Parte 4: Administración de Archivos

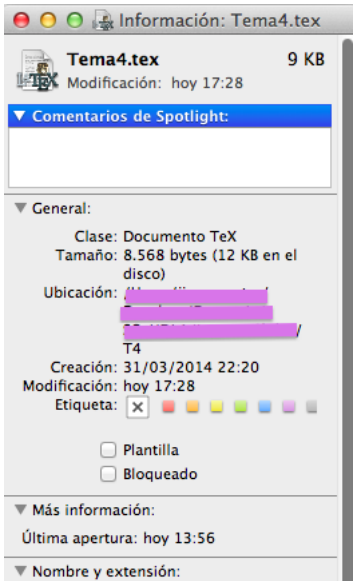
Prof. **Iván Jiménez Utiel**

Contents

2 Interfaz del Sistema de Archivos

2.1 Archivos

- Diferentes SO mantienen una serie de atributos de archivos, incluyendo:
  - Name: algunos sistemas incluyen extensiones (.exe, .txt, .png)
  - Identifier: (número de i-nodo).
  - Type: texto, ejecutable, binario, etc.
  - Location
  - Size
  - Protection
  - Time&Date
  - User ID



- Operaciones con archivos: create, write, read, deleting, truncate.
- Muchos SOs requieren que el fichero esté abierto para que pueda ser utilizado y cerrado después de su uso. Normalmente es el programador el que debe realizar estas operaciones.
- La información de los archivos abiertos actualmente es guardada en una tabla de archivos abiertos que contienen:
  - Puntero de Fichero: guarda la posición actual.
  - Contador de Fichero abierto: cuántas veces ha sido abierto y si lo está aún.
  - Ubicación en disco:
  - Derechos de Acceso.

Podemos ver en este par de ejemplos ciertas funciones sobre archivos:

```
##En Python
$ python
>>> import os
>>> path = 'server.py'
>>> f = open(path, 'rb')
>>> os.path.getsize(path)
110

>>> os.stat(path)
(33204, 37366188L, 143L, 1, 501, 501, 285L, 1402324550, 1400086770, 1400086770)
>>>

>>> os.getcwd()
'/home/vegas'
>>>

>>> os.mkdir("/home/vegas/dir1")
>>> os.listdir("/home/vegas/dir1")
[]
>>> os.mkdir("/home/vegas/dir1/dir2")
>>> os.listdir("/home/vegas/dir1")
['dir2']
>>>

## En Shell
$ stat fork_ping.py
  File: ■fork_ping.py■
  Size: 285             Blocks: 8          IO Block: 4096   fichero regular
Device: 8fh/143d        Inode: 37366188    Links: 1
Access: (0664/-rw-rw-r--)  Uid: (  501/ijimenez)   Gid: (  501/ijimenez)
Access: 2014-06-09 18:35:50.610411656 +0400
Modify: 2014-05-14 20:59:30.442261098 +0400
Change: 2014-05-14 20:59:30.443261098 +0400
```

- Windows (y otros sistemas utilizan extensiones de fichero especiales para indicar qué tipo de fichero utilizan).

file type	usual extension	function
executable	exe, com, bin or none	ejecutar programas
object	obj, o	lenguaje máquina
source code	c, cc, java, perl, asm	código fuente
batch	bat, sh	scripts
markup	xml, html, tex	documentos marcas
word processor	xml, rtf, docx	documentos
library	lib, a, so, dll	librerias
print or view	gif, pdf, jpg	ficheros en binario o ascii para imprimir
archive	rar, zip, tar	archivado y/o compresión
multimedia	mpeg, mov, mp3, mp4, avi	información de audio y video

- MACOS almacena un atributo por cada fichero de acuerdo al programa que lo creó.
- UNIX/LINUX/POSIX almacena número mágicos al comienzo de ciertos archivos. Probemos el comando **file:**

```
$ file /dev/ram
/dev/ram: symbolic link to 'ram1'

$ file /dev/pts/0
/dev/pts/0: character special (136/0)

$ file /bin/grep
/bin/grep: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.9,
    dynamically linked (uses shared libs), stripped
```

Ej. Magic Numbers:
->PDF comienza con"%PDF" (hex 25 50 44 46).
-> Ficheros LLVM comienzan por BC (0x42, 0x43)
-> Ficheros WAD comienza por WAD o PWAD (Doom), WAD2 (Quake) y WAD3 (Half-Life).

2.2 Estructura de Directorios

- Un disco puede ser utilizado en su totalidad por un sistema de archivos.
- Alternativamente un disco físico puede ser dividido en varias particiones, porciones, o mini discos. Cada uno de los cuales puede ser un disco virtual y puede tener su propio sistema de archivos (o también utilizado para **raw store** o **swap space.**
- También puede ser que discos físicos múltiples puedan ser combinados en un **volumen** como si fueran un disco más grande, con su propio sistema de archivos abarcando todos los discos físicos.

http://bit.ly/2i1BnSL

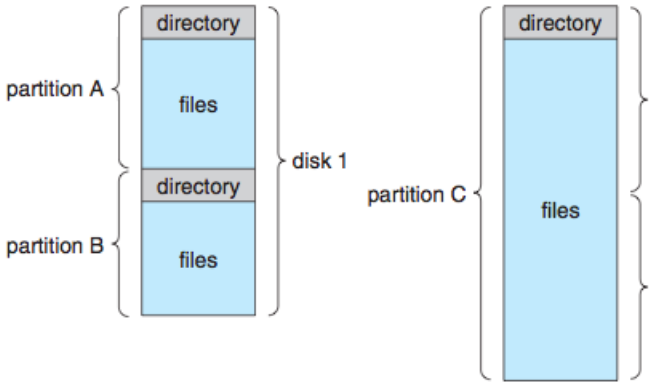


Figure 11.7 Organización tipica de sistema de ficheros

/	ufs
/devices	devfs
/dev	dev
/system/contract	ctfs
/proc	proc
/etc/mnttab	mntfs
/etc/svc/volatile	tmpfs
/system/object	objfs
/lib/libc.so.1	lofs
/dev/fd	fd
/var	ufs
/tmp	tmpfs
/var/run	tmpfs
/opt	ufs
/zpbge	zfs
/zpbge/backup	zfs
/export/home	zfs
/var/mail	zfs
/var/spool/mqueue	zfs
/zpbg	zfs
/zpbg/zones	zfs

Figure 11.8 Sistema de Ficheros Solaris (UNIX)

2.2.1 Árbol de los Directorios

- El uso de una estructura de directorios en árbol minimiza la dificultad en asignar nombres únicos.
- Cada archivo del sistema puede ser accedido en el path (o camino) desde el root (también directorio máster o raíz) bajando por las distintas ramas hasta llegar al archivo.
- El caminio que va desde el raíz hasta el archivo, se llama **pathname.**
- Ejemplo de pathname para el archivo ABC: **/User_B/Word/Unit_A/ABC**
- Ejemplo de pathname para el archivo ABC: **/User_B/Draw/ABC**
- Nótese que están en distintos pathnames aunque sean iguales los archivos.

Es interesante la utilización de estos dos comandos en Linux sobre uso y estadísticas de disco

```
# Comando df espacio utilizado en los discos
$ df -h
Filesystem      Size   Used  Avail Capacity  Mounted on
/dev/disk0s2    74Gi   62Gi   11Gi    85%    /
devfs          111Ki  111Ki    0Bi   100%    /dev
map -hosts       0Bi    0Bi    0Bi   100%    /net
map auto_home    0Bi    0Bi    0Bi   100%    /home

# Estadísticas de uso del disco

$ du -ah
8,0K    ./.DS_Store
 16K    ./ipcop_network.png
4,0K    ./p21.aux
```

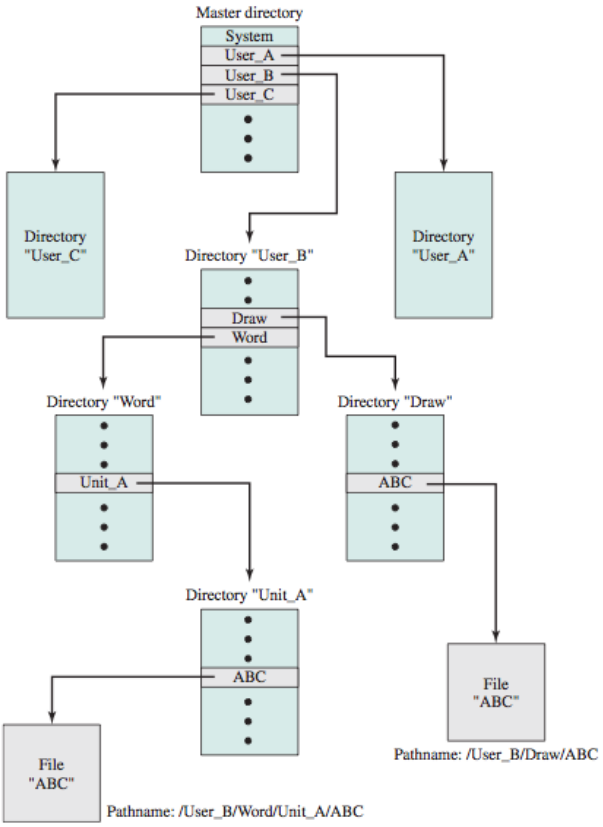


Figure 12.7 **Example of Tree-Structured Directory**

```
 24K    ./p21.log
  0B    ./p21.out
132K    ./p21.pdf
156K    ./p21.synctex.gz
 24K    ./p21.tex
 12K    ./udla.png
376K    .
```

2.3 Protección y Seguridad

- Los archivos deben mantenerse a salvo por seguridad (daños accidentales) y protección (contra accesos maliciosos: virus, hacks, etc.).
- Aparte de realizar copias de backup, un simple esquema de protección elimina todo acceso a fichero. Sin embargo, esto hace al fichero inutilizable, con lo que un acceso **controlado** y gestionado hace que sea una mejor opción.

Tipo de Acceso:

- **Operaciones de bajo nivel:**
  - Read: ver los contenidos del archivo.
  - Write: cambiar los contenidos del archivo.
  - Execute: cargar el fichero en la CPU y seguir las instrucciones que contiene.
  - Append: Añadir al final del archivo.
  - Delete: Eliminar el archivo del sistema.
  - List: Ver el nombre y otros atributos.

- **Operaciones de alto nivel: copy, tar, zip, etc.**

```
#Ejemplo de empaquetado TAR mediante Python

>>> f = open("largefile.txt","w")
>>> statement = "Esto es una gran línea"
>>> x=0
>>> for x in xrange(2000):
...     x += 1
...     f.write("%s\n" % statement)
...
>>>
>>> import tarfile
>>> tar = tarfile.open("largefile.tar","w")
>>> tar.add("largefile.txt")
>>> tar.close()
>>>
```

Utilizado en Windows

Control de Acceso:

- Un enfoque sería tener complicadas ACLs (Access Control Lists), las cuales epecifican exactamente qué acceso es permitido o denegado para cada usuario o grupos específicos. → Realmente es bastante complicada su administración aunque muy precisa.
- Otro enfoque: UNIX/LINUX utiliza un conjunto de 9 bits de control en tres grupos de tres. Estos corresponden a los permisos R, W y X para cada u(ser), g(roup), o(thers). La combinación de RWXcontrola los privilegios de los archivos y directorios:

Bit	Files	Directories
R	Read (ver el contenido del archivo)	Leer el contenido de los directorios. Requerido para listar el contenido de un directorio.
W	Write (cambiar el contenido del archivo)	Cambia el contenido del directorio. Requerido para crear o eliminar archivos.
X	eXecution (ejecuta el fichero como un programa)	Información detallada de la información. Requerida para listar el contenido de forma extendida o acceso a determinados archivos*.

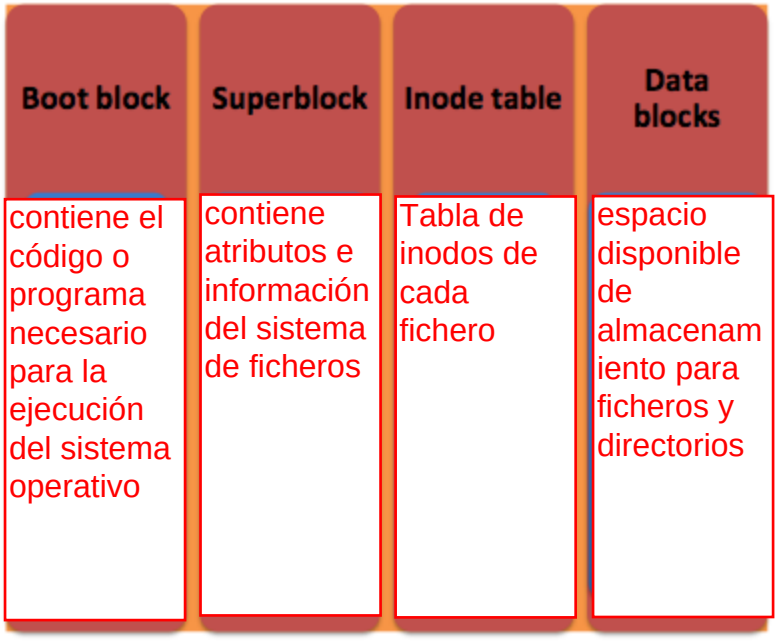
(* Nótese que si el usuario tiene permisos X pero no permisos de R en un directorio, puede acceder a un determinado archivo sí y sólo sí, conocen el nombre del archivo al que quieren acceder).

3 Implementación del Sistema de Archivos

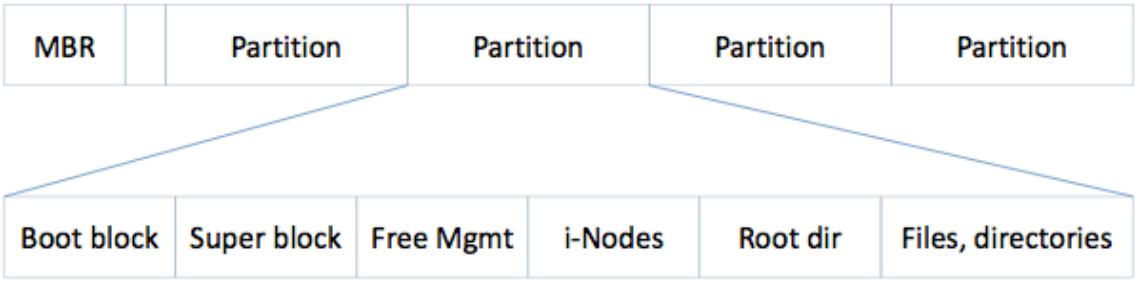
3.1 Sistemas de Archivos

- Gestionan el almacenamiento de datos en disco.
- La unidad de organización es el archivo → objeto de datos que ocupa espacio en disco.
- Organiza el espacio en disco → El disco llegar a ser un objeto contenedor de archivos.
- Gestiona:
  - Localización: registra dónde y como están almacenados los archivos.
  - Estructura: Archivos se organizan en directorios / carpetas.
  - Acceso: Permite la creación de archivos y operaciones R y WR.
  - Rendimiento: reduce operaciones I/O
  - Fiabilidad: puede recuperarse ante fallos.
  - Seguridad: privilegios de acceso

3.2 Unix Volume Management



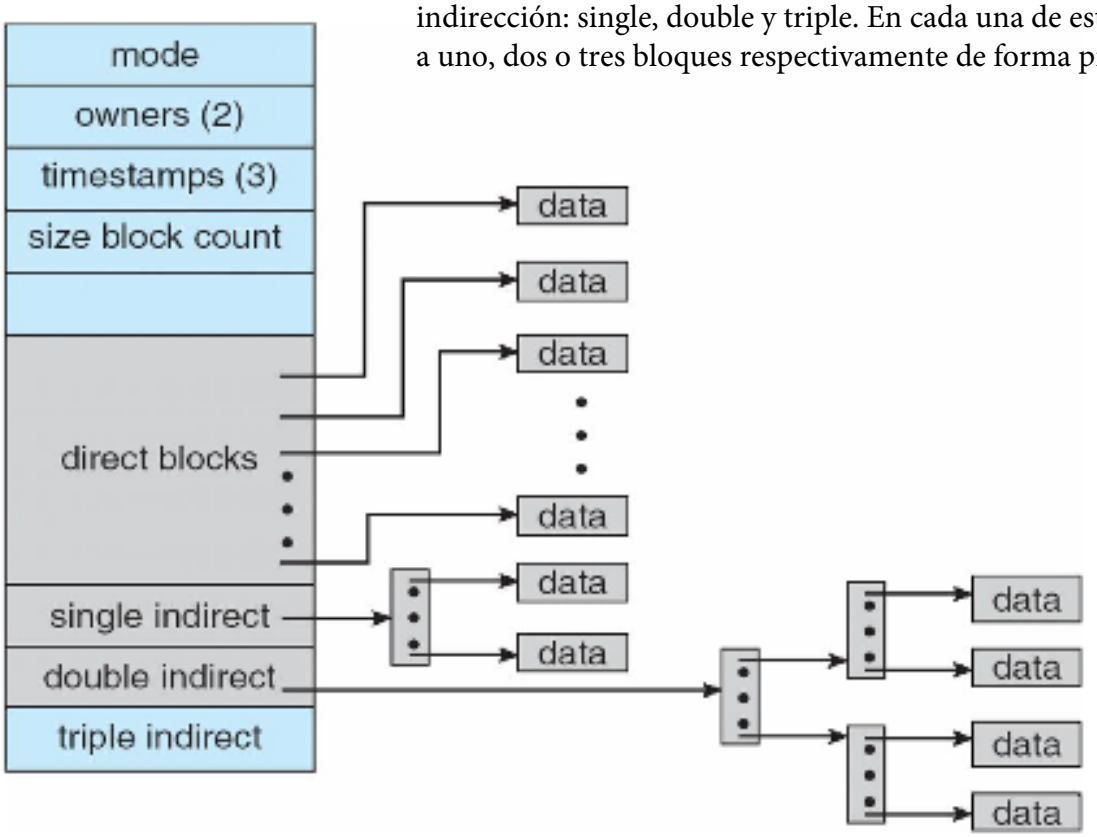
- El sistema de archivos de UNIX reside un único disco lógico o partición.
- Se compone de:
  - Boot block: continue el código requerido para iniciar el SO.
  - Superblock: contiene atributos e información sobre el sistema de archivos.
  - inode table: colección de indos para cada archivo.
  - Data blocs: espacio de almacenamiento disponible para archivos y directorios.
- La capa de sistema de archivos y disco para UNIX funciona de la siguiente manera:
  - Master Boot Record (MBR): sector 0 del disco contiene el código de arranque y la tabla de particiones.
  - Arranque de Sistema: el programa contenido en el MBR carta el Boot Block de la partición activa o provee un menú para cargar otra.



3.3 Virtual File System (Linux)

- El VFS provee un interfaz común para diferentes sistemas de archivos. Además, añade un identificador único para los archivos a través de todo el disco incluyendo en otros Sistemas de Archivos de diferentes tipos → Esto lo hace bastante potente a la hora de poder utilizarse por otros Sistemas.
- El VFS de Linux está basado en cuatro objetos clave:
  - El objeto **inode**: que representa el archivo individual.
  - El objeto **archivo**: que representa un archivo abierto.
  - El objeto **supeblock**: que representa el sistema de archivos.
  - El objeto **dentry**: representa una entrada a directorio.
- El **i-Node:**
  - Todos los tipos de archivos UNIX/Linux están gestionados por el SO mediante los i-Nodes.
  - Es una estructura de control (índex node) que contiene la clave de información necesaria para el OS para un archivo particular (describe sus atributos y punteros a los bloques del disco donde está ubicado el archivo).
  - El i-Node es un índice los bloques de disco del archivo (un i-node por archivo).
  - Puede haber una serie de archivos para un sólo i-node (ligaduras o links, que son como alias).

En los i-node puede referenciar directamente a los bloques o tener indirección: single, double y triple. En cada una de estas accede antes a uno, dos o tres bloques respectivamente de forma previa.



3.4 Sistema de Ficheros en Windows

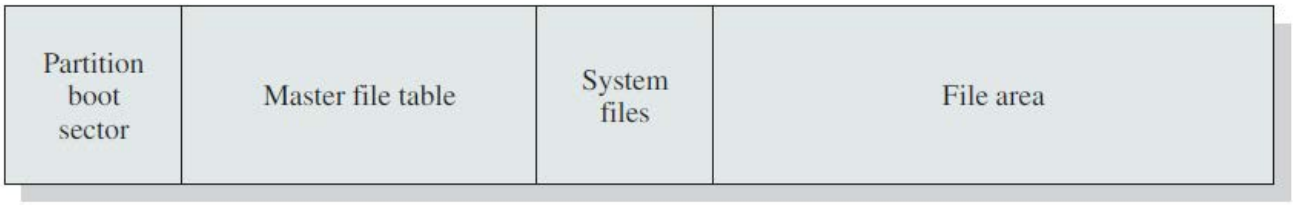


Figure 12.21 NTFS Volume Layout

Sistema de Ficheros en Windows : en la imagen de arriba se observa la organización de un Volumen NTFS, el cual se divide en Partición del sector de arranque, MFT o Tabla de ficheros maestro, ficheros de sistema y area de ficheros.

* Partición de sector de arranque: contiene información sobre el volumen, estructura del sistema de archivos e información de arranque.

* MFT: contiene la información de todos los ficheros y carpetas de este volumen NTFS. En ensencia, el MFT es un listado de todos los ficheros/directorios y sus atributos organizado en estructura de tabla.

Se puede ver más abajo una imagen de la MFT.

* Ficheros de sistema: es donde están algunos ficheros necesarios para el sistema como:
- MFT2, una copia de seguridad de la MFT
- Un fichero de logs para recuperación
- Cluster bitmap.: un mapa de los clusters utilizados.
- Atributos: necesarios para recavar información sobre indexación de ficheros para recuperación.

* File area: donde están los datos: ficheros y directorios.

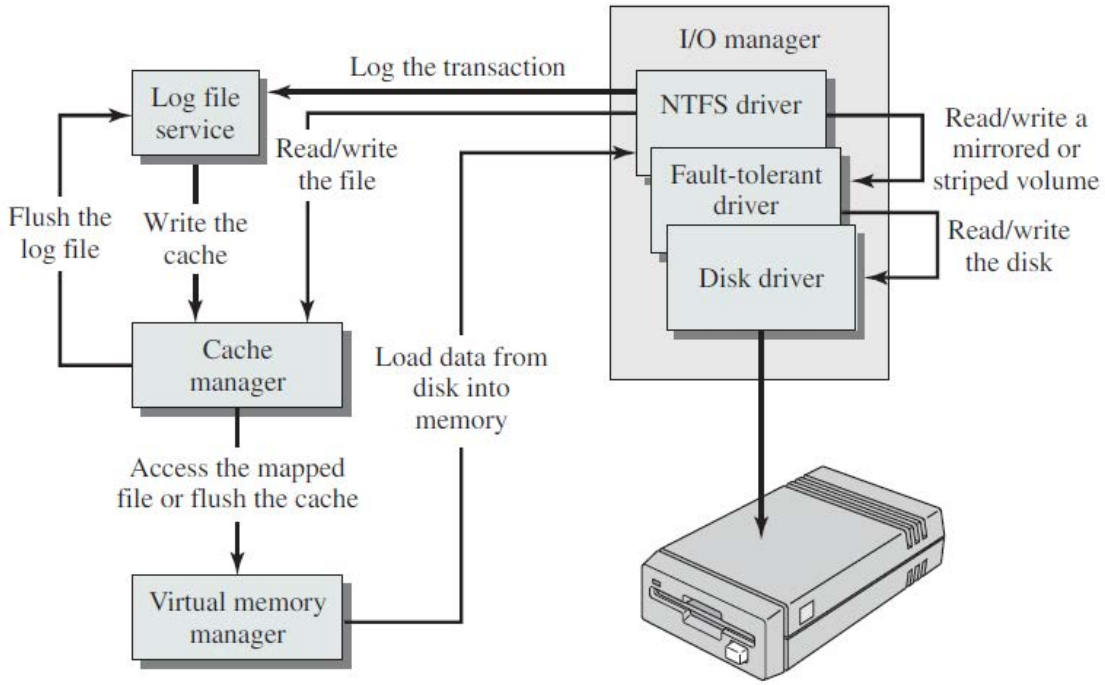
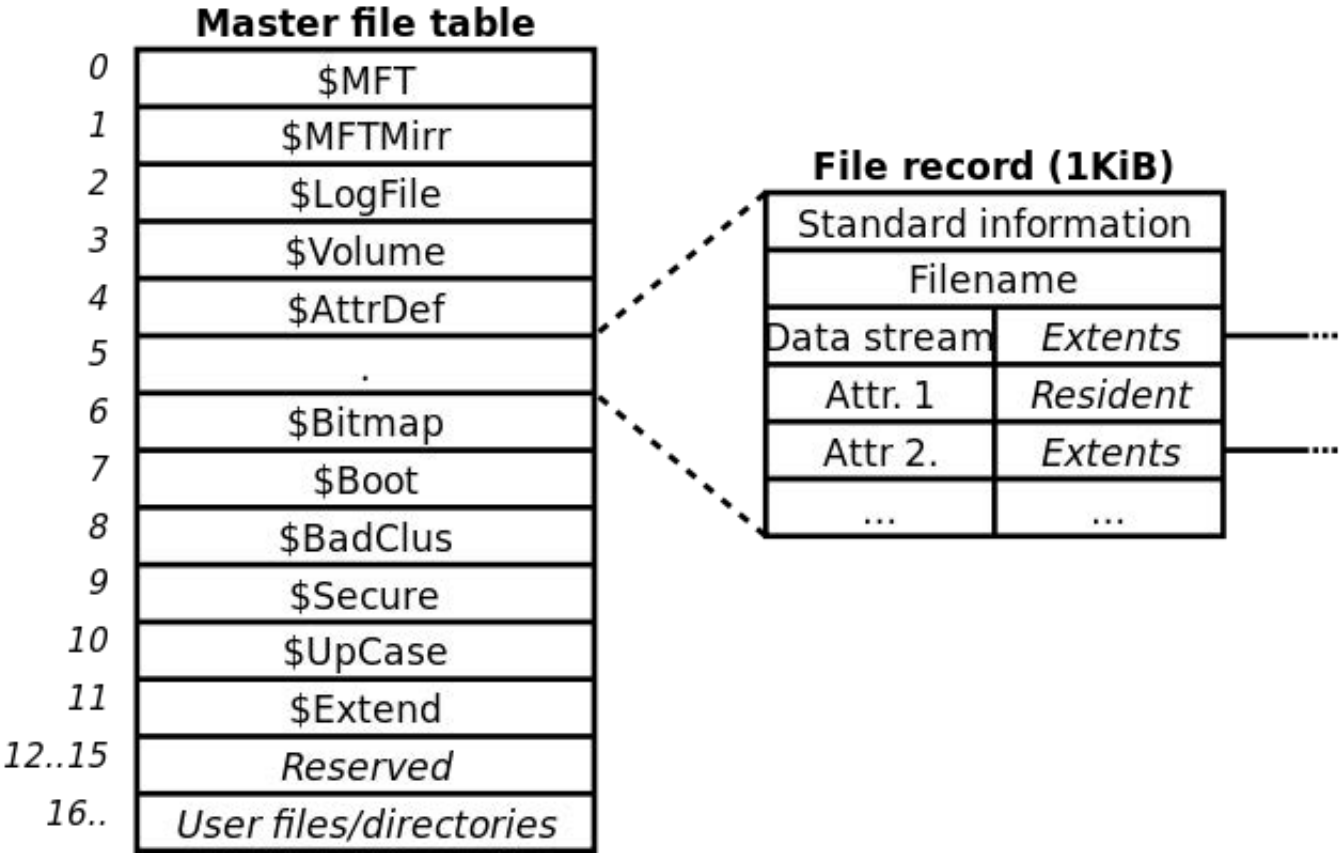


Figure 12.22 Windows NTFS Components

Más arriba se puede apreciar los componentes y flujo de trabajo en un sistema de ficheros NTFS..

- NTFS realizar cualquier log de la transacción antes de realizarse y lo hace en la caché.
- NTFS modifica el volumen en la caché
- La Cache manager llama al fichero de logs y hace un volcado de la información al disco.
- Sólo cuando se han almacenado los datos se borra la caché



Comandos útiles para ver la estructura del MFT en Windows:

fsutil volume filelayout c:\$mft
fsutil fsinfo ntfsinfo c:
ntfsinfo c:

MFT Directory Entry (with extents)