Programación y Administración de Sistemas

6. Sistemas de ficheros y discos

Pedro Antonio Gutiérrez

Asignatura "Programación y Administración de Sistemas"
2º Curso Grado en Ingeniería Informática
Escuela Politécnica Superior
(Universidad de Córdoba)
pagutierrez@uco.es

22 de marzo de 2015

Introducción

- La función principal de un disco duro es almacenar la información del PC cuando no se encuentra conectado a la corriente eléctrica.
- También puede servir de extensión para la memoria RAM, gracias al mecanismo de memoria virtual.
- En la actualidad, existen dos tecnologías que conviven en los discos duros: la de los SSD y la de los discos rígidos.
- Los discos rígidos funcionan de forma parecida a un tocadiscos, mientras que los discos SSD (*Solid State Disk* o, mejor, *Solid State Drive*) utilizan una memoria formada por semiconductores para almacenar la información (similar a *pendrives* o tarjetas de memoria).

Discos rígidos

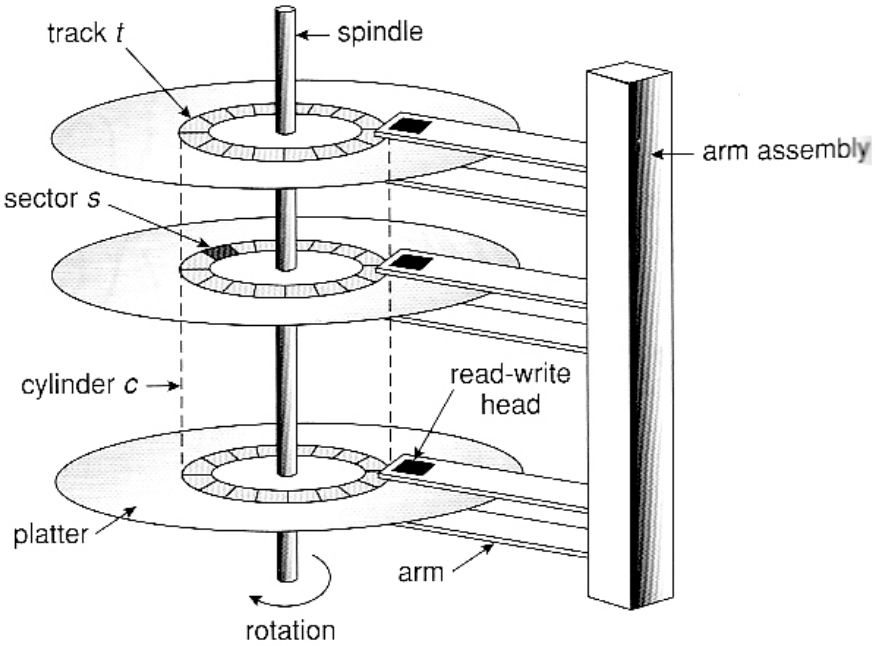


Figure 12.1 Moving-head disk mechanism.

Discos rígidos

- ¿De qué esta compuesto un disco duro rígido?
  - Plato: Cada uno de los discos que se encuentran apilados en su interior, cubiertos de un material magnetizable (de aluminio o cristal). La escritura cambia el estado de este material.
  - Cabezal: es un brazo que se mueve sobre el plato. Como los discos giran, permite acceder a cualquier punto de los mismos.
  - Pista: Se trata de cada una de las líneas esféricas que se pueden formar sobre cada plato.
  - Cilindro: Conjunto de varias pistas que se encuentran una encima de otra.
  - Sector: Cada una de las divisiones que se hace de la circunferencia que se forma en el disco. Normalmente en un sólo sector tendremos varios cientos de bytes de información.
- Indicando el cilindro, la cabeza y el sector podemos acceder a cualquier dato del disco.

Discos rígidos: escritura

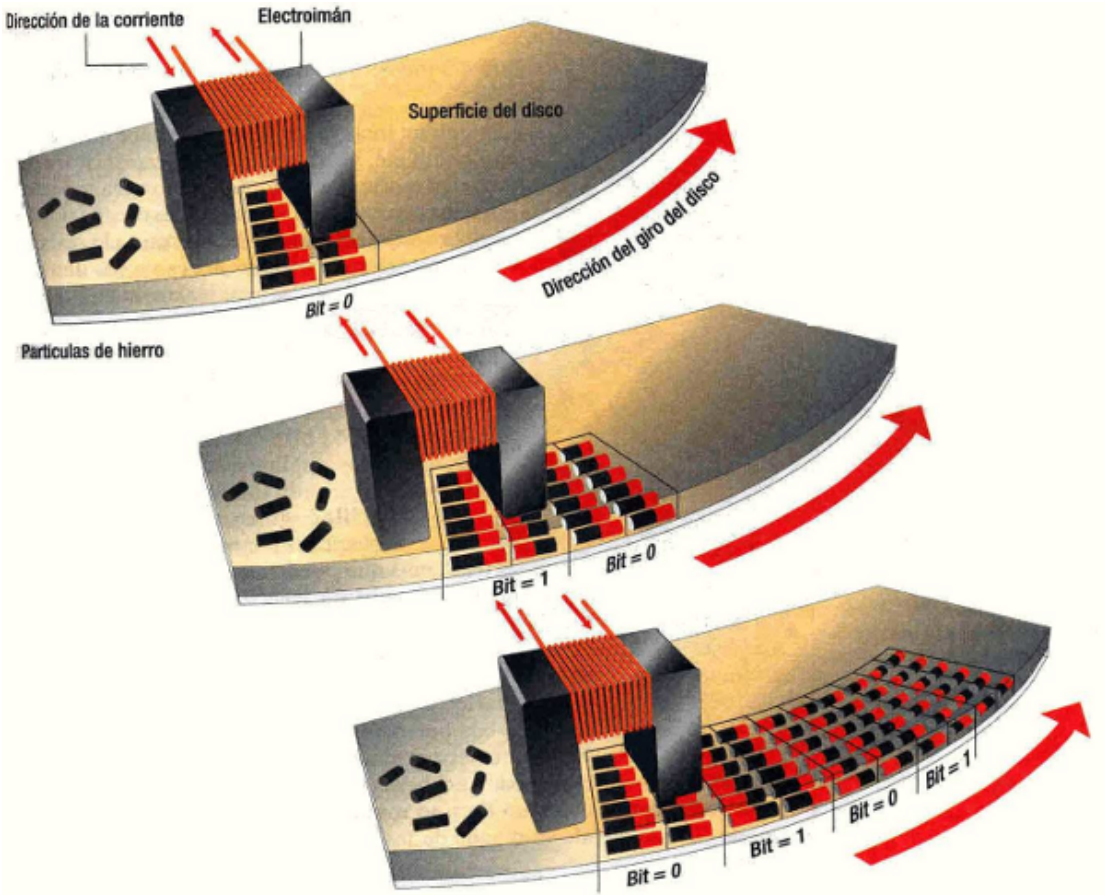


Figura 4.2
Datos que están siendo grabados por una cabeza de lectura/escritura.

Archivos

- Archivo: unidad de almacenamiento lógico no volátil que agrupa un conjunto de información relacionada entre si bajo un mismo nombre.
  - Un archivo debe poseer un nombre que permita acceder al mismo de forma unívoca.
  - Este nombre incluye una extensión (`.txt`, `.zip`...) que identifica el tipo de archivo.
  - El acceso a un archivo puede ser secuencial (para acceder a una posición hay que acceder antes a las anteriores) o directo/aleatorio (se puede acceder a cualquier posición).

Sistema de Archivos

- Sistema de Archivos/Ficheros (SA, SF):
  - Organiza la información de los dispositivos de almacenamiento secundario (disco duro, disco extraíble, DVDs, CDRom...).
  - El dispositivo se divide manera lógica para que quede organizado de una forma inteligible para el SO.
  - La división se hace a múltiples niveles:
    - Particiones o volúmenes.
    - Bloques.
    - Agrupaciones.

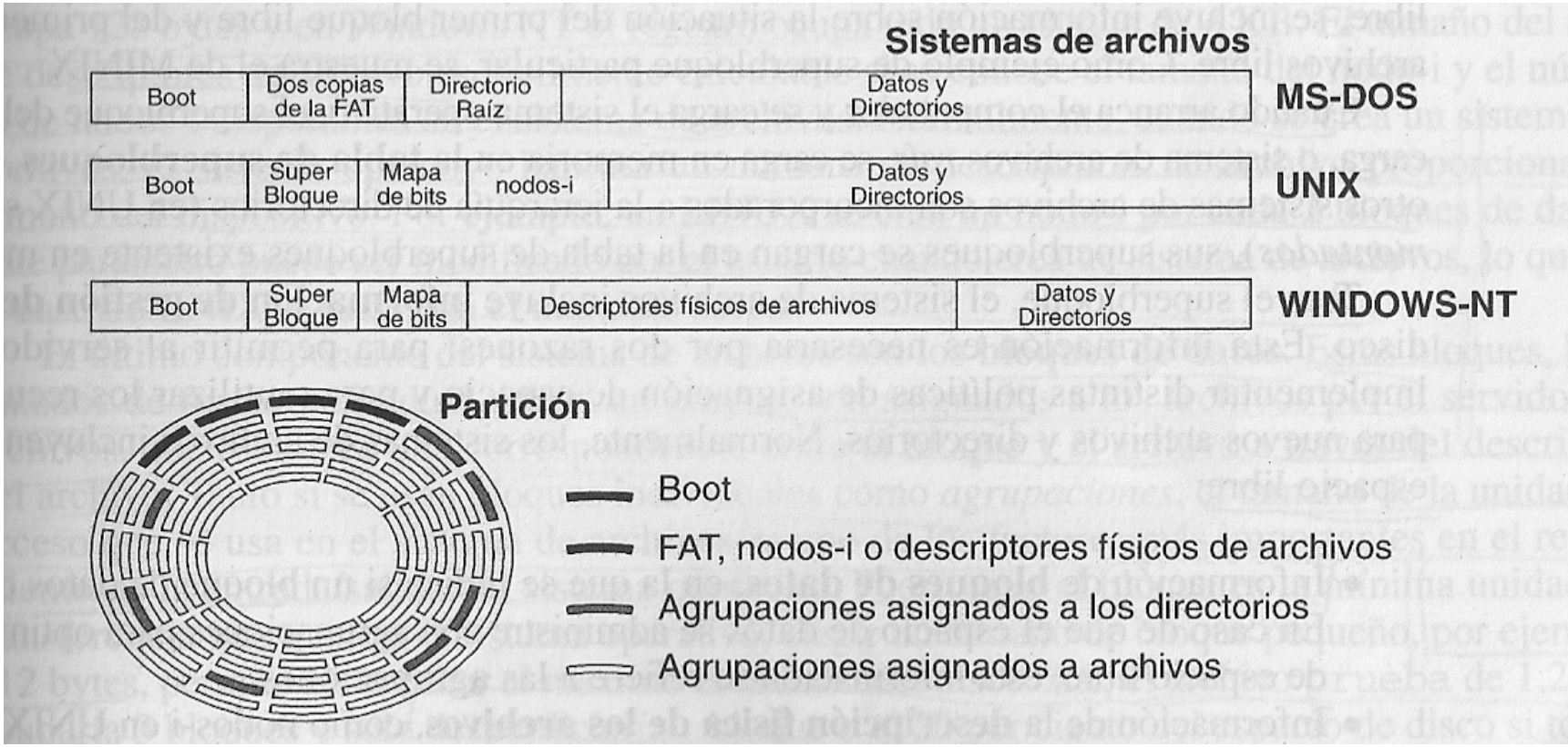
Sistema de Archivos

- Partición: porción de un disco a la que se le dota de una identidad propia y que se manipula como un entidad lógica independiente.
  - Las particiones deben formatearse para que se creen las estructuras necesarias que permiten al SO manipular el disco.
- Bloque: agrupación lógica de sectores físicos del disco, la cual supone la unidad de transferencia mínima que usa el SA.
  - El tamaño de bloque es un parámetro decisivo que afecta a la eficiencia del acceso a disco y a la fragmentación del mismo.
    - Tamaño de bloque pequeño: Mayor número de operaciones de Entrada/Salida (E/S) para acceder al archivo. Menor fragmentación.
    - Tamaño de bloque grande: Menor número de operaciones E/S para acceder al archivo. Mayor fragmentación.

Sistema de Archivos

- Agrupación: conjunto de bloques gestionado como una unidad lógica de almacenamiento.
- El componente del SO que gestiona todo esto es el Servidor de Archivos.
- A la hora de implantar el SA, surgen dos problemas de diseño:
  1. Definir cómo debe ver el usuario el SA.
     - Estructura de los archivos y sus atributos.
     - Operaciones permitidas sobre los archivos.
     - Estructura de directorios.
  2. Algoritmos y estructuras de datos que deben crearse para establecer la correspondencia entre el SA lógico y los dispositivos físicos donde se almacenan.

Estructura del Sistema de Archivos



Estructura del Sistema de Archivos

- El bloque de carga (boot) contiene código ejecutado al arrancar el ordenador por el iniciador ROM.
  - Este código, iniciará el SO cargando los archivos de sistema alojados en el disco duro.
  - Se suele incluir en todas las particiones (aunque no contengan el SO) para así mantener una estructura uniforme.
  - Se añade un número mágico, el cuál será comprobado por el iniciador ROM para demostrar que el bloque de carga es válido.
- Metainformación: super-bloques, FAT, nodos-i, mapas de bits, descriptores físicos...
  - Describe el SA y la distribución de sus componentes.
  - Es necesaria para poder acceder a los datos.

Estructura del Sistema de Archivos

- Superbloque: características del SA, posición de los distintos elementos, tamaño...
  - Se mantiene una serie de información común para todos los SAs y una entrada característica para cada tipo de SA.
  - Al arrancar la máquina, los superbloques de todos los SAs que son cargados se mantienen en memoria.
- Descriptores físicos de archivos: nodos-i, registros de Windows-NT...
  - Describen cada uno de los archivos almacenados.
  - Tienen una estructura y tamaño muy dependiente del SO y el número de descriptores debe ser proporcional al tamaño total del disco.
  - Incluyen: tamaño, apuntadores a los bloques del archivo, permisos, propietarios...

Estructura del Sistema de Archivos

- Gestión del espacio libre: distintos mecanismos permiten gestionar el espacio libre.
  - Se pueden utilizar mapas de bits o listas de recursos libres.
  - Gestión de dos tipos de recursos:
    - Mapas de bloques: indican qué bloques (o agrupaciones) están libres.
    - Mapas de descriptores de archivos: indican qué descriptores de archivos (nodos-i, registros...) están libres.
- Bloques de datos: es dónde se almacena realmente la información.

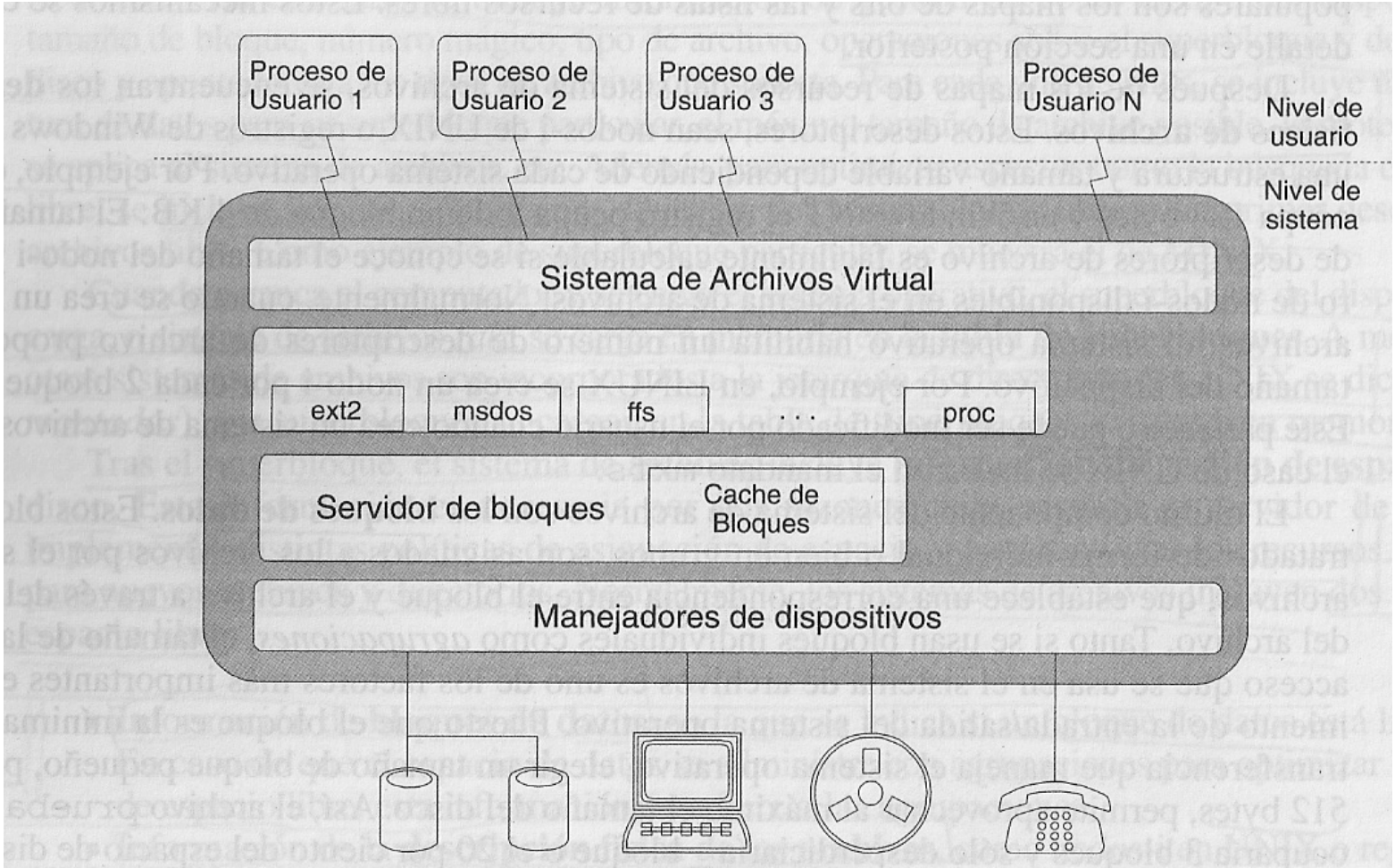
Servidor de Archivos

- Servidor de Archivos: es el componente del SO que se encargará de gestionar el acceso a archivos.
- Se sigue una filosofía de organización en capas.
  - Los niveles inferiores proporcionan servicios a los niveles superiores, y en cada nivel se aumenta la abstracción de las operaciones.

Capas del servidor de archivos

1. Sistema de archivos virtual (+ abstracto).
2. Módulo de organización de archivos.
3. Servidor de bloques.
4. Manejadores de dispositivos (- abstracto).

Servidor de Archivos



Servidor de Archivos

1. Sistema de archivos virtual:
   - Proporciona la interfaz para las llamadas de E/S que deseen realizar los procesos de usuario, interactuando con el módulo de organización de archivos.
   - Cumple las funciones de manejo de directorios, gestión de nombres, servicios de seguridad, integración de archivos de distintos dispositivos/particiones...
   - Por ello, es necesario utilizar una estructura adicional (nodos virtuales, nodos-v en UNIX), que incluye las características comunes a todos los sistemas de archivos y un enlace al descriptor de archivo particular (nodo-i o registro).

Servidor de Archivos

1. Sistema de archivos virtual:
   - Hay operaciones genéricas que se pueden realizar en cualquier SA (caché de nombres, gestión de nodos virtuales...).
   - Otras operaciones deben ser implementadas independientemente para cada SA.
   - Los nodos virtuales contienen la siguiente información:
     - Atributos del archivo.
     - Puntero al nodo-i real.
     - Punteros a funciones que realizan las operaciones genéricas de cualquier SA.
     - Punteros a funciones que realizan las operaciones propias del SA concreto.

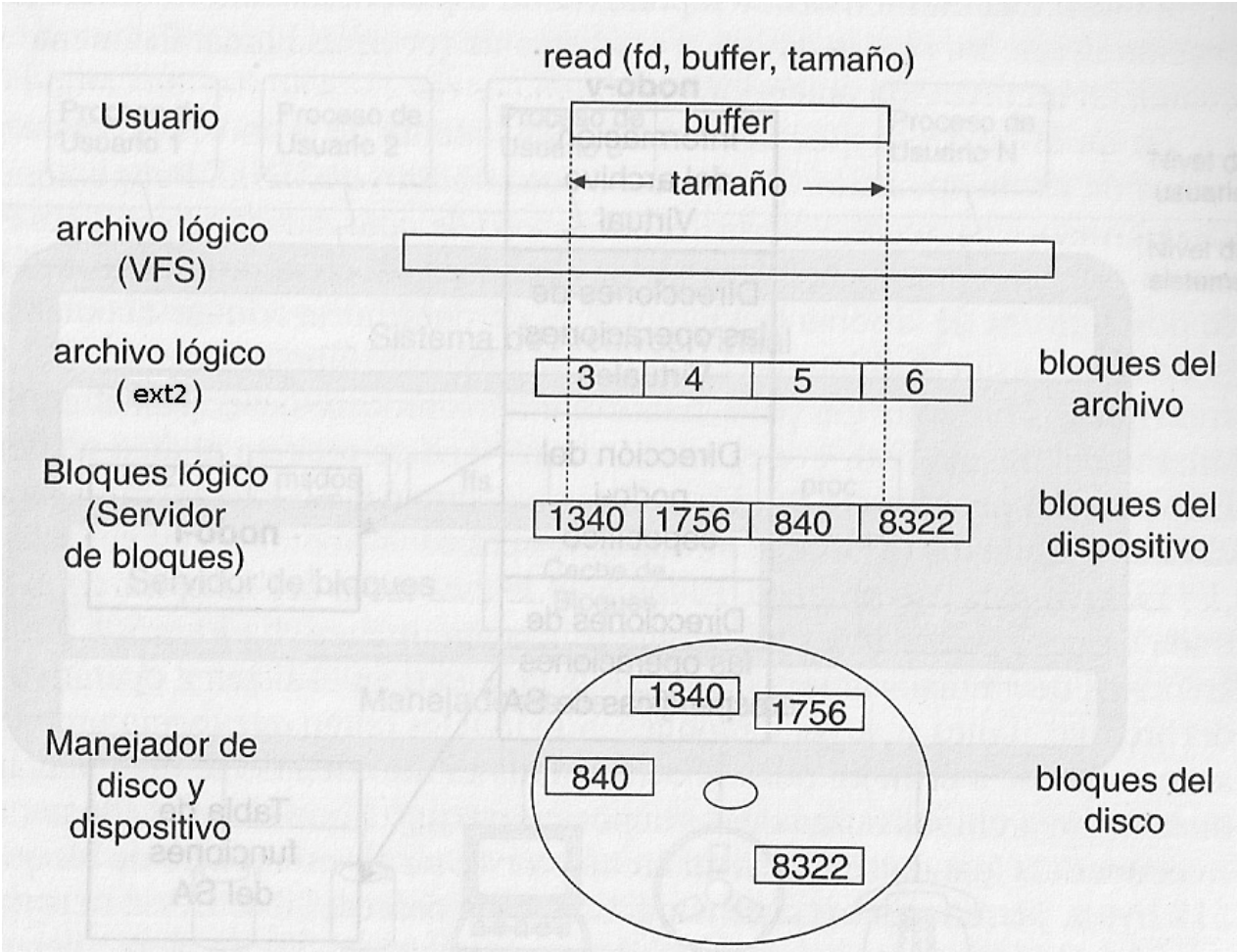
Servidor de Archivos

2. Módulo de organización de archivos:
   - Se implementa por separado para cada SA.
   - Se relaciona la imagen lógica de un archivo con su imagen física, traduciendo direcciones lógicas (contiguas) del archivo a las direcciones físicas (normalmente dispersas) del dispositivo.
   - Se prestan los servicios de gestión de espacio libre y manejo de descriptores de archivos físicos (no virtuales).
   - Este nivel se basa en la información de los nodos-i y utiliza los servicios del servidor de bloques para realizar las operaciones correspondientes.

Servidor de Archivos

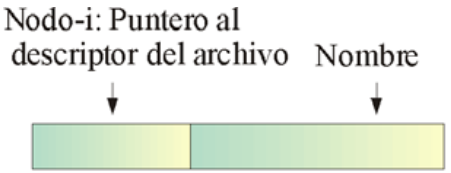
3. Servidor de bloques:
   - Este nivel emite los mandatos genéricos para leer y escribir bloques en los manejadores de dispositivo (E/S de bloques).
   - Se traducirán en llamadas al manejador específico del SA.
   - En este nivel se realiza la caché de bloques.
4. Manejadores de dispositivos:
   - Son específicos para cada SA y para cada *hardware*.
   - Traducen órdenes de E/S de alto nivel a un formato que pueda entender el dispositivo (dependiente del *hardware*).

Servidor de Archivos

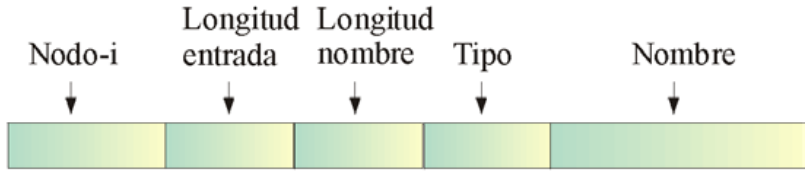


Directorios

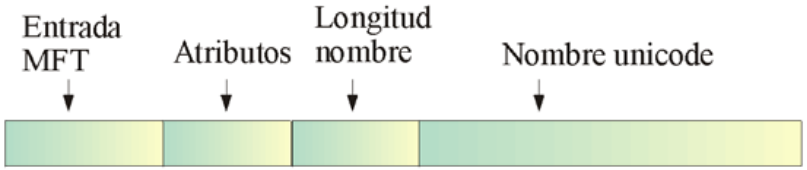
- Un directorio es un fichero con un formato determinado.
- El contenido de un directorio es una serie de entradas (registros), una por cada fichero contenido en él.
- Cada registro tiene, al menos, el nombre del fichero y el puntero al descriptor físico correspondiente.



Directorio de UNIX SV

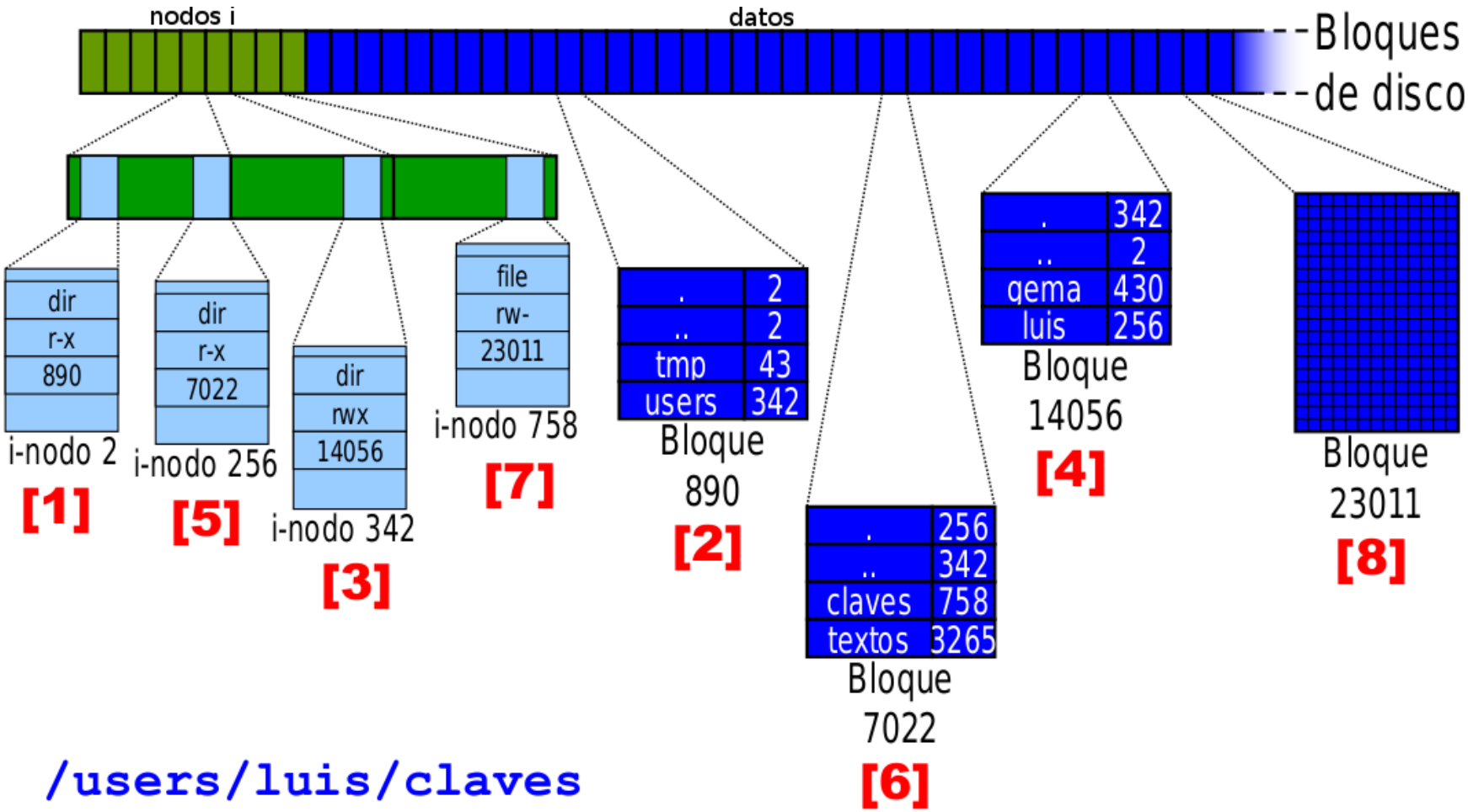


Directorio de LINUX Ext2



Directorio de Windows NTFS

Directorios

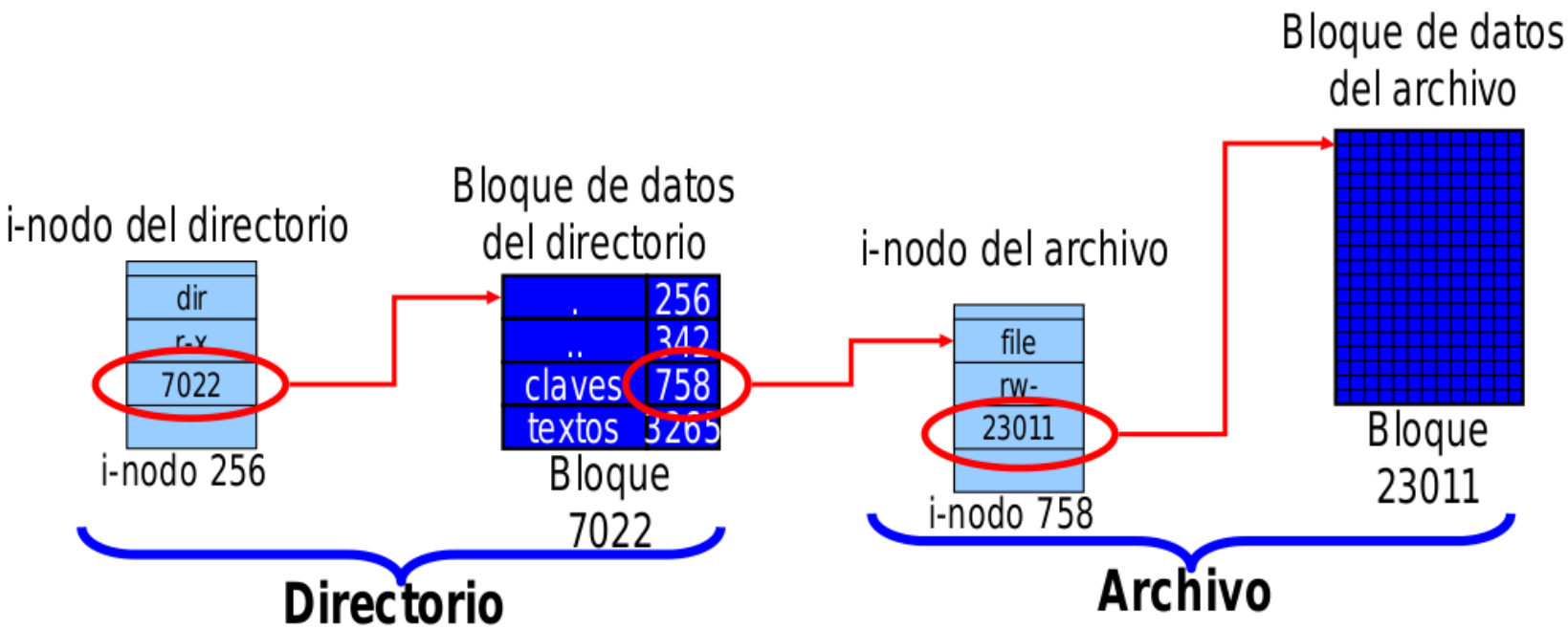


Directorios

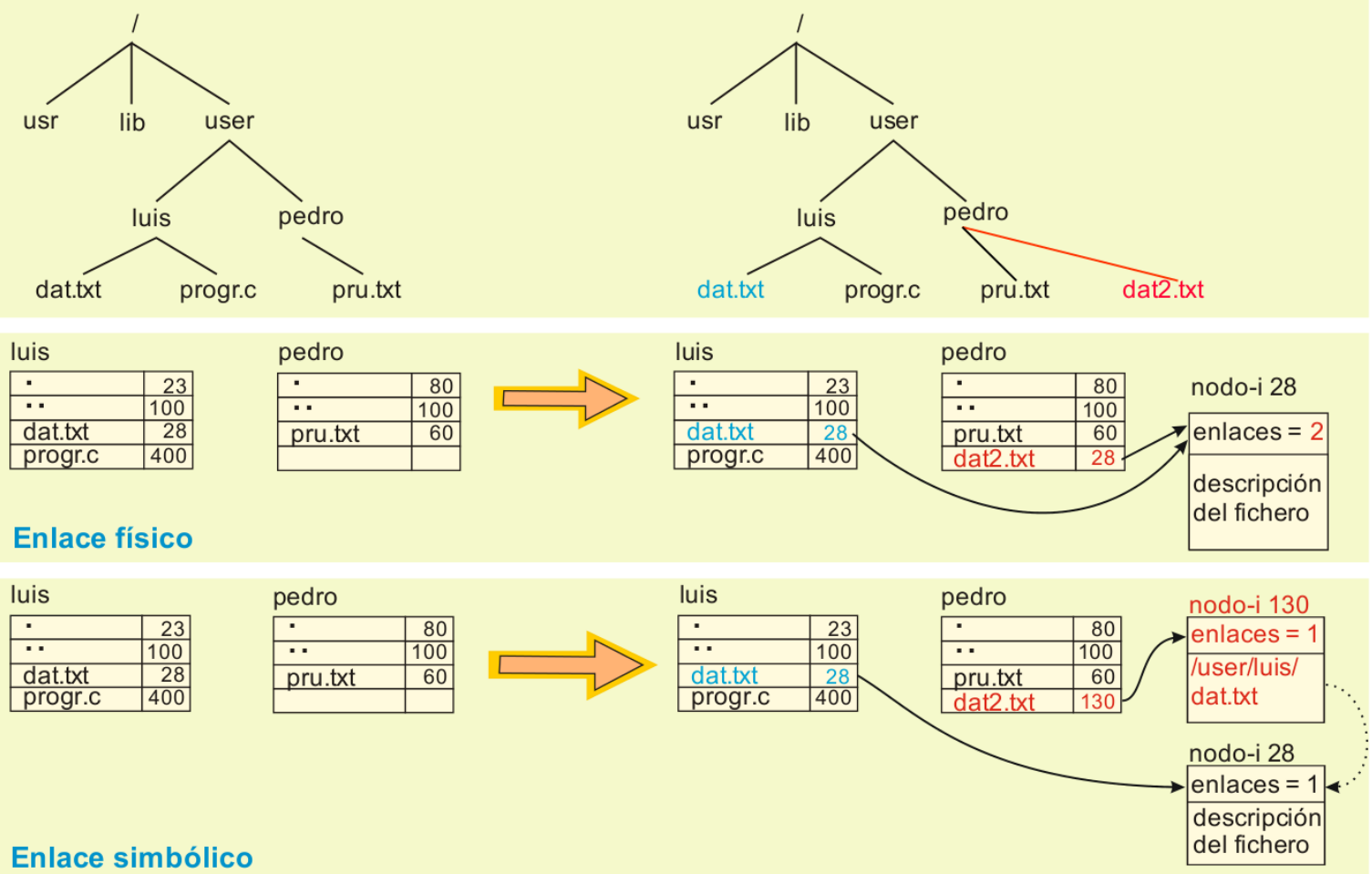
- La ruta `/users/luis/claves` se interpreta de forma recursiva:
  1. Traer a memoria bloque del i-nodo 2 (i-nodo raíz, conocido).
  2. Se busca dentro `users` y se obtiene el i-nodo 342.
  3. Traer a memoria bloque del i-nodo 342.
  4. Se busca dentro `luis` y se obtiene el i-nodo 256.
  5. Traer a memoria bloque del i-nodo 256.
  6. Se busca dentro `claves` y se obtiene el i-nodo 758.
  7. Al leer el i-nodo 758, se detecta que es un fichero y ya se tienen dónde están los datos del archivo.
  8. Leer los bloques del fichero.
- ¿Cuándo parar?
  - No se tienen permisos.
  - Se ha encontrado el i-nodo del archivo.
  - No se encuentra el siguiente elemento de la ruta.

Directorios

- La llamada `open()` termina con la lectura del i-nodo.
- La verificación de permisos se hace con los datos del i-nodo.
- Un directorio no es un i-nodo:



Enlaces


/
usr
lib
user
luis
pedro
dat.txt
progr.c
pru.txt
dat2.txt
luis
pedro
nodo-i 28
enlaces = 2
descripción del fichero
Enlace físico
nodo-i 130
enlaces = 1
/user/luis/ dat.txt
Enlace simbólico

Asignación de bloques

- Asignación: cómo se hace la correspondencia entre los bloques físicos del disco y los bloques lógicos del archivo.
- Mecanismos de asignación:
  - Asignación de bloques contiguos:
    - Todos los bloques del archivo se encuentran contiguos en el disco.
      ☺ Muy sencillo de implementar.
      ☺ Accesos secuencial y directo muy rápidos.
      ☹ Necesario saber el tamaño del archivo al crearlo.
      ☹ Fragmentación del disco.
      ☹ Para añadir datos al archivo, puede que haya que moverlo.
    - Por todo ello, no se utiliza.

Asignación de bloques

- Mecanismos de asignación:
  - Asignación de bloques no contiguos:
    - Los bloques del archivo se encuentran en cualquier posición del disco.
      ☺ Se produce menos fragmentación → el primer bloque asignado es el primero que hay libre.
      ☹ Es necesario traducir el número de bloque lógico al número de bloque en el dispositivo.
    - Es la opción utilizada en la mayoría de SOs.
- Para tener constancia de qué bloques no contiguos pertenecen a cada archivo, se utilizan listas enlazadas o índices (que pueden ser multinivel).
  - ISO9660: Inicio y tamaño (fichero contiguo).
  - SF MS-DOS: FAT (fichero enlazado).
  - SF UNIX: i-nodo (fichero indexado).
  - NTFS: Registro Windows (fichero indexado).

Asignación de bloques

Lista enlazada

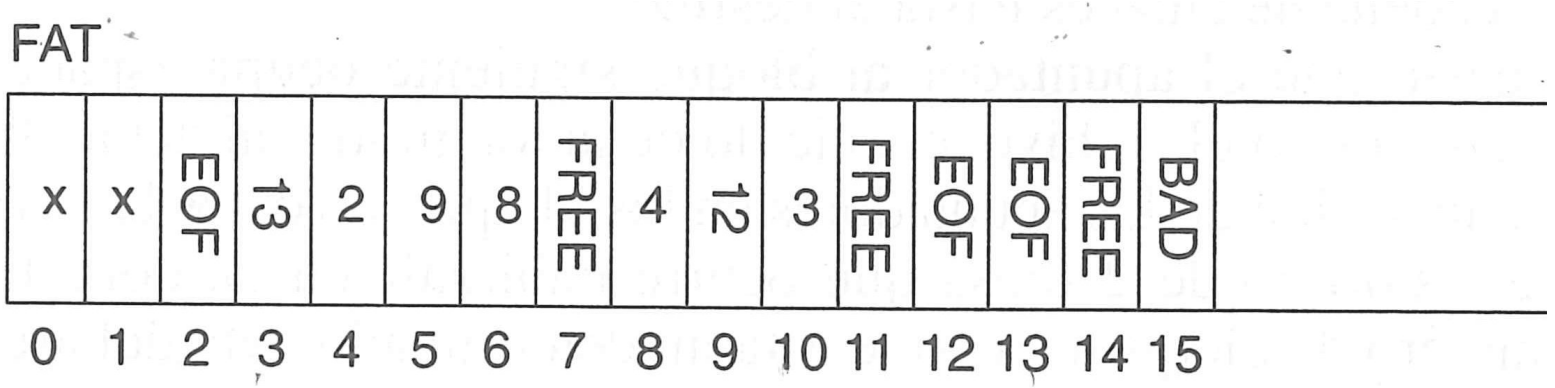
- Cada bloque tiene un apuntador al siguiente bloque que seguiría en el archivo.
- El descriptor del archivo solo debe incluir la referencia al primer bloque.
  ☺ El acceso secuencial es muy rápido.
  ☹ El acceso aleatorio a un bloque concreto de un archivo es muy costoso.
  ☹ Cada bloque incluye un apuntador que aumenta su tamaño (y complica el cálculo de espacio libre).
  ☹ La pérdida de un bloque supone perder el archivo completo.

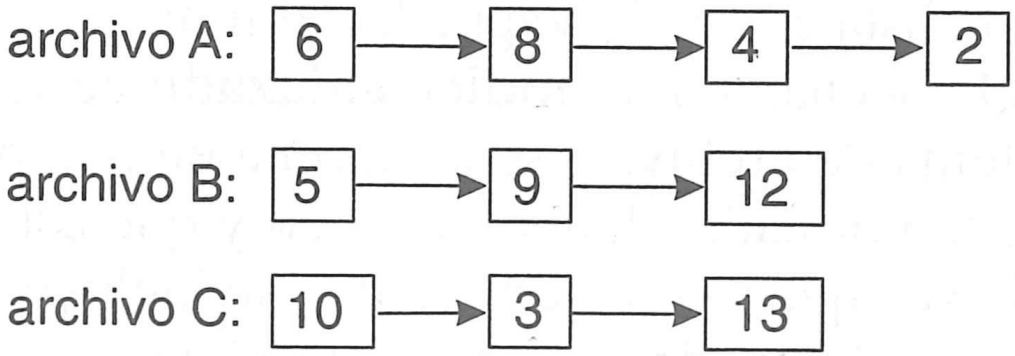
Asignación de bloques

Tabla de asignación de archivos

- Es una variación del método lista enlazada.
- Los apuntadores se almacenan en una tabla independiente de los bloques (*File Allocation Table*, FAT).
- La tabla posee una entrada por cada bloque del SA.
- La FAT ocupará un espacio prefijado en la partición.
- Descriptor fichero $\rightarrow$ incluye su primera posición en la tabla.
- Acceso aleatorio al archivo: recorriendo la tabla.
- La tabla se aloja en caché para mejorar las prestaciones y se mantiene una copia doble en el disco para mayor fiabilidad.
  ☹ ¡La FAT puede llegar a ocupar mucho! $\rightarrow$ agrupaciones.

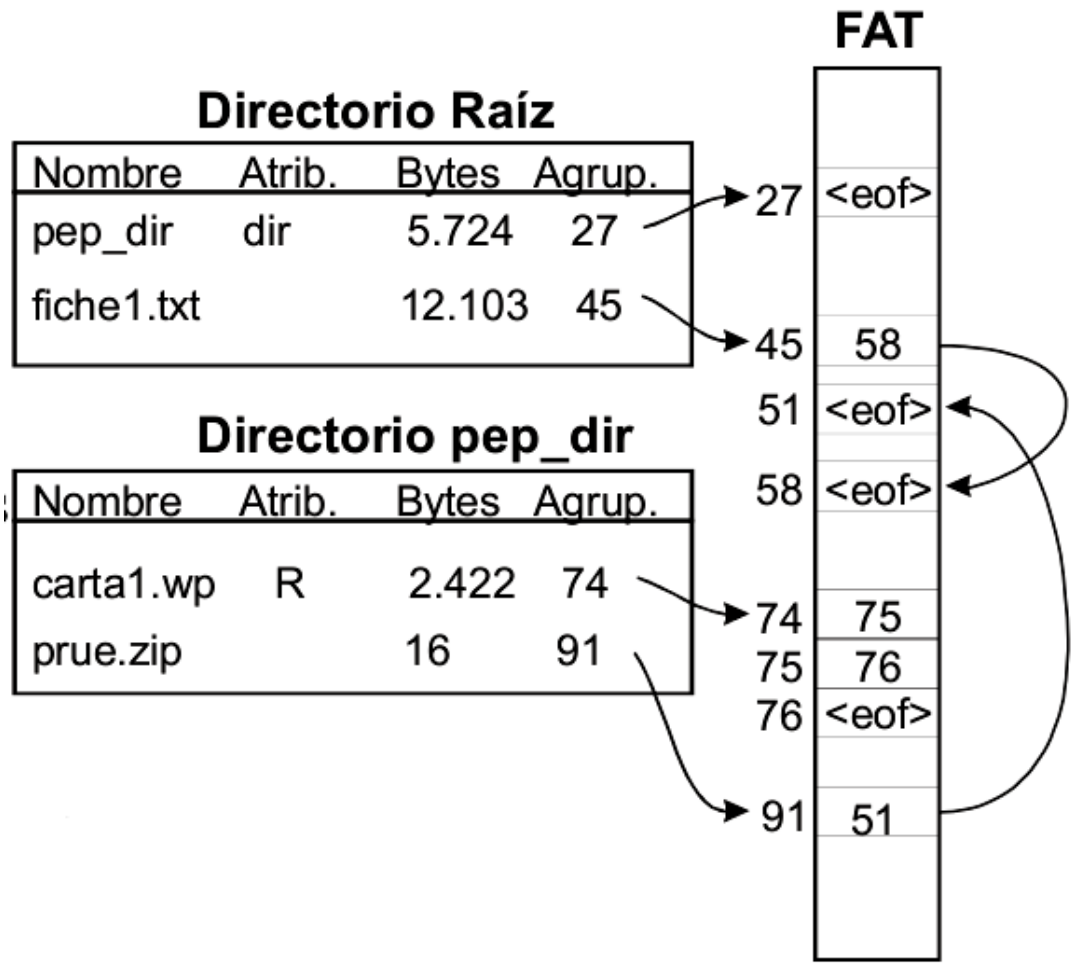
Servidor de Archivos





Servidor de Archivos

- FAT de 12 bits: 4K agrupaciones.
- FAT de 16 bits: 64K agrupaciones.
- FAT de 32 bits: 2^{28} agrupaciones (solo usa 28 bits).
  Tamaño de fichero en directorios $\rightarrow$ 32 bits.
  Tamaño máximo $\rightarrow 2^{32} - 1 = 4GB - 1$



Asignación de bloques

Índices

- Los punteros a los bloques están juntos y contiguos en una localización concreta → Bloques índice.
- Cada archivo tiene un bloque índice.
- Para buscar el *i*-ésimo bloque de un fichero, buscamos la *i*-ésima entrada en su bloque índice
  ☺ Buen acceso directo.
  ☺ Se evita la fragmentación.
  ☹ ¿Tamaño del bloque índice? → debe fijarse un número de entradas y hay que reservar espacio para todas ellas.
  ☹ Limitamos el tamaño máximo de los archivos.

Asignación de bloques

Índice multinivel

- Consiste en introducir n niveles de apuntadores, de manera que los apuntadores del descriptor apuntan a otros.
- Índice multinivel de nivel 1: el bloque índice apunta a otros bloques índices que finalmente apunta a un bloque de datos del fichero.

  ☺ Evita tener que prefijar el tamaño del bloque índice (podemos poner apuntadores a NULL).

  ☺ El bloque índice tendrá un número pequeño de entradas.

  ☹ Cada nivel, supone un acceso a disco adicional.

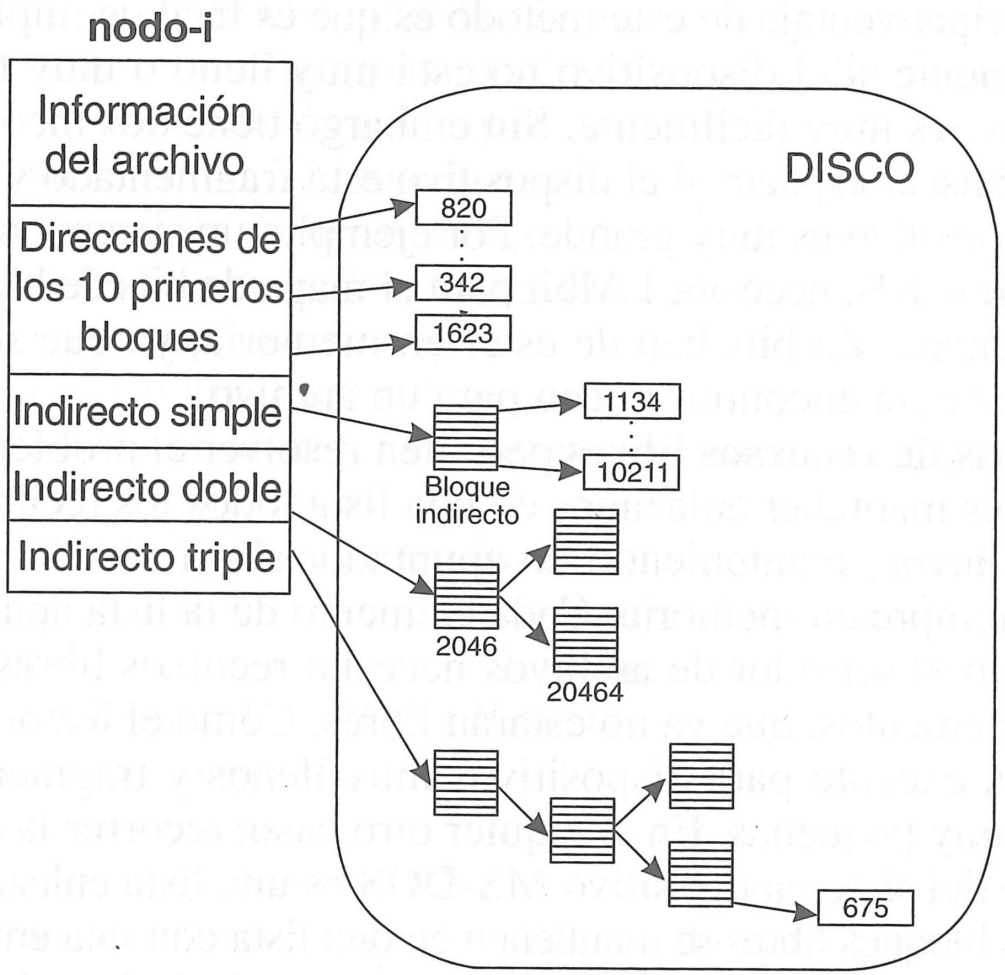
  ☹ Para archivos pequeños, se desaprovechan muchos bloques índice.

Asignación de bloques

Solución UNIX → Esquema híbrido

- Por cada nodo-i incluir:
  - Punteros directos a los 10 primeros bloques (para archivos pequeños).
  - Puntero a un bloque índice de primer nivel (donde encontraremos punteros a bloques).
  - Puntero a un bloque índice de segundo nivel (donde encontraremos punteros a punteros a bloques).
  - etc.
- Bloques del disco: bloques de datos o bloques índice.
- En ext4 y en NTFS existen los extents (bloques índice especiales que marcan una zona contigua del disco "numeroBloqueInicial, numeroBloques").

Asignación de bloques: indexado multinivel



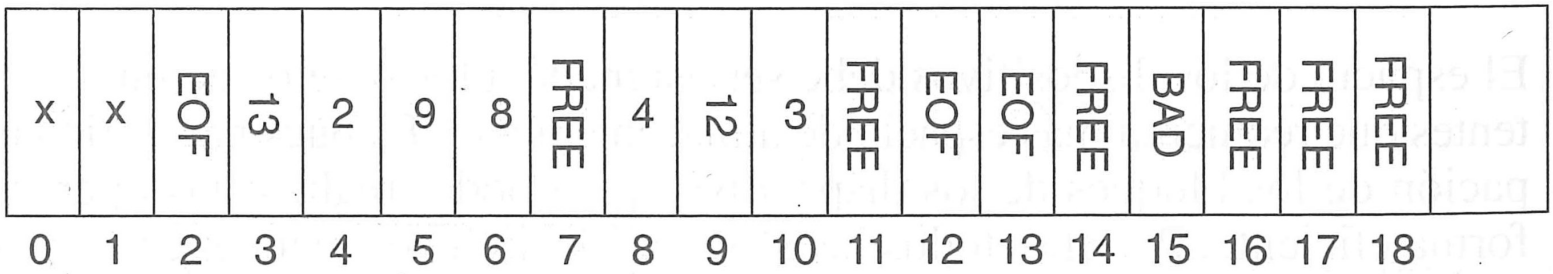
Gestión del espacio libre

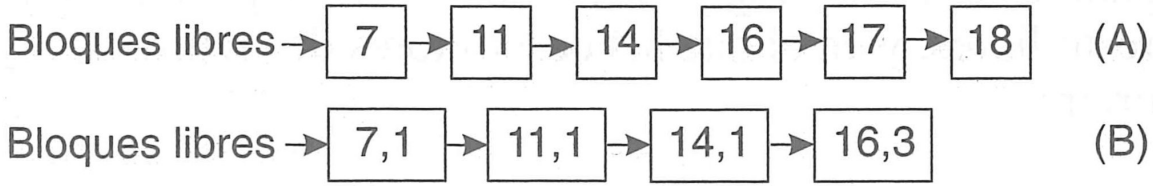
- Gestión del espacio libre: se necesita para asignar espacio a los archivos nuevos o a los que se les desea añadir datos.
- Se mantienen mapas de recursos, implementados como mapas de bits o listas de recursos libres.
  - Mapas de bits
    - Se incluye un bit por recurso (descriptor de archivo, bloque o agrupación), que será 1 si el recurso esta libre y 0 en caso contrario.
      ☺ Muy sencillo de implementar y de usar.
      ☺ Disco poco fragmentado → bloques libres al final, búsqueda muy eficiente.
      ☹ Disco fragmentado → búsqueda más costosa.
      ☹ Espacio adicional requerido por el mapa.
    - FAT: la propia tabla actúa como mapa de recursos.

Gestión del espacio libre

- Gestión del espacio libre.
  - Listas de recursos libres
    - Mantener una lista de apuntadores a los recursos libres.
    - Al ocupar un recurso lo borramos de la lista.
      ☺ Muy eficiente para discos muy llenos y fragmentados.
      ☹ Disco con mucho espacio libre → Ineficiente debido a que hay que cargar la lista.
    - Solución → Incluir número de bloques libres consecutivos en la lista.

Gestión del espacio libre: FAT + lista enlazada





Incremento de prestaciones

- Acceso a memoria $\rightarrow$ orden de nanosegundos.
- Acceso a disco $\rightarrow$ orden de milisegundos.
  - Almacenamiento intermedio de los datos
    - Mantener una caché de datos en Memoria Principal (MP).
    - Aprovecha la proximidad espacial y temporal en las referencias a los datos accedidos.
    - Caché de nombres: lista con {`nombre`,`nodo-i`}.
      Si se vuelve a acceder al archivo, no hay que hacer toda la búsqueda del `nodo-i`.
    - Caché de bloques: colección de bloques leídos o escritos recientemente.
      Si se vuelve a acceder a ese bloque, no hay que cargarlo de nuevo.

Incremento de prestaciones

- Caché de bloques:
  - Si el bloque está en MP, se escribirá o leerá en MP.
  - Posteriormente, se moverán los bloques de MP al dispositivo.
  - Si la caché está llena, hay que eliminar algún bloque:
    - Políticas de reemplazo: *First In First Out* (FIFO), *Most Recently Used* (MRU), *Least Recently Used* (LRU)...
    - Lo más común es LRU: aprovecha que los bloques no utilizados durante mucho tiempo, posiblemente no volverán a ser utilizados.
      ☹ Peligroso si hay un fallo del SA.

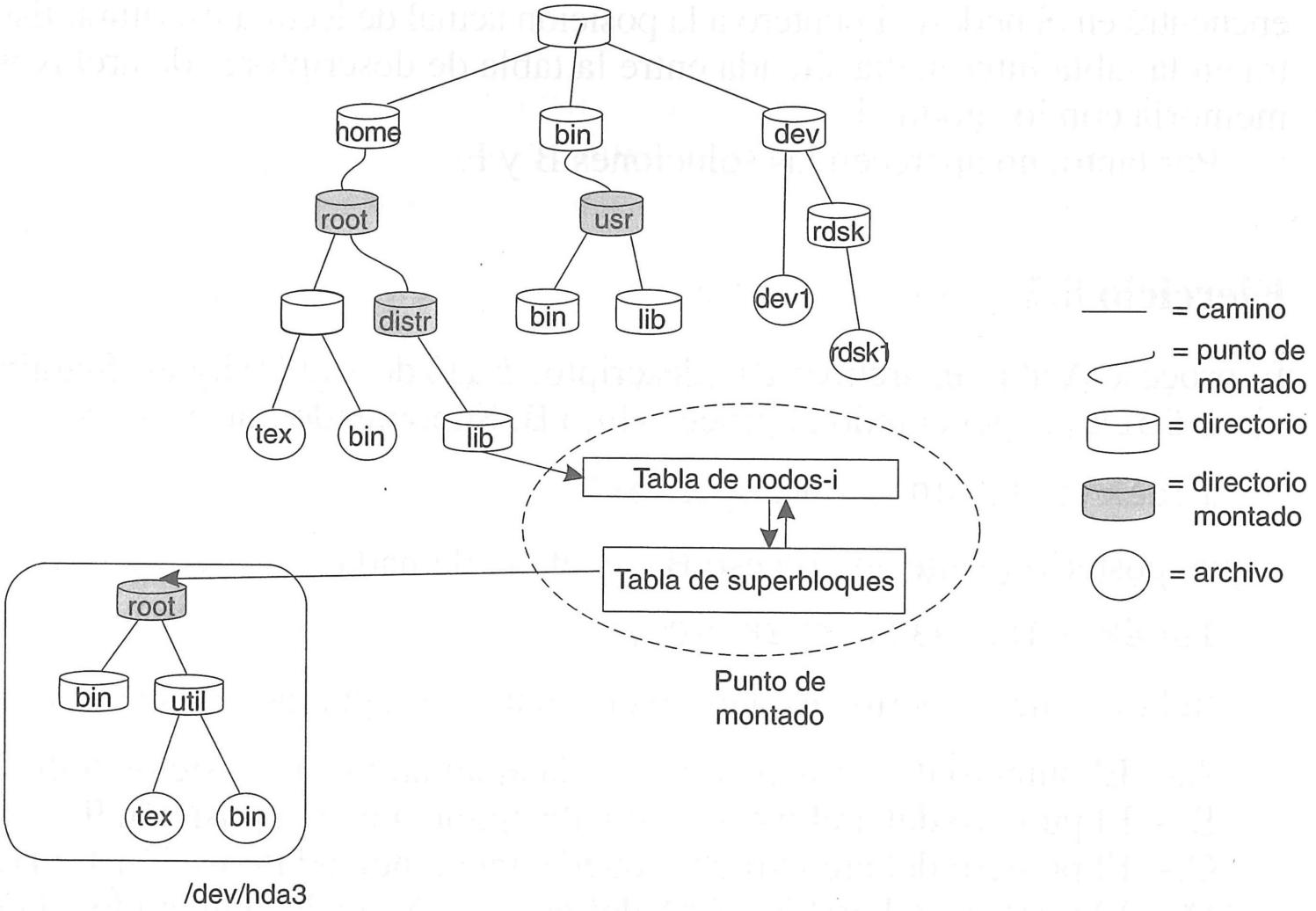
Incremento de prestaciones

- Caché de bloques:
  - Bloques sucios (cambiados en caché pero no en el disco). Distintas políticas a la hora de mantener la coherencia:
    - Escritura inmediata (*write-through*) → Siempre actualizado.
    - Escritura diferida (*write-back*) → Actualizamos cuando el bloque salga de la caché.
    - Escritura periódica (*delayed-write*) → Establecer un tiempo periódico para las actualizaciones.
      Compromiso entre rendimiento y fiabilidad
      Reduce la extensión de los posibles daños por caídas.
  - Se puede distinguir entre bloques especiales (directorios, nodos-i o bloques índice) y bloques de datos.
    Bloques especiales → *write-through*.
  - No se debe quitar un disco del sistema sin antes volcar los datos de la cache (comando `sync`).

Montar y desmontar sistemas de ficheros

- En GNU/Linux, hay un único sistema de ficheros lógico (o una única "jerarquía de directorios"), en la que se organizan todos los dispositivos de almacenamiento disponibles.
- Cada partición tiene su propio sistema de ficheros, con su directorio raíz y su jerarquía.
  - Montar un sistema de ficheros: añadirlo al sistema de ficheros lógico. Sus datos (desde su propia raíz) están disponibles a partir de un punto de montaje (p.ej. `/media/usb1`).
  - Desmontar un sistema de ficheros: el sistema de ficheros deja de estar disponible, dejándolo además consistente.
- Los ficheros principales del SO están siempre disponibles desde la raíz del sistema de ficheros lógico (/).
- En el arranque, se monta primero la partición correspondiente a dicha raíz (root) y luego cualquier partición auxiliar.

Montar y desmontar sistemas de ficheros



Montar y desmontar sistemas de ficheros

- `mount [opci] <FicheroEspecialBloque> <PtoMontaje>`
  - `-t tipo-sf` ⇒ tipo de sistema de ficheros.
  - `-r` ⇒ montaje en modo sólo lectura.
  - `-w` ⇒ montaje en modo lectura/escritura.
  - `-o opcionesMontaje` ⇒ opciones del proceso de montaje (`nosuid`, `exec`, `remount`, etc.).
- `umount <PtoMontaje> (ó <FicheroEspecialBloque>)` ⇒ desmontar un sistema de ficheros. Si está siendo utilizado (busy), no se podrá desmontar.
- `fuser` ⇒ saber qué ficheros se están usando y qué procesos los usan (`f`: fichero abierto, `c`: directorio de trabajo, `e`: ejecutando un fichero, etc.)
- `lsof` ⇒ obtener un listado de todos los ficheros abiertos...

Montar y desmontar sistemas de ficheros

```
1  pedroa@pagutierrezLaptop:~$ fuser -mv / # -m: ficheros montados; -v: verbose
2                       USER        PID ACCESS COMMAND
3  /:                   root     kernel mount /
4                       pedroa      2363 Frce. gnome-keyring-d
5                       pedroa      2760 Fr.e. icedove-bin
6                       pedroa      3206 Fr.e. evince
7  pedroa@pagutierrezLaptop:~$ lsof
8  COMMAND   PID   USER    FD TYPE  DEVICE SIZE/OFF     NODE NAME
9  ...
10 kile     2764 pedroa  mem REG     8,4   499320  5246682 libgcrypt.so.11.6.0
11 kile     2764 pedroa  mem REG     8,4   659656  5775741 libgnutls.so.26.14.12
12 kile     2764 pedroa  mem REG     8,4    68416  5775700 libavahi-client.so.3.2.9
13 kile     2764 pedroa  mem REG     8,4    47448  5775702 libavahi-common.so.3.5.3
14 kile     2764 pedroa  mem REG     8,4    14480  5775706 libavahi-glib.so.1.0.2
15 kile     2764 pedroa  mem REG     8,4     4016  5770905 libcanberra.so.0.2.5
16 kile     2764 pedroa  mem REG     8,4    31304  5771046 libgailutil.so.18.0.1
17 ...
18 lsof     3271 pedroa  cwd DIR     8,4     4096  6301896 /home/pagutierrez
19 lsof     3271 pedroa  rtd DIR     8,4     4096        2 /
20 lsof     3271 pedroa  txt REG     8,4   131312  5767875 lsof
21 lsof     3271 pedroa  mem REG     8,4    68352  5774521 locale-archive
22 lsof     3271 pedroa  mem REG     8,4   642216  5248383 libc-2.13.so
23 lsof     3271 pedroa  mem REG     8,4   141088  5242884 ld-2.13.so
```

Montar y desmontar sistemas de ficheros

- `/etc/fstab`: fichero con información sobre todos los sistemas de ficheros a montar (o ya montados) y las zonas de intercambio a activar.

  `fi_especial pto tipo opciones dump_freq pass_num`

  - `fi_especial` ⇒ fichero especial de bloques (`/dev/...`).
  - `pto` ⇒ directorio que sirve de punto de montaje (¿permisos?).
  - `tipo` ⇒ tipo de SF (ext2, ext3, ext4, vfat, iso9660, swap, ntfs, nfs, etc.).
  - Opciones para el proceso de montaje (separadas por "," y sin espacios).
  - `dump_freq` ⇒ "frecuencia del dump" para hacer una copia de seguridad de ese SF mediante el comando `dump` (no se usa).
  - `pass_num` ⇒ en tiempo de arranque, en qué orden hay que chequear los SFs (ejecutar `fsck` para comprobar su estado).

Opciones de `fstab` I

- Opciones del fichero `/etc/fstab`:
  - `rw` ⇒ Lectura-escritura (por defecto).
  - `ro` ⇒ Sólo lectura
  - `suid/nosuid` ⇒ Permitido (o no) que los bits suid o sgid tengan efecto.
  - `auto/noauto` ⇒ Montar automáticamente (o no) (ejecutando `mount -a` ⇒ siempre se ejecuta al arrancar el sistema).
  - `exec/noexec` ⇒ Permitir (o no) la ejecución de ficheros.
  - `usrquota, grpquota` ⇒ Activar cuotas.
  - `uid=500, gid=100` ⇒ Propietario y grupo propietario de los ficheros del SF (si el SF no incorpora esta información).
  - `umask=137` ⇒ Máscara para los permisos de los ficheros (permisos 640) (si el SF no incorpora esta información o si se quieren cambiar).

Opciones de `fstab` II

- `dev` ⇒ Interpretar ficheros especiales en el sistema de archivos.
- `sync` ⇒ Forzar a que todas las operaciones sean síncronas (puede disminuir el tiempo de vida de la unidad de disco).
- `user` ⇒ permite que los usuarios puedan montar el sistema de ficheros. Solo el mismo usuario podrá desmontarlo. Implica las opciones `noexec`, `nosuid` y `nodev`.
- `users` ⇒ igual que `user` pero cualquier usuario podrá desmontarlo.
- `nouser` ⇒ Solo `root` puede montar el SF.
- `owner` ⇒ permite que un usuario pueda montar el sistema de ficheros, siempre que sea dueño del fichero de dispositivo. Implica las opciones `nosuid` y `nodev`.
- `defaults` ⇒ `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`.

Montar y desmontar sistemas de ficheros

- Ejemplo de contenido del fichero `/etc/fstab`:

```
1 LABEL=/     /               ext3     defaults,usrquota  0 1
2 /dev/sda3   /windows        vfat     defaults           0 0
3 /dev/dvd    /media/dvd      iso9660  noauto,owner,ro    0 0
4 /dev/fd0    /media/floppy   vfat     noauto,uid=500     0 0
5 /dev/sda4   /otrolinux      ext3     defaults           0 2
6 /dev/sda2   swap            swap     defaults           0 0
```

- Al ejecutar `mount` como *root*:
  - `mount /media/dvd`: coge las opciones que faltan del fichero.
  - `mount -t iso9660 -r /dev/dvd /media/dvd`: no las coge.
- Si se asigna permisos de montaje a los usuarios (opciones `user`, `users` u `owner`), sólo pueden ejecutar `mount /media/dvd` (sin opciones).
- `mount -a`: montar todas las unidades que sean `auto`.
- Automontado de unidades: `udev` y `dbus`.

Comprobación del sistema de ficheros

- Durante el arranque, `fsck` o `e2fsck` chequearán la consistencia o estado del sistema de ficheros, detectando problemas e intentando repararlos.
- Se actúa sobre la estructura (no sobre el contenido):
  - Bloques que pertenezcan a varios ficheros.
  - Bloques que están marcados como libres, pero que se encuentran en uso.
  - Bloques que se encuentran marcados como en uso, pero que están libres.
  - Inconsistencias en cuanto al número de enlaces hacia un nodo-i.
  - Nodos-i marcados como libres, pero que están en uso.
  - Nodos-i marcados como en uso, pero que están libres.

Comprobación del sistema de ficheros

- Para chequear un SF siempre debe estar desmontado o montado en modo de sólo lectura.
- El SF raíz debe estar montado en modo de sólo lectura (el SF raíz no se puede desmontar, ¿por qué?).
- Si al arrancar el proceso de chequeo encuentra problemas que no puede solucionar, obliga al administrador a que realice el chequeo "a mano" ejecutando la orden `fsck` o `e2fsck` (modo monousuario).

Comprobación del sistema de ficheros

- *Journaling*: para evitar la verificación completa (`fsck`) de sistemas de ficheros de gran tamaño, que sería muy costosa, se implementa un modelo de control transaccional basado en *logging* (un diario).
  - Las suboperaciones que modifiquen los metadatos y datos de un mismo archivo se agrupa en la misma transacción.
  - Si el sistema falla, las acciones parcialmente realizadas se deshacen o completan, recorriendo el *log*.
  - No se garantiza que el sistema esté actualizado al finalizar la recuperación, sino que es consistente.
- Sistemas con esta filosofía: JFS (IBM), ext3fs y la gran mayoría de sistemas de archivos modernos.

Comprobación del sistema de ficheros

```
1  TxBegin(&tid);
2  /* ...
3     Sub-operaciones
4     ... */
5  TxCommit(tid);
6  TxEnd(tid);
```

- Por cada sub-operación que altera las estructuras de disco se escribe un registro en el *log*, que incluye las modificaciones en los *buffers* de i-nodos y de bloques.
- Cuando se ha copiado a disco (*log*) el registro de *commit*, se empiezan a procesar realmente los *buffers*.
- Después de una caída:
  - Se completan las transacciones *commited*.
  - Se descartan el resto de transacciones.

Creación del sistema de ficheros

Añadir un nuevo disco o SF:

1. Realizar la conexión física.
2. Crear un fichero especial de dispositivo (si es necesario).
3. Crear las particiones: `fdisk` (o `parted`).
4. Crear sistema de ficheros: `mke2fs -t ext2 /dev/sdb3`
5. Etiquetar la partición usando `e2label` ⇒ asigna una etiqueta al SF que se puede usar en el fichero `/etc/fstab`, en el campo `fi_especial`, mediante LABEL=etiqueta.
6. Crear el directorio que hará de punto de montaje.
7. Montar el nuevo sistema de ficheros.
8. Actualizar `/etc/fstab` con las opciones necesarias.

Creación del sistema de ficheros

- Diferencias ext2, ext3 y ext4:
  - ext3 tiene el mismo formato que ext2 pero además es transaccional: añade un registro o journal que permite recuperar la consistencia tras una caída del sistema.
  - ext4 tiene un formato similar a ext3, pero además incluye:
    - Una extensión describe un conjunto de bloques lógicos contiguos de un fichero que también se encuentran contiguos en disco: muy útil para ficheros grandes.
    - Se retrasa la reserva de bloques de disco hasta que se va a escribir en él: mayor número de bloques contiguos en disco.
    - Implementa una herramienta de desfragmentación *online*, `e4defrag` (capaz de funcionar mientras se usa el SF).
    - Manejo de sistemas de ficheros y ficheros de mayor tamaño.

Creación del sistema de ficheros

- ¿Qué sistema elegir?
  - ext2 ⇒ muy rápido en general, pero no tiene *journaling*. Se puede usar en un SF en el que se guardarán ficheros temporales.
  - ext3 ⇒ buen rendimiento en general y *journaling*.
  - ext4 ⇒ menor uso del CPU y mayor rapidez en los procesos de lectura y escritura que ext3. Estándar de facto en Linux.

Creación del sistema de ficheros

- `tune2fs` ⇒ Conocer y ajustar parámetros de un SF ext4/ext3/ext2.
  - `-l dispositivo`: Listar el contenido del superbloque del SF.
  - `-c max-mount-counts dispositivo`: Establecer el nº de montajes máximo sin realizar un fsck.
  - `-i numero[d|m|w] dispositivo`: Indicar el tiempo máximo entre dos chequeos.
  - `-L etiqueta dispositivo`: Poner una etiqueta al sistema de ficheros.
  - `-m porcentaje dispositivo`: Fijar el porcentaje de bloques reservados para procesos especiales (de `root`). Por defecto, 5 %.

Parámetros del SF ext4 I

```
1  pedroa@pedroa-laptop ~ $ sudo tune2fs -l /dev/sda2
2  tune2fs 1.42.8 (20-Jun-2013)
3  Filesystem volume name:   ROOT
4  Last mounted on:          /
5  Filesystem UUID:          d73f541a-e887-4885-b42b-98dd433e99de
6  Filesystem magic number:  0xEF53
7  Filesystem revision #:    1 (dynamic)
8  Filesystem features:      has_journal ext_attr resize_inode dir_index filetype
       needs_recovery extent flex_bg sparse_super large_file huge_file uninit_bg
       dir_nlink extra_isize
9  Filesystem flags:         signed_directory_hash
10 Default mount options:    user_xattr acl
11 Filesystem state:         clean
12 Errors behavior:          Continue
13 Filesystem OS type:       Linux
14 Inode count:              3145728
15 Block count:              12582912   Reserved block count:     629142
16 Free blocks:              7726646
17 Free inodes:              2545229
18 First block:              0
19 Block size:               4096
20 Fragment size:            4096
21 Reserved GDT blocks:      1021
22 Blocks per group:         32768
23 Fragments per group:      32768
```

Parámetros del SF ext4 II

```
24 Inodes per group:          8192
25 Inode blocks per group:    512
26 RAID stride:               32710
27 Flex block group size:     16
28 Filesystem created:        Thu Mar 27 20:07:38 2014
29 Last mount time:           Sun Mar 22 08:50:58 2015
30 Last write time:           Fri Aug 29 21:26:46 2014
31 Mount count:               391
32 Maximum mount count:       -1
33 Last checked:              Fri Aug 29 21:26:46 2014
34 Check interval:            0 (<none>)
35 Lifetime writes:           922 GB
36 Reserved blocks uid:       0 (user root) Reserved blocks gid:      0 (group root)
37 First inode:               11
38 Inode size:                256
39 Required extra isize:      28
40 Desired extra isize:       28
41 Journal inode:             8
42 First orphan inode:        813409
43 Default directory hash:    half_md4
44 Directory Hash Seed:       c5d4f69f-cea8-4574-894b-166152ae5a40
45 Journal backup:            inode blocks
```

Concepto de cuotas de disco

Cuotas de disco

Permiten limitar el número de bloques y/o ficheros (nodos-i) que un usuario puede usar en una partición (también se pueden establecer para grupos de usuarios).

- Hay dos tipos de límites:
  - Límite *hard*: el usuario no puede sobrepasarlo. Si lo hace, ya no podrá usar más bloques o crear más ficheros.
  - Límite *soft*: es inferior al límite *hard* y se puede sobrepasar durante cierto tiempo, siempre que no se alcance el límite *hard*.
- Periodo de gracia: tiempo durante el que se puede sobrepasar el límite *soft*. Se informa al usuario de que ha superado el límite y que debe liberar espacio o nodos-i (ficheros).
  - Los periodos y los límites se establecen, de forma independiente, para bloques y nodos-i.

Establecer cuotas de disco I

- Pasos a realizar para establecer las cuotas de disco:
  1. Indicarlo en `fstab` (diferente en `ext3` y `ext4`):

```
1 /dev/sdb1 /home    ext4   defaults,usrjquota=aquota.user,grpjquota=
      aquota.group,jqfmt=vfsv0   0   2
```

```
1 /dev/sdb1 /home    ext3   defaults,usrquota,grpquota 0   2
```

  2. Crear ficheros de control de cuotas vacíos (aquota.user y aquota.group) en la raíz de la partición:

```
1 touch aquota.user aquota.group
2 chmod 600 aquota.*
```

  3. Remontar la partición para que se activen las opciones: `mount -o remount /home`.

Establecer cuotas de disco II

4. `quotacheck -avugm`: añade el contenido de los ficheros de control de cuotas.
   - a: todos los dispositivos con cuotas.
   - v: *verbose*.
   - u: cuotas para usuarios.
   - g: cuotas para grupos.
   - m: no remontar los archivos en modo solo lectura.
5. Activar las cuotas: `quotaon -avug`.
6. Desactivarlas: `quotaoff -avug`.
7. Editar la cuota del usuario pagutierrez: `edquota pagutierrez`

```
1 Cuotas de disco para user pagutierrez (uid 1008):
2   Sist. arch. bloques    blando      duro inodos blando  duro
3   /dev/sdb1  39884712 130000000 140000000 507523      0     0
```

Establecer cuotas de disco III

8. Establecer el periodo de gracia: `edquota -t`

```
Período de gracia antes de imponer límites blandos para users:
La unidad de tiempo puede ser: días, horas, minutos, o segundos
  Sist. arch.          Periodo gracia bloque   Periodo gracia inodo
  /dev/sdb1                        7día                   7día
```

9. Copiar cuotas: `edquota -up pagutierrez jsanchez`
10. Estadísticas de las cuotas: `repquota /dev/sdb1`

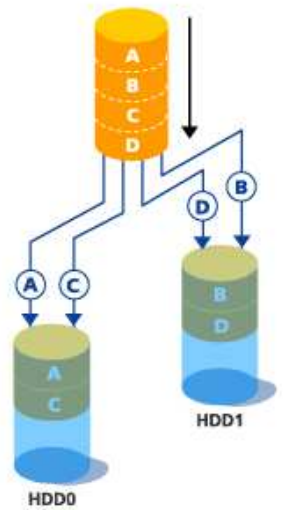
```
*** Informe para user quotas en dispositivo /dev/sdb1
Periodo de gracia de bloque: 7días; periodo de gracia de inodo: 7
     días
                        Límites de bloque límites de archivo
Usuario         usado       blando      duro        usado blando duro
-----------------------------------------------------------------------

root      --   10093264          0         0       199316      0      0
pagutierrez -- 39884712 130000000 140000000       507523      0      0
i22fenaf  --   31940744 130000000 140000000       864578      0      0
jsanchezm  --  31940744 130000000 140000000       864578      0      0
```

Administración de volúmenes dinámicos

- RAID: Array redundante de discos independientes.
  - Varias unidades de disco se ven como una sola unidad lógica.
  - Se pueden implementar por *software* o por *hardware*.
- *Logical Volume Management* (LVM): agrupar las particiones en volúmenes. Permite redimensionar y mover particiones.

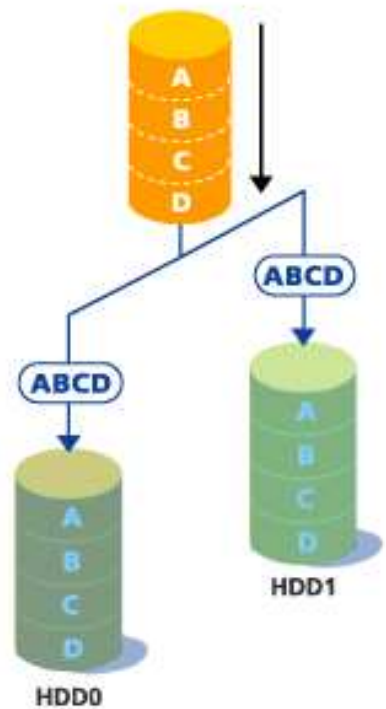
- RAID nivel 0:
  - Expande la información en diversos discos, que se ven como un único SF.
  - Aumenta el espacio según el número de discos usado.
  - Se consigue E/S paralela en lecturas y escrituras, siempre que los bloques a tratar no sean del mismo disco.
  - No hay redundancia de datos.



(C) Fujitsu

Administración de volúmenes dinámicos

- RAID nivel 1:
  - Se utilizan dos o más discos duros, que forman un único SF (SF replicado en varios discos).
  - Son discos espejos (todos guardan la misma información).
  - **SI** hay redundancia de datos.
  - Las lecturas pueden ser en paralelo, las escrituras no.
  - Cuando uno de los discos falla el sistema sigue trabajando con el otro sin problemas.
  - La recuperación de un disco es transparente al usuario.

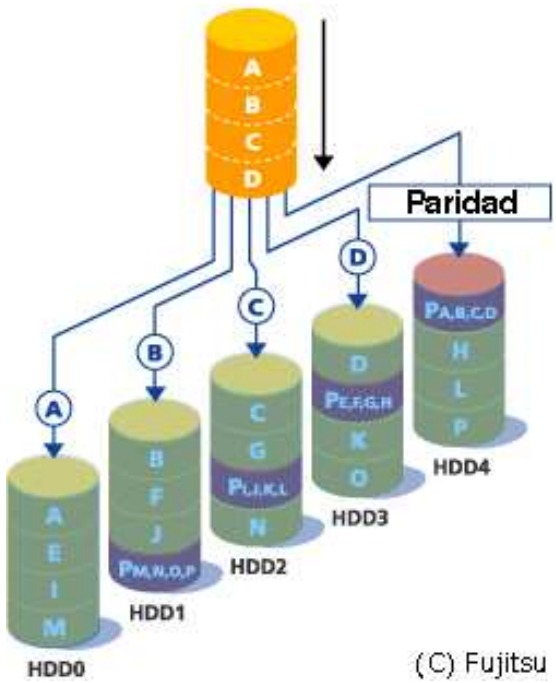


(C) Fujitsu

Administración de volúmenes dinámicos

- RAID nivel 4/5:
  - División de los datos a nivel de bloques.
  - Mínimo 3 discos duros, de los cuales 1 almacenará la paridad de los otros discos, que son usados para datos ⇒ RAID 4.
  - Problema: el disco con paridad es un cuello de botella, repartir paridad entre todos los discos ⇒ RAID 5.
  - Se consigue un dispositivo de almacenamiento más grande.
  - **SI** hay redundancia de datos.
  - Lecturas y escrituras en **paralelo**.

RAID5



Administración de volúmenes dinámicos

- Paridad:
  - Cada vez que se escriben datos, se calcula el XOR bit a bit (1 número de unos impar, 0: número de unos par) de los bloques implicados en cada cada disco.
  - Basada en operaciones XOR:

Disco 1:	00101010	(Datos)
Disco 2:	10001110	(Datos)
Disco 3:	11110111	(Datos)
Disco 4:	10110101	(Datos)
Disco 5:	11100110	(Paridad)

Administración de volúmenes dinámicos

- Paridad:
  - Si uno de los discos falla (p.ej. el disco 4), el contenido se puede restaurar a partir de la paridad:

Disco 1:	00101010	(Datos)
Disco 2:	10001110	(Datos)
Disco 3:	11110111	(Datos)
Disco 5:	11100110	(Paridad)
Disco 4:	10110101	(Datos)

Administración de volúmenes dinámicos

- Control de dispositivos de entrada/salida:
  - La herramienta `mdadm` permite crear o administrar un dispositivo RAID, convertir un disco "normal" en un RAID...
  - Tiene distintos modos de funcionamiento:
    - create: configurar y activar sistemas RAID.
  - `/proc/mdstat` lista todos los sistemas RAID (dispositivos `md`) activos con información sobre su estado.
  - Las particiones que formen el RAID tienen que un *flag* RAID (*Linux raid auto*), de esta manera serán detectadas y activadas en el proceso de arranque.

Administración de volúmenes dinámicos

- Ejemplo de creación de un RAID1.
  - Activar el *flag* RAID en la partición (tipo `fd` en `fdisk`, *flag* en gparted).
  - Crear el RAID:

```
1 mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sda1 /dev/
      sdc1
```

  - Crear un SF sobre el sistema RAID: `mke2fs -t ext3 /dev/md1`.
  - Añadirlo al fichero `/etc/fstab` para montarlo en tiempo de arranque:

```
1 /dev/md1 /home ext3 defaults 1 2
```

Administración de volúmenes dinámicos

- Creación de un RAID1 con un disco que ya tiene datos:
  - Crear el RAID con la partición que tiene los datos:

```
1 mdadm --create /dev/md2 --force --level=1 --raid-devices=1 /dev/sda4
```

  - Añadir nuevo disco al RAID como disco de repuesto (`spare`):

```
1 mdadm /dev/md2 -a /dev/sdc3
```

  - Activar el nuevo disco: `mdadm --grow /dev/md2 -n 2`.
  - A continuación, introducirlo en `/etc/fstab`.
- Información sobre el estado:

```
1 mdadm --detail --scan /dev/md1
```

- Todo esto se puede configurar utilizando el fichero `/etc/mdadm.conf`.

Referencias

Fernando Pérez-Costoya, Jesús Carretero-Pérez y Félix García-Carballeira.
Problemas de Sistemas Operativos. De la base al diseño. Tema 8. Archivos y directorios. Sección 8.4. Sistemas de Archivos.
Mc Graw Hill, Segunda Edición, 2003.

Nemeth, Snyder y Seebass.
Linux Administration Handbook Capítulos 5 y 7.
Prentice Hall. Segunda Edición. 2007.

Programación y Administración de Sistemas

6. Sistemas de ficheros y discos

Pedro Antonio Gutiérrez

Asignatura "Programación y Administración de Sistemas"
2º Curso Grado en Ingeniería Informática
Escuela Politécnica Superior
(Universidad de Córdoba)
pagutierrez@uco.es

22 de marzo de 2015