Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Manejo e Implementación de Archivos
Segundo Semestre 2024

Catedráticos: Ing. Álvaro Díaz, Ing. Oscar Paz, Ing. William Escobar, Ing. Jurgen Ramiez
Tutores académicos: Yonathan Hernandez, Kevin Garcia, Allen Román, Andres Pontaza

Proyecto 1

(Unidad 1 y 2)

Introducción

El Proyecto 1 tiene como objetivo desarrollar una aplicación web para la interacción y gestión de un sistema de archivos EXT2. Esta herramienta web moderna permitirá acceder y administrar el sistema de archivos desde cualquier lugar y en cualquier sistema operativo. En el backend, el sistema de archivos se gestionará en una distribución Linux, que atenderá todas las solicitudes provenientes del frontend. Dado que se trata del primer proyecto, la implementación se realizará de manera local.

Objetivos

- Aprender a administrar archivos y escribir estructuras en Go.
- Comprender el sistema de archivos EXT2.
- Aplicar el formateo rápido y completo en una partición.
- Crear una aplicación web de comandos.
- Aplicar la teoría de ajustes.
- Aplicar la teoría de particiones.
- Utilizar Graphviz para mostrar reportes.
- Restringir y administrar el acceso a los archivos y carpetas en ext2 por medio de usuarios.
- Administrar los usuarios y permisos por medio de grupo.

Índice

Arquitectura

El proyecto tendrá la siguiente arquitectura:

Frontend:

- Se recomienda utilizar un framework moderno para el desarrollo del frontend. Se sugiere especialmente Angular debido a su robustez, estructura modular y amplia comunidad de soporte. No obstante, la elección del framework queda a discreción del desarrollador, pudiendo optar por otras opciones como React, Vue.js, entre otros.

- El frontend será responsable de la interfaz de usuario, brindando una experiencia interactiva y dinámica. Deberá incluir componentes reutilizables, manejo eficiente del estado y comunicación fluida con el backend mediante APIs.

Backend:

- Es obligatorio utilizar el lenguaje Go (Golang) para el desarrollo del backend. Go es conocido por su eficiencia, concurrencia nativa y rendimiento, lo cual es ideal para construir aplicaciones robustas y escalables.

- El backend manejará la exposición de APIs RESTful para la comunicación con el frontend.

- La arquitectura del backend deberá seguir principios de diseño limpio y separación de responsabilidades según sea conveniente.

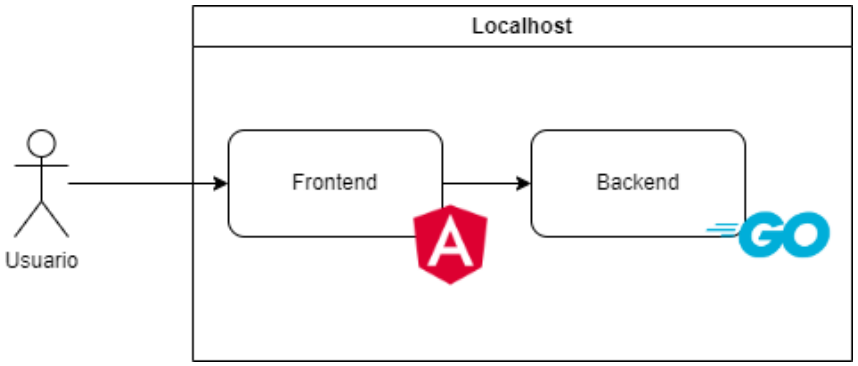


Figura 1 - Arquitectura del proyecto 1

Primer Parte (Frontend)

Para este apartado se solicita un frontend con una página home, que contenga:

- **Textarea de Entrada:** En este apartado se ingresan los comandos que el usuario desea ejecutar, permite el ingreso manual y también permite la carga de un archivo script.
    - **Botón de Carga de archivos:** Permite seleccionar el archivo script que desea ejecutar y permite insertar rápidamente en el textarea de entrada los comandos a ejecutar.
    - **Botón de Ejecutar:** Inicia la ejecución de todos los comandos que se encuentra en el textarea de entrada y muestra los mensajes del lado del servidor en el textarea de salida.
- **Textarea de Salida:** En este apartado se mostrarán todas las salidas de los comandos ejecutados del textarea entrada que retorna el backend luego de ejecutarlos .

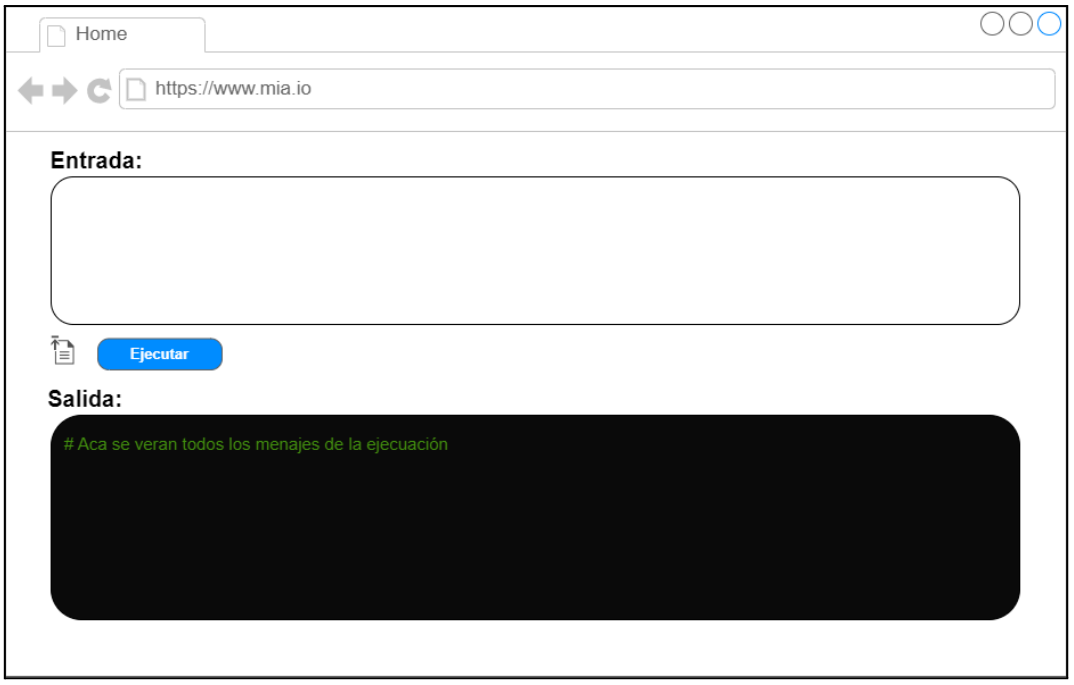


Figura 2 - Frontend del proyecto

Segunda Parte (Backend)

Del lado del backend se deberá crear un sistema de archivos **EXT2**, por tanto el manejo de discos no será en dispositivos reales si no serán simulados por medio de **archivos binarios** con extensión **.mia** donde se almacenarán las distintas estructuras para simular el funcionamiento del almacenamiento de un sistema de archivos y las distintas particiones que pudiera contener.

Discos

Se presenta la siguiente figura para mostrar de manera gráfica cómo es la escritura de las diferentes estructuras para entender el funcionamiento y la simulación del disco.

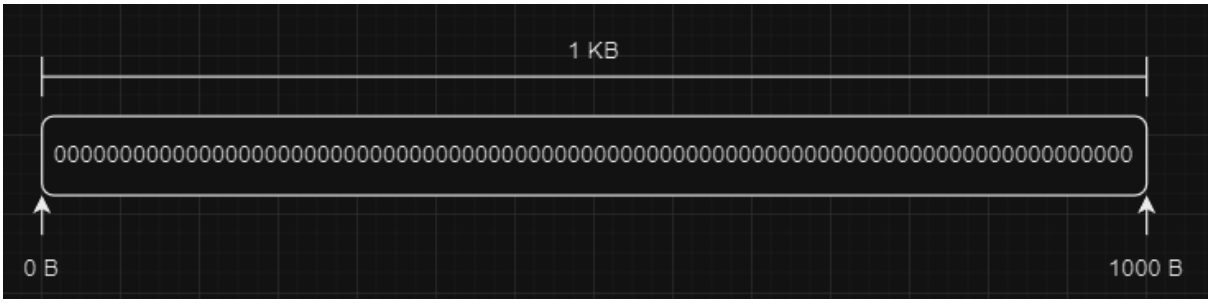


Figura 3 - Espacio del archivo

La creación y detalles del mismo estarán especificadas más adelante en los respectivos comandos, pero se aclara que al crear un disco deberá estar llenado con ceros binarios para representar que es espacio disponible.

Observaciones:

1. El tamaño del archivo NO deberá cambiar de tamaño una vez creado y designado un almacenamiento definido.

2. Para crear el archivo del disco se recomienda utilizar un **char[1024]** como buffer para crear el archivo, si se utiliza un char[1] normalmente se tarda demasiado al momento de crear el archivo.

Estructura para discos

Master Boot Record (MBR)

Cuando se crea un nuevo disco este debe contener un MBR ya que provee información del sistema de archivos y de las particiones, este deberá estar en el primer sector del disco. Tendrá los siguientes valores:

Nombre	Tipo	Descripción
mbr_tamano	int	Tamaño total del disco en bytes
mbr_fecha_creacion	time	Fecha y hora de creación del disco
mbr_dsk_signature	int	Número random, que identifica de forma única a cada disco
dsk_fit	char	Tipo de ajuste de la partición. Tendrá los valores **B** (Best), **F** (First) o **W** (worst)
mbr_partitions	partition[4]	Estructura con información de las 4 particiones

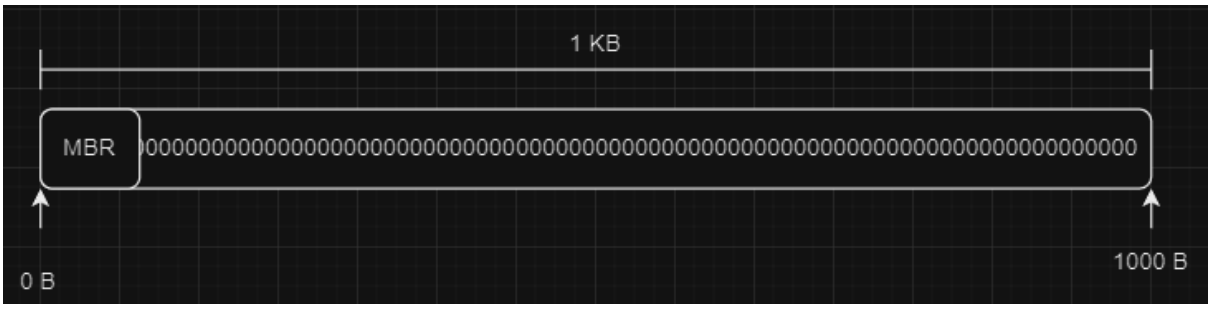


Figura 4 - Espacio del archivo

El MBR se crea en el primer sector del disco, es decir se escribirá al inicio de dicho disco conteniendo la información del sistema de archivos.

Partition

Una partición es una división lógica de un disco que los sistemas de archivos tratan como una unidad separada. Tendrá los siguientes valores:

Nombre	Tipo	Descripción
part_status	char	Indica si la partición está montada o no
part_type	char	Indica el tipo de partición, primaria o extendida. Tendrá los valores **P** o **E**
part_fit	char	Tipo de ajuste de la partición. Tendrá los valores **B** (Best), **F** (First) o **W** (worst)
part_start	int	Indica en qué byte del disco inicia la partición
part_s	int	Contiene el tamaño total de la partición en bytes
part_name	char[16]	Nombre de la partición
part_correlative	int	Indica el correlativo de la partición.
part_id	char[4]	Indica el ID de la partición generada al montar esta partición, esto se explicará más adelante

- **Particiones Primarias:** una partición primaria puede ser usada para iniciar un sistema operativo y contener distintos archivos no relacionados directamente con el sistema operativo.
- **Particiones Extendidas:** una partición extendida es usada para contener unidades lógicas, estas particiones son manejadas por un EBR, al crear esta partición se creará el primer EBR.
- **Unidades Lógicas:** estas unidades contienen archivos no relacionados con el sistema operativo, como podrían ser datos, audio, video y entre otros. Estas unidades lógicas son manejadas por un EBR.

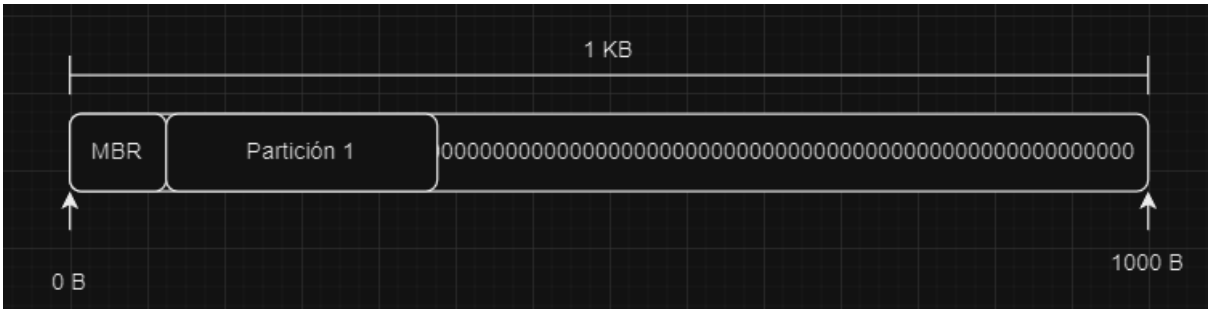


Figura 5 - Espacio del archivo

Al escribir una partición primaria se **reservará** dicho espacio para escribir **otras** estructuras dentro de dicho espacio para simular el funcionamiento de la partición, se aclara que el **objeto partición sólo se actualizará en el MBR.**

Observaciones:

1. El manejo de archivos se utilizará a nivel de particiones primarias.
2. Las particiones lógicas se utilizaran para el comando mount y reportes.

Extended Boot Record (EBR)

El EBR es un descriptor de una unidad lógica ya que es contiene la información y datos de la misma y apunta hacia el espació donde se escribirá el siguiente EBR, el EBR se puede ver como una clase de lista enlazada. Tendrá los siguientes valores:

Nombre	Tipo	Descripción
part_mount	char	Indica si la partición está montada o no
part_fit	char	Tipo de ajuste de la partición. Tendrá los valores **B** (Best), **F** (First) o **W** (worst)
part_start	int	Indica en qué byte del disco inicia la partición
part_s	int	Contiene el tamaño total de la partición en bytes.
part_next	int	Byte en el que está el próximo EBR. -1 si no hay siguiente
part_name	char[16]	Nombre de la partición

Figura 6 - Espacio del archivo

En este caso se toma en cuenta que la partición 2 es una partición extendida e igualmente como el caso anterior se reserva el espacio para la escritura de las unidades lógicas con sus respectivos EBR.

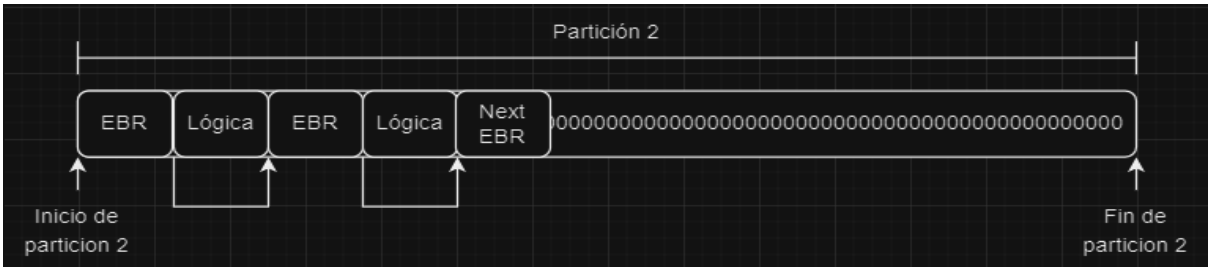


Figura 7 - Espacio de la partición 2

En este caso se simula 2 particiones lógicas y el último EBR sabe donde se escribirá el siguiente EBR, la parte nombrada lógica es espacio reservado para otras estructuras.

Sistema de Archivos Ext2

EXT2

Para el caso del sistema de archivos EXT2, se deberán implementar las estructuras como se especifican a continuación. La estructura en bloques es la siguiente:

El número de bloques será el triple que el número de inodos. El número de inodos y bloques a crear se puede calcular despejando n de la primera ecuación y aplicando la función floor al resultado:

- **tamaño_particion = sizeOf(superblock) + n + 3 * n + n * sizeOf(inodos) + 3 * n * sizeOf(block)**
- **numero_estructuras = floor(n)**

Observaciones:

1. sizeof es el tamaño de los Structs.
2. En el Bitmap de Bloques y Bloques se multiplica por tres debido a que existen tres tipos de bloque que son: bloques carpetas, bloques archivos y bloques de contenido.

Figura 8 - Espacio de la partición 1

En este caso se tomó la estructura de bloques del sistema EXT2 que en este ejemplo se toma como si se hubiera asignado a la partición 1 con dicho sistema, **se aclara** que los bloques de Inodos y Bloques solo son reserva de espacio, ya que dentro de ese espacio se escribirán múltiples estructuras contiguas según corresponda como por ejemplo la siguiente figura.

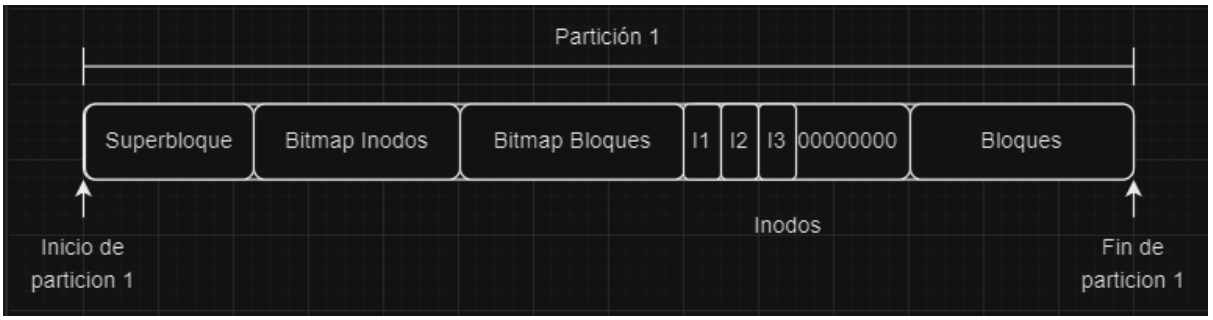


Figura 9 - Espacio de la partición 1

En la figura se muestra como dentro del espacio reservado de Inodos se escribieron múltiples estructuras de inodos los cuales se entrar a detalle a continuación de dichas estructuras.

Estructuras para carpetas y archivos

Súper Bloque

Esta estructura contiene la información sobre el sistema de archivos al que pertenece (en este caso EXT2), de esta estructura solo se escribe 1 vez en la partición según el espacio designado. Tendrá los siguientes valores:

NOMBRE	TIPO	DESCRIPCIÓN
s_filesystem_type	int	Guarda el número que identifica el sistema de archivos utilizado
s_inodes_count	int	Guarda el número total de inodos
s_blocks_count	int	Guarda el número total de bloques
s_free_blocks_count	int	Contiene el número de bloques libres
s_free_inodes_count	int	Contiene el número de inodos libres
s_mtime	time	Última fecha en el que el sistema fue montado
s_umtime	time	Última fecha en que el sistema fue desmontado
s_mnt_count	int	Indica cuantas veces se ha montado el sistema
s_magic	int	Valor que identifica al sistema de archivos, tendrá el valor 0xEF53
s_inode_s	int	Tamaño del inodo
s_block_s	int	Tamaño del bloque
s_firts_ino	int	Primer inodo libre (dirección del inodo)
s_first_blo	int	Primer bloque libre (dirección del inodo)
s_bm_inode_start	int	Guardará el inicio del bitmap de inodos
s_bm_block_start	int	Guardará el inicio del bitmap de bloques
s_inode_start	int	Guardará el inicio de la tabla de inodos
s_block_start	int	Guardará el inicio de la tabla de bloques

Es similar al MBR solo que se enfoca en las particiones, guarda las posiciones de la partición para permitir movimiento en ella.

Esta información no cambia de tamaño y se debe actualizar, según se vayan realizando las operaciones en el sistema de archivos. Por ejemplo, al usar un X comando, debe actualizar los valores que puedan ser modificados según corresponda.

Inodos (index node)

Esta estructura contiene las características e información sobre un fichero usado por una carpeta o archivo. Tendrá los siguientes valores:

NOMBRE	TIPO	DESCRIPCIÓN
i_uid	int	UID del usuario propietario del archivo o carpeta
i_gid	int	GID del grupo al que pertenece el archivo o carpeta.
i_s	int	Tamaño del archivo en bytes
i_atime	time	Última fecha en que se leyó el inodo sin modificarlo
i_ctime	time	Fecha en la que se creó el inodo
i_mtime	time	Última fecha en la que se modifica el inodo
i_block	int	Array en los que los primeros 12 registros son bloques directos. El 13 será el número del bloque simple indirecto. El 14 será el número del bloque doble indirecto. El 15 será el número del bloque triple indirecto. ● Bloques directos (Pueden ser bloque carpeta o archivo) ● Bloques indirectos (Estos expanden la capacidad de un archivo, utilizan bloques apuntadores para tener más bloques indirectos) Si no son utilizados tendrá el valor: -1.
i_type	char	Indica si es archivo o carpeta. Tendrá los siguientes valores: 1 = Archivo 0 = Carpeta
i_perm	char[3]	Guardará los permisos del archivo o carpeta, Se trabajarán usando los permisos UGO (User Group Other) en su forma octal. Linux File Permission Cheatsheet

Observaciones:

1. El inodo guarda la información del archivo o carpeta, por cada archivo o carpeta debe existir un inodo.
2. Se debe iniciar del inodo hasta el bloque.

3. Este tendrá la información necesaria para el manejo de permisos a partir del usuario que creó el documento.
4. Contiene el tipo de bloque 1 o 0.
5. Si es un inodo carpeta puede apuntar en sus apuntadores directos otros bloques carpeta o un inodo si es un archivo.
6. Si es un inodo archivo puede apuntar en sus apuntadores directos un bloque de datos para expandir la capacidad del archivo.

Bloques

Estas estructuras son la unidad mínima de almacenamiento a nivel lógico. Estos bloques son un conjunto de sectores contiguos que componen la unidad de almacenamiento más pequeña de un disco, para este proyecto de procuró que todos los bloques tengan un tamaño de 64 bytes y los distintos bloques son:

Bloques de carpetas

Esta estructura guardará la información sobre el nombre de de los archivos que contiene y a que Inodo apuntan. Tendrá los siguientes valores:

NOMBRE	TIPO	DESCRIPCIÓN
b_content	content[4]	Array con el contenido de la carpeta

La estructura **b_content** tendrá los siguientes valores:

NOMBRE	TIPO	DESCRIPCIÓN
b_name	char[12]	Nombre de la carpeta o archivo
b_inodo	int	Apuntador hacia un inodo asociado al archivo o carpeta

En cada inodo de carpeta, en el primer apuntador directo, en los primeros dos registros se guardará el nombre de la carpeta y su padre.

Tamaño en bytes: 4 (estructuras b_content) * 12 (chars b_name) * 4 (int b_inodo) = 64

Observaciones:

1. El bloque carpeta puede guardar otras carpetas u otros archivos.

Bloques de Archivos

Esta estructura guardará la información sobre contenido de un archivo. Tendrá los siguientes valores:

NOMBRE	TIPO	DESCRIPCIÓN
b_content	char[64]	Array con el contenido del archivo

Tamaño en bytes: 64 (chars b_content).

Observaciones:

1. El bloque archivo guarda el contenido del archivo (64 caracteres).

Bloques de Apuntadores

Esta estructura guardará la información de los apuntadores indirectos (simples, dobles y triples). Tendrá los siguientes valores:

NOMBRE	TIPO	DESCRIPCIÓN
b_pointers	int[16]	Array con los apuntadores a bloques (de archivo o carpeta)

Tamaño en bytes: 16 (cantidad de int) * 4 (int) = 64

Observaciones:

- **Bloque de apuntadores** tiene 16 Posiciones
    - **Bloque Simple Indirecto:** Inodo → Bloque apuntadores → bloque de datos.
    - **Bloque Doble Indirecto:** Inodo → Bloque de apuntadores → Bloque de apuntadores → bloque de datos.
    - **Bloque Triple Indirecto:** Inodo → Bloque de apuntadores → Bloque de apuntadores → Bloque de apuntadores → bloque de datos.

Limitaciones de estructuras

La cantidad de estructuras máximas a generar serán detalladas más adelante, pero se aclara que se debe realizar mediante un cálculo basado en el tipo de sistema en este proyecto solo será EXT2 y el tamaño del disco.

Bitmap

Un Bitmap como su nombre lo indica es un mapa de bits, es decir es la representación binaria en la cual un bit o conjunto de bits corresponde a alguna parte de un objeto específico.

- **Bitmap Inodos:** indica el estado de los inodos, usando 0 como usable y 1 como ocupado.

- **Bitmap bloques:** indica el estado de los bloques (independientemente el tipo de bloque), usando 0 como usable y 1 como ocupado.

Ejemplo:

Si en el cálculo de estructuras resultó en tener 10 Inodos y 20 bloques, entonces tendremos 10 bits en el bitmap de inodos y 20 bits en el bitmap de bloques, donde la creación de un inodo supondrá el cambió de un bit según si correlativo en el bitmap de inodos y de la misma manera con el bitmap de bloques, estos bitmaps son cambiados en el espacio designado como en los ejemplos de las figuras anteriores.

Comandos del sistema de archivos

La aplicación web contará con dos áreas de texto. La primera se utilizará para la introducción de comandos o la carga de scripts, mientras que la segunda mostrará la salida resultante de la ejecución. La aplicación no distinguirá entre mayúsculas y minúsculas en los comandos ingresados.

Se han definido parámetros obligatorios y opcionales para los comandos. Si se utiliza un parámetro que no está especificado en la documentación proporcionada, se mostrará un mensaje de error indicando que el parámetro no es reconocido. Los parámetros deben estar separados por espacios en blanco. Si un valor contiene espacios en blanco, deberá ser encerrado entre comillas dobles ("X").

La aplicación sólo reconocerá los comandos y parámetros definidos en el presente enunciado. No se aceptarán variaciones ni versiones de semestres anteriores. Los parámetros de todos los comandos pueden ser ingresados en cualquier orden.

Administración de discos

Estos comandos están diseñados para crear archivos que simularán discos duros virtuales. Estos archivos virtuales podrán ser formateados posteriormente con los sistemas de archivos EXT2, permitiendo la simulación de operaciones típicas en un entorno de disco duro real.

Desde el momento en que se inicie la aplicación, los usuarios tendrán acceso a estos comandos, que estarán disponibles para facilitar la gestión de los discos virtuales. La creación y manipulación de estos discos virtuales proporcionará una forma práctica de experimentar con sistemas de archivos y operaciones relacionadas sin necesidad de hardware físico adicional.

A continuación se detallan los comandos disponibles:

1. MKDISK

Este comando creará un archivo binario que simulará un disco, estos archivos binarios tendrán la extensión .**mia** y su contenido al inicio será 0 binarios. Deberá ocupar físicamente el tamaño indicado por los parámetros, (no importa que el sistema operativo no muestre el tamaño exacto). Recibirá el nombre del archivo que simulará el disco duro y tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-size	**Obligatorio**	Este parámetro recibirá un número que indicará el tamaño del disco a crear. Debe ser positivo y mayor que cero, si no se mostrará un error.
-fit	**Opcional**	Indicará el ajuste que utilizará el disco para crear las particiones dentro del disco Podrá tener los siguientes valores: **BF**: Indicará el mejor ajuste (Best Fit) **FF**: Utilizará el primer ajuste (First Fit) **WF**: Utilizará el peor ajuste (Worst Fit) Ya que es opcional, se tomará el primer ajuste (FF) si no está especificado en el comando. Si se utiliza otro valor que no sea alguno de los anteriores mostrará un mensaje de error.
-unit	**Opcional**	Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro size. Podrá tener los siguientes valores: **K:** Indicará que se utilizarán Kilobytes (1024 bytes) **M:** Indicará que se utilizarán Megabytes (1024 * 1024 bytes) Este parámetro es opcional, si no se encuentra se creará un disco con tamaño en Megabytes. Si se utiliza otro valor debe mostrarse un mensaje de error.
-path	**Obligatorio**	Este parámetro será la ruta en el que se creará el archivo que representará el disco duro. Si las carpetas de la ruta no existen deberán crearse.

Ejemplos:

```
#Crea un disco de 3000 Kb en la carpeta home
mkdisk -Size=3000 -unit=K -path=/home/user/Disco1.mia
```

```
#No es necesario utilizar comillas para la ruta en este caso ya
que la ruta no tiene ningún espacio en blanco
mkdisk -path=/home/user/Disco2.mia -Unit=K -size=3000

#Se ponen comillas por la carpeta "mis discos" ya que tiene
espacios en blanco, se crea si no está no existe
mkdisk -size=5 -unit=M -path="/home/mis discos/Disco3.mia"

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

2. RMDISK

Este parámetro elimina un archivo que representa a un disco duro. Debe de mostrarse un mensaje solicitando la confirmación de eliminar el disco. Tendrá los siguientes parámetros

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	**Obligatorio**	Este parámetro será la ruta en el que se eliminará el archivo que representará el disco duro. Si el archivo no existe, debe mostrar un mensaje de error.

Ejemplo:

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

3. FDISK

Este comando se encarga de la administración de particiones en el archivo que representa al disco duro virtual. Permite a los usuarios crear, eliminar o modificar particiones dentro del archivo de disco duro.

En caso de que no se pueda realizar la operación solicitada sobre una partición, la aplicación deberá mostrar un mensaje de error detallado. El mensaje de error especificará claramente la razón por la cual la operación falló, como podría ser la falta de espacio disponible, restricciones en el número máximo de particiones permitidas, errores en los parámetros ingresados, o cualquier otra causa relevante. Esta retroalimentación detallada ayudará a los usuarios a entender y corregir los problemas para completar la operación de manera exitosa.

Tendrá los siguientes parámetros:

Parámetro	Categoría	Descripción
-size	**Obligatorio al crear**	Este parámetro recibirá un número que indicará el tamaño de la partición a crear. Debe ser positivo y mayor a cero, de lo contrario se mostrará un mensaje de error.
-unit	**Opcional**	Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro s. Podrá tener los siguientes valores: **B**: indicará que se utilizarán bytes. **K**: indicará que se utilizarán Kilobytes(1024 bytes) **M**:indicará que se utilizarán Megabytes(1024 * 1024 bytes). Este parámetro es opcional, si no se encuentra se creará una partición en Kilobytes. Si se utiliza un valor diferente mostrará un mensaje de error.
-path	**Obligatorio**	Este parámetro será la ruta en la que se encuentra el disco en el que se creará la partición. Este archivo ya debe existir, si no se mostrará un error.
-type	**Opcional**	Indicará que tipo de partición se creará. Ya que es opcional, se tomará como primaria en caso de que no se indique. Podrá tener los siguientes valores: **P**: Se creará una partición primaria. **E**: Se creará una partición extendida. **L**: Se creará una partición lógica. Si se utiliza otro valor diferente a los anteriores deberá mostrar un mensaje de error. Las particiones lógicas sólo pueden estar dentro de la extendida sin sobrepasar su tamaño. Deberá tener en cuenta las restricciones de teoría de particiones: ● La suma de primarias y extendidas debe ser como máximo 4. ● Solo puede haber una partición extendida por disco. ● No se puede crear una partición lógica si no hay una extendida.

-fit	**Opcional**	Indicará el ajuste que utilizará la partición para asignar espacio. Podrá tener los siguientes valores: **BF**: Indicará el mejor ajuste (Best Fit) **FF**: Utilizará el primer ajuste (First Fit) **WF**: Utilizará el peor ajuste (Worst Fit) Ya que es opcional, se tomará el peor ajuste (WF) si no está especificado en el comando. Si se utiliza otro valor que no sea alguno de los anteriores mostrará un mensaje de error.
-name	**Obligatorio**	Indicará el nombre de la partición. El nombre no debe repetirse dentro de las particiones de cada disco. Si se va a eliminar, la partición ya debe existir, si no existe debe mostrar un mensaje de error.

Ejemplos:

```
#Crea una partición primaria llamada Particion1 de 300kb
#con el peor ajuste en el disco Disco1.mia
fdisk -Size=300 -path=/home/Disco1.mia -name=Particion1

#Crea una partición extendida dentro de Disco2 de 300kb
#Tiene el peor ajuste
fdisk -type=E -path=/home/Disco2.mia -Unit=K -name=Particion2
-size=300

#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"

#Intenta crear una partición extendida dentro de Disco2 de 200 kb
#Debería mostrar error ya que ya existe una partición extendida
#dentro de Disco2
fdisk -type=E -path=/home/Disco2.dk -name=Part3 -Unit=K -size=200
```

4. MOUNT

Este comando montará una partición del disco en el sistema. Cada partición se identificará por un id que tendrá la siguiente estructura utilizando el número de carnet:

- Últimos dos dígitos del Carnet + Número de Partición + Letra

Ejemplo: carnet = 202401234

Id´s = 341A, 341B, 341C, 342A, 343A

El número será el mismo para particiones en el mismo disco y la letra diferente para particiones en el mismo disco. (NOTA: Este Comando Debe Realizar el montaje en memoria ram no debe escribir esto en el disco).

Observaciones:

1. Este comando debe realizar el montaje, la cual cambia el atributo estatus de la estructura partición y actualizará su correlativo junto su ID .
2. Las únicas particiones que se pueden montar por teoría son Primarias y Lógicas, por temas de que es una simulación de dichos sistemas de archivos solo se trabajarán los montajes con particiones primarias.

Los parámetros admitidos por este comando son:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	**Obligatorio**	Este parámetro será la ruta en la que se encuentra el disco que se montará en el sistema. Este archivo ya debe existir.
-name	**Obligatorio**	Indica el nombre de la partición a cargar. Si no existe debe mostrar error.

Ejemplos:

```
#Monta las particiones de Disco1.mia *canet = 202401234
mount -path=/home/Disco1.mia -name=Part1 #id=341a
mount -path=/home/Disco2.mia -name=Part1 #id=342a
mount -path=/home/Disco3.mia -name=Part2 #id=343a
```

Administración del Sistema de Archivos

Los comandos de este apartado simularán el formateo de las particiones, administración de usuarios, carpetas y archivos a excepción de los comandos MKFS y LOGIN todos los demás comandos requieren que exista una **sesión activa** para su ejecución en caso de no ser así se debería de indicar mediante un error que no existe una sesion activa.

5. MKFS

Este comando realiza un formateo completo de la partición, se formatea como ext2. También creará un archivo en la raíz llamado users.txt que tendrá los usuarios y contraseñas del sistema de archivos. La estructura de este archivo se explicará más adelante. Tendrá los siguientes parámetros

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-id	**Obligatorio**	Indicará el id que se generó con el comando mount. Si no existe mostrará error. Se utilizará para saber la partición y el disco que se utilizará para hacer el sistema de archivos.
-type	Opcional	Indicará que tipo de formateo se realizará. Podrá tener los siguientes valores: **Full**: en este caso se realizará un formateo completo. Ya que es opcional, se tomará como un formateo completo si no se especifica esta opción.

Ejemplos:

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

#Realiza un formateo completo de la partición que ocupa el id 342A
mkfs -id=342A
```

Observaciones

- Este comando es el que agrega las estructuras requeridas en el disco.

6. CAT

Este comando permitirá mostrar el contenido del archivo, si el usuario que actualmente está logueado tiene acceso al **permiso de lectura**.

Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-filen	Obligatorio	Permitirá admitir como argumentos una lista de N ficheros que hay que enlazar. Estos se deben encadenar en el mismo orden en el cual fueron especificados. Si no existe el archivo o no tiene permiso de lectura, debe mostrarse un mensaje de error.

Ejemplos:

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

#En la área de texto de salida debería mostrar el contenido, en
este ejemplo #01234567890123
#enlazara los archivos
# a.txt (datos archivo a) # b.txt (01234567890123) # c.txt (0123)
y debería mostrar el contenido
# siguiente, cada archivo va separado por salto de línea
# datos archivo a
# 01234567890123
# 0123
cat -file1="/home/a.txt" -file2="/home/b.txt" -file3="/home/c.txt"
```

Administración de Usuarios y Grupos

Este archivo lógico almacenado en el disco será llamado users.txt guardado en el sistema ext2 de la raíz de cada partición. Existirán dos tipos de registros, unos para grupos y otros para usuarios. Un id 0 significa que el usuario o grupo está eliminado, el id de grupo o de usuario irá aumentando según se vayan creando usuarios o grupos.

Tendrá la siguiente estructura:

GID, Tipo, Grupo \n
UID, Tipo, Grupo, Usuario, Contraseña \n

El estado ocupará una letra, el tipo otra, el grupo ocupará como máximo **10 letras** al igual que el usuario y la contraseña.

Al inicio existirá un grupo llamado **root**, un usuario **root** y una contraseña (123) para el usuario root. El archivo lógico almacenado en el disco al inicio debería ser como el siguiente:

1, G, root \n
1, U, root, root, 123 \n

- Este archivo se podrá modificar con comandos que se explicarán más adelante.
- Al crear el archivo user.txt en el inodo se agrega número 1 de usuario y 1 de grupo que son del usuario root ya que solo él tendrá los permisos para modificarlo en ese momento.

7. LOGIN

Este comando se utiliza para iniciar sesión en el sistema. No se puede iniciar otra sesión sin haber hecho un **LOGOUT** antes, en caso contrario debe mostrar un mensaje de error indicando que debe cerrar sesión con anterioridad. Este comando recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-user	**Obligatorio**	Especifica el nombre del usuario que iniciará sesión. Si no se encuentra mostrará un mensaje indicando que el usuario no existe. ***distinguir mayúsculas de minúsculas.**
-pass	**Obligatorio**	Indicará la contraseña del usuario que inicia sesión. Si no coincide debe mostrar un mensaje de autenticación fallida. ***distinguirá entre mayúsculas y minúsculas.**
-id	**Obligatorio**	Indicará el id de la partición montada de la cual van a iniciar sesión. De lograr iniciar sesión todas las acciones se realizarán sobre este id.

Ejemplos:

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

```
#Debe dar error porque ya hay un usuario logueado
login -user="mi usuario" -pass="mi pwd" -id=062A
```

8. LOGOUT

Este comando se utiliza para cerrar sesión. Debe haber una sesión activa anteriormente para poder utilizarlo, si no, debe mostrar un mensaje de error. Este comando no recibe parámetros.

Ejemplos:

```
#Termina la sesión del usuario
Logout

#Si se vuelve a ejecutar deberá mostrar un error ya que no hay
sesión actualmente
Logout
```

Observaciones:

- Los siguientes comandos que se verán a continuación necesitan que exista una sesión en el sistema ya que se ejecutan sobre la partición en la que inicio sesión. Si no, debe mostrar un mensaje de error indicando que necesita iniciar sesión.

9. MKGRP

Este comando creará un grupo para los usuarios de la partición y se guardará en el archivo users.txt de la partición, este comando solo lo puede utilizar el usuario **root**. **Si otro usuario lo intenta ejecutar, deberá mostrar un mensaje de error**, si el grupo a ingresar ya existe deberá mostrar un mensaje de error. Distinguirá entre mayúsculas y minúsculas. Recibirá los Siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-name	**Obligatorio**	Indicará el nombre que tendrá el grupo

Ejemplos:

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

El archivo users.txt debería quedar como el siguiente:

1, G, root \n
1, U, root, root, 123 \n
2, G, usuarios \n

---

10. RMGRP

Este comando eliminará un grupo para los usuarios de la partición. Solo lo puede utilizar el usuario root, si lo utiliza alguien más debe mostrar un error. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-name	**Obligatorio**	Indicará el nombre del grupo a eliminar. Si el grupo no se encuentra dentro de la partición debe mostrar un error.

Ejemplo:

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

#Debe mostrar mensaje de error ya que el grupo no existe porque ya
#fue eliminado
rmgrp -name=usuarios
```

El archivo users.txt debería quedar como el siguiente:

1, G, root \n
1, U, root, root, 123 \n
0, G, usuarios \n

---

11. MKUSR

Este comando crea un usuario en la partición. Solo lo puede ejecutar el usuario root, si lo utiliza otro usuario deberá mostrar un error. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-user	**Obligatorio**	Indicará el nombre del usuario a crear, si ya existe, deberá mostrar un error indicando que ya existe el usuario. **Máximo: 10 caracteres.**
-pass	**Obligatorio**	Indicará la contraseña del usuario **Máximo 10 Caracteres**
-grp	**Obligatorio**	Indicará el grupo al que pertenece el usuario. Debe de existir en la partición en la que se está creando el usuario, si no debe mostrar un mensaje de error. **Máximo 10 Caracteres**

Ejemplos:

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

#Debe mostrar mensaje de error ya que el usuario ya existe
independientemente que esté en otro grupo
mkusr -user=user1 -pass=usuario -grp=usuarios2
```

El archivo users.txt debería quedar así:

1, G, root \n
1, U, root, root, 123 \n
2, G, usuarios \n
2, U, usuarios, user1, usuario \n

---

12. RMUSR

Este comando elimina un usuario en la partición. Solo lo puede ejecutar el usuario root, si lo utiliza otro usuario deberá mostrar un error. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-user	**Obligatorio**	Indicará el nombre del usuario a eliminar. Si el usuario no se encuentra dentro de la partición debe mostrar un error.

Ejemplo:

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

#Debe mostrar mensaje de error porque el user1 ya no existe
rmusr -user=user1
```

El archivo users.txt debería quedar así:

1, G, root \n
1, U, root, root, 123 \n
2, G, usuarios \n
0, U, usuarios, user1, usuario \n

---

13. CHGRP

Cambiará el grupo al que pertenece el usuario. Únicamente lo podrá utilizar el usuario root. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-user	**Obligatorio**	Especifica el nombre del usuario al que se le cambiará de grupo. Si no existe debe mostrar un error.
-grp	**Obligatorio**	Contendrá el nombre del nuevo grupo al que pertenece el usuario. Si no existe o está eliminado debe mostrar un error.

Ejemplos:

```
#Cambia el grupo del user2
chgrp -user=user2 -grp=grupo1
#Cambia el grupo del user1
chgrp -user=user1 -grp=grupo2
```

USUARIO ROOT

Este usuario es especial y no importando que permisos tiene el archivo o carpeta, se manejan permisos UGO en su forma octal y este siempre tendrá los **permisos 777** sobre cualquier archivo o carpeta. Podrá mover, copiar, eliminar, crear, etc. Todos los archivos o carpetas que desee. No se le negará ninguna operación por permisos, ya que él los tiene todos. Los permisos únicamente se pueden cambiar con *chmod* que se explicará posteriormente en el proyecto 2.

Se debe tomar en cuenta en qué categoría está el usuario, si es el propietario, si pertenece al mismo grupo en que está el propietario o si es otro usuario que no pertenece al grupo del propietario. En base a esta comprobación, el usuario puede estar en tres distintas categorías:

- User (Propietario) (**U**)
- Grupo (**G**)
- Otro (**O**)

Dependiendo de estas categorías se determinan los permisos hacia el archivo o carpeta.

Linux File Permission Cheatsheet

Administración de Carpetas Archivos y Permisos

Estos comandos permitirán crear archivos y carpetas, así como editarlos, copiarlos, moverlos y eliminarlos. Los permisos serán para el usuario propietario del archivo, para el grupo al que pertenece y para otros usuarios, así como en Linux

14. MKFILE

Este comando permitirá crear un archivo, **el propietario será el usuario que actualmente ha iniciado sesión**. Tendrá los permisos **664**. El usuario

deberá tener el permiso de escritura en la carpeta padre, si no debe mostrar un error. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	**Obligatorio**	Este parámetro será la ruta del archivo que se creará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si ya existe debe mostrar un mensaje si se desea sobreescribir el archivo. Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro **r**, que se explica posteriormente.
-r	**Opcional**	Si se utiliza este parámetro y las carpetas especificadas por el parámetro **path** no existen, entonces deben crearse las carpetas padres. Si ya existen, no deberá crear las carpetas. No recibirá ningún valor, si lo recibe debe mostrar error.
-size	**Opcional**	Este parámetro indicará el tamaño en bytes del archivo, El contenido serán números del 0 al 9 cuantas veces sea necesario hasta cumplir el tamaño ingresado. Si no se utiliza este parámetro, el tamaño será 0 bytes. Si es negativo debe mostrar error.
-cont	**Opcional**	Indicará un archivo en el disco de la computadora, (entiéndase su computadora) que tendrá el contenido del archivo. Se utilizará para cargar contenido en el archivo. La ruta ingresada debe existir, sino mostrará un mensaje de error.

- Si se ingresan los parámetros cont y size, tendrá mayor prioridad el parámetro cont *.

Ejemplos:

```
#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

#Crea "archivo 1.txt" la carpeta "mis documentos" ya debe existir
#el tamaño es de 0 bytes
mkfile -path="/home/mis documentos/archivo 1.txt"
```

```
#Crea el archivo b.txt
#El contenido del archivo será el mismo que el archivo b.txt #que
se encuentra en el disco duro de la computadora.
mkfile -path=/home/user/docs/b.txt -r -cont=/home/Documents/b.txt
```

15. MKDIR

Este comando es similar a mkfile, pero no crea archivos, sino carpetas. El propietario será el usuario que actualmente ha iniciado sesión. Tendrá los **permisos 664**. El usuario deberá tener el permiso de escritura en la carpeta padre, si no debe mostrar un error. Tendrá los siguientes parámetros

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	**Obligatorio**	Este parámetro será la ruta de la carpeta que se creará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro **r**.
-p	**Opcional**	Si se utiliza este parámetro y las carpetas padres en el parámetro path no existen, entonces deben crearse. Si ya existen, no realizará nada. No recibirá ningún valor, si lo recibe debe mostrar error.

Ejemplos:

```
#Crea la carpeta usac
#Si no existen las carpetas home user o docs se crean
mkdir -p -path=/home/user/docs/usac

#Crea la carpeta "archivos diciembre" #La carpeta padre ya debe
existir
mkdir -path="/home/mis documentos/archivos clases"
```

Otras Operaciones

Aquí se aclaran algunas otras operaciones que se deben realizar. Por ejemplo, que se utilizará el ajuste de la partición para buscar bloques libres y contiguos al momento de crear los archivos.
Al momento de modificar archivos pueden darse tres casos:

- **Ocupa más espacio:** En este caso se utiliza el ajuste de la partición para buscar nuevos bloques contiguos y almacenar el contenido que exceda a los bloques reutilizados. El contenido se escribe en los bloques que ya se están utilizando y el excedente en los bloques nuevos.
- **Ocupa menos espacio:** Si utiliza menos bloques, únicamente los marcará como libres en el bitmap y eliminará las referencias hacia los bloques en los inodos o bloques de apuntadores indirectos.
- **Ocupa igual espacio:** Si utiliza la misma cantidad, solo modifica los bloques.

En cualquiera de los casos anteriores debe modificarse el bitmap si es necesario y los datos del inodo (fecha de modificación, etc.)

- Si un bloque de apuntadores indirectos queda vacío, se debe marcar como libre en el bitmap y quitar la referencia del inodo o bloque de apuntadores que lo estaba utilizando.
- Si un archivo ocupa 0 bytes no tendrá bloque asociado.
- Cada comando anterior debe modificar las características de los inodos según considere necesario, por ejemplo, un cambio de permisos sobre el archivo modificará el campo i_perm del inodo.
- Siempre debe existir la carpeta raíz y el archivo de usuarios users.txt en la raíz.

Script

Estos archivos contienen comandos definidos en este documento, y pueden incluir también comentarios y líneas en blanco. Los archivos tendrán la extensión **.smia** y se utilizarán para cargar comandos de manera masiva en el área de texto de entrada.

Los comentarios en estos archivos estarán limitados a una sola línea y comenzarán siempre con el símbolo **#**. Estos comentarios serán mostrados tal como aparecen en el archivo, permitiendo a los usuarios entender mejor el contenido o propósito de los comandos incluidos. Las líneas en blanco también serán preservadas, proporcionando una mayor claridad en la estructura del archivo.

Reportes

Se deberán generar los reportes con el comando rep. Se generarán en graphviz. Se puede utilizar html dentro de los reportes si el estudiante lo considera necesario. Deberá mostrarlos de forma similar a los ejemplos mostrados.

Observaciones:

- Los reportes son de suma importancia ya que son por medio de donde podremos validar visualmente que el resultado de las operaciones es el esperado.

16. REP

Recibirá el nombre del reporte que se desea y lo generará con graphviz en una carpeta existente. Tendrá los siguientes parámetros

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-name	**Obligatorio**	Nombre del reporte a generar. Tendrá los siguientes valores: ● mbr ● disk ● inode ● block ● bm_inode ● bm_block ● sb ● file ● ls Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error.
-path	**Obligatorio**	Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error. Indica una carpeta y el nombre que tendrá el reporte. Si no existe la carpeta, deberá crearla. Si lleva espacios se encerrará entre comillas
-id	**Obligatorio**	Indica el id de la partición que se utilizará. Si el reporte es sobre la información del disco, se utilizará el disco al que pertenece la partición. Si no existe debe mostrar un error.
-path_file_ls	**Opcional**	Funcionará para el reporte file y ls. Será el nombre del archivo o carpeta del que se mostrará el reporte. Si no existe muestra error.

Reporte MBR

Mostrará tablas con toda la información del MBR, así como de los EBR que se pudieron haber creado.

Ejemplo:

```
#MBR y EBR Disco1.dsk
rep -id=A118 -path=/home/user/reports/reporte1.jpg -name=mbr
```

REPORTE DE MBR	
mbr_tamano	52428800
mbr_fecha_creacion	2020-12-12 02:22
mbr_disk_signature	74
Particion	
part_status	0
part_type	e
part_fit	w
part_start	7864512
part_size	7864320
part_name	part2
Particion Logica	
part_status	0
part_next	10322208
part_fit	b
part_start	7864512
part_size	1228800
part_name	part5
Particion Logica	
part_status	1
part_next	-1
part_fit	w
part_start	10322208
part_size	1228800
part_name	part7
Particion	
part_status	1
part_type	p
part_fit	w
part_start	15728832
part_size	7864320
part_name	part3
Particion	
part_status	0
part_type	p
part_fit	b
part_start	23593152
part_size	7864320
part_name	part4

Reporte de MBR

EBR

Particion	
part_status	1
part_type	p
part_fit	w
part_start	15728832
part_size	7147520
part_name	part3
Particion	
part_status	0
part_type	p
part_fit	b
part_start	23593152
part_size	7045120
part_name	part4

Reporte DISK

Este reporte mostrará la estructura de las particiones, el mbr del disco y el porcentaje que cada partición o espacio libre tiene dentro del disco (La sumatoria de los porcentajes debe de ser 100%).

Ejemplo:

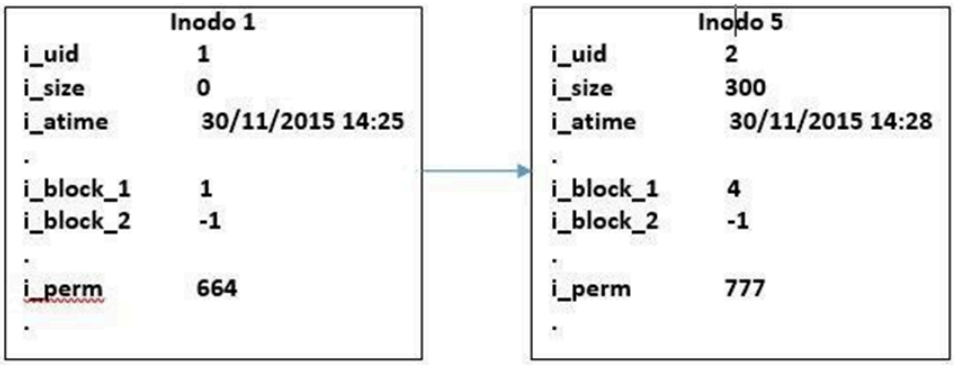
```
#Reporte de discos
rep -id=A118 -path=/home/user/reports/report2.pdf -name=disk
```

Reporte Inode

Mostrará bloques con toda la información de los inodos utilizados. Si no están utilizados no debe mostrarlos.

Ejemplo:

```
#Reporte de inodos
rep -id=A118 -path=/home/user/reports/report3.jpg -name=inode
```

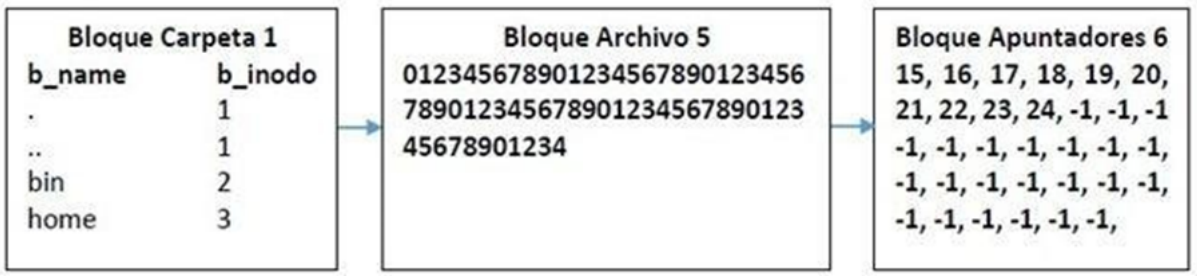


Reporte Block (Carpetas y Archivos)

Mostrará la información de todos los bloques utilizados. Si no están utilizados no debe mostrarlos.

Ejemplo:

```
#Reporte del bloque
rep -id=A118 -path=/home/user/reports/report4.jpg -name=block
```

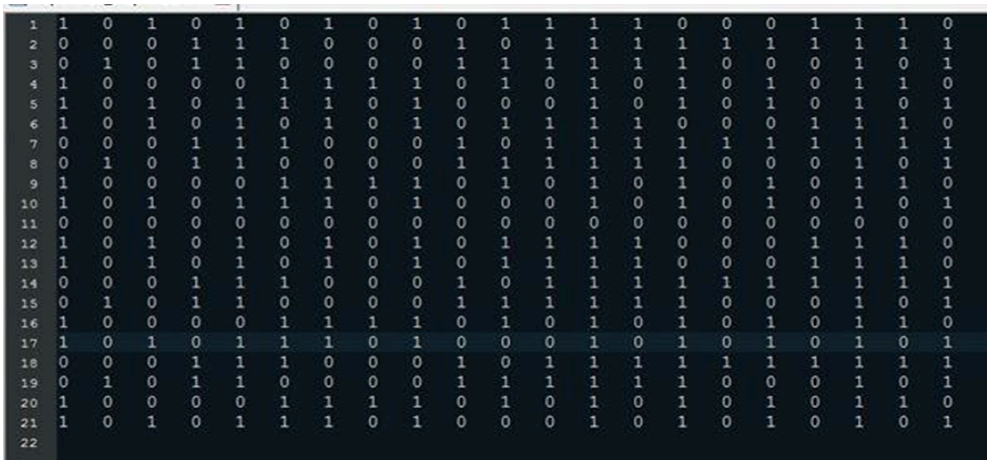


Reporte bm_inode

Este reporte mostrará la información del bitmap de inodos, mostrará todos los bits, libres o utilizados. Este reporte se generará en un archivo de texto mostrando 20 registros por línea.

Ejemplo:

```
#Genera el bitmap de inodos de los bloques
rep -id=A118 -path=/home/user/reports/report5.txt -name=bm_inode
```



Reporte bm_bloc

Este reporte mostrará la información del bitmap de inodos, desplegará todos los bits, libres o utilizados. Este reporte se generará en un archivo de texto que mostrará 20 registros por línea.

Ejemplo:

```
#Genera el bitmap de inodos de los bloques
rep -id=A118 -path=/home/user/reports/report6.txt -name=bm_bloc
```

Reporte Sb

Muestra toda la información del superbloque en una tabla.

Ejemplo:

```
#SuperBloque Partición Correlativo 1 en disco A.dsk
rep -id=A118 -path=/home/user/reports/report8.jpg -name=sb
```

Reporte de SUPERBLOQUE	
sb_nombre_hd	disco1.dsk
sb_arbol_virtual_count	160
sb_detalle_directorio_count	160
sb_inodos_count	735
sb_bloques_count	2940
sb_arbol_virtual_free	133
sb_detalle_directorio_free	132
sb_inodos_free	735
sb_bloques_free	2940
sb_date_creacion	2020-12-09 18:10
sb_date_ultimo_montaje	2020-12-09 18:10
sb_montajes_count	1
sb_ap_bitmap_arbol_directorio	544
sb_ap_arbol_directorio	691
sb_ap_bitmap_detalle_directorio	17155
sb_ap_detalle_directorio	17302
sb_ap_bitmap_inodos	59638
sb_ap_inodos	60373
sb_ap_bitmap_bloques	119173
sb_ap_bloques	122113
sb_ap_log	195613
sb_size_struct_arbol_directorio	112
sb_size_struct_detalle_directorio	288
sb_size_struct_inodo	80
sb_size_struct_bloque	25
sb_first_free_bit_arbol_directorio	15
sb_first_free_bit_detalle_directorio	16
sb_first_free_bit_tabla_inodos	19
sb_first_free_bit_bloques	73
sb_magic_num	201314821

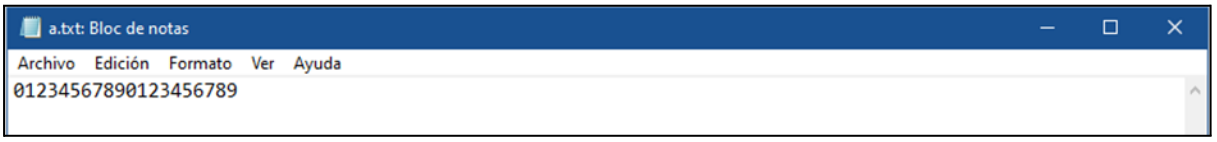
Reporte de SUPERBLOQUE

Reporte File

Este reporte muestra el nombre y todo el contenido del archivo especificado en el parámetro file.

Ejemplo:

```
# Genera el reporte de tipo file
rep -id=A118 -path=/home/user/reports/report9.txt
-path_file_ls=/home/a.txt  -name=file
```



Reporte Ls

Este reporte mostrará la información de los archivos y carpetas con permisos, propietario, grupo propietario, fecha de modificación, hora de modificación, tipo, fecha de creación.

Ejemplo:

```
#Reporte de archivos y carpetas
rep -id=A118 -path=/home/user/reports/report10.jpg -path_file_ls=/
-name=ls
```

Permisos	Owner	Grupo	Size (en Bytes)	Fecha	Hora	Tipo	Name
-rw-rw-r--	User1	Mi grupo	40661	24/02/2019	9:53	Archivo	Ejemplo.txt
-rw-r--rwx	User2	Otro grupo	123	20/08/2019	8:13	Carpeta	Home

Instrucciones de Entrega

El proyecto se entregará el **9 de septiembre hasta las 23:59 horas**. Se utilizará un repositorio de github para que suban su proyecto y se habilitará una opción en UEDI para que puedan subir el link de su repositorio, los auxiliares de cada curso deberán tener acceso a los repositorios respectivos en cualquier momento de la duración del laboratorio, si no se cuenta con acceso se anulara el proyecto, se recomienda que sea un repositorio privado para evitar copias. La impuntualidad anulara el trabajo entregado. Se calificará el último commit que suban a la hora estipulada.

Nombre del repositorio: **MIA_2S_P1_carnet**

Usuarios de github de los auxiliares de cada sección:

1. ***Sección A***: yonaldez0
2. ***Sección B***: keviingarciah
3. ***Sección C***: Allenrovas
4. ***Sección D***: AndresPontaza

Requisitos Mínimos

Para tener derecho a calificación se deberá contar con requisitos mínimos los cuales son:

● Aplicación Web
● Creación de Particiones con la aplicación de los ajustes y Mount
● Ejecución Completa del Script

Consideraciones

El proyecto debe realizarse de forma individual, ***Se utilizará software para la detección de copias, las copias tendrán una nota de 0 y serán reportadas a la escuela.***

- Para la creación del frontend se permite el uso de cualquier framework.
- Para el backend el lenguaje a utilizar es **Go**. No se permite el uso de otro lenguaje.
- Únicamente se calificará el proyecto sobre una **instalación física** de una distribución GNU/Linux.
- No se permite la modificación de código durante la calificación.
- El archivo binario que representa a los discos no debe crecer.
- No se permite la utilización de estructuras en memoria (listas, árboles, etc.) para el manejo de los archivos o carpetas.
- No se permite agregar o quitar atributos a las estructuras que se utilizarán en el proyecto.
- Se calificará basado en **reportes** en su mayor parte.

Aprobaciones de los Ingenieros