SISTEMAS OPERATIVOS: SISTEMAS DE FICHEROS

Ficheros, directorios y sistema de ficheros

A recordar…

Alejandro Calderón Mateos

Antes de clase | Clase | Después de clase

Preparar los pre-requisitos.

Estudiar el material asociado a la **bibliografía**:
las transparencias solo no son suficiente.
Preguntar dudas (especialmente tras estudio).

Ejercitar las competencias:

- Realizar todos los **ejercicios**.
- Realizar los **cuadernos de prácticas** y las **prácticas** de forma progresiva.

Lecturas recomendadas

Alejandro Calderón Mateos

Base

1. Carretero 2020:
   1. Cap. 6
2. Carretero 2007:
   1. Cap. 9.1-9.5,
   2. Cap. 9.8-9.10 y 9.12

Recomendada

1. Tanenbaum 2006:
   1. (es) Cap. 6
   2. (en) Cap. 6
2. Stallings 2005:
   1. 12.1-12.8
3. Silberschatz 2006:
   1. 10.3-10.4,
   2. 11.1-11.6 y 13

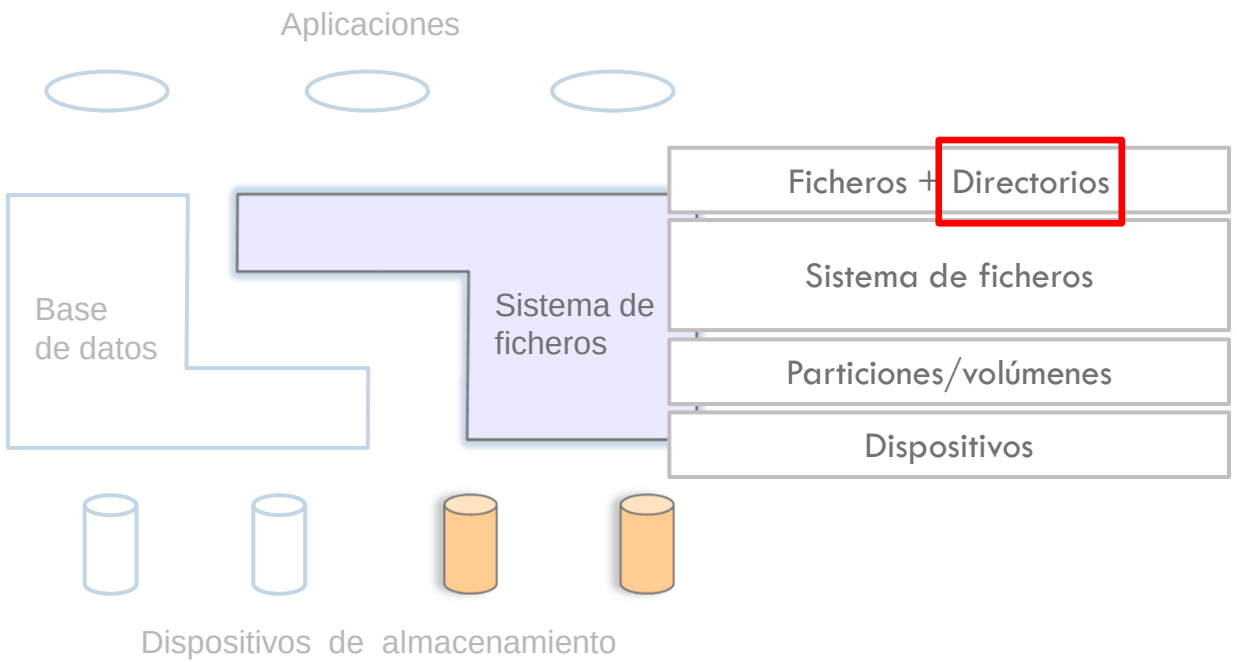
Contenidos

Alejandro Calderón Mateos

- Introducción
- Fichero
- **Directorio**
  - Metadatos
  - Interfaz
- Sistema de ficheros
- Particiones/Volúmenes
- Dispositivos
- Software de sistema
- Sistema de ficheros (gestor)

Directorio (carpetas)

Alejandro Calderón Mateos

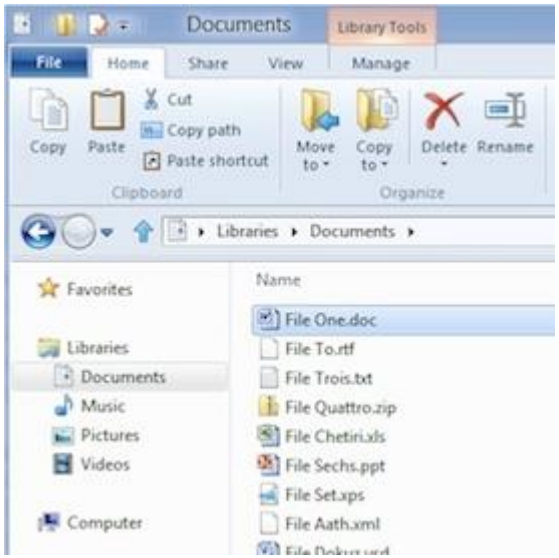


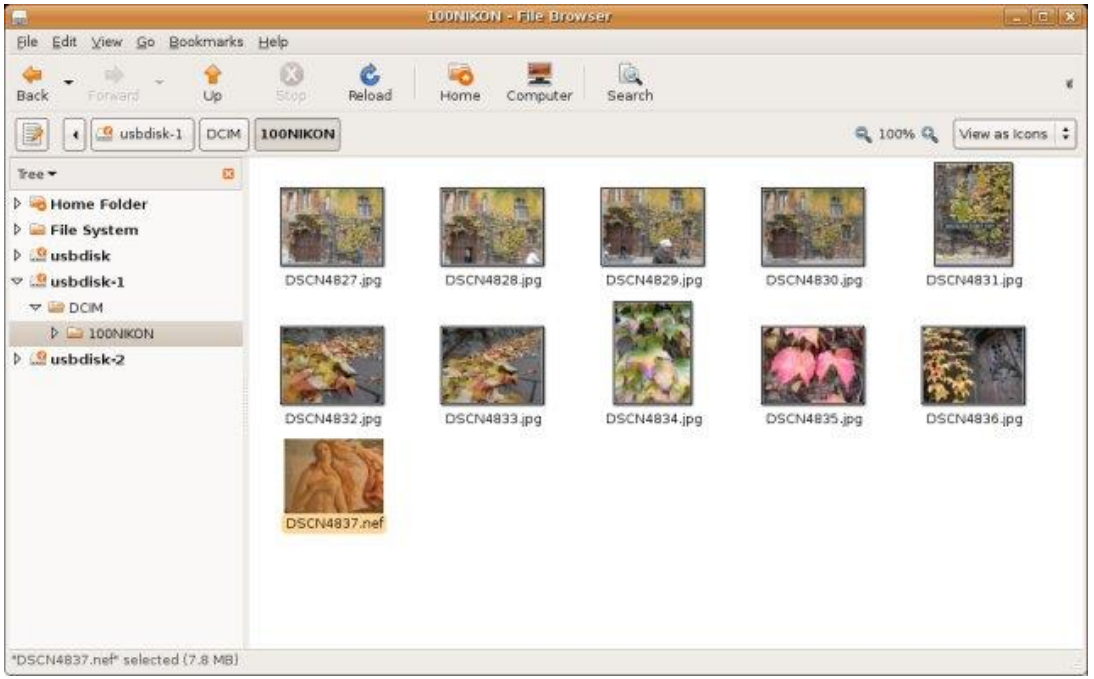
Directorio (carpetas)

importante

Alejandro Calderón Mateos

- Estructura de datos que permite agrupar un conjunto de ficheros según el criterio del usuario.





Directorio (carpetas): objetivos

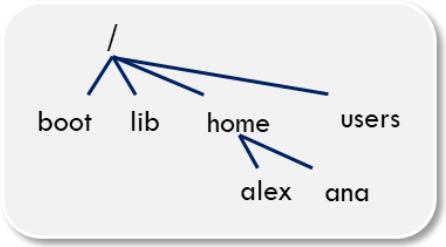
importante

- Estructura de datos que permite agrupar un conjunto de ficheros según el criterio del usuario.
  - Eficiencia de búsqueda: localizar un fichero rápidamente.
  - Agrupación: agrupación lógica de ficheros según sus propiedades
    - Por ejemplo: programas C11, juegos, etc.
  - Nombrado: conveniente y sencillo para los usuarios/as:
    - Nombres de longitud variable.
    - Dos usuarios/as pueden usar el mismo nombre para ficheros distintos.
    - Los mismos ficheros pueden tener nombres distintos.
  - Estructurado: operaciones claramente definidas y ocultación
    - C.R.U.D.: mkdir <d>, ls <d>, mv <d> <c>, rmdir <c>
    - cd <d>, cd .., rm <f>, rm –fr <d>
  - Sencillez: la entrada de directorio debe ser lo más sencilla posible.

Directorios: nombres jerárquicos

importante

- Nombres jerárquicos para la identificación:
  - Lista de nombres hasta llegar al directorio/fichero.
  - Los nombres se separan con un carácter especial:
    - / en LINUX y \ en Windows



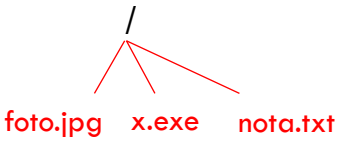
- Nombres especiales de directorio:
  - . Directorio actual o directorio de trabajo (Ej.: cp /home/alex/correo.txt .)
  - . . Directorio padre o directorio anterior (Ej.: ls ..)
  - ~ Directorio base del usuario+a en UNIX (Ej.: ls –las ~ ; ls –las $HOME)
  - / Directorio raíz en UNIX (Ej.: ls –las /)
- Dos tipos de nombrado usado:
  - **Nombre absoluto** o completo (empieza por el directorio raíz)
    - `/usr/include/stdio.h` (linux)
    - `c:\usr\include\stdio.h` (windows)
  - **Nombre relativo** (es relativo al directorio actual, no empieza por raíz)
    - `stdio.h` asumiendo que `/usr/include` es el directorio actual.
    - `../include/stdio.h`

Directorios: organización

importante

Alejandro Calderón Mateos

- Organizan y proporcionan información sobre la estructuración de los sistemas de archivos:



  - De un nivel
    - **1** dir con **n** ficheros
    - **1** fichero con **1** dir.

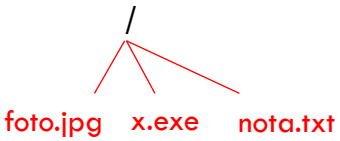
- Un nivel:
  - Único directorio para todos/as los/as usuarios/as.
  - [I] alta probabilidad de coincidencia en nombres de ficheros.
- Dos niveles:
  - Primer nivel con un directorio por usuario/a.
  - [V] mismo nombre fichero para distintos/as usuarios/as pero [I] no problemas de agrupación.

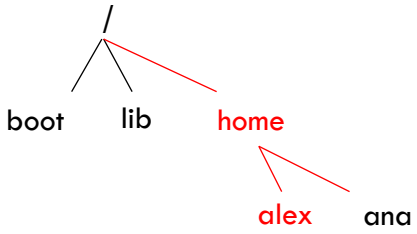
Directorios: organización

importante

Alejandro Calderón Mateos

- Organizan y proporcionan información sobre la estructuración de los sistemas de archivos:





- De un nivel
  - **1** dir con **n** ficheros
  - **1** fichero con **1** dir.
- Jerárquico (árbol)
  - **1** dir con **n** entradas
  - **1** entrada con **1** dir.

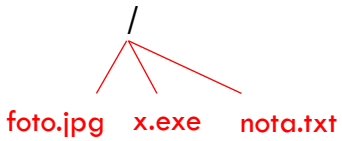
- Jerárquico o en árbol:
  - [V] Jerarquía y agrupación.
  - [V] Búsqueda eficiente.
  - Nombres absolutos y nombres relativos (directorio trabajo)

Directorios: organización

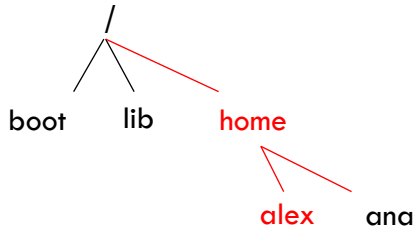
importante

Alejandro Calderón Mateos

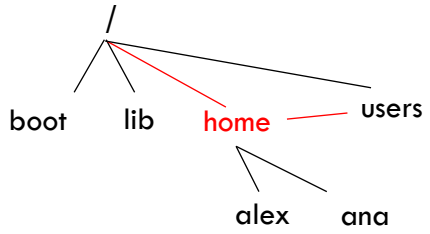
- Organizan y proporcionan información sobre la estructuración de los sistemas de archivos:



- De un nivel
  - **1** dir con **n** ficheros
  - **1** fichero con **1** dir.



- Jerárquico (árbol)
  - **1** dir con **n** entradas
  - **1** entrada con **1** dir.



- Árbol a-cíclico
  - **1** dir. con **n** entradas
  - **1** entrada con **n** dir.

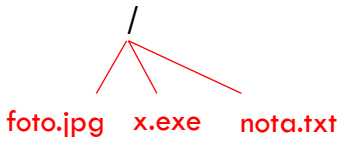
- Grafo acíclico:
  - Añade al jerárquico la posible de que dos directorios compartan ficheros y/o subdirectorios
  - Uso del concepto de enlace.
    - Linux: a) enlace simbólico/blando y b) enlace duro.
    - Windows: a) en UI con accesos directos y *symlinks* y b) *junctions* (uso de *NTFS reparse points*)

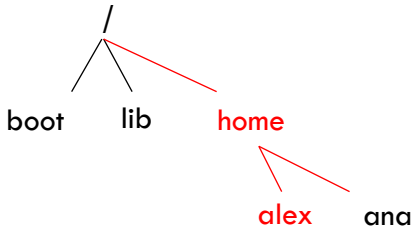
Directorios: organización
resumen

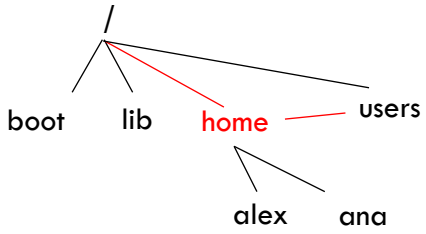
importante

Alejandro Calderón Mateos

- Organizan y proporcionan información sobre la estructuración de los sistemas de archivos:







De un nivel

- **1** dir con **n** ficheros
- **1** fichero con **1** dir.

- Único directorio para todos/as los/as usuarios/as.
- [I] alta probabilidad de coincidencia en nombres.

Jerárquico (árbol)

- **1** dir con **n** entradas
- **1** entrada con **1** dir.

- Jerarquía y agrupación.
  - Búsqueda eficiente.
- Nombres absolutos.
- Nombres relativos:
  - Directorio trabajo

Árbol a-cíclico

- **1** dir. con **n** entradas
- **1** entrada con **n** dir.

- Posible compartir fich. y subdirs.
- Uso del concepto de enlace.
  - Importante: evitar bucles.
- Físicos/Duros o blando/simbólico:
  - [I] Físicos dentro el mismo sist. fichs.
  - [V] Borrar físico decrementa contador y solo al llegar a 0 se borra.
  - [I] No físico a directorio.

Contenidos

Alejandro Calderón Mateos

- Introducción
- Fichero
- **Directorio**
  - **Metadatos**
  - Interfaz
- Sistema de ficheros
- Particiones/Volúmenes
- Dispositivos
- Software de sistema
- Sistema de ficheros (gestor)

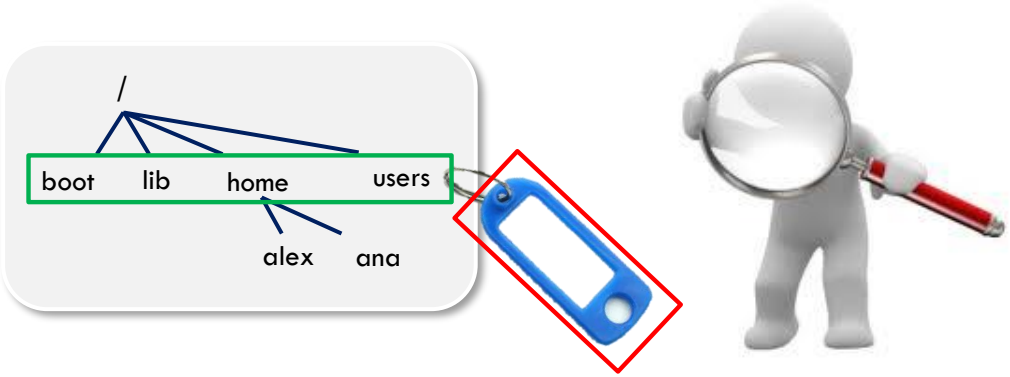
Directorio (carpetas)

Alejandro Calderón Mateos

- Información de un directorio:
  - Datos
    - **"fichero especial" cuyo contenido es un listado con los entradas que contiene.**

fichero | directorio

  - Metadatos
    - **Información sobre el directorio en sí.**
    - Distintos **atributos** sobre el directorio (+ información usada por el S.O.)



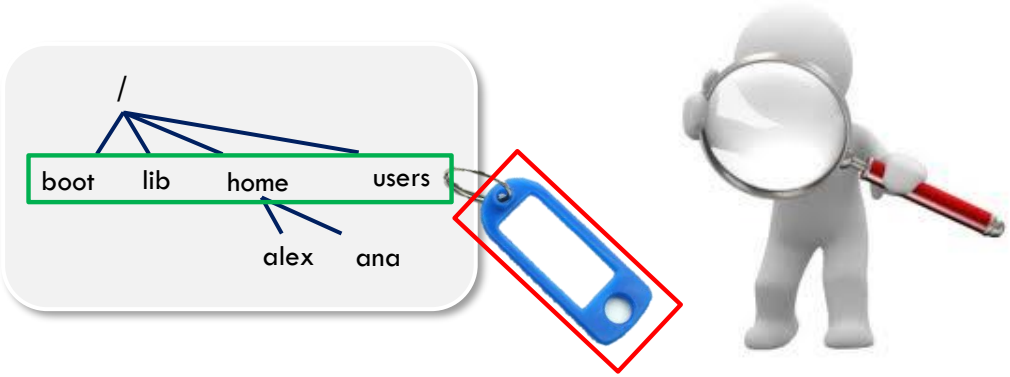
Directorio (carpetas)

Alejandro Calderón Mateos

- Información de un directorio:
  - Datos
    - **"fichero especial" cuyo contenido es un listado con los entradas que contiene.**

fichero | directorio

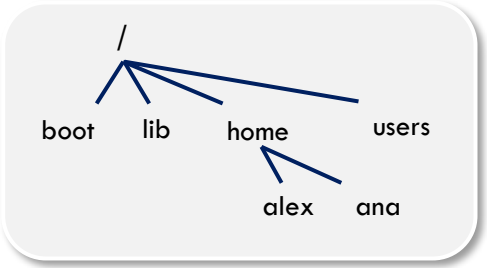
  - Metadatos
    - **Información sobre el directorio en sí.**
    - Distintos **atributos** sobre el directorio (+ información usada por el S.O.)



Directorios: atributos

importante

- Atributos típicos de un directorio:
  - Nombre: identificador para los usuarios del directorio.
  - Tamaño: número de ficheros en el directorio.
  - Protección: control de qué usuario puede leer, acceder, etc.
  - Día y hora: instante de tiempo de último acceso, de creación, etc. que permite la monitorización del uso del directorio.
  - Identificación de usuario: identificador del creador, etc.



Contenidos

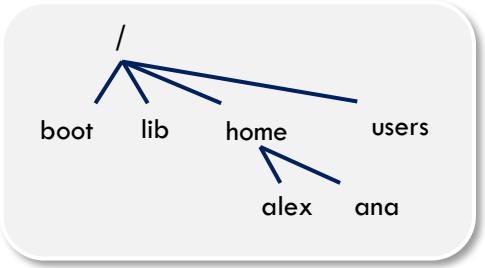
Alejandro Calderón Mateos

- Introducción
- Fichero
- **Directorio**
  - Metadatos
  - **Interfaz**
- Sistema de ficheros
- Particiones/Volúmenes
- Dispositivos
- Software de sistema
- Sistema de ficheros (gestor)

Directorios: interfaz

importante

- Interfaz genérica para gestión de directorios:
  - mkdir (nombre,modo)
  - rmdir (nombre)
  - chdir (nombre)
  - getcwd (nombre, tamaño_nombre)
  - descriptor ← opendir (nombre)
  - closedir (descriptor)
  - estructura ← readdir (descriptor)
  - rewindir (descriptor)
  - unlink (nombre)
  - rename (antiguo_nombre, nuevo_nombre)



Ejemplo: listar entradas de /tmp

Alejandro Calderón Mateos

lectura de /tmp

```
#include <unistd.h>
#include <sys/types.h>
#include <dirent.h>
#include <stdio.h>

int main ( int argc, char *argv[] )
{
  DIR *dir1 ;
  struct dirent *dp ;
  char nombre[256] ;
  int ret ;

  ret = chdir ("/tmp/") ;
  if (ret < 0) exit(-1) ;

  getcwd (nombre, 256);
  printf("%s\n",nombre);

  dir1 = opendir (nombre);
  if (NULL == dir1) exit(-1) ;
  while ( (dp = readdir (dir1)) != NULL) {
     printf("%/%s\n",nombre,dp->d_name);
  }
  closedir (dir1);

  return (0) ;
}
```

Cambiar de directorio de trabajo

Imprimir el directorio actual de trabajo

Abrir un directorio para trabajar con él

Leer entradas del directorio e imprimir el nombre de cada entrada

Cerrar el directorio de trabajo

Ejemplo: ¿es fichero o directorio?

importante

Alejandro Calderón Mateos

lectura de argv[1]

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <stdio.h>

int main ( int argc, char *argv[] )
{
  DIR *dir1 ;
  struct dirent *dp ;
  struct stat s ;

  dir1 = opendir (argv[1]);
  if (NULL == dir1) {
     perror("opendir:");
     return (-1);
  }

  while ( (dp = readdir (dir1)) != NULL) {
      stat(dp->d_name,&s);
      if (S_ISDIR(s.st_mode))
           printf("dir: %s\n",dp->d_name);
      else printf("fch: %s\n",dp->d_name);
  }

  closedir (dir1);
  return (0) ;
}
```

Abrir un directorio para trabajar con él

Leer entradas del directorio…

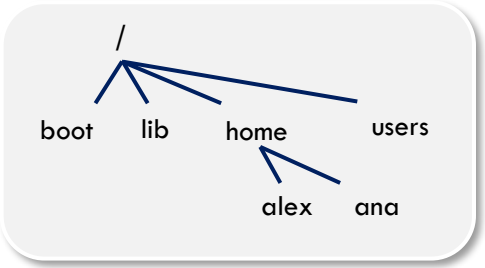
…para cada entrada obtener los metadatos de la misma e imprimir si es fichero o directorio junto con el nombre de la entrada

Cerrar el directorio de trabajo

Directorios: interfaz

Alejandro Calderón Mateos

- Interfaz genérica para gestión de directorios:
  - mkdir (nombre,modo)
  - rmdir (nombre)
  - chdir (nombre)
  - getcwd (nombre, tamaño_nombre)
  - descriptor ← opendir (nombre)
  - closedir (descriptor)
  - estructura ← readdir (descriptor)
  - rewindir (descriptor)
  - unlink (nombre)
  - rename (antiguo_nombre, nuevo_nombre)



OPENDIR – Abrir un directorio

Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`DIR *opendir ( char *dirname );`

Argumentos	▫ `dirname` puntero al nombre del directorio
Devuelve	Un puntero para utilizarse en `readdir()`, `closedir()` , etc. o `NULL` si hubo error.
Descripción	▫ Abre una sesion de trabajo con un directorio de manera que pueda trabajarse con las entradas del mismo. ▫ Se coloca en la primera entrada.

READDIR – Lectura de entradas de directorio

Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`struct dirent *`**`readdir`**` ( DIR *dirp );`

Argumentos	▫ `dirp` puntero retornado por `opendir()`.
Devuelve	Un puntero a una estructura de tipo `struct dirent` que representa una entrada de directorio o `NULL` si hubo error.
Descripción	▫ Devuelve la siguiente entrada del directorio asociado a `dirp`. ▫ Avanza el puntero a la siguiente entrada. ▫ La estructura es dependiente de la implementación. Debería asumirse que tan solo se obtiene un miembro: `char *d_name`.

REWINDDIR – Posicionar a la 1ª entrada

Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`void *rewinddir ( DIR *dirp );`

Argumentos	▫ `dirp` puntero retornado por `opendir()`.
Devuelve	Nada.
Descripción	▫ Sitúa el puntero de posición dentro del directorio en la primera entrada.

CLOSEDIR – Abrir un directorio

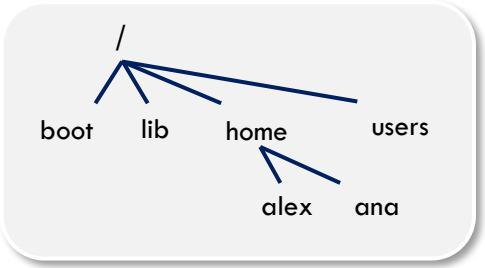
Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`int *closedir ( DIR *dirp );`

Argumentos	▫ `dirp` puntero devuelto por `opendir()`
Devuelve	Cero si todo bien o -1 si hubo error.
Descripción	▫ Cierra la sesion de trabajo con el directorio.

Directorios: interfaz

Alejandro Calderón Mateos

- Interfaz genérica para gestión de directorios:
  - mkdir (nombre,modo)
  - rmdir (nombre)
  - chdir (nombre)
  - getcwd (nombre, tamaño_nombre)
  - descriptor ← opendir (nombre)
  - closedir (descriptor)
  - estructura ← readdir (descriptor)
  - rewindir (descriptor)
  - unlink (nombre)
  - rename (antiguo_nombre, nuevo_nombre)



MKDIR – Crear un directorio

Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`int mkdir ( const char *name, mode_t mode );`

Argumentos	`name` nombre del directorio. `mode` bits de protección.
Devuelve	Cero si todo bien o -1 si hubo error.
Descripción	Crea un directorio de nombre `name`. UID_dueño = UID_efectivo GID_dueño = GID_efectivo

RMDIR – Borra un directorio

Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`int rmdir ( const char *name );`

Argumentos	`name` nombre del directorio.
Devuelve	Cero si todo bien o -1 si hubo error.
Descripción	Borra el directorio si está vacío. Si el directorio no está vacío no se borra.

CHDIR – Cambia el directorio actual

Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`int chdir ( const char *name );`

Argumentos	`name` nombre del directorio.
Devuelve	Cero si todo bien o -1 si hubo error.
Descripción	Modifica el directorio actual de trabajo. Los nombres relativos se forman a partir del directorio actual.

GETCWD – Obtener el directorio actual

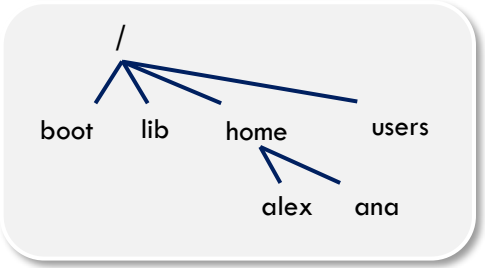
Servicio	`#include <sys/types.h>`
	`#include <dirent.h>`
	
	`char *getcwd ( char *buf, size_t bufSize );`

Argumentos	▫ `buf` puntero al espacio donde guardar el nombre del directorio actual de trabajo. ▫ `bufMaxSize` tamaño en bytes del espacio.
Devuelve	Puntero a `buf` relleno o `NULL` si error.
Descripción	▫ Obtiene el nombre del directorio actual de trabajo, lo guarda en `buf` y devuelve `buf` (donde se ha guardado). ▫ Si el nombre es mayor que el tamaño de `buf` entonces puede que se trunque dicho nombre (por la limitación de espacio).

Directorios: interfaz

Alejandro Calderón Mateos

- Interfaz genérica para gestión de directorios:
  - mkdir (nombre,modo)
  - rmdir (nombre)
  - chdir (nombre)
  - getcwd (nombre, tamaño_nombre)
  - descriptor ← opendir (nombre)
  - closedir (descriptor)
  - estructura ← readdir (descriptor)
  - rewindir (descriptor)
  - unlink (nombre)
  - rename (antiguo_nombre, nuevo_nombre)



RENAME – Cambiar el nombre de un fichero

Servicio	`#include <unistd.h>`
	
	`int rename ( char *old, char *new );`

Argumentos	▫ `old` puntero al array de caracteres que contiene el nombre actual del fichero a cambiar. ▫ `new` puntero al array de caracteres que contiene el nuevo nombre del fichero.
Devuelve	Cero ó -1 si error.
Descripción	▫ Cambia el nombre del archivo que actualmente es `old` por el nuevo nombre que será `new`.

UNLINK – Eliminar entrada de directorio

Servicio	`#include <unistd.h>`
	
	`int unlink ( char *name );`

Argumentos	▫ `name` puntero al array de caracteres que contiene el nombre de la entrada de directorio a intentar borrar.
Devuelve	Cero ó -1 si error.
Descripción	▫ Elimina la entrada de directorio y decrementa el número de enlaces del archivo correspondiente. ▫ Cuando el número de enlaces es igual a cero y ningún proceso lo mantiene abierto, se libera el espacio ocupado por el archivo y el archivo deja de ser accesible. Si algún proceso lo mantiene abierto entonces se espera hasta que lo cierre para liberar el espacio.

LINK – Crear una entrada de directorio

Servicio	`#include <unistd.h>`
	
	`int link ( const char *actual, const char *new );`

Argumentos	▫ `actual` puntero al array de caracteres que contiene el nombre de la entrada de directorio existente con la que trabajar. ▫ `new` puntero al array de caracteres que contiene el nombre de la nueva entrada de directorio que enlazará con la `actual`.
Devuelve	Cero ó -1 si error.
Descripción	▫ Crea un nuevo enlace, físico o simbólico, para un archivo existente. ▫ El sistema no registra cuál es el enlace original. ▫ `actual` no debe ser el nombre de un directorio salvo que: a) se tenga privilegio suficiente y b) la implementación soporte el enlace de directorios

SYMLINK – Creación de enlace blando

Servicio	`#include <unistd.h>`
	
	`int symlink ( const char* oldpath,`
	`const char* newpath );`

Argumentos	▫ `oldpath` nombre del fichero existente a enlazar. ▫ `newpath` nombre del enlace blando a crear.
Devuelve	Devuelve 0 si todo fue bien ó -1 si error.
Descripción	▫ Crea un enlace blando o simbólico de un entrada (archivo o directorio) existente. ▫ Puede enlazarse entradas de otra partición, pero si se borra se pierde el acceso al contenido.

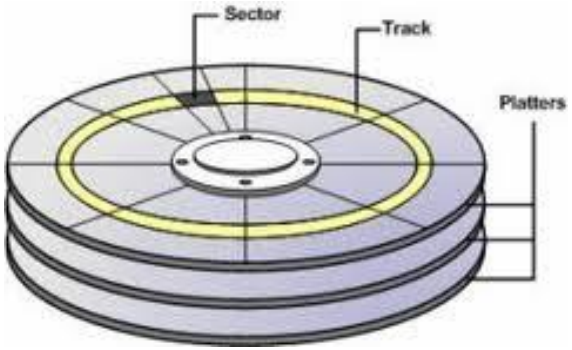
Contenidos

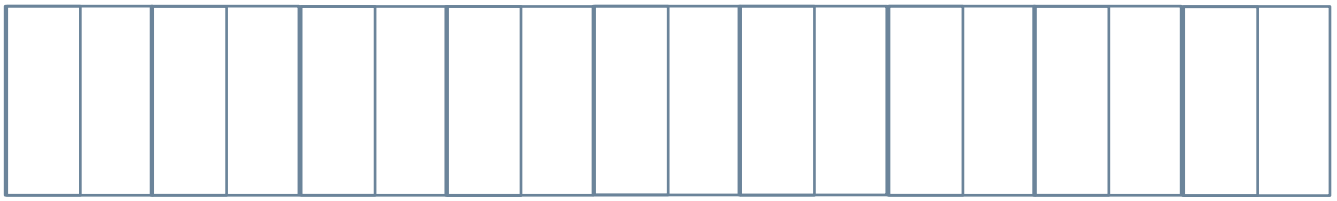
- Introducción
- Fichero
- Directorio
- **Sistema de ficheros**
- Particiones/Volúmenes
- Dispositivos
- Software de sistema
- Sistema de ficheros (gestor)

Sectores

importante

- El dispositivo de almacenamiento se divide en **sectores**, pistas y cilindros.

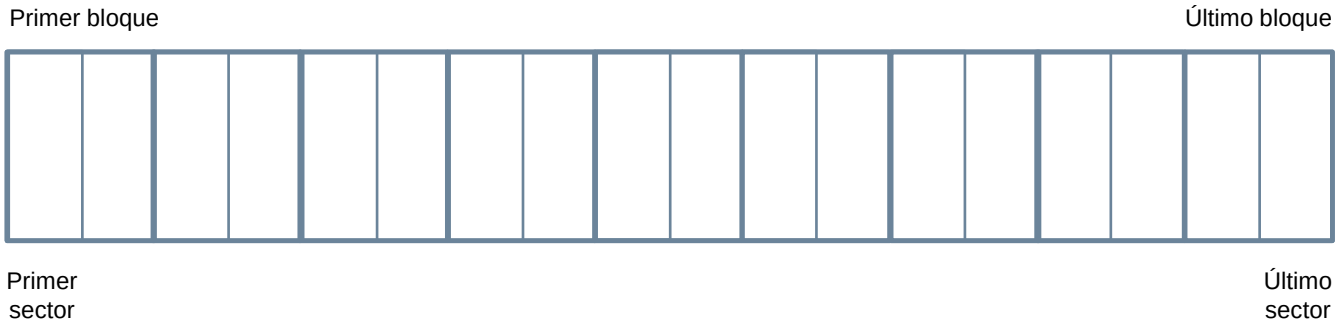




Bloques

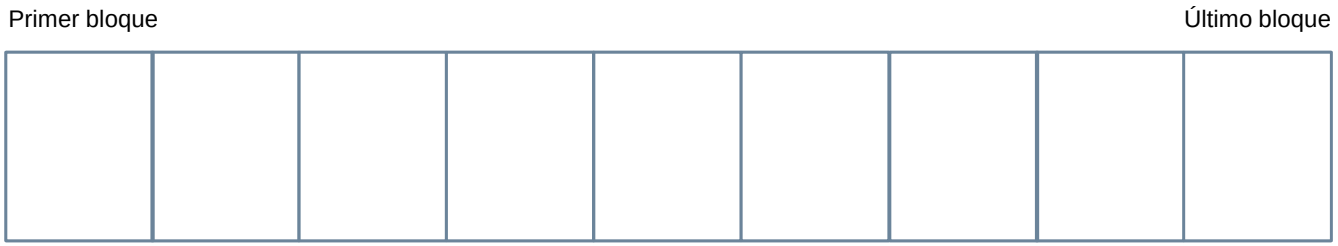
importante

- **Bloque**: *agrupación lógica de sectores de disco (2^n sectores)*
  - Es la unidad de transferencia mínima usado por el S.O.
  - Optimizar la eficiencia de la entrada/salida de los dispositivos.
  - Los usuarios pueden definir el tamaño de bloque al crear el sistema de ficheros, o usar el ofrecido por defecto en el S.O.

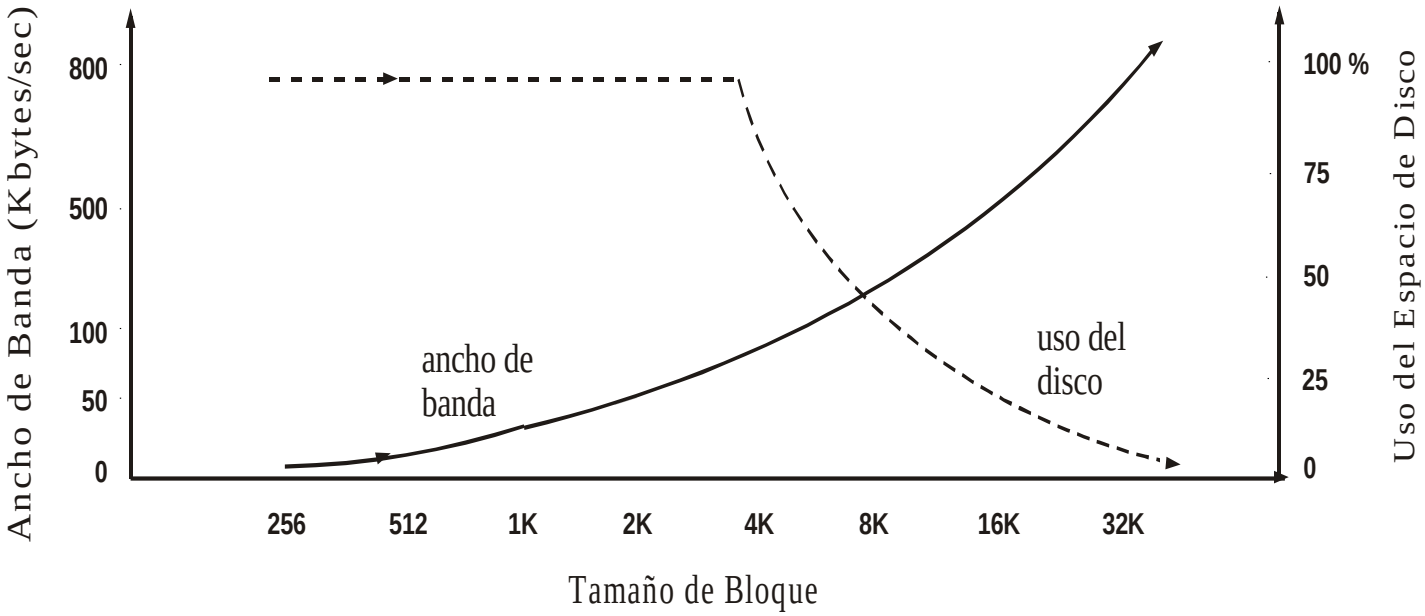


Bloques

- **Bloque**: *agrupación lógica de sectores de disco (2^n sectores)*
  - Es la unidad de transferencia mínima usado por el S.O.
  - Optimizar la eficiencia de la entrada/salida de los dispositivos.
  - Los usuarios pueden definir el tamaño de bloque al crear el sistema de ficheros, o usar el ofrecido por defecto en el S.O.



Tamaño de bloque

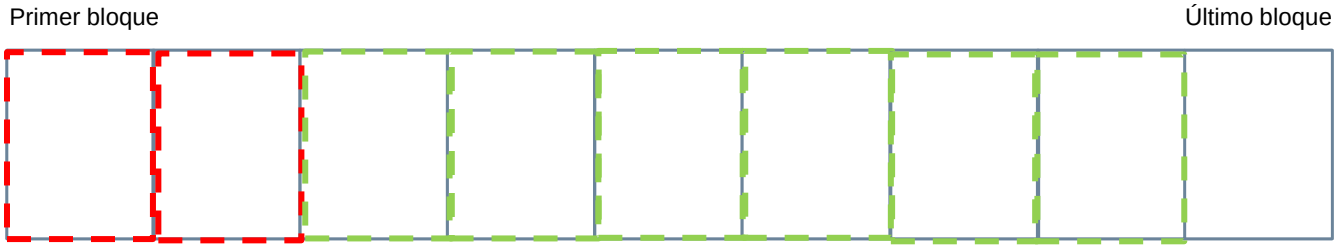


- La elección del tamaño del bloque es importante para balancear:
  - Ancho de banda: mayor número de sectores inicialmente, mejor ancho de banda
  - Uso del disco: menor número de sectores, menos fragmentación interna

Sistema de ficheros

importante

- El sistema de archivos permite organizar la información en los dispositivos de almacenamiento en un formato inteligible para el sistema operativo:
  - Permite definir una unidad de asignación (bloque/agrupación).
  - Permite la gestión del espacio libre y ocupado (asignación de espacio en disco a cada fichero).



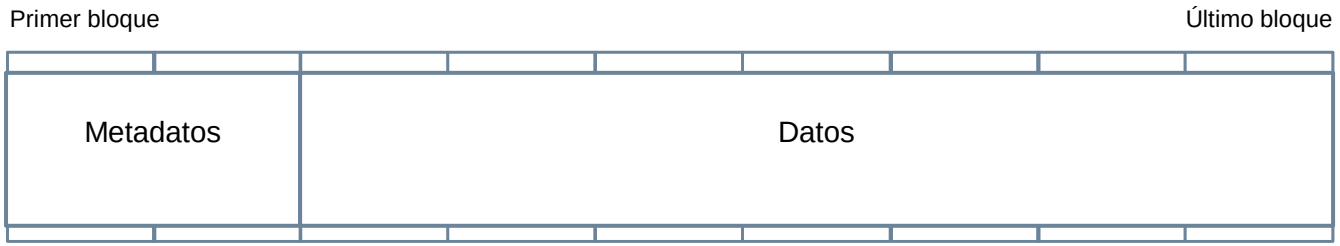
Sistema de ficheros

- El sistema de archivos permite organizar la información en los dispositivos de almacenamiento en un formato inteligible para el sistema operativo:
  - Permite definir una unidad de asignación.
    - ¿**Qué** unidad de asignación se utiliza?
  - Permite la gestión del espacio libre y ocupado (asignación de espacio en disco a cada fichero).
    - ¿**Qué** estructura de datos representa la asignación del fichero?
    - ¿**Se asigna** el espacio máximo en creación o de forma dinámica?
      - **Pre-asignación**: asignación de espacio máximo en creación.
      - **Dinámica**: asignación de espacio según se va necesitando.

Sistema de ficheros

importante

- El sistema de archivos permite organizar la información en los dispositivos de almacenamiento en un formato inteligible para el sistema operativo:
  - Es un *conjunto coherente de metainformación y datos*.



Sistema de ficheros: atributos

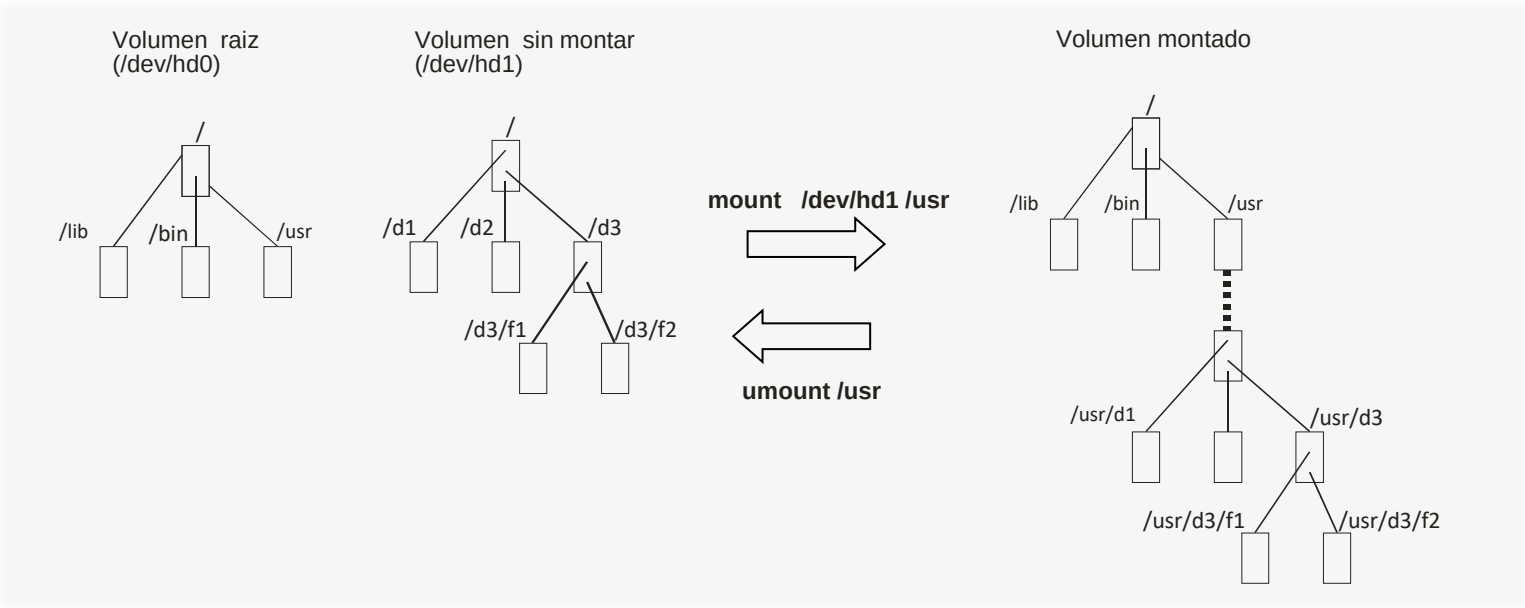
importante

Alejandro Calderón Mateos

- Atributos típicos de un sistema de fichero:
  - Tamaños usados:
    - Número de bloques: cantidad de bloques gestionados (datos + metadatos)
    - Tamaño de bloque: tamaño del bloque (en bytes o en sectores).
    - Número de entradas: número de entradas (ficheros y directorios) gestionados.
    - Tamaño de la zona de metadatos: número de bloques dedicados.
  - Gestión de espacio libre: identificación de qué bloque está libre.
  - Gestión de entradas: para cada entrada (fichero o directorio) se reserva un espacio para los metadatos que la describe:
    - Atributos generales: fechas, permisos, identificación de usuario, etc.
    - Atributos para la gestión de ocupado: bloques usados por esta entrada.
  - Referencia a la entrada del directorio raíz: identificación de la entrada que contiene la información del directorio raíz.

Sistema de ficheros: operaciones

- Operaciones con sistemas de ficheros:
  - Crear
  - Montar (en árbol único como Unix o en unidad d:, e: como ms-dos)
  - Desmontar



Sistema de ficheros

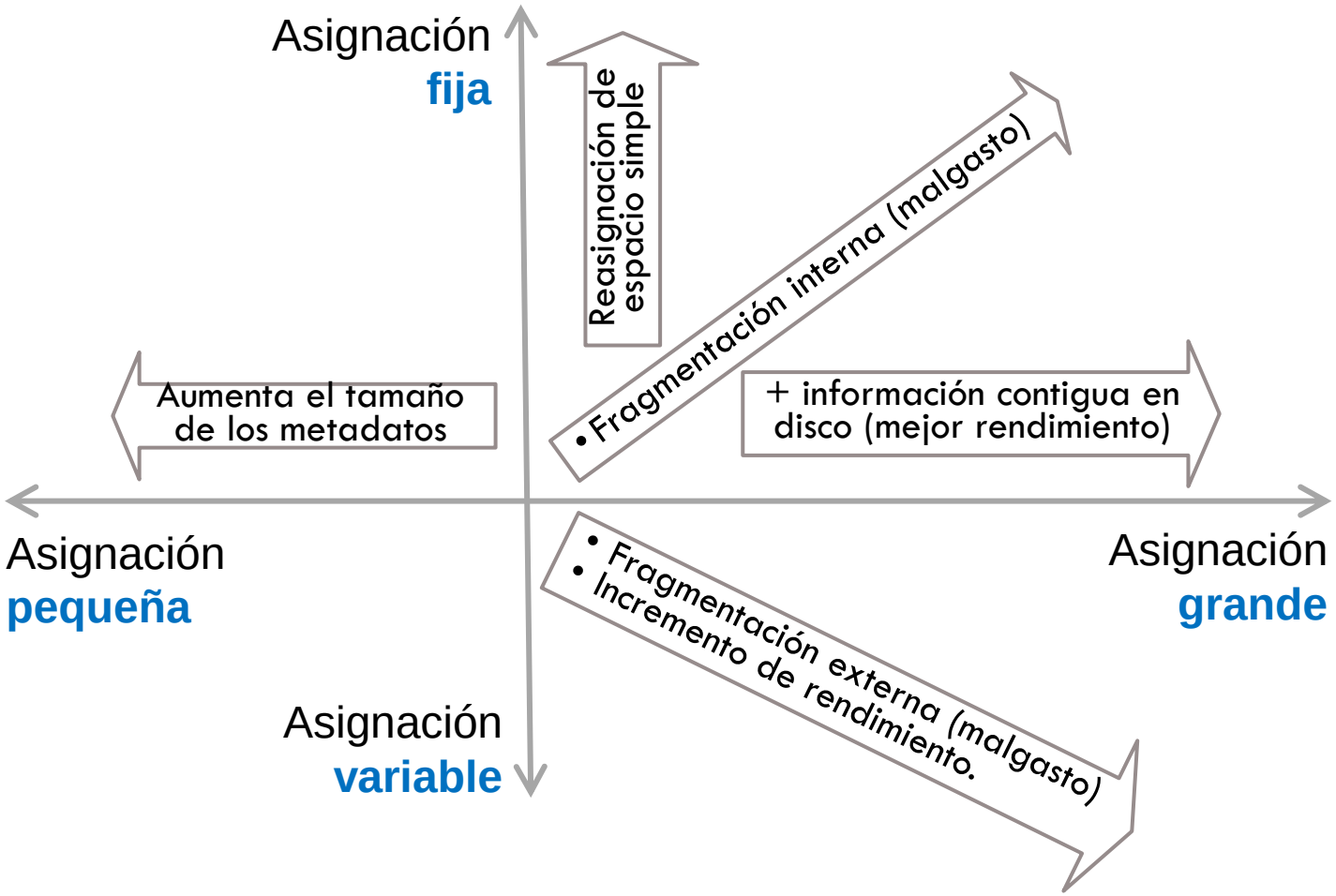
http://en.wikipedia.org/wiki/List_of_file_systems
Alejandro Calderón Mateos

- Gran cantidad de sistemas de ficheros.

- Para dispositivos de almacenamiento:
  - minix (Minix)
  - ext2 (Linux)
  - ext3 (Linux)
  - ufs (BSD)
  - fat (DOS)
  - vfat (win 95)
  - hpfs (OS/2)
  - hfs (Mac OS)
  - ntfs (win NT/2K/XP)
  - …
- Especiales:
  - procfs (/proc)
  - devFS (/dev)
  - umsdos (Unix sobre DOS)
  - …
- En red:
  - NFS
  - CODA
  - SMBFS
  - NCPFS (Novell)
  - …

Tamaño de asignación

importante



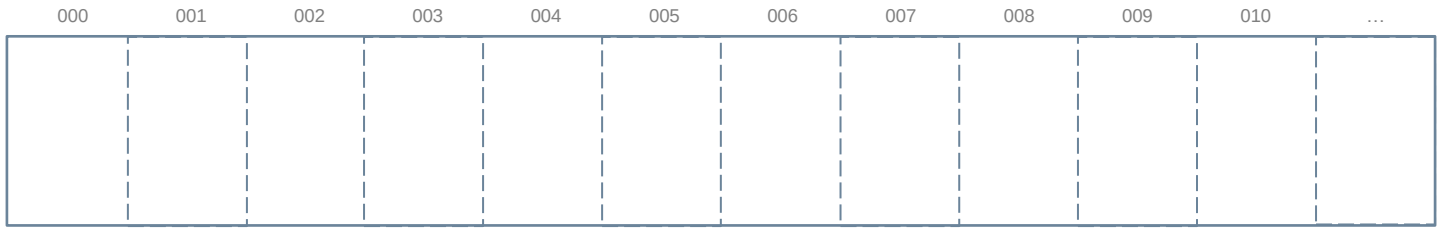
Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos

Disco lógico

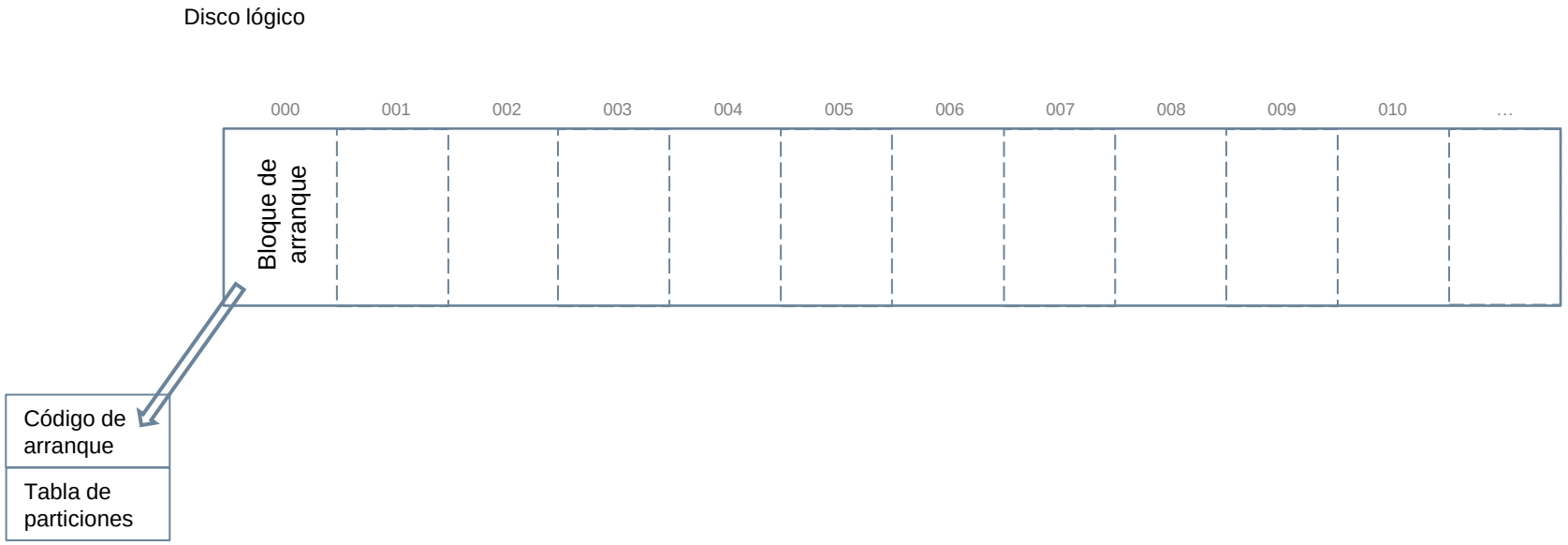


Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos

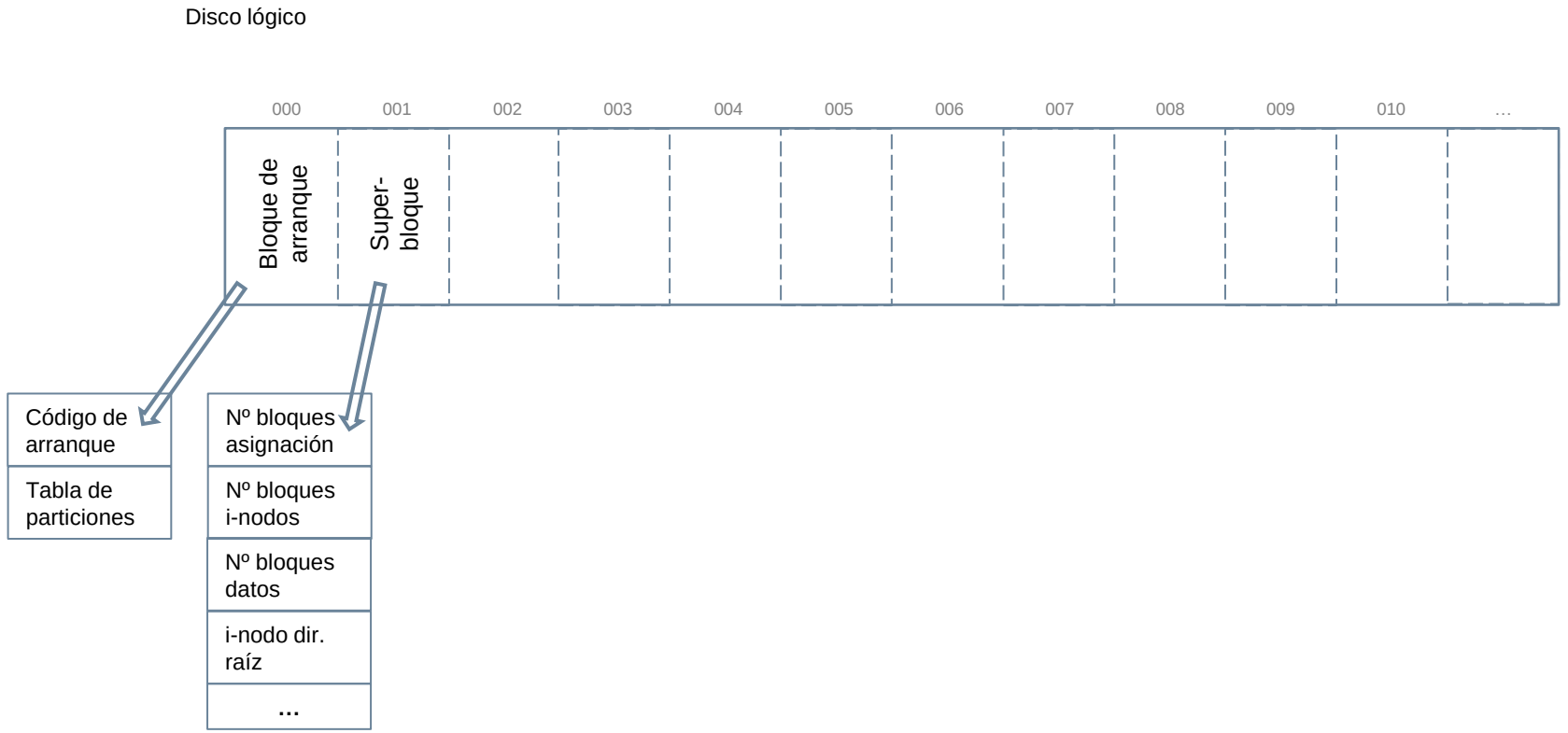


Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos

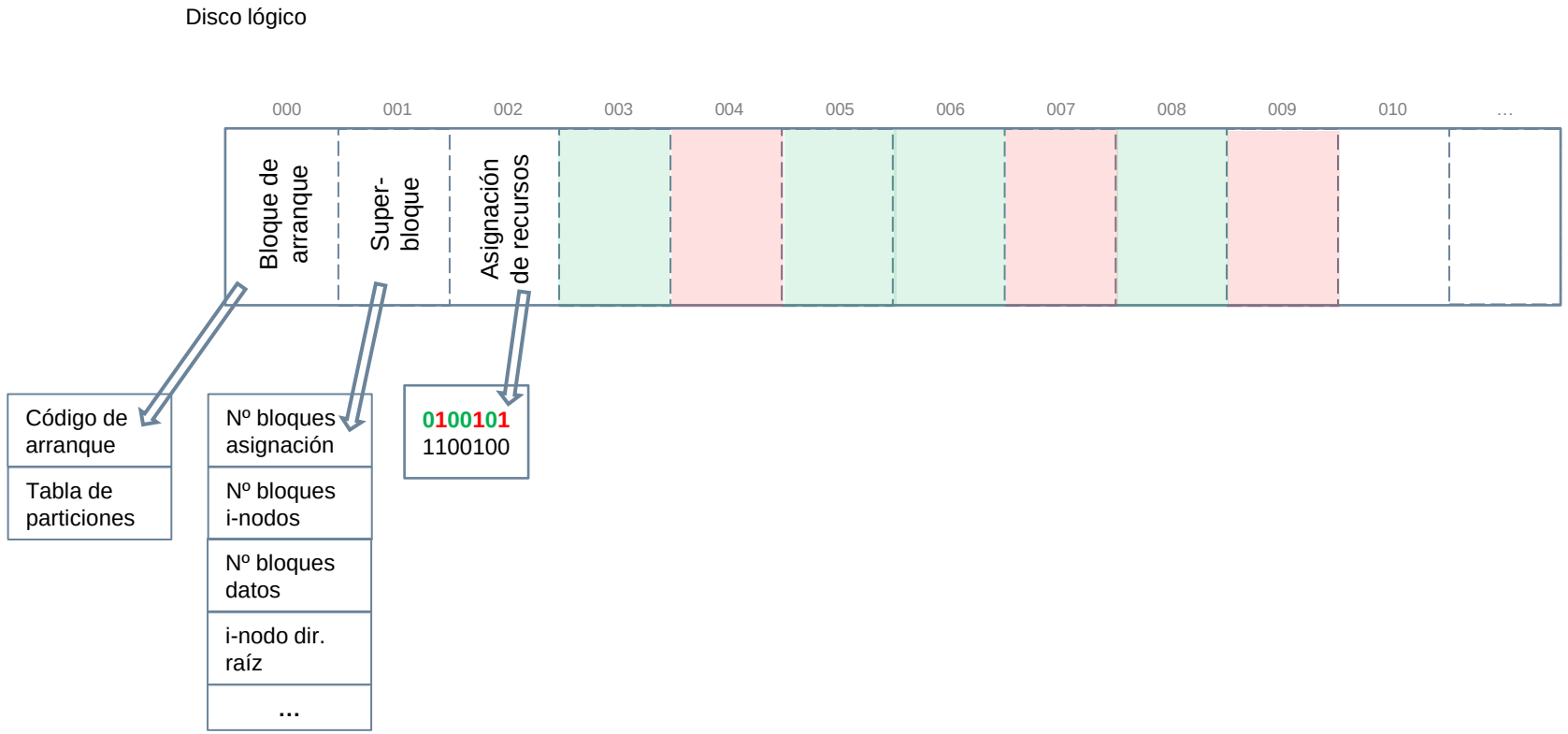


Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos

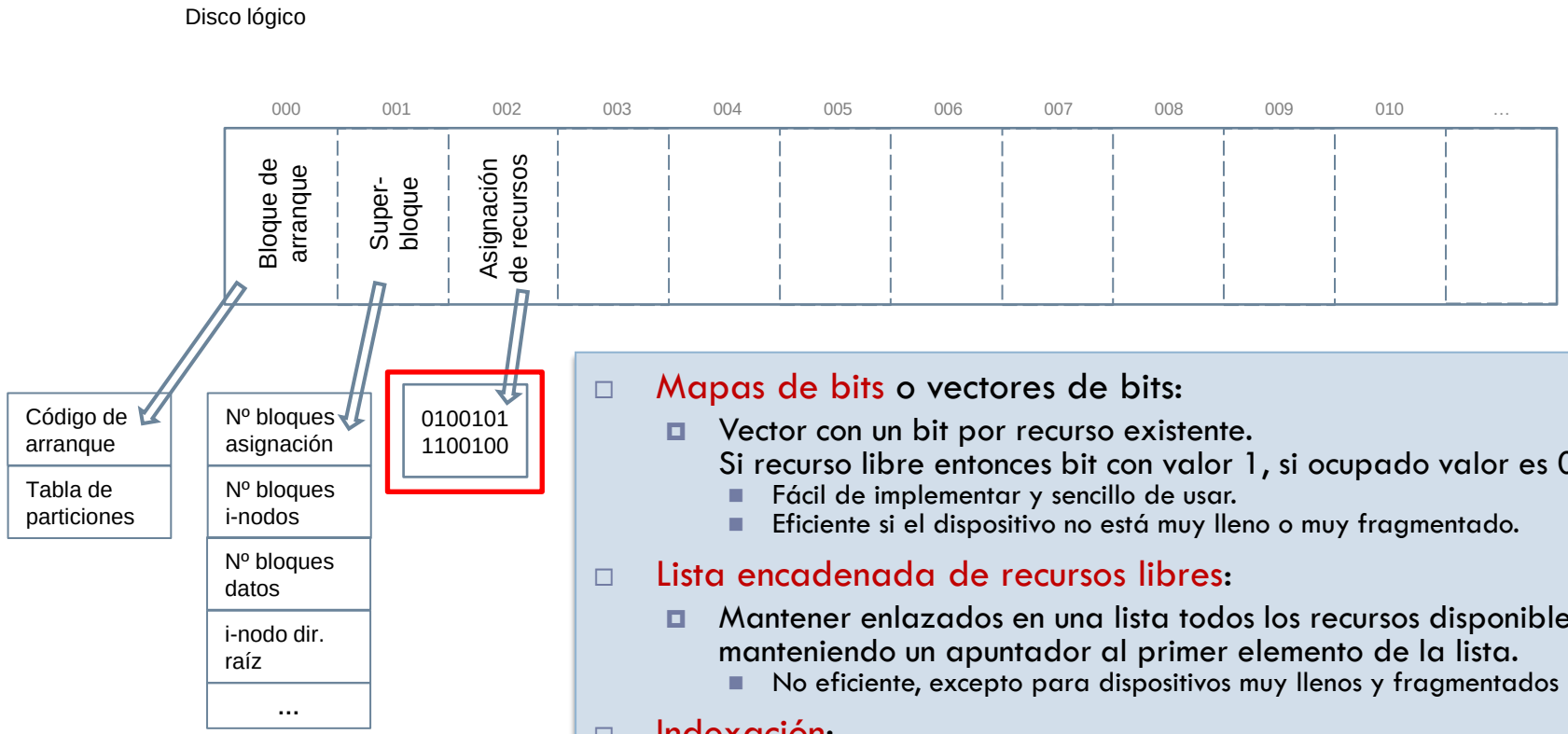


Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos



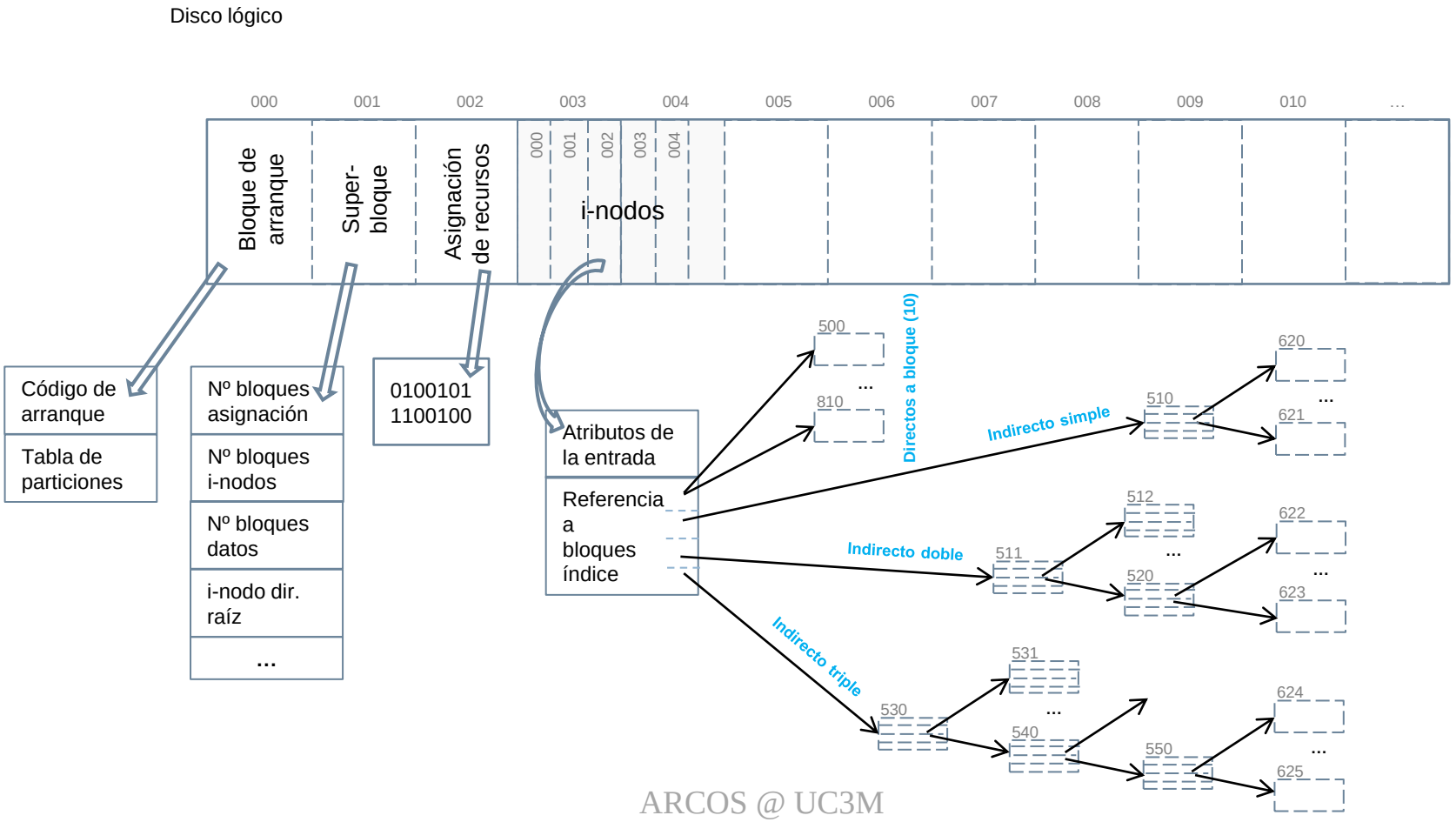
- Mapas de bits o vectores de bits:
  - Vector con un bit por recurso existente.
    Si recurso libre entonces bit con valor 1, si ocupado valor es 0.
    - Fácil de implementar y sencillo de usar.
    - Eficiente si el dispositivo no está muy lleno o muy fragmentado.
- Lista encadenada de recursos libres:
  - Mantener enlazados en una lista todos los recursos disponibles manteniendo un apuntador al primer elemento de la lista.
    - No eficiente, excepto para dispositivos muy llenos y fragmentados
- Indexación:
  - Tabla índice de porciones libres.

Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos

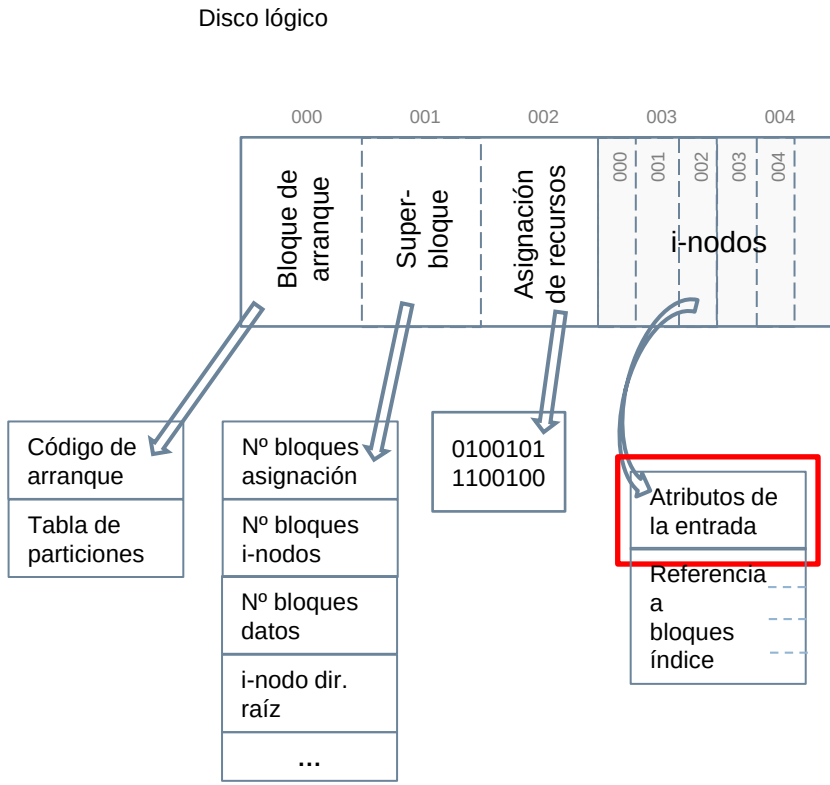


Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos



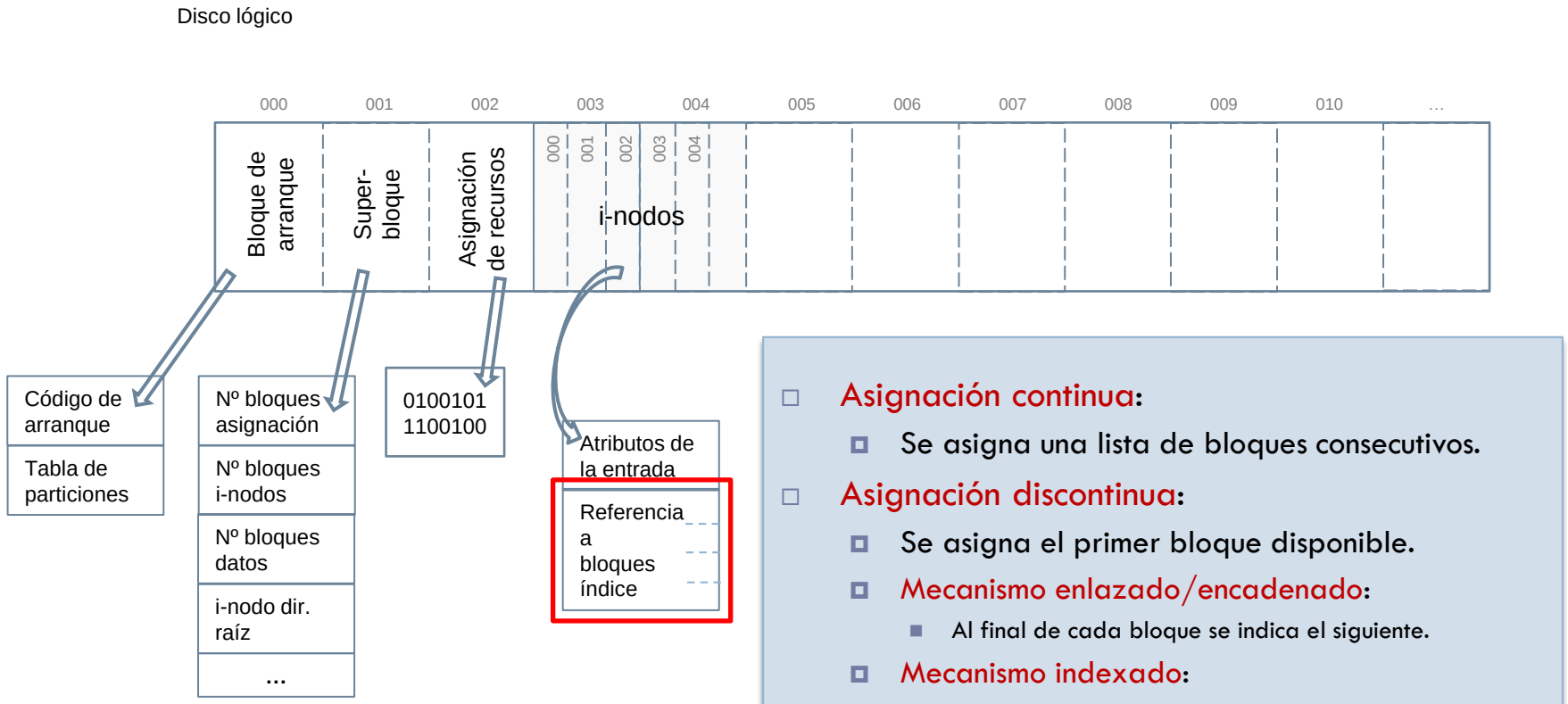
- Fechas y horas:
  - De creación, modificación, acceso, etc.
- Tamaño:
  - En bytes o bloques de disco usado.
- Usuario propietario y grupo
- Protección:
  - Atributos, ACL, capacidades, etc.
- Tipo de fichero
- Contador de enlaces
- Etc.

Sistema de ficheros:

representación usada en Minix

importante

Alejandro Calderón Mateos



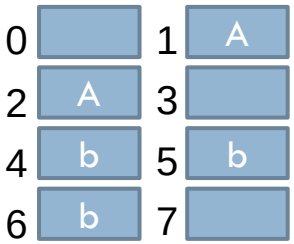
- Asignación continua:
  - Se asigna una lista de bloques consecutivos.
- Asignación discontinua:
  - Se asigna el primer bloque disponible.
  - Mecanismo enlazado/encadenado:
    - Al final de cada bloque se indica el siguiente.
  - Mecanismo indexado:
    - Bloques con índices de todos los bloques de la entrada.

Sistema de ficheros:

representación usada en Minix

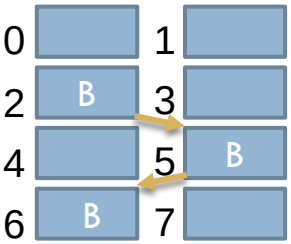
importante

Alejandro Calderón Mateos



F	I	L
A	1	2
B	4	3

- Asignación contigua:
  - Los bloques del ficheros están consecutivamente.
  - Precisa: primero (I) y nº de bloques (L)
  - Necesita compactar para optimizar.



F	I	L
B	2	3

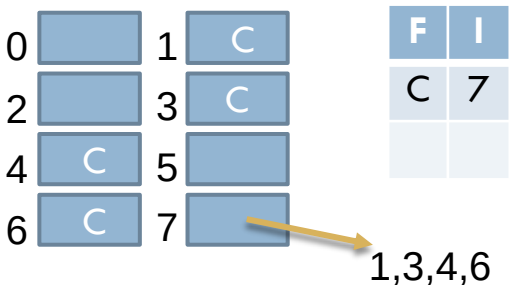
- Asignación encadenada:
  - Cada bloque contiene la referencia al siguiente bloque (bloque a bloque).
  - Precisa: primero (I) y nº de bloques (L)
  - Necesita consolidar para optimizar.

Sistema de ficheros:

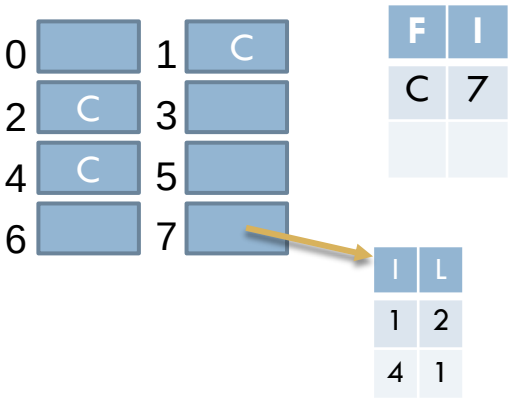
representación usada en Minix

importante

Alejandro Calderón Mateos



- Asignación indexada (bloques):
  - Se usa bloques con referencias a los bloques que contendrán los datos.
  - Precisa: id. del 1er bloque índice.
  - Desfragmentar.



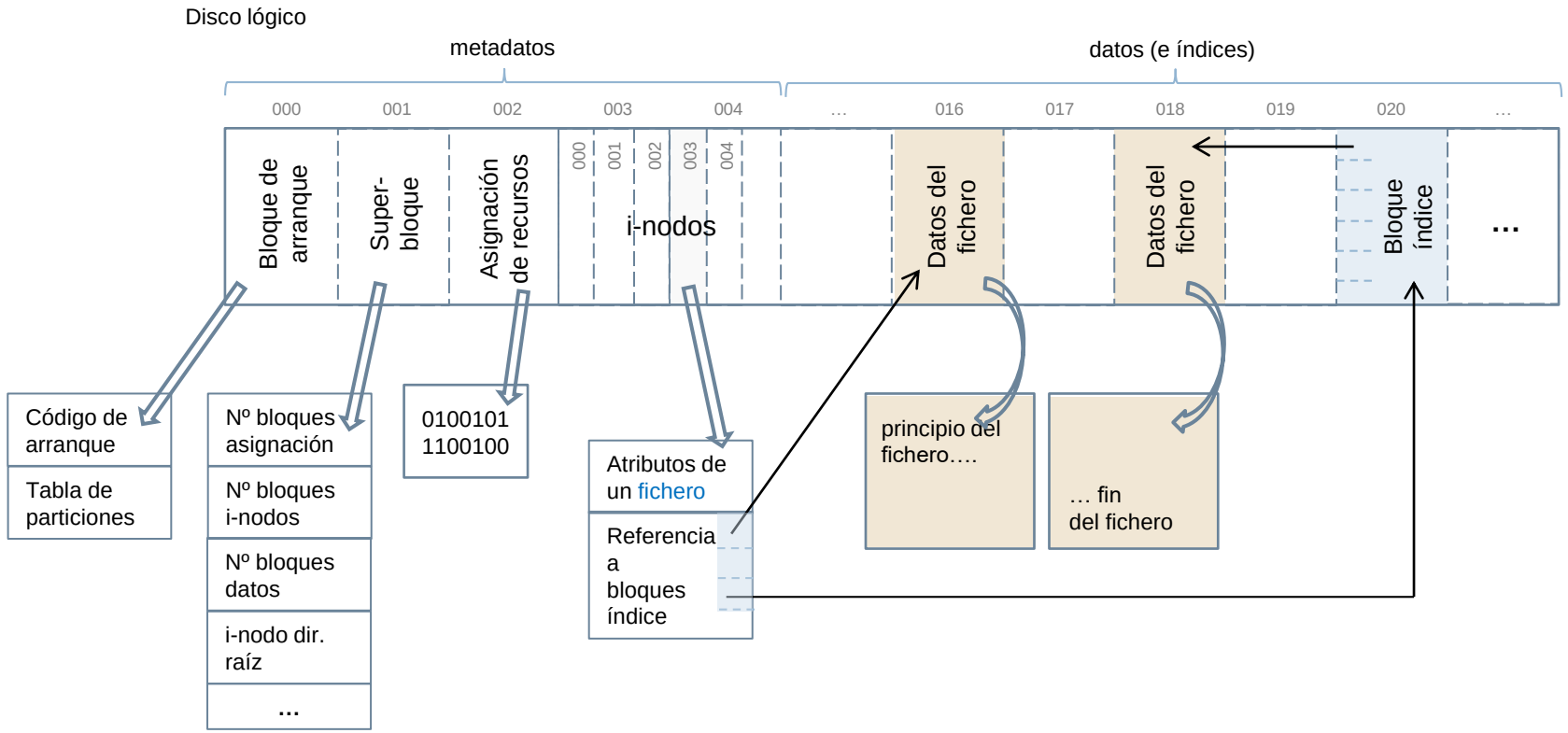
- Asignación indexada (extends):
  - Se usa bloques con referencias al comienzo a los bloques que contendrán los datos (porciones/extends).
  - Precisa: id. del 1er bloque índice.
  - Desfragmentar.

Sistema de ficheros:

representación usada en Minix: ficheros

importante

Alejandro Calderón Mateos

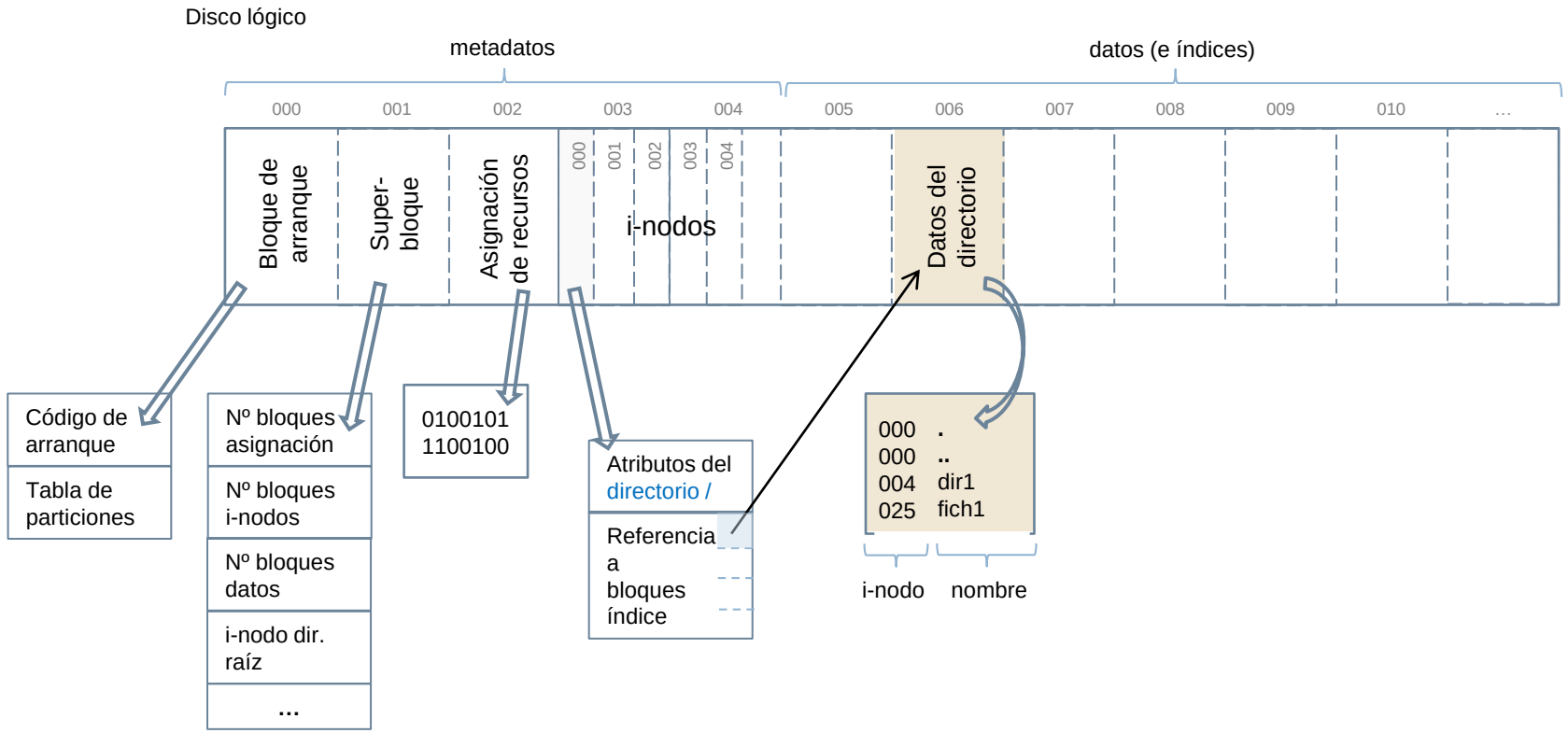


Sistema de ficheros:

representación usada en Minix: directorios

importante

Alejandro Calderón Mateos

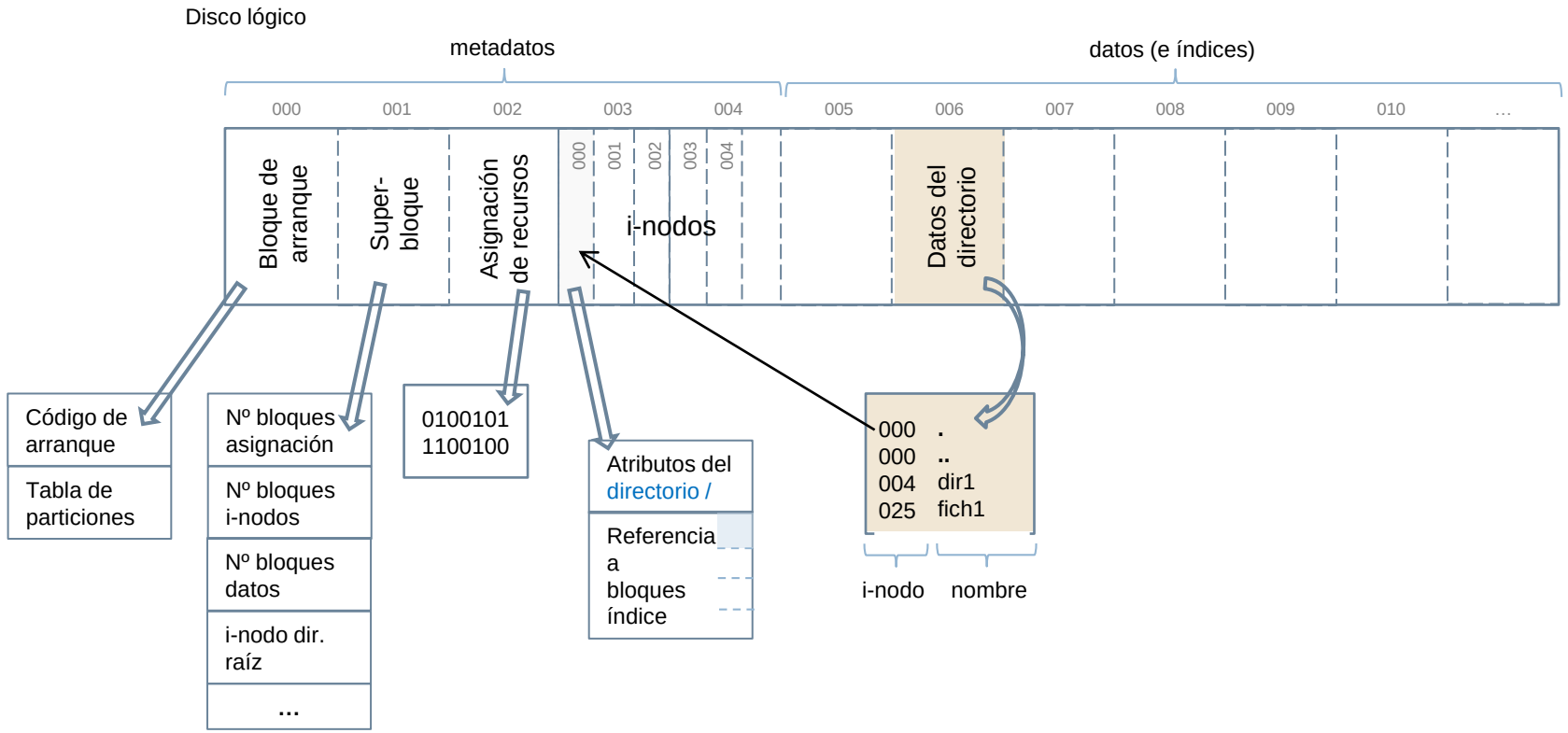


Sistema de ficheros:

representación usada en Minix: directorios

importante

Alejandro Calderón Mateos



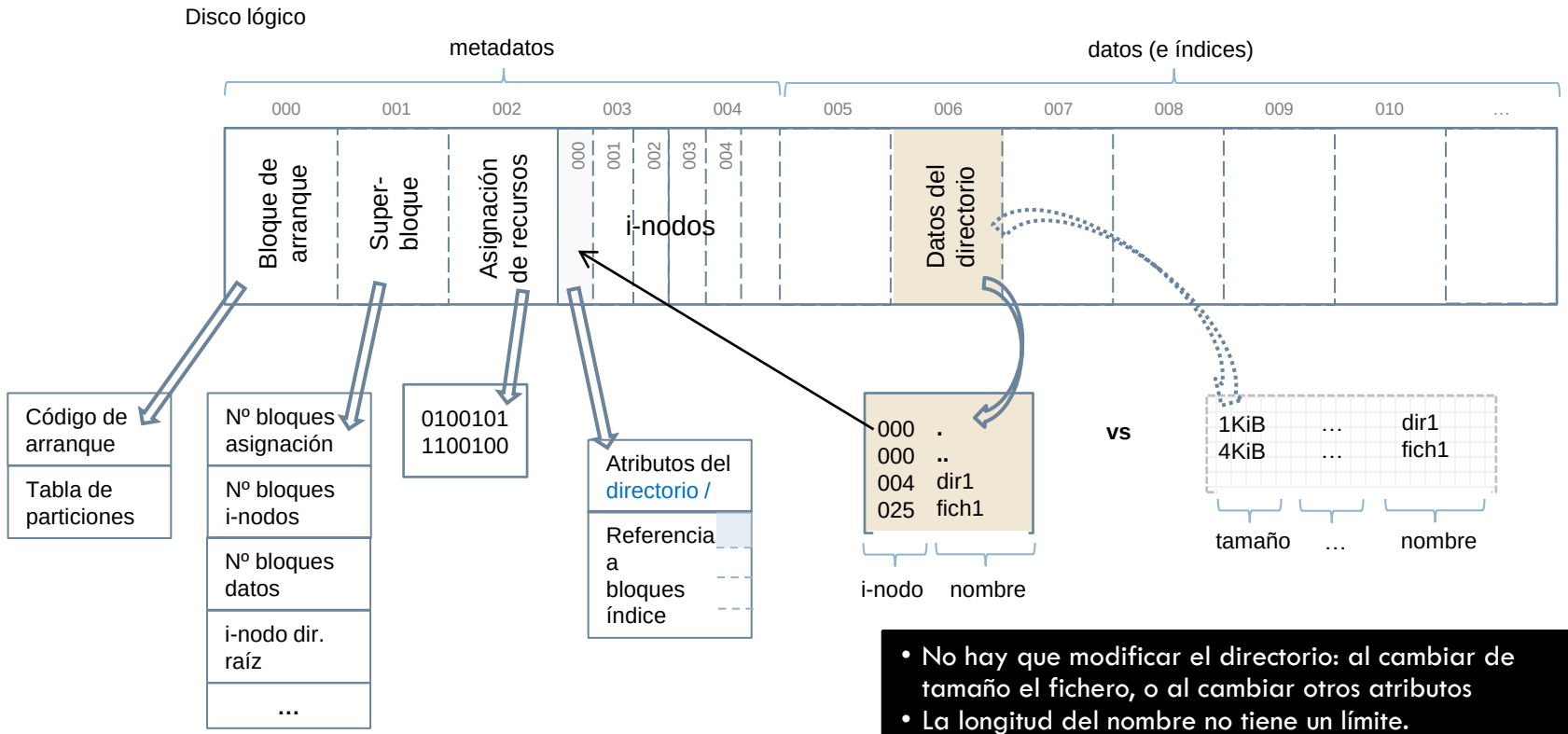
Sistema de ficheros:

representación usada en Minix: directorios

importante

Alejandro Calderón Mateos

- Entrada de directorio = nombre + identificador de i-nodo (con el resto de atributos)



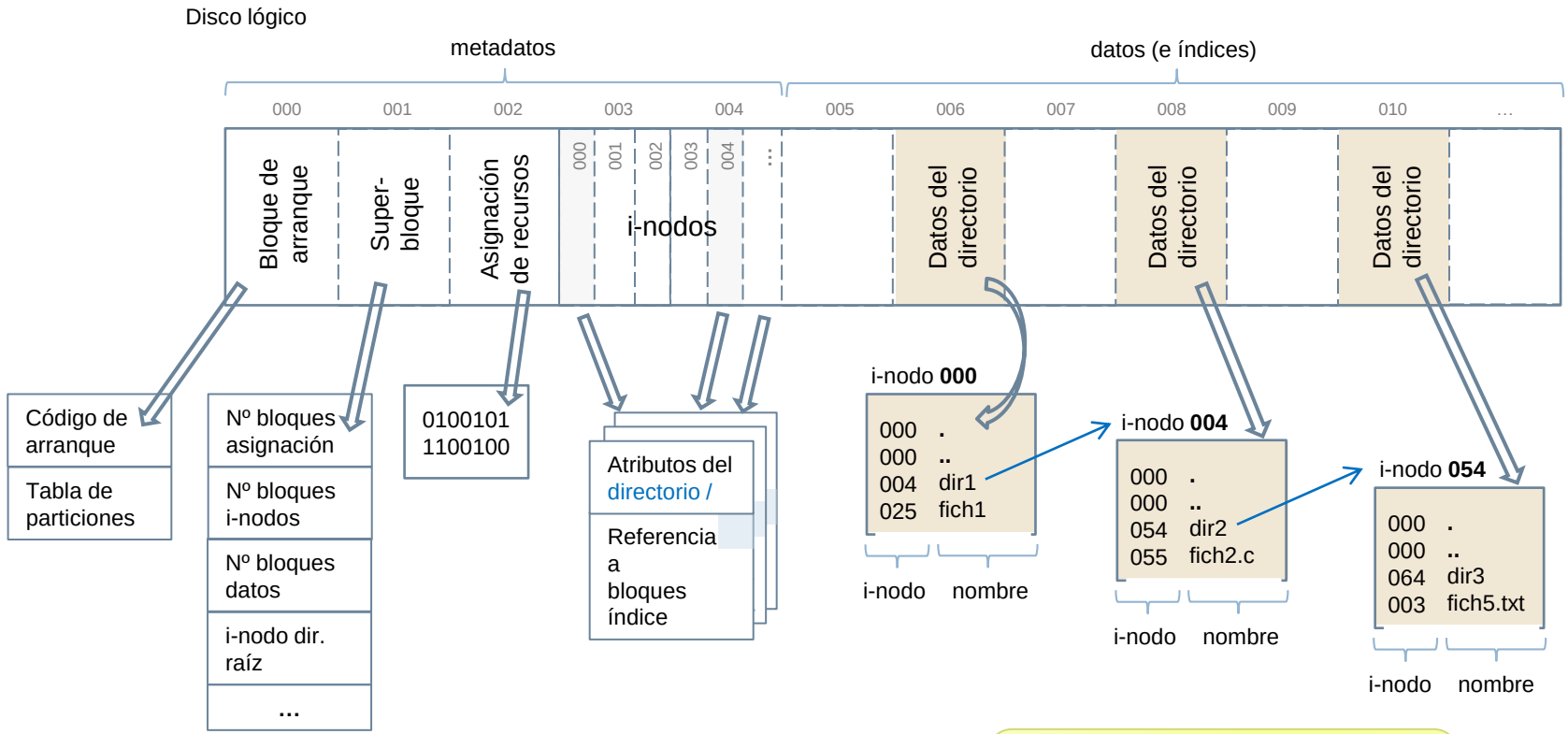
- No hay que modificar el directorio: al cambiar de tamaño el fichero, o al cambiar otros atributos
- La longitud del nombre no tiene un límite.
- Fácil creación de sinónimos para un fichero (enlaces)
- i-nodo con atributos de ficheros y/o directorios.

Sistema de ficheros:

representación usada en Minix: directorios

importante

Alejandro Calderón Mateos



ls –l /dir1/dir2/fich5.txt

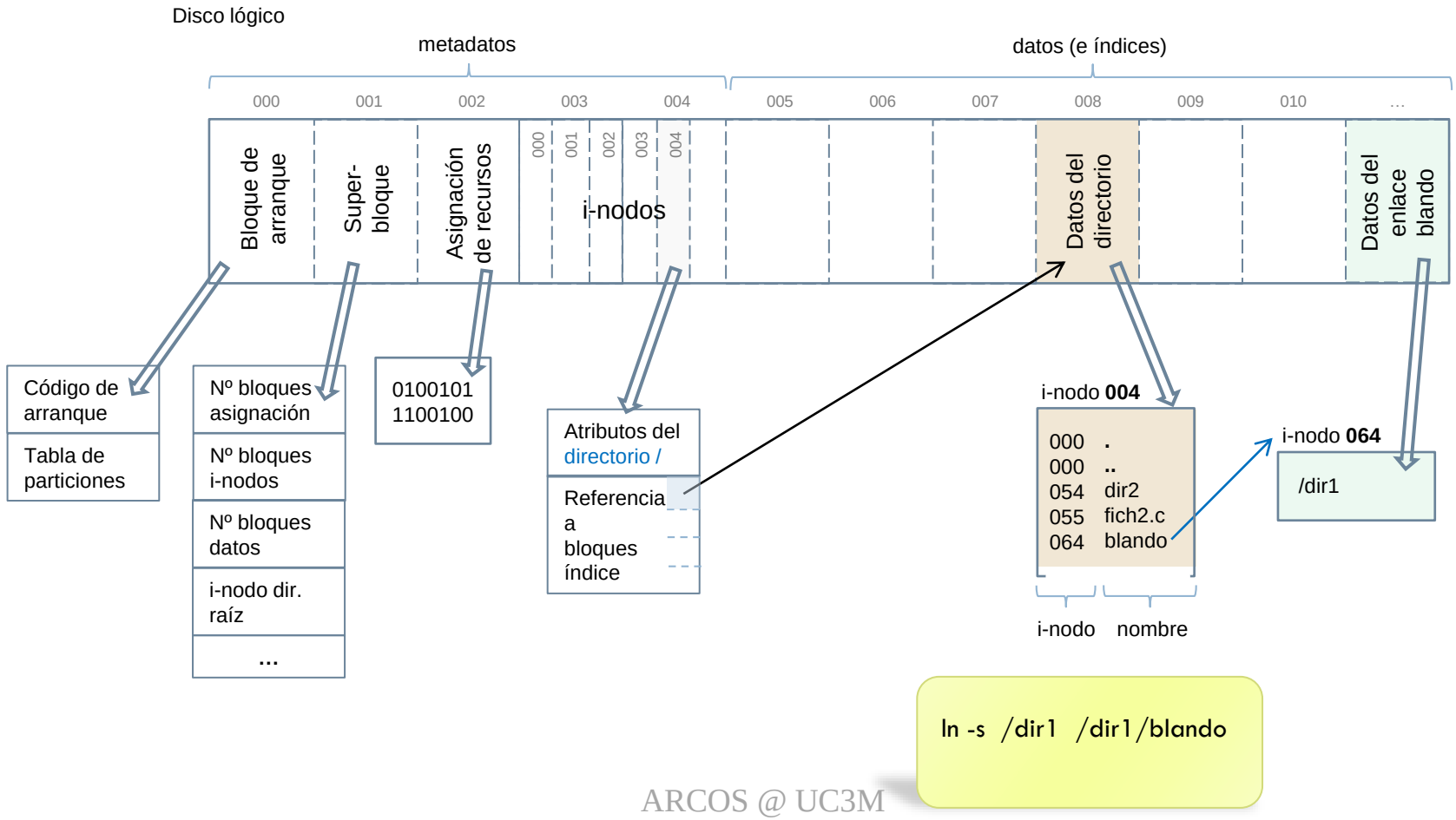
- / + dir1 + dir2 + fich5.txt
- 4 i-nodos + 3 bloques de datos

Sistema de ficheros:

representación usada en Minix: enlace simbólico (o blando)

importante

Alejandro Calderón Mateos

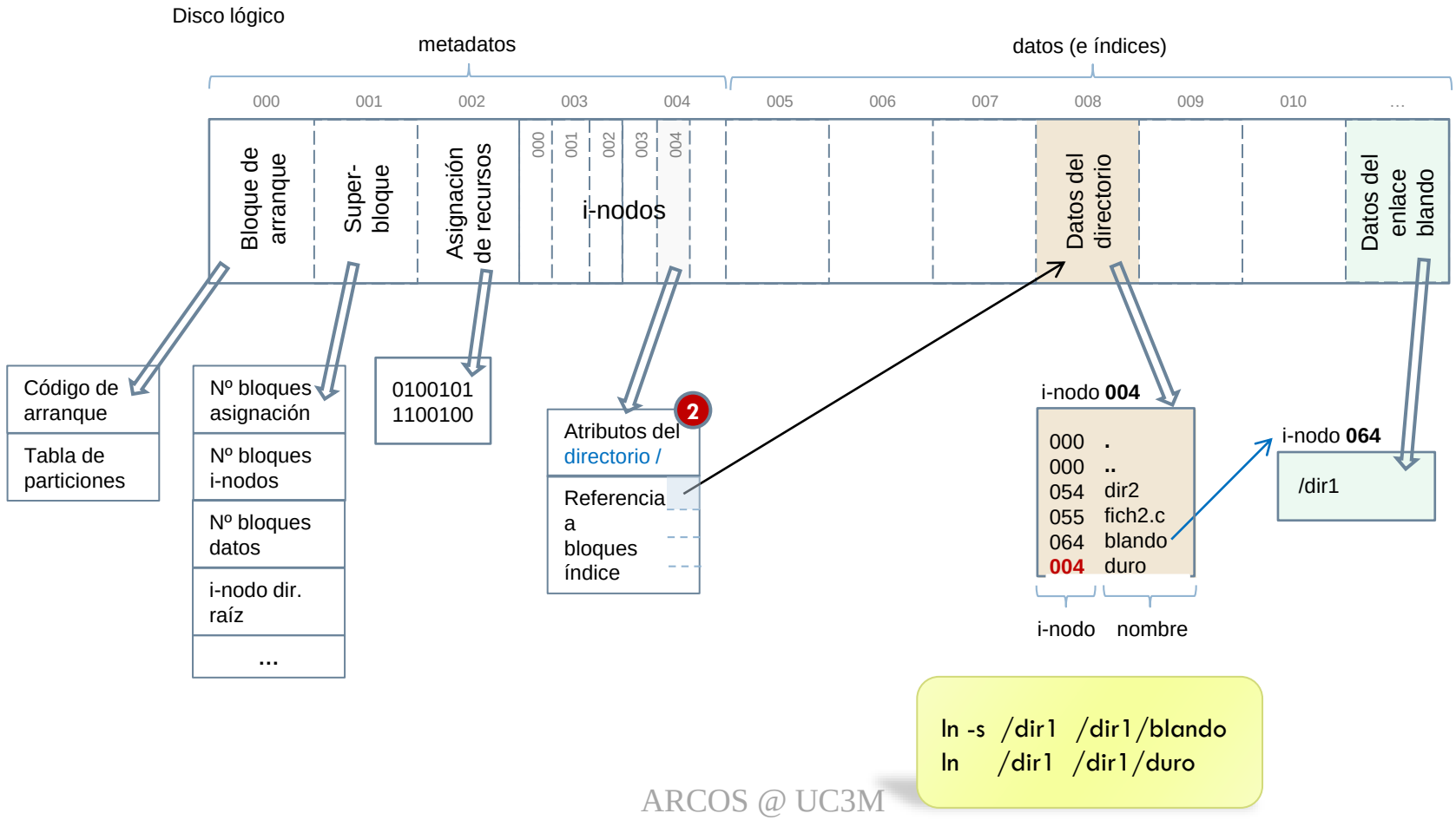


Sistema de ficheros:

representación usada en Minix: enlace duro

importante

Alejandro Calderón Mateos



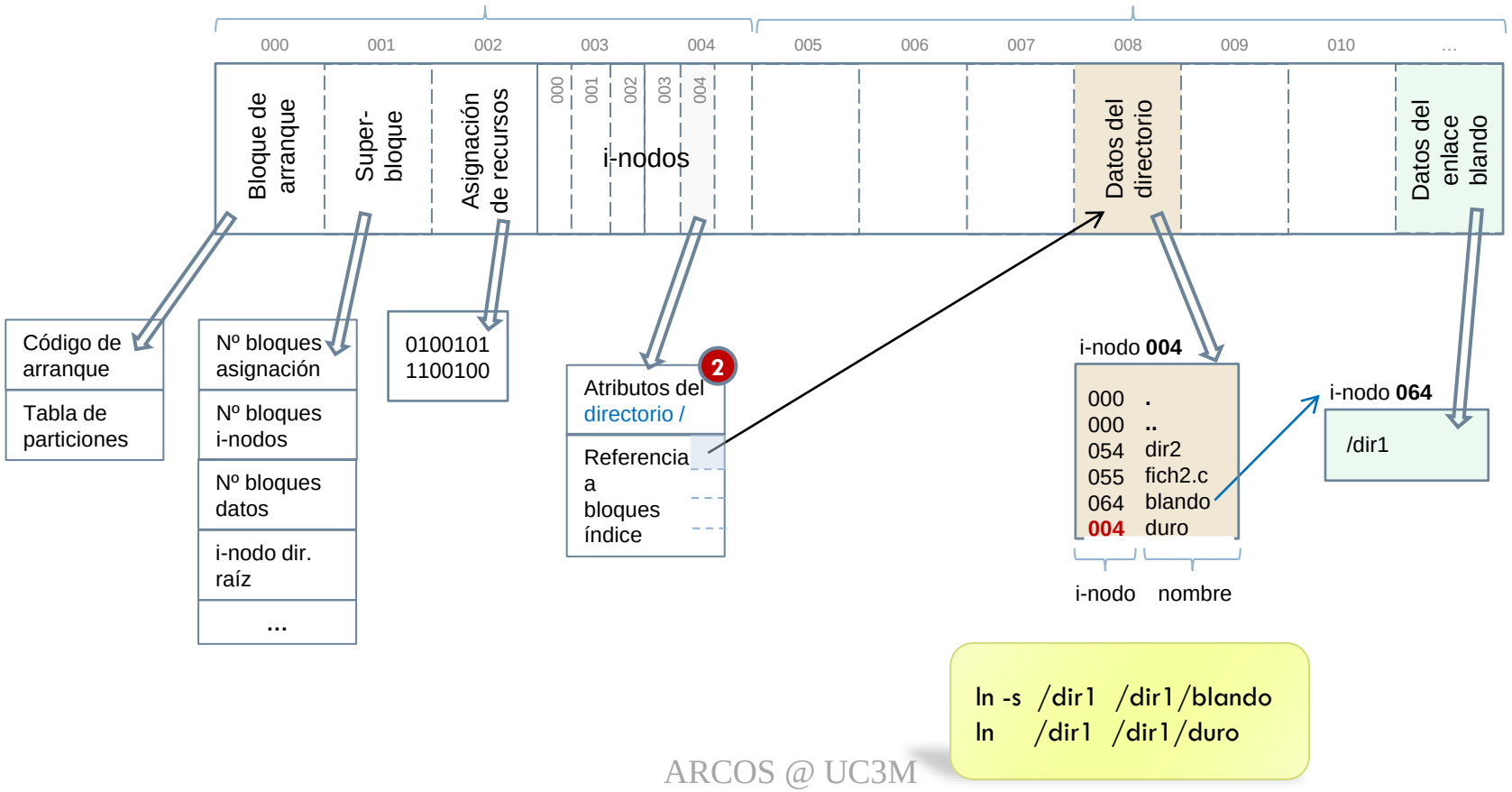
Sistema de ficheros:

representación usada en Minix: enlace duro

importante

Alejandro Calderón Mateos

- Físico/duro: nueva entrada directorio a un i-nodo existente (contador de enlaces en i-nodo)
  - Enlaces duros/físicos solo a otros ficheros dentro de la partición
- Simbólico/blando: se crea fichero nuevo que contiene el nombre del fichero/directorio destino.



Sistema de ficheros:

representación usada en FAT: directorios

importante

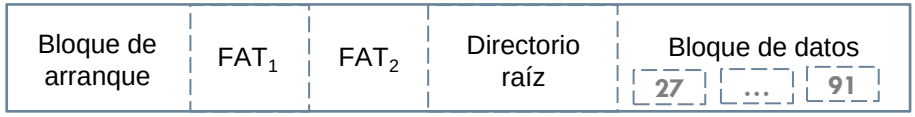
- Entrada de directorio = nombre + resto de atributos (usado para fich. entrelazados y contiguos)

Directorio Raíz

Nombre	Atrib.	KB	Agrup.
dir1	dir	5	27
fich1.txt		12	45

Directorio dir1

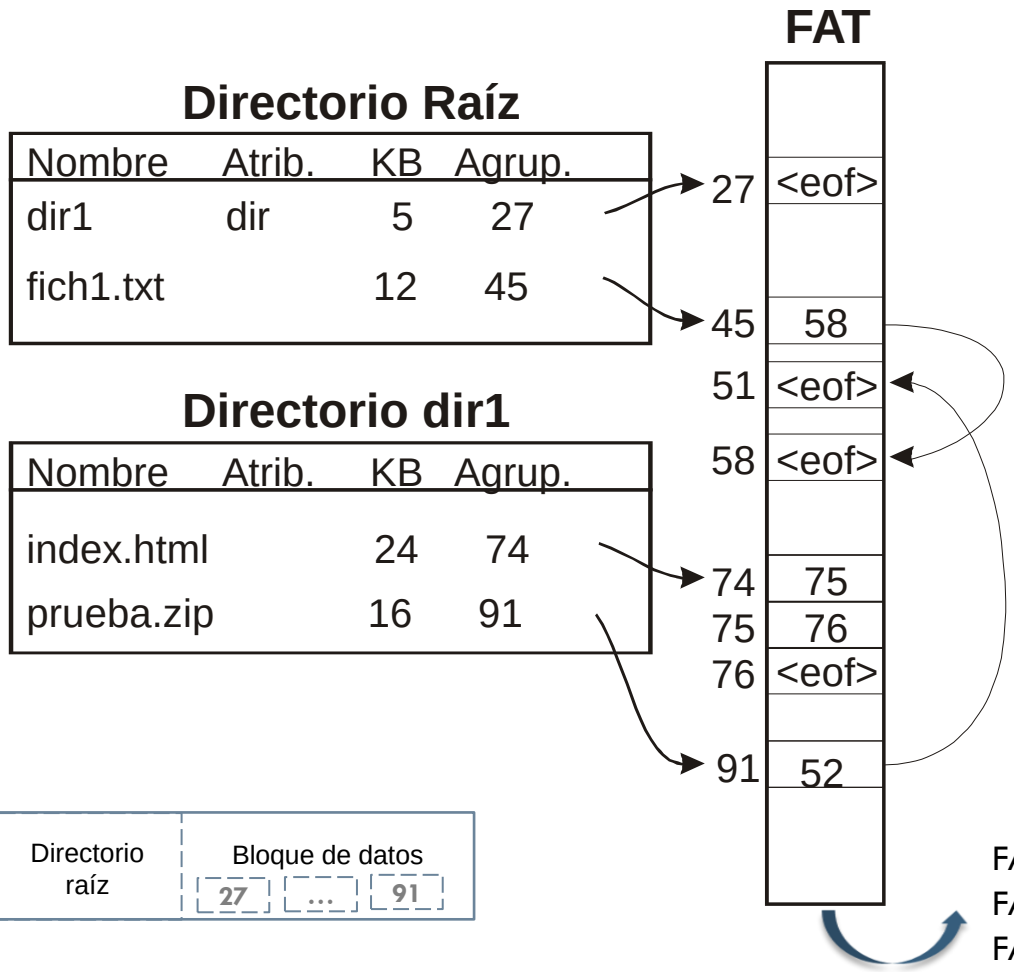
Nombre	Atrib.	KB	Agrup.
index.html		24	74
prueba.zip		16	91



Sistema de ficheros:

representación usada en FAT: ficheros

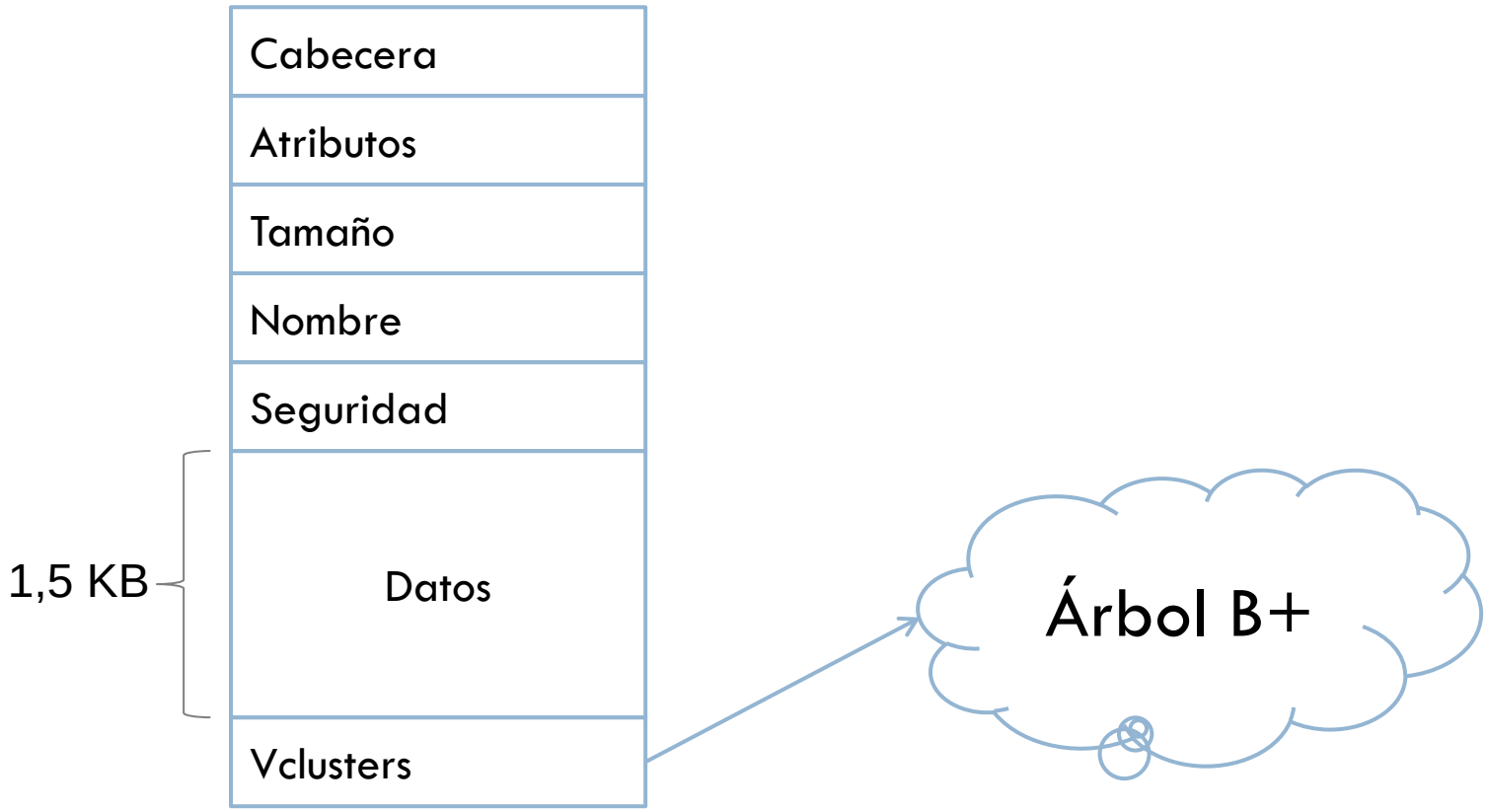
importante



FAT 12 bits (2^{12} bloques)
FAT 16 bits (2^{16} bloques)
FAT 32 bits (2^{32} bloques)

Sistema de ficheros:

representación usada en NTFS



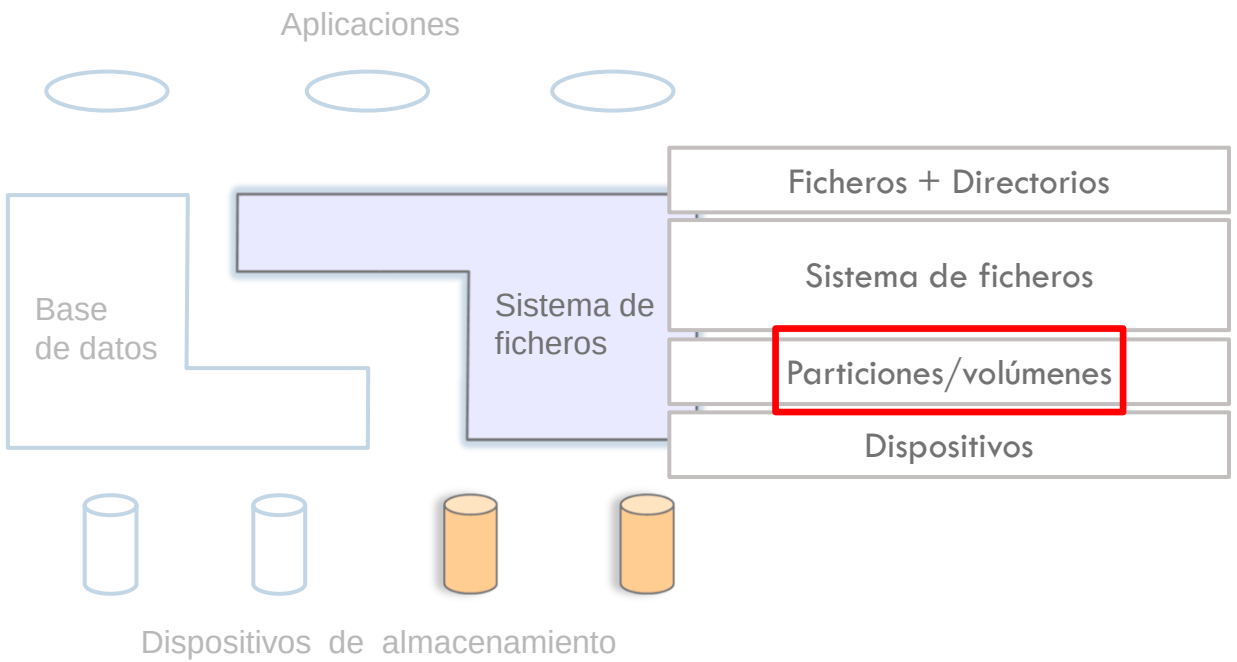
Contenidos

Alejandro Calderón Mateos

- Introducción
- Fichero
- Directorio
- Sistema de ficheros
- **Particiones/Volúmenes**
- Dispositivos
- Software de sistema
- Sistema de ficheros (gestor)

Particiones/Volúmenes

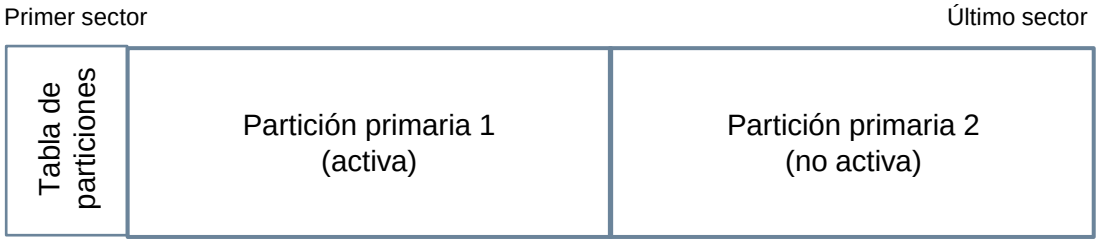
Alejandro Calderón Mateos



Particiones

importante

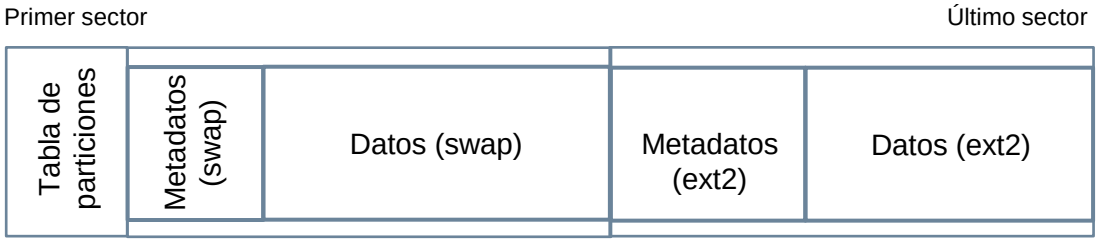
- Contenedor de un sistema de ficheros.



- 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 sistema operativo como una entidad lógica independiente*.
- Típicamente al principio del dispositivo se guarda la tabla de particiones:
  - Cada entrada de la tabla de particiones guarda los atributos de la partición asociada.
  - Un dispositivo se puede dividir en una o más particiones (la tabla de particiones lista todas).

Particiones

- Contenedor de un sistema de ficheros.



- Una vez creadas las particiones, el sistema operativo debe crear las estructuras de datos de los sistemas de archivos dentro de esas particiones:
  - El sector de arranque en MS-DOS/DR-DOS
  - En el superbloque en Unix
- Para ello se proporcionan mandatos como `format` o `mkfs` al usuario:
  - # mkswap –c /dev/hda1 20800
  - # mkfs -c /dev/hda2 –b 8196 123100

Particiones

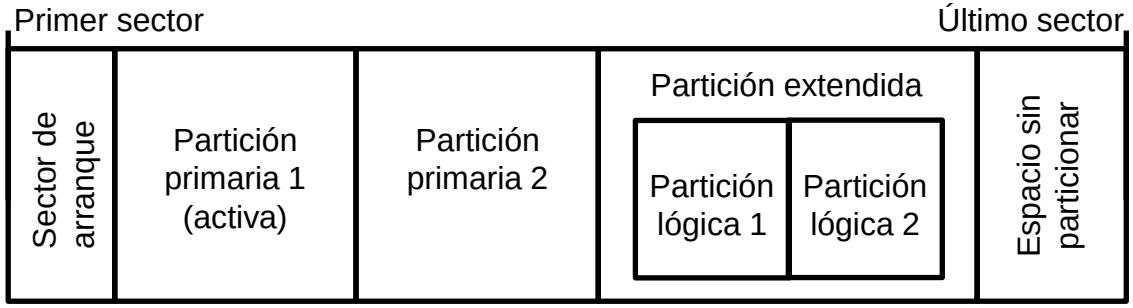
importante

Alejandro Calderón Mateos

- Atributos típicos de una partición:
  - Tipo: primaria, secundaria, unidad lógica, con arranque, etc..
  - Tamaño: inicio y fin de partición.
  - Sistema albergado: linux, linux swap, vfat, etc.
  - Identificación: número de partición (orden o UUID).

Particiones: particionado tradicional en PC

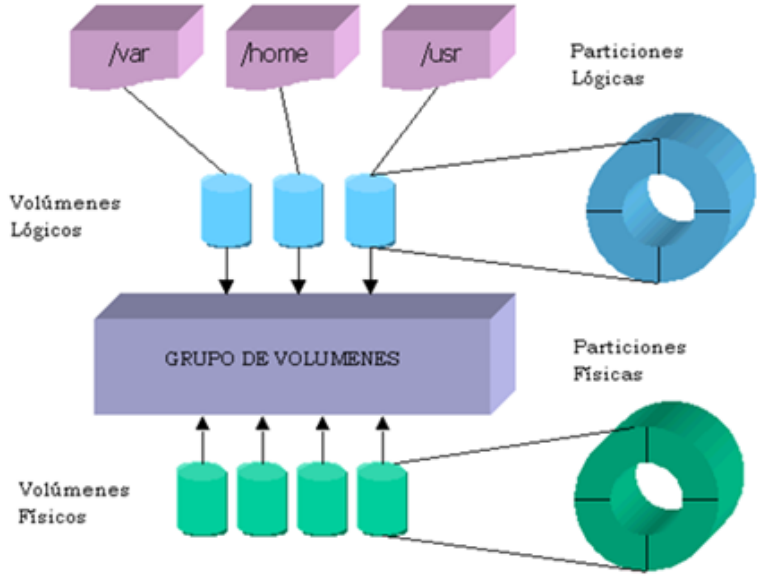
Alejandro Calderón Mateos



- Sector de arranque contiene la tabla de particiones
- Partición primaria o secundaria (con unidades lógicas)
- Antiguo y limitado:
  - 4 particiones en total (primarias + secundarias)
  - No es posible cambiar el tamaño sin perder los datos

Volúmenes

importante

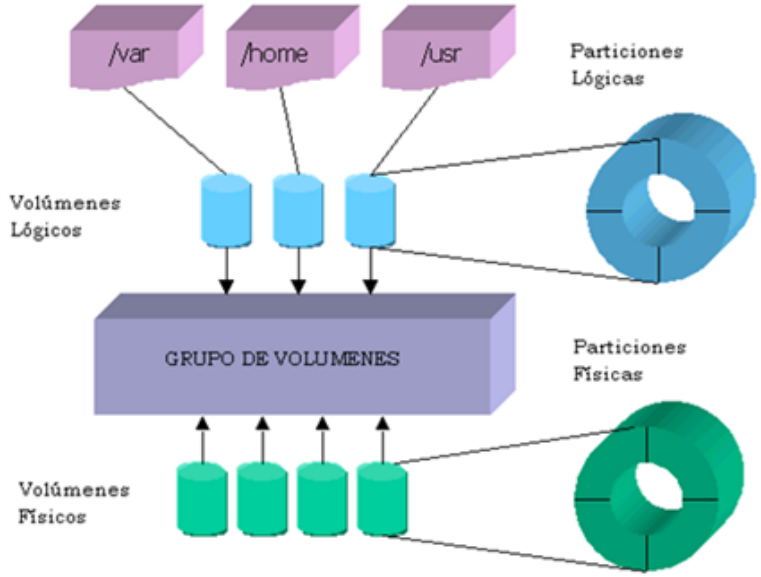


http://www.howtoforge.com/linux_lvm

- Volúmenes lógicos, sobre grupo de volúmenes, compuestos de volúmenes físicos.
  - Volumen lógico se manifiesta de forma similar a las antiguas particiones
- Más moderno y flexible:
  - Mayor número (+ límite), cambio dinámico, uso de múltiples discos, etc.

Volúmenes

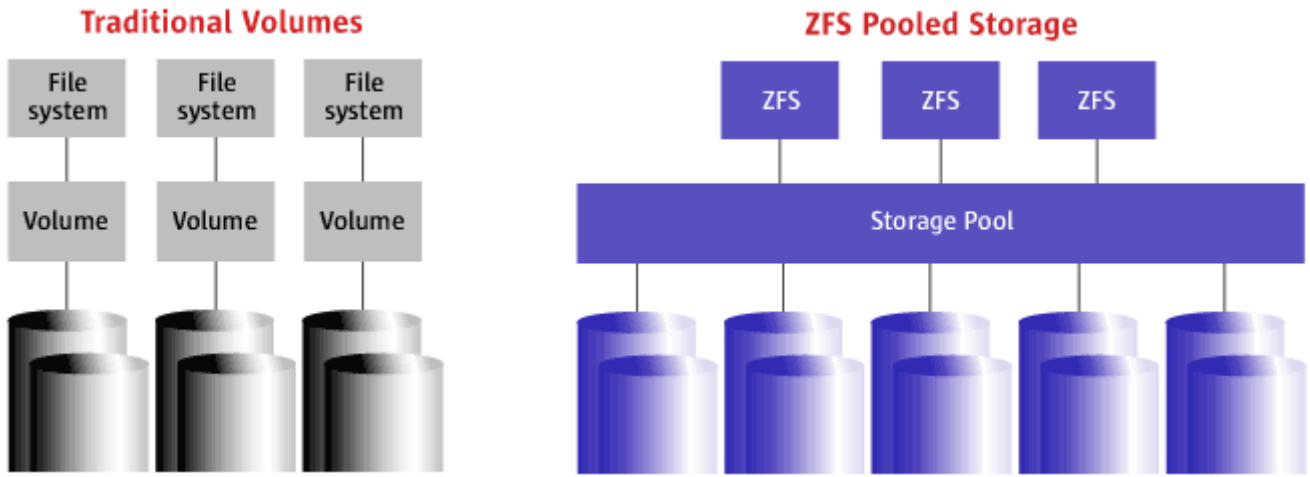
Alejandro Calderón Mateos



- Crear un volumen físico, un grupo de volúmenes y uno lógico:
  - # pvcreate /dev/sdb1
  - # vgcreate vol_infoso /dev/sdb1
  - # lvcreate –L100M –nweb vol_infoso
  - # mkfs –t ext3 /dev/vol_infoso/web
  - # mount /dev/vol_infoso/web /mnt

Pool de almacenamiento (ZFS)

Alejandro Calderón Mateos

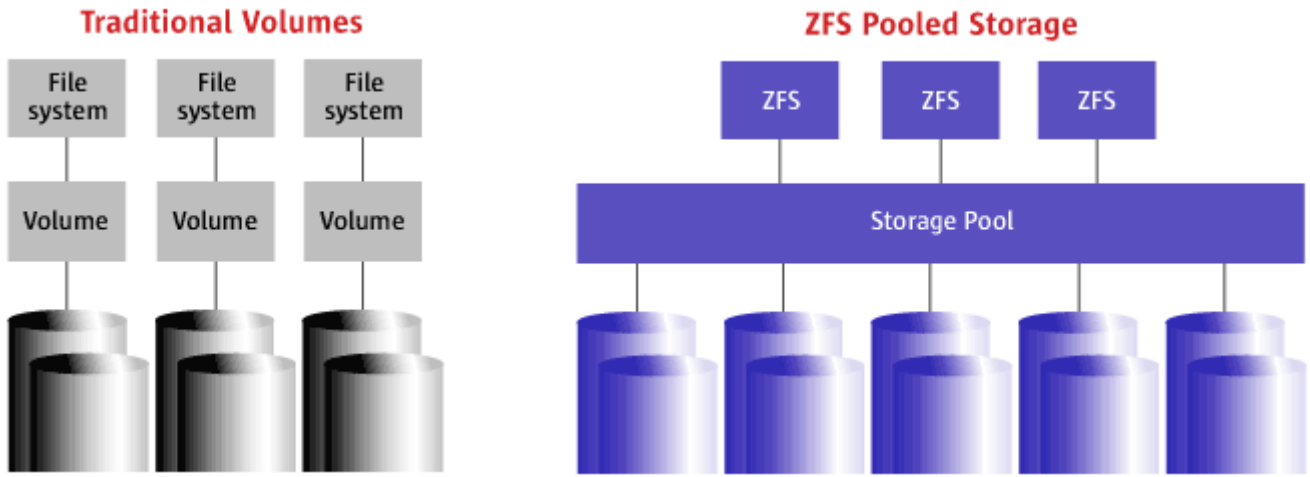


http://hub.opensolaris.org/bin/download/Community+Group+zfs/docs/zfslast.pdf

- Simplificación en el uso de dispositivo, volúmenes y sistemas de ficheros mediante la integración de los mismos.
- Se crean los sistemas de ficheros sobre un pool de almacenamiento compuesto por dispositivos físicos (o partes de ellos)

Pool de almacenamiento (ZFS)

Alejandro Calderón Mateos



- Crear el *pool*, un sistema de ficheros y establecer opciones:
  - # zpool create infoso /dev/disk1
  - # zfs create infoso/practicas
  - # zfs set mountpoint=/export/practicas/infoso infoso/practicas
  - # zfs set quota=10g infoso/practicas

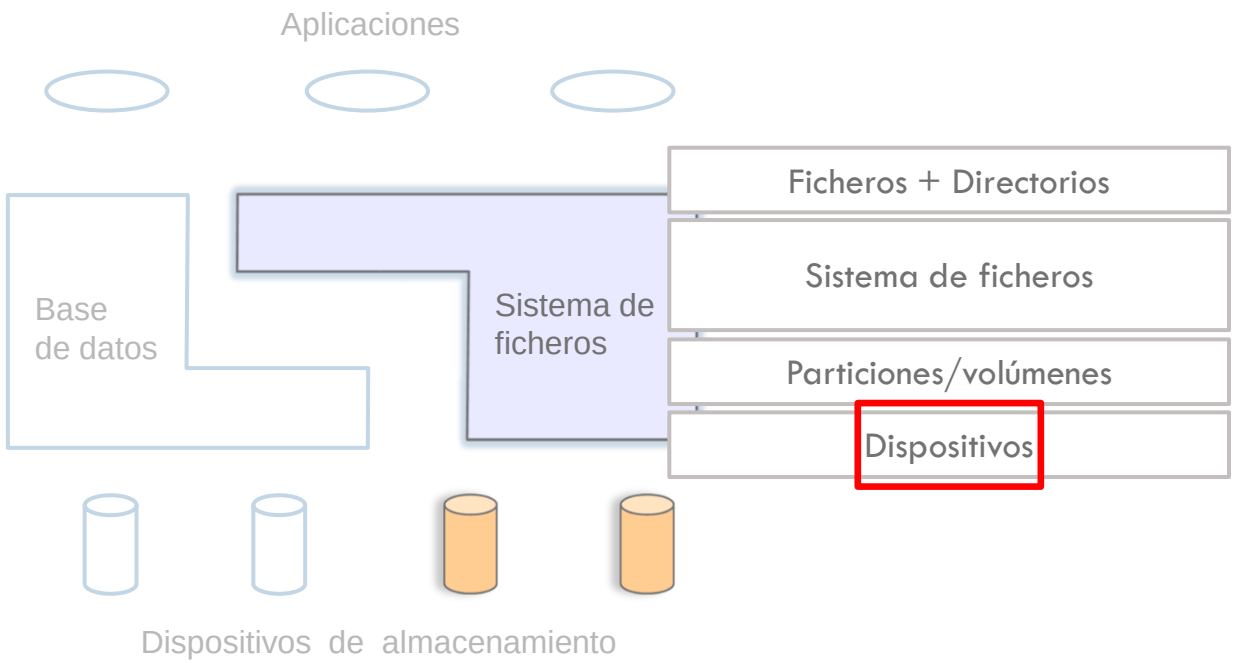
Contenidos

Alejandro Calderón Mateos

- Introducción
- Fichero
- Directorio
- Sistema de ficheros
- Particiones/Volúmenes
- **Dispositivos**
- Software de sistema
- Sistema de ficheros (gestor)

Dispositivos

Alejandro Calderón Mateos



Dispositivos

reales

- Disco duro

- SSD (estado sólido)

- Sistemas ópticos

- Etc.

Dispositivos

reales

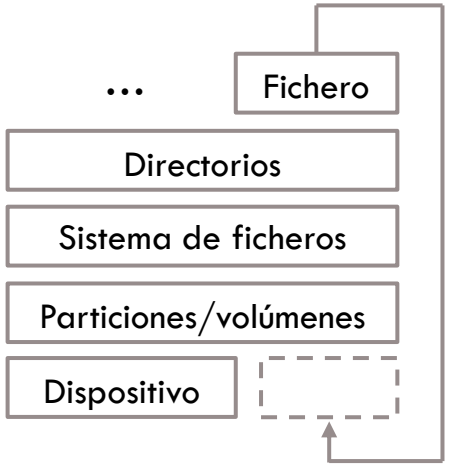
- Listar los dispositivos PCI:

```
acaldero@phoenix:~/infodso/$ lspci
00:00.0 Host bridge: Intel Corporation 82Q35 Express DRAM Controller (rev 02)
00:01.0 PCI bridge: Intel Corporation 82Q35 Express PCI Express Root Port (rev 02)
00:03.0 Communication controller: Intel Corporation 82Q35 Express MEI Controller (rev 02)
00:03.2 IDE interface: Intel Corporation 82Q35 Express PT IDER Controller (rev 02)
00:03.3 Serial controller: Intel Corporation 82Q35 Express Serial KT Controller (rev 02)
...
```

- Listar los dispositivos USB:

```
acaldero@phoenix:~/infodso/$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
...
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 002: ID 1241:1166 Belkin MI-2150 Trust Mouse
Bus 005 Device 002: ID 0c45:600d Microdia TwinkleCam USB camera
```

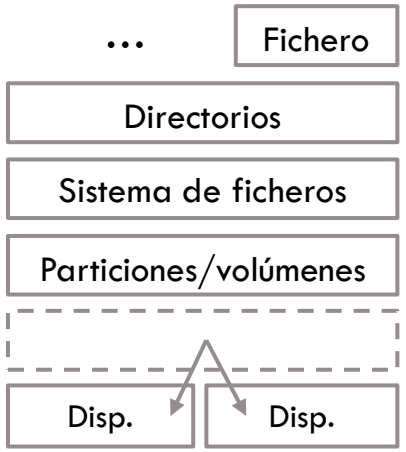
Dispositivos especiales



- Dispositivo *loopback*
  - Fichero como dispositivo de bloques

- Ejemplo de sesión de trabajo:
  1. [1] Usar una imagen de CD-ROM/DVD:
     ```
     wget ftp://ftp.rediris.es/sites/releases.ubuntu.com/releases/21.04/ubuntu-21.04-desktop-i386.iso
     ```
  2. Asociar el fichero al dispositivo de *loopback*:
     ```
     sudo losetup /dev/loop1 /tmp/ubuntu-21.04-desktop-i386.iso
     ```
  3. Montar como dispositivo de bloques (disco):
     ```
     mount  /dev/loop1 /mnt
     ```
  4. Usar el sistema de ficheros de /mnt
  5. Desmontar el dispositivo:
     ```
     umount /dev/loop1
     ```
  6. Desasociar el dispositivo:
     ```
     losetup -d /dev/loop1
     ```

Dispositivos especiales



- Dispositivo *md*
  - Dispositivo de dispositivos

- Ejemplo de sesión de trabajo:
  1. [1] Crear el dispositivo md espejo:
     ```
     mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/loop1 /dev/loop2
     ```
  2. [1] Crear el sistema de ficheros:
     ```
     mkfs -t ext3 /dev/md0
     ```
  3. Montar y desmontar el dispositivo:
     ```
     mount  /dev/md0 /mnt
     umount /dev/md0
     ```
  4. Parar el dispositivo md:
     ```
     mdadm --stop /dev/md0
     ```
  5. Arrancar el dispositivo md:
     ```
     mdadm --assemble /dev/md0  /dev/loop1 /dev/loop2
     ```

SISTEMAS OPERATIVOS: SISTEMAS DE FICHEROS

Ficheros, directorios y sistema de ficheros