Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Manejo e implementación de archivos
Ing. Oscar Alejandro Paz Campos
Tutor de curso: Monica Raquel Calderon Muñoz

PROYECTO 1

INTRODUCCIÓN

El curso de Manejo e Implementación de Archivos busca que el estudiante comprenda los conceptos sobre la administración de la información en los diferentes dispositivos, tanto en hardware como software, sistemas de archivos, particiones, entre otros conceptos. Es debido a esto que en la primera fase de proyecto consiste en la simulación de un RAID tipo 1, dentro del sistema se podrán simular una serie de particiones de acuerdo a la teoría, dichos sistemas serán administrados por un MBR y un gestor de arranque que serán los encargados de la carga del disco duro y de las particiones generadas según lo que se genere en la aplicación de comandos.

Los sistemas de archivos tendrán aspectos similares a los sistemas de archivos de Windows y GNU/Linux. Aplicando conocimientos adquiridos en el curso y el laboratorio de Manejo e implementación de Archivos.

OBJETIVOS ESPECÍFICOS

- Comprender el funcionamiento de los sistemas de discos RAID
- Aplicar y comprender claramente la teoría de particiones (primaria, lógica y extendida).
- Aprender a administrar archivos y escribir estructuras en C
- Comprender y aplicar la teoría dada en clase sobre los sistemas de archivos.
- Realizar diferentes implementaciones y forma de acceso a un archivo, para manejo de información.
- Generar reportes del estado del disco duro y la información que se almacena.
- Aplicar el formato rápido y completo en una partición
- Utilizar GraphViz y archivos de texto para mostrar reportes de estado del disco duro.

DESCRIPCIÓN GENERAL

RAID 1 es una de las mejores configuraciones en cuanto a redundancia y tolerancia a fallos. Esta configuración, también conocida como “espejo” o “mirroring”, permite duplicar la información en dos discos; es decir, el sistema verá un único volumen de almacenamiento que, en realidad, está formado por dos discos iguales en los que se escriben los mismos datos. De esta forma, si un disco se estropea, el sistema seguirá funcionando y trabajando con el disco que aún funciona.

La aplicación debe de ser capaz de simular el control y manejo de un RAID tipo 1 creando dos discos duros simulados en dos archivos binarios, los cuales podrán ser particionados según la teoría de particiones. Estos sistemas serán administrados a través de un gestor de arranque. Así mismo deberá poder crear y eliminar archivos con información y con ello simular las distintas estrategias de colocación en un disco duro.

La aplicación tiene como funcionamiento teorías principales:

- Teoría de Particiones.
- Manejo de RAID, redundancia de datos.

Esta aplicación es totalmente en consola desarrollada en lenguaje C por lo que no tendrá interfaz gráfica.

DESCRIPCIÓN GENERAL

La aplicación será totalmente en consola, a excepción de los reportes en Graphviz.

Consta únicamente de una consola en donde el usuario podrá ingresar comandos que le servirán para la realización de las diferentes acciones permitidas.

Para el correcto funcionamiento de la consola se posen los siguientes requerimientos y restricciones por considerar:

- No se distinguirá entre mayúsculas y minúsculas.
- Los parámetros obligatorios se identifican con el símbolo de dólar ($)
- Los parámetros opcionales se identifican con el símbolo de arroba (@)
- Solamente se puede colocar un comando por línea.
- Al utilizar un parámetro que no se encuentre especificado, debe mostrar un mensaje de error.
- Los parámetros pueden venir en cualquier orden.
- Los espacios en blanco son utilizados para separar cada parámetro
- Es posible ejecutar archivos de scripts que contengan los comandos.
- En los archivos scripts pueden haber comentarios; Estos comenzarán con /* y finalizan con **/*, serán únicamente de una línea.

No se permite utilizar herramientas para interpretar estos comandos (flex, jison, bison entre otros).

COMANDOS

ADMINISTRACIÓN DE DISCOS

Se tendrá con una serie de comandos que podrán modificar cualquier característica lógica del sistema del disco, con estos comandos se realizará la creación de archivos, los cuales simulan el disco duro.
Estos comandos deben estar disponibles desde que se inicia el programa.

MKDISK

Este comando crea 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**. Sus parámetros son:

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. • El tamaño del mismo deberá ser un múltiplo de 8, ej. 8, 16, 32, 40, 48, 56, 64,72 etc. (Mb=Tamaño del Archivo Binario, simulando un Disco Duro Virtual). Si el valor no es múltiplo de 8 debe mostrar error y no podrá generar el disco.
$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. Tome en cuenta que la aplicación debe tener los permisos necesarios para la generación de carpetas en cualquier ruta. *NOTA: La dirección de la carpeta llevará comillas dobles (“) siempre:*
$name	Obligatorio	• Este parámetro será el nombre del disco con extensión dsk. De no contener la extensión debe mostrar un mensaje de error. • El nombre únicamente puede estar compuesto por letras, números y el único símbolo aceptado será “_” para unir caracteres, como por ejemplo: Disco_1.dsk *NOTA: este parámetro NO se encerrará entre comillas de ningún tipo*.

EJEMPLO

```
/*Crea un disco de 3000 Kb en la carpeta home*/
        Mkdisk $size=>32 $path=>”/home/user/” $name=>Disco1.dsk

/*Se crearán carpetas si no existen*/
        mkdisk $SiZe=>8 $pAth=>”/home/mis discos/DISCO Prueba/” $namE=>Disco_3.dsk
```

RMDISK

Este parámetro elimina un archivo que representa a un disco duro mostrando un mensaje de confirmación para eliminar, sí el usuario acepta realizar el cambio el disco duro debe ser eliminado.
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 Disco_4.dsk*/
        rmDisk $path=>"/home/mis discos/Disco_4.dsk"
```

FDISK

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

Se debe de mostrar un error si no es posible realizar la operación solicitada sobre la partición, especificando por qué razón no pudo crearse, debe incluirse entre los errores dado sea el caso el comando posea parámetros no correspondientes a él.

IMPORTANTE: Este primer ajuste solo es para buscar espacio para crear la partición, es diferente del ajuste que utilizará cada partición para crear los archivos que se utilizarán más adelante en el proyecto.

Parámetro	Categoría	Descripción
$size	Obligatorio	Recibe un número que indicará el tamaño de la partición a crear. Debe ser positivo y mayor a cero, caso contrario, debe mostrar un mensaje de error.
@unit	Opcional	Recibe una letra para indicar las unidades del parámetro size. Los valores que puede poseer son: ● 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) Por ser un parámetro opcional, si no se indica, usará Kilobytes por defecto. Al utilizar un valor diferente a los indicados mostrará un mensaje de error.
$path	Obligatorio	Indica 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. *NOTA: La dirección de la carpeta siempre llevará comillas dobles (“).* *Ejemplos:* *“/home/user/documents/”* *“/home/user/documents/Monica/Mis Discos/”*
@type	Opcional	Indica el tipo de partición a crear. Por ser opcional, el valor por defecto a tomar es primaria . Contendrá los valores: ● P: Crea una partición primaria. ● E: Crea una partición extendida. ● L: Crea una partición lógica. *Nota: 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.* Si se utiliza otro valor diferente a los anteriores deberá mostrar un mensaje de error.
@fit	Opcional	Indica el ajuste que utilizará la partición para asignar espacio. Contendrá los valores: ● BF: Mejor ajuste o Best Fit ● FF: Primer ajuste o First Fit ● WF: Peor ajuste o Worst Fit Por ser opcional, el peor ajuste será el valor tomado por defecto. Si se utiliza otro valor que no sea alguno de los anteriores mostrará un mensaje de error.
@delete	Opcional	Indica la eliminación de una partición. Tome en cuenta que este parámetro se utiliza junto con $name y $path y debe de mostrar un mensaje que permita confirmar la eliminación de dicha partición. Recibirá los siguientes valores: ● Fast: Marca como vacío el espacio en la tabla de particiones. ● Full: Además marcar como vacío el espacio en la tabla de particiones, rellena el espacio con el carácter \0. Si se utiliza otro valor diferente o bien la partición no existe, mostrará un mensaje de error. Recuerde que, al eliminar la partición extendida, deben eliminarse las particiones lógicas que tenga adentro.

$name	Obligatorio	Indica el nombre de la partición, dicho nombre no debe repetirse dentro de las particiones de cada disco. Considere que, si se va a eliminar, la partición ya debe existir, si no existe debe mostrar un mensaje de error.
@add	Opcional	Agrega o quita espacio de la partición y puede ser positivo o negativo. Tomará el parámetro @UNIT para las unidades a agregar o eliminar. Tome en cuenta que, en el caso de agregar espacio, debe comprobar que exista espacio libre después de la partición mientras que, en el caso de quitar espacio se debe comprobar que quede espacio en la partición **(no espacio negativo).**
@mov	Opcional	Mueve la partición indicada, media vez haya espacio suficiente disponible dentro del sistema para poder hacer el cambio.

EJEMPLO:

```
/*Crea una partición primaria llamada Particion1 de 72 kb*/
/*con el peor ajuste y con asignación indexada en el disco Disco1.dsk*/
        fdisk $sizE=>72 $path=>/home/Disco1.dsk $name=>Particion1

/*Crea una partición extendida dentro de Disco2 de 56 kb*/
/*Tiene el peor ajuste y asignación Enlazada*/
        fdisk @TyPE=>E $path=>"/home/Disco2.dsk" @Unit=>K $name=>Particion2
        $sizE=>56

 /*Crea una partición lógica con el mejor ajuste, llamada Particion3 de 1 Mb en el Disco3 */
/*asignación contigua*/
        fdisk $sizE=>1 @tipo=>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 @tipo=>E $path=>"/home/Disco2.dsk" $name=>Part3  @Unit=>K $sizE=>200

/*Elimina de forma rápida una partición llamada Particion1*/
        fdisk @delete=>fast $name=>"Particion1"\ $path=>"/home/Disco1.dsk"

/*Elimina de forma completa una partición llamada Particion1*/
        fdisk $name=>"Particion1"  @delete=>full \ $path=>"/home/Disco1.dsk"

 /*Agrega 1 Mb a la partición Particion4 del Disco4.dsk*/
/*Se debe validar que haya espacio libre después de la partición*/
        fdisk @add=>1 @unit=>M –path=>"/home/mis discos/Disco4.dsk"
        $name=>"Particion 4"
```

MOUNT

Permite montar 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.*

La letra debe ser la misma para particiones en el mismo disco y el número diferente para particiones en el mismo disco.

Si únicamente viene la palabra reservada MOUNT deberá mostrar un listado de particiones montadas.

Parámetro	Categoría	Descripción
$path	Obligatorio	Ruta en la que se encuentra el disco que se montará en el sistema. Tome en cuenta que ***este archivo ya debe existir.***
$name	Obligatorio	Nombre de la partición a cargar. Si no existe debe mostrar error.

EJEMPLO

```
/*Monta las particiones de Disco1.dsk*/
    mount $path=>”/home/Disco1.dsk” $name=>Part1 /*id=>vda1 */ mount
    $path=>”/home/Disco2.dsk” $name=>Part1 /*id=>vdb1 */ mount
    $path=>”/home/Disco3.dsk” $name=>Part2 /*id=>vdc1 */ mount
    $path=>”/home/Disco1.dsk” $name=>Part2 /*id=>vda2*/

 /*Si se coloca el comando mount sin parametros mostrará en la consola las particiones montadas*/
    id=>vda1 $path=>"/home/Disco1.dsk" $name=>Part1
    id=>vdb1 $path=>"/home/Disco2.dsk" $name=>Part1
    id=>vdc1 $path=>"/home/Disco3.dsk" $name=>Part2
    id=>vda2 $path=>"/home/Disco1.dsk" $name=>Part2
```

UNMOUNT

Desmonta una partición del sistema. Se utilizará el id que se le asignó a la partición al momento de cargarla. Debe poder desmontar por listado de particiones.
Los parámetros esperados son

Parámetro	Categoría	Descripción
$id#	Obligatorio	Especifica una lista de id de las particiones que serán desmontadas. Tome en cuenta que de no existir algún id debe mostrar un error.

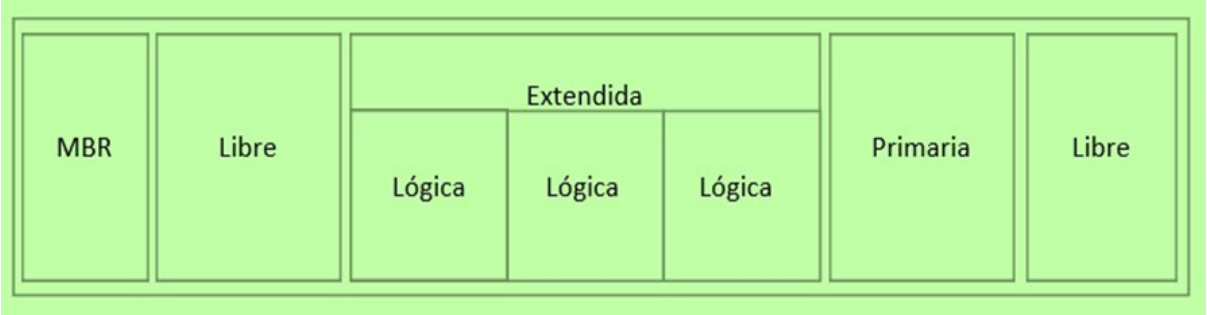
EJEMPLO:

```
/*Desmonta la partición con id vda1 (En Disco1.dsk) */
        unmount $id1=>vda1

/*Si no existe, se debe mostrar error*/
        unmount $id1=>vdx1

/*Desmonta una lista de particiones.*/
        unmount $id1=>vda1 $id2=>vdb2 $id3=>vdc1
```

ESTRUCTURAS DE DISCOS

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

Ejemplo de bloques en un disco con particiones en el que ya se ha eliminado una partición



MASTER BOOT RECORD (MBR)

Cuando se crea una partición debe utilizarse el primer ajuste para crearla.
El MBR tendrá los 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 aleatorio 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

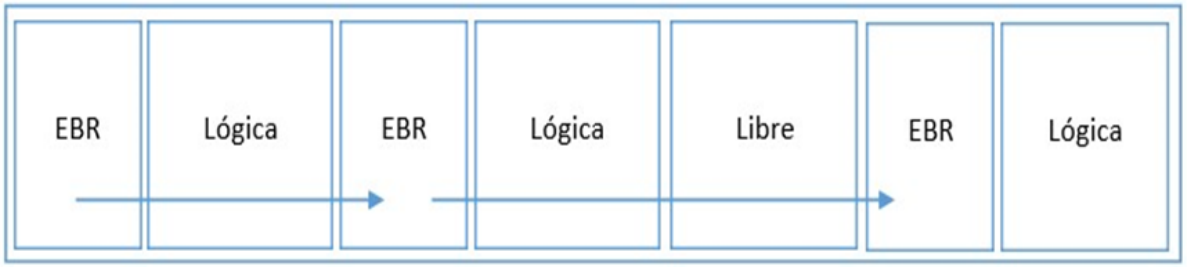
Nombre	Tipo	Descripción
part_status	char	Indica si la partición está activa o no.
part_type	char	Indica si la partición es primaria o extendida con los valores P o E.
part_fit	char	Tipo de ajuste de la partición. Tendrá los valores ● BF (Best fit) ● FF (First Fit) ● WF (Worst fit)
part_start	int	Indica el byte del disco en donde inicia la partición
part_size	int	Contiene el tamaño total de la partición en bytes
part_name	char[16]	Contiene el 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_fit	char	Tipo de ajuste de la partición. Tendrá los valores ● BF (Best fit) ● FF (First Fit) ● WF (Worst fit)
part_start	int	Indica el byte del disco en donde inicia la partición
part_size	int	Contiene el tamaño total de la partición en bytes
part_next	int	Byte que en el que está el próximo EBR. Tendrá el valor de $1 si no hay siguiente
part_name	char[16]	Contiene el nombre de la partición

La estructura lógica de la partición extendida es como la siguiente imagen.



Tome en cuenta que:

- EBR inicial siempre debe existir, aunque se elimine la primera partición.
- 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.

NOTA: Al estarse simulando una configuración del nivel RAID 1, si se elimina el disco u ocurre algún problema en el y la copia espejo está disponible, se debe poder tener acceso a la información del mismo como si se estuviera utilizando el disco original sin que sea visible para el usuario el cambio.

MKFS

Comando para dar un formato completo a la partición. Más adelante se indicará el tipo de archivos que se estará trabajando.

Parámetro	Categoría	Descripción
$id	Obligatorio	Indica el id que se generó con el comando mount de la fase anterior. Sirve para saber la partición y el disco que se utilizará para hacer el sistema de archivos. Tome en cuenta que si no existe debe mostrar un mensaje de error.
@type	Opcional	Indica que tipo de formato se realizará. Por ser opcional, se tomará como un formato completo si no se especifica esta opción. Podrá tener los siguientes valores: ● Fast: Formato rápido. ● Full: Formato completo.
@add	Opcional	Aumenta o reduce el tamaño del sistema de archivos, el valor puede ser positivo o negativo. Tome en cuenta que, si es negativo se debe validar que no se elimine información dentro del sistema al momento de reducir el sistema de archivos. En el caso de que sea positivo se debe validar que no sobrepase el tamaño de la partición.
@unit	Opcional	Recibe una letra que indicará las unidades que utilizará el parámetro @add. Podrá tener los siguientes valores: ● B: Bytes ● K: Kilobytes (1024 bytes) ● M: Megabytes (1024 * 1024 bytes) Si el parámetro no es indicado, se utilizará Kilobytes como el valor por default.

```
/*Se realizará el formato de la partición 1 del disco 1 al sistema de archivos EC*/
        MkfS $id=>vda1 @type=>fast @FS=>EC

 /*Se agregarán 30k al sistema de archivos*/
        MkfS $id=>vda1 @add=>30 @unit=>k
```

SISTEMA DE ARCHIVOS EXT3

Sistema de archivos con registro por diario (journaling). A continuación, se enuncian las estructuras del sistema. Estas estructuras deben de ser implementadas tal y como son especificadas en el apartado.

ESTRUCTURA DEL SISTEMA DE ARCHIVOS EXT3

El número de bloques será el triple que el número de inodos. El número de Journaling, 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 + n*Sizeof(Journaling) + 3 * n + n * sizeof(inodos) + 3 * n * Sizeof(block)

numero_estructuras = floor(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 cuántos bloques se agregan o quitan 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.

SÚPER BLOQUE

Contiene información sobre la configuración del sistema de archivos. Posee los valores descritos a continuación.

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_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 usar mount, debe actualizar s_mtime, al utilizar umount actualizará s_umtime, etc.

JOURNALING

Un sistema con journaling es un sistema de ficheros tolerante a fallos en el cual la integridad de los datos está asegurada porque las modificaciones de la meta-información de los ficheros son primero grabadas en un registro cronológico (log o journal, que simplemente es una lista de transacciones) antes que los bloques originales sean modificados. En el caso de un fallo del sistema, un sistema con journaling asegura que la consistencia del sistema de ficheros es recuperada. El método más común es el de grabar previamente cualquier modificación de la meta-información en un área especial del disco, el sistema realmente grabará los datos una vez que la actualización de los registros haya sido completada.

El journaling maneja todas las transacciones que realiza el sistema de Archivos EXT3 (respaldo para recuperación), la cual maneja la siguiente información.

Dato en la estructura	Descripción
Journal_tipo_operacion	El tipo de operación a realizarse
Journal_tipo	Si es archivo (0), si es directorio(1)
Journal_nombre	Nombre archivo o directorio
Journal_contenido	Si hay datos contenidos

Journal_fecha	Fecha de transacción
Journal_propietario	Es el propietario del archivo o directorio
Journal_Permisos	Son los permisos que tiene el archivo o directorio

BITMAP DE INODOS

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

- 1 = Ocupado
- 0 = Libre

Cada valor se guardará como un char.

BITMAP DE BLOQUES

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

- 1 = Ocupado
- 0 = Libre

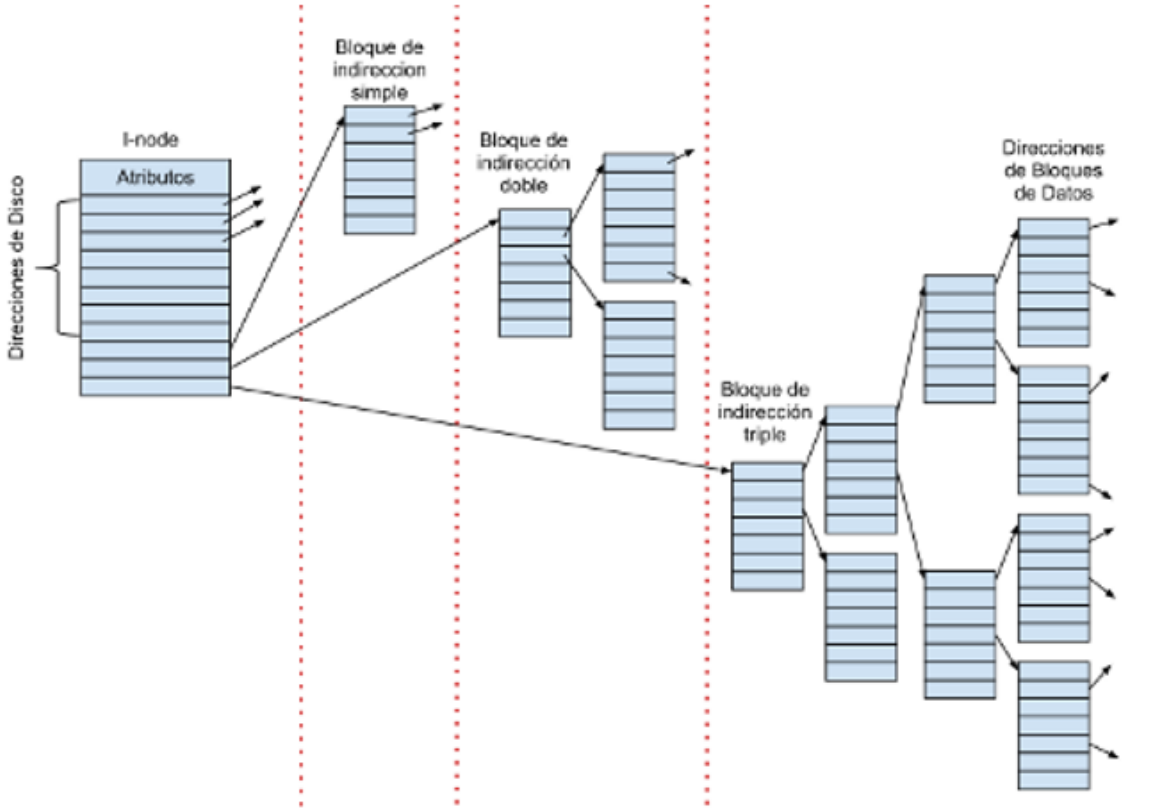
Cada valor se guardará como un char.

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_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	Última fecha en la que de modificación del 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

La siguiente imagen muestra el funcionamiento de los bloques indirectos.



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

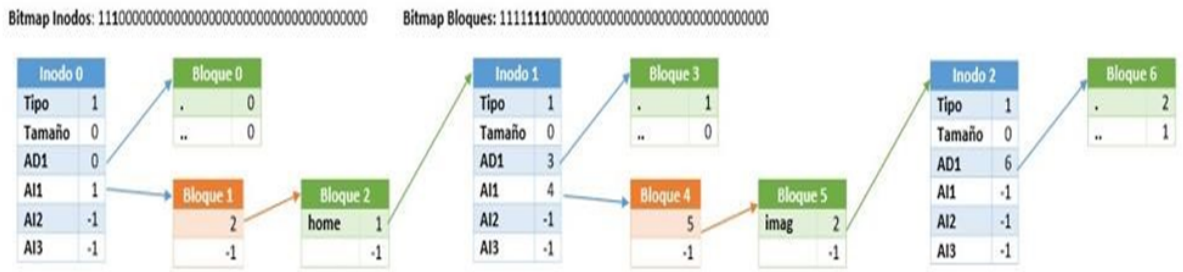
La estructura content será como la siguiente:

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.

EJEMPLO

Por motivos de ejemplo se usaron bloques de carpeta (color verde) con capacidad de 2 items, bloques de apuntadores (color anaranjado) con capacidad de 2 apuntadores.



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

BLOQUE DE APUNTADORES

Estos bloques se utilizarán para los apuntadores indirectos (simples, dobles y triples).

La 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 qué apuntador esté utilizando (directo, simple, doble o triple indirecto)

LIMITACIONES

Limitantes que poseerá el sistema de archivos

- 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

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

OTRAS OPERACIONES

Apartado de aclaración de algunas otras operaciones que se deben realizar. Por ejemplo, que se utilizará el ajuste de la partición para buscar bloques libres y contiguos, correspondiente 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 se están utilizando. 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 bitmap de inodos y bloques. El full aplica un formateo fast y además limpia los bloques e inodos. Siempre debe existir la carpeta raíz y el archivo de usuarios users.txt en la raíz.

ADMINISTRACIÓN DE ARCHIVOS

MKFILE

Comando para la creación de un archivo. Sus parámetros son:

Parámetro	Categoría	Descripción
$id	Obligatorio	Especifica el id de la partición en la que se creará el archivo. Si no existe debe mostrar error.
$path	Obligatorio	Este parámetro será la ruta del archivo que se creará. Si ya existe debe sobreescribir el archivo. Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro @p, que se explica posteriormente.
@p	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. 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, si no mostrará un mensaje de error.

EJEMPLO

```
/*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 $id=>vdb1 $PatH=>"/home/user/docs/a.txt" @p

/*Crea "archivo 1.txt" la carpeta "mis documentos" ya debe existir*/
/*el tamaño es de 0 bytes*/
        mkfile $id=>vda1 $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 $id=>vda1 $path=>”/home/user/docs/b.txt" @p
        @cont=>"/home/Documents/b.txt"
```

Para almacenar el archivo la aplicación deberá dividirlo en “n” caracteres el cual es el tamaño de dicho bloque, para conocer la cantidad de bloques que se requieren para almacenarlos, luego se buscarán bloques de datos “CONTIGUOS” libres y de acuerdo al ajuste que se esté trabajando se almacenará el archivo en los bloques contiguos libres que se encontraron.
Si al momento de crear un archivo, la partición donde se quiere almacenar está llena, entonces mostrará un error que indique que esa partición está llena.

SCRIPT

Son archivos con los comandos definidos en este documento. También puede haber comentarios y líneas en blanco. Serán de extensión sh y serán ejecutados a través del comando exec

EXEC

Permite la ejecución de los scripts. Debe de mostrar el contenido de la línea que está leyendo y su resultado así como los comentarios del scripts

Parámetro	Categoría	Descripción
$path	Obligatorio	Especifica el nombre del script que se va a ejecutar.

RM

Este comando permitirá eliminar un archivo Tendrá los siguientes parámetros:

Parámetro	Categoría	Descripción
$id	Obligatorio	Especifica el id de la partición en la que se eliminará el archivo. Si no existe debe mostrar error.
$FILEID	Obligatorio	Este parámetro será el nombre del archivo que se eliminará. Si no existe el archivo, debe mostrarse un mensaje de error. Deberán poder eliminarse una lista de archivos.

```
/*Se eliminará el archivo con nombre archivo0*/
        Mkfile $id=>vda1 $FILEID=>archivo0

/*Se eliminará los archivos con nombre archivo1, archivo3, archivo4*/
        Mkfile $id=>vda1 $FILEID=>archivo1 $FILEID=>archivo3 $FILEID=>archivo4
```

MKDIR

Este comando es similar a mkfile, pero no crea archivos, sino carpetas. 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. Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro @p, que se explica posteriormente.
@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 $id=>vda1 $path->"/home/user/docs/usac"

/*Crea la carpeta "archivos 2022*/
/*La carpeta padre ya debe existir*/
        Mkdir $ID=>vda1 $path=>"/home/mis documentos/archivos 2022"
```

CP

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

Parámetro	Categoría	Descripción
$id	Obligatorio	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	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se desea copiar. Debe copiar todos los archivos y carpetas con todo su contenido, a los cuales tenga permiso de lectura. Si no tiene permiso de lectura, no realiza la copia únicamente de ese archivo o carpeta. Muestra un error si no existe la ruta.
$dest	Obligatorio	Este parámetro será la ruta de la carpeta a la que se copiará el archivo o carpeta.
$iddest	Obligatorio	Especifica el id de la partición en la que se copiará el archivo o carpeta. Si no existe debe mostrar un error. La partición en la cual se copiará debe estar montada para dicha operación.

CP

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	Categoría	Descripción
$id	Obligatorio	Especifica el id de la partición en la que se buscará el archivo o carpeta. Si no existe, debe mostrar un error.
$path	Obligatorio	Este parámetro será la ruta de la carpeta en el que se inicia la búsqueda, deberá buscar en todo su contenido. Debe tener permisos de lectura en los archivos que buscará.

$name	Obligatorio	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.

LS

Este comando permitirá listar el contenido de una carpeta según el path que se indique.

Parámetro	Categoría	Descripción
$id	Obligatorio	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	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se desea obtener el contenido.
$Order	Obligatorio	Este parámetro indica el orden en el cual se desplegarán los datos de la carpeta: ● F = muestra el contenido del directorio sin ordenar ● C = ordena el contenido de acuerdo a la fecha de creación ● R = coloca la lista de contenido el orden alfabético inverso

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

Parámetro	Categoría	Descripción
$id	Obligatorio	Especifica el id de la partición en la que se leerá el archivo. Si no existe debe mostrar error.
$filen	Obligatorio	Permitirá admitir como argumentos una lista de n ficheros que hay que enlazar. Estos se encadenarán en el mismo orden en el cual fueron especificados. Si no existe el archivo, debe mostrarse un mensaje de error.

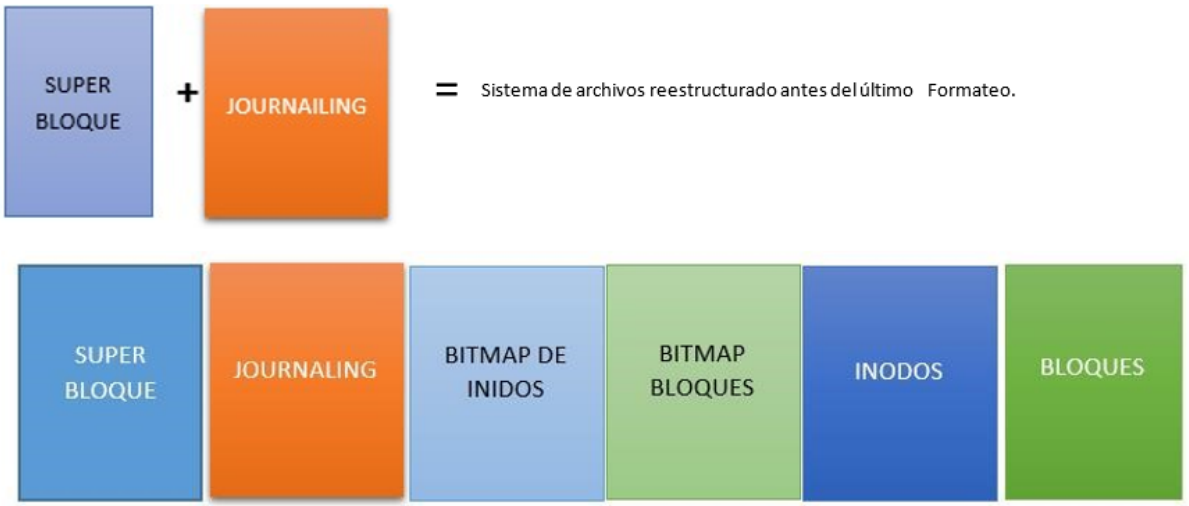
PAUSE

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.

PÉRDIDA Y RECUPERACIÓN DEL SISTEMA DE ARCHIVOS EXT3

RECOVERY FILE SYSTEM

La recuperación del sistema se hará por medio del journaling y el superbloque. Se recuperará el sistema a un estado consistente antes del último formateo.



/*Recuperando el sistema de archivos EXT3 de la partición 1*/

```
Recovery $id=>vda1
```

SIMULATE SYSTEM LOSS

Este formatea los siguientes bloques de datos para simular un fallo en el disco (una partición en específica), una inconsistencia o pérdida de información. Se deberán limpiar los siguientes bloques con el carácter /0.

- Bloque de bitmap de árbol virtual de directorio
- Bloque de bitmap de Bloques
- Inodos
- Bloque de datos

Parámetro	Categoría	Descripción
$id	Obligatorio	Especifica el id de la partición a la que se le simulara la pérdida del sistema.

/*Simulando la pérdida del sistema de archivos EXT3 de la partición1*/

```
Loss $id=>vda1
```

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.** Es decir, si no hace reporte de bloques, no tendrá derecho a la calificación de todos los aspectos relativos a los bloques, ya que no se puede comprobar que el estudiante haya implementado dicha funcionalidad.

REP

Recibirá el nombre del reporte que se desea y lo generará con graphviz y en archivo de texto según se requiera, en una carpeta existente.

Parámetro	Categoría	Descripción
$name	Obligatorio	Nombre del reporte a generar. Tendrá los siguientes valores: ● Mbr ● disk ● block ● Bm_block ● Bm_inode ● inode ● Journaling ● tree ● sb ● file Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error.
$path	Obligatorio	indica una carpeta y el reporte 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.

EJEMPLO

rep $id=vda2 $Path=>"/home/user/reports/reporte 2.jpg" $name=>bm_arbdir
rep $id=>vda1 $Path="/home/user/reports/reporte 3.jpg" $name=>MBR

REPORTE DE BITMAPS

Los reportes se generarán en un imagenes, que deberán seguir el siguiente formato. B para bloques & Y para Inodos Los reportes serán:

- bm_block
- bm_inode

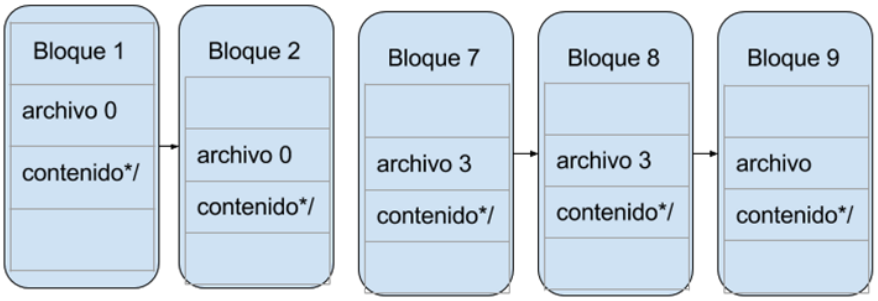
0 B1	0 B2	1 B3	0 B4	1 B5	1 B6	1 B7	1 B8	1 B9	1 B10
0 B11	0 B12	0 B13	0 B14	0 B15	0 B16	0 B17	0 B18	0 B19	0 B20
0 B21	0 B22	0 B23	0 B24	0 B25	0 B26	0 B27	0 B28	0 B29	0 B30
0 B31	0 B32	0 B33	0 B34	0 B35	0 B36	0 B37	0 B38	0 B39	0 B4
0 B41	0 B42	0 B43	0 B44	0 B45	0 B46	0 B47	0 B48	0 B49	0 B50
0 B51	0 B52	0 B53	0 B54	0 B55	0 B56	0 B57	0 B58	0 B59	0 B60
0 B61	0 B62	0 B63	0 B64	0 B65	0 B66	0 B67	0 B68	0 B69	0 B70
0 B71	0 B72	0 B73	0 B74	0 B75	0 B76	0 B77	0 B78	0 B79	0 B80

0 Y1	0 Y2	1 Y3	0 Y4	1 Y5	1 Y6	1 Y7	1 Y8	1 Y9	1 Y10
0 Y11	0 Y12	0 Y13	0 Y14	0 Y15	0 Y16	0 Y17	0 Y18	0 Y19	0 Y20
0 Y21	0 Y22	0 Y23	0 Y24	0 Y25	0 Y26	0 Y27	0 Y28	0 Y29	0 Y30
0 Y31	0 Y32	0 Y33	0 Y34	0 Y35	0 Y36	0 Y37	0 Y38	0 Y39	0 Y4
0 Y41	0 Y42	0 Y43	0 Y44	0 Y45	0 Y46	0 Y47	0 Y48	0 Y49	0 Y50
0 Y51	0 Y52	0 Y53	0 Y54	0 Y55	0 Y56	0 Y57	0 Y58	0 Y59	0 Y60
0 Y61	0 Y62	0 Y63	0 Y64	0 Y65	0 Y66	0 Y67	0 Y68	0 Y69	0 Y70
0 Y71	0 Y72	0 Y73	0 Y74	0 Y75	0 Y76	0 Y77	0 Y78	0 Y79	0 Y80
0 Y81	0 Y82	0 Y83	0 Y84	0 Y85	0 Y86	0 Y87	0 Y88	0 Y89	0 Y90

BLOCK

Para verificar si la estructura lógica del sistema es la correcta, se debe de realizar una gráfica en Graphviz mostrando todos los directorios y subdirectorios mostrando su orden jerárquico similar a la siguiente:

Nota: Los bloques que pertenecen al mismo archivo deben estar unidos por medio de una flecha, tal cual se muestran en el ejemplo

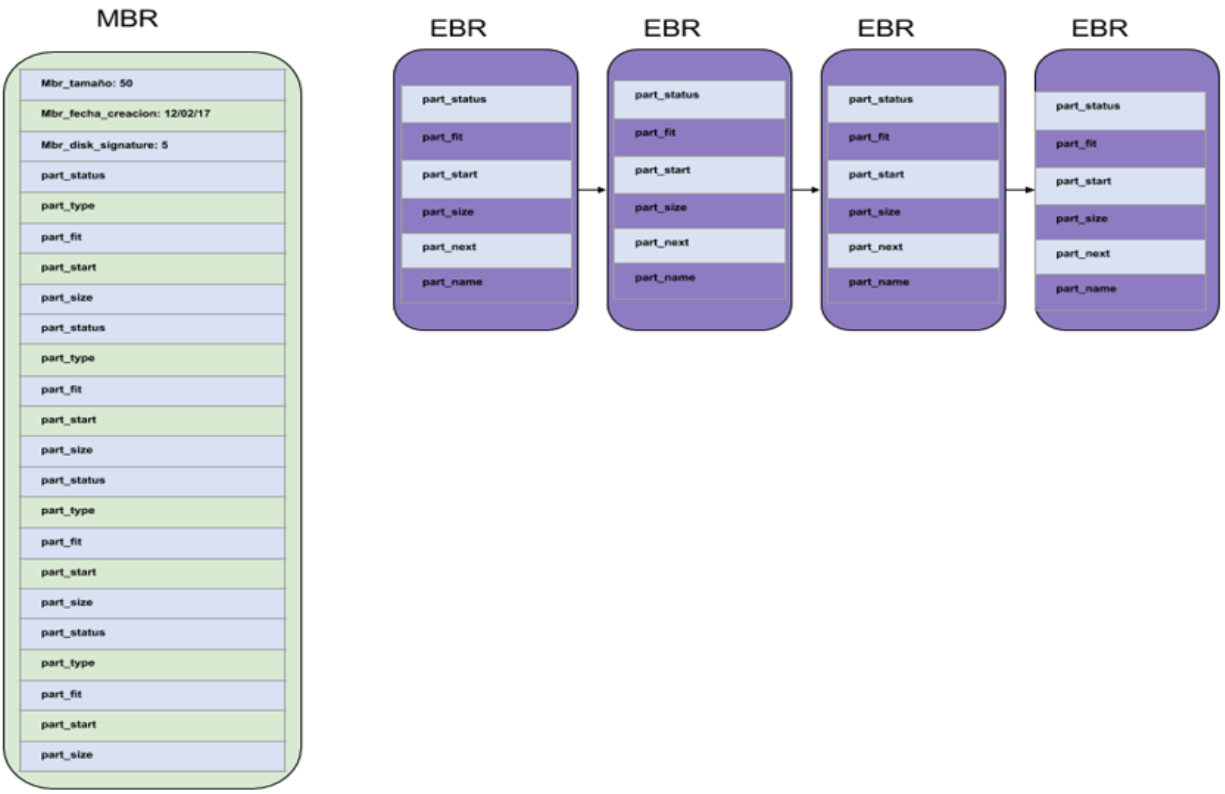


DISK

Este reporte mostrará la estructura de las particiones (mostrar nombre de la partición, el tipo, tamaño total, Si la partición ya tiene formato, debe incluir el número de bloques total, número de bloques ocupados y número de bloques libres) y el MBR del disco.

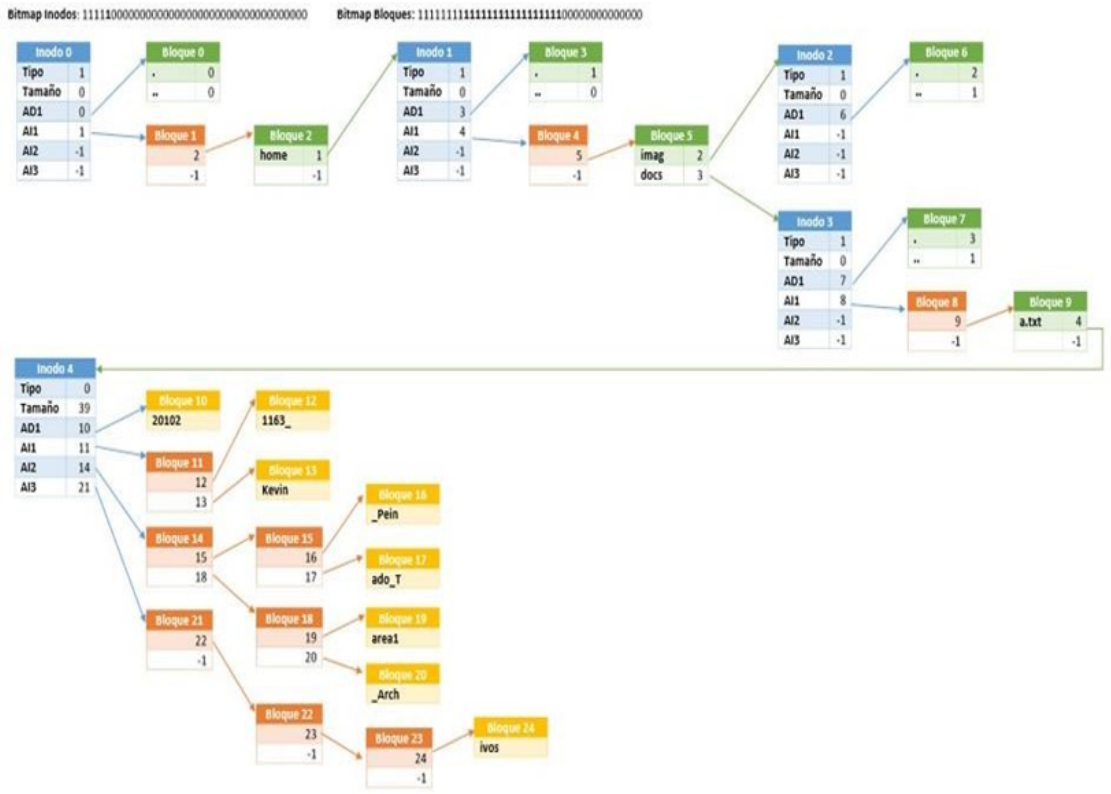
MBR

En este reporte se deben mostrar todos los datos que se indicaron anteriormente en el MBR y la información de cada EBR si existieran particiones lógicas dentro de cada partición extendida.



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, se utilizaron bloques de carpeta con capacidad 2, bloques de apuntadores con capacidad 2 y bloques de archivo con capacidad 5):



INODE

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

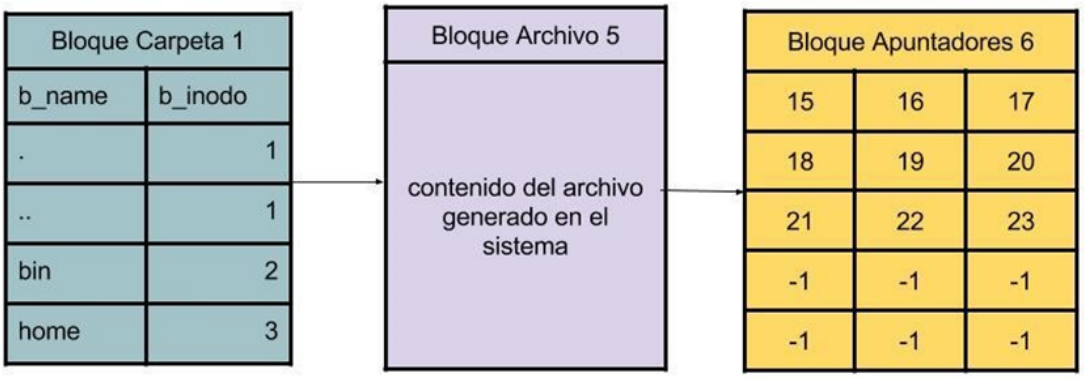
Nombre	información
i_uid	
I_gid	
i_size	
i_atime	
i_ctime	
i_mtime	
i_block	
i_type	
i_perm	

Nombre	información
i_uid	
I_gid	
i_size	
i_atime	
i_ctime	
i_mtime	
i_block	
i_type	
i_perm	

Nombre	información
i_uid	
I_gid	
i_size	
i_atime	
i_ctime	
i_mtime	
i_block	
i_type	
i_perm	

BLOCK

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



SB

Muestra toda la información del superbloque en una tabla.

Nombre	Valor
s_inodes_count	200
s_blocks_count	600
s_free_blocks_count	10
s_free_inodes_count	100
s_mtime	30/07/2016 15:38
s_umtime	30/07/2016 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

SuperBloque Partición 1 en Disco1.dsk

FILE

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

JOURNALING

Tipo Operación: “El tipo de operación a realizarse”
Tipo: “Archivo/Directorio”
Nombre: “Nombre archivo o directorio”
Contenido: “Si hay datos contenidos”
Fecha “Fecha de transacción”

Tipo Operación: “El tipo de operación a realizarse”
Tipo: “Archivo/Directorio”
Nombre: “Nombre archivo o directorio”
Contenido: “Si hay datos contenidos”
Fecha “Fecha de transacción”

RESTRICCIONES

- El lenguaje por utilizar es C/C + +. No se permite el uso de otro lenguaje.
- Unicamente se calificará el proyecto sobre una instalación física de una distribución GNU/Linux.
- NO se permite la modificación de código durante la calificación más de aquellas modificaciones solicitadas por el auxiliar para la comprobación de la autoría del proyecto. El estudiante únicamente podrá utilizar el ejecutable entregado.
- 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.
- No se permite agregar o quitar atributos a los structs que se utilizarán en el proyect
- De no existir una forma gráfica de poder validar el funcionamiento de la aplicación se penalizará con el 100% de la nota.
- **NO HABRÁ PRÓRROGA.**
- **Se utilizará software para la detección de copias, las copias tendrán una nota de 0 y serán reportadas a la escuela**

ENTREGA

El proyecto se entregará el día **Miércoles 15 de junio del 2022** hasta 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, **el auxiliar 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. **Está prohibido generar el ejecutable durante la calificación.**

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

- Aplicación de Comandos
- Creación de Particiones con la aplicación de los ajustes y Mount
- Ejecución Completa del Script
- Creación de Carpetas y Archivos
- Pause
- Manejo de la configuración RAID 1
- Todos los Reportes