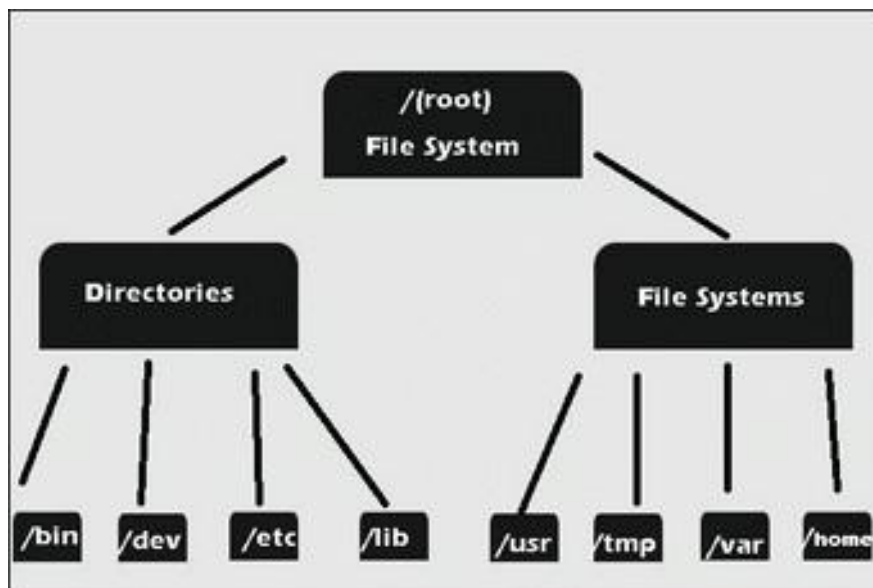




# SISTEMA DE ARCHIVOS



- Introducción de las funciones del Sistema de Archivo
- Tipos de Sistema de archivo
- Los mas populares
- Aspectos para evaluar un F. S.
- Diseño de un F. S.
  - Concepto de Bloque
  - Archivos – Directorios
  - Asignación de Espacio
  - Control de Espacio Libre
- Casos de Estudio
  - FAT
  - Unix File-Systems
    - Inodos
    - Manejo de inodos
  - NTFS
  - MFT
- Abstracción del Sistema de archivo



## *Introducción al Sistema de Archivo*

Es el que indica como se **organiza la información** dentro de los **medios físicos**

**Entre sus funciones están:**

- El guardado y recuperación de la información;
- Administración del espacio libre,
- El guardado de los atributos de seguridad del sistema operativo (En los filesystem nuevos).

Estos programas se encargan de:

- Identificar y localizar archivos
- Seguridad
- Asignación de espacio
- Coordinación de transferencia (sistema multiusuario)



# Introducción al Sistema de Archivo

**Sistema de archivos:** Subsistema del S.O. encargado de la gestión de la memoria secundaria (concretamente del almacenamiento de la información en dispositivos de memoria secundaria).

El medio sobre el que se almacenan los archivos se divide en bloques de longitud fija, siendo el sistema de archivos el encargado de asignar un número adecuado de bloques a cada archivo.

Funciones del sistemas de archivos:

- Crear y borrar archivos
- Permitir el acceso a los archivos para que sean leídos o escritos
- Automatizar la gestión de la memoria secundaria
- Permitir referenciar un archivo por su nombre simbólico
- Proteger los archivos frente a fallos del sistema
- Permitir el uso compartido de archivos a usuarios autorizados

**BLOQUES:**

**SECTOR 512 Bytes.**

**El Bloque puede tomar varios sectores:**

**ej. 1k, 2 k o 4 k**



## Tipos de Sistema de archivo

Pueden ser clasificados por:

- medio
- red
- propósito especial.

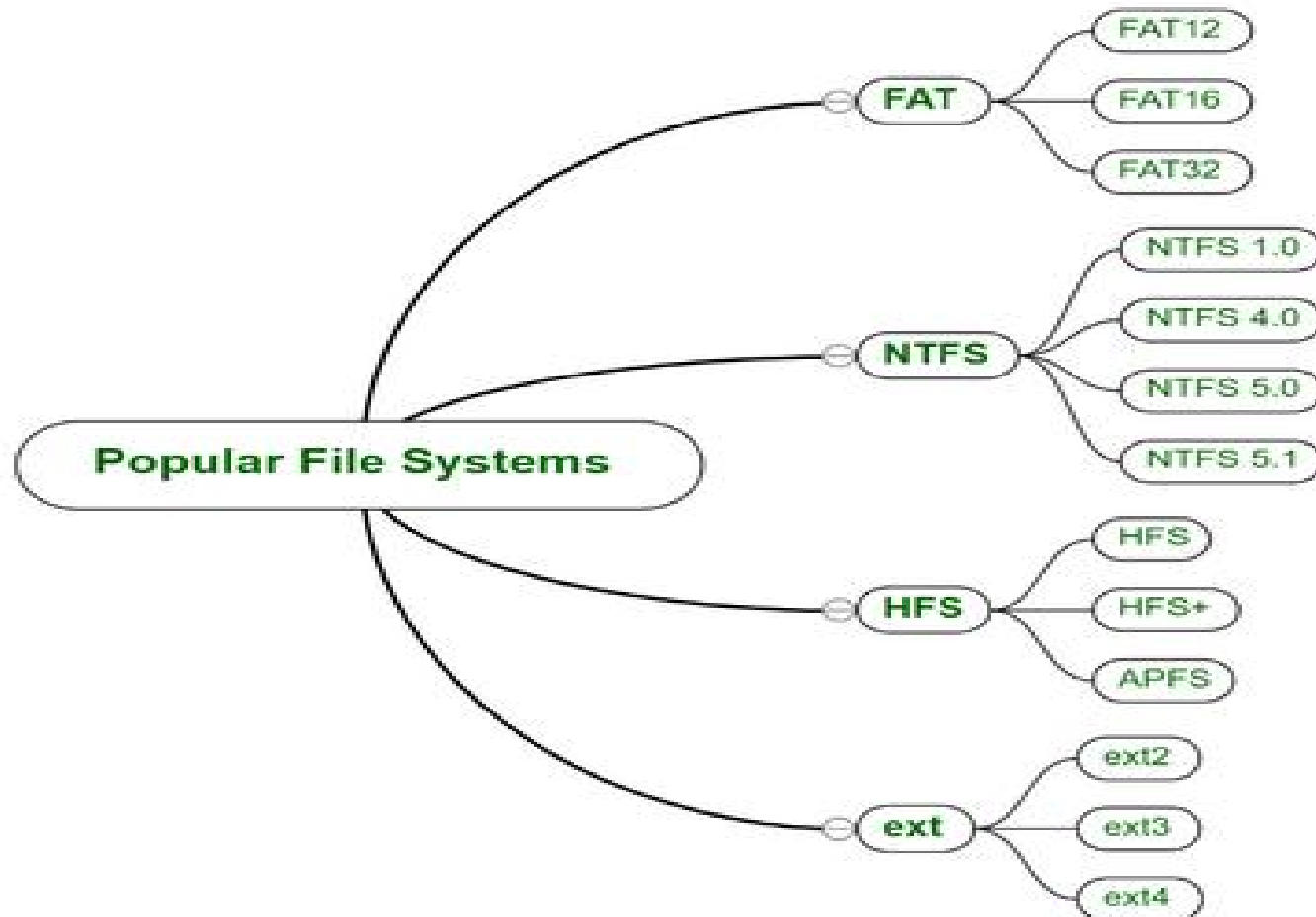
- **Discos de plato magnetico o Flash** (acceso aleatorio): FAT, NTFS, HFS+, UFS, EXT2/3/4, XFS, BTRFS, Veritas File system, ZFS, ReiserFS
- **Discos Opticos:** ISO9660, UDF
- **Flash file System** (Especialmente embebidos): JFFS, UBIFS, LogFS, F2FS, SquashFS

- **Network File System:** NFS, AFS (Sistema de comparticion de apple), SMB (Sistema de comparticion de microsoft), FTP, Webdav, SSHFS.
- **Sistemas de archivos en Cluster:**
  - **Shared-disk / storage area network:** GFS2 (Red hat), GPFS (IBM), CSV (Microsoft), OCFS (Oracle)
  - **Distributed file systems:** GFS (Google), HDFS (Apache), Ceph (Red Hat), DFS (Microsoft), GlusterFS (Red Hat)

- **Especiales:**
  - **procfs** --> Mapeo de procesos y estructuras en linux
  - **configfs y sysfs** --> Muestra como archivos los parametros de consulta y configuración de un Linux
  - **devfs** --> los archivos en esta fs representa a dispositivos. Y permite utilizar las primitivas de cualquier fs para acceder al dispositivo.
  - **Superpuestos** --> overlayFS, UnionFS, aufs



## Los mas populares

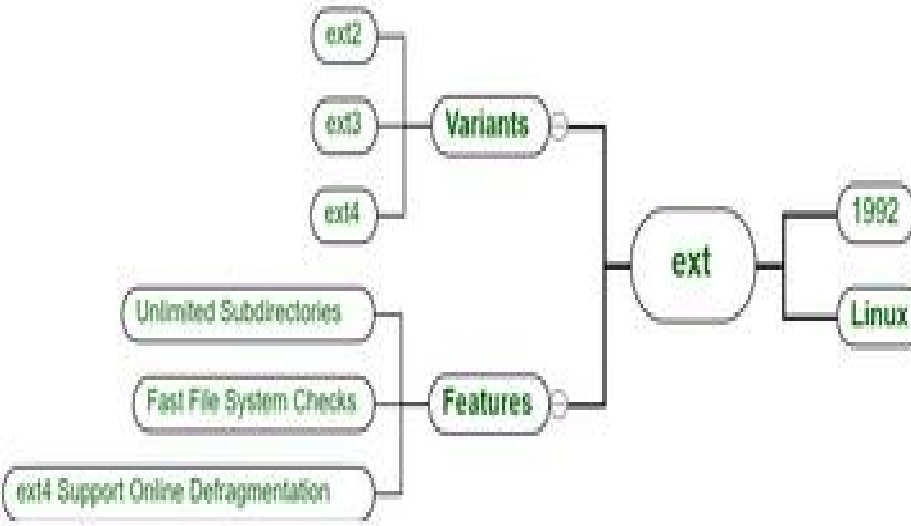
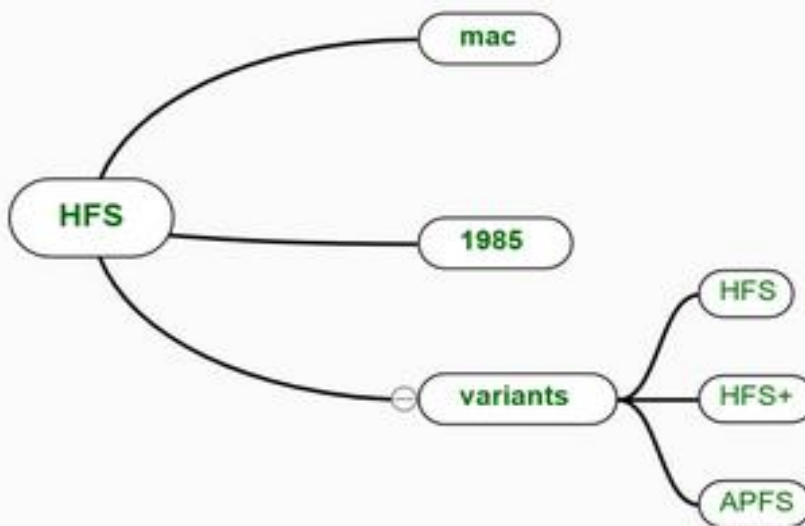
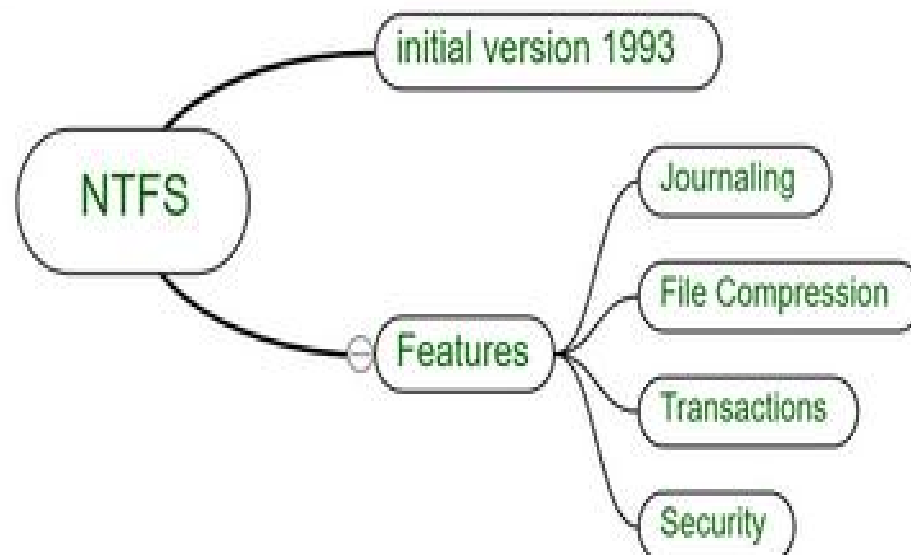
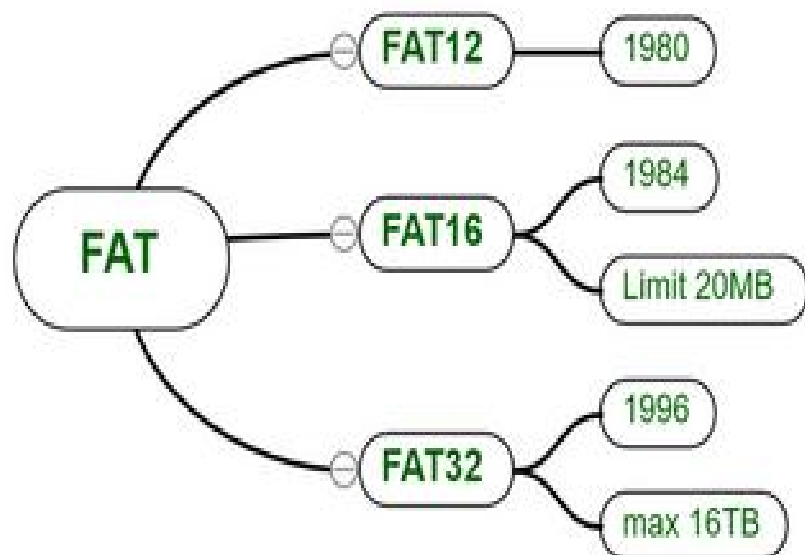


- FAT (File Allocation Table).
- NTFS (New Technology File System).
- HFS (Hierarchical File System).
- EXT (Extended File System).





## Los mas populares





# Aspectos a evaluar para un F.S.

- **Manejo de espacio**
  - Manejo de nombres de los objetos (directorio y archivos)
  - La estructura de directorios utilizados
  - Metadata que se le puede asociar a los objetos (directorio y archivos)
- **Utilidades:**
  - Creación, Chequeo, Defrag, Tunning, Resize, Undelete, etc.
- **Mecanismos de permisos y restricción.**
  - ACL
  - Capacidades
- **Integridad de la información**
- **Limitiaciones impuestas por diseño.**
- **Extras**
  - Encriptado a nivel archivo o directorio
  - Compresión a nivel archivo o directorio
  - Desduplicacion



# Diseño de un FS

## EL CONCEPTO DE BLOQUE

Un bloque es la unidad mínima de asignación de espacio en un soporte específico.

Siempre que se pida un cierto espacio en un soporte de almacenamiento, el sistema de archivo proporciona una cantidad que, por lo regular, es mayor que la que se pide. Problema **Fragmentación Interna**.

### Ejemplo de Archivo lógico Vs Físico:

```
dd if=/dev/urandom of=./uno.txt bs=512 count=1
dd if=/dev/urandom of=./dos.txt bs=512 count=8
dd if=/dev/urandom of=./tres.txt bs=512 count=9
ls -l
du -h *
```

### Ejemplo de direccionamiento:

Lógico Vs Físico

```
blockdev --getss /dev/sdb
blockdev --getpbsz /dev/sdb
```

### Manipulación de Archivos:

Entrada/salida lógica	Relación con Usuarios y Aplicaciones
Supervisor de E/S básico	Comienza y termina E/Salidas
Sistema de archivo básico (E/S física)	Bloques de datos, desconoce estructura, de datos.
Controladores	(device driver) directo sobre el Hardw.





# Diseño de un FS

## ATRIBUTOS DE LOS ARCHIVOS:

**Nombre del archivo:** Univoco dentro de un directorio.

**Tipo de archivo:** Formato (texto, binario, Enlace simbólico, etc)

**Datos de la creación o modificación:** fecha, hora, etc.

**Tamaño del archivo:** Especificado en bytes, bloques, etc.

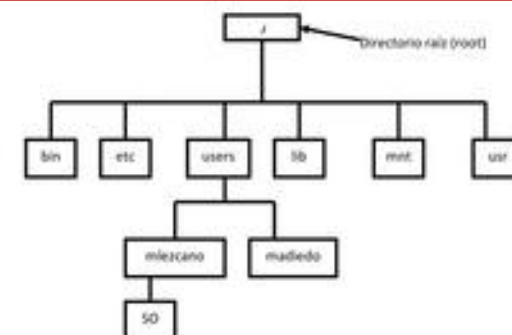
**Atributos de protección.** Permisos.

## Ejemplo comandos:

dir en windows  
ls en linux

## ESTRUCTURA DE DIRECTORIO:

- Todo sistema de archivo posee, como parte de su organización, una estructura de datos denominada tabla de directorio.
- Todos los medios de almacenamiento poseen esa estructura de datos, que se construye cuando se le da formato al soporte.
- La forma que tenga el directorio depende del SO.



## Una Estructura Posible (Solo didáctica)

NOMBRE	TIPO	DIRECCIÓN DE INICIO	FECHA	HORA	TAMAÑO	PROPIETARIO



# Diseño de un FS

## EL CONCEPTO DE BLOQUE

Un bloque es la unidad mínima de asignación de espacio en un soporte específico.

Siempre que se pida un cierto espacio en un soporte de almacenamiento, el sistema de archivo proporciona una cantidad que, por lo regular, es mayor que la que se pide. Problema **Fragmentación Interna**.

## ASIGNACIÓN DE ESPACIO:

Para pensar...

Cuando se cree un nuevo archivo, ¿se le asignará todo el espacio que necesita?

¿Qué tamaño debe tener un bloque?

¿Que pasa si los Bloques son muy pequeños?

¿Que pasa si los Bloques son muy Grandes?

La idea es tener un tamaño optimo de Bloque para no provocar tanta fragmentación interna y para no tener que usar demasiados recursos para controlarlos.

**No es tan fácil definir el tamaño “ideal” de un bloque.**





# Diseño de un FS

## TIPOS DE ASIGNACIÓN DE ESPACIO:

- **Contigua:** Asigna un conjunto de bloques contiguos cuando se crea al archivo.
- **Enlazada:** Cada bloque tiene un puntero al próximo bloque y pueden estar en cualquier parte de la unidad de almacenamiento.
- **Indexada:** Cada archivo usa uno de los bloques para almacenar las localizaciones de los bloques que pertenecen al archivo.

## CADA UNA TIENE:

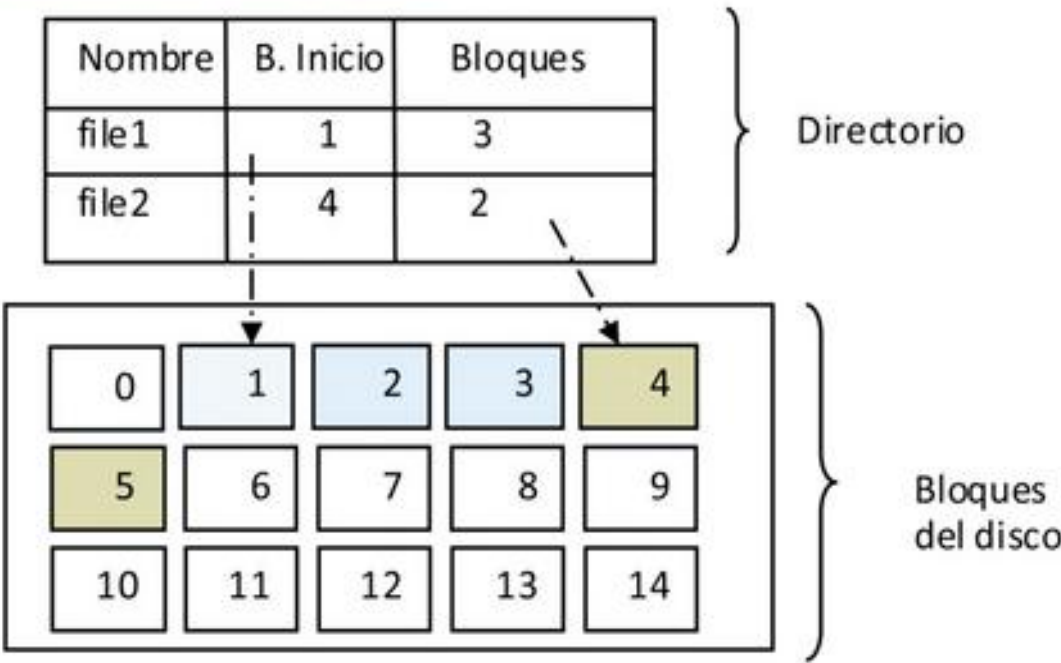
- **Ventajas y desventajas.**
- **Estrategias:** Cada Estructura tiene asociada sus estrategias de manipulación.
- **Problema:** Como pensar un F. S. multi-proposito o bien un F. S. especializado.





# Diseño de un FS

## ASIGNACIÓN CONTINUA:



**Desventajas:**  
**El crecimiento.**  
**Fragmentación Externa.**

**DES FRAGMENTAR:**  
Cual es el problema de la des fragmentación?

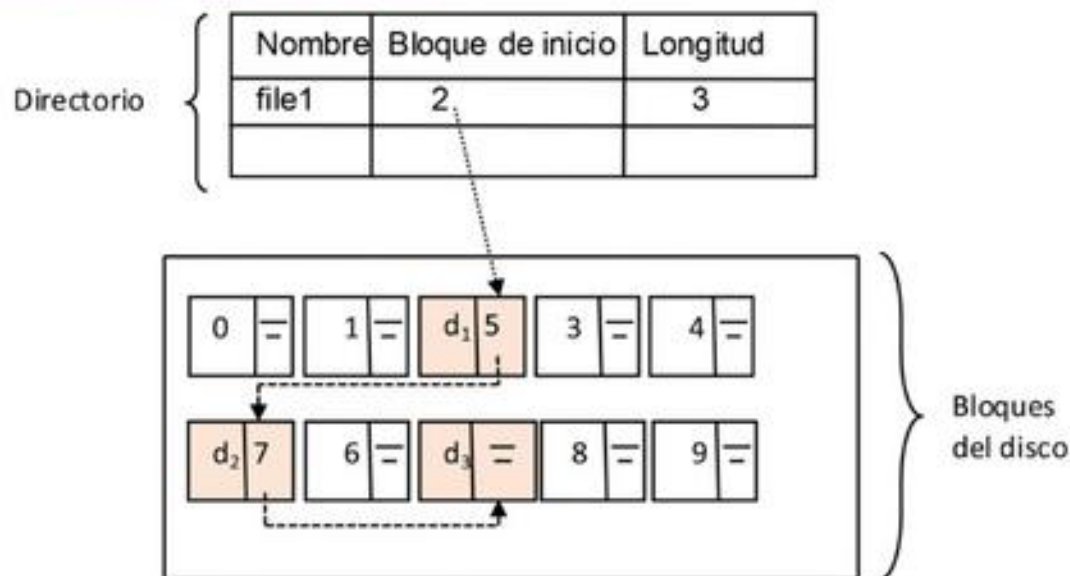
**Ventajas**  
**Acceso directo a los datos.**  
**Rápido:** Se efectúan menos movimientos mecánicos del cabezal de lectura/escritura.  
**Fácil Recuperación de datos Borrados.**





# Diseño de un FS

## ASIGNACIÓN ENLAZADA:



### Problema:

¿ Que tipo de acceso utiliza?  
¿Mejora o empeora la situación?  
Comparando con la anterior....

### Ventajas

Crecimiento mientras existan bloques libres.  
Elimina el problema de la fragmentación externa.

### Desventajas:

**No Tiene acceso directo;** para acceder al bloque  $n$  hay que recorrer los  $n-1$  bloques que le preceden.  
**Lentitud:** Por dispersión de los bloques.  
**Gasta espacio adicional:** Cada bloque contiene un campo dedicado al puntero.





# Diseño de un FS

## ASIGNACIÓN INDEXADA:

Nombre	B. Inicio
file1	4

Directorio

Bloque 4 ampliado

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

5
9
13
Nulo
Nulo
Nulo



### Ventajas

No provoca fragmentación externa.

Permite el acceso aleatorio.

Los archivos pueden crecer mientras haya espacio y forma de hacer referencia a sus bloques.

### Desventajas

Archivos están dispersos.

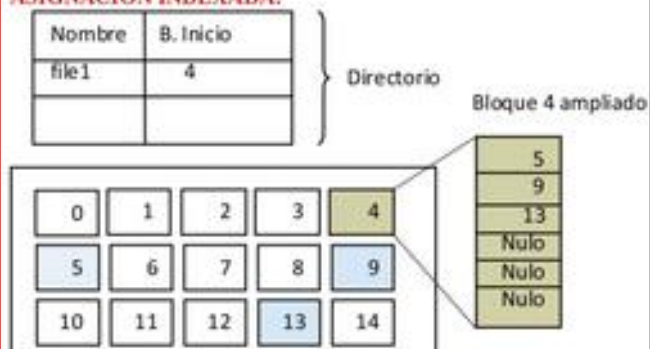
Acceso mas lento que la Continua.

**LA INDEXADA ES LA MÁS UTILIZADA ACTUALMENTE**



# Diseño de un FS

## ASIGNACIÓN INDEXADA:



## Algunas Reflexiones:

- Los Archivos del Directorio apuntan a un Nodo de Direcciones.
- El Primer Nodo de Direcciones contiene todas las direcciones de bloque del Archivo.
- Que hacer Si el Archivo requiere mas bloques que la cantidad que tiene el Nodo de direcciones?



## Algunas Aplicaciones de este modelo:

- La solución que ofrece el SO Unix se basa en esta idea.
- La Solución de Windows, en sus FAT, se basa en Generar una gran tabla de índices para todos los archivos del F. S.
  - FAT tiene una lista enlazada que es necesario recorrer para localizar los bloques de datos.
  - **FAT NO ES UNA LISTA ENLAZADA**

La solución del tipo de asignación depende del problema



# Diseño de un FS

## CONTROL DE ESPACIO LIBRE:

Para controlar el espacio libre, se pueden usar varios tipos de estructuras de datos.

- Utilizar la Misma estructura. (Caso de la FAT). Poco eficiente.
- Control a partir de una Tabla.
- Utilizando un mapa de Bits.

## CONTROL A PARTIR DE UNA TABLA:

DIRECCIÓN DE INICIO	TAMAÑO
10	23
80	12

Tener cuidado con la eliminación de Archivos y revisar antes de actualizar la tabla de Espacio libre.

## CONTROL UTILIZANDO MAPA DE BITS:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	0	1	0	0	0	1	1	0	0	0	0	0	1	1	1	1	0

La longitud del mapa es igual a la cantidad de Bloques del FS.  
Las estrategias cambian de acuerdo a si es Contigua, Enlazada, o Indexada.

## DES FRAGMENTAR:

Cual es el problema de la des fragmentación?







# Caso de Estudio F. S.

## File Allocation Table (FAT)

### Tabla de Directorios:

Nombre	Extensión	Atributos	Reservado	Hora	Fecha	Cluster	Longitud
carta	doc					12	
pepe	txt					23	

\* atributos (oculto, de lectura solamente, sistema y archivo)

### Tabla de Datos

	0	1	2	3	4	5	6	7	8	9
0	0	0	eof	8	0	0	0	0	2	0
1	0	0	3	0	0	0	0	0	0	0
2	0	0	0	30	0	0	0	0	0	0
3	31	38	0	0	0	0	0	0	50	0
4	0	0	0	0	0	0	0	0	0	0
5	eof	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0



### Doble Propósito:

- Localizar los bloques de datos de cada Archivo.
- Controlar el espacio libre.

Los 2 primeros Clúster no se utilizan (el 1ro es MBR)

El S.O. realiza los cálculos para ubicar físicamente el Bloque en el disco.



# Caso de Estudio F. S.

El sistema de archivos FAT se compone de cuatro secciones:

- **El sector de arranque**
  - Siempre es el primer sector de la partición (volumen) e incluye información básica, punteros a las demás secciones, y la dirección de la rutina de arranque del sistema operativo.
  - Contiene además la geometría del disco. (para el cálculo del SO).
- **La región FAT**
  - Contiene dos copias de la tabla de asignación de archivos (por motivos de seguridad).
    - Estos son mapas de la partición, indicando qué clusters están ocupados por los archivos.
- **La región del directorio raíz.**
  - Es el índice principal de carpetas y archivos.
- **La región de datos.**
  - Es el lugar donde se almacena el contenido de archivos y carpetas.
    - Por tanto, ocupa casi toda la partición.
    - El tamaño de cualquier archivo o carpeta puede ser ampliado siempre que queden suficientes clúster libres.
    - Cada clúster está enlazado con el siguiente mediante un puntero.
    - Si un determinado clúster no se ocupa por completo, su espacio remanente se desperdicia.



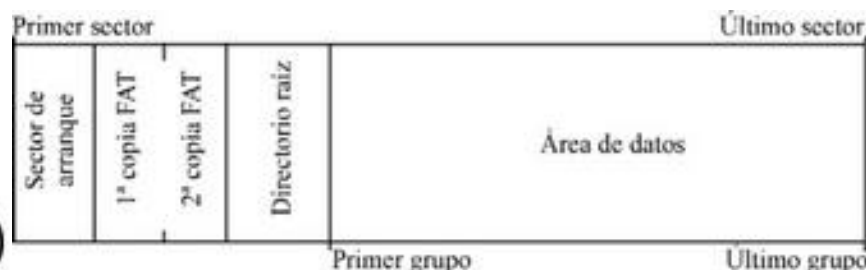




# Caso de Estudio F. S.

## FAT - Estructuras

- Boot (sector de arranque)
- Región FAT (dos copias)
- Directorio raíz (Índice principal de carpetas y directorios)
- Datos



00004000	f0 ff ff 0f	ff ff ff 0f	f8 ff ff 0f	ff ff ff 0f	...	...	...	...
00004010	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...	...	...	...
00004020	09 00 00 00	0a 00 00 00	0b 00 00 00	0c 00 00 00	...	...	...	...
00004030	0d 00 00 00	0e 00 00 00	0f 00 00 00	10 00 00 00	...	...	...	...
00004040	11 00 00 00	12 00 00 00	13 00 00 00	ff ff ff 0f	...	...	...	...
00004050	15 00 00 00	16 00 00 00	17 00 00 00	18 00 00 00	...	...	...	...
00004060	19 00 00 00	1a 00 00 00	1b 00 00 00	1c 00 00 00	...	...	...	...
00004070	1d 00 00 00	1e 00 00 00	1f 00 00 00	20 00 00 00	...	...	...	...
00004080	21 00 00 00	ff 00 ff ff	0f 00 00 00	00 00 00 00	!	...	...	...
00004090	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...	...	...	...

FAT ID	Indicación de fin de cadena
Clusters vacíos	El directorio raíz cabe en un sólo cluster
Primer cadena (un solo cluster: 0003)	Segunda cadena (12 clusters: 0008 a 0013)
Tercera cadena (16 clusters: 0014 a 0021)	



# Caso de Estudio F. S.

## La FAT

- Cada entrada, campo o celda, contiene información sobre un clúster:
  - La dirección del siguiente clúster en la cadena.
  - Si es pertinente, la indicación de "fin de archivo" (EOF).
  - Un carácter especial para indicar que el clúster es defectuoso.
  - Un carácter especial para indicar que el clúster está reservado (es decir, ocupado por un archivo).
  - El número cero para indicar que el clúster está libre.
- El tamaño de estas entradas también depende de la variante FAT
  - FAT12 de 12 bytes.
  - FAT16 de 16 bytes.
  - FAT32 de 32 bytes.
- Una entrada representa un bloque o cluster del volumen.



**Se Mantienen 2 FAT ya que por seguridad Resguarda una copia.**



# Caso de Estudio F. S.

## FAT Directorio Raíz

Este índice es un tipo especial de archivo que almacena las sub carpetas y archivos que componen cada carpeta.

Cada entrada del directorio contiene:

- Nombre del archivo o carpeta (máximo 8 caracteres)
- Extensión (máximo 3 caracteres)
- Atributos (archivo, carpeta, oculto, del sistema, o volumen)
- Fecha
- Hora
- Dirección del primer cluster donde están los datos.
- Tamaño que ocupa.

El directorio raíz ocupa una posición concreta en el sistema de archivos, pero los índices de otras carpetas ocupan la zona de datos como cualquier otro archivo. Los nombres largos se almacenan ocupando varias entradas en el índice para el mismo archivo o carpeta.

En realidad los directorios son un tipo especial de fichero, que en lugar de datos de usuario, contienen metadatos (punteros a otros ficheros).







# Caso de Estudio F. S.



	FAT12	FAT16	FAT32
<b>Desarrollador</b>	Microsoft		
<b>Nombre completo</b>	Tabla de Asignación de Archivos		
	(versión de 12 bits)	(versión de 16 bits)	(versión de 32 bits)
<b>Introducido</b>	FAT12: 1980-08 (SCP 86-DOS 0.42)	FAT16: 1984-08 (PC DOS 3.0), FAT16B: 1987-11 (Compaq MS-DOS 3.31)	FAT32: 1996-08 (Windows 95 OSR2)
<b>Identificador de partición</b>	0x01 (MBR)	0x04, 0x06, 0x0E (MBR)	0x0B, 0x0C (MBR) EBD0A0A2-B9E5-4433-87C0-68B6B72699C7 (GPT)
<b>Estructuras</b>	<b>FAT12</b>	<b>FAT16</b>	<b>FAT32</b>
<b>Contenido de carpeta</b>	Tabla		
<b>Ubicación de archivo</b>	Lista enlazada		
<b>Bloques defectuosos</b>	Lista enlazada		
<b>Limites</b>	<b>FAT12</b>	<b>FAT16</b>	<b>FAT32</b>
<b>Tamaño máximo de archivo</b>	32 MiB	2 GiB (4 GiB - 1)	4 GiB - 1
<b>Número máximo de archivos</b>	4.068 for 8 KiB clusters	65.460 for 32 KiB clusters	268.173.300 for 32 KiB clusters
<b>Longitud máxima del nombre de archivo</b>	8.3 (11) o 255 caracteres cuando se usan LFNs (Long File Names)		
<b>Tamaño máximo del volumen</b>	32 MiB	2 GiB	2 TiB



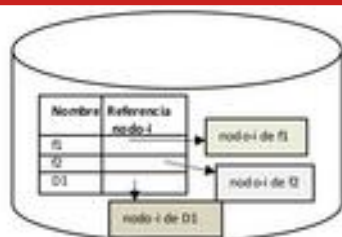
# Caso de Estudio F. S.

## Sistemas de archivos UNIX

### Generalidades de los sistemas de archivos Unix

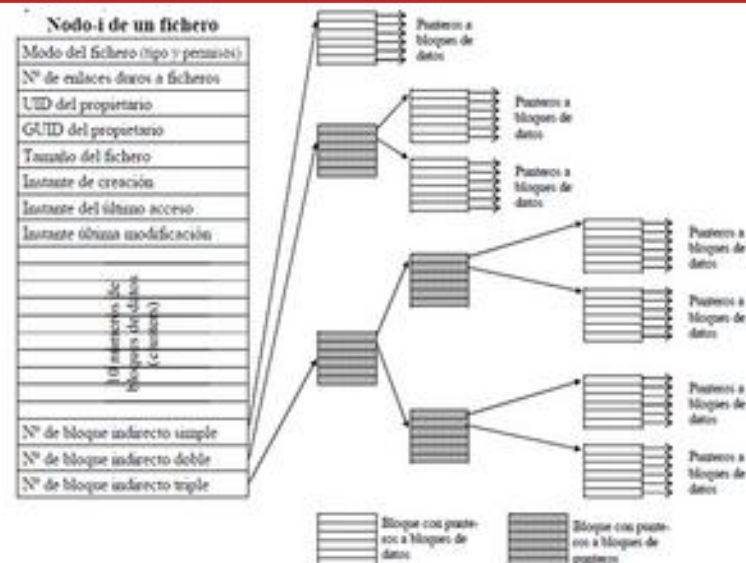
Un archivo Unix es simplemente una secuencia de bytes. Las dos estructuras de datos para localizar un archivo son:

- La tabla de directorio.
- El nodo-i (i-node).



Los conceptos centrales son:

- 1.- superbloque,
- 2.- nodo-i,
- 3.- bloque de datos,
- 4.- bloque de directorio, y
- 5.- bloque de indirección.

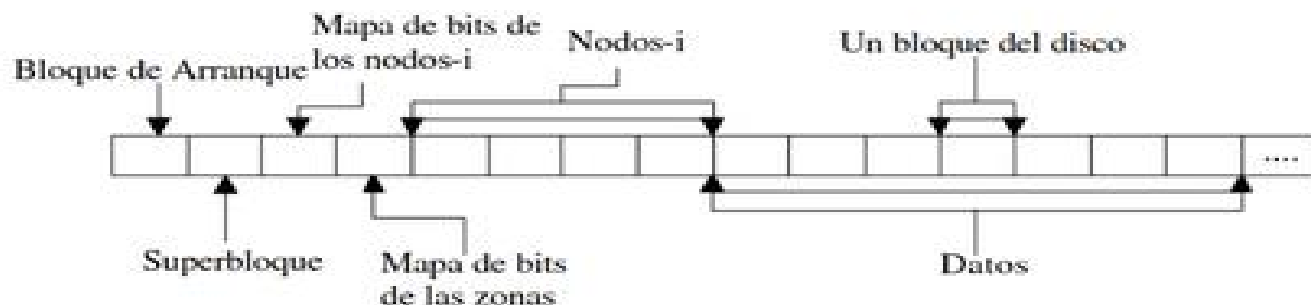






# Caso de Estudio F. S.

## Esquema de un Sistema de Archivos Basado en Minix



### **Bloque de arranque**

Cada sistema de archivo comienza con un bloque de arranque, el cual ocupa siempre un sólo bloque.

### **Superbloque**

Ocupa un solo bloque. Contiene información relativa al tamaño de las distintas partes del sistema de archivos.

### **Mapa de bits de los nodos-i**

Se utiliza para llevar el control de los nodos-i libres y ocupados. Por cada nodo-i existe un bit en dicha tabla que indica su estado. 0,1.

### **Mapa de bits de zonas**

Se utiliza para llevar el control de las zonas de datos. Su funcionamiento es igual del mapa de bits de los nodos-i.

### **Nodos-i**

Son estructuras en las que se encuentra información de cada fichero físico existente en el sistema de archivos, principalmente la relativa a la situación de las zonas de datos de dicho fichero.

### **Datos**

Es el conjunto de bloques que se utiliza para el almacenamiento de los ficheros.

**Basado en Minix**

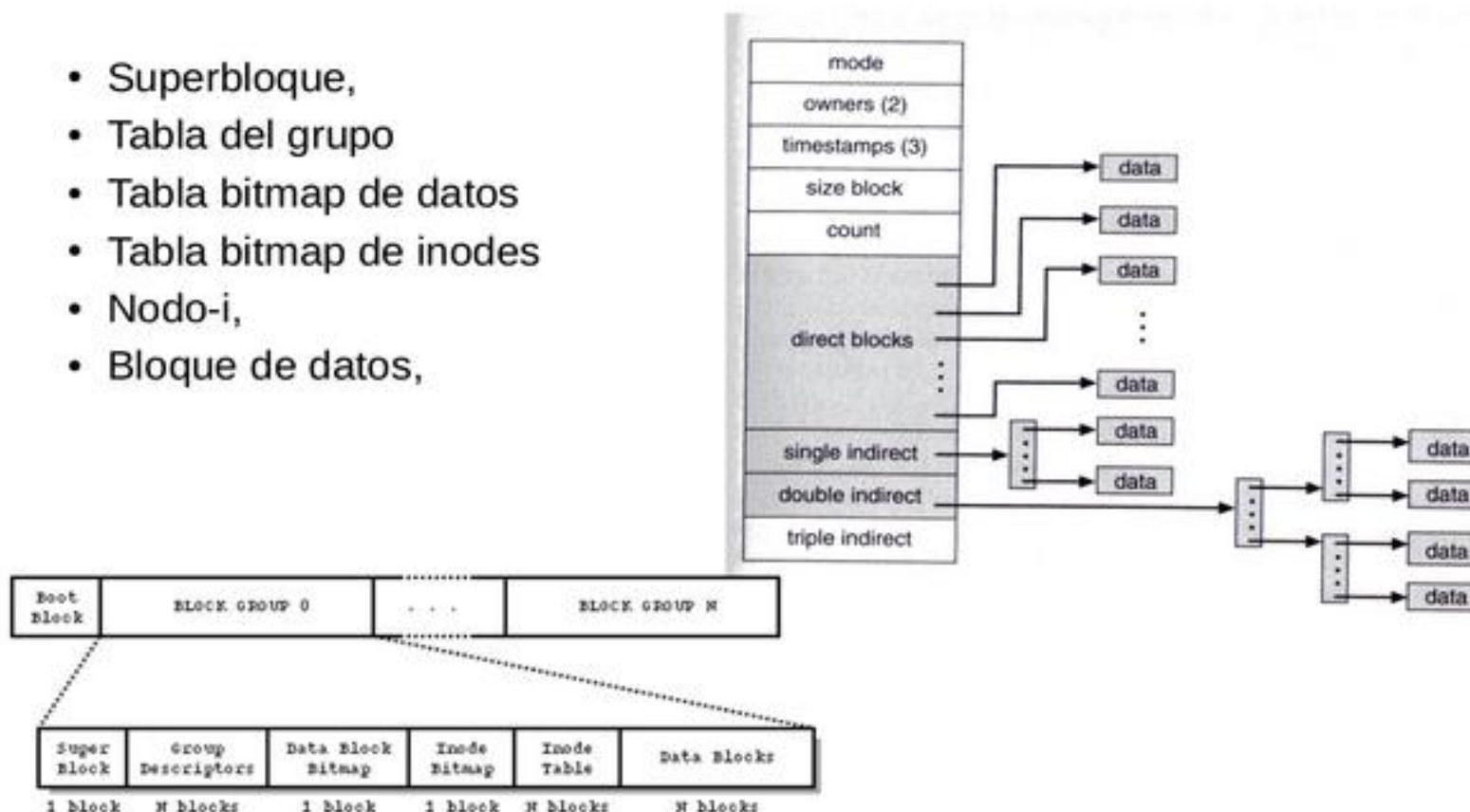


# Caso de Estudio F. S.



## Ext2/3/4

- Superbloque,
- Tabla del grupo
- Tabla bitmap de datos
- Tabla bitmap de inodes
- Nodo-i,
- Bloque de datos,



El superbloque tiene información del sistema de archivos en conjunto, como su tamaño (la información precisa aquí depende del sistema de archivos).



# Caso de Estudio F. S.

## Características Ext4

- Permite archivos de 1(EiB) y hasta 16(TiB)
- Extends
- Compatibilidad con Ext3/2
- Pre alojamiento de espacio
- Alojamiento demorado
- Sin limitaciones de directorios
- Mejoramiento en el timestamp
- Encriptación transparente
- Journal Checksum

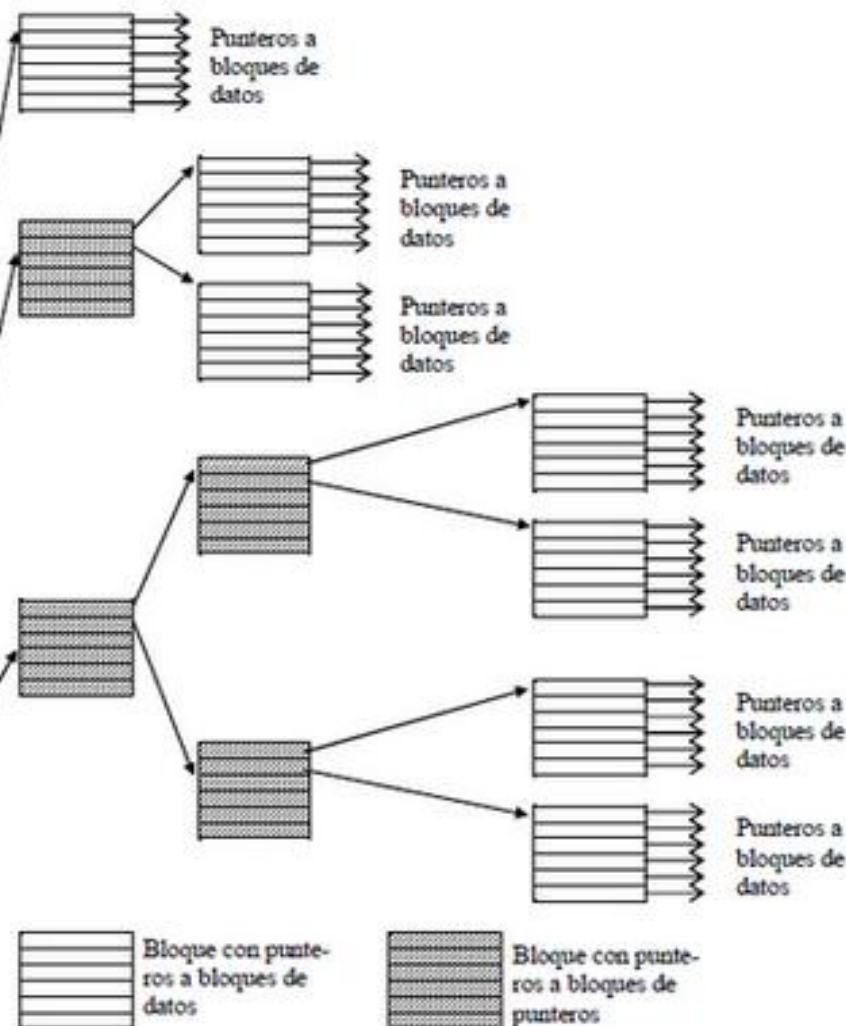




# Caso de Estudio F. S.

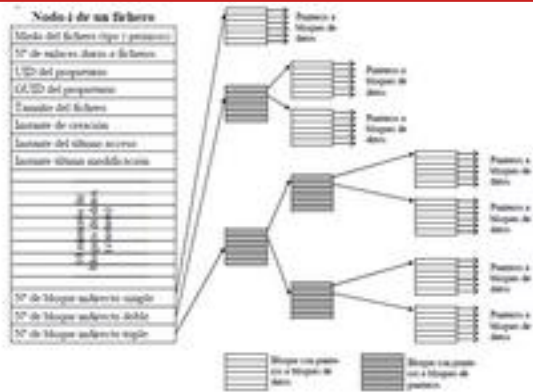


Nodo-i de un fichero	
Modo del fichero (tipo y permisos)	
Nº de enlaces duros a ficheros	
UID del propietario	
GUID del propietario	
Tamaño del fichero	
Instante de creación	
Instante del último acceso	
Instante última modificación	
10 números de bloques de datos (clusters)	
Nº de bloque indirecto simple	
Nº de bloque indirecto doble	
Nº de bloque indirecto triple	



Donde está el Nombre del Archivo?





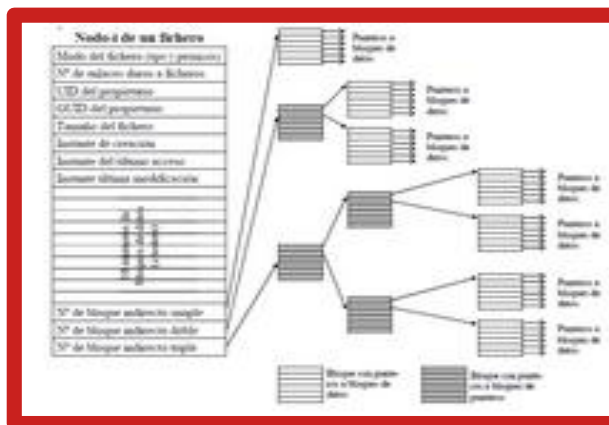
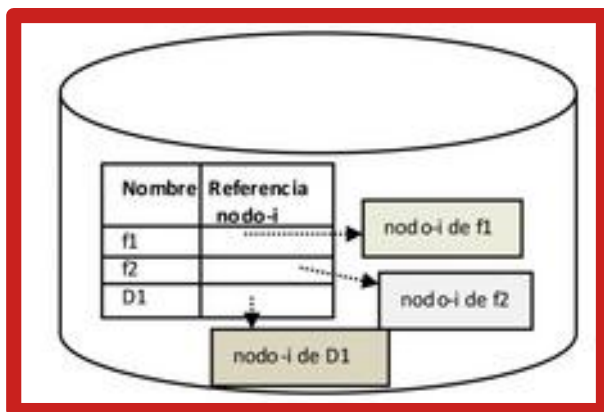
Un nodo-i tiene toda la información de un archivo, **salvo su nombre**. El nombre se almacena en el directorio, junto con el número de nodo-i. Una entrada de directorio consiste en un nombre de archivo y el número de nodo-i que representa al archivo.

**Estos bloques colocados dinámicamente son bloques indirectos; el nombre indica que para encontrar el bloque de datos, primero hay que encontrar su número en un bloque indirecto.**





# Caso de Estudio F. S.

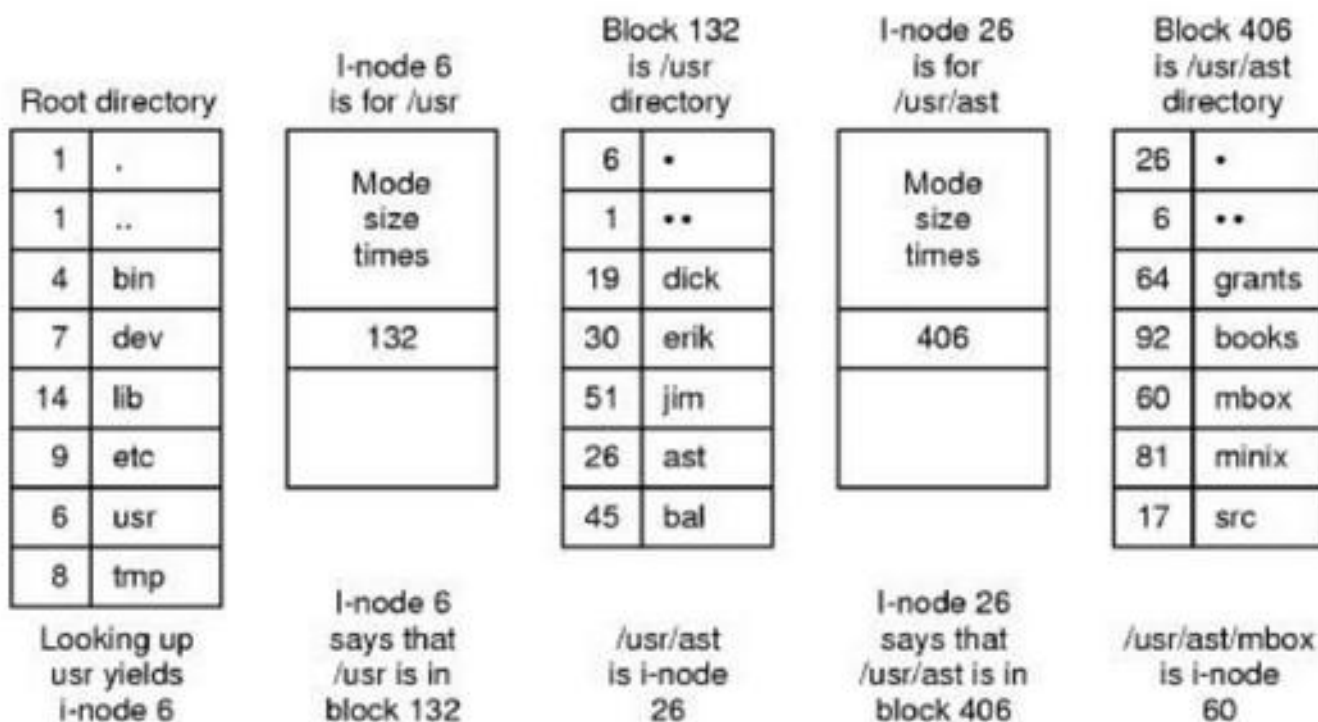


- **Directorios:** Los directorios se estructuran en un árbol jerárquico. Cada directorio puede contener archivos o directorios. Los directorios son un tipo especial de archivos.
- **Enlaces:** Sistemas de archivos Unix implementar el concepto de enlace. Varios nombres se pueden asociar con un inodo. El inodo contiene un campo que contiene el número asociado con el archivo.
- **Archivos especiales de dispositivo:** En los sistemas operativos tipo Unix, los dispositivos pueden acceder a través de ficheros especiales. Un archivo especial de dispositivo no utiliza ningún espacio en el sistema de ficheros. Es sólo un punto de acceso al controlador de dispositivo. Existen dos tipos de archivos especiales: carácter y bloquear archivos especiales. La primera permite operaciones I / O en modo de caracteres mientras que el segundo requiere datos a escribir en modo bloque a través de las funciones de caché del búfer. Cuando una solicitud de E / S se realiza en un archivo especial, se envía a un (pseudo) controlador de dispositivo. Un archivo especial hace referencia a un número importante, que identifica el tipo de dispositivo, y un número menor de edad, que identifica la unidad.



# Caso de Estudio F. S.

apertura del archivo /usr/ast/mbox





# Caso de Estudio F. S.

## Estructura de Ext4

[https://ext4.wiki.kernel.org/index.php/Ext4 Disk Layout](https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout)

## Enlaces:

- <http://web.mit.edu/tytso/www/linux/ext2intro.html>
- <http://www.nongnu.org/ext2-doc/ext2.html#DEFINITIONS>
- <http://upload.wikimedia.org/wikipedia/commons/a/a2/Ext2-inode.gif>
- <http://cs.smith.edu/~nhowe/262/oldlabs/img/ext2.png>
- <https://www.cs.cmu.edu/~mihaib/fs/fs.html>







# Caso de Estudio F. S.



Del ejercicio 5 representese mediante un esquema esa misma situación del disco pero suponiendo que en vez de una FAT se dispone de un sistema de asignación tipo UNIX. Supóngase una estructura de inodo con 3punteros directos, 1 índice simple indirecto, 1 índice doble indirecto y 1 índice triple indirecto.

Dir Simplificada	FAT		Asignado a
A.txt → 6	0	X	
B.txt → 8	1	EOF	C.txt
C.txt → 12	2	BAD	
	3	EOF	
	4	15	A.txt
	5	FREE	
	6	4	A.txt
	7	13	B.txt
	8	7	B.txt
	9	EOF	A.txt
	10	14	C.txt
	11	FREE	
	12	16	C.txt
	13	EOF	B.txt
	14	1	C.txt
	15	9	A.txt
	16	10	C.txt



# Caso de Estudio F. S.



Tabla de inodos

**Inode0** (root directory)

Tipo	Dir
P. Dir 1	Bdatos 0
P. Dir 2	Null
P. Dir 3	Null
P. Ind. Simple	Null
P. Ind. Doble	Null
P. Ind. Triple	Null

**Inode1** (root directory)

Tipo	Dir
P. Dir 1	Bdatos 1
P. Dir 2	Bdatos 2
P. Dir 3	Bdatos 3
P. Ind. Simple	Bdatos 4
P. Ind. Doble	Null
P. Ind. Triple	Null

Tabla de bloques de datos

**Bloque Datos0**

P. a Tabla inodo	Nombre Archivo
Inode1	A.txt
Inode2	B.txt
Inode3	C.txt

**Bloque Datos1**

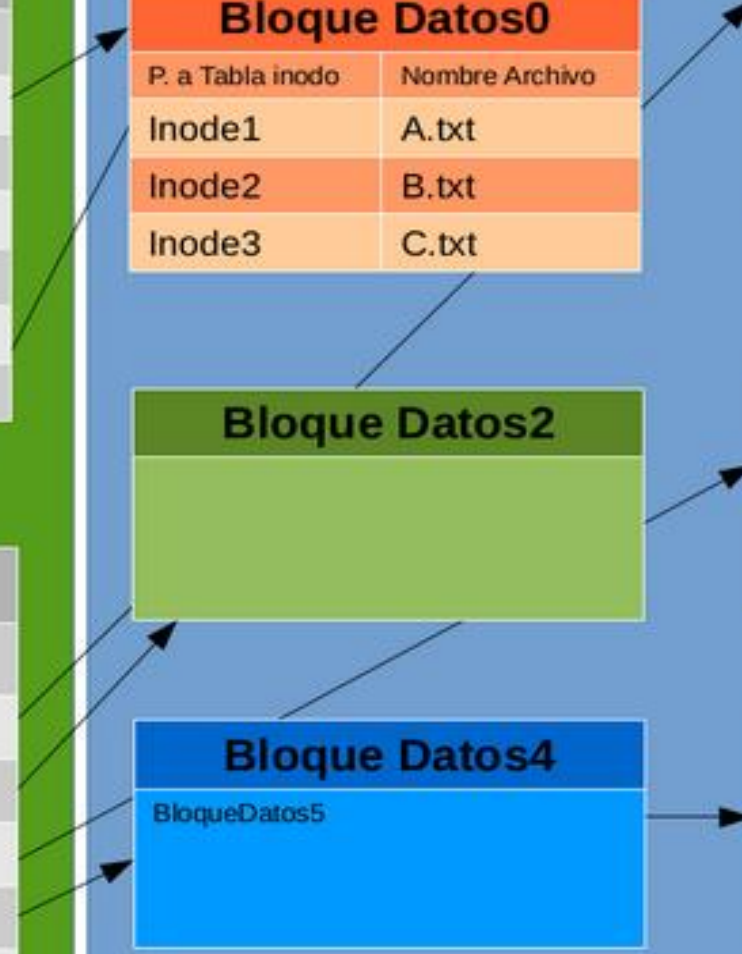
**Bloque Datos2**

**Bloque Datos3**

**Bloque Datos4**

BloqueDatos5

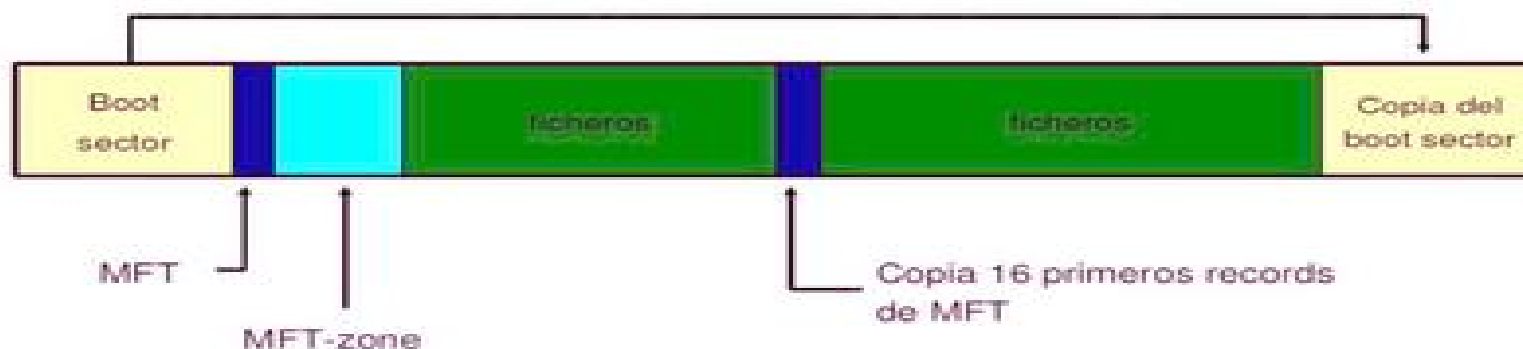
**Bloque Datos5**





# Caso de Estudio F. S.

Estructura de un volumen NTFS:



En **NTFS** todo son archivos, y existen dos categorías, "Metadatos" (Archivos que contienen información sobre el volumen) y "Normal" (Archivos normales que contienen sus datos)

La principal estructura de datos administrativos es la tabla **MFT** (Master File Table) que permite registrar los atributos del archivo y punteros a los bloques, de los archivos del volumen. Cumple una función similar a la tabla de FAT, pero es mucho más completa y sofisticada.





# Caso de Estudio F. S.

**Los archivos de Metadatos más importantes son los siguientes:**

1. **\$MFT.** -Tabla maestra de archivos - Un índice de todos los archivos .
2. **\$MftMirr.** - Una copia de seguridad de los 4 primeros registros de la MFT .
3. **\$Logfile.** - Archivo de registro de transacciones.
4. **\$Volumen.** - Número de serie, el tiempo de creación,...
5. **\$AttrDef.** - Definiciones de atributos.
6. **\$(Punto).** - Directorio raíz del disco.
7. **\$Bitmap.** - Contiene un mapa de clusters de volumen (en uso vs libre).
8. **\$Boot.** - Registro de arranque del volumen.
9. **\$BadClus.** - Listas clústeres defectuosos en el volumen .
10. **\$I30.** - Índice de elementos borrados.
11. **\$.Secure.** - Lista de control de acceso.
12. **\$.Upcase.** - Usado para convertir todos los caractreres en mayúscula a caracteres unicode minúscula.
13. **\$.Extend.** - Listado de cuotas o identificadores de objetivos



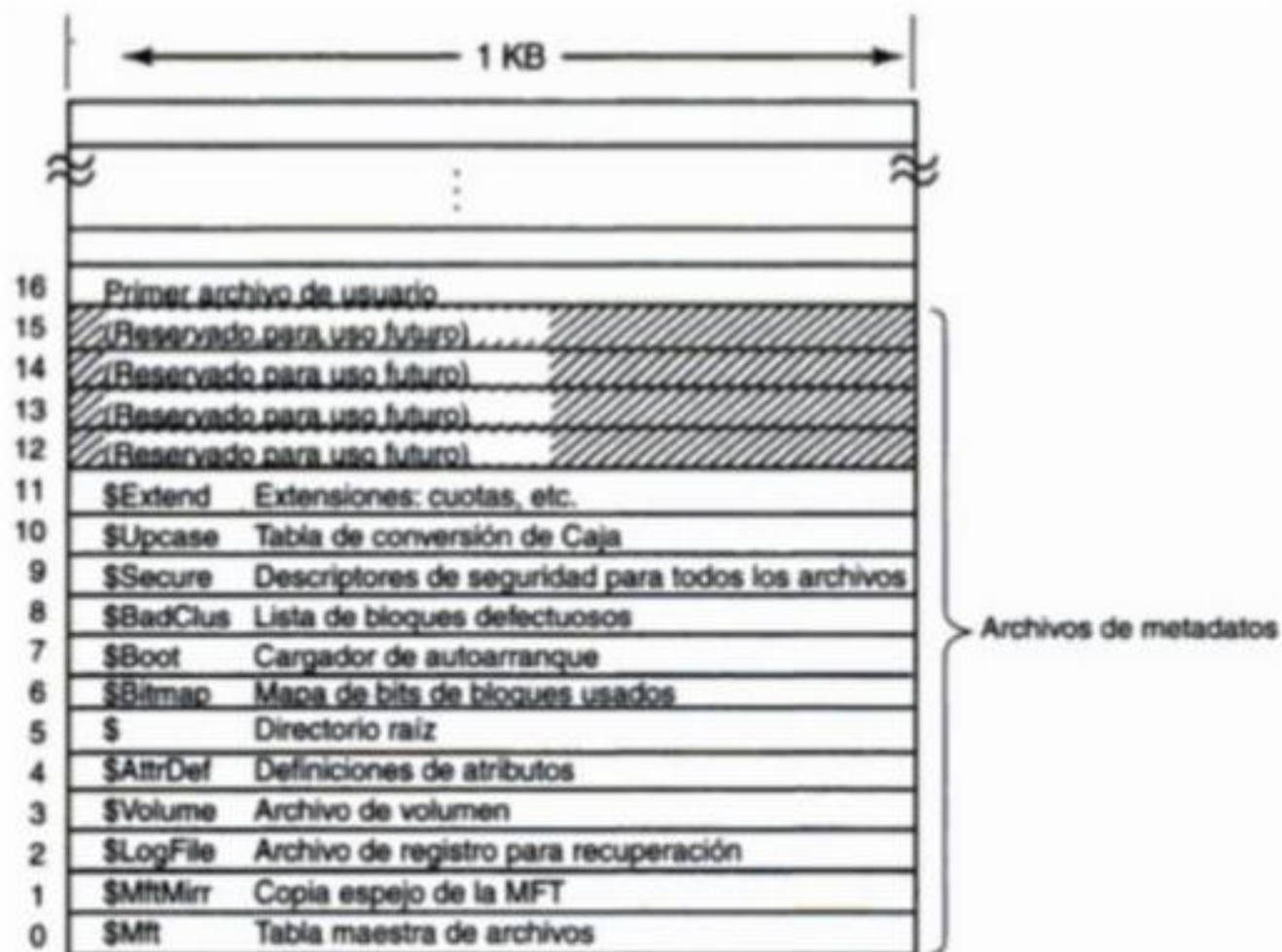
## MFT (Metadata File Table)

- Habrá una fila o record (entrada en la tabla) por cada archivo que exista en el sistema de archivos.
- Ahora bien, el concepto de archivo es bastante amplio: toda entidad almacenada individualmente es un archivo. La propia MFT es considerada un archivo y tiene su propia entrada en la MFT.
- Las primeras 16 entradas de la MFT están reservadas para almacenar archivos del sistema, es decir, información sobre el propio sistema de archivos. El resto de entradas se corresponden con los archivos y carpetas de datos.



# Caso de Estudio F. S.

## Detalles de la tabla MFT:





# Caso de Estudio F. S.

## NTFS

System File	File Name	MFT Record	Purpose of the File
Master file table	\$Mft	0	Contiene en el registro de archivo todos los archivos y directorios en el volumen NTFS. Si supera el registro se utiliza otro.
MTF Mirro	\$MtfMirr	1	Duplica el registro MFT para mejor los problemas de falla.
Log file	\$LogFile	2	Contiene información para recuperacion. Mantiene consistencia de los metadatos despues de una falla.
Volume	\$Volume	3	Contine información sobre el volumen, como el nombre o la versión
Attribute definitions	\$AttrDef	4	Lista de nombres, numeros y descriptores.
Root file name index	.	5	La carpeta raiz
Cluster bitmap	\$Bitmap	6	Representa en bits los cluster usados y no usados.
Boot sector	\$Boot	7	Incluye el BPB y el codigo de carga adicional si el volumen es booteable.
Bad cluster file	\$BadClus	8	Contine la lista de clusters dañados.
Security file	\$Secure	9	Contiene los descriptores unicos de seguridad para todos los archivos del volumen.
Uppcase table	\$Uppcase	10	Convierte caracteres para machear los caracteres UNICODE upcase.
NTFS extension file	\$Extend	11	Información adicional, como cuotas.
		12-15	Reservado para futuros usos.







# Caso de Estudio F. S.

## Características NTFS

- Cuotas
- Maximo Tamaño: 256TiB con cluster 64KiB. Tamaño maximo del archivo: 16EiB a 1KiB o 16TiB a 64KiB.
- Date resolution 100ns.
- Jornaling
- Hard Links
- Alternative Data Stream: Permite mas de un archivo con el mismo nombre. Formato filename:streamname. Comandos del PowerShell Add-Content, Clear-Content, Get-Content, Get-Item, Out-String, Remove-Item, Set-Content.
- Compresión
- Sparse File. Archivos con espacios libres.
- Volumen Shadow Copy (o snapshot). Realiza una copia de todo archivo o directorio modificado.
- Encryption
- Puntos de Montado
- Deduplicación.



## Estructura NTFS

[https://technet.microsoft.com/en-us/library/cc781134\(WS.10\).aspx](https://technet.microsoft.com/en-us/library/cc781134(WS.10).aspx)

## Enlaces Sugeridos:

<http://ftp.kolibrios.org/users/Asper/docs/NTFS/ntfsdoc.html>

<https://technet.microsoft.com/en-us/library/cc781134%28v=ws.10%29.aspx>

<http://www.tuxera.com/community/ntfs-3g-manual/>



# Caso de Estudio F. S.

## VIRTUAL FILE SYSTEMS

Es una capa virtual que sirve como capa de abstracción del sistema de archivo real. Esta capa es la que utiliza el usuario para acceder al sistema de archivo real. Esta capa permite controlar y/o manejar de la misma manera un sistema de archivo local como un sistema de archivo de red.

El primer mecanismo de sistema de archivo virtual en un sistema Unix fue introducido en el SunOS de 1985. Esto permitía acceder a su sistema de archivo local llamado UFS como acceder a sistemas remotos NFS en forma transparente.

Podría decirse que el servicio más importante que la capa VFS ofrece una caché de datos I/O uniforme. Linux mantiene cuatro cachés de datos de E/S: page cache, i-node cache, buffer cache y directory cache. El cache de páginas (page cache) combina datos de la memoria y archivos virtuales. El buffer cache es una interfaz con el dispositivo de bloques y cachea los meta-datos de discos de bloques. El cache de i-nodo mantiene el acceso reciente a los i-nodos de archivo. El cache de directorio (d-cache) mantiene en la memoria de un árbol que representa una parte de la estructura de directorios del sistema de archivos.

La estructura de VFS de linux se parece a la estructura de ext2.



# Caso de Estudio F. S.

## VFS Superblock

Todo sistema de archivo montado esta representado por el VFS superblock. El VFS contiene:

- Device/Dispositivo: Este es el identificador de dispositivo. por ejemplo /dev/sda1
- Puntero a inodos/Inode pointers: Apunta al primer inodo del sistema de archivo. El puntero de inodo cubierto representa el inodo del directorio donde esta montado dentro del sistema. Si es el root file system no tiene puntero cubierto.covered pointer,
- Tamaño de Bloque/Blocksize: es el tamaño de bloque en el sistema de archivo. ej 4096 bytes.
- Operaciones de superbloque/Superblock operations: Apunta a grupo de rutinas de superbloque. Entre otras cosas estas rutinas son las que permite la lectura y escritura de inodos y superbloques.
- Tipo de sistema de archivo/File System type: Es el puntero a los datos de estructura del tipo de sistema de archivo.
- Especifico del sistema de archivo/File System specific: Es el puntero a la información necesario para este sistema de archivo.





# Caso de Estudio F. S.

## El VFS inodo

Como el sistema de archivo Ext2, todo archivo, directorio y otros es representado por algún VFS inodo. FS inode.

La información de cada inodo VFS es construida desde la información extraída del sistema de archivo mediante rutinas específicas. Cada VFS inodo existe solo en memoria del sistema y es mantenida dentro del cache de inodo. VFS inodo contiene los siguientes datos:

- Dispositivo/device: Es el identificador de dispositivo donde se encuentra el inodo real.ode represents,
- Numero de Inodo/inode number: Es un numero unico de inodo dentro del sistema de archivo. Es una combinación entre el dispositivo y el el numero de inodo.
- Modo/mode: Como el EXT2 este campo describe los permisos al inodo.
- ID de usuario/user ids: Describe los ids de usuario
- Tiempos/times: De creación, modificación, y de escritura.
- Tamaño de Bloque/block size: Tamaño de bloques.
- Operaciones de inodo/inode operations: Es un puntero a la direcciones de los procedimientos de inodo. Por ejemplo la opeacion de truncar el archivo que representa el inodo.
- Cuenta/count: Es el numero de componetes de sistema que estan usando este inodo. Cuando la suma llega a cero puede ser descartado o reusado.
- lock: Este campo es usado cuando el inodo es lockeado cuando es usado por el sistema.
- Sucio/dirty: Indica si el inodo VFS ha sido escrito y el sistema de archivo por debajo necesita ser modificado.



# Caso de Estudio F. S.

## Registro de sistema de archivos

Todo en linux todo nuevo sistema de archivo debe ser registrado. Los sistemas de archivos pueden ser creados como modulos y cargados bajo demanda. Cuando son cargados el kernel registra el sistema de archivo.

Uno puede ver si el archivo esta registrado haciendo un cat al archivo /proc/filesystems.

