



**USAC**  
**TRICENTENARIA**  
Universidad de San Carlos de Guatemala

# **PROYECTO 1**

# **MANUAL TÉCNICO**

## **Manejo e Implementación de Archivos**

Miguel Adrian Tubac Agustin  
202101927




# Descripción de la Arquitectura del Sistema

El Second Extended Filesystem (ext2) es un sistema de archivos utilizado en sistemas Linux que fue diseñado como una mejora del sistema de archivos ext (Extended Filesystem). Aunque ha sido reemplazado por sistemas más avanzados como ext3 y ext4, sigue siendo fundamental para comprender la arquitectura de los sistemas de archivos en Linux.

## **1. Superbloque**

El superbloque almacena la información general del sistema de archivos, incluyendo:

- Tamaño del sistema de archivos.
  - Número total de inodos y bloques.
  - Número de bloques libres.
  - Tamaño de los bloques.
  - Estado del sistema de archivos (montado, limpio, etc.).
  - Información sobre copias de seguridad de superbloques.
- 

## 2. Grupos de Bloques

Para mejorar la eficiencia, ext2 divide el sistema de archivos en grupos de bloques. Cada grupo tiene una copia del superbloque y estructuras de control para aumentar la tolerancia a fallos. Cada grupo de bloques contiene:

1. Superbloque (copia de seguridad en algunos grupos).
2. Mapa de bits de bloques (Block Bitmap): Indica qué bloques están libres u ocupados.
3. Mapa de bits de inodos (Inode Bitmap): Indica qué inodos están en uso.
4. Tabla de inodos (Inode Table): Contiene los inodos, que almacenan metadatos de archivos.
5. Bloques de datos: Contienen el contenido real de los archivos y directorios.

REPORTE BLOQUE 12	
b_name	b_inodo
.	8
..	6
carpetal	9
user	14

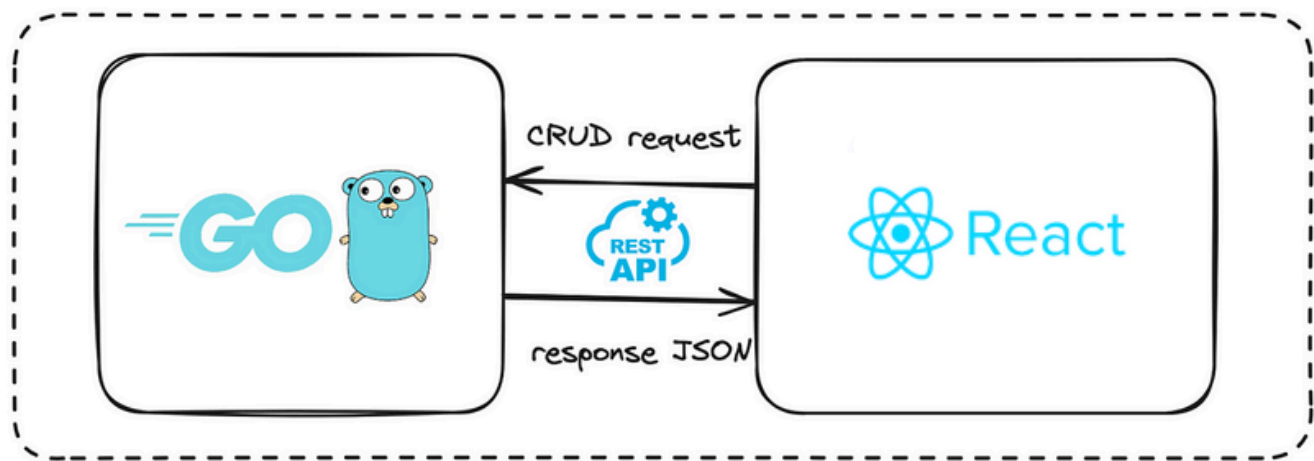
### 3. Inodos

Cada archivo o directorio en ext2 está representado por un inodo, que almacena:

- Permisos y atributos: Propietario, grupo, permisos de lectura/escritura/ejecución.
- Tamaño del archivo.
- Tiempos de acceso/modificación/creación.
- Número de enlaces duros al archivo.
- Punteros a bloques de datos:
  - Bloques directos (almacenan datos directamente).
  - Bloques indirectos (apuntan a otros bloques que contienen datos).
  - Bloques doble y triple indirectos (estructuras jerárquicas para archivos grandes).

REPORTE INODO 12	
i_uid	1
i_gid	1
i_size	0
i_atime	2025-03-31T20:25:04-06:00
i_ctime	2025-03-31T20:25:04-06:00
i_mtime	2025-03-31T20:25:04-06:00
i_type	0
i_perm	664
BLOQUES DIRECTOS	
1	16
2	-1
3	-1
4	-1
5	-1
6	-1

## Conexión entre los módulos frontend y backend



La comunicación entre el frontend (React) y el backend (golang) sigue una arquitectura cliente-servidor, donde React actúa como el cliente que solicita datos al servidor, y el backend en go maneja esas solicitudes y responde con la información necesaria.

# Explicación de las Estructuras de Datos

## Master Boot Record (MBR)

- Ubicado en el primer sector del disco (sector 0).
- Contiene la tabla de particiones, que define cómo está dividido el disco.
- No es exclusivo de ext2, pero es crucial para sistemas que usan este sistema de archivos.

```
You, el mes pasado | 1 author (You)
type MBR struct {
    Mbr_size           int32
    Mbr_creation_date  float32
    Mbr_disk_signature int32
    Mbr_disk_fit       [1]byte
    Mbr_partitions     [4]PARTITION
}
```

Este es el struc que en donde se definen las estructuras que definen los datos del MBR. Este se serializa en el disco .mia, al momento de mostrar un MBR se muestra de la siguiente manera:

REPORTE MBR	
mbr_tamano	13631488
mrb_fecha_creacion	2025-03-31 20:14:24 -0600 CST
mbr_disk_signature	379921525
PARTICIÓN 1	
part_status	1
part_type	P
part_fit	W
part_start	153
part_size	4194304
part_name	Part31

## Extended Boot Record (EBR)

- Presente en discos con particiones extendidas.
- Define la estructura de particiones lógicas dentro de una partición extendida.
- No es parte directa de ext2, pero es relevante cuando el sistema de archivos está en una partición lógica.

```
type EBR struct {  
    Ebr_mount [1]byte  
    Ebr_fit   [1]byte  
    Ebr_start int32  
    Ebr_size  int32  
    Ebr_next  int32  
    Ebr_name  [16]byte  
}
```

Este es el struc que en donde se definen las estructuras que definen los datos del EBR. Este se serializa en el disco .mia, al momento de mostrar un EBR se muestra de la siguiente manera:

PARTICIÓN LOGICA 1	
part_mount	N
part_fit	B
part_start	153
part_size	1048576
part_next	1048759
part_name	Part52
PARTICIÓN LOGICA 2	
part_mount	N
part_fit	B
part_start	1048759
part_size	1048576
part_next	2097365
part_name	Part54
PARTICIÓN LOGICA 3	

## Inodos

- Son estructuras de metadatos que almacenan información sobre archivos y directorios.
- Cada archivo o directorio tiene un inodo asociado.
- Contienen:
  - Permisos y propietario del archivo.
  - Tamaño del archivo.
  - Tiempos de acceso, modificación y creación.
  - Punteros a bloques de datos donde se almacena el contenido del archivo.

```
type Inode struct {  
    I_uid    int32  
    I_gid    int32  
    I_size   int32  
    I_atime  float32  
    I_ctime  float32  
    I_mtime  float32  
    I_block  [15]int32  
    I_type   [1]byte  
    I_perm   [3]byte  
    // Total: 88 bytes  
}
```



## Bloques de Datos

- Son las unidades básicas de almacenamiento donde se guardan los datos reales de los archivos.
- Existen diferentes tipos de bloques:
  - Bloques directos: Apuntan directamente a datos del archivo.
  - Bloques indirectos: Apuntan a otros bloques que contienen referencias a los datos.
  - Bloques doble y triple indirectos: Usados para archivos grandes, apuntan a bloques que a su vez contienen más referencias a datos.

```
type FolderBlock struct {  
    B_content [4]FolderContent  
    // Total: 64 bytes  
}  
  
You, hace 4 semanas | 1 author (You)  
type FolderContent struct {  
    B_name [12]byte  
    B_inodo int32  
    // Total: 16 bytes  
}
```

En los bloques de datos es en donde se definen las estructuras de los bloques. Este se serializa en el disco .mia, en el área de los bloques.



# Descripción de los Comandos Implementados

## MKDISK

Este comando crea un archivo binario que simulará un disco, estos archivos binarios tendrán la extensión .mia y su contenido al inicio es de 0 binarios.

Parámetros:

- size** : indicará el tamaño del disco a crear
- fit** : Indicará el ajuste que utilizará el disco
- unit** : indicará las unidades para el parámetro size
- path** : ruta en el que se creará el archivo

```
#Crearé un disco de 10 Mb ya que no hay parámetro unit  
mkdisk -size=10 -path="/home/mis discos/Disco4.mia"
```

## RMDISK

Este parámetro elimina un archivo que representa a un disco duro.

Parámetros:

- path** : ruta en el que se eliminará el archivo

```
#Elimina Disco4.mia  
rmdisk -path="/home/mis discos/Disco4.mia"
```





## FDISK

Este comando se encarga de la administración de particiones en el archivo que representa al disco duro virtual

Parámetros:

- size** : indicará el tamaño de la partición
- unit** : indicará las unidades para el parámetro size
- path** : ruta en el que se encuentra el disco
- type** : Indicará que tipo de partición se creará
- fit** : Indicará el ajuste que utilizará
- name** : Indicará el nombre de la partición


```
#Crea una partición lógica con el mejor ajuste, llamada Partición  
3 de 1 Mb en el Disco3  
fdisk -size=1 -type=L -unit=M -fit=BF-path="/mis  
discos/Disco3.mia" -name="Particion3"
```

## MOUNT

Este comando montará una partición del disco en el sistema

- path** : ruta en la que se encuentra el disco
- name** : Indica el nombre de la partición

```
#Monta las particiones de los siguientes disco *canet = 202401234  
mount -path=/home/Disco2.mia -name=Part2 #id=341A
```





## MOUNTED

Este comando mostrará todas las particiones montadas en memoria.

```
#Muestra todas las particiones montadas en el sistema (en memoria)
# Por ejemplo:
# 341A, 342A, 341B, 341C
mounted
```

## MKFS

Este comando realiza un formateo completo de la partición como ext2

**-id** : Indicará el id que se generó

**-type** : Indicará que tipo de formateo se realizará


```
#Realiza un formateo completo de la partición en el id 341A en
ext2
mkfs -type=full -id=341A
```

## CAT

Este comando permitirá mostrar el contenido del archivo

**-filen** : ficheros que se enlazar

```
#Lee el archivo a.txt
cat -file1=/home/user/docs/a.txt
```





## LOGIN

Este comando se utiliza para iniciar sesión en el sistema

Parámetros:

- user** : Especifica el nombre del usuario
- pass** : Indicará la contraseña del usuario
- id** : Indicará el id de la partición montada

```
#Se loguea en el sistema como usuario root  
login -user=root -pass=123 -id=062A
```

## LOGOUT

Este comando se utiliza para cerrar sesión

```
#Termina la sesión del usuario  
Logout
```


## MKGRP

Este comando creará un grupo para los usuarios de la partición y se guardará en el archivo users.txt

Parámetros:

- name** : Indicará el nombre que tendrá el grupo

```
#Crea el grupo usuarios en la partición de la sesión actual  
mkgrp -name=usuarios
```





## RMGRP

Este comando eliminará un grupo para los usuarios de la partición

Parámetros:

**-name** : Indicará el nombre del grupo a eliminar

```
#Elimina el grupo de usuarios en la partición de la sesión actual  
rmgrp -name=usuarios
```

## MKUSR

Este comando crea un usuario en la partición

Parámetros:

**-user** : Indicará el nombre del usuario a crear

**-pass** : Indicará la contraseña del usuario

**-grp** : Indicará el grupo al que pertenece

```
#Crea usuario user1 en el grupo 'usuarios'  
mkusr -user=user1 -pass=usuario -grp=usuarios
```


## RMUSR

Este comando elimina un usuario en la partición

Parámetros:

**-user** : Indicará el nombre del usuario a eliminar

```
#Elimina el usuario user1  
rmusr -user=user1
```





## CHGRP

Cambiará el grupo al que pertenece el usuario

Parámetros:

- user** : Especifica el nombre del usuario
- grp** : Contendrá el nombre del nuevo grupo

```
#Cambia el grupo del user2  
chgrp -user=user2 -grp=grupo1
```

## MKFILE

Este comando permitirá crear un archivo

Parámetros:

- path** : ruta del archivo que se creará
- r** : deben crearse las carpetas padres
- size** : tamaño en bytes del archivo
- cont** : Indicará un archivo en el disco

```
#Crea el archivo a.txt  
#Si no existen las carpetas home user o docs se crean  
#El tamaño del archivo es de 15 bytes  
#El contenido sería:  
#012345678901234  
mkfile -size=15 -path=/home/user/docs/a.txt -r
```



## MKDIR

Este comando es similar a mkfile, pero no crea carpetas

Parámetros:

**-path** : ruta de la carpeta que se creará

**-p** : deben crearse las carpetas padres

```
#Crea la carpeta usac
```

```
#Si no existen las carpetas home user o docs se crean
```

```
mkdir -p -path=/home/user/docs/usac
```

## REP

Recibirá el nombre del reporte que se desea y lo generará con graphviz en una carpeta existente

Parámetros:

**-name** : Nombre del reporte a generar. Tendrá los siguientes

valores: mbr, disk, inode, block, bm\_inode, bm\_block, tree, sb, file, ls

**-path** : carpeta y el nombre que tendrá el reporte

**-id** : Indica el id de la partición que se utilizará

**-path\_file\_ls** : nombre del archivo o carpeta del que se mostrará

```
#MBR y EBR Disco1.dsk
```

```
rep -id=A118 -path=/home/user/reports/reporte1.jpg -name=mbr
```