Gebze Technical University

Department of ComputerEngineering

CSE 312 /CSE 504

Operating Systems

Spring 2023

HW3

Due Date: June 17th

(No latesubmissions)

2023

File Systems

In this project, you will design and implement a simplified FAT like file system in C or C++.

Part 1

Design a file system that uses File Allocation Table (FAT12) structure to keep your files. Your file system will use a structure like Fig 4.30 and Fig 4.31 of your textbook (all possible FAT12 blocks are supported). Your file attributes will include size, last modification date and time, and name of the file. Write a design report that specifies the following

- Define your directory table and directory entries;
- Define how and where you keep the free blocks;
- Define your superblock that contains crucial information about the file system such as the block size, root directory position, block positions, etc.

Your report should include the function names of your source code that handles the file system operations listed in the table of Part 3.

Part 2

Write a C/C++ program that creates an empty file system as a (16 MB max) file. This file will include all the information about your file system including the super block, data blocks, free blocks, directories, data, etc. The sample run of the program will be like

```
makeFileSystem 4 mySystem.dat
```

where 4 is the block size of the file system in KB. **`mySystem.dat`** is the file that contains all the file system. When you work on the file system, this file contains all the information for the file system. Note that the size of **`mySystem.dat`** will be exactly the maximum according to table Fig 4.31 MB whether it contains any information or not.

Part 3

You will write a program that performs file system operation on the file system. The program will work like following

```
fileSystemOper fileSystem.data operation parameters
```

where **`fileSystemOper`** is your program, **`fileSystem.data`** is the file system data file that you have created in Part 2. You will keep modifying the same **`fileSystem.data`** file for all your operations. Allowable operations and parameters for these operations are given below in the following table.

I

Operation	Parameters	Explanation	Example
dir	Path	Lists the contents of the directory shown by path on the screen.	**fileSystemOper fileSystem.data dir "\"** lists the contents of the root directory. The output will be similar to dir command of DOS
mkdir rmdir	Path and dir name	Makes or removes a directory	**fileSystemOper fileSystem.data mkdir "\ysa\fname"** makes a new directory under the directory "ysa" if possible. These two works exactly like mkdir and rmdir commands of DOS shell
dumpe2fs	None	Gives information about the file system.	**fileSystemOper fileSystem.data dumpe2fs** works like simplified and modified Linux dumpe2fs command. It will list block count, free blocks, number of files and directories, and block size. Different from regular dumpe2fs, this command lists all the occupied blocks and the file names for each of them.
write	Path and file name	Creates and writes data to the file	**fileSystemOper fileSystem.data write "\ysa\file" linuxFile** Creates a file named **file** under **"\usr\ysa"** in your file system, then copies the contents of the Linux file into the new file.
read	Path and file name	Reads data from the file	**fileSystemOper fileSystem.data read "\ysa\file" linuxFile** Reads the file named file under **"/usr/ysa"** in your file system, then writes this data to the Linux file. This again works very similar to Linux copy command.
del	Path and file name	Deletes file from the path	**fileSystemOper fileSystem.data del "\ysa\file"** Deletes the file named **file** under **"\ysa\file"** in your file system. This again works very similar to Linux del command.

Here is a sequence file system operation commands that you can use to test your file system. Suppose you have a file named linuxFile.data in your Linux current directory.

```
makeFileSystem 4 mySystem.data
fileSystemOper fileSystem.data mkdir "\usr"
fileSystemOper fileSystem.data mkdir "\usr\ysa"
fileSystemOper fileSystem.data mkdir "\bin\ysa"                    ; Should print error!
fileSystemOper fileSystem.data write "\usr\ysa\file1" linuxFile.data
fileSystemOper fileSystem.data write "\usr\file2" linuxFile.data
fileSystemOper fileSystem.data write "\file3" linuxFile.data
fileSystemOper fileSystem.data dir "\"                    ; Should list 1 dir, 1 file
fileSystemOper fileSystem.data del "\usr\ysa\file1"
fileSystemOper fileSystem.data dumpe2fs
fileSystemOper fileSystem.data read "\usr\file2" linuxFile2.data
cmp linuxFile2.data linuxFile.data          ; Should not print any difference
```

Part 4

In this part, you will integrate and run your file system (developed in Part 1-3) on the operating system that you have already worked in Homework 1.

You may use the whole source code of him. He talks on the file systems in the video series in two parts such as partition table and FAT32.

Watch: https://youtu.be/IGpUdIX-Z5A

Watch also: https://youtu.be/tEYgVwN1nRk

The main steps of your program will be as follows

1. Generate your file system on OS running by Part 1 and Part 2
2. Implement and run any system calls for file and directory operations
    a. Watch : https://youtu.be/g5Ej3RxIZJI
3. Test Part 3 on the OS.