OS Report

- محمد فتحي سيد احمد
- محمد مدحت محمد علي

We have 6 classes in our Project :

1-

Fat_Table class :

- we have 11 methods to create,print,and make some mitions at fat table They called:

- (createFAT, WriteFatTable, readFAT, splitBytes, PrintFAT, Getavaliableblock, ToInt, GetAvilaibleBlocks, ToBytes, getnext, setnext).

```
31 references
class FatTable
{
    public static int[] fat_table = new int[1024];
    1 reference
    public static void createFAT()...
    5 references
    public static void WriteFatTable()...
    1 reference
    public static void readFAT()...
    0 references
    public static void printFAT()...
    6 references
    public static int Getavaliableblock()...
    1 reference
    public static byte[] ToBytes(int[] array)...
    7 references
    public static void setnext(int Index, int value)...
    3 references
    public static List<byte[]> splitBytes(byte[] bytes)...
    8 references
    public static int getnext(int Index)...
    1 reference
    public static int[] ToInt(byte[] bytes)...
    0 references
    public static int GetAvilaibleBlocks()...
}
```

2-

virsualDisk :

- we have 3 methods to initialize,write and read virsualDisk They called:

- (initialize, writeBlock, readBlock)

```
8 references
class VirtualDisk
{
    public static FileStream Vir_disk;
    1 reference
    public static void Initalize(string path)...
    4 references
    public static void WriteBlock(byte[] cluster, int clusterIndex, int offset = 0, int count = 1024)...
    3 references
    public static byte[] ReadBlock(int clusterIndex)...
}
```

3- Directory_Entry.

4- Directory.

5- file_Entry.

6-

CMD class:

- Consists of 13 method:
- (help, cls, md, rd, cd, del, copy, rename, export, import, type, quit, dir)

1-

HELP: command

- Notes:
- To display informations about all commands write help
- To display informations about specific command write help + [command]

- Purpose
- Use the HELP command to display informations about the commands. The HELP facility provides information for new and experienced users.

- help command syntax is
- help

or

- help [command]

For more information on a specific command.

```
1 reference
public static void Help()
{
    Console.WriteLine("cd       - Change the current default directory to .");
    Console.WriteLine("cls      - Clear the screen.");
    Console.WriteLine("dir      - List the contents of directory .");
    Console.WriteLine("quit     - Quit the shell.");
    Console.WriteLine("copy     - Copy files from and to your virual disk");
    Console.WriteLine("del      - Delete files.");
    Console.WriteLine("help     - Provides information for commands.");
    Console.WriteLine("md       - Creates a directory.");
    Console.WriteLine("rd       - Removes a directory.");
    Console.WriteLine("rename   - Renames a file.");
    Console.WriteLine("type     - Displays the contents of a text file.");
    Console.WriteLine("import   - import text file(s) from your computer");
    Console.WriteLine("export   - export text file(s) to your computer");
}
```

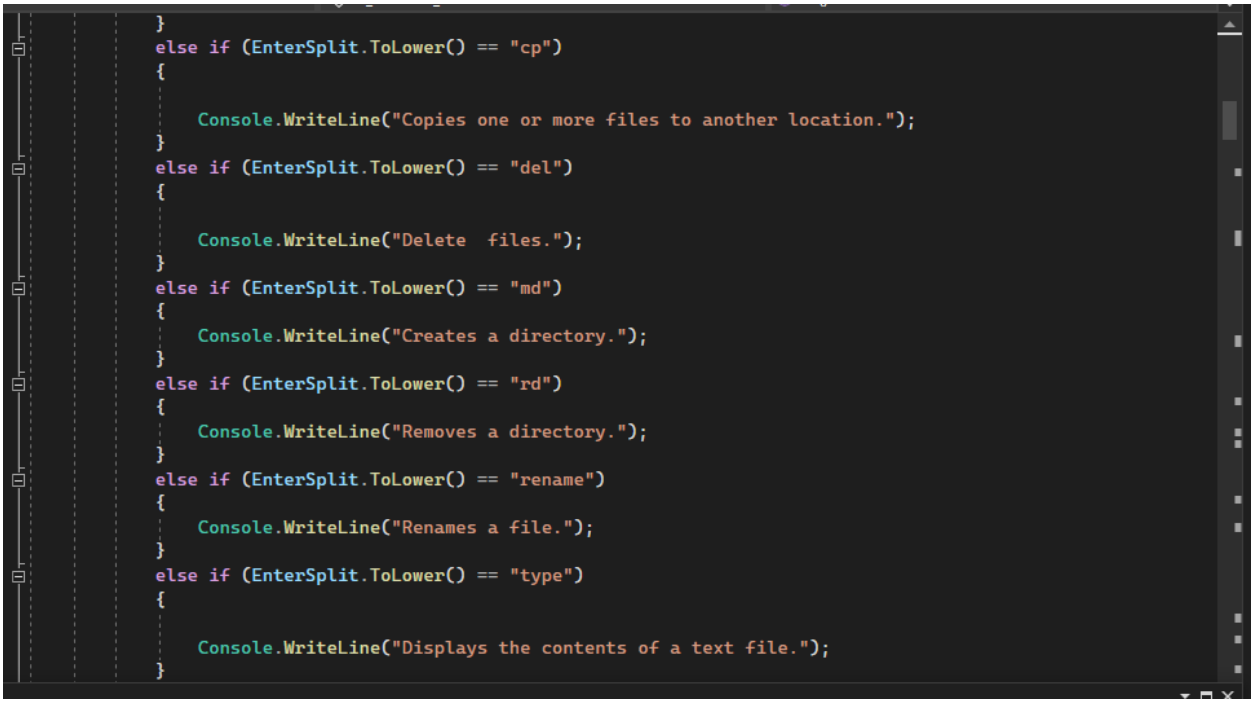
```
1 reference
public static void Help(string EnterSplit)
{
    if (EnterSplit.ToLower() == "cd")
    {
        Console.WriteLine("Change the current default directory to the directory given in the argument.
        Console.WriteLine("If the argument is not present, report the current directory.");
    }
    else if (EnterSplit.ToLower() == "help")
    {

        Console.WriteLine("Provides information for commands.");
    }
    else if (EnterSplit.ToLower() == "cls")
    {

        Console.WriteLine("Clear the screen.");
    }
    else if (EnterSplit.ToLower() == "dir")
    {

        Console.WriteLine("List the contents of directories or Files given in the argument.");
    }
    else if (EnterSplit.ToLower() == "quit")
    {

        Console.WriteLine("Quit the shell.");
    }
```

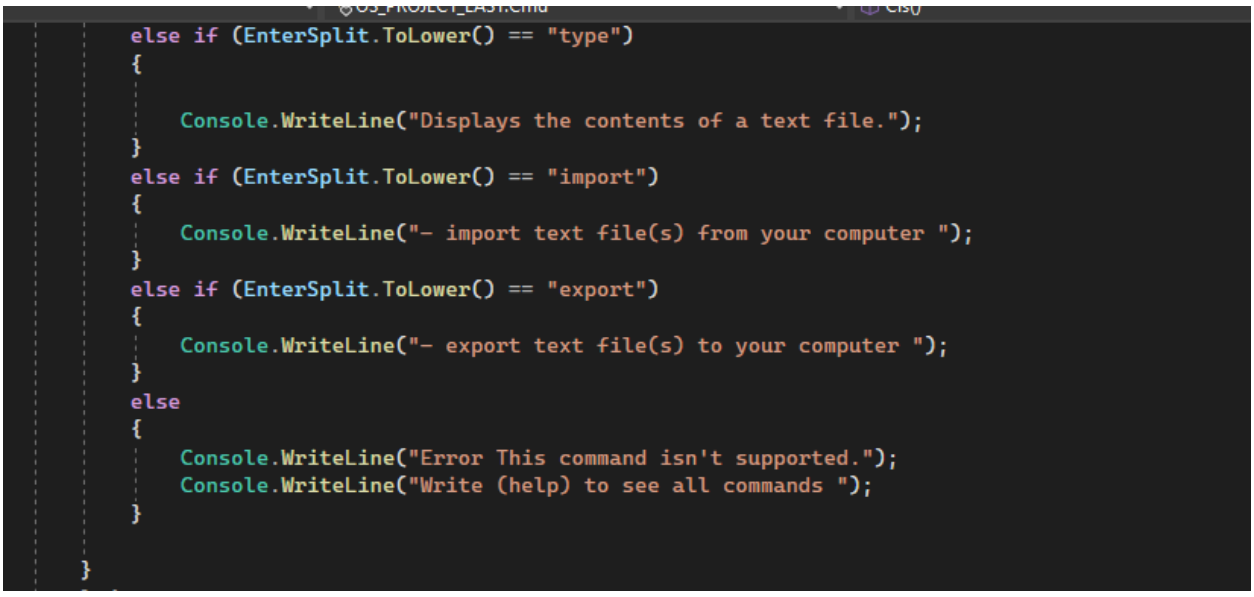


```
        }
        else if (EnterSplit.ToLower() == "cp")
        {

            Console.WriteLine("Copies one or more files to another location.");
        }
        else if (EnterSplit.ToLower() == "del")
        {

            Console.WriteLine("Delete  files.");
        }
        else if (EnterSplit.ToLower() == "md")
        {
            Console.WriteLine("Creates a directory.");
        }
        else if (EnterSplit.ToLower() == "rd")
        {
            Console.WriteLine("Removes a directory.");
        }
        else if (EnterSplit.ToLower() == "rename")
        {
            Console.WriteLine("Renames a file.");
        }
        else if (EnterSplit.ToLower() == "type")
        {

            Console.WriteLine("Displays the contents of a text file.");
        }
```



```
        else if (EnterSplit.ToLower() == "type")
        {

            Console.WriteLine("Displays the contents of a text file.");
        }
        else if (EnterSplit.ToLower() == "import")
        {
            Console.WriteLine("- import text file(s) from your computer ");
        }
        else if (EnterSplit.ToLower() == "export")
        {
            Console.WriteLine("- export text file(s) to your computer ");
        }
        else
        {
            Console.WriteLine("Error This command isn't supported.");
            Console.WriteLine("Write (help) to see all commands ");
        }

    }
```

2-

CD : command

- Notes

- If we write (~) in the argument, it will return to the root directory
- If we write (..) in the argument, it will return to the parent directory (a step back)
- If the directory does not exist an appropriate error should be reported.

- Purpose
- Use the CD command to change the working directory or display the current working directory.

- cd command syntax is
- cd

or

- cd [directory]
- [directory] can be directory name or fullpath of a directory or more.

```
1 reference
public static void Cd()
{
    Console.WriteLine("Current Path : " + Program.currentPath);

}
```

```
3 references
public static void Cd(string Name)
{
    char[] separators = new char[2];
    separators[0] = '\\';

    string[] nam = Name.Split(separators);
    int index = Program.currentDirectory.SearchDirectory(Name);
    if (nam.Length > 1)
    {
        string s = " ";
        string y = " ";
        if (Program.currentDirectory.parent == null)
        {
            for (int q = 0; q < nam.Length - 1;)
            {

                s = new string(Program.currentDirectory.Name).Trim();
                y = new string(nam[q]) + "\0";

                if (y == s)
                {
                    int inde = Program.currentDirectory.SearchDirectory(nam[q + 1]);
                    if (inde != -1)
                    {


                        if (Program.currentDirectory.DirectoryTable[inde].fileAttribute == 0x10)
                        {
                            int firstCluster = Program.currentDirectory.DirectoryTable[inde].FileFirstCluster;
                            int fileSize = Program.currentDirectory.DirectoryTable[inde].FileSize;
                            Directory1 d1 = new Directory1(nam[q + 1], 0x10, firstCluster, fileSize, Program.currentDirectory);
                            Program.currentPath = Program.currentPath.Trim();
                            Program.currentPath += "\\" + new string(d1.Name).Trim();
```

```
                    Program.currentDirectory = d1;
                    Program.currentDirectory.ReadDirectory();
                }
                else
                {
                    Console.WriteLine(nam[q + 1] + "is not a Directory");

                    break;
                }

            }
            else
            {

                Console.WriteLine("Path not correct");
                break;
            }

        }
        q++;
    }

}
else
{
    while (true)
    {
        if (Program.currentDirectory.parent == null)
        {
            break;
        }
        Program.currentDirectory = Program.currentDirectory.parent;
    }
```

```
else if (nam.Length == 1)
{
    if (Name == "..")
    {
        if (Program.currentDirectory.parent != null)
        {
            int goodPrint = Program.currentPath.LastIndexOf("\\");
            Program.currentPath = Program.currentPath.Substring(0, goodPrint);
            Program.currentPath = Program.currentPath + " ";
            Program.currentDirectory = Program.currentDirectory.parent;
            Program.currentDirectory.ReadDirectory();
        }
        else
            Console.WriteLine("This is The root Directory " + Program.currentPath);
    }
    if (Name == "~")
    {
        if (Program.currentDirectory.parent != null)
        {
            bool a = true;
            while (a)
            {
                if (Program.currentDirectory.parent == null)
                {
                    a = false;
                    break;
                }
                Program.currentDirectory = Program.currentDirectory.parent;
            }
            Program.currentPath = new string(Program.currentDirectory.Name) + " ";
            Program.currentDirectory.ReadDirectory();
        }
        else
            Console.WriteLine("This is The root Directory " + Program.currentPath);
```

3-

CLS : command

- Purpose

- The cls command clears the screen of all previously entered commands and other text.

- cls command syntax is

- cls

```
public static void Cls()
{
    Console.Clear();
}
```

4-

DIR : command

- Purpose

- The dir command is used to display a list of files and folders contained inside the folder that you are currently working in.

- dir command syntax is

- dir

```
public static void Dir()
{
    string name = " ";
    int numOfFiles = 0, numOfOFolders = 0, sizeOfFiles = 0;
    Console.WriteLine("\nDirectory of : " + Program.currentPath.Trim()+"\n");

    for (int i = 0; i < Program.currentDirectory.DirectoryTable.Count; i++)
    {
        if (Program.currentDirectory.DirectoryTable[i].fileAttribute == 0x10)
        {
            numOfOFolders += 1;
            for (int j = 0; j < Program.currentDirectory.DirectoryTable[i].Name.Length; j++)
            {
                name += Program.currentDirectory.DirectoryTable[i].Name[j];
            }
            Console.WriteLine("<DIR>\t  " + name);
            name = " ";
        }
        else
        {
            numOfFiles += 1;
            sizeOfFiles += Program.currentDirectory.DirectoryTable[i].FileSize;
            for (int j = 0; j < Program.currentDirectory.DirectoryTable[i].Name.Length; j++)
            {
                name += Program.currentDirectory.DirectoryTable[i].Name[j];
            }
            Console.WriteLine("       " + Program.currentDirectory.DirectoryTable[i].FileSize + " " + name);
            name = " ";
        }
    }
    int FreeSpace = FatTable.Getavaliableblock() * 1024;
    Console.Write("\t    " + numOfFiles + " File(s)\t" +   + sizeOfFiles +"bytes"+ "\n\t    " + numOfOFolders + " Dir(s)"  + "\t"
}
```

5-

QUIT : command

- Purpose
- The command causes the shell or program to terminate.

- quit command syntax is
- quit

```
2 references
public static void Quit()
{
        FatTable.WriteFatTable();
        VirtualDisk.Vir_disk.Close();
        Environment.Exit(0);
}
1 reference
```

6-

COPY : command

- Purpose
- The copy command does simply that — it copy files from one location to another.

- copy command syntax is
- cp [source] [destination]
- [source] can be file Name (or fullpath of file).
- [destination] can be file Name (or fullpath of file) or directory name or fullpath of a directory.

```
public static void cp(string Path, string dest)
{
    string s = Program.currentPath;
    Directory1 d2 = Program.currentDirectory;
    export(Path, "F:");
    Cd(dest);
    string n = "F:\\" + Path;

    Import(n);
    Program.currentDirectory = d2;
    Program.currentPath = s;
}
```

7-

Rename : command

- Purpose

- the rename command is used to change the name of the individual file that you specify. The rename command is available in all versions of Windows

- rename command syntax is

- rename [fileName] [new fileName]
- [fileName] can be a file name or fullpath of a filename

[new fileName] can be a new file name not fullpath

```
1 reference
public static void rename(string oldName, string newName)
{
    int oldIndex = Program.currentDirectory.SearchDirectory(oldName);
    if (oldIndex != -1)
    {
        int newIndex = Program.currentDirectory.SearchDirectory(new string(newName));

        if (newIndex == -1)
        {
            Directory_Entry d1 = new Directory_Entry(newName, Program.currentDirectory.DirectoryTable[oldIndex].fileAttribute, Pro
            Program.currentDirectory.DirectoryTable.RemoveAt(oldIndex);
            Program.currentDirectory.DirectoryTable.Insert(oldIndex, d1);
            Program.currentDirectory.WriteDirectory();
        }
        else
        {
            Console.WriteLine("This Name(" + newName + ") is already exist");
        }
    }
    else
    {
        Console.WriteLine(oldName + " isn't exist");
    }
}
```

8-

Md: command

- Purpose

- The md command is the shorthand version of the mkdir command. The md command is available in all versions of Windows

- md command syntax is

- md [directory]
- [directory] can be a new directory name or fullpath of a new directory

```
1 reference
public static void Md(string Name)
{
    int index = Program.currentDirectory.SearchDirectory(Name);
    if (index == -1)
    {
        Directory_Entry newdirectory = new Directory_Entry(Name, 0x10, 0, 0);
        Directory1 name = new Directory1(Name, 0x10, 0, 0, Program.currentDirectory);
        Program.currentDirectory.DirectoryTable.Add(newdirectory);
        Program.currentDirectory.WriteDirectory();
        if (Program.currentDirectory.parent != null)
        {
            Program.currentDirectory.parent.Update(Program.currentDirectory.parent);
            Program.currentDirectory.parent.WriteDirectory();
        }
    }
    else
    {
        if (Program.currentDirectory.DirectoryTable[index].fileAttribute == 0x10)
        {
            Console.WriteLine( Name + " already exists.");
        }
        else
        {
            Console.WriteLine( Name + " already exists.");
        }
    }
```

9-

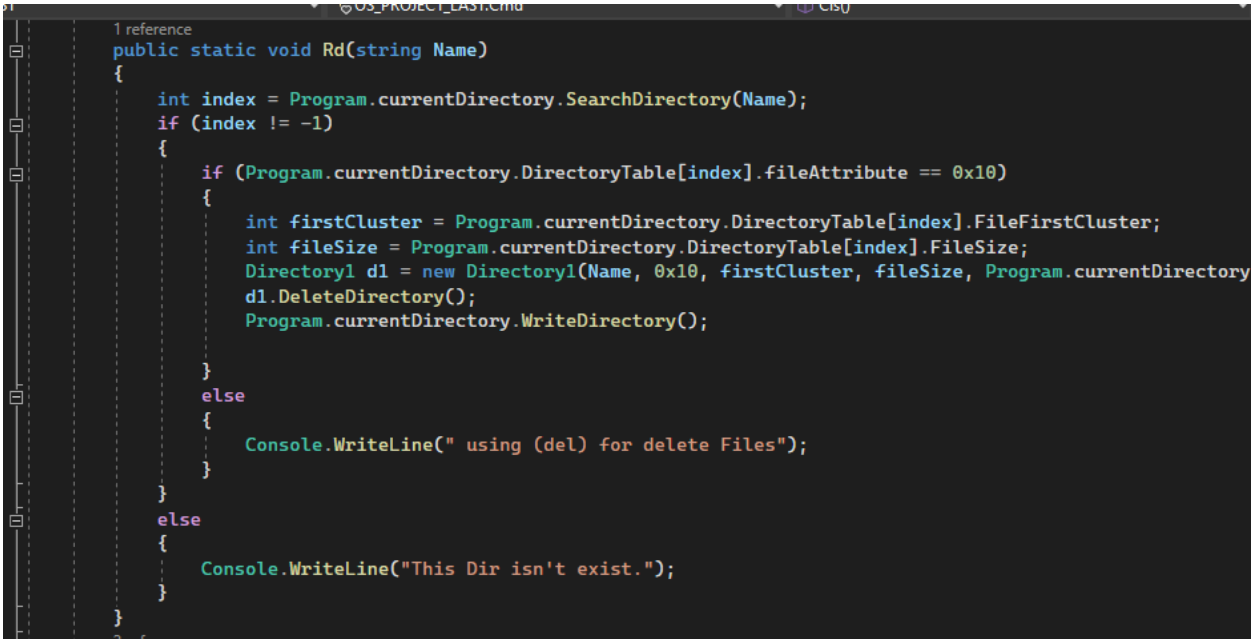
Rd : command

- Purpose

- is a command which will remove an empty directory on various operating systems.

- rd command syntax is

- rd [directory]
- [directory] can be a directory name or fullpath of a directory



```
1 reference
public static void Rd(string Name)
{
    int index = Program.currentDirectory.SearchDirectory(Name);
    if (index != -1)
    {
        if (Program.currentDirectory.DirectoryTable[index].fileAttribute == 0x10)
        {
            int firstCluster = Program.currentDirectory.DirectoryTable[index].FileFirstCluster;
            int fileSize = Program.currentDirectory.DirectoryTable[index].FileSize;
            Directory1 d1 = new Directory1(Name, 0x10, firstCluster, fileSize, Program.currentDirectory
            d1.DeleteDirectory();
            Program.currentDirectory.WriteDirectory();

        }
        else
        {
            Console.WriteLine(" using (del) for delete Files");
        }
    }
    else
    {
        Console.WriteLine("This Dir isn't exist.");
    }
}
```

10-

Type : command

- Purpose

- The type command is used to display the information contained in a text file. The type command is available in all versions of Windows

- type command syntax is

- type [file]+
- [file] can be file Name (or fullpath of file) of text file

```
1 reference
public static void Type(string Name)
{
    int index = Program.currentDirectory.SearchDirectory(Name);
    if (index != -1)
    {
        if (Program.currentDirectory.DirectoryTable[index].fileAttribute == 0x0)
        {
            int FirstCluster = Program.currentDirectory.DirectoryTable[index].FileFirstCluster;
            int FileSize = Program.currentDirectory.DirectoryTable[index].FileSize;
            string Content = string.Empty;
            File_Entry file = new File_Entry(Name, 0x0, FirstCluster, FileSize, Program.currentDirectory, Content);
            file.ReadFileContent();
            Console.WriteLine(file.content);
        }
        else
        {
            Console.WriteLine("Write file name  ");
        }
    }
    else
    {
        Console.WriteLine("No file such name");
    }
}
```

11-

import : command

- Purpose

- import text file(s) from your computer

- import command syntax is

- import [source]
- [source] can be file Name (or fullpath of file) or directory Name (or fullpath of directory) from your physical disk

```
2 references
public static void Import(string Path)
{
    string Name = "";
    if (File.Exists(Path))
    {
        char[] separators = new char[1];
        separators[0] = '\\';
        string[] nam = Path.Split(separators);
        Name = nam[nam.Length - 1];

        string Content = "";
        int index = Program.currentDirectory.SearchDirectory(Name);
        if (index == -1)
        {
            Content += File.ReadAllText(Path);
            int size = Content.Length;
            int firstCluster = 0;
            if (size > 0)
            {
                firstCluster = FatTable.Getavaliableblock();
            }

            File_Entry file = new File_Entry(Name, 0x0, firstCluster, size, Program.currentDirectory, Content);
            file.writeFileContent();
            Directory_Entry f = new Directory_Entry(Name, 0x0, firstCluster, size);
            Program.currentDirectory.DirectoryTable.Add(f);

            Program.currentDirectory.WriteDirectory();

            if (Program.currentDirectory.parent != null)
```

```
            if (Program.currentDirectory.parent != null)
            {
                Program.currentDirectory.parent.Update(Program.currentDirectory.parent);
                Program.currentDirectory.parent.WriteDirectory();
            }
                        (field) static Directory1 Program.currentDirectory


        }
        else
        {
            if (Program.currentDirectory.DirectoryTable[index].fileAttribute == 0x10)
            {
                Console.WriteLine(Name + "  exists as a Directory  ");
            }
        }
    }
    else if (!File.Exists(Path))
    {
        Console.WriteLine("Path isn't correct");
    }
}
```

12-

export: command

- **purpose**

- export text file(s) to your computer

- export command syntax is

- export [source]
- [source] can be file Name (or fullpath of file) or directory

Name (or fullpath of directory) from your virtual disk

```
public static int export(string source, string dest)
{
    int name_start = source.LastIndexOf(".");
    string filename = source.Substring(name_start + 1);
    if (filename == "txt")
    {
        int index = Program.currentDirectory.SearchDirectory(source);
        if (index != -1)
        {
            if (Program.currentDirectory.DirectoryTable[index].fileAttribute == 0x0)
            {
                if (Directory.Exists(dest))
                {
                        int cluster = Program.currentDirectory.DirectoryTable[index].FileFirstCluster;
                        int size = Program.currentDirectory.DirectoryTable[index].FileSize;
                        string content = null;
                        File_Entry file = new File_Entry(source, 0x0, cluster, size, Program.currentDirectory, content);
                        file.ReadFileContent();
                        StreamWriter st = new StreamWriter(dest + "\\" + source);
                        st.Write(file.content);
                        st.Flush();
                        st.Close();
                        return 1;
                }
                else
                {
                    Console.WriteLine("The system can't find this Path");
                    return 0;
                }
            }
            else
            {
                Console.WriteLine(source + "  This is Directory");
```

(parameter) string source

13-

del: command

- **purpose**

- delete files from the virsual disk.

- del command syntax is

- del [File Name]

```
1 reference
public static void del(string fileName)
{
    int index = Program.currentDirectory.SearchDirectory(fileName);
    if (index != -1)
    {
        if (Program.currentDirectory.DirectoryTable[index].fileAttribute == 0x0)
        {
            int cluster = Program.currentDirectory.DirectoryTable[index].FileFirstCluster;
            int size = Program.currentDirectory.DirectoryTable[index].FileSize;
            File_Entry f = new File_Entry(fileName, 0x0, cluster, size, Program.currentDirectory, null);
            Program.currentDirectory.DirectoryTable.RemoveAt(index);
            Program.currentDirectory.WriteDirectory();
        }
        else
        {
            Console.WriteLine("use (rd) command fot deleting Directory");
        }

    }
    else
    {
        Console.WriteLine("  no file with this name");
    }
}
```