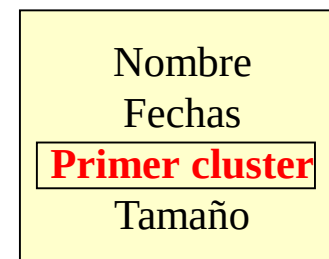
0 -> 1 -> 2 -> FF

3 -> 5 -> 6 -> 4 -> FF

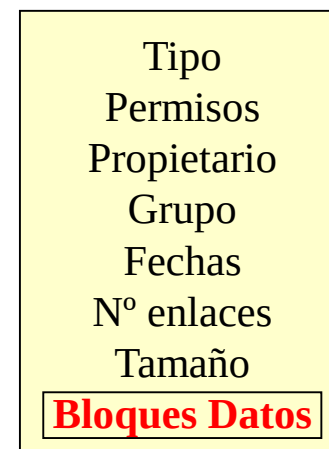
➤ Vector de Nº de Bloques

P.E. UNIX

*Entrada del
directorio*

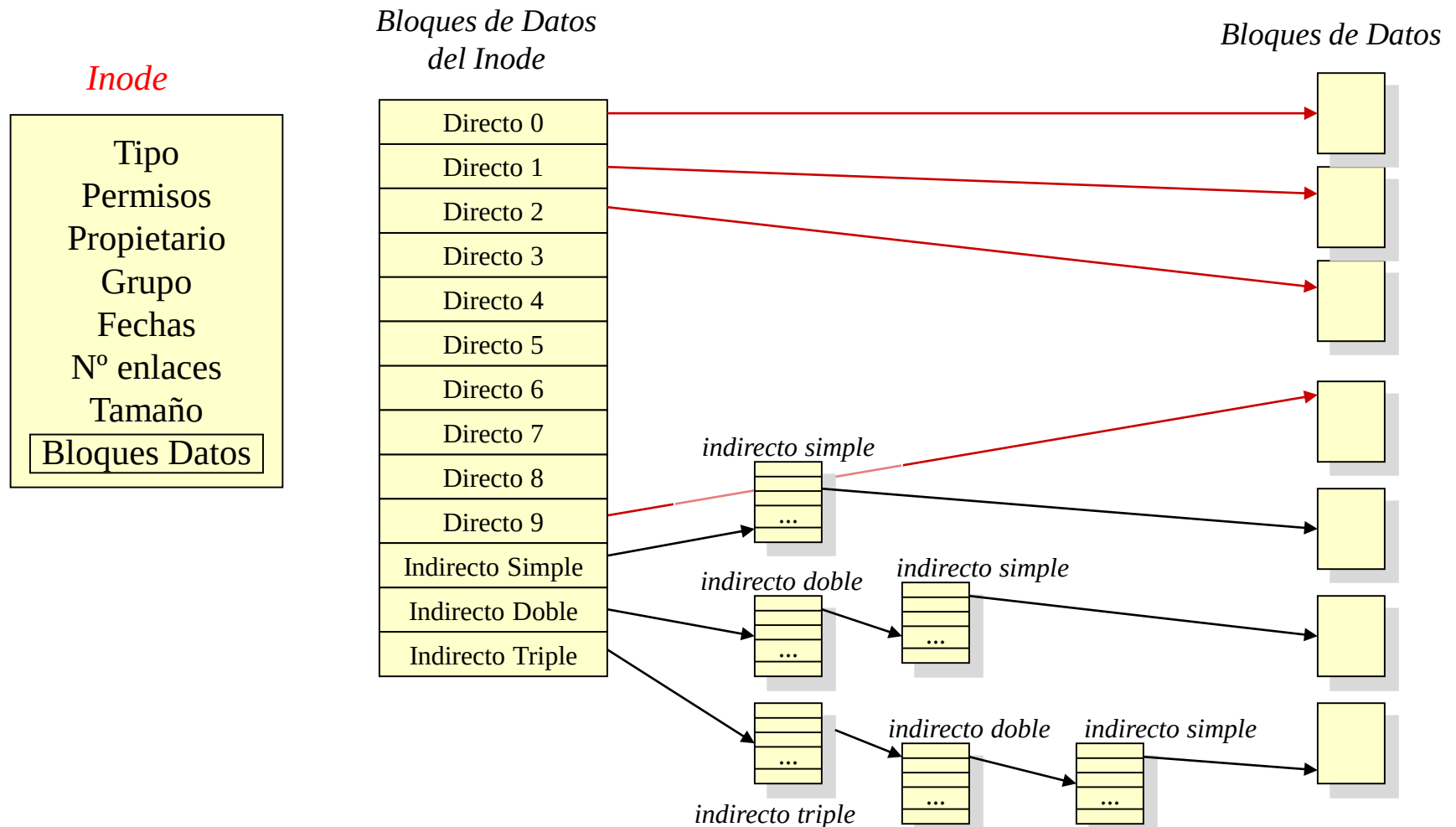


Inode



Datos o Contenido en UNIX/Linux (Información de un fichero)

➤ Vector de Nº de Bloques



Directorios

❑ Windows - FAT

■ Información de control

- ▣ En el **directorio** (FAT)

*Entrada del
directorio*

Nombre
Fechas
Primer bloque
Tamaño

Directorio

<u>Nombre</u>	<u>Fechas</u>	<u>1er Bloque</u>	<u>Tamaño</u>
F1.dat	130102	234567	1024
F2.dat	130102	234555	39000
Fichero.txt	130302	269999	2367
...			

❑ UNIX-Linux

■ Información de control

- ▣ En el **i-node**
(EXT, ...)

Inode

Tipo
Permisos
Propietario
Grupo
Fechas
Nº enlaces
Tamaño
Bloques Datos

*Directorio
(fichero especial)*

<u>Nombre</u>	<u>NºInode</u>
.	8733108
..	5243238
asignar.h	8733916
colas.c	8733917
colas.h	8733918
...	

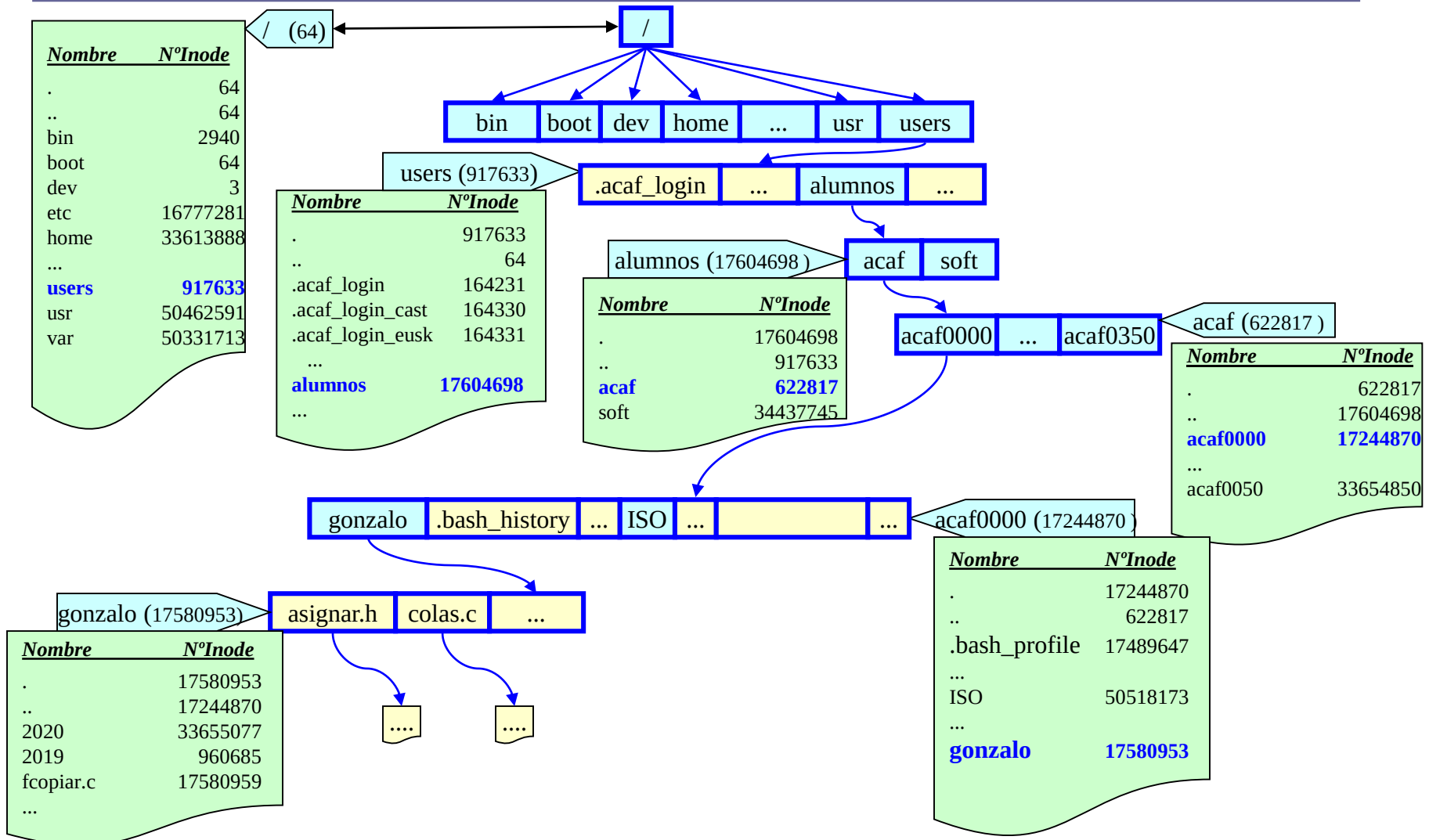
Directorios en UNIX

- Están estructurados en forma de árbol jerárquico.
- Pueden contener ficheros (archivos) y subdirectorios.
- Se implementan como ficheros especiales:
 - Contiene registros de tipo *struct dirent* (entrada de directorio).
 - Cada entrada (registro) consta al menos de dos datos: un número de i-node y de un nombre de fichero

```
struct dirent {  
    long    d_ino;           // N° de Inode  
    off_t   d_off;  
    unsigned short int d_reclen; // Long. de d_name  
    char    d_name [NAME_LEN]; // Nombre de la entrada  
                                // (acabado en nulo)  
};
```

Nombre	N°Inode
.	8733108
..	5243238
asignar.h	8733916
colas.c	8733917
colas.h	8733918
...	

Directorios en UNIX



Gestión de directorios

Funciones de Biblioteca de C

▣ Funciones relacionadas con directorios

- `DIR *opendir (char *path);`
- `int closedir (DIR *dir);`
- `struct dirent * readdir (DIR *dir);`

Ejemplo de uso

```
#include <sys/types.h>
#include <dirent.h>
...
DIR *ddir;
struct dirent *entrada;

ddir = opendir("dir1");
entrada = readdir(ddir);

write(1, entrada->d_name, strlen(entrada->d_name));

closedir(ddir);
...
```

```
struct dirent
{
    long d_ino;
    ...
    char d_name[256];
}
```

Llamadas al Sistema (UNIX)

Gestión de directorios

■ Control de ficheros-directorios

Crear directorio:

- `int mkdir (char *path, mode_t mode);`
`mode` : Permisos

Borrar directorio:

- `int rmdir (char *path);`

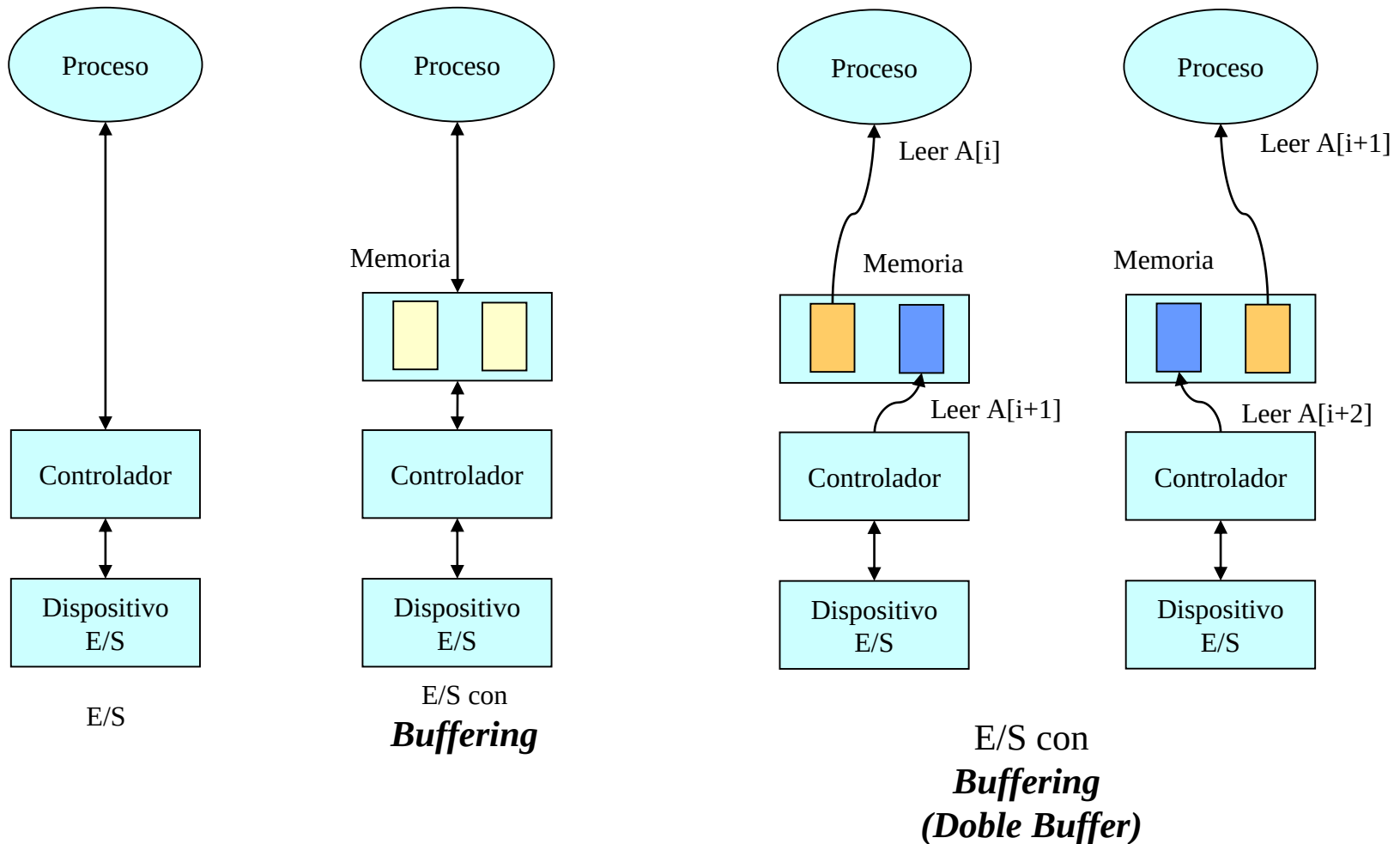
Cambiar el directorio actual (cwd):

- `int chdir (char *path);`
- `int fchdir (int fd);`

3.5 “Buffering” de E/S: En las bibliotecas de E/S y en las llamadas al sistema.

- Actividad **X02. Pero, ¿realmente escribe en el disco?**

E/S con *Buffering* (buffer en las llamadas al sistema)



3.6 Acceso avanzado a las propiedades de los dispositivos (Ficheros)

Información de un fichero

Un fichero está compuesto por Información de control y por Datos

□ **Información de control: ¿Ubicación?**

- En el **directorio** (FAT-16)

*Entrada del
directorio*

Nombre
Fechas
Primer cluster
Tamaño

- En propio **fichero/i-node**
(UNIX/Linux)

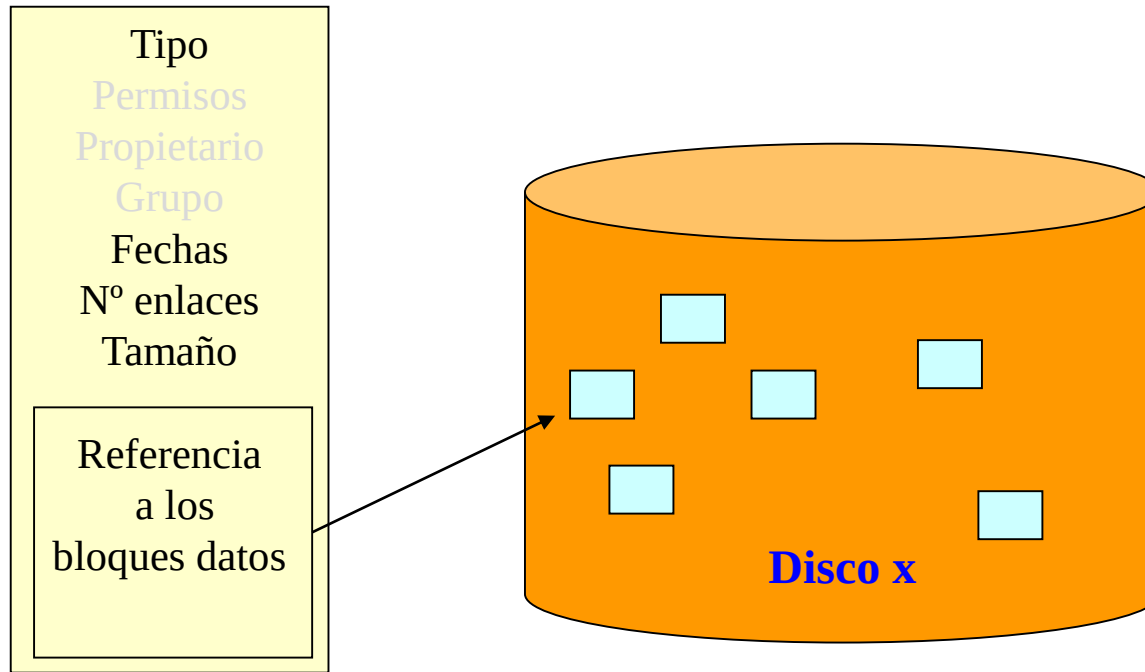
Inode

Tipo
Permisos
Propietario
Grupo
Fechas
Nº enlaces
Tamaño
Bloques Datos

□ **Datos o contenido**

Información de control de un fichero

Inode



Datos: Indica de alguna forma los bloques del dispositivo de almacenamiento donde se encuentran los datos del fichero

Estructura del i-node (UNIX)

i-node

Tipo

Tipo de fichero: Ordinario, Directorio, caracteres, de bloques, Pipe, Enlace, Shocket

Tamaño

Fechas

Nºenlaces

...

Tamaño en bytes del fichero

Varias fechas como la del último acceso o la de la última modificación de datos.

Nº de enlaces al fichero (nº de nombres en el sistema de ficheros)

**Índices de
Bloques Datos**

Otros datos

Estructura *stat*

(una copia de parte de la información contenida en el i-node)

```
#include <sys/stat.h>
```

```
struct stat {  
    dev_t      st_dev          : identificador del dispositivo que contiene el  
                                fichero (short)  
    ino_t      st_ino          : número de inodo (ushort)  
    mode_t     st_mode        : modo (short) bits de permisos  
    nlink_t    st_nlink       : número de enlaces (short)  
    uid_t      st_uid        : identificador del dueño (ushort)  
    gid_t      st_gid        : identificador de grupo (ushort)  
    ...  
    dev_t      st_rdev       : tipo de dispositivo para ficheros especiales (short)  
    off_t      st_size       : Tamaño en bytes (long) (0 en los ficheros  
    blksize_t  st_blksize    : Tamaño en bytes (long) (0 en los ficheros especiales)  
    blkcnt_t   st_blocks    : Numero de bloques asignados  
  
    struct timespec st_atim; /* time of last access */  
    struct timespec st_mtim; /* time of last modification */  
    struct timespec st_ctim; /* time of last status change */  
  
    #define st_atime    st_atim.tv_sec /* Backward compatibility */  
    #define st_mtime    st_mtim.tv_sec  
    #define st_ctime    st_ctim.tv_sec  
  
};
```

Propiedades de los ficheros.

Llamadas al Sistema (UNIX)

■ Obtener las propiedades de Ficheros de dispositivos

```
#include <sys/stat.h>
```

```
#include <unistd.h>
```

- `int stat (char *path, struct stat *sbuf);`
- `int lstat (char *path, struct stat *sbuf);`
- `int fstat (int fd, struct stat *sbuf);`

■ Ejemplo de uso:

```
#include <sys/stat.h>
#include <unistd.h>

...
int          Ret;
struct stat  StVar;
off_t        Size;
char         NombreFich[256];

...
strcpy(NombreFich, argv[1]);
Ret = stat(NombreFich, & StVar);
Size = StVar.st_size;
printf("%20s Tam: (%8d)\n", NombreFich, Size);

...
```

Propiedades de los ficheros.

Llamadas al Sistema (UNIX): Campo **st_mode**

Se definen las siguientes macros POSIX para comprobar el **tipo de fichero** (el tipo es parte del campo **st_mode**):

m:	<table><tr><th>tipo</th><th>Permisos</th></tr></table>	tipo	Permisos
tipo	Permisos		

`S_ISLNK(m)` es un enlace simbólico?

`S_ISREG(m)` un fichero regular

`S_ISDIR(m)` un directorio

`S_ISCHR(m)` un dispositivo de caracteres

`S_ISBLK(m)` un dispositivo de bloques

`S_ISFIFO(m)` un fifo con nombre

`S_ISSOCK(m)` un socket

Propiedades de los ficheros:

Ejemplo de código para obtener el tipo de un fichero

Ejemplo de uso:

```
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
...

struct stat    statVar;
mode_t        Tipo;
char          NombreFich[256];
int           ret;
...
strcpy(NombreFich, "fich1.dat");
ret = stat(NombreFich, &statVar);
Tipo = statVar.st_mode;
if (S_ISREG(Tipo))
    printf("%s es un fichero Regular\n", NombreFich);
...
```