

Sistemas Operativos

Curso
2014-2015

Práctica 2: Sistema de Ficheros

Dpto. ArTeCS UCM

Christian Tenllado José Ignacio Gómez Katzalin Olcoz José Luis Risco Joaquín Recas Juan
Carlos Saez

1 Objetivo

2 Mi Sistema De Ficheros

3 Librería myFS

4 Fuse

5 Parte Obligatoria

Objetivo

- Crear nuestro propio sistema de ficheros sobre un disco virtual representado por un fichero del SF nativo de Linux
- Montar nuestro sistema de ficheros con FUSE para poder interactuar con él con las herramientas habituales (ls, cat, nautilus, ...)

1 Objetivo

2 **Mi Sistema De Ficheros**

3 Librería myFS

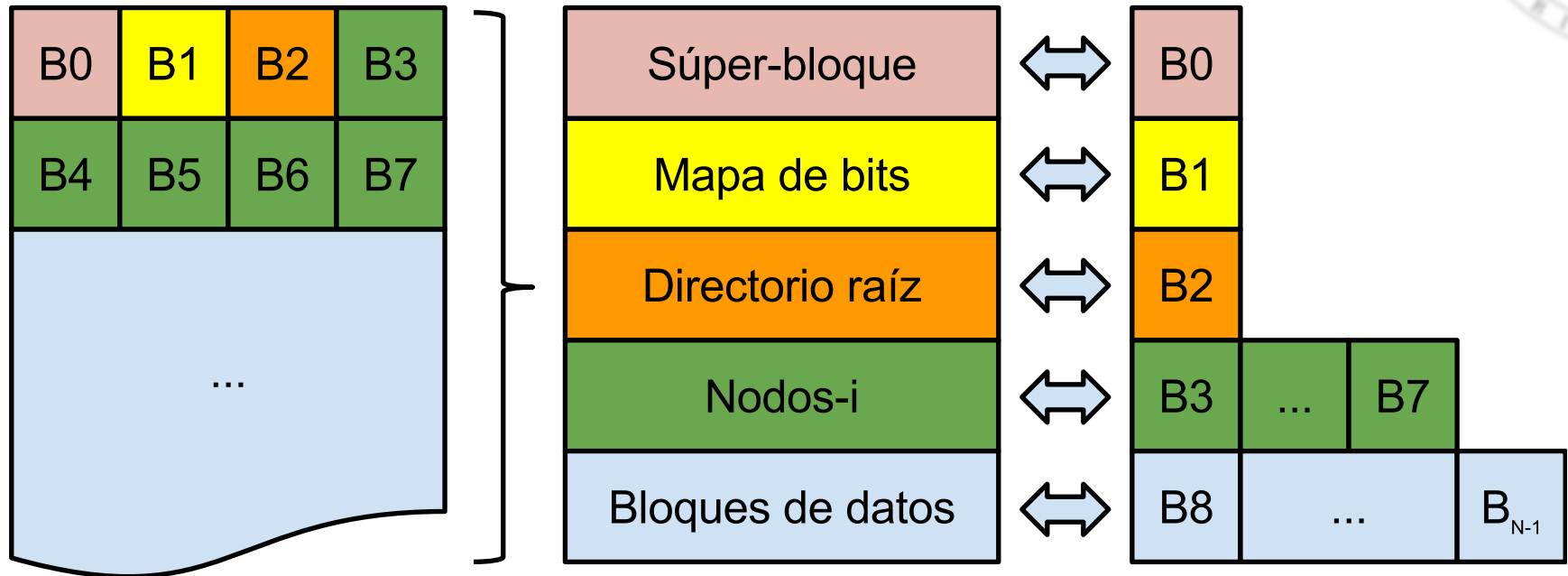
4 Fuse

5 Parte Obligatoria

Sistema de Ficheros Simple

- Sólo un directorio, de tamaño 1 bloque
 - Limitamos el tamaño del nombre de los ficheros
- Estructura del nodo-i:
 - sólo enlaces directos, todo el índice en el nodo-i
- 1 bloque para el superbloque
- 1 bloque para el Mapa de bits
- Tabla de nodos-i: 5 bloques
- Resto para bloques de datos
 - $(\text{Tamaño Disco Virtual} / \text{Tamaño de bloque})$
 - Debe haber 1 como mínimo
- Limitamos el tamaño de los ficheros (en bloques)

Estructura del disco Virtual



Archivo \Leftrightarrow SF \Leftrightarrow Conjunto de bloques

Correspondencia estructura SF \Leftrightarrow bloques del archivo

1 Objetivo

2 Mi Sistema De Ficheros

3 Librería myFS

4 Fuse

5 Parte Obligatoria

myFS.h (I)



Sistema de Ficheros

```
typedef struct MiSistemaDeFicheros {  
    int fdDiscoVirtual;           // FD del fichero de mi SF  
    EstructuraSuperBloque superBloque; // Superbloque  
    BIT mapaDeBits[NUM_BITS];     // Mapa de bits  
    EstructuraDirectorio directorio; // Directorio raíz  
    EstructuraNodoI *nodosI[MAX_NODOSI]; // Punteros a Nodos-i  
    int numNodosLibres;           // Número de nodos-i libres  
} MiSistemaDeFicheros;
```


myFS.h (II)



Superbloque

```
typedef struct EstructuraSuperBloque {  
    time_t fechaCreacion;        // Fecha en la que se creó el SF  
    int tamDiscoEnBloques;       // Núm. de bloques en disco  
    int numBloquesLibres;        // Núm. de bloques libres  
    int tamBloque;               // Tamaño de bloque  
    int maxTamNombreArchivo;     // Tamaño máx. de nombre de archivo  
    int maxBloquesPorArchivo;    // Tamaño máx. de bloques por archivo  
} EstructuraSuperBloque;
```

myFS.h (III)



Directorio

```
typedef struct EstructuraDirectorio {  
    int numArchivos;  
    EstructuraArchivo archivos[MAX_ARCHIVOS_POR_DIRECTORIO];  
} EstructuraDirectorio;  
  
typedef struct EstructuraArchivo {  
    int idxNodoI; // Nodo-i asociado  
    char nombreArchivo[MAX_TAM_NOMBRE_ARCHIVO + 1]; // Nombre archivo  
    BOOLEAN libre; // Archivo libre  
} EstructuraArchivo;
```

myFS.h (IV)



Nodo-i

```
typedef struct EstructuraNodoI {  
    int numBloques;                // Núm. bloques  
    int tamArchivo;                // Tamaño archivo  
    time_t tiempoModificado;       // Tiemp. de modif.  
    DISK_LBA idxBloques[MAX_BLOQUES_POR_ARCHIVO]; // Bloques  
    BOOLEAN libre;                 // Nodo libre  
} EstructuraNodoI;
```

myFS.h (V)



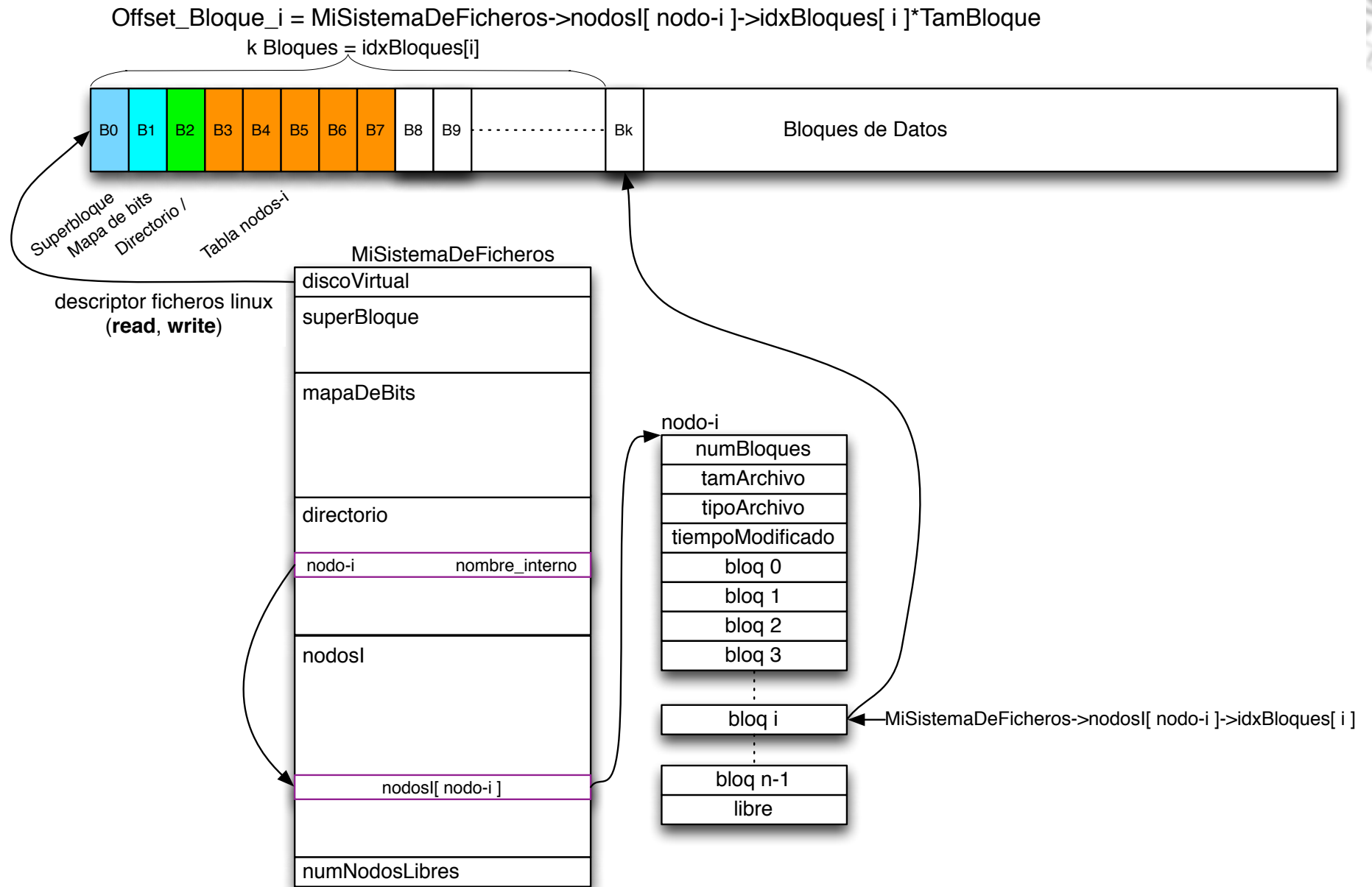
Macros

```
#define false 0
#define true 1

#define BIT unsigned
#define TAM_BLOQUE_BYTES 4096
#define NUM_BITS (TAM_BLOQUE_BYTES/sizeof(BIT))
#define MAX_BLOQUES_CON_NODOSI 5
#define MAX_BLOQUES_POR_ARCHIVO 100
#define MAX_ARCHIVOS_POR_DIRECTORIO 100
#define MAX_TAM_NOMBRE_ARCHIVO 15
#define DISK_LBA int
#define BOOLEAN int

#define SUPERBLOQUE_IDX 0
#define MAPA_BITS_IDX 1
#define DIRECTORIO_IDX 2
#define NODOI_IDX 3
```

Acceso a MiSistemaDeFicheros



Funciones Manejo del SF (I)

■ Escritura sobre disco virtual:

- `int` escribeSuperBloque(MiSistemaDeFicheros* miSistemaDeFicheros)
- `int` escribeMapaDeBits(MiSistemaDeFicheros* miSistemaDeFicheros)
- `int` escribeDirectorio(MiSistemaDeFicheros* miSistemaDeFicheros)
- `int` escribeNodoI(MiSistemaDeFicheros* miSistemaDeFicheros, `int` numNodoI, EstructuraNodoI* nodoI)

■ Lectura del disco virtual:

- `int` leeNodoI(MiSistemaDeFicheros* miSistemaDeFicheros, `int` numNodoI, estructuraNodoI* nodoI)

■ Auxiliares:

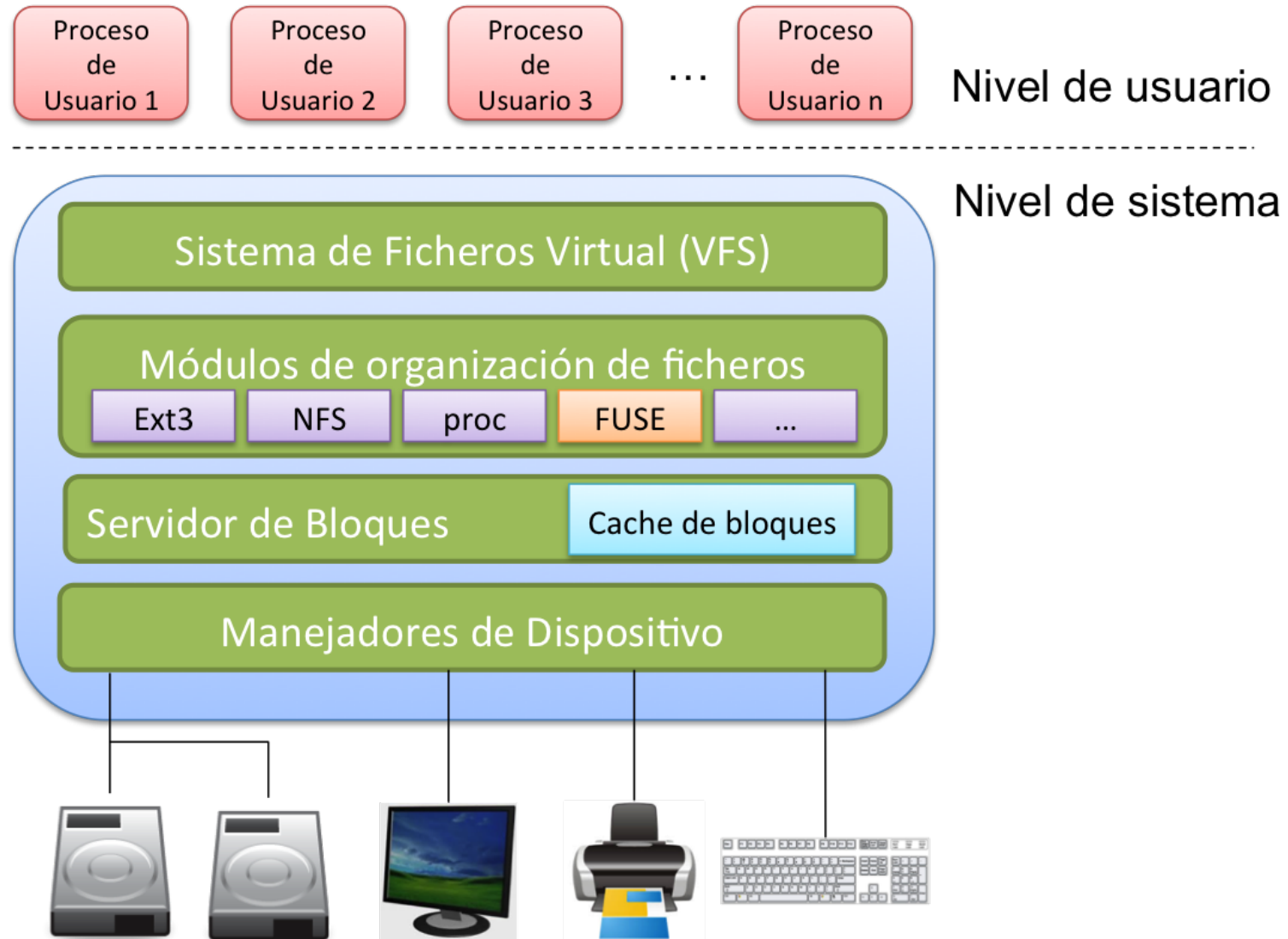
- `int` calculaPosNodoI(`int` numNodoI)

Funciones Manejo del SF (II)

- `void` `initSuperBloque`(`MiSistemaDeFicheros*` `miSistemaDeFicheros`, `int` `tamDisco`)
- `void` `initNodosI`(`MiSistemaDeFicheros*` `miSistemaDeFicheros`)
- `void` `copiaNodoI`(`EstructuraNodoI*` `dest`, `EstructuraNodoI*` `src`)
- `int` `buscaNodoLibre`(`MiSistemaDeFicheros*` `miSistemaDeFicheros`)
- `int` `buscaPosDirectorio`(`MiSistemaDeFicheros*` `miSistemaDeFicheros`, `char*` `nombre`)
- `int` `myQuota`(`MiSistemaDeFicheros*` `miSistemaDeFicheros`)
- `void` `reservaBloquesNodosI`(`MiSistemaDeFicheros*` `miSistemaDeFicheros`, `DISK_LBA` `idxBloques[]`, `int` `numBloques`)
- `void` `myFree`(`MiSistemaDeFicheros*` `miSistemaDeFicheros`)

- 1 Objetivo
- 2 Mi Sistema De Ficheros
- 3 Librería myFS
- 4 Fuse**
- 5 Parte Obligatoria

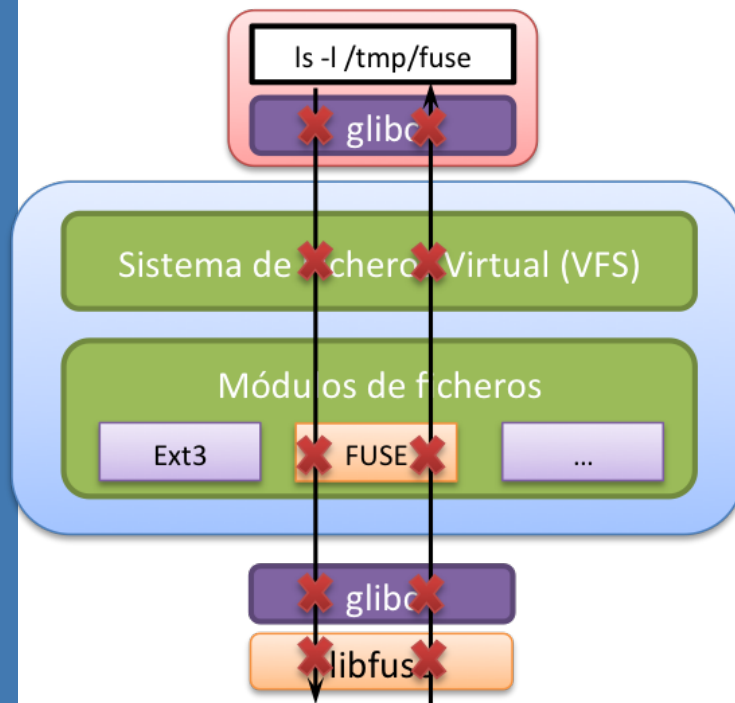
Estructura del servidor de ficheros



Módulo de organización de ficheros

- Modelo de fichero del SO
- Mapeo direcciones lógicas de bloques a direcciones físicas.
- Manejo de los descriptores internos de fichero
 - i-nodos de UNIX
 - registros de Windows NT
- Un módulo por cada tipo de SF soportado
 - Incluye pseudo ficheros como /proc

FUSE: Filesystem in Userspace



- Módulo de kernel: manejador SF Fuse
- Montaje:
 - 1 solicitud proceso a módulo (/proc)
 - 2 registro SF Fuse en punto de montaje
 - 3 creación socket entre módulo y proceso
 - 4 Accesos al SF redirigidas al proceso por el socket
- Acciones realizadas por el proceso de usuario

FUSE: ¿Cómo se usa?

- 1 Creamos controlador (archivo .c)
 - Hay que incluir fuse.h y enlazar con libfuse
- 2 Declarar estructura llamada `fuse_operations`
 - Contiene punteros a funciones que serán llamados por cada operación
- 3 Se termina el programa con la llamada a `fuse_main`
 - El proceso se queda atendiendo al socket

FUSE: fuse_operations

- `int (*getattr) (const char *, struct stat *);`
- `int (*open) (const char *, struct fuse_file_info *);`
- `int (*read) (const char *, char *, size_t, off_t, struct fuse_file_info *);`
- `int (*readdir) (const char *, void *, fuse_fill_dir_t, off_t, struct fuse_file_info *);`
- `int (*mknod) (const char *, mode_t, dev_t);`
- `int (*unlink) (const char *);`
- `int (*rename) (const char *, const char *);`
- `int (*truncate) (const char *, off_t);`
- `int (*write) (const char *, const char *, size_t, off_t, struct fuse_file_info *);`
- Auxiliar:
 - `int resizeInodo(uint64_t idxNodoI, size_t newSize)`

- 1 Objetivo
- 2 Mi Sistema De Ficheros
- 3 Librería myFS
- 4 Fuse
- 5 Parte Obligatoria**

Programa de la Práctica

- Argumentos:
 - -t tamaño_en_bytes_del_SF
 - -a fichero_que_representará_el_disco
 - -f argumentos_a_fuse

terminal

```
./MiSistemaDeFicheros -t 2097152 -a disco-virtual -f '-d -s punto-montaje'
```

- ¿Qué hace?
 - 1 Crea el sistema de ficheros sobre el disco virtual
 - 2 Invoca `fuse_main` con los argumentos pasados con la opción `-f`
 - Dejaremos montado nuestro sistema de ficheros en un directorio
 - Podremos interactuar con nuestro sistema de ficheros con los comandos habituales (`ls`, `cat`, `nautilus`, ...)

¿Qué debe hacer el alumno?

Lista de tareas

- Implementar las operaciones `unlink` y `read`:
 - Devuelve un valor negativo en caso de error
 - El mínimo entre el número de bytes del fichero y el número de bytes solicitados (0 si no hay más datos)
- Registrar estas operaciones en el campo correspondiente de `fuse_operations`.
- Desarrollar un script de test (descrito en el guión de la práctica)