ID LABS

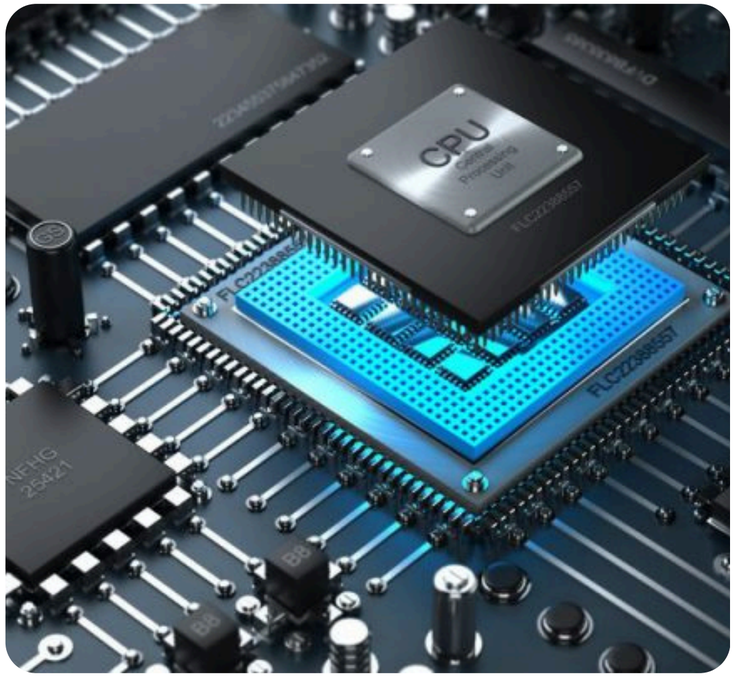
FS-M8K

SISTEMA DE FICHEROS CON FUSE

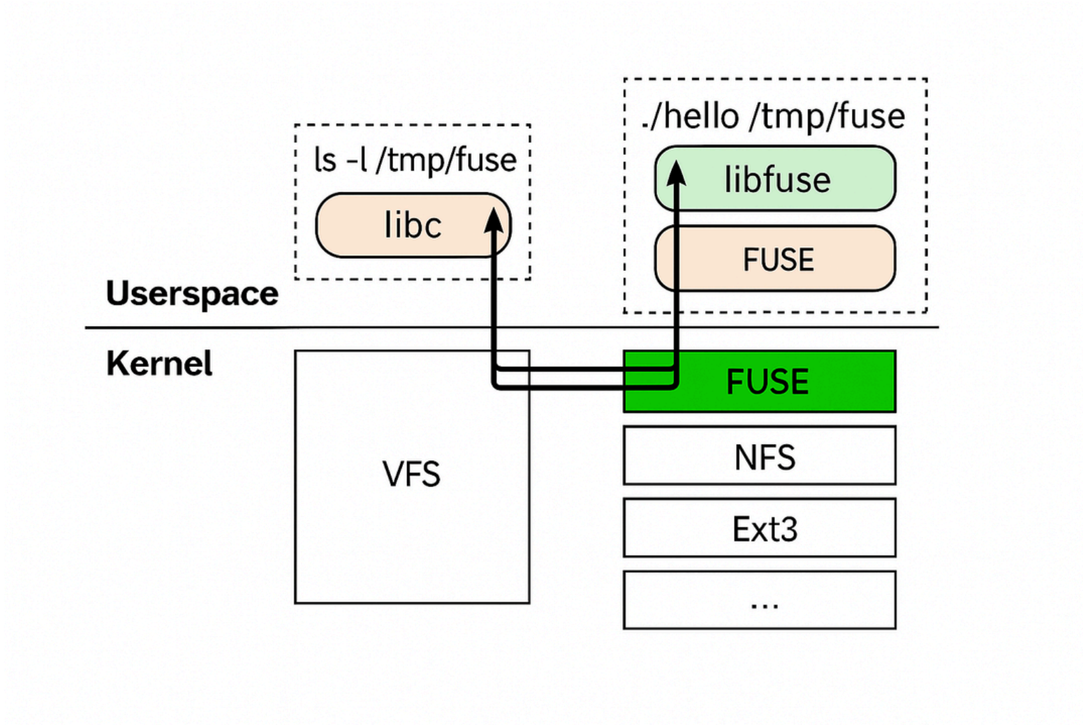
GRUPO K

INTRODUCCIÓN

Este proyecto consiste en el diseño e implementación de un sistema de archivos ligero, pensado para entornos con recursos limitados, como sistemas embebidos, microcontroladores o sistemas IoT.
Posee una estructuras como la de Ext2, compuesta por: superbloque, inodos y entradas.
El sistema es totalmente funcional y puede montarse en Linux mediante FUSE (Filesystem in Userspace).



MONTAJE CON FUSE



01

¿QUÉ ES FUSE?

Es un interfaz de software para sistemas operativos Unix-like que permite a usuarios no privilegiados crear y gestionar sus propios sistemas de archivos sin tener que modificar el kernel

02

COMUNICACIÓN ENTRE USUARIO, VFS Y FUSE

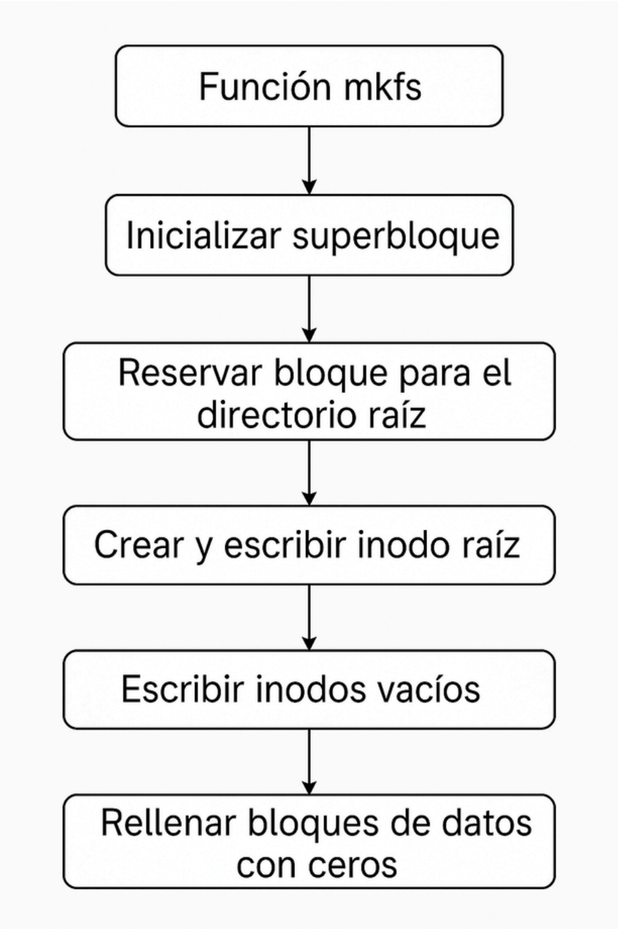
El usuario ejecuta comandos (como ls), que se traducen en syscalls gestionadas por el VFS. Si el FS es FUSE, el VFS redirige la operación al programa en espacio de usuario que maneja la lógica del sistema.

ARQUITECTURA DEL SISTEMA DE ARCHIVOS

```c
// Superbloque
typedef struct {
    int magic_number;
    int num_blocks;
    int block_size;
    int num_inodes;
    int root_inode_index;

    int free_inode_bitmap[NUM_INODES
    int free_block_bitmap[NUM_BLOCKS
} Superblock;
```

```c
// Entrada de un directorio
typedef struct {
    char name[FILENAME_LEN];
    int inode_index;
} DirEntry;

// Inodo
typedef struct {
    int used;                          // 1 si ocupado, 0 si libre
    int is_directory;                  // 1 si es directorio
    int size;                          // en bytes
    int block_pointers[DIRECT_BLOCKS]; // bloques directos
    int indirect1;                     // bloque indirecto simple
    int indirect2;                     // bloque indirecto doble
    long atime;                        // tiempo de último acceso
    long mtime;                        // tiempo de última modificación
    long ctime;                        // tiempo de cambio
    mode_t mode;                       // permisos y tipo de archivo
} Inode;
```

FORMATO Y CREACIÓN DEL SISTEMA (MKFS)

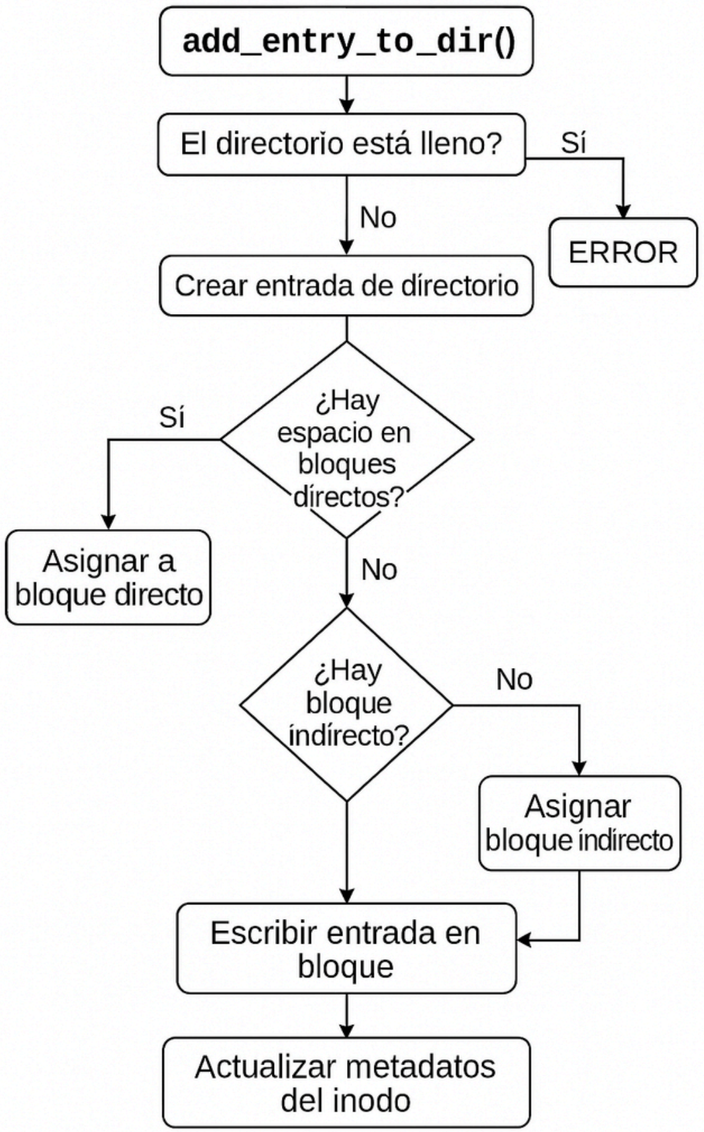



```c
void mkfs() {
    FILE* fp = fopen(FS_IMAGE, "w+b");
    if (!fp) {
        perror("fopen");
        return;
    }

    // 1. Inicializar superbloque
    Superblock sb;
    init_superblock(&sb);
    printf("[DEBUG] Bitmap de inodos tras init: %08X\n", sb.free_inode_bitmap[0]);

    // 2.- Vamos a reservar un bloque de datos para el directorio raíz
    int root_block = allocate_block(&sb);
    if (root_block == -1) {
        printf("Error: no se pudo asignar bloque para el root.\n");
        fclose(fp);
        return;
    }
    // 3. Crear y escribir inodo raíz
    Inode root = {0};
    root.used = 1;
    root.is_directory = 1;
    root.size = 0;
    for (int i = 0; i < DIRECT_BLOCKS; i++) root.block_pointers[i] = -1;
    root.indirect1 = -1;
    root.indirect2 = -1;
    root.block_pointers[0] = root_block;
    root.mode = 0755;

    if (write_inode(fp, 0, &root) != 0) {
        perror("write_inode root");
        fclose(fp);
        return;
    }

    // 4. Escribir el resto de los inodos vacíos
    Inode empty = {0};
    for (int i = 1; i < NUM_INODES; i++) {
        write_inode(fp, i, &empty);
    }

    // 5. Rellenar los bloques de datos con ceros
    DataBlock zero = {0};
    int inode_table_blocks = (NUM_INODES * sizeof(Inode) + BLOCK_SIZE - 1) / BLOCK_SIZE;
    int reserved_blocks = SUPERBLOCK_BLOCKS + inode_table_blocks;
    for (int i = 0; i < NUM_BLOCKS - reserved_blocks; i++) {
        fwrite(&zero, sizeof(DataBlock), 1, fp);
    }

    if (write_superblock(fp, &sb) != 0) {
        perror("write_superblock");
        fclose(fp);
        return;
    }
```

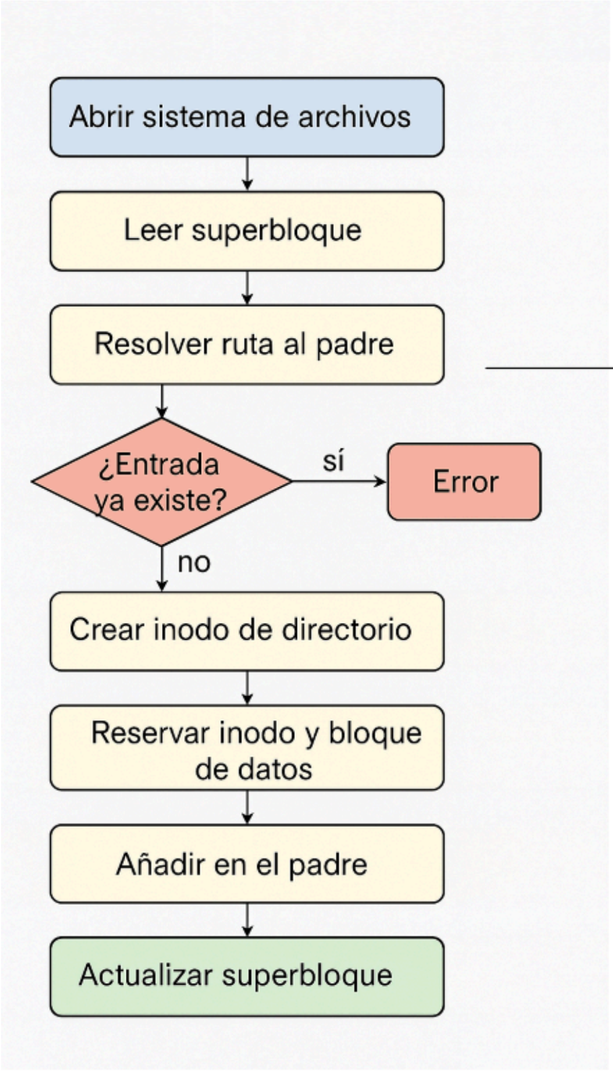
OPERACIONES PRINCIPALES

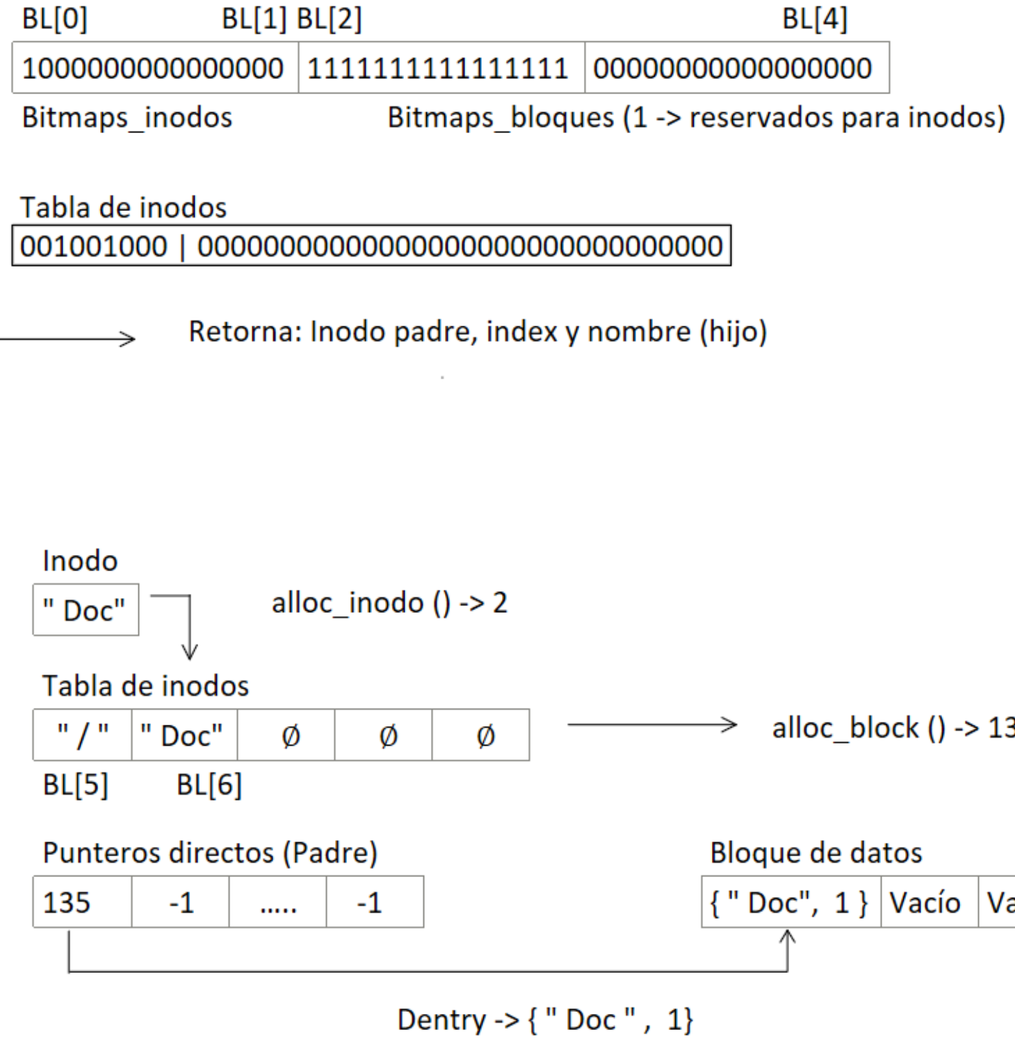
Las funciones implementadas permiten gestionar tanto directorios como archivos regulares mediante el uso de inodos y entradas de directorio (DirEntry).
Cada inodo almacena atributos clave, como el tipo de archivo, permisos, tamaño, marcas de tiempo (timestamps) y punteros a bloques de datos.
Gracias a la estructura jerárquica del sistema de archivos y a la asociación entre inodos y DirEntry, es posible realizar las siguientes operaciones fundamentales:

- Crear archivos y directorios
- Leer y escribir en archivos
- Renombrar archivos y directorios
- Eliminar archivos y directorios



CREACIÓN DE UN DIRECTORIO





AUTOMATIZACIÓN Y PRUEBAS

Se ha desarrollado un script en Bash que automatiza una batería de pruebas sobre el punto de montaje FUSE, incluyendo:

- Creación, lectura, renombrado y eliminación de archivos y directorios
- Validación de timestamps (atime, mtime, ctime)
- Cambios de permisos (chmod) y restricciones de borrado

Makefile:

- make install → monta el sistema con FUSE
- make uninstall → desmonta el punto de montaje
- make all / run / clean → compila, ejecuta y limpia el entorno

```
OK: ctime tras renombrado
[14] Pruebas de chmod (cambio de
OK: chmod archivo 400
OK: chmod archivo 600
OK: chmod dir 111
*****
Pruebas: 28/28
Exitoso
*****
```

GRACIAS POR SU ATENCIÓN