UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS

MANEJO E IMPLEMENTACIÓN DE ARCHIVOS
Catedráticos: Ing. Álvaro Díaz Ardavin, Ing. Oscar Paz Campos
Auxiliares: Kevin Esquivel, Denis Vásquez
Primer Semestre 2018

ENUNCIADO DE PRIMER PROYECTO

FASE 1 y 2

INDICE

1. INTRODUCCIÓN

Manejo e Implementación de Archivos es el curso que introduce los conceptos de sistema de archivo y sistema operativo con una fuerte base final en los gestores de base de datos.

El primer proyecto busca abarcar los conceptos principales de la creación de un sistema de archivos y sobre la administración del espacio en un disco duro a nivel lógico para reforzar los conocimientos y aplicación sobre la teoría del particionamiento y administración de espacio además implementando estrategias para la integridad de los datos la cual será principalmente mediante la implementación de RAID por software.

Durante la segunda fase del proyecto se abarcarán específicamente los conocimientos sobre la creación y manejo de archivos mediante la implementación del sistema de archivos EXT2 utilizando todos los conceptos a nivel de lógica implementándose en el lenguaje de programación C++, añadido a esto se aplicaran estrategias de colocación con el fin de aplicar los ordenamientos deseados para cada implementación de disco duro que se realice.

El sistema de archivos EXT2, que inicialmente fue el sistema de archivos de distribuciones de Linux como Debian y Red Hat que sustituyo la implementación de EXT será el objetivo principal de desarrollo del sistema de archivos para el presente enunciado en el cual se aplicarán los conceptos de i-nodos, bloques de datos, arboles de directorio y demás aplicaciones a nivel de apuntadores de dirección en disco duro.

Otra parte muy importante que se debe destacar es que debido a la alta tasa de probabilidad de fragmentación externa que puede existir en los discos al realizar la escritura y eliminación de datos dentro del mismo, durante la segunda fase del presente proyecto el estudiante deberá aplicar de manera lógica, conceptual y práctica la desfragmentación de disco para realizar la optimización del espacio en disco duro.

El presente proyecto corresponde a dos terceras partes del contenido del laboratorio siendo este dividido en dos fases que deberán ser elaboradas por los estudiantes durante la primera parte del laboratorio que corresponde del mes de febrero parcialmente al mes de marzo parcialmente.

2. OBJETIVOS

- Comprender y aplicar los conocimientos sobre el funcionamiento de los sistemas de archivos.
- Aplicar el concepto de particionamiento, formateo, MBR y Gestor de Arranque.
- Aprender y complementar conceptos del lenguaje C y C++.
- Aprender el manejo de distintos tipos de archivos en lenguaje C y C++.
- Manipular el uso de archivos en el entorno GNU/Linux.
- Implementar diferentes estructuras de datos para el manejo de los sistemas de archivos en lenguaje C y C++.
- Comprender el sistema de archivos EXT2
- Aplicar el formateo rápido y completo en una partición
- Realizar diferentes implementaciones y formas de acceso a un archivo, para manejo de información.
- Aplicar y comprender claramente la teoría de particiones, estrategias de colocación.
- Comprender los diferentes tipos de acceso a los archivos y su administración.
- Comprender y aplicar los conocimientos sobre RAID y su aplicación a nivel de software.
- Investigar y aplicar los conocimientos sobre fragmentación de datos en disco duro.
- Generar reportes a través de GraphViz.

3. CONTENIDO GENERAL DEL PROYECTO

El contenido del proyecto estará divido en dos fases de las cuales se presenta a continuación la distribución de contenido para cada una de las mismas:

- **PRIMERA FASE**
  1. CONSOLA DE COMANDOS
  2. ADMINISTRACIÓN DE DISCOS
     2.1. mkdisk
     2.2. rmdisk

2.3. fdisk
2.4. mount
2.5. unmount
2.6. executable

3. REPORTES
   3.1. mbr
   3.2. repdisk
   3.3. exec
4. RAID

- **SEGUNDA FASE**
  5. Todos los comandos (operaciones) que no se incluyen en la fase 1.
  6. Sistema de Archivos EXT2
     6.1. Súper Bloque
     6.2. Bitmap de Inodos
     6.3. Bitmap de Bloques
     6.4. Tabla de inodos
     6.5. Bloque de Carpetas
     6.6. Bloque de Archivos
     6.7. Bloque de Apuntadores
  7. Limitaciones
  8. Otras operaciones
  9. Reportes
     9.1. rep
     9.2. mbr
     9.3. disk inode
     9.4. block
     9.5. bm_inode
     9.6. bm_block
     9.7. tree
     9.8. sb
     9.9. file
  10. Consola de monitoreo de contenido

NOTA: La funcionalidad de la consola de monitoreo de contenido será entregada a partir de inicio de trabajo de la segunda fase.

4. CONSOLA DE COMANDOS

Consola de comandos será una aplicación que utilizará la terminal de Linux para ingreso de texto que simulará ser una consola de ingreso de texto a excepción de los reportes gráficos en Graphviz los cuales se mostrarán como paquetes de imágenes o como una salida de imagen con cualquier visor de imágenes después de ejecutar el comando.

Los comandos tendrán una estructura definida en la cual todos los parámetros serán asignados y el valor por defecto será identificado y el orden siempre será el mismo sin embargo no se distinguirá entre mayúsculas y minúsculas. La ejecución de los comandos por consola será únicamente por

línea. Dada la simplicidad del ingreso de comando en consola **no se podrá** utilizar ninguna herramienta para interpretar las líneas de comando (ejemplo: flex, bison u otros.).

Las líneas de entrada que no sean válidas como parámetros no especificados deberán ser reportadas como error total de línea y el comando no será ejecutado.

Los comandos se podrán ejecutar por medio de archivos de scripts con comandos por línea. No existirán comentarios dentro de los scripts.

NOTA IMPORTANTE:

- La elaboración de la fase 1 y 2 se deberá realizar completamente en lenguaje de programación C con el IDE Code::Blocks y la consola de monitoreo de contenidos será una aplicación en lenguaje C++ desarrollado en el IDE Qt.

5. 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:

5.1. Mkdisk

Este comando creará un archivo binario que simulará un disco duro, estos archivos binarios tendrán la extensión **disk** 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	Descripción
-size	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.
-unit	Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro size. Podrá tener los siguientes valores: **k** que indicará que se utilizarán Kilobytes (1024 bytes) **m** en el que se utilizarán Megabytes (1024 * 1024 bytes)
-path	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:

mkdisk –size=3000 –unit=k –path=/home/user/Disco1.dsk

Línea de comando que realizará la creación de un disco duro en un archivo de texto con un tamaño de 3000kb en la path física descrita.

5.2. Rmdisk

Comando utilizado para realizar la eliminación de un archivo que representa a un disco duro mostrando un mensaje de confirmación para eliminar. 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.

Ejemplos:

rmDisk –path=/home/misdiscos/Disco4.dsk

Línea de comando que elimina el disco duro creado.

5.3. Fdisk

Este comando administra las particiones en el archivo que representa al disco duro. Siempre se utilizará el primer ajuste para buscar espacio dentro del disco y crear la partición.

Nota: Para la creación de particiones se utilizará el primer ajuste para colocar la partición y será distinto al utilizado por el sistema de archivos.

Deberá mostrar un error si no se pudo realizar la operación sobre la partición, especificando por qué razón no pudo crearse (Por espacio, por restricciones de particiones, etc.).

Los parámetros para el comando son los siguientes:

Parámetro	Descripción
-size	Este parámetro recibirá un número que indicará el tamaño de la partición a crear. Debe ser positivo y mayor a cero, si no mostrará un mensaje de error.
-unit	Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro size. Podrá tener los siguientes valores: **B:** que indicará que se utilizarán bytes **K:** que indicará que se utilizarán Kilobytes (1024 bytes) **M:** en el que se utilizarán Megabytes (1024 * 1024 bytes)
-path	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	Indicará que tipo de partición se creará. **P:** en este caso se creará una partición primaria. **E:** en este caso se creará una partición extendida. **L:** Con este valor se creará una partición lógica. Las particiones lógicas solo pueden estar dentro de la extendida sin sobrepasar su tamaño. Para la aplicación de tipos, se debe tener en cuenta las restricciones de teoría de particiones: 1. La suma de primarias y extendidas debe ser como máximo 4. 2. Solo puede haber una partición extendida por disco. 3. No se puede crear una partición lógica si no hay una extendida. Si se utiliza otro valor diferente a los anteriores deberá mostrar un mensaje de error.
-fit	Indicará el ajuste que utilizará la partición. 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)
-delete	Este parámetro indica que se eliminará una partición. Este parámetro se utiliza junto con **-name** y **-path.** Se deberá mostrar un mensaje que permita confirmar la eliminación de dicha partición. Si la partición no existe deberá mostrar error. Si se elimina la partición extendida, deben eliminarse las particiones lógicas que tenga adentro. Recibirá los siguientes valores: **Null:** Indica que no se utilizará la línea de comando para realizar una eliminación de partición en el disco duro **Fast:** Esta opción marca como vacío el espacio en la tabla de particiones. **Full:** Esta opción además marcar como vació el espacio en la tabla de particiones, rellena el espacio con el carácter \0. Si se utiliza otro valor diferente, mostrará un mensaje de error.
-name	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.
-add	Este parámetro se utilizará para agregar o quitar espacio de la partición. Puede ser positivo o negativo y los valores posibles son: 1. Tomará el parámetro **-units** para las unidades a agregar o eliminar. 2. Tomará el valor de **null** si no se desea agregar ningún tipo de espacio a la partición y solo no se tomará en cuenta. Los casos posibles de aplicación son: 1. Si agregar espacio, deberá comprobar que exista espacio libre después de la partición. 2. Si desea quitar espacio se debe comprobar que quede espacio en la partición (no espacio negativo).

Ejemplos:

1. Creación de partición primaria en el disco2
   Fdisk –size=300 -unit=k -path=/home/Disco2.dsk -type=P -fit=BF -delete=null -name=Particion0 -add=null

2. Eliminación en tipo fast de la partición 0 en el disco 2
   Fdisk –size=0 -unit=k -path=/home/Disco2.dsk -type=null -fit=null -delete=fast -name=Particion0 -add=null

3. Agregar espacio en una partición
   Fdisk –size=0 -unit=k -path=/home/Disco2.dsk -type=null -fit=null -delete=null -name=Particion0 -add=500

5.4. Mkfs

El comando Mkfs realizará la aplicación de formato de sistema de archivos EXT2 a una partición específica.

Como consideración especial el estudiante deberá realizar la creación de la carpeta raíz de manera automática (/) ya que sin esto el sistema de archivos no podría conocer el inicio del árbol de directorios.

Dentro de la aplicación de mkfs se deberá tomar en cuenta que el comando no solo aplicará formato de sistema de archivos sino que podrá sustituir uno ya existente considerado este dentro del parámetro type.

Parámetro	Descripción
-path	Este parámetro será la ruta en la que se encuentra el disco en el que se formateará la partición. Este archivo ya debe existir.
-type	Indicará que tipo de formateo se realizará. Ya que es opcional, se tomará como un formateo completo si no se especifica esta opción. Podrá tener los siguientes valores: • **Fast:** en este caso se realizará un formateo rápido. • **Full:** en este caso se realizará un formateo completo (Reinicio de todos los bloques, i-nodos, árbol de directorios y mapas de bits). La diferencia entre estos dos tipos se explicará más adelante.
-name	Indicará el nombre de la partición que se formateará. Si no existe la partición deberá mostrar un error.
-add	Este parámetro se utilizará para aumentar o reducir el tamaño del sistema de archivos (en caso de que el tamaño de la partición hubiese cambiado), el valor puede ser positivo o negativo. • En caso de que sea negativo se debe validar que no se eliminen archivos al momento de reducir la partición. • En el caso de que sea positivo se debe validar que no sobrepase el tamaño de la partición. • **Null:** en caso de que no se quiera hacer ningún tipo de cambio en el tamaño o se esté formateando la partición nuevamente o por primera vez.

-unit	Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro **add**. Podrá tener los siguientes valores: **B:** que indicará que se utilizarán bytes **K:** que indicará que se utilizarán Kilobytes (1024 bytes) **M:** en el que se utilizarán Megabytes (1024 * 1024 bytes)

Ejemplos:

1. Realizará el formateo completo (inicial o de reemplazo según sea el caso) en la partición 1 del Disco1.dsk
   mkfs -path=/home/Disco1.dsk -type=fast -name=Particion1 –add=null –unit=K

2. Agregará (o quitará) espacio al sistema de archivos.
   mkfs -path=/home/Disco1.dsk -type=fast -name=Particion1 –add=200 –unit=K

5.5. Mount

Este comando montará una partición del disco en el sistema. Cada partición se identificará por un id que tendrá la siguiente estructura: vd + letra + NUMERO.

Por ejemplo: vda1, vda2, vdb1, vdc1… La letra será la misma para particiones en el mismo disco y el número diferente para particiones en el mismo disco.

Nota: Para realizar el montado de particiones y su uso el estudiante podrá elegir una estructura de datos de memoria dinámica a elección propia.

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

Ejemplos:

mount -path=/home/Disco1.dsk –name=Part1
Realiza el montaje de la partición Part1 del Disco1 con el nombre vda1

mount -path=/home/Disco2.dsk –name=Part1
Realiza el montaje de la partición Part1 del Disco1 con el nombre vdb1

mount -path=/home/Disco3.dsk –name=Part2
Realiza el montaje de la partición Part1 del Disco1 con el nombre vdc1

mount -path=/home/Disco1.dsk –name=Part2
Realiza el montaje de la partición Part1 del Disco1 con el nombre vda2

5.6. Unmount

El comando unmount realizará el desmontaje de las particiones disponibles en el sistema.

Se utilizará el id que se le asignó a la partición al momento de cargarla. Recibirá los siguientes parámetros.

Parámetro	Descripción
-id	Especifica el id de la partición que se desmontará. Si no existe el id deberá mostrar un error

Ejemplos:

1. **Desmontaje de la partición vda1**
   unmount -id=vda1

5.7. Executable

El programa podrá ejecutar scripts con el comando exec. Debe mostrar el contenido de la línea que está leyendo y su resultado. También debe mostrar los comentarios del script.

Parámetro	Descripción
-path	Especifica el nombre del script que se va a ejecutar.

Ejemplo:

```
#ejecuta el script
executable /home/Desktop/calificacion.sh
```

6. ADMINISTRACIÓN DE CARPETAS Y ARCHIVOS

Estos comandos permitirán crear archivos y carpetas, así como editarlos, copiarlos, moverlos y eliminarlos.

6.1. mkfile

Este comando permitirá crear un archivo. Tendrá los permisos 664. Tendrá los siguientes parámetros:

Parámetro	Descripción
-id	Especifica el id de la partición en la que se creará el archivo. Si no existe debe mostrar error.

-path	Este parámetro será la ruta del archivo que se creará. Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro **–p**, que se explica posteriormente.
-p	El parámetro describirá si se crea o no las carpetas necesarias de la **path** para el árbol de directorio del nuevo archivo. • **Y**: Si las carpetas especificadas no existen, se crearan. • **Null**: Si las carpetas especificadas no existen, no se crearan. Si ya existen, no realizará nada.
-size	Este parámetro indicará el tamaño en bytes del archivo. Los valores que puede tener son: • Valor mayor a cero = se tomará el valor en bytes y el contenido serán números del 0 cuantas veces sea necesario. • 0 = el tamaño del archivo será de 0 bytes. • Si es negativo debe mostrar error.
-cont	Contiene la path de un archivo en el disco duro físico en el cual existe contenido el cual se desea copiar en el nuevo archivo que se quiere crear. Si no existe el archivo de origen de contenido se mostrará un archivo de error y el contenido será “”. Si el valor del parámetro es: null, no tendrá contenido asignado.

Ejemplos:

mkfile –id=Particion1 –path=/home/user/docs/a.txt –p=Y –size=15 –cont=/home/Documents/b.txt

Se crea el archivo a.txt en el sistema de archivos de la partcion1 en la path de su respectivo árbol de directorios.

6.2. cat

Este comando permitirá mostrar el contenido del archivo. Tendrá los siguientes parámetros:

Parámetro	Descripción
-id	Especifica el id de la partición en la que se leerá el archivo. Si no existe debe mostrar error.
-path	Este parámetro será la ruta del archivo que se leerá. Si no existe el archivo debe mostrarse un mensaje de error.

Ejemplos:

Cat -PatH=/home/user/docs/a.txt -Id=vdb1

En la terminal debería mostrar el contenido, ejemplo: >> HOLAMUNDO

6.3. rem

Este comando permitirá eliminar un archivo o carpeta y todo su contenido.

- En el caso de carpetas, eliminará todos los archivos o subcarpetas.

Tendrá los siguientes parámetros:

Parámetro	Descripción
-id	Especifica el id de la partición en la que se eliminará el archivo o carpeta. Si no existe debe mostrar error.
-path	Este parámetro será la ruta del archivo o carpeta que se eliminará. Si no existe debe mostrar error.

Ejemplos:

1. Elimina el archivo a.txt de la partición montada
   rmfile -PatH=/home/user/docs/a.txt -Id=vdb1

2. Elimina la carepta user y todo su contenido (docs, a.txt)
   rmfile -PatH=/home/user -Id=vdb1

6.4. edit

Este comando permitirá editar el contenido de un archivo para que ocupe un tamaño especifico, o bien, para asignarle otro contenido.

Este comando no eliminará y creará nuevamente el archivo sino que modificará el contenido como tal y todos los elementos del sistema de archivos necesarios para dicha operación.

Si no existe debe mostrar error.

Parámetro	Descripción
-id	Especifica el id de la partición en la que se modificará el archivo. Si no existe debe mostrar error.
-path	Este parámetro será la ruta del archivo que se modificará. Si no existe, debe mostrar un mensaje de error.
-size	Este parámetro indicará el nuevo tamaño en bytes del archivo, el contenido serán números del 0 al 9 cuantas veces sea necesario. • Si el valor es **null**, no se modificará el tamaño del archivo. Si es negativo debe mostrar error.
-cont	Indicará un archivo en el disco duro de la computadora que tendrá el nuevo contenido del archivo. Se utilizará para cargar contenido en el archivo. La ruta ingresada debe existir, si no mostrará un mensaje de error.

Ejemplos:

Modifica el archivo a.txt, el tamaño del archivo es de 22 bytes y su contenido sería lo que se encuentre físicamente en la path del parámetro cont.

Edit -id=vdb1 -path=/home/user/docs/a.txt -size=22 –cont=/home/Documents/c.txt

6.5. Rename

Este comando permitirá cambiar el nombre de un archivo o carpeta. Tendrá los siguientes parámetros:

Parámetro	Descripción
-id	Especifica el id de la partición en la que se renombrará el archivo o carpeta. Si no existe debe mostrar error.
-path	Este parámetro será la ruta del archivo o carpeta al que se le cambiará el nombre. Si no existe el archivo o carpeta, debe mostrarse un mensaje de error.
-name	Especificará el nuevo nombre del archivo

Ejemplos:

Rename -PatH=/home/user/docs/holamundo.txt -Id=vdb1 –name=programa.txt

Se cambia el nombre del archivo holamundo.txt a programa.txt

6.6. mkdir

El comando mkdir realiza las operaciones de creación de archivos.

El propietario será el usuario que actualmente ha iniciado sesión. Tendrá los permisos 664. Tendrá los siguientes parámetros:

Parámetro	Categoría	Descripción
-id	Obligatorio	Especifica el id de la partición en la que se creará la carpeta. Si no existe debe mostrar error.
-path	Obligatorio	Este parámetro será la ruta de la carpeta que se creará. Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro **–p**, que se explica posteriormente.
-p	Opcional	El parámetro describirá si se crea o no las carpetas necesarias de la **path** para el árbol de directorio del nuevo archivo. • **Y**: Si las carpetas especificadas no existen, se crearan. • **Null**: Si las carpetas especificadas no existen, no se crearan. Si ya existen, no realizará nada.

Ejemplos:

Crea la carpeta ecys y si no existen las carpetas home user o docs se crean
Mkdir –P=y -id=vda1 -path=/home/user/docs/sistemas/ecys

6.7. cp

Este comando permitirá realizar una copia del archivo o carpeta y todo su contenido hacia otro destino.

Parámetro	Descripción
-id	Especifica el id de la partición en la que está el archivo o carpeta que se quiere copiar. Si no existe, debe mostrar un error.
-path	Este parámetro será la ruta del archivo o carpeta que se desea copiar. Debe copiar todos los archivos y carpetas con todo su contenido. Muestra un error si no existe la ruta.
-dest	Este parámetro será la ruta de la carpeta a la que se copiará el archivo o carpeta. Debe mostrar un mensaje de error si la carpeta no existe.
-iddest	Especifica el id de la partición en la que se copiará el archivo o carpeta. Si no existe debe mostrar un error.

Ejemplos:
Árbol de directorios.

```
>> /
>> |_home
>> |_user
>> | |_archivo0.txt
>> | |_documents
>> |   |_archivo1.txt
>> |   |_archivo2.txt
>> |_images
```

Copia documents a images

```
cp -id=vda2 -Path=/home/user/documents -iddest=vda2 -dest=/home/images
```

La carpeta images será ahora:

```
>> |_images
>> | |_documents
>> |   |_archivo1.txt
>> |   |_archivo2.txt
```

6.8. mv

Este comando moverá un archivo o carpeta y todo su contenido hacia otro destino.

Si el origen y destino están dentro de la misma partición, solo cambiará las referencias, para que ya no tenga el padre origen sino, el padre destino, y que los padres de la carpeta o archivo ya no tengan como hijo a la carpeta o archivo que se movió. Si el origen y destino están en una partición diferente, funcionará como copiar y después eliminar.

Parámetro	Descripción
-id	Especifica el id de la partición en la que está el archivo o carpeta que se quiere mover. Si no existe, debe mostrar un error.
-path	Este parámetro será la ruta del archivo o carpeta que se desea mover.
-dest	Este parámetro será la ruta de carpeta a la que se moverá el archivo o carpeta. Debe mostrar un mensaje de error si la carpeta no existe.
-iddest	Especifica el id de la partición en la que se moverá el archivo o carpeta. Si no existe debe mostrar un error.

Mueve documents a images

```
MV -id=vda1 -iddest=vda1 -Path=/home/user/documents -dest=/home/images
```

6.9. find

Este comando permitirá realizar una búsqueda por el nombre del archivo o carpeta. Permitirá los siguientes caracteres especiales:

Carácter	Descripción
-	Un solo carácter
+	Uno o más caracteres

Recibirá los siguientes parámetros:

Parámetro	Descripción
-id	Especifica el id de la partición en la que se buscará el archivo o carpeta. Si no existe, debe mostrar un error.
-path	Este parámetro será la ruta de la carpeta en el que se inicia la búsqueda, deberá buscar en todo su contenido.
-name	Indica el nombre del archivo o carpeta que se está buscando. Debe aceptar los caracteres especiales definidos anteriormente

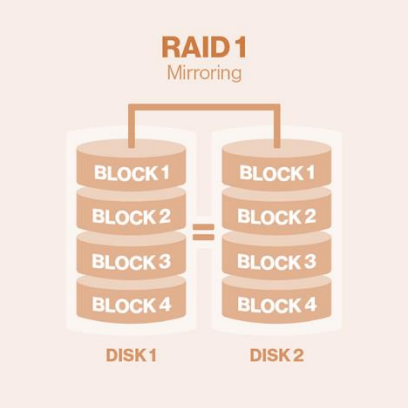
El resultado debe mostrarse en forma de árbol, mostrando todos los archivos encontrados. find * mostrará toda la estructura del sistema de archivos a partir de la carpeta indicada ya que no hay filtro.

Ejemplos:

```
Busca los archivos que tengan una letra como nombre y cualquier extensión
find -id=vda1 -Path=/home -name=?.*
```

7. RAID

Para la primera fase se requiere que el estudiante aplique los conocimientos sobre raid para lo cual se requiere que el alumno desarrolle RAID por software para la creación de discos duros y todo lo que conlleva su administración de manejo. El tipo asignado a desarrollar es un RAID 1 en el cual el estudiante deberá desarrollar lo necesario para poder realizar la administración de discos.



Para realizar la administración de los RAID se realizará los procesos de operaciones en dos fases:

- Fase de operación:
  El RAID realizará las operaciones sobre un disco duro del arreglo de máximo 2 discos para cada elemento creado (disco duro) en el cual se realizarán las operaciones en el disco 0.
- Fase de confirmación
  Las operaciones realizadas se confirmarán en el disco duro 1 de copia de seguridad.

Entre la fase de operación y de confirmación se deberá generar un reporte DISK para confirmar el estado de cada disco antes y después de cada operación.

Para las operaciones EXEC, se ejecutará todo el scrip y después se generará el primer reporte DISK que deberá abrirse automáticamente, después de eso se ejecutará la fase de confirmación y se generará nuevamente el reporte DISK.

8. 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:

- **Disco Duro**

Master Boot Record

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_tamaño	int	Tamaño total del disco en bytes
mbr_fecha_creacion	time	Fecha y hora de creación del disco
mbr_disk_signature	int	Número random, que identifica de forma única a cada disco
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

El contenido de la estructura partition será el siguiente:

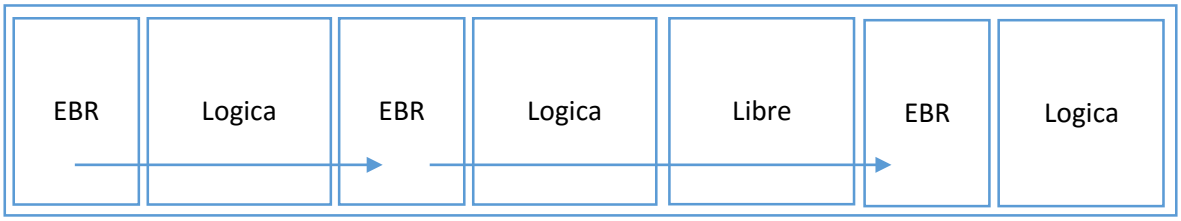
Nombre	Tipo	Descripción
part_status	char	Indica si la partición está activa o no
part_type	char	Indica el tipo de partición, primaria o extendida. Tendrá los valores **P** o **E**
part_fit	char	Tipo de ajuste de la partición. Tendrá los valores **B** (Best), **F** (First) o **W** (worst)
part_start	int	Indica en que byte del disco inicia la partición
part_size	int	Contiene el tamaño total de la partición en bytes.
part_name	char[16]	Nombre de la partición

Extended Boot Record

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 que byte del disco inicia la partición
part_size	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 como la siguiente:



Para crear el 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.

9. Sistemas 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 ceil al resultado:

tamaño_particion = sizeof(superblock) + n + 3 * n + n * sizeof(inodos) + 3 * n * sizeof(block)

numero_estructuras = ceil(n)

Ya que el comando mkfs con el parámetro –add agrega o quita espacio de la partición, se deberán modificar estas estructuras sin eliminar información. Se calcula cuantos bloques se agregarán o

quitarán y se empieza moviendo de derecha a izquierda en el caso de agregar y en el caso de quitar bloques se empieza moviendo de izquierda a derecha para evitar sobrescribir información.

9.1. Súper Bloque

Contiene información sobre la configuración del sistema de archivos. Tendrá los siguientes valores:

Nombre	Tipo	Descripción
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	Ultima fecha en el que el sistema fue montado
s_umtime	time	Ultima 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_size	int	Tamaño del inodo
s_block_size	int	Tamaño del bloque
s_first_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 inico 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 usuar mount, debe actualizar s_mtime, al utilizar unmount actualizará s_umtime, etc.

9.2. Bitmap de Inodos

Indica que inodos están ocupados o libres, siendo los posibles valores 1 o 0.

1 = Inodo ocupado
0 = Inodo libre

Cada valor se guardara como un char.

9.3. Bitmap de Bloques

Indica que bloques están ocupados o libres, siendo los posibles valores 1 o 0.

1 = Bloque ocupado
0 = Bloque libre

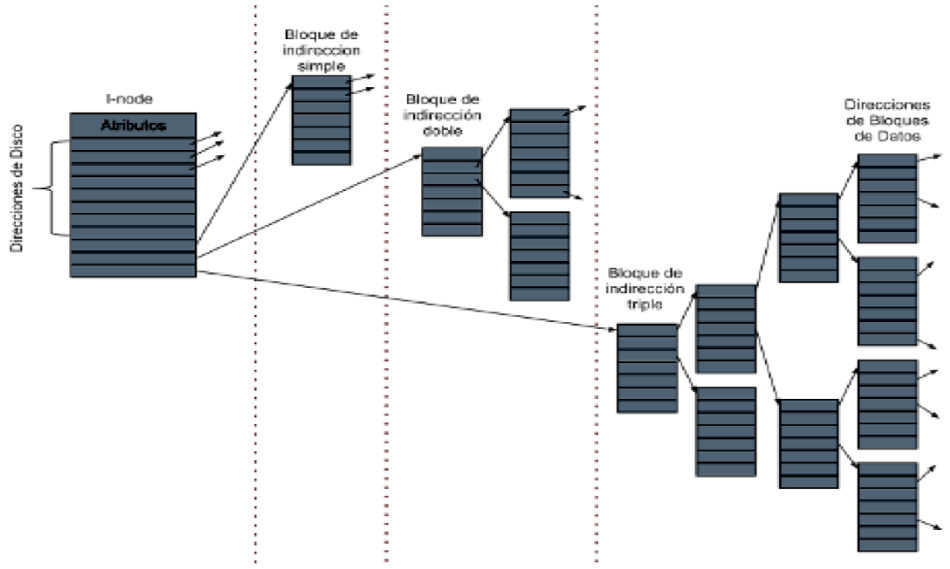
Cada valor se guardara como un char.

9.4. 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, siempre tendrá el valor 1
i_size	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	Úlitma fecha en la que se modificó el inodo
i_block	int[15]	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 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. Será 664 para todos los archivos o carpetas

La siguiente imagen muestra el funcionamiento de los bloques indirectos.



9.5. Bloque de Carpetas

Esta estructura estará asociada a un inodo de carpetas. Aquí se guardará la información sobre el nombre de los archivos que contiene y a que inodo apuntan. La estructura es la siguiente:

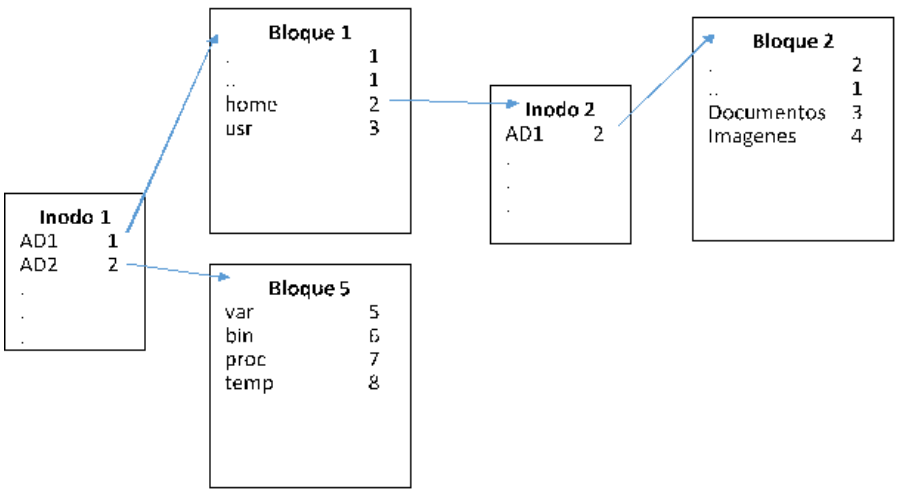
Nombre	Tipo	Descripción
b_content	content[4]	Array con el contenido de la carpeta

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) = 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.

Ejemplos:



9.6. Bloque 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.

9.7. Bloque de Apuntadores

Estos bloques se utilizarán para los apuntadores indirectos (simples, dobles y triples). Su estructura es la siguiente:

Nombre	Tipo	Descripción
b_pointers	int[16]	Array con los apuntadores hacia bloques (de archivo o carpeta)

Su tamaño será de 16 * 4 = 64, igual que los otros dos bloques anteriores.

El tipo de bloque que debe leer o utilizarse se puede determinar según el tipo de inodo (archivo o carpeta) y en base a que apuntador esté utilizando (directo, simple, doble o triple indirecto)

9.8. 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.

numero_bloques_por_inodo = 12 + 16^1 + 16^2 + 16^3 = 12 + 16 + 256 + 4096 = 4380 bloques

Cada bloque de carpeta tiene una capacidad de 4 hijos. Por lo que su capacidad es de 17518 carpetas o archivos dentro de una carpeta.

capacidad_carpeta = 4380 * 4 - 2= 17518

Cada bloque de archivo tiene una capacidad de 64 bytes. Por lo que el tamaño máximo de un archivo es aproximadamente de 273 Kilobytes.

capacidad_archivo = 4380 * 64 = 280320 bytes = 273 Kilobytes

Al formatear se debe crear la carpeta raíz (/).

9.9. Otras operaciones

Aquí se aclararán 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 ya utilizados. 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 el comando edit, modificará la fecha de modificación, el comando cat la fecha de lectura, etc.

10. Reportes

Los reportes serán divididos en gráficos, de salida gráfica y de texto los cuales son de carácter **OBLIGATORIO** para realizar la calificación, de lo contrario no se podrá realizar la calificación y conocer si el sistema funciona.

Se deberán generar los reportes con el comando rep. Este comando no necesita tener una sesión activa. 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. Por ejemplo si no hace reporte de i-nodos, no tendrá derecho a la calificación de todos los aspectos relativos a los i-nodos, ya que no se puede comprobar que el estudiante haya implementado dicha funcionalidad.

10.1. rep

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

Parámetro	Descripción
-name	Nombre del reporte a generar. Tendrá los siguientes valores: • mbr • disk • inode • block • bm_inode • bm_block • tree • sb • file Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error.
-path	Indica una carpeta y el nombre que tendrá el reporte. Si no existe la carpeta, deberá crearla.
-id	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.
-ruta	Funcionará para el reporte file. Será el nombre del archivo o carpeta del que se mostrará el reporte. Si no existe muestra error.

Ejemplos

rep -id=vda1 -Path=/home/user/reports/reporte1.jpg -name=mbr
rep -id=vda2 -Path=/home/user/reports/reporte2.pdf -name=disk
rep -id=vda1 -Path=/home/user/reports/reporte3.jpg -name=tree

10.2. mbr

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

Ejemplo:

MBR Disco1.dsk

Nombre	Valor
mbr_tamaño	10485760
mbr_fecha_creacion	30/11/2015 13:45
mbr_disk_signature	16684811
part_status_1	1
part_type_1	**P**
part_fit_1	**B**
part_start_1	300
part_size_1	1048576
part_name_1	Particion 1
part_status_2	1
part_type_2	**E**
part_fit_2	**W**
part_start_2	1048876
part_size_2	1048576
part_name_2	Particion 2

EBR_1

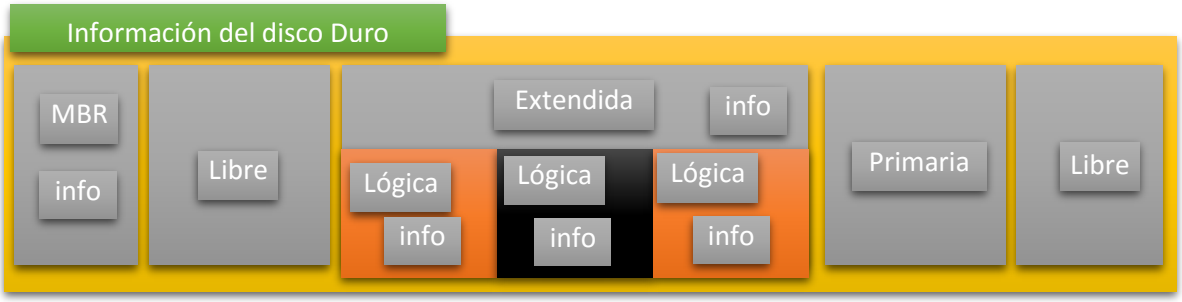
Nombre	Valor
part_status_1	1
part_fit_1	**F**
part_start_1	1048876
part_size_1	524438
part_next_1	1573314
part_name_1	Logica 1

EBR_2

Nombre	Valor
part_status_1	1
part_fit_1	**B**
part_start_1	1573514
part_size_1	524438
part_next_1	-1
part_name_1	Logica 2

10.3. disk

El comando DISK presentará gráficamente por medio de GRAPHVIZ una tabla con la estructura del disco duro tanto a nivel de partición primaria como de extendida siendo su principal propósito el mostrar gráficamente y a detalle los discos creados.
Todos los espacios en donde se indica info se deberá colocar la información de dicho componente o característica.

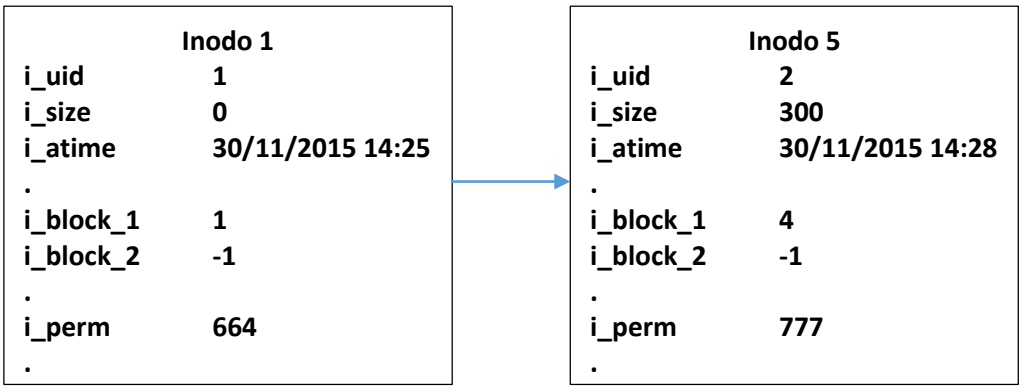


La información presentada en este reporte es a consideración del estudiante pero será tomada en cuenta la presentación y la cantidad de información proporcionada en el mismo.

Ejemplo de comando rep para reporte de disk:
rep -id:vda1 -Path:/home/user/reports/reporte1.jpg -name:disk

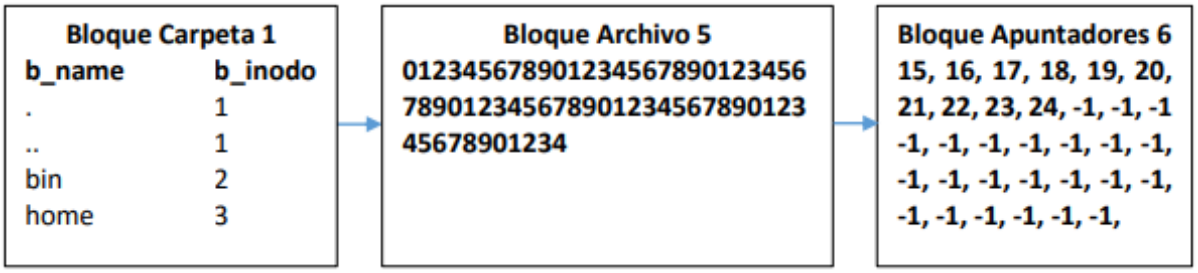
10.4. inode

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



10.5. block

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



10.6. bm_inode

Este reporte mostrará la información del bit map de inodos, mostará todos los bits, libres o utilizados.

Ejemplo:

Bitmap de Inodos

Posición	Estado
1	1
2	1
3	1
4	0
5	0
6	1
7	1
8	1
9	0
10	0
.	.
.	.
.	.

10.7. bm_block

Este reporte mostrará la información del bit map de inodos, mostará todos los bits, libres o utilizados de la siguiente forma:

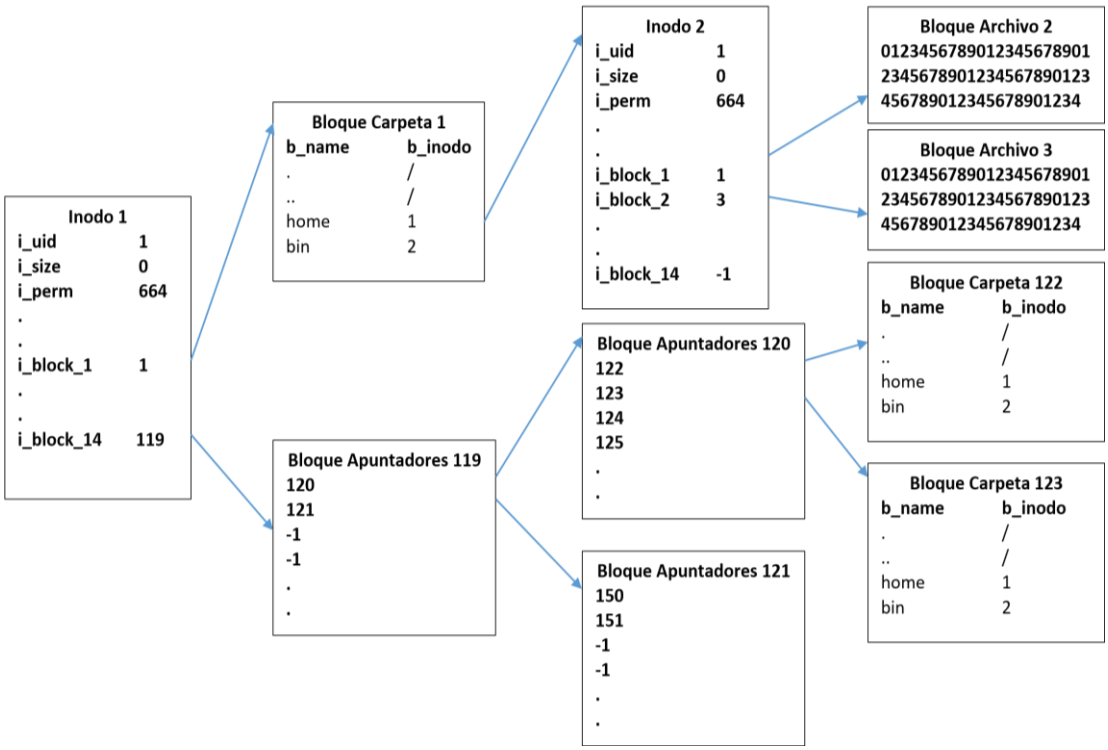
Ejemplo:

Bitmap bloques:

Posición	Estado
1	1
2	1
3	1
4	1
5	0
6	1
7	1
8	0
9	0
10	1
.	.
.	.
.	.

10.8. tree

Este reporte genera el árbol de **todo** el sistema ext3. 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 (En este ejemplo no se ponen todos los datos, bloques y flechas por falta de espacio).



10.9. sb

Muestra **toda** la información del superbloque en una tabla.

Ejemplo:

SuperBloque Partición 1 en Disco1.dsk

Nombre	Valor
s_inodes_count	200
s_blocks_count	600
s_free_blocks_count	10
s_free_inodes_count	100
s_mtime	30/11/2015 15:38
s_umtime	30/11/2015 15:38
s_mnt_count	4
s_magic	0xEF53
s_inode_size	128
s_block_size	64
s_first_ino	50
s_first_blo	180
s_bm_inode_start	128
s_bm_block_start	328
s_inode_start	630
s_block_start	15852

10.10. file

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

Ejemplo:

a.txt

01234567890123456

11. Manuales y código fuente

Para comprender el proyecto, se deberá documentar el proyecto con dos manuales y documentar cada método del código con una pequeña descripción sobre lo que realiza y cómo lo realiza.

11.1. Manual de usuario

Deberá explicar para que sirven los comandos del sistema y mostrar screenshots que demuestren como funciona cada comando. También deberá mostrar cómo generar reportes y que significa el contenido que tienen los reportes.

11.2. Manual técnico

Deberán crearse diagramas de flujo para los comandos de la aplicación, mostrando el algoritmo utilizado para ejecutar la operación y explicando dicho algoritmo.

12. INSTRUCCIÓN DE ENTREGA

12.1. Entrega Fase 1.

Fecha de entrega: viernes 23 de febrero de 2018 antes de las 23:59 pm.

La entrega se realizará a los siguientes links de solicitud de datos en Dropbox:

- Sección impar:
  https://www.dropbox.com/request/3uyuKtCRAA2aqLD82Wvg
- Sección par:
  https://www.dropbox.com/request/45lsTjsXgX2pfpP9XtPQ

El archivo a entregar deberá ser un comprimido con el nombre:

[MIA]Fase1_#carnet.zip en extensión .zip o .rar

El envío de los archivos es responsabilidad únicamente del alumno y el link será inhabilitado a la hora exacta por lo cual entregas tarde no tendrán derecho a calificación y no se aceptarán entregas tarde por ningún otro medio.

Día de calificación: sábado 24 de febrero de 2018 en el área de columnas.

12.2. Entrega Fase 2.

Fecha de entrega: viernes 23 de marzo de 2018 antes de las 23:59 pm.

La entrega se realizará a los siguientes links de solicitud de datos en Dropbox:

- Sección impar:
  https://www.dropbox.com/request/yAnsdKnbQ9YpouU3U21L
- Sección par:
  https://www.dropbox.com/request/ivt1ajBM9rjLYkK8sqny

El archivo a entregar deberá ser un comprimido con el nombre:

[MIA]Fase2_#carnet.zip en extensión .zip o .rar

El envío de los archivos es responsabilidad únicamente del alumno y el link será inhabilitado a la hora exacta por lo cual entregas tarde no tendrán derecho a calificación y no se aceptarán entregas tarde por ningún otro medio.

Día de calificación: sábado 24 de marzo de 2018 en el área de columnas.

Requisitos mínimos para tener derecho a calificación de esta segunda fase:

Se requiere que tengan como mínimo lo siguiente para poder realizar la calificación:

- Comandos
- Reportes
- Particiones (primarias, lógicas y extendidas)
- Mount y Unmount
- Crear carpetas y archivos

Consideraciones a tomar en cuenta:

- Forma de calificación: presencial y según asignación de horario que se publicará.
- El proyecto debe realizarse de forma individual, copias tendrán una nota de 0 y serán reportadas a la escuela.
- Las dudas se responderán en el grupo y aplicarán a todos los estudiantes.
- El lenguaje a utilizar es C y C++ para la consola de monitoreo de contenido.
- Solo se calificará sobre una instalación física de una distribución GNU/Linux. De no ser así se penalizará la nota obtenida.
- No se permite la modificación de código durante la calificación. El estudiante no tendrá acceso al código fuente durante la calificación.
- El archivo binario que representa a los discos no debe crecer.
- No se permite la utilización de estructuras en memoria (listas, arboles, etc.) para el manejo de los archivos o carpetas únicamente en las partes donde se especifique dentro del enunciado (montaje de particiones).
- **La elaboración del proyecto deberá ser realizada en el lenguaje de programación C IDE Code::Blocks y la consola de monitoreo de contenido será desarrollada en lenguaje C++ en el IDE Qt.**