MANUAL TECNICO

PROYECTO 1

Nombre:	Victor Alfonso López Morales
Carnet:	2011-13915
Curso:	Manejo e implementación de archivos
Sección:	A

INTRODUCCIÓN

Proyecto 1 es una aplicación de consola que permite gestionar la creación de discos, particiones y formateo con el sistema de archivos LWH v 2.0. Tambien permite generar reportes para visualizar el estado de sus particiones y sistema de archivos.

REQUISITOS

- Tener un sistema con una distribución de Linux basada en debian.
- Tener un visor de documentos PDF.
- Se utilizo QT Creator para la codificación junto a las herramientas Flex y Bison.

Comandos

La aplicación es totalmente en consola, a excepción de los reportes en Graphviz. La aplicación consta únicamente de una consola en la cual el usuario podrá ingresar comando que le servirán para la realización de las diferentes acciones posibles. Algunas normas que debe cumplir se presentan a continuación, no distinguirá entre **mayúsculas** y **minúsculas**. Hay parámetros obligatorios y opcionales. Solo se puede colocar un comando por línea.

Si se utiliza un parámetro no permitido, mostrara un mensaje de error. Se utilizarán espacios en blanco para separar cada parámetro. Los parámetros pueden venir en cualquier orden.

Si un comando necesita más de una línea se debe utilizar \^ al final de la línea para indicar que continúa en la siguiente línea. También se puede ejecutar archivos de scripts con estos comandos a través del comando exec, que se describe más adelante. Se podrá comentar cada comando Los comentarios de estos scripts empezarán con #, dichos comandos son únicamente de línea. Estos comandos se explicarán en detalle a continuación.

MKDISK

Parametros

- **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.
- **Path:** Este parámetro será la ruta en el que se creará el archivo que representará el disco duro.
  NOTA: La dirección de la carpeta únicamente llevará comillas dobles (“) cuando algún name dentro de la dirección contenga espacios, de lo contrario podría venir sin comillas.
- **Name:** Este parámetro será el name del disco con extensión dsk. Si no contiene la extensión dsk debe mostrar un mensaje de 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)

Ejemplo:

```
#Crea un disco de 3000 Kb en la carpeta home
Mkdisk  &size->32 &path->/home/user/  \^
&name->Disco1.dsk %uniT&>k
#Se crearán carpetas si no existen
mkdisk &SiZe->8 &pAth->”/home/mis discos/DISCO  Prueba/” \^
&namE->Disco_3.dsk
#Creará un disco de 16 Mb ya que no hay parámetro unit
Mkdisk &size->16 &path->”/home/mis discos/” &NaMe->Disco4.dsk
```

RMDISK

Parametros

- **Path:** 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:

```
#Eliminar Disco_4.dsk
rmDisk   &path&>"/home/mis discos/Disco_4.dsk"
```

FDISK

Parametros

- **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.
  NOTA: La dirección de la carpeta únicamente llevará comillas dobles (") cuando algún name dentro de l dirección contenga espacios, de lo contrario podría venir sin comillas.
- **Type:** Indicará que tipo de partición se creará. Ya que es opcional, se tomará como primaria en caso de que no se indique. Podrá tener los siguientes valores:
  - **P:** 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.
- **Fit:** Indicará el ajuste que utilizará la partición para asignar espacio. Podrá tener los siguientes valores:
  - **BF:** Indicará el mejor ajuste Best Fit
  - **FF:** Utilizará el primer ajuste First Fit
  - **WF:** Utilizará el peor ajuste Worst Fit
- **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.
  - **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.
- **Name:** Indicará el name de la partición. El name 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. Tomará el parámetro %UNIT para las unidades a agregar o eliminar. En el caso de agregar espacio, deberá comprobar que exista espacio libre después de la partición. En el caso de quitar espacio se debe comprobar que quede espacio en La partición (no espacio negativo).

Ejemplo:

```
#Crear una Particion primaria llamada Particion1 de 72kb
#Con el peor ajuste y con asignacion Indexada en
el 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 %Allocation->E
#Crea una partición lógica con el mejor ajuste, llamada Particion3,y
#de 1 Mb en el Disco3 y asignación
```

```
contigua fdisk &sizE->1 %tipo->L
%unit->M %fit->BF \^
%Allocation->C &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

Parametros

- **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 name de la partición a cargar. Si no existe debe mostra 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→vdc1mount
&path→/home/Disco1.dsk &name→Part2 #id→
mount &path->/home/Disco1.dsk &name->Part2
#id->vda2
```

UNMOUNT

Parametros

- **Idn:** Especifica una lista de id de las particiones que se Desmontará, Si no existe algun ID mostrara 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→vdc
```

MKFS

Parametros

- **Id:** 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:** Indicará que tipo de formato se realizará. Ya que es opcional, se tomará como un formato completo si no se especifica esta opción. Podrá tener los siguientes valores:
  - **Fast:** en este caso se realizará un formato rápido.
  - **Full:** en este caso se realizará un formato completo.

Ejemplo:

```
#Realiza un formateo rápido con LWH de la "Particion 1" del
#Disco1.dsk


mkfs &id->vda1 &type->fast

#Realiza un formateo completo con LWH de
Particion2 en  Disco2.dsk

mkfs &id->vdb1

#Agrega 500 Kb a Particion3 en
Disco3.dsk
mkfs &add->500 &id->vdc1

#Agrega 1 Mb a Particion3 en

Disco3.dsk

Mkfs &add->1 &id->vdc1 &unit-

>m
```

LOGIN

Parametros

- **Usr:** Especifica el nombre del usuario que iniciará sesión. Si no se encuentra en la primera partición mostrará un mensaje indicando que el usuario no existe. En este caso si distinguirá mayúsculas de minúsculas.
- **Pwd:** Indicará la contraseña del usuario, si no coincide debe mostrar un mensaje de autenticación fallida. Distinguirá entre mayúsculas y minúsculas.
- **Id:** Indica el id de la partición a la cual se desea obtener acces Únicamente se puede obtener acceso a las particiones que se encuentren montadas en ese momento. De lo contrario la aplicación debe mostrar un mensaje de error indicando que la partición no se encuentra montada.

Ejemplo:

```
#Se loguea en el sistema como
usuario root login &usr->root &pwd-
>201020576 &id->vda1

login &usr->"mi usuario" &pwd->"mi pwd" /^
&id->vda2
```

LOGOUT

Parametros

No cuenta cuenta con parametros.

Ejemplo:

```
Logout
```

MKGROUP

Parametros

- **Id:** Especifica el id de la partición en la que se creará el grupo. Si no existe, debe mostrar un error.
- **Name:** Indicará el nombre que tendrá el grupo

Ejemplo:

```
#Crea el grupo usuarios en la
partición vda1
mkgrp &id->vda1 &nombre-
>"usuarios"

#Debe mostrar mensaje de error ya que
el grupo ya existe
mkgrp &id->vda1 &nombre->"usuarios"
```

RMGROUP

Parametros

- **Id:** Especifica el id de la partición en la que se creará el grupo. Si no existe, debe mostrar un error.
- **Name:** Indicará el nombre que tendrá el grupo

Ejemplo:

```
#Elimina el grupo de usuarios en la
partición vda1 rmgrp &id->vda1
&nombre->"usuarios"

#Debe mostrar mensaje de error ya que el grupo no existe porque ya fue eliminado
rmgrp &id->vda1 &nombre->"usuarios"
```

MKUSR

Parametros

- **Id:** Indicará la contraseña del usuario, si no coincide debe mostrar un

mensaje de autenticación fallida. Distinguirá entre mayúsculas y minúsculas.

- **Usr:** Indicará el nombre del usuario a crear, si ya existe, deberá mostrar un error indicando que ya existe el usuario. Máximo: 10
- caracteres.
- **Pwd:** Indicará la contraseña del usuario. Máximo: 10 caracteres.
- **Grp:** Indicará el grupo al que pertenece el usuario. Debe de existir en l partición en la que se está creando el usuario, si no debe mostrar un mensaje de error. Máximo: 10 caracteres.

Ejemplo:

```
#Crea el grupo usuarios en la partición vda1
Mkusr &id->vda1 &nombre->"user1" –grp->usuarios –pwd->usuario
```

RMUSR

Parametros

- **Id:** Especifica el id de la partición en la que se eliminará el usuario. Si no existe, debe mostrar un error.
- **Usr:** Indicará el nombre del usuario a eliminar, si no existe, deberá mostrar un error indicando que el usuario no existe.

Ejemplo:

```
#Elimina el usuario user1 de la
partición rmusr &id->vda1 &usr-
>user
```

CHMOD

Parametros

- **Id:** Especifica el id de la partición en la que se encuentra el archivo o carpeta a la que se le cambiarán los permisos.
- **Path:** Este parámetro será la ruta en la que se encuentra el archivo O carpeta a la que se le cambiarán los permisos.
- **UGO:** Indica los permisos que tendrán los usuarios. Serán tres números, uno para el Usuario, el siguiente para el Grupo al que pertenece el usuario y el último para Otros usuarios fuera del grupo. Cada número tendrá los valores desde el 0 al 7.
- **R:** 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

Ejemplo:

```
#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 &id->vda1 &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
```

MKFILE

Parametros

- **Id:** Especifica el id de la partición en la que se creará el archivo. Si n existe debe mostrar error.
- **Path:** 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:** Si se utiliza este parámetro y las carpetas especificadas por el parámetro –path no existen, entonces deben crearse las carpetas padres.
- **Size:** Este parámetro indicará el tamaño en bytes del archivo, el contenido será el abecedario cuantas veces sea necesario.
- **Cont:** Indicará la Cadena que sera Escrita dentro del Archivo.

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"
```

CAT

Parametros

- **Id:** Especifica el id de la partición en la que se leerá el archivo. Si no existe debe mostrar error.
- **Filen:** 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 o no tiene permiso de lectura, debe mostrarse un mensaje de error.

Ejemplo:

```
#Lee el archivo a.txt En la terminal debería mostrar el contenido, en #este
ejemplo 01234567890123
Cat &file1->"/home/user/docs/a.txt" &Id->vdb1
```

```
#enlazara los archivos a.txt (datos archivo a) b.txt (01234567890123) #c.txt
(0123) y debería mostrar el contenido siguiente, cada archivo #va separado
por salto de línea
# datos archivo a
# 01234567890123
# 0123
Cat &file1->"/home/a.txt" &Id->vdb1 &file2->"/home/b.txt" \ &file3-
>"/home/c.txt"
```

RM

Parametros

- **Id:** Especifica el id de la partición en la que se eliminará el archivo
- **Path:** Este parámetro será la ruta del archivo o carpeta que se eliminará. Deberá encerrarse entre comillas.
- **RF:** Permite hacer borrados recursivos. Con esta opción se pueden borrar sin importar los permisos siempre y cuando sea el usuario root.

Ejemplo:

```
#Elimina el archivo a.txt, b.txt muestra
error si no tiene permiso
rm &PatH->"/home/user/docs/a.txt" &Id-
>vdb1
rm &PatH->"/home/user/docs/b.txt" &Id->vdb1
#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
rm &PatH->"/home/user" &Id->vdb1
#Elimina la carpeta user y todo su contenido (docs, a.txt) sin importar los
permisos(logueado como usuario root)
Rm &Id->vdb1 &path->"/home/user" +RF
```

EDIT

Parametros

- **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. Deberá encerrarse entre comillas. Si no existe, debe mostrar un Mensaje de error.
- **Size:** Este parámetro indicará el nuevo tamaño en bytes del a archivo, el contenido serán el abecedario cuantas veces sea necesario. Si es negativo debe mos trar error.

- **Cont:** Indicará una cadena la cual sera el nuevo contenido del Archivo.

Ejemplo:

```
#Modifica el archivo a.txt
#El tamaño del archivo es de 22 bytes
#El contenido sería: abcdefghijklmnopqrstu
Edit %SIZE->22 &id->vdb1 &PatH->"/home/user/docs/a.txt”
#Modifica el archivo b.txt
#El contenido del archivo será el mismo que el archivo c.txt
#que se encuentra en el disco duro de la computadora.
edit &id->vda1 &path->"/home/user/docs/b.txt"  %cont ->"/home/Documents/c.txt"

#Modifica nuevamente el archivo b.txt
edit &id->vda1 &path->"/home/user/docs/b.txt" &cont ->"/home/Documents/d.txt"
```

REN

Parametros

- **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. Deberá encerrarse entre comillas. Si no existe el archivo o carpeta o no tiene permisos de escritura sobre la carpeta o archivo, debe mostrarse un mensaje de error.
- **Name:** 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. Deberá encerrarse entre comillas.

Ejemplo:

```
#Cambia el nombre del archivo a.txt a b1.txt
ren &PatH->"/home/user/docs/a.txt" &Id->vdb1 &nombre->"b1.txt"
#Debera mostrar error ya que el archivo b1.txt ya existe
ren &PatH->"/home/user/docs/c.txt" &Id->vdb1 &nombre->"b1.txt"
```

MKDIR

Parametros

- **Id:** Especifica el id de la partición en la que se creará la carpeta. Si No existe debe mostrar error.
- **Path:** 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:** Si se utiliza este parámetro y las carpetas padres en el Parámetro path no existen, entonces deben crearse.

Ejemplo:

```
#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 2016"
#La carpeta padre ya debe existir
Mkdir &ID->vda1 &path->"/home/mis documentos/archivos 2016"
```

CP

Parametros

- **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.
- **Destiny:** Este parámetro será la ruta de carpeta a la que se copiara el archivo o carpeta. Debe tener permiso de escritura sobre la carpeta. Debe mostrar un mensaje de error si no tiene permiso de escritura o si la carpeta no existe.

Ejemplo:

```
#/
#|_home     #664
#  |_user     #664
#  | |_documents  #664
#  |   |_a.txt  #664
#  |   |_b.txt #224
#  |_images #664
#Copia documents a images
cp   &id->vda2   &Path->"/home/user/documents"   &iddestiny->vda2   &dest-
>"/home/images"
```

MV

Parametros

- **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.
- **Destiny:** Este parámetro será la ruta de carpeta a la que se moverá el archivo o carpeta. Debe tener permiso de escritura sobre la carpeta. Debe mostrar un mensaje de error si no tiene permiso de escritura 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 &id->vda1 &iddestiny->vda1 &Path->"/home/user/documents" \^
&destiny->"/home/images"
```

FIND

Parametros

- **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. Debe tener permisos de lectura en los archivos que buscará.
- **Name:** Indica el nombre del archivo o carpeta que se está buscando. Debe aceptar los caracteres especiales definidos anteriormente

Ejemplo:

```
#Arbol actual
#/
#|_home     #664
#  |_user     #664
#  | |_a.txt   #664
#  | |_b.txt   #420
#  |_images #664
#Busca los archivos que tengan una letra como nombre
#y cualquier extensión
find &id->vda1 &Path->"/home" &nombre->"?.*"
```

CHOWN

Parametros

- **Id:** Especifica el id de la partición en la que se encuentra el archivo o carpeta a la que se le cambiará el propietario.
- **Path:** 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:** 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:** Nombre del nuevo propietario del archivo o carpeta. Si no Existe o está eliminado debe mostrar error.

Ejemplo:

```
#Cambia el propietario de la carpeta home
recursivamente chown &id->vda1 &path-
>"/home" +R &usr->user2

#Cambia los permisos de la carpeta
home   chown   &id->vda1   &path-
>"/home" &usr->user1
```

CHGRP

Parametros

- **Usr:** Especifica el nombre del usuario al que se le cambiará de grupo. Si no existe debe mostrar un error.
- **Grp:** Contendrá el nombre del nuevo grupo al que pertenece El usuario. Si no existe o está eliminado debe mostrar un error.

Ejemplo:

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

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

LOSS

Parametros

- **Id:** Especifica el id de la partición en la que se desea simular la perdida.

Ejemplo:

```
#Simulando la perdida del sistema de archivos de la partición1 Loss &id->vda1
```

RECOVERY

Parametros

- **Id:** Especifica el id de la partición en la que se desea recuperar la partición que simulo la perdida la perdida.

Ejemplo:

```
#Recuperando el sistema de archivos EXT3 de la partición 1
Recovery &id->vda1
```

REP

Parametros

- **Name:** Nombre del reporte a generar. Tendra los siguientes valores:
  - MBR
  - DISK
  - INODE
  - SB
  - BM_ARBDIR
  - BM_DETDIR
  - BM_INODE
  - BM_BLOCK
  - BITACORA
  - DIRECTORIO
  - TREE_FILE
  - TREE_DIRECTORIO
  - TREE_COMPLETE

- LS

- **Path:** Indicara una carpeta y el nombre que tendra el reporte. Si no existe la carpeta, debera crearla. Si lleva espacios se encerrara entre comillas.
- **Id:** Indica el id de la partición a que se utilizara.
- **Ruta:** Funcionara para el reporte file y ls. Sera el nombre de directorio o archivo del que se mostrara el reporte. Sino existe muestra error.

Ejemplo:

```
rep &id->vda2 &Path="/home/user/reports/reporte 2.pdf" &nombre=bm_arbdir –
ruta="/home/mis  documentos"
rep &id->vda1 &Path="/home/user/reports/reporte 3.jpg" &nombre=archive_directorio
```

EXEC

Parametros

- **Path:** Especifica el nombre del script que se va a ejecutar.

Ejemplo:

```
#ejecuta el script
exec  &path->/home/Desktop/calificacion.sh
```

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.

Estructuras utilizadas

```
typedef struct PARTITION
{
   char       part_status;
   char       part_type;
   char       part_fit;
   int        part_start;
   int        part_size;
   char       part_name[16];
}PARTITION;

typedef struct MBR
{
   int        mbr_tamano;
   time_t     mbr_fecha_creacion;
   int        mbr_disk_signature;
   PARTITION  mbr_partition_1;
   PARTITION  mbr_partition_2;
   PARTITION  mbr_partition_3;
   PARTITION  mbr_partition_4;
}MBR;

typedef struct EBR
{
   char       part_status;
   char       part_fit;
   int        part_start;
```

```
    int         part_size;
    int         part_next;
    char        part_name[16];
}EBR;

typedef struct SUPERBOOT

{
    char        sb_nombre_hd[16];
    int         sb_arbol_virtual_count;
    int         sb_detalle_directory_count;
    int         sb_inodos_count;
    int         sb_bloques_count;
    int         sb_arbol_virtual_free;
    int         sb_detalle_directory_free;
    int         sb_inodos_free;
    int         sb_bloques_free;
    time_t      sb_date_creacion;
    time_t      sb_date_ultimo_montaje;
    int         sb_montaje_count;
    int         sb_ap_bitmap_arbol_directorio;
    int         sb_ap_arbol_directory;
    int         sb_ap_bitmap_detalle_directory;
    int         sb_ap_detalle_directory;
    int         sb_ap_bitmap_table_inodo;
    int         sb_ap_table_inodo;
    int         sb_ap_bitmap_bloques;
    int         sb_ap_bloques;
    int         sb_ap_log;
    int         size_struct_arbol_directorio;
    int         size_struct_detalle_directorio;
    int         size_struct_inodo;
    int         size_struct_bloque;
    int         sb_first_free_bit_arbol_directorio;
    int         sb_first_free_bit_detalle_directorio;
    int         sb_first_free_bit_tabla_inodo;
    int         sb_first_free_bit_bloques;
    int         sb_magic_num;
}SUPERBOOT;

typedef struct AVD
{
    time_t      avd_fecha_creacion;
    char        avd_nombre_directorio[20];
    int         avd_ap_array_subdirectorios[6];
    int         avd_ap_detalle_directorio;
    int         avd_ap_arbol_virtual_directorio;
    int         avd_proper;
    int         avd_perm;
}AVD;

typedef struct BLOCK_FILE
```

```
{
    char        dd_file_nombre[20];
    int         dd_file_ap_inodo;
    time_t      dd_file_date_creation;
    time_t      dd_file_date_modificacion;
}BLOCK_FILE;

typedef struct DD
{
    BLOCK_FILE  dd_array_files[5];
    int         dd_ap_detalle_directorio;
}DD;

typedef struct BLOCK_DATA
{
    char        bd_data[25];
}BLOCK_DATA;

typedef struct INODO
{
    int         i_count_inodo;
    int         i_size_archivo;
    int         i_count_bloques_asignados;
    int         i_array_bloques[4];
    int         i_ap_indirecto;
    int         i_id_proper;
    int         i_perm;
}INODO;

typedef struct BITACORA{
    int         log_tipo_operacion;
    char        log_tipo;
    char        log_nombre[256];
    char        log_contenido[256];
    time_t      log_fecha;
    char        id[10];
    bool        recursividad;
}BITACORA;
```

Metodos de set y get y constructores

```
MBR Procedures::createMBR(int size, QString unit)
{
    MBR retorno;
    PARTITION nula;
    nula.part_status = '0';
    nula.part_type = '0';
    nula.part_fit = '0';
    nula.part_start = 0;
```

```
    nula.part_size = 0;
    for (int i=0;i<16;i++)
        nula.part_name[i]='\0';

    if(unit==""||unit.toUpper()=="M")
        retorno.mbr_tamano = size*1024*1024;
    else if(unit.toUpper()=="K")
        retorno.mbr_tamano = size*1024;

    retorno.mbr_fecha_creacion = time(nullptr);

    srand(static_cast<unsigned int>(time(nullptr)));
    retorno.mbr_disk_signature = qrand();

    retorno.mbr_partition_1 = nula;
    retorno.mbr_partition_2 = nula;
    retorno.mbr_partition_3 = nula;
    retorno.mbr_partition_4 = nula;

    return retorno;
}
void Procedures::setMBR(QString path, MBR mbr,int pos)
{
    ifstream exist(path.toUtf8(),ios::in);

    if(exist.good())
    {

        ofstream file( path.toUtf8(), ios::in | ios::out | ios::binary);

        writeLine("Insertando el MBR espere...");

        file.seekp(pos);
        file.write(reinterpret_cast<char*>(&mbr),sizeof(MBR));
        file.close();
        exist.close();
        writeLine("se inserto el MBR con éxito");
    }
    else
        writeError("Disco no existe en la ubicación");
}
MBR Procedures::getMBR(QString path,int pos){
    MBR retorno;
    ifstream file(path.toUtf8());
    if(file.is_open())
    {
        file.seekg(pos,ios::beg);
        file.read(reinterpret_cast<char*>(&retorno),sizeof(MBR));
        file.close();
    }
    return retorno;
```

```
}
```

Expresiones regulares

```
RW_Add          "add"
RW_Cat          "cat"
RW_Chgrp        "chgrp"
RW_Chmod        "chmod"
RW_Chown        "chown"
RW_Cont         "cont"
RW_Cp           "cp"
RW_Delete       "delete"
RW_Destiny      "destiny"
RW_Edit         "edit"
RW_Exec         "exec"
RW_Fdisk        "fdisk"
RW_File         "file"
RW_Filen        "file"[0-9]+
RW_Find         "find"
RW_Fit          "fit"
RW_Grp          "grp"
RW_Id           "id"
RW_Idn          "id"[0-9]+
RW_Iddestiny    "iddestiny"
RW_Login        "login"
RW_Logout       "logout"
RW_Loss         "loss"
RW_Mkdir        "mkdir"
RW_Mkdisk       "mkdisk"
RW_Mkfile       "mkfile"
RW_Mkfs         "mkfs"
RW_Mkgrp        "mkgrp"
RW_Mkusr        "mkusr"
RW_Mount        "mount"
RW_Mv           "mv"
RW_Name         "name"
RW_P            "p"
RW_Path         "path"
RW_Pause        "pause"
RW_Pwd          "pwd"
RW_R            "r"
RW_Recovery     "recovery"
RW_Ren          "ren"
RW_Rep          "rep"
RW_Rf           "rf"
RW_Rm           "rm"
RW_Rmdisk       "rmdisk"
RW_Rmgrp        "rmgrp"
RW_Rmusr        "rmusr"
RW_Size         "size"
RW_Type         "type"
```

```
RW_Ugo              "ugo"
RW_Unit             "unit"
RW_Unmount          "unmount"
RW_Usr              "usr"
RW_Ruta             "ruta"
SYM_Arrow           "->"
SYM_Ampersand       "&"
SYM_Hyphen          "-"
RE_Delete           "fast"|"full"
RE_Fit              "bf"|"ff"|"wf"
RE_Id               [a-zA-Z_0-9]+
RE_Name
"mbr"|"disk"|"inode"|"sb"|"bm_arbdir"|"bm_detdir"|"bm_inode"|"bm_block"|"bitacora"|"directorio
"|"tree_file"|"tree_directorio"|"tree_complete"|"ls"
RE_Name2            [a-zA-Z_0-9]+[\.][a-zA-Z0-9_]+
RE_Number           [0-9]+
RE_Path             ([\/]?[a-zA-Z0-9_]+)*([\/]|[\.][a-zA-Z0-9_]+)?
RE_Type             "e"|"l"
RE_Unit             "b"|"k"|"m"
```

Gramatica utilizada en bison

```
Gramática

    0 $accept: Start $end

    1 Start: NT_Instructions

    2 NT_Instructions: NT_Instructions NT_Instruction
    3                | NT_Instruction

    4 NT_Instruction: NT_Cat
    5               | NT_Chgrp
    6               | NT_Chmod
    7               | NT_Chown
    8               | NT_Cp
    9               | NT_Edit
   10               | NT_Exec
   11               | NT_Fdisk
   12               | NT_Find
   13               | NT_Login
   14               | NT_Logout
   15               | NT_Loss
   16               | NT_Mkdir
   17               | NT_Mkdisk
   18               | NT_Mkfile
   19               | NT_Mkfs
   20               | NT_Mkgrp
   21               | NT_Mkusr
```

```
22              | NT_Mount
23              | NT_Mv
24              | NT_Pause
25              | NT_Recovery
26              | NT_Rm
27              | NT_Ren
28              | NT_Rep
29              | NT_Rmdisk
30              | NT_Rmgrp
31              | NT_Rmusr
32              | NT_Unmount

33 NT_Cat: RW_Cat NT_Parameterscat

34 NT_Parameterscat: NT_Parameterscat SYM_Ampersand NT_Parametercat
35               | SYM_Ampersand NT_Parametercat

36 NT_Parametercat: RW_Id SYM_Arrow RE_Id
37              | RW_Filen SYM_Arrow RE_String
38              | RW_Filen SYM_Arrow RE_Path

39 NT_Chgrp: RW_Chgrp NT_Parameterschgrp

40 NT_Parameterschgrp: NT_Parameterschgrp SYM_Ampersand NT_Parameterchgrp
41                   | SYM_Ampersand NT_Parameterchgrp

42 NT_Parameterchgrp: RW_Usr SYM_Arrow RE_Id
43                  | RW_Usr SYM_Arrow RE_String
44                  | RW_Grp SYM_Arrow RE_Id
45                  | RW_Grp SYM_Arrow RE_String

46 NT_Chmod: RW_Chmod NT_Parameterschmod

47 NT_Parameterschmod: NT_Parameterschmod SYM_Ampersand NT_Parameterchmod
48                   | SYM_Ampersand NT_Parameterchmod

49 NT_Parameterchmod: RW_Path SYM_Arrow RE_Path
50                  | RW_Path SYM_Arrow RE_String
51                  | RW_Id SYM_Arrow RE_Id
52                  | RW_Ugo SYM_Arrow RE_Number
53                  | RW_R

54 NT_Chown: RW_Chown NT_Parameterschown

55 NT_Parameterschown: NT_Parameterschown SYM_Ampersand NT_Parameterchown
56                   | SYM_Ampersand NT_Parameterchown

57 NT_Parameterchown: RW_Path SYM_Arrow RE_Path
58                  | RW_Path SYM_Arrow RE_String
59                  | RW_Id SYM_Arrow RE_Id
60                  | RW_R
61                  | RW_Usr SYM_Arrow RE_Id
```

```
62              | RW_Usr SYM_Arrow RE_String

63 NT_Cp: RW_Cp NT_Parameterscp

64 NT_Parameterscp: NT_Parameterscp SYM_Ampersand NT_Parametercp
65              | SYM_Ampersand NT_Parametercp

66 NT_Parametercp: RW_Path SYM_Arrow RE_Path
67              | RW_Path SYM_Arrow RE_String
68              | RW_Id SYM_Arrow RE_Id
69              | RW_Destiny SYM_Arrow RE_Path
70              | RW_Destiny SYM_Arrow RE_String

71 NT_Edit: RW_Edit NT_Parametersedit

72 NT_Parametersedit: NT_Parametersedit SYM_Ampersand NT_Parameteredit
73              | SYM_Ampersand NT_Parameteredit

74 NT_Parameteredit: RW_Path SYM_Arrow RE_Path
75              | RW_Path SYM_Arrow RE_String
76              | RW_Id SYM_Arrow RE_Id
77              | RW_Size SYM_Arrow RE_Number
78              | RW_Cont SYM_Arrow RE_String

79 NT_Exec: RW_Exec SYM_Ampersand RW_Path SYM_Arrow RE_Path
80      | RW_Exec SYM_Ampersand RW_Path SYM_Arrow RE_String

81 NT_Fdisk: RW_Fdisk NT_Parametersfdisk

82 NT_Parametersfdisk: NT_Parametersfdisk SYM_Ampersand NT_Parameterfdisk
83              | SYM_Ampersand NT_Parameterfdisk

84 NT_Parameterfdisk: RW_Size SYM_Arrow RE_Number
85              | RW_Unit SYM_Arrow RE_Unit
86              | RW_Path SYM_Arrow RE_Path
87              | RW_Path SYM_Arrow RE_String
88              | RW_Type SYM_Arrow RE_Type
89              | RW_Type SYM_Arrow RW_P
90              | RW_Fit SYM_Arrow RE_Fit
91              | RW_Delete SYM_Arrow RE_Delete
92              | RW_Name SYM_Arrow RE_String
93              | RW_Name SYM_Arrow RE_Id
94              | RW_Add SYM_Arrow RE_Number
95              | RW_Add SYM_Arrow SYM_Hyphen RE_Number

96 NT_Find: RW_Find NT_Parametersfind

97 NT_Parametersfind: NT_Parametersfind SYM_Ampersand NT_Parameterfind
98              | SYM_Ampersand NT_Parameterfind

99 NT_Parameterfind: RW_Path SYM_Arrow RE_Path
100             | RW_Path SYM_Arrow RE_String
```

```
101             | RW_Id SYM_Arrow RE_Id
102             | RW_Name SYM_Arrow RE_Name2
103             | RW_Name SYM_Arrow RE_String

104 NT_Login: RW_Login NT_Parameterslogin

105 NT_Parameterslogin: NT_Parameterslogin SYM_Ampersand NT_Parameterlogin
106              | SYM_Ampersand NT_Parameterlogin

107 NT_Parameterlogin: RW_Usr SYM_Arrow RE_Id
108             | RW_Usr SYM_Arrow RE_String
109             | RW_Pwd SYM_Arrow RE_Id
110             | RW_Pwd SYM_Arrow RE_String
111             | RW_Pwd SYM_Arrow RE_Number
112             | RW_Id SYM_Arrow RE_Id

113 NT_Logout: RW_Logout

114 NT_Loss: RW_Loss SYM_Ampersand RW_Id SYM_Arrow RE_Id

115 NT_Mkdir: RW_Mkdir NT_Parametersmkdir

116 NT_Parametersmkdir: NT_Parametersmkdir SYM_Ampersand NT_Parametermkdir
117              | SYM_Ampersand NT_Parametermkdir

118 NT_Parametermkdir: RW_Path SYM_Arrow RE_Path
119             | RW_Path SYM_Arrow RE_String
120             | RW_Id SYM_Arrow RE_Id
121             | RW_P

122 NT_Mkdisk: RW_Mkdisk NT_Parametersmkdisk

123 NT_Parametersmkdisk: NT_Parametersmkdisk SYM_Ampersand NT_Parametermkdisk
124              | SYM_Ampersand NT_Parametermkdisk

125 NT_Parametermkdisk: RW_Size SYM_Arrow RE_Number
126             | RW_Unit SYM_Arrow RE_Unit
127             | RW_Path SYM_Arrow RE_String
128             | RW_Path SYM_Arrow RE_Path
129             | RW_Name SYM_Arrow RE_Name2
130             | RW_Name SYM_Arrow RE_String

131 NT_Mkfile: RW_Mkfile NT_Parametersmkfile

132 NT_Parametersmkfile: NT_Parametersmkfile SYM_Ampersand NT_Parametermkfile
133              | SYM_Ampersand NT_Parametermkfile

134 NT_Parametermkfile: RW_Path SYM_Arrow RE_Path
135             | RW_Path SYM_Arrow RE_String
136             | RW_Id SYM_Arrow RE_Id
137             | RW_P
138             | RW_Size SYM_Arrow RE_Number
```

```
139                 | RW_Cont SYM_Arrow RE_String

140 NT_Mkfs: RW_Mkfs NT_Parametersmkfs

141 NT_Parametersmkfs: NT_Parametersmkfs SYM_Ampersand NT_Parametermkfs
142                 | SYM_Ampersand NT_Parametermkfs

143 NT_Parametermkfs: RW_Id SYM_Arrow RE_Id
144                 | RW_Type SYM_Arrow RE_Delete

145 NT_Mkgrp: RW_Mkgrp NT_Parametersmkgrp

146 NT_Parametersmkgrp: NT_Parametersmkgrp SYM_Ampersand NT_Parametermkgrp
147                 | SYM_Ampersand NT_Parametermkgrp

148 NT_Parametermkgrp: RW_Id SYM_Arrow RE_Id
149                 | RW_Name SYM_Arrow RE_Id
150                 | RW_Name SYM_Arrow RE_String

151 NT_Mkusr: RW_Mkusr NT_Parametersmkusr

152 NT_Parametersmkusr: NT_Parametersmkusr SYM_Ampersand NT_Parametermkusr
153                 | SYM_Ampersand NT_Parametermkusr

154 NT_Parametermkusr: RW_Usr SYM_Arrow RE_Id
155                 | RW_Usr SYM_Arrow RE_String
156                 | RW_Id SYM_Arrow RE_Id
157                 | RW_Pwd SYM_Arrow RE_Id
158                 | RW_Pwd SYM_Arrow RE_String
159                 | RW_Pwd SYM_Arrow RE_Number
160                 | RW_Grp SYM_Arrow RE_Id
161                 | RW_Grp SYM_Arrow RE_String

162 NT_Mount: RW_Mount NT_Parametersmount

163 NT_Parametersmount: NT_Parametersmount SYM_Ampersand NT_Parametermount
164                 | SYM_Ampersand NT_Parametermount

165 NT_Parametermount: RW_Path SYM_Arrow RE_Path
166                 | RW_Path SYM_Arrow RE_String
167                 | RW_Name SYM_Arrow RE_String
168                 | RW_Name SYM_Arrow RE_Id

169 NT_Mv: RW_Mv NT_Parametersmv

170 NT_Parametersmv: NT_Parametersmv SYM_Ampersand NT_Parametermv
171               | SYM_Ampersand NT_Parametermv

172 NT_Parametermv: RW_Path SYM_Arrow RE_Path
173               | RW_Path SYM_Arrow RE_String
174               | RW_Destiny SYM_Arrow RE_Path
175               | RW_Destiny SYM_Arrow RE_String
```

```
176             | RW_Id SYM_Arrow RE_Id

177 NT_Pause: RW_Pause

178 NT_Recovery: RW_Recovery SYM_Ampersand RW_Id SYM_Arrow RE_Id

179 NT_Ren: RW_Ren NT_Parametersren

180 NT_Parametersren: NT_Parametersren SYM_Ampersand NT_Parameterren
181             | SYM_Ampersand NT_Parameterren

182 NT_Parameterren: RW_Path SYM_Arrow RE_Path
183             | RW_Path SYM_Arrow RE_String
184             | RW_Id SYM_Arrow RE_Id
185             | RW_Name SYM_Arrow RE_Name2
186             | RW_Name SYM_Arrow RE_String

187 NT_Rep: RW_Rep NT_Parametersrep

188 NT_Parametersrep: NT_Parametersrep SYM_Ampersand NT_Parameterrep
189             | SYM_Ampersand NT_Parameterrep

190 NT_Parameterrep: RW_Name SYM_Arrow RE_Name
191             | RW_Name SYM_Arrow RW_File
192             | RW_Path SYM_Arrow RE_Path
193             | RW_Path SYM_Arrow RE_String
194             | RW_Id SYM_Arrow RE_Id
195             | RW_Ruta SYM_Arrow RE_Path
196             | RW_Ruta SYM_Arrow RE_String

197 NT_Rm: RW_Rm NT_Parametersrm

198 NT_Parametersrm: NT_Parametersrm SYM_Ampersand NT_Parameterrm
199             | SYM_Ampersand NT_Parameterrm

200 NT_Parameterrm: RW_Path SYM_Arrow RE_Path
201             | RW_Path SYM_Arrow RE_String
202             | RW_Id SYM_Arrow RE_Id
203             | RW_Rf

204 NT_Rmdisk: RW_Rmdisk SYM_Ampersand RW_Path SYM_Arrow RE_String
205         | RW_Rmdisk SYM_Ampersand RW_Path SYM_Arrow RE_Path

206 NT_Rmgrp: RW_Rmgrp NT_Parametersrmgrp

207 NT_Parametersrmgrp: NT_Parametersrmgrp SYM_Ampersand NT_Parameterrmgrp
208               | SYM_Ampersand NT_Parameterrmgrp

209 NT_Parameterrmgrp: RW_Name SYM_Arrow RE_Id
210             | RW_Name SYM_Arrow RE_String
211             | RW_Id SYM_Arrow RE_Id
```

```
212 NT_Rmusr: RW_Rmusr NT_Parametersrmusr

213 NT_Parametersrmusr: NT_Parametersrmusr SYM_Ampersand NT_Parameterrmusr
214                 | SYM_Ampersand NT_Parameterrmusr

215 NT_Parameterrmusr: RW_Usr SYM_Arrow RE_Id
216                 | RW_Usr SYM_Arrow RE_String
217                 | RW_Id SYM_Arrow RE_Id

218 NT_Unmount: RW_Unmount NT_Parametersunmount

219 NT_Parametersunmount: NT_Parametersunmount SYM_Ampersand NT_Parameterunmount
220                   | SYM_Ampersand NT_Parameterunmount

221 NT_Parameterunmount: RW_Idn SYM_Arrow RE_Id
```