



Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Manejo e Implementación de Archivos

Segundo Semestre 2023

Catedráticos: Ing. Álvaro Díaz, Ing. Oscar Paz, Ing. William Escobar, Ing. Jorgen Ramiez

Tutores académicos: Angel Arteaga, Jonathan Alvarado, Sergie Arizandieta, Daniel Chicas

Proyecto 2

Introducción:

El curso de Manejo e Implementación de Archivos busca que los estudiantes aprendan los conceptos sobre la administración de archivos, tanto en hardware como software, sistemas de archivos, particiones, uso de tecnologías en la nube para la ejecución de la aplicación, entre otros conceptos, así mismo trata que los estudiantes apliquen estos conceptos en el desarrollo de un proyecto para que de esta manera puedan aprender cada uno de los temas impartidos durante la clase magistral y el laboratorio para que luego se le pueda dar paso a los conocimientos que se impartirán en cursos posteriores como lo son las bases de datos.

Objetivos:

- Aprender a administrar archivos y escribir estructuras en Python
- Tener acercamiento con servicios básicos en nube
- Crear una aplicación de comandos
- Aplicar el formateo rápido y completo en una partición
- Comprender el sistema de archivos EXT2
- Aplicar la teoría de particiones
- Utilizar Graphviz para mostrar reportes
- Formateo del sistema de archivos
- Creación de Archivos dentro del sistema
- Carga de información de archivos reales a archivos en el sistema

Aplicación de comandos

Se dispondrá de un **Ciente** que tendrá una **interfaz gráfica** para poder ingresar los diferentes archivos de entrada, a excepción de los reportes en Graphviz. No distinguirá entre mayúsculas y minúsculas. Hay parámetros obligatorios y opcionales. Solo se puede colocar un comando por línea.

Si se utiliza un parámetro que no está especificado en este documento, debe mostrar un mensaje de error. Se utilizarán espacios en blanco para separar cada parámetro. Si se necesita que algún valor lleve espacios en blanco se encerrará entre comillas " ", **Los parámetros pueden venir en cualquier orden.**

Comandos

Parte 1: Administración de Discos

Estos comandos permitirán crear archivos que simularán discos duros en los que se podrá formatear más adelante con el sistema de archivos ext2. Estos comandos estarán disponibles desde que se inicia el programa. Estos comandos son:

1) MKDISK

Este comando creará un archivo binario que simulará un disco duro, estos archivos binarios tendrán la extensión **dsk** y su contenido al inicio será 0. 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.
-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.
-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	<p>Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro size. Podrá tener los siguientes valores:</p> <p>K: Indicará que se utilizarán Kilobytes (1024 bytes) M: Indicará que se utilizarán Megabytes (1024 * 1024 bytes)</p> <p>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.</p>

Ejemplos:

#Crea un disco de 3000 Kb en la carpeta home

```
mkdisk -size=3000 -unit=K -path=/home/user/Disco1.dsk
```

#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.dsk -unit=K -size=3000
```

#Se ponen comillas la ruta de path ya que uno de sus elementos tiene espacios en blanco, se crea si no está o no existe

```
mkdisk -size=5 -unit=M -path="/home/mis discos/Disco3.dsk"
```

#Crearé un disco de 10 Mb ya que no hay parámetro unit

```
mkdisk -size=10 -path="/home/mis discos/Disco4.dsk"
```

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 con rmdisk Disco4.dsk

```
rmdisk -path="/home/mis discos/Disco4.dsk"
```

3) FDISK

Este comando administra las particiones en el archivo que representa al disco duro. Deberá mostrar un error si no se pudo realizar la operación solicitada sobre la partición, especificando porqué razón no pudo crearse (Por espacio, por restricciones de particiones, etc.).

No se considerará el caso de que se pongan parámetros incompatibles, por ejemplo, en un mismo comando fdisk llamar a delete y add. La estructura de cada disco se explicará más adelante. 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.
-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.
-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.
-unit	Opcional	<p>Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro s. Podrá tener los siguientes valores:</p> <p>B: indicará que se utilizarán bytes. K: indicará que se utilizarán Kilobytes(1024 bytes) M: indicará que se utilizarán Megabytes(1024 * 1024 bytes).</p> <p>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.</p>

-type	Opcional	<p>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> <p>P: Se creará una partición primaria. E: Se creará una partición extendida. L: Se creará una partición lógica.</p> <p>Si se utiliza otro valor diferente a los anteriores deberá mostrar un mensaje de error.</p> <p>Las particiones lógicas sólo pueden estar dentro de la extendida sin sobrepasar su tamaño. Deberá tener en cuenta las restricciones de teoría de particiones:</p> <ul style="list-style-type: none"> • 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	<p>Indicará el ajuste que utilizará la partición para asignar espacio. Podrá tener los siguientes valores:</p> <p>BF: Indicará el mejor ajuste (Best Fit) FF: Utilizará el primer ajuste (First Fit) WF: Utilizará el peor ajuste (Worst Fit)</p> <p>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.</p>

Ejemplos:

#Crea una partición primaria llamada Particion1 de 300 kb

#con el peor ajuste en el disco Disco1.dsk

```
fdisk -size=300 -path=/home/Disco1.dsk -name=Particion1
```

#Crea una partición extendida dentro de Disco2 de 300 kb

#Tiene el peor ajuste

```
fdisk -type=E -path=/home/Disco2.dsk -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.dsk"
-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.dsk -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 Partición + NombreDisco Ejemplo:
por ejemplo para el carnet -> 201404106

Id's -> 061Disco1, 062Disco1, 061Disco2, 062Disco2

El número representará las particiones en el mismo disco y la letra representará al disco. (**NOTA: Este Comando Debe Realizar el montaje en memoria ram no debe escribir esto en el disco**).

Debe de existir alguna manera en la cual mostrar las particiones montadas por la aplicación (Esto queda a discreción de cada estudiante). 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.dsk, Disco2.dsk y Disco3.dsk,

#Carnet Ejemplo -> 201404106

```
mount -path=/home/Disco1.dsk -name=Part1 #id=061Disco1
```

```
mount -path=/home/Disco2.dsk -name=Part1 #id=061Disco2
```

```
mount -name=Part2 -path=/home/Disco3.dsk #id=061Disco3
```

```
mount -path=/home/Disco1.dsk -name=Part2 #id=062Disco1
```

```
mount -path=/home/Disco2.dsk -name=Part2 #id=062Disco2
```

mount -name=Part3 -path=/home/Disco3.dsk #id=062Disco3

Parte 2: 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 sesión 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.

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 061Disco1 en ext2
mkfs -type=full -id=061Disco1

Administración de Usuarios y Grupos:

Este archivo lógico almacenado en el disco será llamado users.txt guardado en el sistema ext2/ext3 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
UID, Tipo, Grupo, Usuario, Contraseña

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.

6) 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=061Disco1
```

#Debe dar error porque ya hay un usuario logueado

```
login -user="mi usuario" -pass="mi pwd" -id=061Disco1
```


7) 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

8) 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

Ejemplo:

#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
```

9) 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
```

10) 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

Ejemplo:

#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
```

11) 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
```

Administración de Carpetas y Archivos

Los comandos de este apartado permiten la creación de carpetas y archivos dentro del sistema de archivos al igual que los demás comandos posteriores a la creación de discos y particiones se requiere la existencia de una sesión activa para poder ejecutar los siguientes comandos.

12) 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 sobrescribir 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 duro de la 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.

Nota: 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
```

13) 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 .
-r	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 -r -path=/home/user/docs/usac
```

#Crea la carpeta "archivos diciembre"

#La carpeta padre ya debe existir

```
mkdir -path="/home/mis documentos/archivos diciembre"
```

14) PAUSE

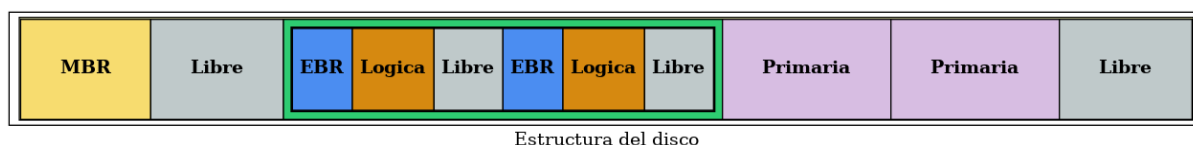
Este comando será solo la palabra "pause" no tiene atributos al ingresar este comando se pondrá en pausa solicitando que presione cualquier tecla para

continuar. Este comando NO detiene la ejecución de un archivo solo queda a la espera de presionar una tecla para continuar su ejecución.

Discos

Los discos serán archivos binarios que tendrán información del MBR, y espacio con particiones o bien, espacio sin utilizar.

La siguiente figura es un ejemplo de los bloques en un disco con particiones en el que ya se ha eliminado una partición:



Estructuras

Master Boot Record (MBR)

Cuando se crea una partición debe utilizarse el primer ajuste para crearla. En la figura anterior debería utilizarse el primer bloque libre para crear una nueva partición. El MBR tendrá los siguientes campos:

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_partition_1	partition	Estructura con información de la partición 1
mbr_partition_2	partition	Estructura con información de la partición 2
mbr_partition_3	partition	Estructura con información de la partición 3
mbr_partition_4	partition	Estructura con información de la partición 4

Partition

Nombre	Tipo	Descripción
part_status	char	Indica si la partición está activa 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

Extended Boot Record (EBR)

Las particiones extendidas tendrán una estructura diferente. Se utilizará una estructura llamada EBR (Extended Boot Record) en forma de lista enlazada, que será como la siguiente:

Nombre	Tipo	Descripción
part_status	char	Indica si la partición está activa 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

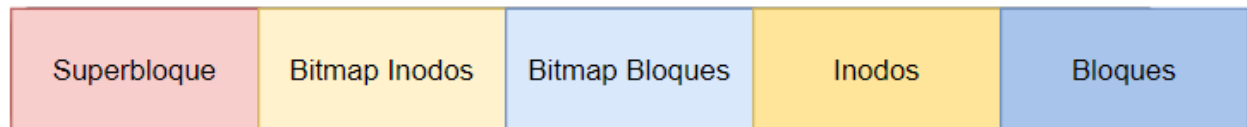


La estructura lógica de la partición extendida será la siguiente: El EBR inicial siempre debe existir, aunque se elimine la primera partición.

Sistema de Archivos EXT2

A continuación, se explicarán las estructuras 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** = $\text{sizeof}(\text{superblock}) + n + 3 * n + n * \text{sizeof}(\text{inodos}) + 3 * n * \text{sizeof}(\text{block})$
- **número_estructuras** = $\text{floor}(n)$

Súper Bloque

Contiene información sobre la configuración del sistema de archivos. 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
s_first_blo	int	Primer bloque libre
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

Esta información estará al inicio del sistema de archivos, no cambia de tamaño y se debe actualizar, según se vayan realizando las operaciones en el sistema de archivos. Por ejemplo, al usar mount, debe actualizar s_mtime, al utilizar unmount actualizará s_umtime, etc.

Tabla de inodos

Se utilizará un inodo por carpeta o archivo. Cada inodo tendrá la siguiente información:

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 15 registros son bloques directos. 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	int	Guardará los permisos del archivo o carpeta. Se trabajará a nivel de bits, estará dividido de la siguiente forma: Los primeros tres bits serán para el Usuario i_uid. Los siguientes tres bits serán para el Grupo al que pertenece el usuario. Y los últimos tres bits serán para los permisos de Otros usuarios. Cada grupo de tres bits significa lo siguiente: El primer bit indica el permiso de lectura R . El segundo bit indica el permiso de escritura W . El tercer bit indica el permiso de ejecución X .

Bloques de carpetas

información sobre el nombre de los archivos que contiene y a que inodo apuntan. La estructura es la siguiente:

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

La estructura content será como la siguiente:

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

El tamaño de este bloque será de $4 * (12 + 4) \rightarrow 64$ bytes. 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.

Bloques de Archivos

Este bloque guardará el contenido de un archivo. Su estructura es la siguiente:

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

Por lo que su tamaño, al igual que el bloque de carpetas, es de 64 bytes.

Limitaciones

En esta sección se calcularán las limitantes del sistema descrito anteriormente. Primero se calculará el máximo de bloques que puede tener asociados un inodo.

$$\text{numero_bloques_por_inodo} = 16 = 16 \text{ bloques}$$

Cada bloque de carpeta tiene una capacidad de 4 hijos. Por lo que su capacidad es de 62 carpetas o archivos dentro de una carpeta.

$$\text{capacidad_carpeta} = 16 * 4 - 2 = 62$$

Cada bloque de archivo tiene una capacidad de 64 bytes. Por lo que el tamaño máximo de un archivo es aproximadamente de 1 Kilobytes.

$$\text{capacidad_archivo} = 16 * 64 = 1024 \text{ bytes} = 1 \text{ Kilobytes}$$

Al formatear se debe crear la carpeta raíz (/) y el archivo users.txt dentro de la raíz.

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.

El formateo fast, únicamente limpia con 0s los bitmaps de inodos y bloques. El full aplica un formateo rápido y además limpia los bloques e inodos. Siempre debe existir la carpeta raíz y el archivo de usuarios `users.txt` en la raíz.

Script

Son archivos con los comandos definidos en este documento. También puede haber comentarios y líneas en blanco. Tendrán la extensión `.adsj` y se utilizarán para que los ejecute el comando `execute`.

Los comentarios serán únicamente de una línea y siempre iniciarán con el símbolo `#`.

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.

IMPORTANTE: Esta parte es obligatoria para tener derecho a la calificación de los aspectos que muestre el reporte. Si falta alguno de los reportes no se calificará.

15) REP

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

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-----------	-----------	-------------

-name	Obligatorio	<p>Nombre del reporte a generar. Tendrá los siguientes valores:</p> <ul style="list-style-type: none"> • mbr • disk • bm_inode • bm_block • tree • sb • file <p>Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error.</p>
-path	Obligatorio	<p>Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error.</p> <p>Indica una carpeta y el nombre que tendrá el reporte. Si no existe la carpeta, deberá crearla. Si lleva espacios se encerrará entre comillas</p>
-id	Obligatorio	<p>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.</p>
-ruta	Opcional	<p>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.</p>

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=061Disco1 -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:

rep -id=061Disco1 -path=/home/user/reports/report2.pdf -name=disk

Disco1.dsk

MBR	Libre 25% del disco	Extendida					Primaria 20% del disco	Libre 15% del disco
		EBR	Lógica 10% del Disco	Libre 10% del Disco	EBR	Lógica 10% del Disco		

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:

```
rep -id=061Disco1 -path=/home/user/reports/report5.txt -name=bm_inode
```

[illegible]

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:

```
rep -id=061Disco1 -path=/home/user/reports/report6.txt -name=bm_bloc
```

[illegible]

Reporte Sb

Muestra toda la información del superbloque en una tabla.

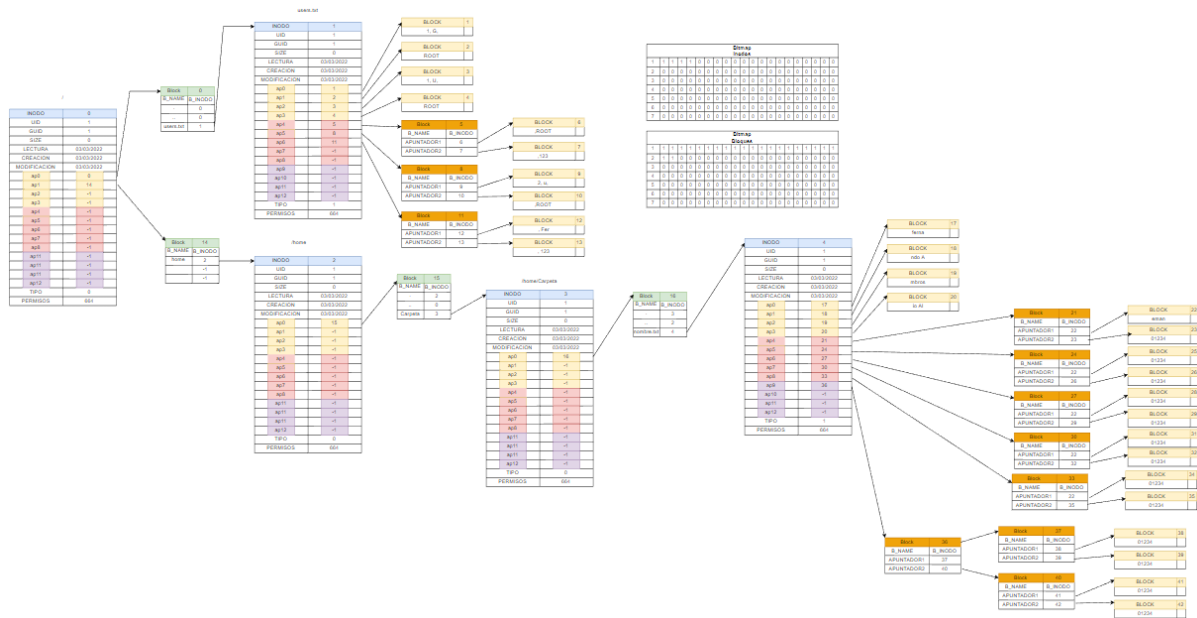
Ejemplo:

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 Tree

Este reporte genera el árbol de todo el sistema ext2. Se mostrará toda la información de los inodos o bloques. No deben ponerse los bloques o inodos libres, únicamente se pondrán los bloques que están siendo utilizados. Deberá ser como el siguiente :



Reporte File

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

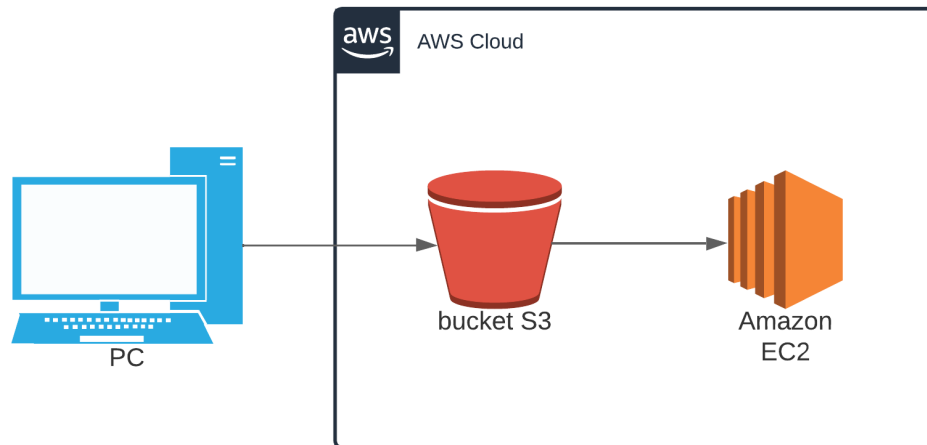


Parte 3: Ejecución en la Nube

Para esta nueva característica en el proyecto se le solicita al estudiante realizar la ejecución de la aplicación descrita mediante las partes 1 y 2 desde una máquina virtual (EC2) en la plataforma en la nube de amazon (AWS).

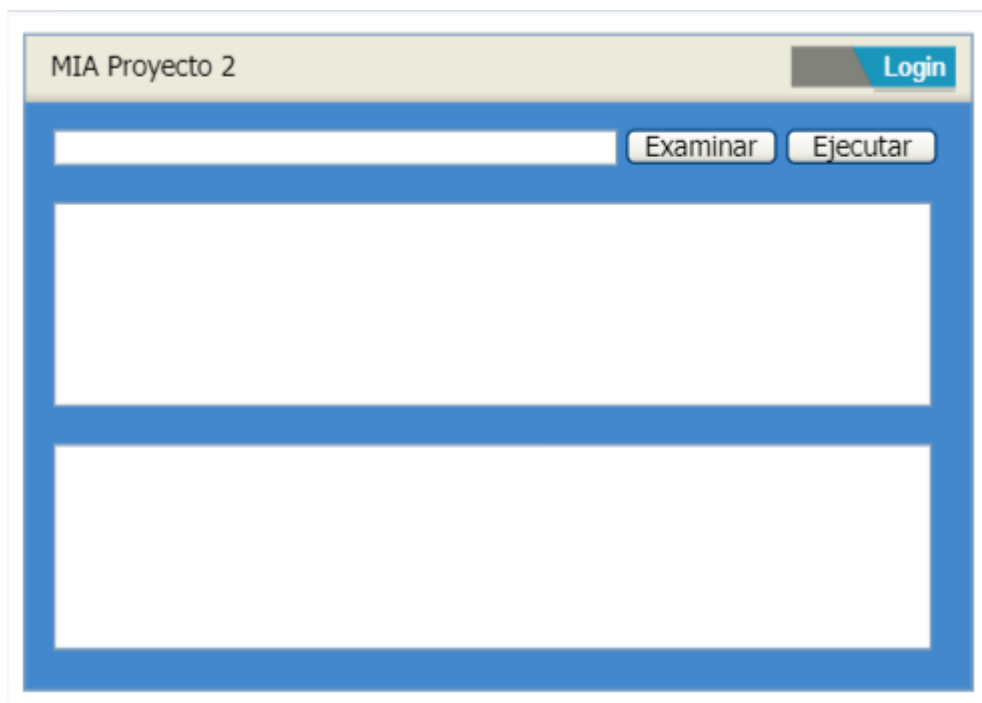
El cliente deberá estar en un bucket (S3), la cual provee una URL la cual se puede acceder a través de cualquier navegador web.

Para realizar esto es necesaria la creación de una cuenta de AWS y haciendo uso de los beneficios que nos brinda la capa gratuita. Desde el cual se debe de crear una EC2, y un bucket S3 para realizar la ejecución de la aplicación desarrollada en este sistema.

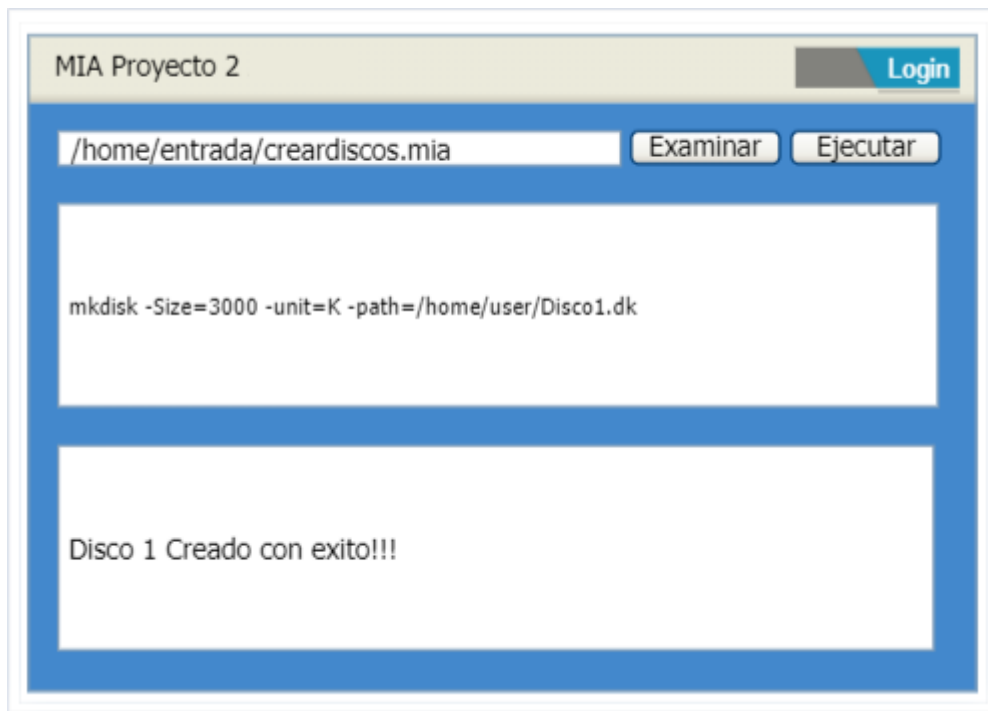


Parte 4: Cliente

- Apartado para cargar archivos de entrada a la aplicación web que dispondrá las opciones para ingresar comandos, archivos y ejecución



- Resultado luego de cargar un archivo de entrada y realizar la ejecución



The screenshot shows a web application window titled "MIA Proyecto 2". In the top right corner, there is a "Login" button. Below the title bar, there is a text input field containing the path "/home/entrada/creardiscos.mia". To the right of this field are two buttons: "Examinar" and "Ejecutar". Below the input field, there is a large white rectangular area displaying the command: `mkdisk -Size=3000 -unit=K -path=/home/user/Disco1.dk`. At the bottom of the window, another large white rectangular area displays the message: "Disco 1 Creado con exito!!!".

- Luego de la ejecución inicial, se podrá realizar el login correspondiente con el uso del ID de la partición montada y las credenciales necesarias.



The screenshot shows the same "MIA Proyecto 2" application window, but now it is in the login phase. The top right corner features a "Carga Archivo" button. The main area of the window has a blue background with the title "Login" centered at the top. Below the title, there are three labels: "ID Particion", "User", and "Password", each followed by a white text input field. At the bottom center of the login area, there is a button labeled "Ingresar".

MIA Proyecto 2

Carga Archivos

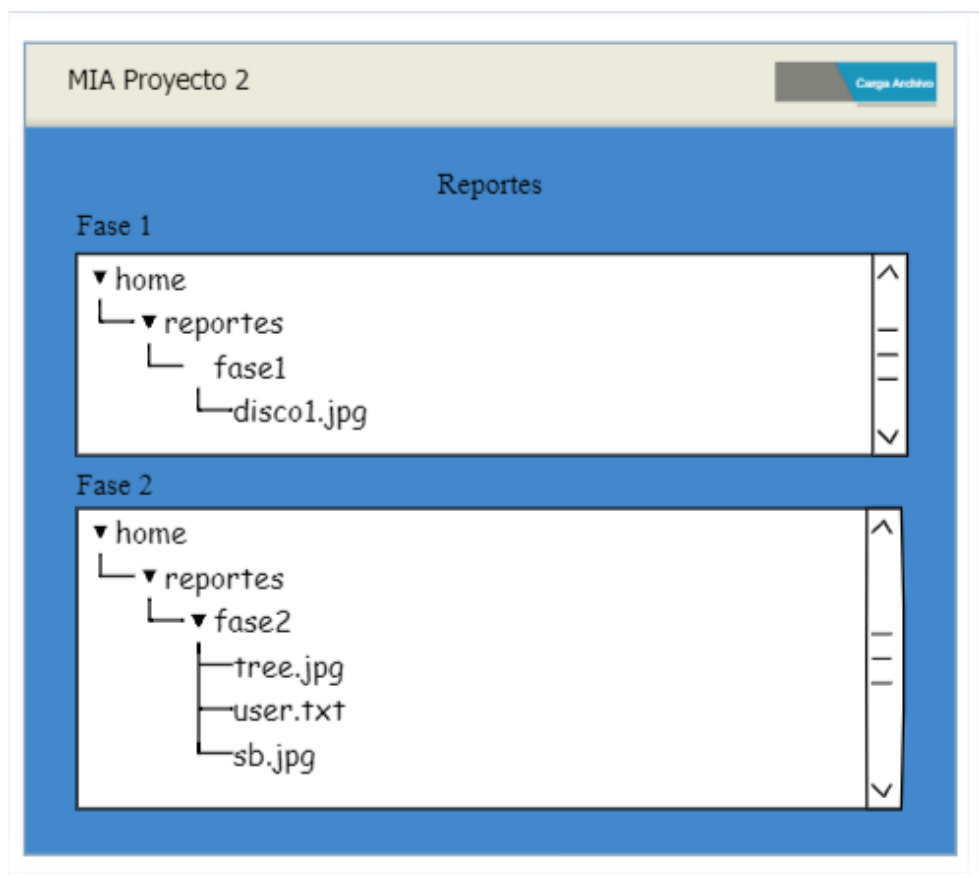
Login

ID Particion

User

Password

- En el siguiente apartado es donde se podrán visualizar los reportes correspondientes.



Para el desarrollo de la interfaz gráfica se puede hacer uso de cualquier framework web a gusto del estudiante.

El proyecto se entregará el día 5 de noviembre antes de las 23:59. 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 y se deberá de encontrar dentro del repositorio un ejecutable, desde el cual se calificará.

Nombre del repositorio: MIA_P2_carnet

Usuarios de github de los auxiliares de cada sección:

1. **Sección A:** jona140
2. **Sección B:** Daniel-Chicas
3. **Sección C:** AngelArteaga1
4. **Sección N:** SergieArizandieta

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

- Implementación de la nube con AWS (S3,EC2)
- Creación de Particiones con la aplicación de los ajustes
- Ejecución Completa del Script
- Pause
- Reportes fase 1

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.**

- El lenguaje por utilizar para el backend es Python. No se permite el uso de otro lenguaje.
- El análisis de la información debe de realizarse en Python.
- El cliente puede hacer uso de cualquier framework web.
- 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 los structs que se utilizarán en el proyecto.
- No se permite la utilización de código descargado desde internet (foros, repositorios, artículos, etc).