Manual Técnico

Fase 1

Asunción Mariana Sic Sor **201504051**

09 Junio, 2020

Índice

Descripción del Sistema

Objeto

- Que el usuario pueda hacer uso de este programa
- Describir a detalle el programa
- Que el usuario pueda crear particiones simuladas mediante archivos binarios en c/c++

Alcance

- Este programa está dirigido a usuario que deseen explorar más en el ámbito de sistema de archivos.

Funcionalidad

El programa funciona mediante comandos a través de consola los cuales no hace distinción entre mayúsculas y minúsculas, los cuales son los siguientes:

1. MKDISK: Funciona para crear un disco duro[1] en blanco, recibe como parámetros (los cuales se detallan más adelante):
    a. **SIZE[2]**
    b. FIT
    c. UNIT
    d. **PATH**
2. RMDISK: Este comando elimina un disco duro existente
    a. **PATH**
3. FDISK: Con este comando se puede hacer un manejo de las distintas particiones.
    a. **SIZE**
    b. UNIT
    c. **PATH**
    d. TYPE
    e. FIT
    f. DELETE
    g. **NAME**

[1] Cada vez que se menciona la creación/eliminación/edición de discos duros, se refiere a los simulados mediante archivos binarios.
[2] Parametros marcados en negrilla, indica de caracter obligatorio, es decir, siempre se le debe de dar su respectivo valor.

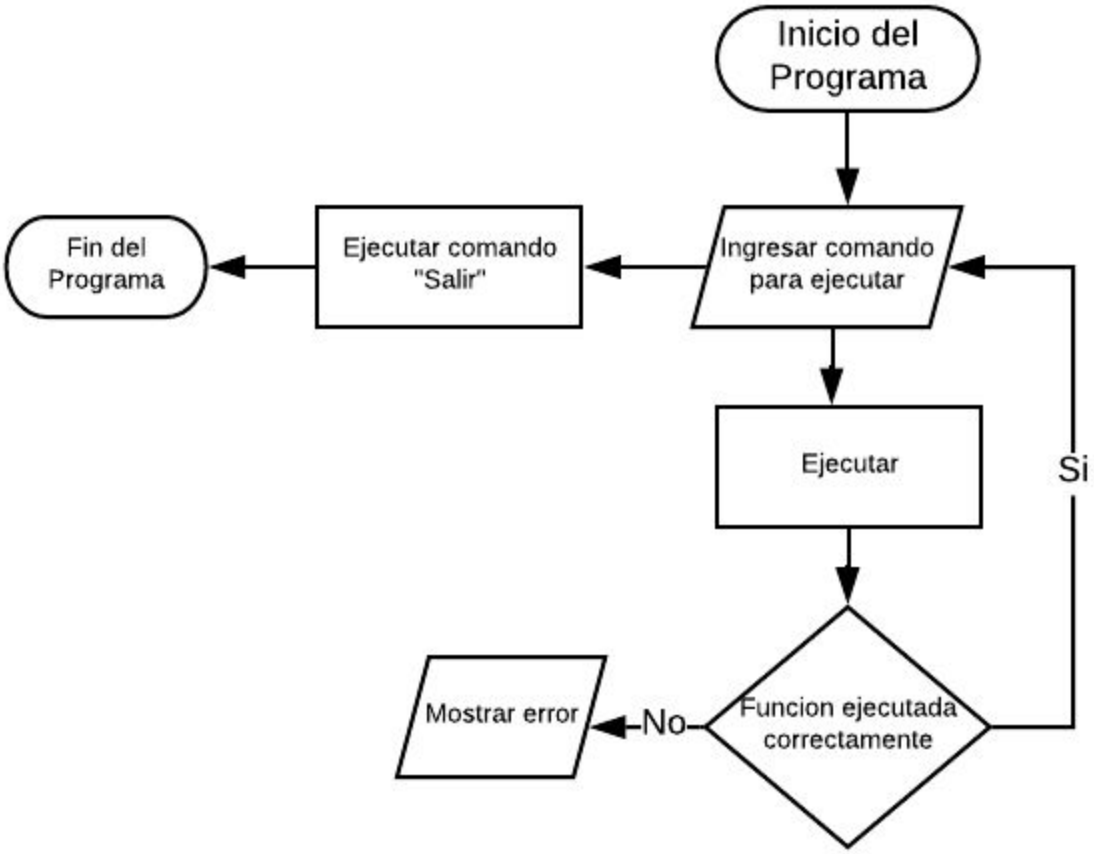
    h. ADD
4. MOUNT: Con este comando se pueden montar particiones en memoria RAM
    a. PATH
    b. NAME
5. UNMOUNT: Desmonta la partición deseada.
    a. ID
6. REP: Con este comando se pueden generar reportes en imágenes donde se pueda visualizar los discos duros.
    a. NAME
    b. PATH
    c. ID
7. EXEC: Mediante este comando se puede ejecutar un script de extensión .sh que contenga comandos que se deseen ejecutar
    a. PATH
8. SALIR: Con este comando se podrá salir del sistema, no recibe ningún parámetro.

Los parámetros admitidos para los distintos comandos son los siguientes:

1. SIZE: Recibe únicamente números, indica tamaño y debe ser mayor a cero.
2. FIT: Sirve para indicar el ajuste de disco duro o particiones, su valor puede ser FF (primer ajuste), WF (peor ajuste), BF (mejor ajuste).
3. UNIT: Recibe la unidad para manejar el tamaño de particiones o disco duro. Su valor puede ser K (KiloBytes), M (MegaBytes) o B (Bytes).
4. PATH: Indica ruta absoluta.
5. TYPE: Indica el tipo de partición que se pueda crear; P (primaria), E (extendida) o L (lógica). Cabe mencionar que únicamente se pueden crear 4 particiones en total incluyendo primarias y, en dado caso haya, solo una extendida. Las particiones lógicas se pueden crear media vez exista una extendida.
6. DELETE: Este comando elimina la partición indicada, su valor puede ser FAST (el cual solo indica que la partición no existe) y FULL (el cual, aparte de indicar que no existe, rellena el espacio con el carácter \0)
7. ADD: Este comando se usa únicamente para indicar la cantidad de espacio que se desea agregar o quitar a una partición en específico. Su valor debe ser numérico y puede ser mayor, menor o igual a cero.
8. ID: Con este comando indicamos el id de alguna partición existente, recibe una cadena de caracteres como valor.

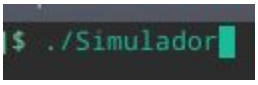
Navegación

1. Programa en general

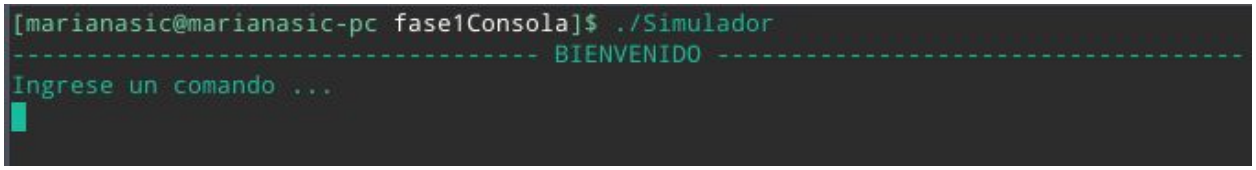


Mapa del Sistema

Para acceder al sistema, se debe de correr en consola el ejecutable, llamado “Consola”, con el siguiente comando:



Donde al momento de ejecutar, aparecerá la pantalla de “Bienvenido”



Y desde ese momento, se pueden escribir los comandos anteriormente especificados.

Para todo el programa se hizo uso de los siguientes structs:

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

typedef struct
{
    string nombre = "";
    char idPart[200];
    int size = 0;
    char fit;
    char unit;
    string path = "";
    char type;
    char eliminar;
    string name = "";
    int add = 0;
}Funcion;
```

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

typedef struct
{
    int mbr_tamano;
    char mbr_fecha_creacion[32];
    int mbr_disk_signature;
    char letra;
    char disk_fit;
    Partition mbr_partition[4];
}MBR;
```

Los cuales sirven de principal ayuda para el correcto manejo de todos los comandos.

Descripción del Sistema

A continuación se mostrará la ejecución de cada comando admitido

1. MKDISK

Este comando creará un disco duro llamado “Disco1” en el path especificado con tamaño de 21Kb

Para crear los discos, se utilizo FILE* para poder crear los respectivos archivos con extension .disk

```
FILE* archivo = fopen(pth, "wb");
FILE* archivo2 = fopen(pth2, "wb");
if(archivo == NULL || archivo2 == NULL){
    cout << "Error al acceder al archivo " << endl;
    ErrorCrearDisco++;
}else{
    int fin=(tamano/1024);
    char buffer[1024];
    int i=0;
    for(i=0;i<1024;i++){
        buffer[i]='\0';
    }
    int j=0;
    while(j!=fin){
        fwrite(&buffer,1024 , 1, archivo);
        fwrite(&buffer, 1024, 1, archivo2);
        j++;
    }
    fclose(archivo);
    fclose(archivo2);
}

mbr.mbr_tamano = tamano;

//fecha y hora de creacion
strcpy(mbr.mbr_fecha_creacion, getDate().c_str());
consola("Fecha y Hora de creacion de disco ... " + getDate());

//disk signature
mbr.mbr_disk_signature = atoi(getSign().c_str());
mbr.disk_fit = func.fit;

for(int i = 0; i < 4; i++){
    mbr.mbr_partition[i].part_status='I';
    mbr.mbr_partition[i].part_start=-1;
    mbr.mbr_partition[i].part_size=0;
    mbr.mbr_partition[i].part_fit = 'N';
    mbr.mbr_partition[i].part_name[0] = 'N';
    mbr.mbr_partition[i].part_type = 'N';
}
```

Y luego se procedia a escribir con la ayuda de la funcion fseek y fwrite la estructura MBR en tal disco.

```
FILE* file= fopen(pth, "r+b");
FILE* file2= fopen(pth2, "r+b");
fseek(file,0,SEEK_SET);// estableciendo puntero al inicio
fseek(file2,0,SEEK_SET);// estableciendo puntero al inicio

if (file==NULL || file2 ==NULL)//si el archivo no contiene nada
{
    cout << "Error al acceder al archivo " << endl;
    ErrorCrearDisco++;
}
else
{
    fwrite(&mbr, sizeof(MBR), 1, file);//escribiendo la estructura del MBR
    fclose(file);
    fwrite(&mbr, sizeof(MBR), 1, file2);
    fclose(file2);
    cout << "Discos en " << func.path << " creado con exito ..." << endl;
}
```

2. RMDISK

```
Ingrese un comando ...
rmdisk -path=/home/marianasic/Discos/final_ra1.disk
Reconocio path ... /home/marianasic/Discos/final_ra1.disk
Ejecutando comando rmdisk ...
./final_ra1.disk
Seguro que desea eliminar? [Y/n]
```

Al momento de querer eliminar un disco, se muestra un mensaje de confirmación. Si se escribe “n” o “N”, el disco no se elimina, cualquier otro caso, elimina el disco duro indicado.

Antes de eliminar el disco, se procedia a verificar que existiera dicho archivo.

Y despues de la confirmacion del usuario, se procedia a eliminar por medio de system, para ejecutar codigo en consola, con el comando de rm (nativo de linux)

```
string confirm;
cout << "Seguro que desea eliminar? [Y/n] " << endl;
getline(cin, confirm);
//cout << "Eligio " << confirm << endl;
if(confirm == "n" || confirm == "N"){
    cout << "Disco " << name << " no eliminado " << endl;
}else{
    char e[1024];
    strcpy(e,apath);
    strcat(e, "&& rm \"");
    strcat(e, name.c_str());
    strcat(e, "\"");
    cout << "Eliminando " << name << " ... " << endl;
    if(!system(e)){
        cout << "Disco " << name << " eliminado ... " << endl;
    }
}
```

```
Ingrese un comando ...
rmdisk -path=/home/marianasic/Discos/final_ra1.disk
Reconocio path ... /home/marianasic/Discos/final_ra1.disk
Ejecutando comando rmdisk ...
./final_ra1.disk
Seguro que desea eliminar? [Y/n]
N
Disco final_ra1.disk no eliminado
---------------------------------- BIENVENIDO ----------------------------------
Ingrese un comando ...
rmdisk -path=/home/marianasic/Discos/final_ra1.disk
Reconocio path ... /home/marianasic/Discos/final_ra1.disk
Ejecutando comando rmdisk ...
./final_ra1.disk
Seguro que desea eliminar? [Y/n]
y
Eliminando final_ra1.disk ...
Disco final_ra1.disk eliminado ...
---------------------------------- BIENVENIDO ----------------------------------
Ingrese un comando ...
```

3. FDISK

```
Ingrese un comando ...
fdisk -path=/home/marianasic/Discos/Disco1.disk -type=e -name=extPrim -size=8 -unit=
k -fit=wf
```

Con ese comando, se crea una partición extendida de tamaño 8Kb y con el peor ajuste en el disco “Disco1.disk”. Cada vez que se utilice este comando, mostrará un pequeño detalle de las particiones primarias y extendidas en el disco.

Resultado del comando anterior:

```
Creando primer particion ...
Colocando particion al final ...
Donde comienza a escribir es 13312
Particion extendida agregada ...
En  13312
No se ha agregado ni quitado espacio a ninguna particion
No se ha eliminado ninguna particion
```

Antes de hacer la accion solicitada por el comando, se procedio a hacer algunas validaciones.

Como la de verificar que no excedieran los tipos de particiones.

```
char tipoPart;
if(func.type == 'P'){
    if(numParticiones<=4){
        partPrimaria++;
        tipoPart = 'P';
    }else{
        cout << "Error: Las particiones primarias exceden " << endl;
        ErrorFdisk++;
    }
}else if(func.type == 'E'){
    if(partExtendida == 0 ){
        if(numParticiones == 4){
            cout << "Error: Ya no hay espacio para crear una particion extendida" << endl;
            ErrorFdisk++;
        }else{
            partExtendida++;
            hayExtendida++;
            tipoPart = 'E';
        }
    }else{
        cout << "Error: Ya no hay espacio para crear una particion extendida" << endl;
        ErrorFdisk++;
    }
}else if(func.type == 'L'){
    if(partExtendida == 1){
        tipoPart = 'L';
    }else{
        cout << "Error: No se puede crear una particion logica sin una extendida" << endl;
        ErrorFdisk++;
    }
}
```

O colocar el tamano ya en Bytes de las particiones

```
//tamano de la particion
int tamano = 0;
if(func.unit == 'B'){
    tamano = func.size;
}else if(func.unit == 'K'){
    tamano = func.size * 1024 ;
}else if(func.unit == 'M'){
    tamano = func.size * (1024 * 1024);
}
```

Seguido de eso, se procedia a hacer las funciones de:

- Crear

```
//si no hay ninguna particion
if(numParticiones == 0){
    cout << "Creando primer particion ... " << endl;
    if(func.fit == 'B' || func.fit == 'F'){
        if(hayExtendida == 1){
            nuevaEbr.part_status = 'I';
            nuevaEbr.part_start = -1;
            nuevaEbr.part_next = -1;
        }
        mbr.mbr_partition[0].part_type = tipoPart;
        mbr.mbr_partition[0].part_fit = func.fit;
        strcpy(mbr.mbr_partition[0].part_name, func.name.c_str());
        mbr.mbr_partition[0].part_size = tamano;
        mbr.mbr_partition[0].part_start = sizeof(MBR);
        mbr.mbr_partition[0].part_status = 'A';
        //igualando la particion a escribir en el disco
        nprt = mbr.mbr_partition[0];

        if(hayExtendida==1){
            fseek(disco,nprt.part_start,SEEK_SET);
            fwrite(&nuevaEbr, sizeof(EBR), 1, disco);
            cout << "Particion extendida agregada ... " << endl;
            cout << "En  " << nprt.part_start << endl;
        }else{
            fseek(disco,nprt.part_start,SEEK_SET);
            fwrite(&nprt, sizeof(Partition), 1, disco);
            cout << "Particion primaria agregada ... " << endl;
            cout << "En  " << nprt.part_start << endl;
        }
    }else if(func.fit == 'W'){
        cout << "Colocando particion al final ..." << endl;
        if(hayExtendida == 1){
            nuevaEbr.part_status = 'I';
            nuevaEbr.part_start = -1;
            nuevaEbr.part_next = -1;
        }
```

```
//si hay mas de una particion pero menos de 4
else if(numParticiones > 0 && numParticiones < 4){
    cout << "Agregando nuevas particiones ... " << endl;
    if(func.fit == 'F' || func.fit == 'B'){
        int colocar;
        for(int i = 3; i >= 0; i--){
            if(mbr.mbr_partition[i].part_status == 'I'){
                colocar = i;
            }
        }
        cout << "La particion se colocara en la posicion " << colocar << endl;
        int inicioPart;
        if(colocar == 0){
            inicioPart = sizeof(MBR);
            cout << "Si colocar es 0, inicia en " << inicioPart << endl;
        }else{
            inicioPart = (mbr.mbr_partition[colocar-1].part_start + mbr.mbr_partition[colocar-1].part_size);
            cout << "Si colocar es otro valor, inicia en " << inicioPart << endl;
        }

        cout << "La particion va a comenzar en " << inicioPart << endl;
        mbr.mbr_partition[colocar].part_type = tipoPart;
        mbr.mbr_partition[colocar].part_fit = func.fit;
        strcpy(mbr.mbr_partition[colocar].part_name, func.name.c_str());
        mbr.mbr_partition[colocar].part_size = tamano;
        mbr.mbr_partition[colocar].part_start = inicioPart;
        mbr.mbr_partition[colocar].part_status = 'A';
        //igualando la particion a escribir en el disco
        nprt = mbr.mbr_partition[colocar];

        if(hayExtendida==1){
            nuevaEbr.part_status = 'I';
            nuevaEbr.part_start = -1;
            nuevaEbr.part_next = -1;
            fseek(disco,nprt.part_start,SEEK_SET);
            fwrite(&nuevaEbr, sizeof(EBR), 1, disco);
            cout << "Particion extendida agregada ... " << endl;
            cout << "En  " << nprt.part_start << endl;
```

- Editar

```
////////////////////////////////////////////// ADD
//agregar espacio
if(func.add > 0){

    int agregar = 0;
    if(func.unit == 'B'){
        agregar = func.add;
    }else if(func.unit == 'K'){
        agregar = func.add * 1024 ;
    }else if(func.unit == 'M'){
        agregar = func.add * (1024 * 1024);
    }

    cout << "Agregrando " << agregar << " espacio en la particion  " << particionAgregar+1 << endl;
    if(nombresIguales == 1){
        if(particionAgregar == 3){
            if((agregar + mbr.mbr_partition[particionAgregar].part_size + mbr.mbr_partition[particionAgregar].part_start) >= mbr.mbr_tamano){
                cout << "Error: No se puede agregar porque ya no hay espacio disponible " << endl;
            }else{
                mbr.mbr_partition[particionAgregar].part_size += agregar;
            }
        }else{
            if((agregar + mbr.mbr_partition[particionAgregar].part_size + mbr.mbr_partition[particionAgregar].part_start) >= mbr.mbr_partition[particion
                cout << "Error: No se puede agregar porque ya no hay espacio disponible " << endl;
            }else{
                mbr.mbr_partition[particionAgregar].part_size += agregar;
            }
        }
    }else{
        cout << "Particion no encontrada ... " << endl;
    }
}else if(func.add < 0){
    cout << "Quitando espacio ... " << endl;
    int agregar = 0;
    if(func.unit == 'B'){
        agregar = func.add * -1;
    }else if(func.unit == 'K'){
        agregar = func.add * -1024 ;
    }else if(func.unit == 'M'){
        agregar = func.add * (1024 * -1024);
    }

    cout << "Quitando " << agregar << " espacio ... " << endl;
```

- Eliminar

```
////////////////////////// DELETE
if(func.eliminar == 'R'){ //opcion FAST de delete
    if(nombresIguales == 1){
        string confirm;
        cout << "Seguro que desea eliminar? [Y/n] " << endl;
        getline(cin, confirm);
        //cout << "Eligio " << confirm << endl;
        if(confirm == "n" || confirm == "N"){
            cout << "Particion no eliminada " << endl;
        }else{
            mbr.mbr_partition[particionAgregar].part_status='I';
            mbr.mbr_partition[particionAgregar].part_start=-1;
            mbr.mbr_partition[particionAgregar].part_size=0;
            mbr.mbr_partition[particionAgregar].part_fit = 'N';
            mbr.mbr_partition[particionAgregar].part_name[0] = 'N';
            mbr.mbr_partition[particionAgregar].part_type = 'N';
            cout << "La particion " << particionAgregar+1 << " fue eliminada ..." << endl;
        }
    }
    else{
        cout << "Particion no encontrada ... " << endl;
    }
}else if(func.eliminar == 'T'){//opcion FULL de delete
    if(nombresIguales == 1){
        string confirm;
        cout << "Seguro que desea eliminar? [Y/n] " << endl;
        getline(cin, confirm);
        //cout << "Eligio " << confirm << endl;
        if(confirm == "n" || confirm == "N"){
            cout << "Particion no eliminada " << endl;
        }else{
            cout << "Rellenando con 0s ... " << endl;
            fseek(disco, mbr.mbr_partition[particionAgregar].part_start, SEEK_SET);
            //rellenando con 0s
            int fin = mbr.mbr_partition[particionAgregar].part_size + mbr.mbr_partition[particionAgregar].part_start;
            char buffer[1024];
            int i=0;
            for(i=0;i<1024;i++){
                buffer[i]='\0';
            }
            int j=0;
            while(j!=fin){
                fwrite(&buffer,1024 , 1, disco);
                j++;
            }
```

4. REP

```
Ingrese un comando ...
rep -path=/home/marianasic/Discos/Disco1.disk -name=mbR -id=repFinal
Reconocio path ... /home/marianasic/Discos/Disco1.disk
Reconocio name ... mbR
Reconocio id ... repFinal
Generando el dot ...
Imagen generada ...
```

Ya que se ha generado la imagen, el reporte de MBR es el siguiente:

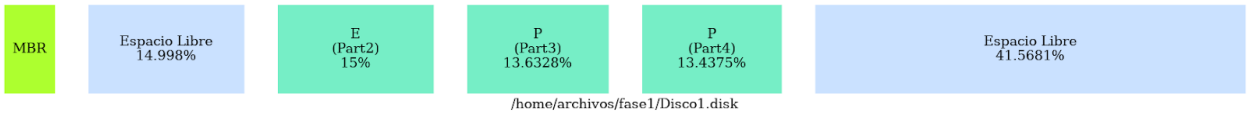
MBR	
mbr_tamano	52428800
mbr_fecha_creacion	F 09/06/2020 H 22:54:30
mbr_disk_signature	1756599702
disk_fit	B
part_status_1	A
part_type_1	P
part_fit_1	B
part_start_1	156
part_size_1	7864320
part_name_1	Part1
part_status_2	A
part_type_2	E
part_fit_2	F
part_start_2	7863452
part_size_2	7864320
part_name_2	Part2
part_status_3	A
part_type_3	P
part_fit_3	W
part_start_3	15726748
part_size_3	7864320
part_name_3	Part3
part_status_4	A
part_type_4	P
part_fit_4	B
part_start_4	23590044
part_size_4	7864320
part_name_4	Part4

/home/archivos/fase1/Disco1.disk

Y el reporte de DISK

```
Ingrese un comando ...
rep -path=/home/marianasic/Discos/Disco1.disk -name=diSk -id=repDisco
Reconocio path ... /home/marianasic/Discos/Disco1.disk
Reconocio name ... diSk
Reconocio id ... repDisco
Generando el dot ...
Creando reporte de discos ...
Imagen generada ...
```

El cual es:



/home/archivos/fase1/Disco1.disk

5. EXEC

Por último, al usar ese comando:

```
Ingrese un comando ...
exec -path=/home/marianasic/Documents/archivos/lab/proyecto1/fase1/fase1Consola/ejec
utar.sh
```

El cual, en el archivo “ejecutar.sh” tiene lo siguiente:

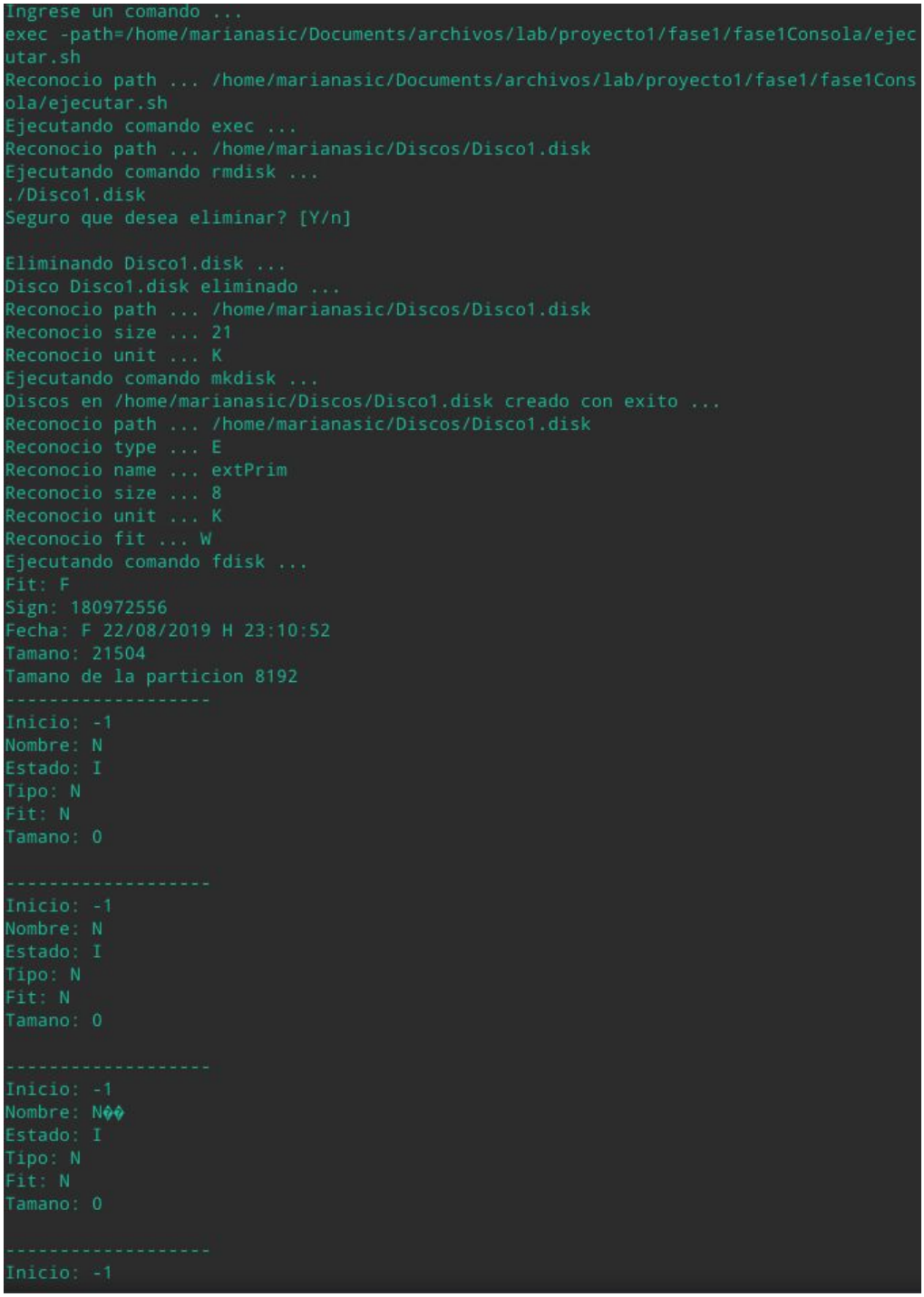
ejecutar.sh

```
rmdisk -path=/home/marianasic/Discos/Disco1.disk
mkdisk -path=/home/marianasic/Discos/Disco1.disk -size=21 -unit=k
fdisk -path=/home/marianasic/Discos/Disco1.disk -type=e -name=extPrim -size=8 -unit=k -fit=wf
fdisk -path=/home/marianasic/Discos/Disco1.disk -name=Prim1 -fit=fF -size=2852 -unit=b
fdisk -path=/home/marianasic/Discos/Disco1.disk -name=primW -fit=wF -size=3 -unit=k
fdisk -path=/home/marianasic/Discos/Disco1.disk -name=primf -fit=wF -size=5 -unit=k
fdisk -path=/home/marianasic/Discos/Disco1.disk -name=primdff -fit=wF -size=5 -unit=k

fdisk -path=/home/marianasic/Discos/Disco1.disk -name=primW -unit=k -add=2
fdisk -path=/home/marianasic/Discos/Disco1.disk -name=primW -unit=k -add=-1

rep -path=/home/marianasic/Discos/Disco1.disk -name=mbR -id=repFinal
rep -path=/home/marianasic/Discos/Disco1.disk -name=diSk -id=repDisco

```

Se ejecutan todos los comandos contenidos en dicho archivo (ver Anexo 1).

Anexo

1. Resultado de ejecutar el script con el archivo de ejemplo “ejecutar.sh”



Nótese que todos los comandos contenidos en ese archivo se van ejecutando uno tras otro.

```
Fit: W
Tamano: 8192

Por el momento, hay 3 particion(es) primaria(s)
Por el momento, hay 1 particion(es) extendida(s)
Hacen un total de 4 particiones
No se ha creado ninguna particion
Agregrando 2048 espacio en la particion  3
Error: No se puede agregar porque ya no hay espacio disponible
No se ha eliminado ninguna particion
Reconocio path ... /home/marianasic/Discos/Disco1.disk
Reconocio name ... primW
Reconocio unit ... K
Reconocio size ... -1
Ejecutando comando fdisk ...
Fit: F
Sign: 180972556
Fecha: F 22/08/2019 H 23:10:52
Tamano: 21504
Tamano de la particion 0
--------------------
Inicio: 220
Nombre: Prim1
Estado: A
Tipo: P
Fit: F
Tamano: 2852
```