Sistemas de ficheros

Daniel García Moreno
danigm@gnome.org

Qué son los sistemas de ficheros

- La información se puede almacenar en un dispositivo sin sistema de ficheros, pero así es poco útil
- Los sistemas de ficheros nos proporcionan una forma de separar la información almacenada, ponerle nombre (ficheros) y agruparla (carpetas)

Diseñemos un sistema de ficheros simple

- Tenemos un cuaderno, con las hojas numeradas del 1 al 100
- Cómo podemos detectar final de un capítulo?
- Cómo podemos encontrar un capítulo rápidamente
- Qué pasa si queremos ampliar un capítulo?

FAT (File Allocation Table)

- Diseñado en 1977 para los disquetes
- Se usó para discos duros en MSDos y posteriormente en windows 9X
- Aún es el sistema de ficheros más usado a día de hoy para pendrives, tarjetas SD etc.
- También está en los ordenadores modernos, en el sistema de arranque UEFI
- Originalmente diseñado para 8bits, pero hay diferentes extensiones, FAT12, FAT16, FAT32

FAT (Diseño)

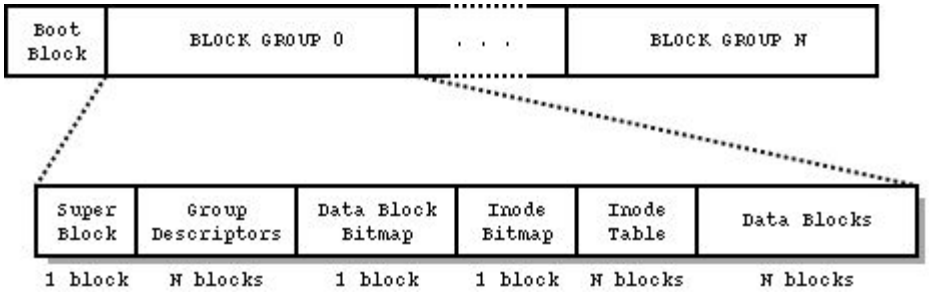
- Utiliza una tabla de indexación, File allocation table, con punteros a la zona de disco de cada trozo de fichero
- Cada entrada contiene o el número del siguiente trozo de fichero o un marcador de: final, espacio no utilizado o espacio reservado

FAT

- FAT1 y FAT2, son idénticos, una lista enlazada con las posiciones de los trozos de ficheros
- La parte de data tiene los sectores con los trozos de los ficheros
- El directorio raíz tiene la dirección del primer trozo en el FAT, a partir de ahí ya podemos leer toda la estructura del sistema de ficheros
- Cada entrada de directorio tiene: el nombre del fichero, atributos, dirección del primer trozo en el FAT de ese fichero

Ext2 (Second extended filesystem)

- Aparece en 1993
- De lo más utilizado en sistemas Linux
- Información organizada en bloques y grupos de bloques
- Cada fichero se representa por un inode (index node)



Ext2 (Block group)

- Superblock (1b): En todos los grupos la misma información, permite recuperación en caso de error
- Group descriptors: Información de los grupos
- Data block bitmap (1block): 1bit por bloque: 0 - libre, 1 - en uso
- Inode bitmap (1block): 1bit por bloque: 0 - libre, 1 - en uso
- Inode table: N bloques con los inodos uno detrás de otro, la información de cuántos nodos hay en un grupo está en el superblock
- Data blocks: N bloques con los datos

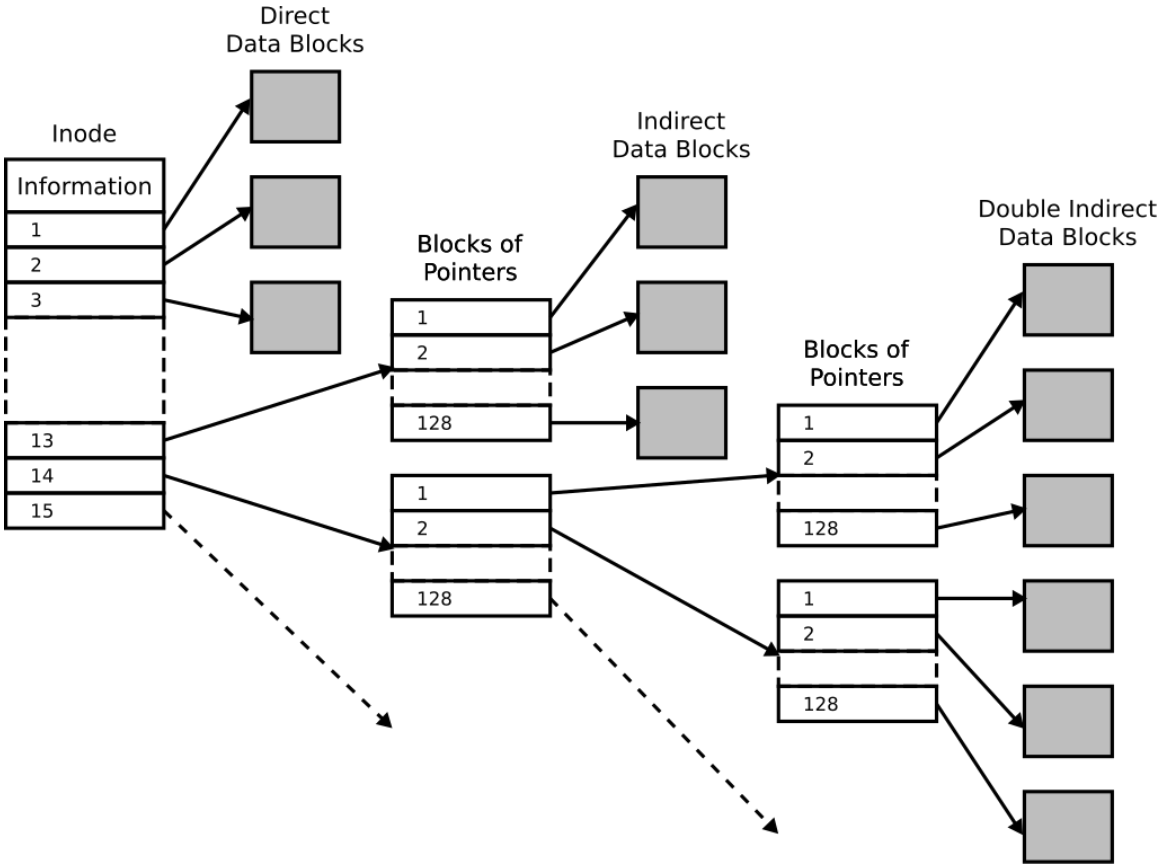
```
sudo dumpe2fs /dev/sda7
```

Ext2 Inodos

Información: modo, tamaño, fechas, etc

15 punteros:

- 12 primeros apuntan a bloques de información
- 13 apunta a un grupo (1 nivel para llegar a la información)
- 14 apunta a un grupo que apunta a grupos (2 niveles para llegar a la información)
- 15 apunta a un grupo que apunta a grupos que apuntan a grupos (3 niveles para llegar a la información)



Ext2 (Directorios)

- Cada directorio es una lista de entradas de directorio
- Una entrada de directorio asocia un nombre de fichero con un inode
- Entrada de directorio:
  - número de inodo
  - tamaño del nombre del fichero
  - nombre del fichero
- El directorio raíz está siempre en el inode 2, fácil de encontrar
- El inode 1 se usa para controlar bloques incorrectos, un fichero oculto que apunta a todos los bloques incorrectos
- Los enlaces duros son una entrada más con otro nombre apuntando al mismo inode
- Los directorios “.” y “..” son dos entradas más que apuntan al inodo del directorio actual y al padre y siempre existen

Ext2 Límites

Block size	1K	2K	4K	8K
Max file size	16G	256G	2T	2T
Max filesystem size	4T	8T	16T	32T

Ext3 (Third Extended Filesystem)

- Ext2 + journaling
- El sistema de ficheros va dejando un registro de cambios lo que permite una rápida recuperación en caso de fallo, no es necesario recorrer todo el disco buscando fallos
- Ext3 es un Ext2 válido
- Añade un índice en árbol para los directorios
- Crecimiento en línea

Ext3 (Qué es eso del journaling)

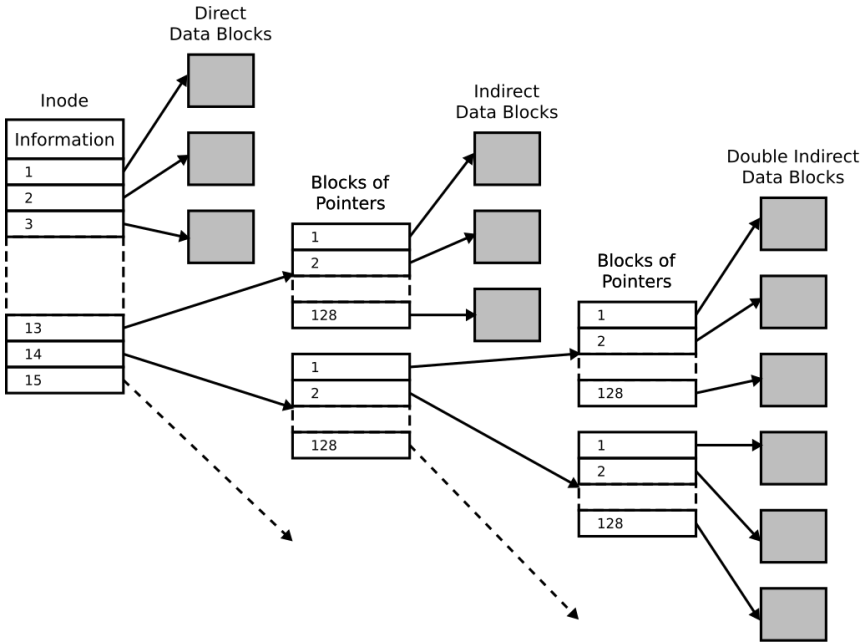
- Guarda un registro de cambios antes de realizarlos
- Si el sistema falla durante una operación de disco, es mucho más fácil corregir eso tan sólo mirando el registro
- Por qué?
  - Un cambio en disco se realiza en varios pasos, modificar la entrada de directorio, liberar el inode, liberar los bloques de datos…
  - Si ocurre un fallo durante un paso intermedio, el sistema de ficheros se queda en un estado incorrecto
  - Detectar y corregir estos fallos requiere recorrer todo el disco, el journaling acelera la recuperación tras fallos

Ext4 (Fourth Extended Filesystem)

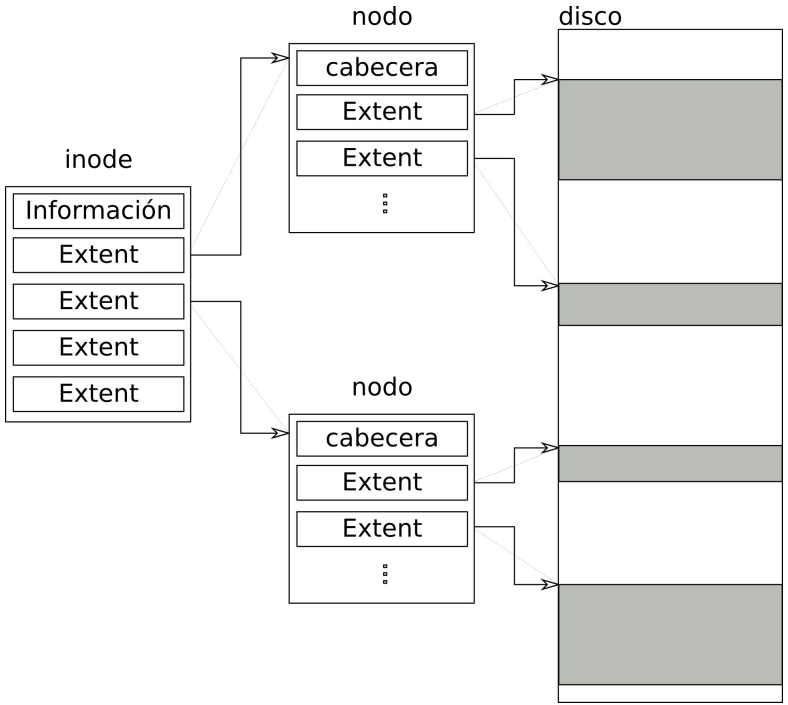
- El sucesor de ext3, utilizado por defecto a día de hoy en la mayoría de las distribuciones linux
- Nació como una serie de extensiones a ext3
- Mejoras sobre ext3:
  - Soporte para sistemas de ficheros grandes, hasta 1EiB (1.000 Peta -> 1.000.000 Tera) con 4k por bloque
  - Extents: Reemplazo del block mapping, en lugar de punteros directos a trozos de memoria, se almacenan extents, zona+tamaño. Si el fichero está desfragmentado se usa un HTree de extents
  - Compatible hacia atrás con ext3 y ext2, se pueden montar estos tipos como ext4
  - Mejoras de asignación de espacio para disminuir la desfragmentación
  - Número ilimitado de subdirectorios

Ext4 (Extents)

Ext2 / Ext3



Ext4



Demo time

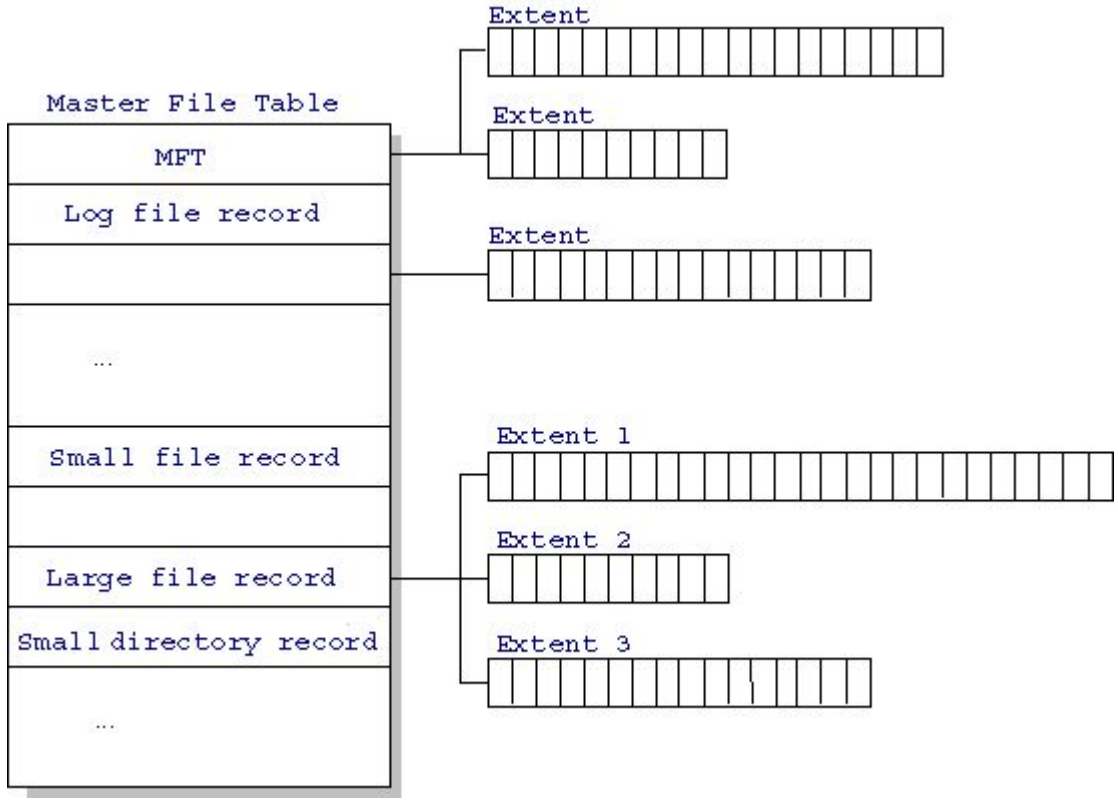
Mostrar ejemplo con “debugfs”, “stat”

- inode_dump
- block_dump, de un directorio y de un fichero

NTFS (New Technology File System)

- Es el sistema de ficheros por defecto de windows, desde NT
- Incluye journaling
- Seguridad y permisos sobre carpetas
- Cifrado
- Quotas
- Todo es un fichero, incluso los metadatos ($Boot, $MFT, $MFTMirr, $LogFile)
- El índice de ficheros es el Master File Table (MFT)

NTFS ($MFT)


Master File Table
MFT
Log file record
...
Small file record
Large file record
Small directory record
...
Extent
Extent
Extent
Extent 1
Extent 2
Extent 3

Otros sistemas de ficheros

- Base de datos
- Red
- πfs: https://github.com/philipl/pifs