UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LABORATORIO DE MANEJO E IMPLEMETANCIÓN DE ARCHIVOS
SECCIÓN B
ABDIEL FERNANDO JOSÉ OTZOY OTZÍN
CARNET: 202300350

Guatemala, 31 de marzo del 2025

Documentación Técnica del Sistema de Archivos ext2

1. Introducción

El presente documento describe el diseño e implementación de un sistema de archivos basado en **ext2 (Second Extended Filesystem)**, desarrollado como parte de un proyecto académico/práctico. Este sistema de archivos simula las estructuras fundamentales de **ext2**, incluyendo:

- **Superbloque** (SuperBlock)
- **Inodos** (Inodes)
- **Bloques de datos** (Data Blocks)
- **Tablas de asignación** (Bitmaps)

El sistema permite operaciones básicas como:
✅ Creación y eliminación de archivos y directorios
✅ Manejo de permisos y usuarios
✅ Montaje y desmontaje de discos virtuales
✅ Generación de reportes estructurados

El proyecto está desarrollado en **Go (Golang)** debido a su eficiencia en manejo de sistemas de bajo nivel y concurrencia, junto con un **servidor API REST** (usando **Fiber**) para interactuar con el sistema de archivos desde un frontend o terminal.

---

2. Requerimientos Básicos

2.1 Requerimientos de Software

- **Lenguaje de programación**: Go (v1.20+)
- **Framework web**: Fiber (v2.0+)
- **Gestión de dependencias**: Go Modules
- **Interfaz de usuario**: Terminal o frontend web (React/HTML)

2.2 Requerimientos de Hardware

- **Sistema operativo**: Linux/Windows/macOS (multi-plataforma)
- **Memoria RAM**: Mínimo 2GB (recomendado 4GB+)
- **Almacenamiento**: Suficiente para discos virtuales (ej. 100MB+ por disco)

3. Objetivos

3.1 Objetivo General

Desarrollar un sistema de archivos **ext2 funcional** que permita gestionar discos virtuales, archivos y directorios mediante una **API REST**,
garantizando **consistencia**, **eficiencia** y **seguridad básica**.

3.2 Objetivos Específicos

- 🔹 **Implementar estructuras de datos clave** (Superbloque, Inodos, Bloques).
- 🔹 **Simular el manejo de particiones** (lectura/escritura en discos virtuales).
- 🔹 **Procesar comandos mediante análisis léxico/sintáctico** (CLI o API).
- 🔹 **Generar reportes en formato gráfico** (Diagramas con Graphviz).
- 🔹 **Asegurar concurrencia** (evitar corrupción de datos en operaciones simultáneas).

4. Funcionalidades Principales

A continuación, se detallan los comandos disponibles en el sistema de archivos **ext2**, su funcionamiento y ejemplos de uso:

4.1 Gestión de Discos

mkdisk - Crear un disco virtual

Descripción:
Crea un archivo binario que simula un disco duro con un tamaño específico, permitiendo configurar parámetros como el nombre, tamaño y unidad de medida.

Ejemplo:

```
mkdisk -size=50 -unit=MB -path=/home/user/disco.dsk
```

📌 **Parámetros**:

- -size: Tamaño del disco (obligatorio).
- -unit: Unidad (MB o KB, por defecto MB).
- -path: Ruta donde se guardará el disco.

---

rmdisk - Eliminar un disco virtual

Descripción:
Elimina un disco creado con mkdisk.

Ejemplo:

rmdisk -path=/home/user/disco.dsk

📌 **Parámetros**:

- -path: Ruta del disco a eliminar (obligatorio).

---

fdisk - Administrar particiones

Descripción:
Crea, elimina o modifica particiones en un disco.

Ejemplo:

fdisk -type=P -size=20 -unit=MB -path=/home/user/disco.dsk -name=part1

📌 **Parámetros**:

- -type: Tipo de partición (P primaria, E extendida, L lógica).
- -size: Tamaño de la partición.
- -name: Nombre de la partición (obligatorio).

4.2 Operaciones con Archivos y Directorios

mkdir - Crear directorio

Descripción:
Crea un directorio en una ruta específica, con permisos personalizables.

Ejemplo:

mkdir -path=/mnt/disk1/docs -p

📌 **Parámetros**:

- -path: Ruta del directorio.
- -p: Crea directorios padres si no existen.

---

mkfile - Crear archivo

Descripción:
Genera un archivo con contenido opcional.

Ejemplo:

```
mkfile -path=/mnt/disk1/docs/readme.txt -cont="Bienvenido al sistema"
```

📌 **Parámetros**:

- -path: Ruta del archivo.
- -cont: Contenido del archivo.

---

cat - Mostrar contenido de archivo

Descripción:
Muestra el contenido de un archivo en la terminal.

Ejemplo:

```
cat -path=/mnt/disk1/docs/readme.txt
```

4.3 Administración de Usuarios y Grupos

mkusr - Crear usuario

Descripción:
Registra un nuevo usuario en el sistema.

Ejemplo:

```
mkusr -user=admin -pass=123 -grp=root
```

📌 **Parámetros**:

- -user: Nombre de usuario.
- -pass: Contraseña.
- -grp: Grupo asignado.

---

login - Iniciar sesión

Descripción:
Autentica a un usuario en el sistema.

Ejemplo:

```
login -user=admin -pass=123
```

4.4 Reportes y Montaje

mount - Montar partición

Descripción:
Monta una partición para su uso.

Ejemplo:

```
mount -path=/home/user/disco.dsk -name=part1
```

---

rep - Generar reporte

Descripción:
Crea un reporte gráfico de la estructura del sistema.

Ejemplo:

```
rep -name=tree -path=/mnt/disk1 -output=/home/user/report.png
```

📌 **Parámetros**:

- -name: Tipo de reporte (tree, inodes, blocks).
- -output: Ruta de salida del reporte.

4.5 Ejemplo de Flujo de Uso

1. **Crear disco y partición**:

   ```
   mkdisk -size=100 -unit=MB -path=/home/user/mydisk.dsk
   fdisk -type=P -size=50 -name=system -path=/home/user/mydisk.dsk
   ```

2. **Montar y formatear**:

   ```
   mount -path=/home/user/mydisk.dsk -name=system
   mkfs -id=1 -type=full
   ```

3. **Crear estructura básica**:

   ```
   mkdir -path=/mnt/system/docs
   mkfile -path=/mnt/system/docs/hello.txt -cont="Hola, mundo!"
   ```

4. **Generar reporte**:

   rep -name=tree -path=/mnt/system -output=/home/user/tree.png

---

5. Conclusión

Estos comandos cubren las operaciones esenciales de un sistema de archivos **ext2**, desde la gestión de discos hasta la administración de usuarios. Cada función está diseñada para ser **intuitiva** y **eficiente**, con parámetros claros que facilitan su uso en entornos reales.

🔹 **Nota**: Todos los comandos devuelven mensajes de éxito/error en formato JSON cuando se usan mediante la API REST.

4.4 Reportes y Visualización

- Genera reportes en formato PNG (estructura de directorios, inodos, bloques).
- **Graphviz**: Usado para visualizar la estructura del sistema de archivos.

4.5 API REST

- **Endpoints** para ejecutar comandos desde un frontend.
- **Manejo de CORS** para comunicación con aplicaciones web.
- **Respuestas estructuradas** en JSON.

5. Conclusiones

El proyecto implementa con éxito un **sistema de archivos ext2 funcional**, cumpliendo con los objetivos planteados. Entre los logros destacan:

✅ **Estructuras de datos eficientes**:

- Uso de **inodos**, **bloques de datos** y **bitmaps** para gestión de almacenamiento.
- Serialización correcta en discos virtuales.

✅ **Interfaz flexible**:

- CLI para depuración y API REST para integración con frontends.

✅ **Extensibilidad**:

- Fácil incorporación de nuevas funcionalidades (ej. soporte para enlaces simbólicos).

5.1 Trabajo Futuro

- 🔸 **Soporte para journaling** (ext3/ext4).
- 🔸 **Mejorar permisos avanzados** (ACLs).
- 🔸 **Optimizar manejo de memoria** para discos grandes.

Nota Final:
Este sistema demuestra los principios fundamentales de un sistema de archivos, siendo útil tanto para **fines educativos** como para **prototipado de soluciones más complejas**.

📌 **Repositorio del proyecto**:
https://github.com/AbdielOtzoy/MIA_1S2025_P1_202300350

- 🔹 **Autor**: Abdiel Fernando José Otzoy Otzín
- 🔹 **Fecha**: 31 de marzo del 2025