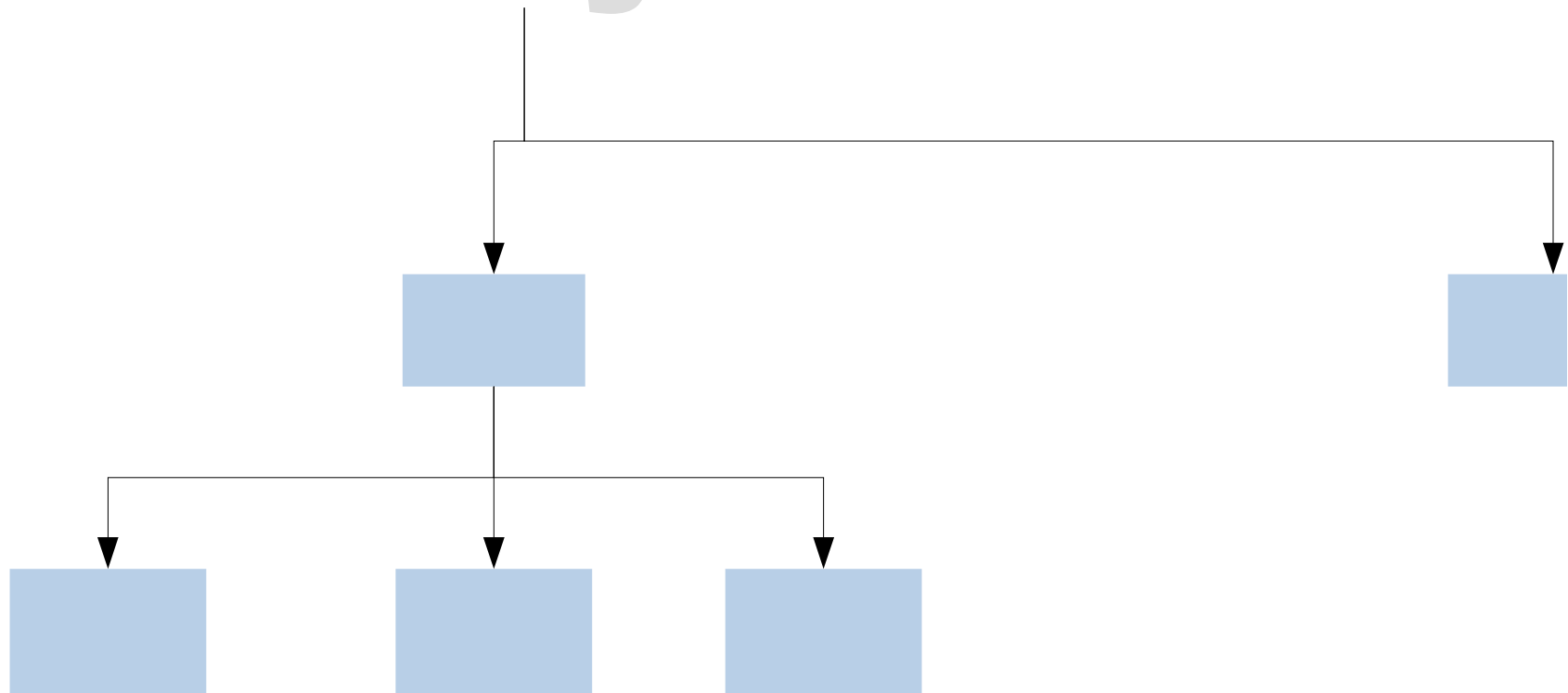


# Organización de Datos

File System





# Indice

- Introducción de las funciones del Sistema de Archivo
- Características del Sistema de archivo
- Tipos de Sistema de archivo
- 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 son el guardado y recuperación de la información; Administración del espacio libre y el guardado de los atributos de seguridad del sistema operativo (En los filesystem nuevos).

El conjunto de programas del sistema operativo encargados de proveer la visión lógica de la información almacenada a usuarios y programas conforman el Sistema de Archivo.

Estos programas se encargan de:

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



# Tipos de Sistema de archivo

Los sistemas de archivos pueden ser clasificados por medio, de red y de proposito especial.

- **Discos de plato magnetico o Flash** (acceso aleatorio): FAT, NTFS, HFS+, UFS, EXT2/3/4, XFS, BRTFS, 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



# Aspectos a evaluar en un sistema de archivos

- 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, Chequeos, Defrag, Tunning, Resize, Undelete, etc.
- Mecanismos de permisos y restricción.
  - ACL
  - Capacidades
- Integridad de la información
- Limitaciones impuestas por diseño.
- Extras
  - Encriptación a nivel archivo o directorio
  - Compresión a nivel archivo o directorio
  - Deduplicación



## Casos de estudio

### Organización de Datos

- **FAT**
- **UNIX**
- **NTFS**



# FAT

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.

- **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 clusters libres. Cada cluster está enlazado con el siguiente mediante un puntero. Si un determinado cluster no se ocupa por completo, su espacio remanente se desperdicia.



# FAT

## FAT - Estructuras

- Boot (sector de arranque)
- Región FAT (dos copias)
- Directorio raiz (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 raiz 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)		





## FAT - Tabla de asignacion

La tabla de asignación de archivos consta de una lista de entradas. Cada entrada contiene información sobre un clúster:

- La dirección del siguiente clúster en la cadena.
- Si es pertinente, la indicación de "fin de archivo" (que es también el fin de la cadena).
- 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 (puede ser usado por un archivo).
- El tamaño de estas entradas también depende de la variante FAT en uso: FAT16 usa entradas de 16 bits, FAT32 usa entradas de 32 bits, etc.



## FAT - Directorio Raíz

Este índice es un tipo especial de archivo que almacena las subcarpetas y archivos que componen cada carpeta. Cada entrada del directorio contiene el nombre del archivo o carpeta (máximo 8 caracteres), su extensión (máximo 3 caracteres), sus atributos (archivo, carpeta, oculto, del sistema, o volumen), la fecha y hora de creación, la dirección del primer cluster donde están los datos, y por último, el 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.



# Sistemas de archivos UNIX

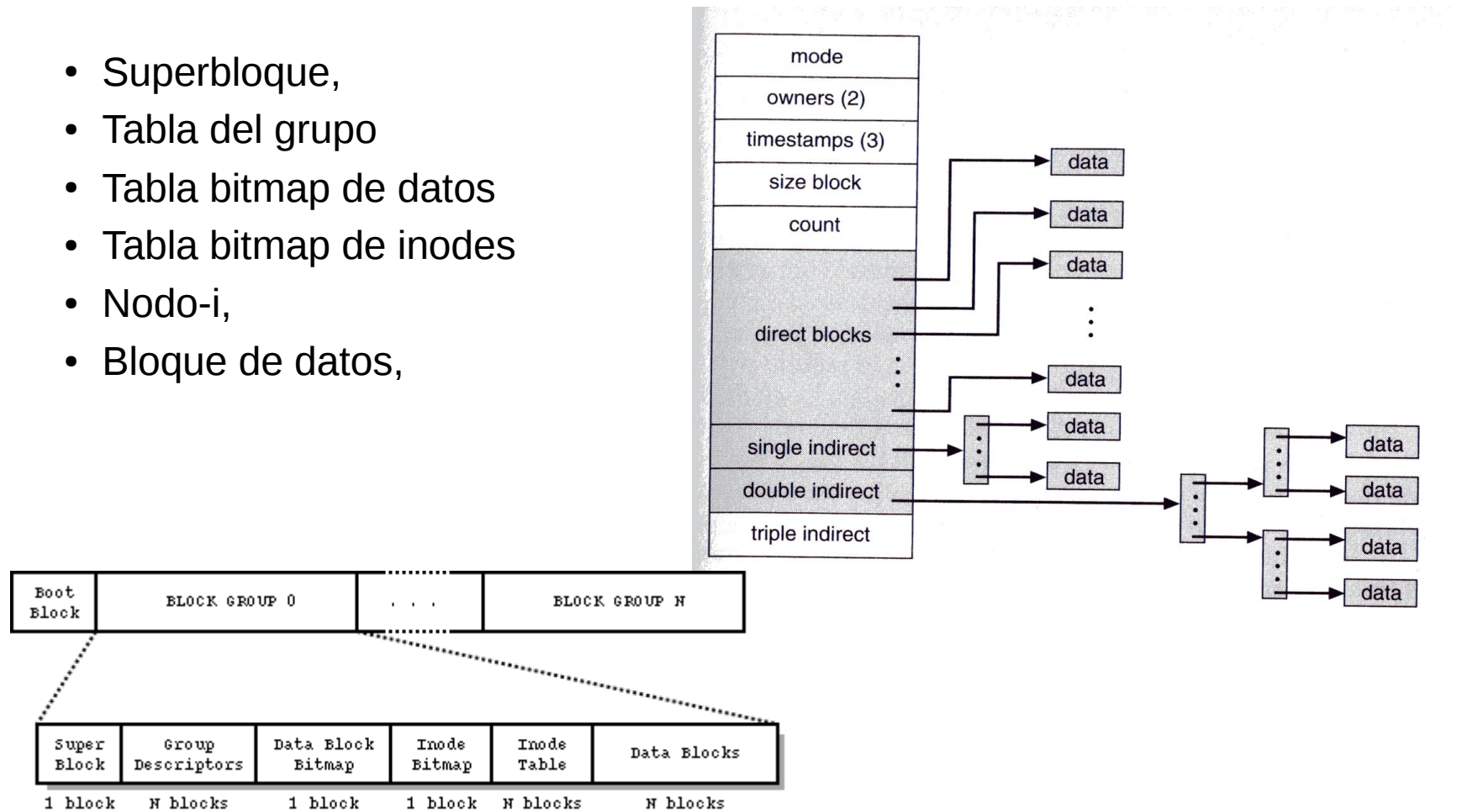
Los conceptos centrales son:

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

# Sistemas de archivos UNIX

## Ext2/3/4

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





# Sistemas de archivos UNIX

## Características Ext4

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



# Sistemas de archivos UNIX

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

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.

El nodo-i contiene los números de varios bloques de datos, que se utilizan para almacenar los datos en el archivo. Sólo hay espacio para unos pocos números de bloques de datos en el nodo-i; en cualquier caso, si se necesitan más, más espacio para punteros a los bloques de datos son colocados de forma dinámica.

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.



# Sistemas de archivos UNIX

- **Superblock** : Contiene toda la información de la configuración del sistema de archivo. Contiene información el total de inodos y bloques del sistemas de archivo y cuantos estan libres, cuando fue montado, etc.
- **Inode**: Cada archivo esta representado por una estructura llamado un inode. Cada inodo contiene la descripción del fichero, tipo de archivo, los derechos de acceso, marcas de tiempo, el tamaño y los punteros a los bloques de datos. El **bloque de indirección** es un inodo que se usa para apuntar a otros inodos cuando la cantidad de inodos se acaban.
- **Directorios**: Los directorios se estructuran en un arbol 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 inode 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.



# Sistemas de archivos UNIX

- Estructura de Ext4

[https://ext4.wiki.kernel.org/index.php/Ext4\\_Disk\\_Layout](https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout)





# Sistemas de archivos UNIX

- **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>



# Sistema de Archivo

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



# Sistema de Archivo

Organización de Datos

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

Bloque Datos0	
P. a Tabla inodo	Nombre Archivo
Inode1	A.txt
Inode2	B.txt
Inode3	C.txt

Bloque Datos2	

Bloque Datos4	
BloqueDatos5	

Tabla de bloques de datos

Bloque Datos1	

Bloque Datos3	

Bloque Datos5	



# NTFS

## Características NTFS

- Cuotas
- Maximo Tamaño: 256TiB con cluster 64KiB. Tamaño maximo del archivo: 16EiB a 1KiB o 16TiB a 64KiB.
- Date resolution 100ns.
- Journaling
- 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.



# NTFS

## Organización de Volumen NTFS

- Se tiene cuatro estructuras basicas:
  - NTFS Boot Sector --> almacena información sobre el diseño del volumen y las estructuras del sistema de archivos, el codigo de arranque de algunos windows servers.
  - Master File Table (MTF) --> Contiene la información necesaria para recuperar los archivos de la partición NTFS , como los atributos de un archivo .
  - File System Data --> Almacena los datos que no se encuentran dentro de la tabla maestra de archivos.
  - Master File Table Copy --> Copia del MFT.

## Estructura NTFS

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



# NTFS

## 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.



# 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 usuados.
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.
Upcase table	\$Upcase	10	Convierte caracteres para machear los caracteres UNICODE upcase.
NTFS extension file	\$Extend	11	Información adicional, como cuotas.
		12-15	Reservado para futuros usos.



# NTFS

## 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/>





# VFS

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 es 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.



# VFS

## 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 archvio/File System specific: Es el puntero a la información necesario para este sistema de archivo.



# VFS

## 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 conuinación 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.



# VFS

## 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`.



# VFS

## Bibliografía

- <http://www.ibm.com/developerworks/library/l-fuse/>
- <http://www.tldp.org/LDP/tlk/fs/filesystem.html>
- <http://www.tldp.org/LDP/khg/HyperNews/get/fs/vfstour.html>
- [https://kaiwantech.files.wordpress.com/2009/08/vfs\\_msdos\\_31.png](https://kaiwantech.files.wordpress.com/2009/08/vfs_msdos_31.png)
- <http://www.inf.fu-berlin.de/lehre/SS01/OS/Lectures/Lecture16.pdf>
- The Linux VFS, Chapter 4 of Linux File Systems by Moshe Bar (McGraw-Hill, 2001). ISBN 0-07-212955-7
- Chapter 12 of Understanding the Linux Kernel by Daniel P. Bovet, Marco Cesati (O'Reilly Media, 2005). ISBN 0-596-00565-2



# FUSE

FUSE permite implementar un sistema de archivo funcional en el espacio de usuario de programas. Las características que incluyen son:

- Una librería simple.
- El módulo está incorporado al kernel.
- Correr el sistema de archivo en "userspace", esto implica que no necesita ser usuario privilegiado.
- Corre en linux 2.4.X en adelante.
- Ha probado ser estable.

FUSE fue desarrollado para soportar el sistema de archivo AVFS, posteriormente se convirtió en un proyecto aparte.



# FUSE

```
Struct fuse_operations {  
    int (*getattr) (const char *, struct stat *);  
    int (*readlink) (const char *, char *, size_t);  
    int (*getdir) (const char *, fuse_dirh_t, fuse_dirfil_t);  
    int (*mknod) (const char *, mode_t, dev_t);  
    int (*mkdir) (const char *, mode_t);  
    int (*unlink) (const char *);  
    int (*rmdir) (const char *);  
    int (*symlink) (const char *, const char *);  
    int (*rename) (const char *, const char *);  
    int (*link) (const char *, const char *);  
    int (*chmod) (const char *, mode_t);  
    int (*chown) (const char *, uid_t, gid_t);  
    int (*truncate) (const char *, off_t);  
    int (*utime) (const char *, struct utimbuf *);  
    int (*open) (const char *, struct fuse_file_info *);  
    int (*read) (const char *, char *, size_t, off_t, struct fuse_file_info *);  
    int (*write) (const char *, const char *, size_t, off_t, struct fuse_file_info *);  
    int (*statfs) (const char *, struct statfs *);  
    int (*flush) (const char *, struct fuse_file_info *);  
    int (*release) (const char *, struct fuse_file_info *);  
    int (*fsync) (const char *, int, struct fuse_file_info *);  
    int (*setxattr) (const char *, const char *, const char *, size_t, int);  
    int (*getxattr) (const char *, const char *, char *, size_t);  
    int (*listxattr) (const char *, char *, size_t);  
    int (*removexattr) (const char *, const char *);  
};
```



# FUSE

Ejemplo: Hello Word en FUSE

----- hello.c -----

```
/*
FUSE: Filesystem in Userspace
Copyright (C) 2001-2007 Miklos Szeredi <miklos@szereadi.hu>

This program can be distributed under the terms of the GNU GPL.
See the file COPYING.

gcc -Wall hello.c `pkg-config fuse --cflags --libs` -o hello
*/

#define FUSE_USE_VERSION 26

#include <fuse.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>

static const char *hello_str = "Hello World!\n";
static const char *hello_path = "/hello";
```



# FUSE

```
// Operacion de lectura de atributos.
static int hello_getattr(const char *path, struct stat *stbuf)
{
    int res = 0;

    memset(stbuf, 0, sizeof(struct stat));
    if (strcmp(path, "/") == 0) {
        stbuf->st_mode = S_IFDIR | 0755;
        stbuf->st_nlink = 2;
    } else if (strcmp(path, hello_path) == 0) {
        stbuf->st_mode = S_IFREG | 0444;
        stbuf->st_nlink = 1;
        stbuf->st_size = strlen(hello_str);
    } else
        res = -ENOENT;

    return res;
}

// Operacion de lectura de directorio
static int hello_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
                        off_t offset, struct fuse_file_info *fi)
{
    (void) offset;
    (void) fi;

    if (strcmp(path, "/") != 0)
        return -ENOENT;

    filler(buf, ".", NULL, 0);
    filler(buf, "..", NULL, 0);
    filler(buf, hello_path + 1, NULL, 0);

    return 0;
}
```



# FUSE

```
// Operacion de apertura.
static int hello_open(const char *path, struct fuse_file_info *fi)
{
    if (strcmp(path, hello_path) != 0)
        return -ENOENT;

    if ((fi->flags & 3) != O_RDONLY)
        return -EACCES;

    return 0;
}

// Operacion de lectura
static int hello_read(const char *path, char *buf, size_t size, off_t offset,
                     struct fuse_file_info *fi)
{
    size_t len;
    (void) fi;
    if (strcmp(path, hello_path) != 0)
        return -ENOENT;

    len = strlen(hello_str);
    if (offset < len) {
        if (offset + size > len)
            size = len - offset;
        memcpy(buf, hello_str + offset, size);
    } else
        size = 0;

    return size;
}
```



# FUSE

```
// Definicion de operaciones...
static struct fuse_operations hello_oper = {
    .getattr  = hello_getattr,
    .readdir  = hello_readdir,
    .open     = hello_open,
    .read     = hello_read,
};

int main(int argc, char *argv[])
{
    return fuse_main(argc, argv, &hello_oper, NULL);
}
```

-----fin-- hello.c -----



# FUSE

Uso de hello...

```
apt install libfuse-dev
```

```
gcc hello2.c -o programa -lfuse -D_FILE_OFFSET_BITS=64
```

```
o
```

```
gcc hello2.c -o programa -lfuse -D_FILE_OFFSET_BITS=64 -DFUSE_USE_VERSION=22
```

```
~/fuse/example$ mkdir /tmp/fuse
```

```
~/fuse/example$ ./hello /tmp/fuse
```

```
~/fuse/example$ ls -l /tmp/fuse
```

```
total 0
```

```
-r--r--r--  1 root root 13 Jan  1  1970 hello
```

```
~/fuse/example$ cat /tmp/fuse/hello
```

```
Hello World!
```

```
~/fuse/example$ fusermount -u /tmp/fuse
```

```
~/fuse/example$
```



# FUSE

## Bibliografía

<http://fuse.sourceforge.net/>  
[http://fuse.sourceforge.net/doxygen/null\\_8c.html](http://fuse.sourceforge.net/doxygen/null_8c.html)  
<https://code.google.com/p/pngdrive/>  
<http://www.buanzo.com.ar/lin/FUSE.html>