UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FCULTAD DE INGENIERIA
INGENIERIA EN CIENCIAS Y SISTEMAS
MANEJO E IMPLEMENTACION DE ARCHIVOS
1ER SEMESTRE 2020

MANUAL TÉCNICO

Eddy Arnoldo Reyes Hernandez – 201612326

DESCRIPCION DE FUNCIONES IMPORTANTES DEL PROGRAMA

int **rDisk**(char *pathDisco, char *pathImagen)

Genera el reporte de tipo DISK, realiza todas las validaciones necesarias para poder generar el archivo graphviz con el esquema correcto.

int **rMBR**(char *pathDisco, char *pathImagen)

Genera el reporte de tipo MBR, realiza todas las validaciones necesarias para poder generar el archivo graphviz con el esquema correcto.

int **verificarParametrosObligatorios**(struct comando *command)

Se encarga de verificar que los parametros obligatorios que se le deben pasar a una instruccione fueron realizados

void **verificarParametrosOpcionales**(struct comando *command)

Se encarga de setear los parametros que son opcionales a las instrucciones en caso de que no se hayan especificado.

Mbr* **crearMBR**(struct comando *command)

Se encarga de crear un nuevo MBR desde cero, este mbr será el que se le asignará a un nuevo disco creado con el comando MKDISK.

void **crearDirectorio**(struct comando *command)

Esta instruccion se encarga de tomar el PATH establecido en el comando y generarl los directorios en caso de que no existieran.

int **crearArchivoBinario**(struct comando *command, Mbr *mbr)

Esta instruccion se encarga de crear un nuevo archivo. Este archivo será el disco creado por el comando MKDISK.

int **eliminarArchivoBinario**(struct comando *command)

Elimina el archivo indicado en el PATH del comando. Se utiliza para procesar la instruccion RMDISK.

int **crearParticion**(struct comando *command)

Esta funcion se encarga de procesar la ardua labor de ejecutar la opcion CREAR PARTICION de la instruccion FDISK.

int **crearParticionLogica**(struct comando *command, Mbr *mbr)

Tiene la mima funcionalidad que la anterior, pero esta opera únicamente con las particiones lógicas.

int **eliminarParticion**(struct comando *command, ListaMontaje *montajes)

Se encarga de eliminar una partición (Primaria, Extendida o Logica). Antes de eliminar, valida de que la particion no esté montada, de ser así la elminación no se lleva a cabo.

int **incrementarParticion**(struct comando *command)

Se encarga de realizar las validaciones necesarias para poder incrementar el tamaño de una particion.

int **decrementarParticion**(struct comando *command)

Se encarga de realizar las validaciones necesarias para poder decrementar el tamaño de una particion.

int **montarParticion**(struct comando *command, ListaMontaje *montajes)

Se encarga de efectuar el trabajo de la función MOUNT. La particion no debe de estar montada con anterioridad y debe existir.

int **desmontarParticion**(struct comando *command, ListaMontaje *montajes)

Se encarga de realizar el trabajo de la funcion UNMOUNT.

int **ejecutarScript**(struct comando *command, ListaMontaje *montajes)

Este comando se encarga de poder leer un archivo de scripts que está constituido por una lista de instrucciones consecutivas.

TIPOS DE DATOS UTILIZADOS

```
typedef struct time{
	int dia;
	int mes;
	int ano;
} Time;

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 mbr_fecha_creacion;
	int mbr_disk_signature;
	char disk_fit;
	Partition partitions[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;
```

LIBRERIAS UTILIZADAS

Se utilizó una amplia variedad de librerias pertenecientes a las que ofrece C++ como apis. Además de ellas también se utilizó el analizador sintáctico Bison en conjunto con el analizador lexico Flex para poder procesar el comando que se intrudia.

Los links de descarga de estas librerias son los siguientes:

http://gnuwin32.sourceforge.net/packages/bison.htm

http://gnuwin32.sourceforge.net/packages/flex.htm

ESTRUCTURAS DE DATOS UTILIZADAS

Se utilizaron dos estructuras de datos.

1. Lista de disponibles (Lista simplemente enlazada)
2. Lista de montaje (Lista de listas simplemente enlazadas)

LISTA DE DISPONIBLES

Esta lista es la encargada de la gestión y administración de las particiones pertenecientes a un disco. Crea, elimina y asigna el espacio de acuerdo a las estrategias de colocación indicadas en el disco (Primer ajuste, Peor ajuste, Mejor ajuste).

Esta lista es el motor central del software ya que se encarga basicamente de generar los nodos de espacio que esta disponible dentro del disco.

LISTA DE MONTAJE

Esta lista se encarga de ver el montaje de un disco en el sistema. No tiene mucha utilidad pero su función principal es la gestión de los identificadores de los discos montados. Es una lista de discos en la que no pueden haber nodos iguales y cada uno de esos nodos posee una lista de las particiones de ese disco que estan montadas en el sistema.

DETALLES TECNICOS

El software fue programado en un hibrido entre c/c++ . Apesar de que se utilizó el IDE QtCreator no se utilizó ninguna de las librerías que este posee.

El programa unicamente corre en distribuciones Linux.

La computadora sobre la que se desrrolló y se hicieron las pruebas cuenta con 8Gb de RAM, 150 Gb de espacio libre en disco duro y cuenta con un procesador AMD.