Virtual File System

Overview Statement:

Assume that you have a virtual file system with a root directory called "root" all the files and folders will be stored under it. The disk size consists of **N** blocks and each block size is **1 KB**.

The aim of this assignment is to simulate the allocation and de-allocation of files and folders using different allocation techniques. Implement the two allocation techniques listed below (Refer to the file system chapter in your text book):

1- Contiguous Allocation (Using Best Fit allocation)

2- Indexed Allocation

After running the application the user will interact with your virtual file system through a series of commands, these commands are illustrated in the table below:

System Commands:

Command	**Summary**
`CreateFile root/file.txt 100`	This command used to create file named “file.txt” with 100 KB size under the path “root” Pre-requests: 1- The path is already exist 2- No file with the same name is already created under this path 3- Enough space exists
`CreateFolder root/folder1`	This command is used to create a new folder named “folder1” under the path “root” Pre-requests: 1- The path is already exist 2- No folder with the same name

	is already created under this path.
`DeleteFile root/folder1/file.txt`	This command used to delete file named “file.txt” form the path "root/folder1". Any blocks allocated by this file should be de-allocated. Pre-requests: 1- The file is already exist under the path specified
`DeleteFolder root/folder1`	This command used to delete folder named “folder1” form the path "root". All files and subdirectories of this folder will also be deleted. Pre-requests: 1- The folder is already exist under the path specified
`DisplayDiskStatus`	This command used to display the status of your Driver the status should contain the following information: 1- Empty space 2- Allocated space 3- Empty Blocks in the Disk 4- Allocated Blocks in the Disk
`DisplayDiskStructure`	This command will display the files and folders in your system file in a tree structure

Saving and loading Virtual File System

In this program we are not creating actual files and folder, we will just simulate having a series of blocks and these blocks will be allocated to files when created and will be de-allocated when these files are deleted.

My virtual file system information like (the files information, the folders information, the allocated blocks and so on) should be saved on a file on your hard disk to be able to load it the next time you run the application.

So when the application starts, the system should automatically load the disk structure form the **V**irtual **F**ile **S**ystem file say named "c:\DiskStructure.vfs". Then the user will start to enter commands which will be executed on the loaded data in memory, and before the application terminates, the data in memory will be written into the file again.

The following information will be stored in the file:

1- Files and Folders Directory Structure.

2- The Empty blocks of the virtual DISK

3- The allocated blocks in your virtual DISK and which files/folders are take these places.

More details about these 3 points are in the following sections.

1- Files and Folders Directory Structure

The **V**irtual **F**ile **S**ystem file will contain the structure of the files and folders which should be loaded in a tree structure in memory. This is required to be able to print the tree structure when the user requests the command "`DisplayDiskStructure`".

Example directory structure:

- **<root>**
  - **File0.txt**
  - **<folder1>**
    - **File1.txt**
    - **<folder2>**

2- Empty Blocks (Free Space Manager)

A free space manager component which will be used by the allocation technique to know which blocks are free, to search for free contiguous blocks, to de-allocate blocks when a file is deleted and to allocate blocks when a file is created and so on.

As mentioned before in the previous section, Our virtual file system file will contain the empty blocks, this can be done using a series of zeros of ones, where a zero at index 2 in the series means that block 2 is free, and a one at index 2 means that block 2 is allocated.

Example, if you have 10 blocks where the first 5 are allocated and the last 5 are free, the series that should be saved in the file system file should be: `1111100000`

3- The allocated blocks for files

The needed information to represent the allocated blocks by files depends on the allocation technique, for example:

In the **contiguous allocation** we will store the start block number and the number of blocks a file is allocating like:

```
root/folder1/file.txt   0  5

root/folder2/test.txt   5  3
```

This means that the file root/folder1/file.txt is allocating 5 blocks starting at block number 0. And the file root/folder2/test.txt is allocating 3 blocks starting at block number 5.

In the **indexed allocation**, a block called the index block is allocated for each file to contain that file blocks numbers. So we will save the index block number for each file and the content of the index block like:

```
root/folder1/file.txt   7

7    1 2 3 4
```

This means that the index block of the file root/folder1/file.txt is 7. And the content of the block 7 is 1 2 3 4, which means that this file is allocating the blocks 1 2 3 and 4.