Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas
Manejo e Implementación de Archivos
Segundo semestre 2019
Catedráticos: Ing. Álvaro Díaz, Ing. Oscar Paz
Tutores académicos: Diego Osorio, Carlos Monterroso

Proyecto 1 - Fase 2

Introducción:

El curso de Manejo e Implementación de Archivos busca que el estudiante aprenda los conceptos sobre la administración de archivos, tanto en hardware como software, sistemas de archivos, particiones, entre otros conceptos, para después introducirse en las bases de datos. El proyecto busca que el estudiante aplique estos conceptos y pueda aprenderlos implementando su funcionalidad.

Objetivos:

- Aprender a administrar archivos y escribir estructuras en C
- Comprender el sistema de archivos EXT3
- Aplicar el formateo rápido y completo en una partición
- Crear una aplicación de comandos
- Aplicar la teoría de ajustes
- Aplicar la teoría de particiones
- Utilizar GraphViz para mostrar reportes
- Restringir y administrar el acceso a los archivos y carpetas en ext2/ext3 por medio de usuarios
- Administrar los usuarios y permisos por medio de grupos

Aplicación de Comandos:

La aplicación será totalmente en consola, a excepción de los reportes en Graphviz. Esta no tendrá menús, sino que se utilizarán comandos.

No distinguirá entre mayúsculas y minúsculas. Hay parámetros obligatorios y opcionales. Solo se puede colocar un comando por línea.

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

ADMINISTRACIÓN DE DISCOS:

1.MKFS

Este comando realiza un formateo completo de la partición, se formateará como ext2 por defecto si en caso el comando fs no está definido. También creará un archivo en la raíz llamado users.txt que tendrá los usuarios y contraseñas del sistema de archivos. La estructura de este archivo se explicará más adelante.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-id	Obligatorio	Indicará el id que se generó con el comando mount de la fase anterior. Si no existe mostrará error. Se utilizará para saber la partición y el disco que se utilizará para hacer el sistema de archivos.
-type	Opcional	Indicará que tipo de formateo se realizará. 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.
-fs	Opcional	Indica el sistema de archivos a formatear ext2 / ext3. Por defecto será ext2. Y los valores serán. 2fs para ext2 o 3fs para ext3

Ejemplos:

```
#Realiza un formateo rápido de la partición en el id vda1 en ext2
mkfs -type=fast -id=vda1

#Realiza un formateo completo de la particion que ocupa el id vdb1
mkfs -id=vdb1
```

ADMINISTRACIÓN DE Usuarios y Grupos:

Este archivo será un archivo de texto, llamado users.txt guardado en el sistema ext2/ext3 de la raíz de cada partición. Existirán dos tipos de registros, unos para grupos y otros para usuarios. Un id 0 significa que el usuario o grupo está eliminado, el id de grupo o de usuario irá aumentando según se vayan creando usuarios o grupos. Tendrá la siguiente estructura:

GID, Tipo, Grupo

UID, Tipo, Grupo, Usuario, Contraseña

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

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

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

Este archivo se podrá modificar con comandos que se explicarán más adelante.

2.Login

Este comando se utiliza para iniciar sesión en el sistema. No se puede iniciar otra sesión sin haber hecho un logout antes, si no, debe mostrar un mensaje de error indicando que debe cerrar sesión. Recibirá los Siguientes parámetros:

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

Ejemplos:

```
#Se loguea en el sistema como usuario root
login -usr=root -pwd=123 -id=vda1
```

```
login -usr="mi usuario" -pwd="mi pwd" -id=vdb2
```

3.Logout

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

Ejemplos:

```
#Termina la sesión del usuario
Logout

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

Todos los siguientes comandos que se explicarán de aquí en adelante, **necesitan que exista una sesión en el sistema ya que se ejecutan sobre la partición en la que inicio sesion**. Si no, debe mostrar un mensaje de error indicando que necesita iniciar sesión.

4.Mkgrp

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

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

Ejemplo:

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

#Debe mostrar mensaje de error ya que el grupo ya existe
```

```
mkgrp -name="usuarios"
```
El archivo users.txt debería quedar como el siguiente:

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

5.Rmgrp

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

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

Ejemplo:

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

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

El archivo users.txt debería quedar como el siguiente:

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

6.mkusr

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

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

7.Rmusr

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

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

Ejemplo:

```
#Elimina el usuario user1 de la partición de la sesión actual
rmusr -usr=user1
```

El archivo users.txt debería quedar así:

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

USUARIO ROOT:

Este usuario es especial y no importando que permisos tenga el archivo o carpeta, el siempre tendrá los permisos 777 sobre cualquier archivo o carpeta (Esto se explica en detalle posteriormente). Podrá mover, copiar, eliminar, crear, etc. Todos los archivos o carpetas que desee. No se le negará ninguna operación por permisos, ya que él los tiene todos. Los permisos únicamente se pueden cambiar con chmod que se explicará posteriormente. Se debe tomar en cuenta en que categoría está el usuario, si es el propietario, si pertenece al mismo grupo en que está el propietario o si es otro usuario que no pertenece al grupo del propietario. En base a esta comprobación, el usuario puede estar en tres distintas categorías: Propietario (**U**), Grupo (**G**) u Otro (**O**). Dependiendo de estas categorías se determinan los permisos hacia el archivo o carpeta.

ADMINISTRACIÓN De Carpetas Archivos y Permisos:

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

8.chmod

Cambiará los permisos de uno o varios archivos o carpetas. Lo podrá utilizar el usuario root en todos los archivos o carpetas y también lo podrán utilizar

otros usuarios, pero solo sobre sus propios archivos. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta en la que se encuentra el archivo o carpeta a la que se le cambiarán los permisos.
-ugo	Obligatorio	Indica los permisos que tendrán los usuarios. Serán tres números, uno para el **U**suario, el siguiente para el **G**rupo al que pertenece el usuario y el último para **O**tros usuarios fuera del grupo. Cada número tendrá los valores desde el 0 al 7. Si el número esta fuera de este rango se mostrará un error. A nivel de bits significan permisos para lectura, escritura y ejecución. Por ejemplo el número 5 (101) indica permisos para leer y ejecutar. El número 7 indica permisos para las tres operaciones anteriores. El número 0 indica que no tendrá permisos para utilizar el archivo.
-r	Opcional	Indica que el cambio será recursivo en el caso de carpetas. El cambio afectará a **todos** los archivos y carpetas en la que la ruta contenga la carpeta especificada por el parámetro **–path** y que sean propiedad del usuario actual

Ejemplos:

```
#Cambia los permisos de la carpeta home recursivamente
#Todos los archivos o carpetas que tengan /home cambiarán
#Por ejemplo si existiera /home/user/docs/a.txt
#Cambiaría los permisos de las tres carpetas y del archivo
chmod -path=/home -R -ugo=764
```

```
#Cambia los permisos de la carpeta home
#Se debe comprobar que la carpeta home pertenezca al usuario
#actual, si no deberá mostrar un mensaje de error.

chmod -id=vda1 -path=/home -ugo=777
```

9.MKFILE:

Este comando permitirá crear un archivo, el propietario será el usuario que actualmente ha iniciado sesión. Tendrá los permisos 664. El usuario deberá tener el permiso de escritura en la carpeta padre, si no debe mostrar un error. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORíA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo que se creará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si ya existe debe 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

De venir -cont y -size en un mismo comando se deberá tomar solo el -cont.

Ejemplos:

```
#Crea el archivo a.txt
#Si no existen las carpetas home user o docs se crean
#El tamaño del archivo es de 15 bytes #El contenido sería: 012345678901234
mkFile -SIZE=15 -PatH=/home/user/docs/a.txt –p

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

#Crea el archivo b.txt
#El contenido del archivo será el mismo que el archivo b.txt
#que se encuentra en el disco duro de la computadora.

```
mkfile -id=vda1 -path=/home/user/docs/b.txt -p
-cont=/home/Documents/b.txt
```

10.cat

Este comando permitirá mostrar el contenido del archivo, si el usuario que actualmente está logueado tiene acceso al permiso de lectura. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORíA	DESCRIPCIÓN
-file	Obligatorio	Contiene la Path del archivo que está contenido en nuestro sistema de archivos.

Ejemplos:
#Lee el archivo a.txt

```
Cat -file=/home/user/docs/a.txt
```

#En la terminal debería mostrar el contenido, en este ejemplo
#01234567890123

11.Rem

Este comando permitirá eliminar un archivo o carpeta y todo su contenido, si el usuario que actualmente está logueado tiene acceso al permiso de escritura sobre el archivo y en el caso de carpetas, eliminará todos los archivos o subcarpetas en los que el usuario tenga permiso de escritura. Si no pudo eliminar un archivo o subcarpeta dentro de la carpeta por permisos, no deberá
eliminar nada dentro de esa carpeta ni la carpeta como tal. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se eliminará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si no existe el archivo o no tiene permisos de escritura en la carpeta o en el archivo, debe mostrarse un mensaje de error. Si no pudo eliminar algún archivo o carpeta no deberá eliminar los padres.

Ejemplos:

```
#Elimina el archivo a.txt, b.txt muestra error si no tiene permiso rem
-PatH=/home/user/docs/a.txt
rem -PatH=/home/user/docs/b.txt #Error por permisos
#Elimina la carpeta user y todo su contenido (docs, a.txt)
#Si el usuario no tuviera permiso de escritura sobre b.txt #No debería eliminar las carpetas padre docs ni user, solo a.txt rem -PatH=/home/user
rem -PatH=/home/user
```

12.Edit

Este comando permitirá editar el contenido de un archivo para asignarle otro contenido. Funcionará si el usuario que actualmente está logueado tiene acceso al permiso de lectura y escritura sobre el archivo, si no debe mostrar error. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo que se modificará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si no existe, debe mostrar un mensaje de error.
-cont	Obligatorio	Contiene el contenido que será Agregado a la edición.

Ejemplos:

```
#Modifica el archivo a.txt
 #El contenido viejo sería: 12345
Edit -PatH=/home/user/docs/a.txt -cont=Hola mundo

 #El contenido Nuevo sería: 12345Hola mundo
```

13. ren

Este comando permitirá cambiar el nombre de un archivo o carpeta, si el usuario actualmente logueado tiene permiso de escritura sobre el archivo o carpeta. Tendrá los siguientes parámetros:

PARÁMETR	CATEGORÍA	DESCRIPCIÓN

O		
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta al que se le cambiará el nombre. Si lleva espacios en blanco deberá encerrarse entre comillas. Si no existe el archivo o carpeta o no tiene permisos de escritura
-name	Obligatorio	Especificará el nuevo nombre del archivo, debe verificar que no exista un archivo con el mismo nombre, de ser así debe mostrar un mensaje de error.

Ejemplos:

```
#Cambia el nombre del archivo a.txt a b1.txt
ren -PatH=/home/user/docs/a.txt –name=b1.txt

#Deberá mostrar error ya que el archivo b1.txt ya existe ren
-PatH=/home/user/docs/c.txt –name=b1.txt
```

14. Mkdir

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

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta de la carpeta que se creará. Si lleva espacios en blanco deberá encerrarse entre comillas.

		Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro –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 diciembre"
#La carpeta padre ya debe existir
Mkdir -ID=vda1 -path="/home/mis documentos/archivos 2018"
```

15. cp

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

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se desea copiar. Si lleva espacios en blanco deberá encerrarse entre comillas.

		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. Debe tener permiso de escritura sobre la carpeta. Si lleva espacios en blanco deberá encerrarse entre comillas. Debe mostrar un mensaje de error si no tiene permisos para escribir o si la carpeta no existe

Ejemplos:

```
#/
#|_home          #664
#  |_user            #664
#  | |_documents  #664
#  |   |_a.txt    #664
#  |   |_b.txt    #224
#  |_images         #664


#Copia documents a images

cp -Path="/home/user/documents" -dest="/home/images"

#No copia b.txt por falta de permisos

#/
```

#|_home #664

|_user #664

| |_documents #664

| |_a.txt #664

| |_b.txt #224

|_images #664

|_documents #664

|_a.txt #664

16. 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. Solo se deberán verificar los permisos de escritura sobre la carpeta o archivo origen.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se desea copiar. Si lleva espacios en blanco deberá encerrarse entre comillas. 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. Debe tener permiso de escritura sobre la carpeta. Si lleva espacios en blanco deberá encerrarse entre comillas. Debe mostrar un mensaje de error si no tiene permisos para escribir o si la carpeta no existe.

Ejemplo:

```
#/
#|_home        #664
#  |_user      #664
#  | |_documents  #664
#  |   |_a.txt    #664
#  |   |_b.txt    #224
#  |_images     #664


#Mueve documents a images
mv -Path="/home/user/documents" -dest="/home/images"

#Mueve b.txt, ya que solo se comprueban los permisos de documents
#/
#|_home        #664
#  |_user      #664
#  |_images     #664
#    |_documents  #664
#      |_a.txt    #664
#      |_b.txt    #224
```

16. find

Este comando permitirá realizar una búsqueda por el nombre del archivo o carpeta Específico. Recibe los siguiente parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta de la carpeta en el que se inicia la búsqueda, deberá buscar en todo su contenido. Si lleva espacios en blanco deberá encerrarse entre comillas.
-name	Obligatorio	Debe tener permisos de lectura en los archivos que buscará. Indica el nombre de la carpeta o archivo que se desea buscar.

Ejemplos:

```
find -Path="/" -name=a.jpg

#Arbol actual que debe imprimir
#/
#|_home       #664
#  |_user     #664
#  | |_a.txt   #664
#  | |_b.txt   #420
#  |_images    #664
#    |_a.jpg   #664
#    |_abcd.jpg   #664
```

17. chown

Cambiará el propietario de uno o varios archivos o carpetas. Lo podrá utilizar el usuario root en **todos** los archivos o carpetas y también lo podrán utilizar otros usuarios, pero **solo** sobre sus **propios** archivos.

Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta en la que se encuentra el archivo o carpeta a la que se le cambiará el propietario. Si no existe la ruta deberá mostrar mensaje de error
-R	Opcional	Indica que el cambio será recursivo en el caso de carpetas. El cambio afectará a **todos** los archivos y carpetas en la que la ruta contenga la carpeta especificada por el parámetro **–path**.
-usr	Obligatorio	Nombre del nuevo propietario del archivo o carpeta. Si no existe o está eliminado debe mostrar error.

Ejemplos:

```
#Cambia el propietario de la carpeta home recursivamente
chown -path=/home -R -usr=user2


#Cambia los permisos de la carpeta home
chown  -path=/home -usr=user1
```

18. chgrp

Cambiará el grupo al que pertenece el usuario. Únicamente lo podrá utilizar el usuario root.

Recibirá los siguientes parámetros:

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

Ejemplos:

```
#Cambia el grupo del user2
chgrp -usr=user2 -grp=grupo1
```

```
#Cambia el grupo del user1
chgrp -usr=user1 -grp=grupo2
```

20. Pause

Este comando será solo la palabra “pause” no tiene atributos al ingresar este comando se pondrá en pausa solicitando que apache 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.

21. Pérdida y Recuperación del Sistema de Archivos EXT3

22.1.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
```

22. Simulate System Loss

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

1. Bloque de bitmap de árbol virtual de directorio
2. Bloque de bitmap de Bloques
3. Inodos
4. Bloque de datos.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN

-id	Obligatorio	Especifica el id de la partición a la que se le simulara la pérdida del sistema.

Ejemplo:

#Simulando la perdida del sistema de archivos EXT3 de la partición1

```
Loss –id~:~vda1
```

23.Sistema de Archivos EXT3

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

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

$$tamaño\_particion = sizeof(superblock) + n + 3 * n + n * sizeof(inodos) + 3 * n * sizeof(block)$$

$$número\_estructuras=floor(n)$$

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:

$$tamaño\_particion = sizeof(superblock) + n + n*Sizeof(Journaling) + 3 * n + n * sizeof(inodos) + 3 * n * Sizeof(block)$$

$$numero\_estructuras = floor(n)$$

24.Súper Bloque

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

NOMBRE	TIPO	DESCRIPCIÓN
s_filesystem_type	int	Guarda el número que identifica el sistema de archivos utilizado
s_inodes_count	int	Guarda el número total de inodos
s_blocks_count	int	Guarda el número total de bloques
s_free_blocks_count	int	Contiene el número de bloques libres
s_free_inodes_count	int	Contiene el número de inodos libres
s_mtime	time	Última fecha en el que el sistema fue montado
s_umtime	time	Última fecha en que el sistema fue desmontado
s_mnt_count	int	Indica cuantas veces se ha montado el sistema
s_magic	int	Valor que identifica al sistema de archivos, tendrá

		el valor 0xEF53
s_inode_size	int	Tamaño del inodo
s_block_size	int	Tamaño del bloque
s_firts_ino	int	Primer inodo libre
s_first_blo	int	Primer bloque libre
s_bm_inode_start	int	Guardará el inicio del bitmap de inodos
s_bm_block_start	int	Guardará el inicio del bitmap de bloques
s_inode_start	int	Guardará el inicio de la tabla de inodos
s_block_start	int	Guardará el inicio de la tabla de bloques

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

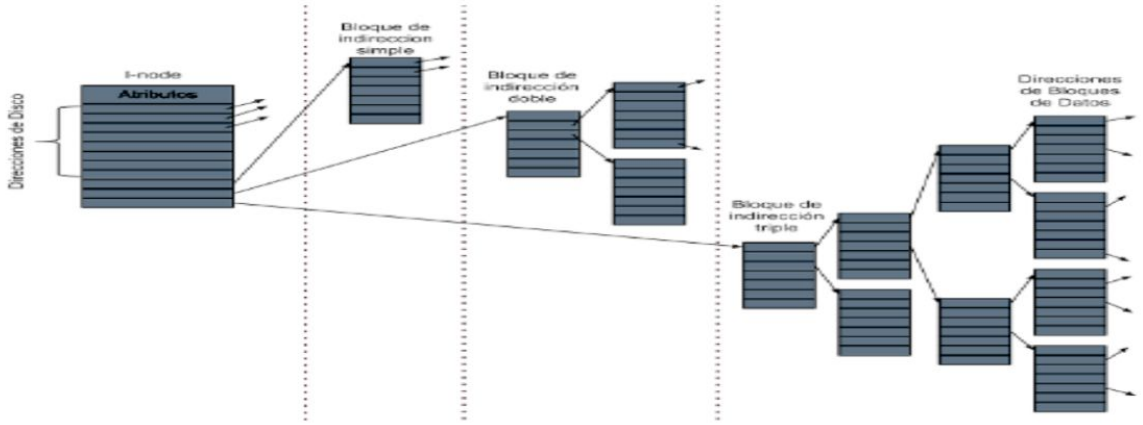
25.Tabla de inodos

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	Úlitma fecha en la que se modificó el inodo
i_block	int	Array en los que los primeros 12 registros son bloques directos. El 13 será el número del bloque simple indirecto. El 14 será el número del bloque doble indirecto. El 15 será el número del bloque triple indirecto Si no son utilizados tendrá el valor -1.
i_type	char	Indica si es archivo o carpeta. Tendrá los siguientes valores: 1 = Archivo 0 = Carpeta
i_perm	int	Guardará los permisos del archivo o carpeta. Se trabajará a nivel de bits, estará dividido de la siguiente forma: Los primeros tres bits serán para el **U**suario **i_uid**. Los siguientes tres bits serán para el **G**rupo al que pertenece el usuario. Y los últimos tres bits serán para los permisos de **O**tros usuarios. Cada grupo de tres bits significa lo siguiente: El primer bit indica el permiso de lectura **R**. El segundo bit indica el permiso de escritura **W**. El tercer bit indica el permiso de ejecución **X**.

La siguiente imagen muestra el funcionamiento de los bloques indirectos.



26.Bloques 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

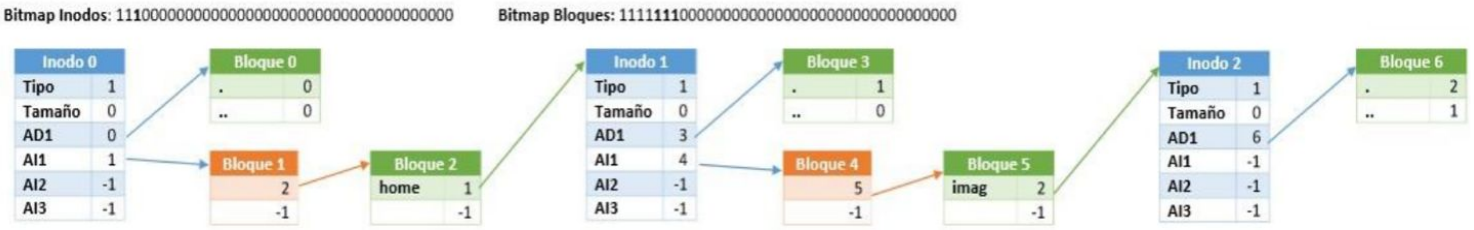
La estructura content será como la siguiente:

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:

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



27.Bloques de Archivos

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

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

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

28.Bloques 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)

29.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 (/) y el archivo users.txt dentro de la raiz.

29.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 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.

30. Reportes

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 inodos, no tendrá derecho a la calificación de todos los aspectos relativos a los inodos, ya que no se puede comprobar que el estudiante haya implementado dicha funcionalidad.

30.1.rep

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

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

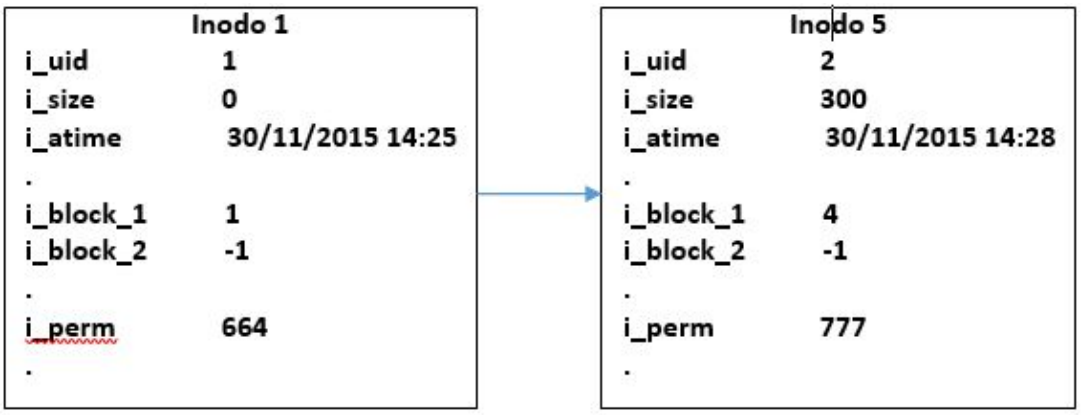
Ejemplos

```
rep -id=vda2 -Path="/home/user/reports/reporte
2.pdf" -name=ls -ruta="/home/mis documentos"
```

```
rep -id=vda1 -Path="/home/user/reports/reporte
3.jpg" -name=tree
```

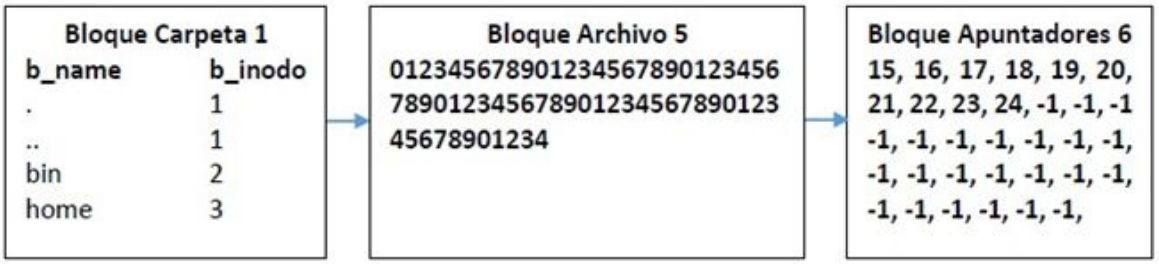
30.2. Inode

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



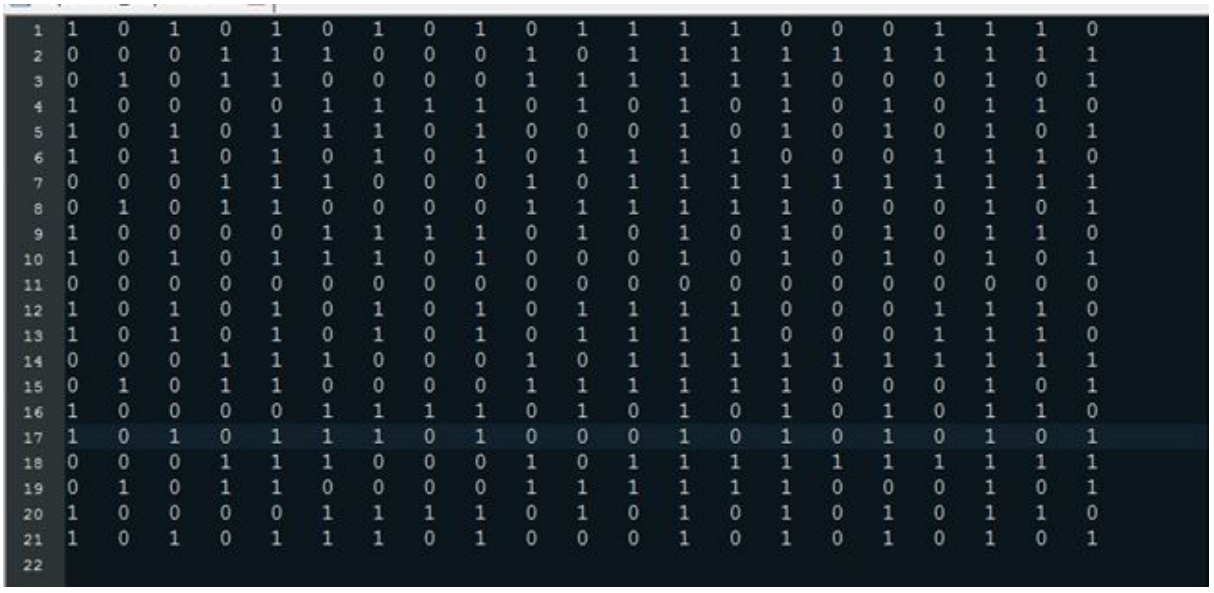
30.3. Block

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



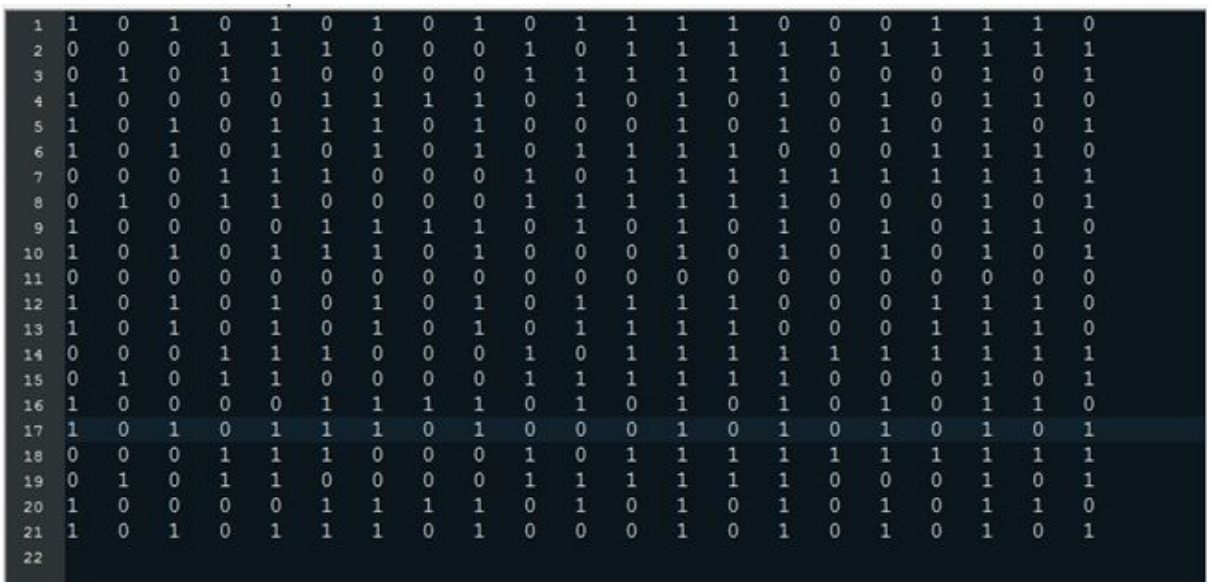
30.4. bm_inode

Este reporte mostrará la información del bit map de inodos, mostará todos los bits, libres o utilizados. Este reporte se generara en un archivo de texto mostrando **20** registros por línea.



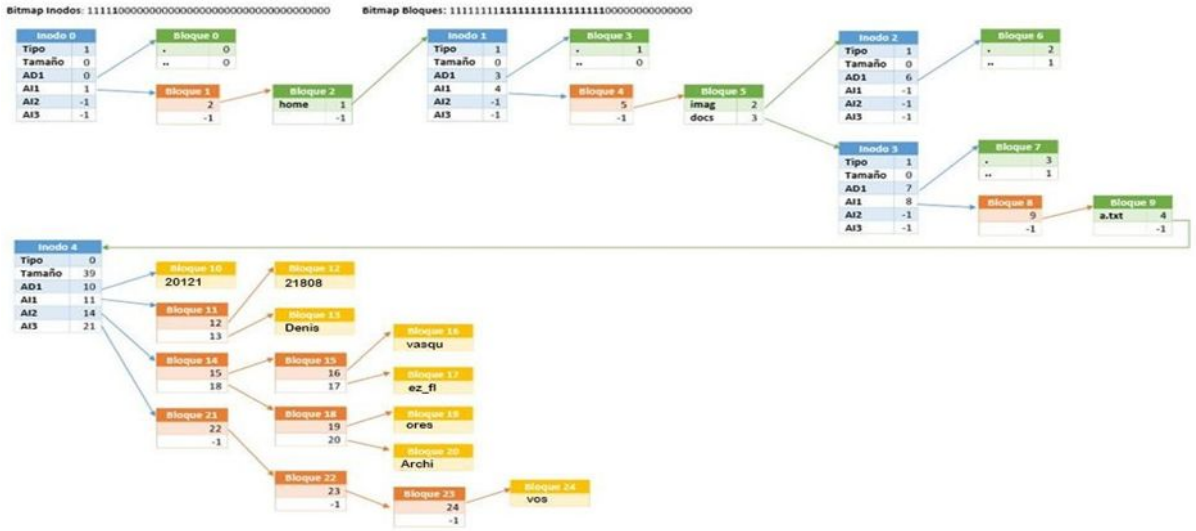
30.5. bm_block

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



30.6. Tree

Este reporte genera el árbol de **todo** el sistema ext2/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):



30.7. 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	17/08/2019 15:38
s_umtime	17/08/2019 15:36
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

30.8. File

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

Ejemplo:

a.txt

```
01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
890123456
```

30.9. Ls

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

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

31.Entrega

El proyecto se entregará el día martes 29 de Septiembre del 2019 hasta las 23:59 se habilitará una opción en el classroom para que puedan subir su proyecto como la primera fase. **No se aceptan proyectos fuera de la fecha y hora establecida.** Se debe informar si alguien no puede asistir a la calificación.

Se deberán entregar 2 archivos con el nombre **Carnet.tar.gz y Carnet_o.txt.** Dentro del comprimido pondrán todo el código fuente que utilizaron, así como el ejecutable.

- 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.

- El tutor podrá hacerle preguntas acerca de su código para verificar su validez y autoría.

- Las dudas se responderán en el grupo y aplicarán a todos los estudiantes

- El lenguaje por utilizar es C o C++. No se permite ningún otro lenguaje de programación.

- Solo se calificará sobre una instalación física de una distribución GNU/Linux. De no ser así se anulará el proyecto.

- **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, árboles, etc.) para el manejo de los archivos o carpetas.

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

- Aplicación de Comandos
- Creación de Particiones Y Mount
- Ejecución **COMPLETA** del script
- Crear carpetas y archivos
- Crear Usuario y Grupos
- Cat
- Pause
- Simulación de Pérdida y Recuperación

- **Todos** los Reportes.