Manual Técnico del Sistema de Archivos EXT3 Simulado

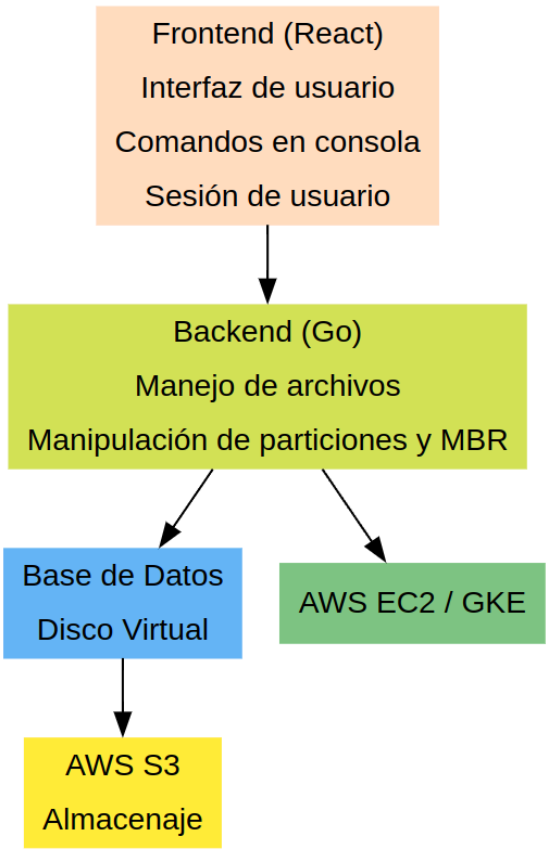
1. Descripción de la Arquitectura del Sistema

Arquitectura General

La arquitectura del sistema se basa en una estructura cliente-servidor, donde el **frontend** se comunica con el **backend** mediante una API RESTful. El frontend es responsable de mostrar la interfaz de usuario (UI) y de enviar comandos para interactuar con el sistema de archivos. El backend, implementado en Go, gestiona las operaciones del sistema de archivos, como la manipulación de particiones, archivos y bloques, así como el manejo de sesiones y comandos de usuario.

Diagrama de Arquitectura

A continuación se muestra el diagrama que ilustra la conexión entre los componentes del sistema:



Frontend (React)

El frontend está desarrollado utilizando **React.js** y **Vite**. Es responsable de gestionar la interacción del usuario con la aplicación web y de enviar comandos al backend para interactuar con el sistema de archivos EXT3 simulado. Los componentes principales incluyen:

- **Consola de Comandos**: Permite al usuario ejecutar comandos sobre el sistema de archivos y ver la salida.
- **Explorador de Archivos**: Proporciona una vista visual de las particiones y archivos disponibles.
- **Formulario de Login**: Gestiona la autenticación de usuarios.

Comunicación con el Backend

El frontend se comunica con el backend a través de solicitudes HTTP usando la API RESTful. Para cada operación (como crear, eliminar o modificar archivos y particiones), el frontend envía una solicitud `POST` o `GET` al backend, que procesa la operación y responde con los resultados.

Backend (Go)

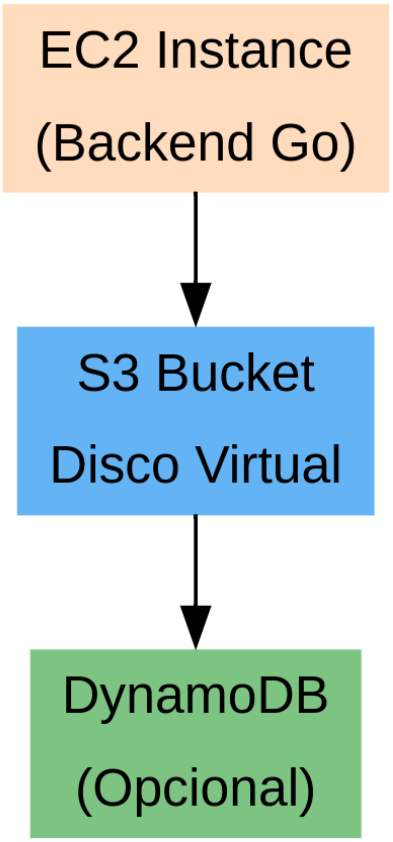
El backend, implementado en Go, gestiona las operaciones del sistema de archivos. Utiliza un modelo de **sesión activa** que se mantiene durante la interacción del usuario con el sistema. Los componentes clave del backend son:

- **Manejo de Particiones**: Se encarga de la manipulación de las particiones (creación, eliminación, montado, desmontado).
- **Sistema de Archivos**: Gestiona el acceso y manipulación de los archivos y directorios en las particiones.
- **Autenticación de Usuario**: Gestiona el login y logout de los usuarios, asegurando que las operaciones del sistema de archivos solo se realicen con una sesión activa.
- **Journaling**: Lleva un registro de las transacciones realizadas en el sistema de archivos para permitir la recuperación en caso de fallos.

Despliegue en AWS

El sistema se despliega en AWS utilizando **EC2** para la ejecución del backend y **S3** para el almacenamiento de datos. La base de datos del sistema de archivos reside en un archivo binario `.mia` almacenado en un volumen persistente de EC2.

Diagrama de Despliegue



2. Explicación de las Estructuras de Datos

MBR (Master Boot Record)

El **MBR** es la primera estructura en el disco y se encarga de almacenar información crítica sobre las particiones del disco, como:

- **Particiones**: Información sobre la ubicación y tamaño de las particiones.
- **Identificador de partición**: Cada partición tiene un ID único.
- **Arranque**: Sector de arranque para la inicialización del sistema operativo.

En el contexto de este sistema, el MBR es esencial para la gestión de particiones, permitiendo identificar y organizar las particiones disponibles en el sistema.

EBR (Extended Boot Record)

El **EBR** es utilizado para particiones extendidas. En el caso de particiones extendidas, el EBR contiene información adicional sobre las particiones lógicas dentro de esa partición extendida. El EBR tiene:

- **Puntero al siguiente EBR**: En caso de que haya más particiones lógicas.
- **Información de la partición lógica**: Ubicación, tamaño y otras propiedades.

Inodos

Los **inodos** son estructuras que almacenan la metadata de los archivos y directorios:

- **Número de inodo**: Identificador único del inodo.
- **Permisos**: Propietarios, permisos de lectura, escritura, ejecución.
- **Tamaño del archivo**: Tamaño del archivo en bytes.
- **Punteros a bloques de datos**: Los bloques donde se almacena el contenido del archivo.

Bloques

Los **bloques** son las unidades de almacenamiento real de los datos. En este sistema se consideran tres tipos de bloques:

- **Bloques de archivos**: Contienen los datos de los archivos.
- **Bloques de directorios**: Contienen las referencias a los archivos en un directorio.
- **Bloques de contenido**: Almacenan los datos reales de los archivos y directorios.

Bitmap de Bloques

El **Bitmap de Bloques** es una estructura que mantiene un registro de los bloques ocupados y libres. Cada bit representa un bloque del disco. Un bit en `0` indica que el bloque está libre, mientras que un bit en `1` indica que el bloque está ocupado.

Superblock

El **Superblock** contiene la información crucial sobre el sistema de archivos, como:

- **Tamaño del sistema de archivos**.

- **Número de inodos**.
- **Número de bloques**.
- **Estado del sistema de archivos**.

3. Descripción de los Comandos Implementados

FDISK

El comando **FDISK** permite gestionar particiones en el sistema:

- **Crear Partición**: Se utiliza con el parámetro `-size` para definir el tamaño de la partición.
- **Eliminar Partición**: Permite eliminar particiones con `-delete`.
- **Agregar y Quitar Espacio**: Modifica el tamaño de las particiones con `-add` o `-resize`.

Ejemplo de uso:

```
fdisk -add=100 -unit=M -path=/home/Disco1.dk -name=Particion2
fdisk -delete=fast -name=Particion2 -path=/home/Disco1.dk
```

UNMOUNT

El comando **UNMOUNT** desmonta una partición del sistema, asegurando que no se realicen operaciones sobre ella.

Ejemplo de uso:

```
unmount -id=141A
```

MKFS (FS)

El comando **MKFS** formatea una partición, pudiendo especificar el tipo de sistema de archivos (EXT2 o EXT3).

Ejemplo de uso:

bash
Copiar
```
mkfs -id=141A -fs=3fs
```

REMOVE

El comando **REMOVE** elimina archivos o directorios. Verifica que el usuario tenga permisos de escritura antes de eliminar cualquier archivo.

Ejemplo de uso:

```
remove -path="/home/user/docs/a.txt"
```

EDIT

El comando **EDIT** permite modificar el contenido de un archivo.

Ejemplo de uso:

```
edit -path="/home/user/docs/a.txt" -contenido="/root/user/files/a.txt"
```

RENAME

El comando **RENAME** cambia el nombre de un archivo o directorio.

Ejemplo de uso:

```
rename -path="/home/user/docs/a.txt" -name="b1.txt"
```

COPY

El comando **COPY** realiza una copia de un archivo o directorio en otra ubicación.

Ejemplo de uso:

```
copy -path="/home/user/docs" -destino="/home/images"
```

MOVE

El comando **MOVE** mueve un archivo o directorio a una nueva ubicación.

Ejemplo de uso:

```
move -path="/home/user/docs/a.txt" -destino="/home/images"
```

FIND

El comando **FIND** permite buscar archivos o carpetas dentro de una ruta dada.

Ejemplo de uso:

find -path="/home" -name="*.txt"

CHOWN

El comando **CHOWN** cambia el propietario de un archivo o directorio.

Ejemplo de uso:

chown -path="/home/user/docs/a.txt" -usuario="newuser"

CHMOD

El comando **CHMOD** cambia los permisos de un archivo o directorio.

Ejemplo de uso:

chmod -path="/home/user/docs" -ugo=764

4. Manejo del Sistema de Archivos EXT3

Journaling en EXT3

El sistema de archivos EXT3 introduce el concepto de **journaling**, que actúa como un registro de las transacciones realizadas en el sistema de archivos. Cada operación, como la creación de archivos o la modificación de directorios, se registra en el **journal** antes de que los cambios se realicen en el sistema de archivos. En caso de fallo del sistema, el journaling permite recuperar el sistema a un estado consistente.

Estructura del Journal

La estructura del journal incluye:

- **Operación**: Tipo de operación (creación, modificación, eliminación).
- **Ruta**: Ruta del archivo o directorio afectado.
- **Contenido**: Datos modificados.

- **Fecha y Hora**: Momento en que se realizó la operación.

5. Pérdida y Recuperación del Sistema de Archivos EXT3

Simulación de Pérdida

La simulación de pérdida de datos se realiza al eliminar información clave de los bloques del sistema, como:

- **Bitmap de Inodos**
- **Bitmap de Bloques**
- **Área de Inodos**
- **Área de Bloques**

Recuperación del Sistema de Archivos

La recuperación del sistema se basa en el **journaling** y el **superblock**. Al detectarse una inconsistencia, el sistema puede restaurar el estado anterior utilizando la información registrada en el journal.

Comando de recuperación:

```
recovery -id=061Disco1
```

Journaling

El comando **journaling** muestra las transacciones registradas en el journal, brindando una herramienta para monitorear las operaciones realizadas.

Comando de journaling:

```
journaling -id=061Disco1
```

Conclusión

Este manual técnico proporciona una descripción detallada del sistema de archivos **EXT3** simulado, incluyendo su arquitectura, estructuras de datos, comandos implementados y manejo de recuperación. El sistema está diseñado para ofrecer una experiencia de

administración de archivos robusta y eficiente, con un enfoque en la estabilidad y la recuperación ante fallos mediante el uso de **journaling** y la estructura de bloques.