using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Barca

{

class Virsual\_Disk

{

public static FileStream Disk;

public static void CREATE\_Disk(string path)

{

Disk = new FileStream(path, FileMode.Create, FileAccess.ReadWrite);

Disk.Close();

}

public static int getFreeSpace()

{

return (1024 \* 1024) - (int)Disk.Length;

}

public static void initalize(string path)

{

if (!File.Exists(path))

{

CREATE\_Disk(path);

byte[] b = new byte[1024];

for (int i = 0; i < b.Length; i++)

{

b[i] = 0;

}

writeBlock(b, 0);

Fat\_Table f = new Fat\_Table();

Fat\_Table.initialize();

Directory root = new Directory("K:\\>", 0x10, 5, 0, null);

root.Write\_Directory();

Fat\_Table.set\_Next(5, -1);

Program.currentDirectory = root;

Fat\_Table.Write\_Fat\_Table();

}

else

{

Fat\_Table.get\_fat\_table();

Directory root = new Directory("K:\\>", 0x10, 5, 0, null);

root.readDirectory();

Program.currentDirectory = root;

}

}

public static void writeBlock(byte[] data, int Index, int offset = 0, int count = 1024)

{

Disk = new FileStream("File.txt", FileMode.Open, FileAccess.Write);

Disk.Seek(Index \* 1024, SeekOrigin.Begin);

Disk.Write(data, offset, count);

Disk.Flush();

Disk.Close();

}

public static byte[] readBlock(int clusterIndex)

{

Disk = new FileStream("File.txt", FileMode.Open, FileAccess.Read);

Disk.Seek(clusterIndex \* 1024, SeekOrigin.Begin);

byte[] bytes = new byte[1024];

Disk.Read(bytes, 0, 1024);

Disk.Close();

return bytes;

}

}

class Fat\_Table

{

static int[] fat\_table = new int[1024];

static byte[] arrOfByte = new byte[4 \* 1024];

public Fat\_Table()

{

fat\_table = new int[1024];

}

public static void initialize()

{

for (int i = 0; i < fat\_table.Length; i++)

{

if (i < 5)

fat\_table[i] = -1;

else

fat\_table[i] = 0;

}

}

public static void Write\_Fat\_Table()

{

Virsual\_Disk.Disk = new FileStream("File.txt", FileMode.Open, FileAccess.Write);

Virsual\_Disk.Disk.Seek(1024, SeekOrigin.Begin);

Buffer.BlockCopy(fat\_table, 0, arrOfByte, 0, arrOfByte.Length);

Virsual\_Disk.Disk.Write(arrOfByte, 0, arrOfByte.Length);

Virsual\_Disk.Disk.Close();

}

public static int[] get\_fat\_table()

{

Virsual\_Disk.Disk = new FileStream("File.txt", FileMode.Open, FileAccess.Read);

Virsual\_Disk.Disk.Seek(1024, SeekOrigin.Begin);

Virsual\_Disk.Disk.Read(arrOfByte, 0, arrOfByte.Length);

Buffer.BlockCopy(arrOfByte, 0, fat\_table, 0, 4096);

Virsual\_Disk.Disk.Close();

return (fat\_table);

}

public void Print\_fat\_table()

{

get\_fat\_table();

for (int i = 0; i < fat\_table.Length; i++)

{

Console.WriteLine((i + 1) + "\t-->\t" + fat\_table[i]);

}

}

public static int Getavaliableblock()

{

int freeIndex = -1;

for (int i = 0; i < 1024; i++)

{

if (fat\_table[i] == 0)

{

freeIndex = i;

break;

}

}

return freeIndex;

}

public static int GetAvilaibleBlocks()

{

int count = 0;

for (int i = 0; i < fat\_table.Length; i++)

{

if (fat\_table[i] == 0)

{

count++;

}

}

return count;

}

public static int get\_Next(int index)

{

return (fat\_table[index]);

}

public static void set\_Next(int index, int value)

{

fat\_table[index] = value;

}

public static int Get\_free\_space()

{

return GetAvilaibleBlocks() \* 1024;

}

}

class Directory\_Entry

{

public char[] fileorDirName = new char[11];

public byte filaAttribute;

public byte[] fileEmpty = new byte[12];

public int fileSize;

public int fileFirstCluster;

public Directory\_Entry(string name, byte attribute, int firstCluster, int fsize)

{

filaAttribute = attribute;

if (filaAttribute == 0x0)

{

string[] filename = name.Split('.');

assignFileName(filename[0].ToCharArray(), filename[1].ToCharArray());

}

else

{

assignDIRName(name.ToCharArray());

}

fileFirstCluster = firstCluster;

fileSize = fsize;

}

public void assignFileName(char[] name, char[] extension)

{

if (name.Length <= 7 && extension.Length == 3)

{

int j = 0;

for (int i = 0; i < name.Length; i++)

{

j++;

this.fileorDirName[i] = name[i];

}

j++;

this.fileorDirName[j] = '.';

for (int i = 0; i < extension.Length; i++)

{

j++;

this.fileorDirName[j] = extension[i];

}

for (int i = ++j; i < fileorDirName.Length; i++)

{

this.fileorDirName[i] = ' ';

}

}

else

{

for (int i = 0; i < 7; i++)

{

this.fileorDirName[i] = name[i];

}

this.fileorDirName[7] = '.';

for (int i = 0, j = 8; i < extension.Length; j++, i++)

{

this.fileorDirName[j] = extension[i];

}

}

}

public void assignDIRName(char[] name)

{

if (name.Length <= 11)

{

int j = 0;

for (int i = 0; i < name.Length; i++)

{

j++;

this.fileorDirName[i] = name[i];

}

for (int i = ++j; i < fileorDirName.Length; i++)

{

this.fileorDirName[i] = ' ';

}

}

else

{

int j = 0;

for (int i = 0; i < 11; i++)

{

j++;

this.fileorDirName[i] = name[i];

}

}

}

public byte[] GetBytes()

{

byte[] b = new byte[32];

for (int i = 0; i < 11; i++)

{

b[i] = (byte)fileorDirName[i];

}

b[11] = filaAttribute;

for (int i = 12, j = 0; i < 24 && j < 12; i++, j++)

{

b[i] = fileEmpty[j];

}

for (int i = 24; i < 28; i++)

{

b[i] = (byte)fileFirstCluster;

}

for (int i = 28; i < 32; i++)

{

b[i] = (byte)fileSize;

}

return b;

}

public Directory\_Entry GetDirectoryEntry(byte[] b)

{

for (int i = 0; i < 11; i++)

{

fileorDirName[i] = (char)b[i];

}

filaAttribute = b[11];

for (int i = 12, j = 0; i < 24 && j < 12; i++, j++)

{

fileEmpty[j] = b[i];

}

for (int i = 24; i < 28; i++)

{

fileFirstCluster = b[i];

}

for (int i = 28; i < 32; i++)

{

fileSize = b[i];

}

Directory\_Entry d1 = new Directory\_Entry(new string(fileorDirName), filaAttribute, fileFirstCluster, fileSize);

return d1;

}

}

class Directory : Directory\_Entry

{

public List<Directory\_Entry> directoryTable;

public Directory parent = null;

public Directory(string namefile, byte attributefile, int firstClusturfile, int sizefile, Directory p)

: base

(namefile,

attributefile,

firstClusturfile,

sizefile)

{

directoryTable = new List<Directory\_Entry>();

if (p != null)

{

parent = p;

}

}

public Directory\_Entry GetDirectory\_Entry()

{

Directory\_Entry me = new Directory\_Entry(new string(fileorDirName), filaAttribute, fileFirstCluster, fileSize);

return me;

}

public void Write\_Directory()

{

int lastIndex = -1;

byte[] DTB = new byte[32 \* directoryTable.Count];

byte[] DEB = new byte[32];

for (int i = 0; i < directoryTable.Count; i++)

{

DEB = directoryTable[i].GetBytes();

for (int j = i \* 32, c = 0; c < 32; c++, j++)

{

DTB[j] = DEB[c];

}

}

double NFQB = (DTB.Length / 1024);

NFQB = Math.Ceiling(NFQB);

int Reminder = (DTB.Length % 1024);

int fatIndex;

if (NFQB <= Fat\_Table.GetAvilaibleBlocks())

{

if (this.fileFirstCluster != 0)

{

fatIndex = this.fileFirstCluster;

}

else

{

fatIndex = Fat\_Table.Getavaliableblock();

this.fileFirstCluster = fatIndex;

}

List<byte[]> Blocks = new List<byte[]>();

byte[] block = new byte[1024];

for (int j = 0; j < NFQB \* 1024; j++)

{

if (j % 1024 == 0 && j != 0)

{

Blocks.Add(block);

}

block[j % 1024] = DTB[j];

}

for (int i = 0; i < NFQB; i++)

{

Virsual\_Disk.writeBlock(Blocks[i], fatIndex);

Fat\_Table.set\_Next(fatIndex, -1);

if (lastIndex != -1)

{

Fat\_Table.set\_Next(lastIndex, fatIndex);

}

lastIndex = fatIndex;

fatIndex = Fat\_Table.Getavaliableblock();

}

Fat\_Table.Write\_Fat\_Table();

}

}

public void readDirectory()

{

if (this.fileFirstCluster != 0)

{

int fatIndex = this.fileFirstCluster;

int next = Fat\_Table.get\_Next(fatIndex);

List<byte> ls = new List<byte>();

List<Directory\_Entry> dt = new List<Directory\_Entry>();

do

{

ls.AddRange(Virsual\_Disk.readBlock(fatIndex));

fatIndex = next;

if (fatIndex != -1)

{

next = Fat\_Table.get\_Next(fatIndex);

}

} while (next != -1);

for (int i = 0; i < ls.Count; i++)

{

byte[] b = new byte[32];

for (int k = i \* 32, m = 0; m < b.Length && k < ls.Count; m++, k++)

{

b[m] = ls[k];

}

if (b[0] == 0)

break;

dt.Add(GetDirectoryEntry(b));

}

}

}

public int searchDirectory(string name)

{

if (name.Length < 11)

{

name += "\0";

for (int i = name.Length + 1; i < 12; i++)

name += " ";

}

else

{

name = name.Substring(0, 11);

}

for (int i = 0; i < directoryTable.Count; i++)

{

string n = new string(directoryTable[i].fileorDirName);

if (n.Equals(name))

return i;

}

return -1;

}

public void updateContent(Directory\_Entry d)

{

string name = new string(d.fileorDirName);

readDirectory();

int index = searchDirectory(name);

if (index != -1)

{

directoryTable.RemoveAt(index);

directoryTable.Insert(index, d);

}

Write\_Directory();

}

public void deleteDirectory()

{

if (this.fileFirstCluster != 0)

{

int index = this.fileFirstCluster;

int next = -1;

do

{

Fat\_Table.set\_Next(index, 0);

next = index;

if (index != -1)

index = Fat\_Table.get\_Next(index);

} while (next != -1);

}

if (this.parent != null)

{

parent.readDirectory();

int Index = parent.searchDirectory(new string(fileorDirName));

if (Index != -1)

{

this.parent.directoryTable.RemoveAt(Index);

this.parent.Write\_Directory();

}

}

}

}

class File\_Entry : Directory\_Entry

{

public string content;

public Directory parent;

public File\_Entry(string name, byte f\_attr, int f\_firstCluster, int f\_size, string f\_content, Directory pa) : base(name, f\_attr, f\_firstCluster, f\_size)

{

content = f\_content;

if (pa != null)

parent = pa;

}

public void writeFileContent()

{

byte[] contentBYTES = Encoding.ASCII.GetBytes(content);

double numOfBlocks = contentBYTES.Length / 1024;

int numOfRequiredBlock = Convert.ToInt32(Math.Ceiling(numOfBlocks));

int numOfFullSizeBlock = Convert.ToInt32(Math.Floor(numOfBlocks));

double reminder = contentBYTES.Length % 1024;

int fatIndex = 0;

int lastIndex = -1;

if (numOfRequiredBlock <= Fat\_Table.GetAvilaibleBlocks())

{

if (fileFirstCluster != 0)

{

fatIndex = fileFirstCluster;

}

else

{

fileFirstCluster = Fat\_Table.Getavaliableblock();

fatIndex = Fat\_Table.Getavaliableblock();

}

}

for (int i = 0; i < numOfFullSizeBlock; i++)

{

Virsual\_Disk.writeBlock(contentBYTES, fatIndex);

Fat\_Table.set\_Next(fatIndex, -1);

if (lastIndex != -1)

{

lastIndex = fatIndex;

Fat\_Table.set\_Next(lastIndex, fatIndex);

}

fatIndex = Fat\_Table.Getavaliableblock();

Fat\_Table.Write\_Fat\_Table();

}

}

List<byte> ls;

public void readFileContent()

{

int fatIndex = fileFirstCluster;

int next = Fat\_Table.get\_Next(fatIndex);

if (fileFirstCluster != 0)

{

do

{

ls.AddRange(Virsual\_Disk.readBlock(fatIndex));

fatIndex = next;

if (fatIndex != -1)

{

next = Fat\_Table.get\_Next(fatIndex);

}

} while (next != -1);

}

}

public void deleteFileContent()

{

if (fileFirstCluster != 0)

{

int index = fileFirstCluster;

int next = -1;

do

{

Fat\_Table.set\_Next(index, 0);

next = index;

if (index != -1)

index = Fat\_Table.get\_Next(index);

} while (next != -1);

}

if (parent != null)

{

parent.readDirectory();

int Index = parent.searchDirectory(fileorDirName.ToString());

if (Index != -1)

{

parent.directoryTable.RemoveAt(Index);

parent.Write\_Directory();

}

}

}

}

class cmd

{

public static void cls()

{

Console.Clear();

}

public static void help()

{

Console.WriteLine("CD Change the current default directory ");

Console.WriteLine("CLS Clear the screen ");

Console.WriteLine("DIR List the contents of directory ");

Console.WriteLine("QUIT Quit the shell ");

Console.WriteLine("COPY Copies one or more files to another location ");

Console.WriteLine("DEL Deletes one or more files ");

Console.WriteLine("HELP Provides Help 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");

}

public static void quit()

{

System.Environment.Exit(1);

}

public static void md(string name)

{

if (Program.currentDirectory.searchDirectory(name) == -1)

{

Directory\_Entry newdirectory = new Directory\_Entry(name, 0x10, 0, 0);

Program.currentDirectory.directoryTable.Add(newdirectory);

Program.currentDirectory.Write\_Directory();

if (Program.currentDirectory.parent != null)

{

Program.currentDirectory.parent.updateContent(Program.currentDirectory.parent);

Program.currentDirectory.parent.Write\_Directory();

}

}

else

{

Console.WriteLine("A subdirectory or file " + name + " already exists.");

}

}

public static void rd(string name)

{

int index = Program.currentDirectory.searchDirectory(name);

if (index != -1)

{

int firstCluster = Program.currentDirectory.directoryTable[index].fileFirstCluster;

Directory d1 = new Directory(name, 0x10, firstCluster, 0, Program.currentDirectory);

d1.deleteDirectory();

Program.currentPath = new string(Program.currentDirectory.fileorDirName).Trim();

}

else

{

Console.WriteLine("The system cannot find the path specified.");

}

}

public static void cd(string name)

{

int index = Program.currentDirectory.searchDirectory(name);

if (index != -1)

{

int firstCluster = Program.currentDirectory.directoryTable[index].fileFirstCluster;

Directory d1 = new Directory(name, 0x10, firstCluster, 0, Program.currentDirectory);

Program.currentPath = new string(Program.currentDirectory.fileorDirName).Trim() + "\\" + new string(d1.fileorDirName).Trim();

Program.currentDirectory.readDirectory();

}

else

{

Console.WriteLine("The system cannot find the path specified.");

}

}

public static void dir()

{

int counterDirectory = 0, counterfiles = 0, filesizecounter = 0;

Console.WriteLine("Directory of " + Program.currentPath);

for (int i = 0; i < Program.currentDirectory.directoryTable.Count; i++)

{

if (Program.currentDirectory.filaAttribute == 0x0)

{

Console.WriteLine(Program.currentDirectory.directoryTable[i].fileSize + " " + Program.currentDirectory.directoryTable[i].fileorDirName);

counterfiles++;

filesizecounter += Program.currentDirectory.directoryTable[i].fileSize;

}

else

{

Console.Write("<dir>" + " ");

Console.WriteLine(Program.currentDirectory.directoryTable[i].fileorDirName);

counterDirectory++;

}

}

Console.WriteLine(counterfiles + " File(s) " + filesizecounter + " bytes");

Console.WriteLine(counterDirectory + " Dir(s) " + Fat\_Table.Get\_free\_space() + " bytes Free");

}

public static void import(string path)

{

if (File.Exists(path))

{

int start\_name = path.LastIndexOf("\\");

string name = path.Substring(start\_name + 1);

string content = File.ReadAllText(path);

int size = content.Length;

int index = Program.currentDirectory.searchDirectory(name);

if (index == -1)

{

int firstCluster;

if (size > 0)

{

firstCluster = Fat\_Table.Getavaliableblock();

}

else

{

firstCluster = 0;

}

File\_Entry f1 = new File\_Entry(name, 0x0, firstCluster, 0, content, Program.currentDirectory);

f1.writeFileContent();

Directory\_Entry d1 = new Directory\_Entry(name, 0x0, firstCluster, 0);

Program.currentDirectory.directoryTable.Add(d1);

Program.currentDirectory.Write\_Directory();

}

else

{

Console.WriteLine("This file already exist");

}

}

else

{

Console.WriteLine("This file is not exist");

}

}

public static void type(string name)

{

int index = Program.currentDirectory.searchDirectory(name);

if (index != -1)

{

int first\_cluster = Program.currentDirectory.directoryTable[index].fileFirstCluster;

int filesize = Program.currentDirectory.directoryTable[index].fileSize;

string content = null;

File\_Entry f = new File\_Entry(name, 0x0, first\_cluster, filesize, content, Program.currentDirectory);

f.readFileContent();

Console.WriteLine(f.content);

}

else

{

Console.WriteLine("The system can't find the file ");

}

}

public static void export(string source, string destination)

{

int index = Program.currentDirectory.searchDirectory(source);

if (index != -1)

{

if (System.IO.Directory.Exists(destination))

{

int first\_cluster = Program.currentDirectory.directoryTable[index].fileFirstCluster;

int filesize = Program.currentDirectory.directoryTable[index].fileSize;

string temp = null;

File\_Entry f = new File\_Entry(source, 0x0, first\_cluster, filesize, temp, Program.currentDirectory);

f.readFileContent();

StreamWriter sw = new StreamWriter(destination + "\\" + source);

sw.Write(f.content);

sw.Flush();

sw.Close();

}

else

{

Console.WriteLine("the system can't find this path in computer Disk");

}

}

else

{

Console.WriteLine("This file doesn't exist");

}

}

/\*public static void rename(string oldname, string newname)

{

int index = Program.currentDirectory.searchDirectory(oldname);

if (index != -1)

{

int n = Program.currentDirectory.searchDirectory(newname);

if (n != -1)

{

Directory\_Entry d = new Dirctory\_Entry();

d = Program.currentDirectory.directoryTable[index];

d.file\_name = newname;

Program.currentDirectory.directoryTable.RemoveAt(index);

Program.currentDirectory.directoryTable.Insert(n, d);

Program.currentDirectory.Write\_Directory();

}

else

{

Console.WriteLine("A duplicate file name exists, or the file cannot be found.");

}

}

else

{

Console.WriteLine(" The system cannot find the file specified.");

}

}\*/

public static void del(string name)

{

int index = Program.currentDirectory.searchDirectory(name);

if (index != -1)

{

int f = Program.currentDirectory.directoryTable[index].filaAttribute;

if (f == 0x0)

{

int first\_cluster = Program.currentDirectory.directoryTable[index].fileFirstCluster;

int file\_size = Program.currentDirectory.directoryTable[index].fileSize;

File\_Entry d = new File\_Entry(name, 0x0, first\_cluster, 0, null, Program.currentDirectory);

d.deleteFileContent();

}

else

{

Console.WriteLine(" The system cannot find the file specified. ");

}

}

else

{

Console.WriteLine(" The system cannot find the file specified. ");

}

}

public static void copy(string num1, string num2)

{

int index1 = Program.currentDirectory.searchDirectory(num1);

if (index1 != -1)

{

int start\_index = num2.LastIndexOf("\\");

string name = num2.Substring(start\_index + 1);

int index\_destenation = Program.currentDirectory.searchDirectory(name);

if (index\_destenation == -1)

{

if (num2 != Program.currentDirectory.fileorDirName.ToString())

{

int firstcluster = Program.currentDirectory.directoryTable[index1].fileFirstCluster;

int f\_size = Program.currentDirectory.directoryTable[index1].fileSize;

Directory\_Entry entry = new Directory\_Entry(num1.ToString(), 0x0, firstcluster, f\_size);

Directory dir = new Directory(num2.ToString(), 0x10, firstcluster, f\_size, Program.currentDirectory.parent);

dir.directoryTable.Add(entry);

}

else Console.WriteLine("not fff");

}

}

}

}

class Program

{

public static Directory currentDirectory;

public static string currentPath;

static void Main(string[] args)

{

Console.WriteLine("welcome to Project\_OS\_virsual\_disk\_Shell");

Console.WriteLine("developed by Ashraf & Beshoy.");

Virsual\_Disk.initalize("File.txt");

currentPath = new string(currentDirectory.fileorDirName);

while (true)

{

Console.Write(currentPath.Trim());

string inputuser = Console.ReadLine();

if (!inputuser.Contains(" "))

{

if (inputuser.ToLower() == "help")

{

cmd.help();

}

else if (inputuser.ToLower() == "quit")

{

cmd.quit();

}

else if (inputuser.ToLower() == "cls")

{

cmd.cls();

}

else if (inputuser.ToLower() == "md")

{

Console.WriteLine("The syntax of the command is incorrect.");

}

else if (inputuser.ToLower() == "rd")

{

Console.WriteLine("The syntax of the command is incorrect.");

}

else if (inputuser.ToLower() == "dir")

{

cmd.dir();

}

else if (inputuser.ToLower() == "import")

{

Console.WriteLine("The syntax of the command is incorrect.");

}

else if (inputuser.ToLower() == "type")

{

Console.WriteLine("The syntax of the command is incorrect.");

}

else if (inputuser.ToLower() == "export")

{

Console.WriteLine("The syntax of the command is incorrect.");

}

else if (inputuser.ToLower() == "del")

{

Console.WriteLine("The syntax of the command is incorrect.");

}

else if (inputuser.ToLower() == "copy")

{

Console.WriteLine("The syntax of the command is incorrect.");

}

}

else if (inputuser.Contains(" "))

{

string[] arrInput = inputuser.Split();

if (arrInput[0] == "md")

{

cmd.md(arrInput[1]);

}

else if (arrInput[0] == "rd")

{

cmd.rd(arrInput[1]);

}

else if (arrInput[0] == "cd")

{

cmd.cd(arrInput[1]);

}

else if (arrInput[0] == "import")

{

cmd.import(arrInput[1]);

}

else if (arrInput[0] == "type")

{

cmd.type(arrInput[1]);

}

else if (arrInput[0] == "export")

{

cmd.export(arrInput[1], arrInput[1]);

}

else if (arrInput[0] == "del")

{

cmd.del(arrInput[1]);

}

else if (arrInput[0] == "copy")

{

cmd.copy(arrInput[1], arrInput[1]);

}

}

}

}

}

}