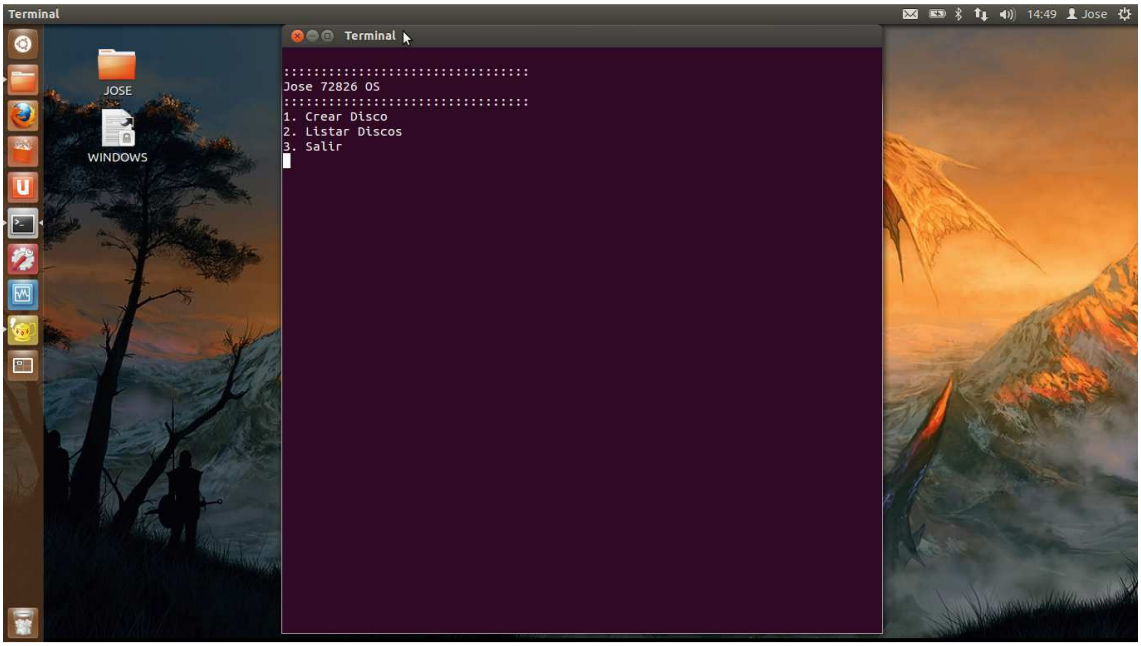
Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Manejo e implementación de Archivos
Sección A+

Proyecto 1
Manual técnico

Jose Pablo Godoy Linares
200915162
06/10/2012

Acerca de la aplicación

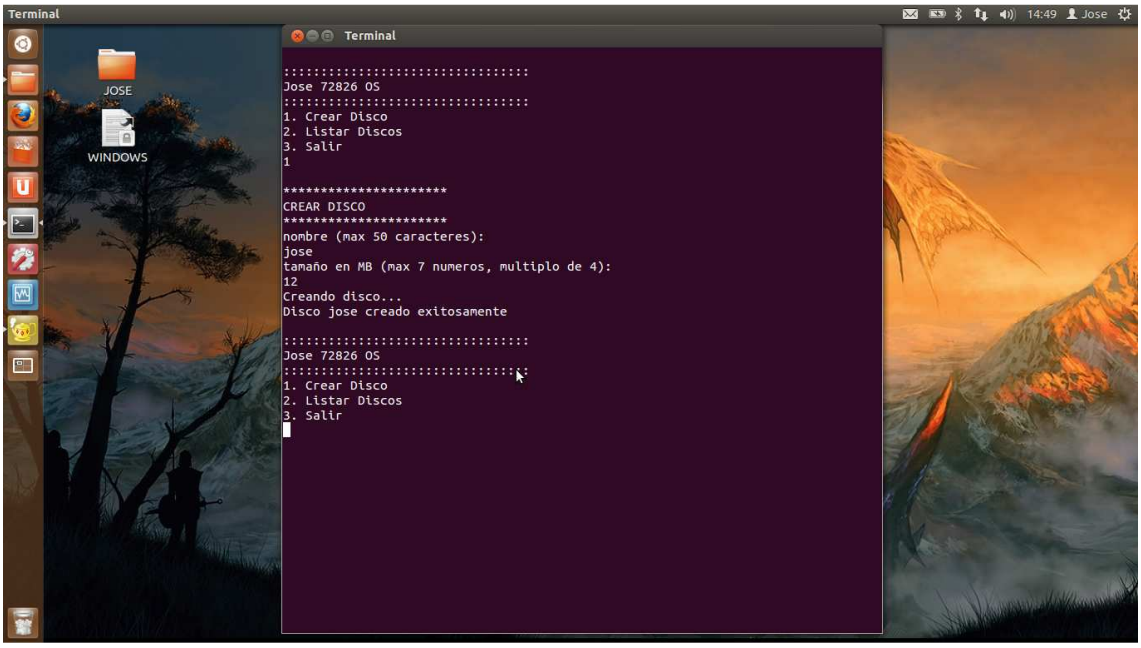
En el presente manual técnico explica la manera en cómo fue programada la practica 1 del curso de manejo e implementación de archivos, indicando paso a paso cómo hacer para ejecutar la aplicación, que opciones tiene disponibles, así como también explicando la estructura lógica que se utilizó para programar y las funciones y librerías propias de c que se utilizaron para poder mostrar el ejecutable.



Como se puede observar la aplicación fue construida bajo el sistema operativo GNU-Linux Ubuntu, la cual corre con la ayuda de la consola del mismo sistema, o bien bajo un ID.

Acerca de la aplicación

La práctica consiste en realizar una aplicación bajo el entorno GNU-Linux la cual deberá de ser capaz de crear discos utilizando archivos binarios para su implementación, la cual deberá de tener la opción de crear particiones y todas las opciones que este requiera, es decir teoría de particionamiento, así como también la capacidad de poder crear archivos y carpetas dentro de cada partición utilizando superbloques, mapa de bits, inodos y bloques. Entre las opciones, tiene que tener la capacidad de crear uno o varios discos, así como también seleccionar el disco con que se desea trabajar para poder crear, eliminar o agregar archivos o carpetas a sus respectivas particiones. En esta aplicación también se deberá de contar con reportes y funciones especiales para ambos sistemas de archivos, los cuales son FAT y EXT3.



Requerimientos del usuario

Para poder utilizar la aplicación, el usuario deberá de estar bajo el entorno GNU-Linux. Para poder editar el programa desde su código fuente, es necesario contar con los compiladores c/c++, para instalarlo solo debe de escribir el siguiente código en la terminal de Linux:

```
$ sudo aptitude install build-essential
```

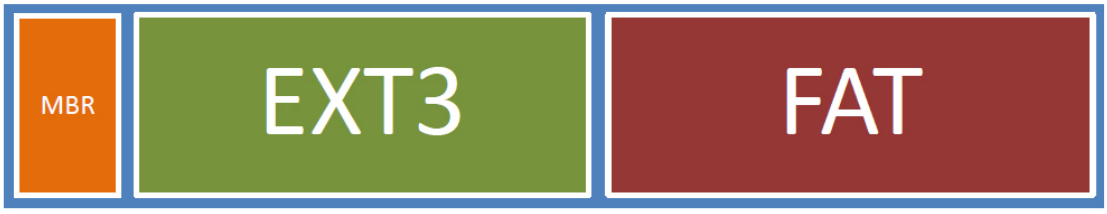
Con este paquete podrá correr los archivos ejecutables así como también editar su código fuente con el compilador de c.

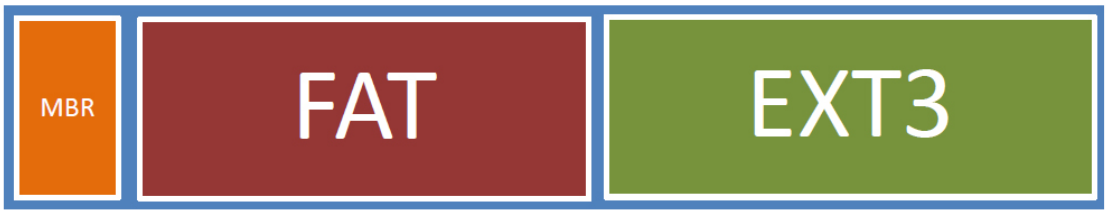
Planteo inicial de la solución

Dado que la aplicación consiste en la creación de discos, estos serán con archivos binarios, existirán más de un disco, así que en el sistema operativo GNU-Linux se creara una carpeta llamada sistema con los permisos necesarios para que el programa cree todos los archivos binarios que cada uno representara un Disco.

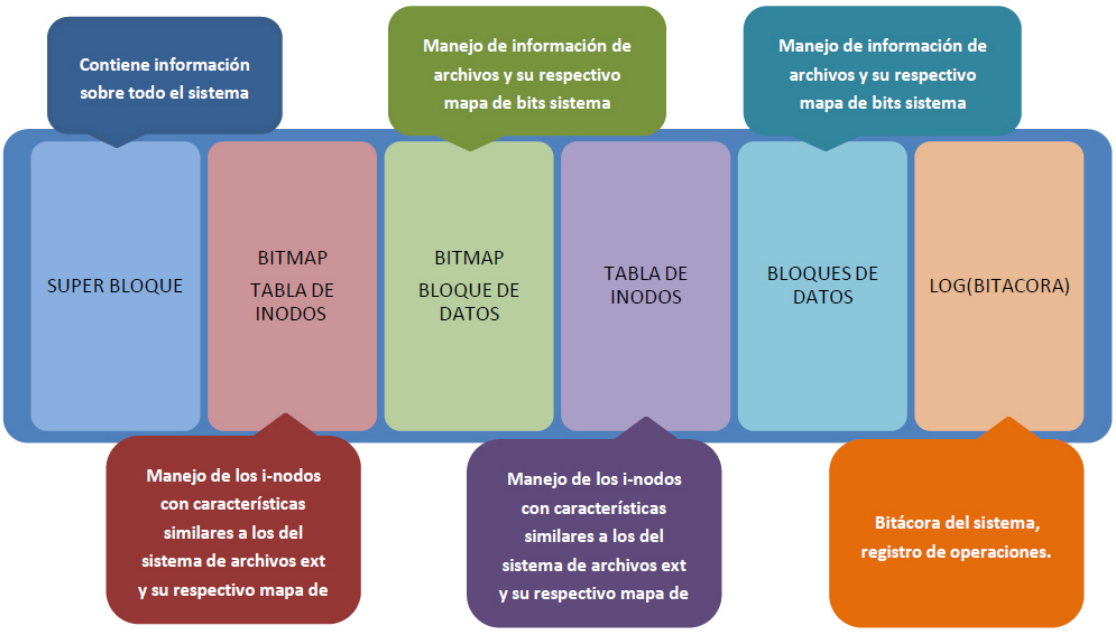
Así como también, para alojar los reportes generados por el usuario se creara una segunda carpeta llamada reporte, en esta se guardaran los archivos txt que el usuario genere al momento de utilizar la aplicación.

El disco duro y sus particiones se representan gráficamente de la siguiente manera:





En esta imagen se puede observar, como la primera estructura utilizada para crear el sistema de archivos es el MBR, el cual indica cual es la partición activa, así como también la que servirá de copia.

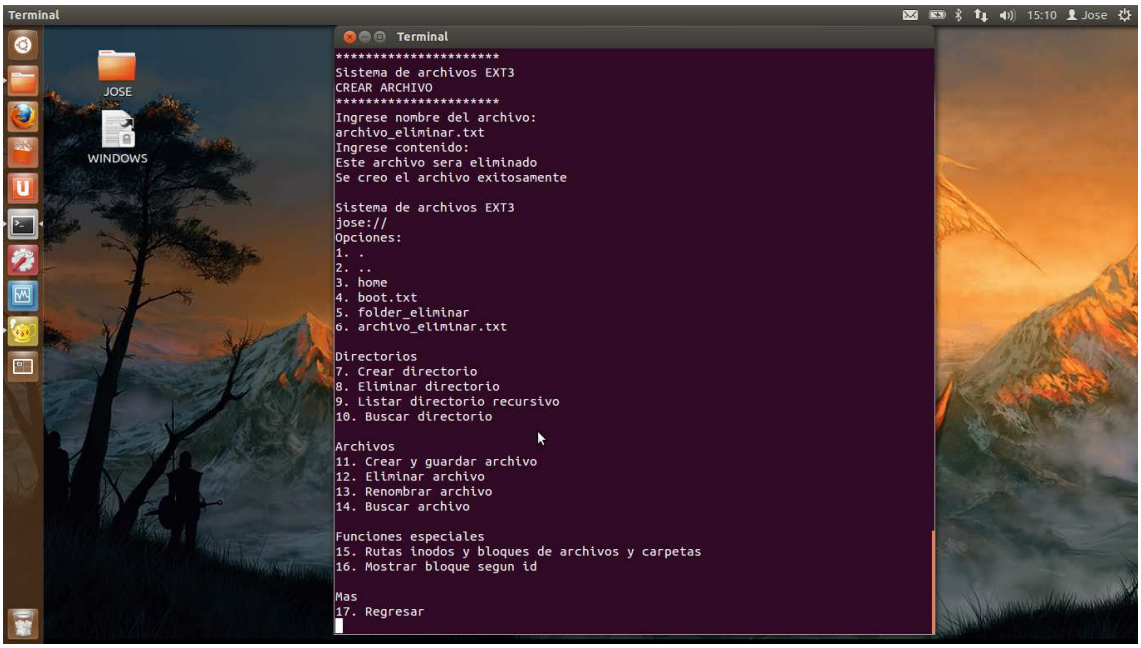


El sistema de archivos EXT3 está compuesto por las estructuras mostradas en la imagen anterior.

Así como también se puede observar el sistema FAT, algo que se puede observar es como la estructura de la bitácora se repite en ambos sistemas, a partir de esto se puede observar que se usara la misma estructura para ambos sistemas de archivos.

Diseño de la aplicación

La aplicación está diseñada para correr en la consola o terminal de gnu-linux, como se puede observar para cada menú se presentan las opciones listadas o enumeradas, el usuario podrá seleccionar escribiendo el numero que le corresponda a la selección deseada.



Estructuras utilizadas

Estructura que guarda la información del disco

```
struct struct_disco{
        char nombre[52];
        char tamano[9];
        char particiones[9];
        char puntero[9];
};
```

MBR que contiene la partición activa y no activa

```
struct struct_mbr{
        char mbr_byte_particion[10];
        char mbr_byte_particion_desactivada[10];
        char mbr_tipo_sistema_archivos[7];
        char mbr_tipo_particion[12];
        char mbr_particion_activa[4];
        char mbr_size_bytes[10];
};
```

Estructuras del EXT3

Estructura log, registra las acciones en el sistema de archivos.

```
struct struct_log{
        char log_activo[4];
        char log_tipo_operacion[53];
        char log_tipo[4];
        char log_nombre[643];
        char log_contenido[515];
        char log_fecha[19];
};
```

El superbloque es el archivo principal del sistema de archivos EXT3

```
struct struct_superbloque{
        char sb_numero_inodos[9];
        char sb_numero_bloques[9];
        char sb_numero_log[9];
        char sb_tamanio_bloque[4];
        char sb_numero_magico[7];
        char sb_bloques_freee[9];
        char sb_inodos_free[9];
        char sb_log_free[9];
        char sb_date_creacion[19];
        char sb_mount_time[19];
        char sb_write_time[19];
        char sb_mount_count[9];
        char sb_ap_bitacora[9];
        char sb_first_inodo[9];
        char sb_first_bloque[9];
        char sb_fist_free_bitmapinodo[9];
        char sb_first_free_bitmapbloque[9];
        char sb_first_free_log[9];
};
```

El inodo es el que almacena los punteros hacia los bloques de las carpetas y archivos

```
struct struct_inodo{
        char Llave[9];
        char i_tipo[6];
        char i_date_mod[19];
        char i_tam_archivo[9];
        char i_asig_bloques[9];
        char i_time_last_access[18];
        char i_ctime[18];
        char i_block1[9];
        char i_block2[9];
        char i_block3[9];
        char i_block4[9];
        char i_Iblock[9];
};
```

El bloque es el que almacena la información de cada directorio, y en archivos el contenido del mismo

```
//bloque
struct struct_bloque{
	char data[64];
	char estado[3];
};
```

Estructuras del boot

El boot es el equivalente a la estructura superbloque en el sistema FAT

```
struct struct_boot{
	char b_nombre_unidad[52];
	char b_usuario[52];
	char b_password[52];
	char b_tamano[9];
	char b_tablas_count[9];
	char b_bloques_count[9];
	char b_log_count[9];
	char b_next_tabla_free[9];
	char b_next_bloque_free[9];
	char b_next_log_free[9];
	char b_tablas_free[9];
	char b_bloques_free[9];
	char b_log_free[9];
	char b_date_creacion[18];
	char b_date_ultimo_montaje[18];
	char b_montajes_count[9];
	char b_ap_bloque_tabla_indexada[9];
	char b_ap_bloque_datos[9];
	char b_ap_log[9];
	char b_size_struct_tabla_indexada[9];
	char b_size_struct_bloque_dato[9];
	char b_size_struct_bitacora[9];
};
```

Esta estructura es el equivalente a los inodos, con la diferencia que las carpetas se listan en esta misma estructura sin necesidad de acceder a los bloques de datos.

```
struct struct_tabla{
	char disponible[3];
	char lista_nombre[52];
	char lista_tipo[6];
	char lista_nombre_padre[52];
	char lista_ap_directo_bloque[9];
	char lista_nivel[3];
	char lista_fecha_creacion[18];
	char lista_fecha_modificacion[18];
};
```

La estructura de bloques de datos en el sistema FAT almacena la información que contienen los archivos.

```
struct struct_datos{
        char disponible[3];
        char bd_data[51];
        char bd_numero[9];
        char bd_ap_directo_siguiente[9];
};
```

Variables generales utilizadas

Guarda la ruta de los discos duros del sistema
```
char sistema[8]="sistema/";
```

Guarda la ruta de los reportes del sistema
```
char sistema_reporte[17]="reporte/";
```

Dirección del disco actual
```
char directorio[58];
```

Dirección del reporte actual
```
char directorio_reporte[640];
```

Bit utilizado para guardar 0's
```
char buffer[1];
```

Contador de fors
```
int ifor=0;
```

Variable directorio global
```
DIR *d;
```

Para listar discos
```
struct dirent *di;
```

Fecha actual
```
char fecha[16];
```

FAT login (0= no ha iniciado sesión, 1= inicio sesión)
```
int login=0;
```

Explicación de funciones utilizadas

Funciones de inicialización

int getsize(char *informacion);
Esta función devuelve el entero con el tamaño de la variable información

void settime();
Establece la hora actual del sistema en la variable fecha

Void settime_reporte();
Establece la hora actual del sistema para la variable utilizada en los reportes

void set_disco(char nombre[50]);
Establece el directorio actual para utilizar con la función FILE

void set_disco_reporte(struct struct_disco tdisco, char reporte[50]);
Establece el directorio actual para almacenar el reporte

void inicializacion();
Esta función es llamada al momento de iniciar el programa y se asegura que la carpeta sistema exista, esta sirve para almacenar los discos creados por la aplicación.

void inicializacion_reporte();
Al momento de iniciar el menú reporte, esta función es llamada y se asegura que exista la carpeta para almacenar los reportes, si no existe es creada.

int probardenuevo(int salir);
Esta función es utilizada por FATlogin y sirve para iniciar sesión en el sistema FAT y sirve para preguntar al usuario si o no, devuelve la selección del usuario

void FATlogin(struct struct_disco tdisco, int activa);
Esta función es utilizada para establecer en la variable de inicio de sesión si el usuario ingresa correctamente su nombre de usuario y su contraseña, es decir es una función para el inicio de sesión.

char* cargarinformacion(char *d, char data[64])
Es utilizada para agregar a la variable char dinámica la información almacenada en data

Funciones para obtener y guardar bloques, inodos, tablas, datos y logs

int Bitacora_nextfree(struct struct_disco tdisco, int activa);
Si active es 1, devuelve el siguiente log libre de la partición activa, si es 0 devuelve el siguiente log libre de la partición no active.

struct struct_log Bitacora_obtener(int Llave, struct struct_disco tdisco, int activa);
Devuelve el log indicado con el id Llave, de la partición activa o no activa.

struct struct_inodo EXT3obtener_inodo(int Llave, struct struct_disco tdisco, int activa);
Función para el sistema de archives EXT3, devuelve el inodo con id Llave

struct struct_inodo EXT3obtener_inodo_set_access(int Llave, struct struct_disco tdisco, int activa);
Esta función es utilizada por el navegador EXT3, la cual cambia la fecha de ultimo acceso, dado que es el inodo que está siendo consultado por el usuario.

struct struct_tabla FATobtener_tabla(int Llave, struct struct_disco tdisco, int activa);
Esta función devuelve la tabla con id Llave del sistema de archives FAT

void EXT3guardar_inodo(int Llave, struct struct_disco tdisco, struct struct_inodo inodo, int activa);
Guarda el inodo enviado en el id Llave para el sistema EXT3

void FATguardar_tabla(int Llave, struct struct_disco tdisco, struct struct_tabla tabla, int activa);
Guarda la tabla enviada en el id Llave para el sistema FAT

void Bitacora_guardar(int Llave, struct struct_disco tdisco, struct struct_log log, int activa);
Guarda el log enviado en la partición active, a partir del MBR lo guardara para el sistema FAT o EXT3 dependiendo si la partición es active o no.

struct struct_bloque EXT3obtener_bloque(int Llave, struct struct_disco tdisco, int activa);
Devuelve el bloque con id Llave para el sistema EXT3

struct struct_datos FATobtener_datos(int Llave, struct struct_disco tdisco, int activa);
Para el sistema FAT devuelve el bloque de dato con id Llave

void EXT3guardar_bloque(int Llave, struct struct_disco tdisco, struct struct_bloque bloque, int activa);
Guarda el bloque enviado para el sistema EXT3 con id Llave

void FATguardar_datos(int Llave, struct struct_disco tdisco, struct struct_datos datos, int activa);
Guarda el bloque de dato para el sistema FAT con id Llave

Funciones del programa

void Bitacora_registro(char tipo_operacion[50], int file0_folder1, char direccion[640], char *informacion, struct struct_disco tdisco, int activa);
Recibe la información del que será almacenada en la bitácora de la partición activa o no activa.

void Bitacora_reporte(struct struct_disco tdisco);
Crea el reporte de bitácora en un archive de texto, a partir de la partición activa.

char* EXT3loaddata(char *informacion, struct struct_disco tdisco, struct struct_inodo inodo, int activa);
Devuelve un char de tamaño dinámico que almacena el inodo enviado, devolviendo la concatenación de la informacion si esta está en varios bloques de datos e inodos.

void EXT3_bloque_mostrar_informacion(struct struct_bloque bloque, int id);

Esta función muestra la información del bloque enviado, así como también el id del mismo, es decir muestra en pantalla el valor de cada variable que compone esta estructura.

void EXT3_inodo_mostrar_informacion(struct struct_inodo inodo);

Muestra la información del inodo enviado, es decir muestra en pantalla el valor de cada variable que compone esta estructura.

char* EXT3loaddata_ruta_bloques(char *informacion, struct struct_disco tdisco, struct struct_inodo inodo, int activa);

Esta función es similar a EXT3loaddata(…) con la excepción que muestra la información en pantalla de cada bloque e inodo consultado.

char* FATloaddata(int apuntador, struct struct_disco tdisco, int activa);

Esta función es similar a EXT3loaddata(…) con la excepción que es para el sistema de archives FAT

struct struct_inodo_ubicacion EXT3get_inodo_from_seleccion(int menu, struct struct_inodo inodo, struct struct_disco tdisco, int activa);

En el menú de navegación del sistema de archivos EXT3, utiliza esta función para enviar la selección del usuario y así obtener el inodo y la dirección de su selección.

struct struct_tabla FATget_tabla_from_seleccion(int menu, int nivel, char padre[50], struct struct_disco tdisco, int activo);

El equivalente de la función anterior para el sistema FAT.

void EXT3_delete_all(char direccion[640], int illave, struct struct_disco tdisco, int activa, int mostrar);

Elimina físicamente el archivo y carpeta, si es carpeta se llama recursivamente para eliminar de primero el archivo o folder de más profundidad, para finalizar se elimina a sí mismo.

void FAT_delete_all_data(int llave, struct struct_disco tdisco, int activa);

Esta función elimina todas los bloques de informacion que componen un archivo, si este aloja más de un bloque de información, se llama recursivamente para eliminar el ultimo para finalizar consigo mismo.

void FAT_delete_all_tabla(char direccion[640], int nivel, int apuntador, char padre[50], struct struct_disco tdisco, int activa, int mostrar);

Elimina la información del sistema FAT, si es archivo elimina todos los bloques de datos utilizando la función anterior, si es folder se llama recursivamente para eliminar cualquier carpeta o folder contenido empezando por el de más profundidad, finaliza con su misma eliminación.

int EXT3guardar(struct struct_disco tdisco, char *informacion, char tipo[4], int activa);

Esta función reciba la información a guardar en el char dinámico, y lo almacena como carpeta o archivo en el sistema de archivos EXT3

```
void FATguardar(char *tinformacion, struct struct_disco tdisco, int activa);
```
El equivalente a la function anterior para el sistema de archives FAT.

```
void EXT3editar(char *informacion, struct struct_disco tdisco, int illave, int activa);
```
Para el sistema EXT3 actualiza la informacion del id illave con la información enviada, es decir es la edición del contenido del archivo.

```
void FATeditar(char *informacion, struct struct_disco tdisco, int nivel, char padre[50], int activa, int apuntador);
```
El equivalente de la función anterior para el sistema de archivos FAT.

```
char* EXT3eliminar(char tipo[4], char *original, char nombre[50], struct struct_disco tdisco, int activa, char tubicacion[640]);
```
Cuando se eliminan los bloques de datos de un archivo es necesario actualizar el inodo en el cual se almacenaba su enlace, esta función actualiza esta información eliminado su nombre del directorio.

```
char* EXT3renombrar(char *original, char nombre[50], char renombre[50], struct struct_disco tdisco, int activa, char tubicacion[640]);
```
Renombra un nombre de archivo del sistema de archivos EXT3.

```
int EXT3repetido_carpeta_folder(char ubicacion[640], int llave, struct struct_disco tdisco, char busqueda[50]);
```
Pare el sistema EXT3 devuelve un 1 si el nombre enviado ya existe en la carpeta indicada.

```
int FATrepetido_carpeta_folder(char ubicacion[640], int nivel, int apuntador, char padre[50], struct struct_disco tdisco, char busqueda[50]);
```
El equivalente para el sistema FAT de la función anterior.

```
void EXT3actualiza(char *informacion, struct struct_disco tdisco, char tipo[4], int illave, int activa);
```
Actualiza la información del inodo con la información enviada, se utiliza cuando se crea una nueva carpeta o archivo.

```
void EXT3_crear_carpeta(char nombre[50], struct struct_disco tdisco, int illave, int activa);
```
Esta función crea físicamente la carpeta indicada en el inodo con id illave.

```
void EXT3_eliminar_carpeta(char nombre[50], struct struct_disco tdisco, int illave, int activa, char tubicacion[640]);
```
Elimina físicamente la carpeta indicada en el inodo illave.

```
void EXT3_eliminar_archivo(char nombre[50], struct struct_disco tdisco, int illave, int activa, char tubicacion[640]);
```
Elimina el archivo indicado con inodo illave

```
void EXT3_renombrar_archivo(char nombre[50], char renombre[50], struct struct_disco tdisco, int illave, int activa, char tubicacion[640]);
```
Cambia de nombre al archivo indicado en el sistema EXT3

```
void FAT_crear_carpeta(char nombre[50], struct struct_disco tdisco, int nivel, char padre[4], int activa);
```
Crea la carpeta en el nivel y bajo el padre indicado para el sistema de archivos FAT

```
void FAT_eliminar_carpeta(char nombre[50], struct struct_disco tdisco, int nivel, char padre[4], int activa, char tubicacion[640]);
```
Elimina la carpeta indicada utilizando las funciones anteriores para asegurar la eliminación de todo lo que este contenido en el mismo.

```
void FAT_eliminar_archivo(char nombre[50], struct struct_disco tdisco, int nivel, char padre[4], int activa, char tubicacion[640]);
```
Como la función anterior, elimina el archivo indicado para el sistema FAT

```
void FAT_renombrar_archivo(char nombre[50], char renombre[50], struct struct_disco tdisco, int nivel, char padre[4], int activa, char tubicacion[640]);
```
Cambia de nombre al archivo indicado en el sistema FAT.

```
void EXT3_crear_archivo(char nombre[50], char *informacion, struct struct_disco tdisco, int illave, int activa);
```
Crea un nuevo archivo para el sistema EXT3

```
void FAT_crear_archivo(char nombre[50], char *informacion, struct struct_disco tdisco, int nivel, char padre[50], int activa);
```
Crea un nuevo archivo para el sistema FAT

```
void EXT3crear_data(char tipo[4], int illave, struct struct_disco tdisco, int nivel, char padre[4], char ubicacion[640]);
```
Esta función es llamada para crear un nuevo archivo o carpeta en el sistema EXT3, de acuerdo a la selección del usuario elimina un archivo o carpeta.

```
void EXT3eliminar_data(char tipo[4], int illave, struct struct_disco tdisco, int nivel, char padre[4], char ubicacion[640]);
```
Elimina un archivo o carpeta en el sistema EXT3, de acuerdo a los datos ingresados se llama a la función de eliminar carpeta o archivo.

```
void EXT3renombrar_data(int illave, struct struct_disco tdisco, int nivel, char padre[4], char ubicacion[640]);
```
Cambia de nombre al archivo indicado por el usuario, envía los datos ingresados a la función que elimina físicamente el archivo.

```
void EXT3editar_data(int illave, struct struct_disco tdisco, int nivel, char padre[4], int apuntador, char ubicacion[640]);
```
Edita la información del archivo, una vez el usuario ingresa los datos necesarios se llama a la función para editar físicamente el archivo.

```
void FATcrear_data(char tipo[4], int nivel, char padre[50], struct struct_disco tdisco, int inodo, char ubicacion[640]);
```
Función para el sistema FAT crea un archivo o carpeta, una vez obtiene los parámetros llama a las funciones para crear una carpeta o archivo.

void FATeliminar_data(char tipo[4], int nivel, char padre[50], struct struct_disco tdisco, int inodo, char ubicacion[640]);

Obtiene el nombre de la carpeta o archivo a eliminar, una vez esto llama a la respectiva función para eliminarlo físicamente de la partición.

void FATeditar_data(int nivel, char padre[50], struct struct_disco tdisco, int inodo, int apuntador, char ubicacion[640]);

Edita la información del archivo especificado.

void FATrenombrar_data(int nivel, char padre[50], struct struct_disco tdisco, int inodo, char ubicacion[640]);

Cambia el nombre del archivo del sistema de archivos FAT.

void EXT3listar_recursivo(char ubicacion[640], int llave, struct struct_disco tdisco);

Lista los directorios y archivos de manera recursiva que estén contenido en el mismo.

void FATlistar_recursivo(char ubicacion[640], int nivel, int apuntador, char padre[50], struct struct_disco tdisco);

Al igual que la función anterior, lista recursivamente los archivos y carpetas de manera recursivamente.

int EXT3buscar_carpeta_folder(char ubicacion[640], int llave, struct struct_disco tdisco, char busqueda[50], int buscar_carpeta, int mostrar);

Busca el archivo indicado, utiliza recursividad para encontrar todos los archivos que coincidan mostrando sus rutas.

void EXT3ruta_inodos_bloques(char ubicacion[640], int llave, struct struct_disco tdisco, char busqueda[640]);

Esta funcione es únicamente del sistema EXT3, muestra la información de cada inodo y bloque consultado para llegar a la ruta indicada.

void FATbuscar_carpeta_folder(char ubicacion[640], int nivel, int apuntador, char padre[50], struct struct_disco tdisco, char busqueda[50], int buscar_carpeta);

Para el sistema de archivos FAT, busca la carpeta o bien folder indicado, mostrando las rutas de los distintos archivos si existen varios.

void EXT3navegador(char ubicacion[640], int llave, struct struct_disco tdisco, int nivel, int apuntador, char padre[4]);

Esta función es una de las más importantes, ya que navega recursivamente a travez del sistema de archivos EXT3, por cada selección realiza la misma operación a travez del RAID en el sistema FAT. Si la posición actual es un archivo muestra el contenido del mismo.

void FATnavegador(char ubicacion[640], int nivel, int apuntador, char padre[50], struct struct_disco tdisco, int inodo);

Este es el equivalente de la función anterior para el sistema FAT.

void montar_particion(struct struct_disco tdisco);

Es llamada para entrar al sistema de archivos, de acuerdo a que particion este activa entrara al navegador FAT o EXT3

```
void EXT3_reporte_inodos(struct struct_disco tdisco);
```
Crea el reporte de inodos del sistema de archives EXT3

```
void EXT3_reporte_bloques(struct struct_disco tdisco);
```
Crea el reporte para el sistema de archivos EXT3 de bloques

```
void FAT_reporte_datos(struct struct_disco tdisco);
```
Crea el reporte de bloques de datos del sistema de archivos FAT.

```
void reportes(struct struct_disco tdisco);
```
Inicia el menú de reportes para ambos sistemas de archives.

```
int ext(int size, int posicion, struct struct_disco tdisco);
```
Crea el formato EXT3 para la partición del disco a partir de la posición indicada y del tamaño indicado.

```
int fat(int size, int posicion, struct struct_disco tdisco);
```
Crea el formato FAT para la partición del disco que inicia y sera del tamaño indicado.

```
void arranque(int posicion, int desactivada, char mbr_tipo_sistema_archivos[4],char tipo[9],int activa,int size, struct struct_disco tdisco);
```
Modifica el MBR del disco para establecer la partición activa y no activa indicados.

```
void editar_arranque(struct struct_disco tdisco);
```
Edita el arranque del disco, con esto se puede activar una partición para usar la que esta no active.

```
void crear_particion_opciones(struct struct_disco tdisco);
```
Muestra las opciones de formato para crear las particiones según el orden deseado.

```
void formatear_disco(struct struct_disco tdisco);
```
Da formato al disco, es decir elimina toda la informacion que este contiene.

```
void formatear_particion(struct struct_disco tdisco);
```
Formatea las particiones del disco seleccionado.

```
void mostrar_mbr(struct struct_disco tdisco);
```
Muestra la informacion que almacena el MBR del disco seleccionado.

```
void mostrar_datos_sistema(struct struct_disco tdisco);
```
Esta función muestra los datos del sistema, es decir si la partición activa es EXT3 muestra la información contenida en el superbloque, si no es así, muestra el boot del sistema de archivos FAT.

```
void entrar_al_disco(char unidad[640]);
```
Recibe el nombre del disco seleccionado, mostrando las opciones de este.

```
void listar_discos();
```
Lista los discos disponibles en el sistema.

```
void crear_disco();
```
Crea un disco físico en el sistema.

```
void menu_principal();
```
Este muestra el menú principal de la aplicación, el usuario elige crear un disco o navegar entre los mismos.

```
int main(int argc, char **argv);
```
Función inicial del programa, crea la carpeta de inicialización si no existe y llama al menú principal.