Gebze Technical University
Department of Computer Engineering
CSE 312 /CSE 504
Operating Systems
Spring 2024

HW4
Due Date: June 6th
(No late submissions)
2024
File Systems

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

Part 1 (10 pts)

Design a file system that uses File Allocation Table (FAT12) structure to keep your files. Your file system will use a similar structure like Fig 4.30 and Fig 4.31 of your textbook (only 0.5 KB and 1 KB block sizes supported). Your file attributes will include

- File name (any length is possible)
- Size
- Owner permissions (R and W only, no X), we will not use group and other permissions.
    - A file can be read if it has R permission.
    - A file can be written if it has W permission.
- Last modification date and time.
- File creation date and time.
- Password if the file is protected.

Write a design report that specifies at least the following

- Define your directory table and directory entries.
- Define how and where you keep the free blocks.
- Define how you solve arbitrary length of file names.
- Define how you handle the permissions.
- Define how you handle password protection.

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 (25 pts)

Write a C/C++ program that creates an empty file system as a (4 MB max) Linux 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 1 mySystem.dat
```

where 1 is the block size of the file system in KB. **`mySystem.dat`** is the Linux 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.

I

Part 3 (65 pts)

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.

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 like ls command of Linux. It should list permissions, modification and creation dates, whether the file is password protected.
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. **Note that Linux file permissions should also be copied.**
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. **Note that Linux file should have the same owner permissions.**
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.
chmod	Path, file name and permissions	Changes the owner permissions	`fileSystemOper fileSystem.data chmod "\ysa\file" +rw` Add read and write permission to the file "\ysa\file". This again works very similar to Linux chmod command.

addpw	Path, file name and password	Adds a protection password to the file	**fileSystemOper fileSystem.data addpw "\ysa\file" test1234** Add password to the file "\ysa\file". After this point, any operation on this file has to be done with the password. For example, **fileSystemOper fileSystem.data read "\ysa\file" linuxFile test1234**

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
fileSystemOper fileSystem.data chmod "\usr\file2" -rw
fileSystemOper fileSystem.data read "\usr\file2" linuxFile2.data ;Should produce an error.
fileSystemOper fileSystem.data chmod "\usr\file2" +rw
fileSystemOper fileSystem.data addpw "\ysa\file2" test1234
fileSystemOper fileSystem.data read "\usr\file2" linuxFile2.data ;Should produce an error.
fileSystemOper fileSystem.data read "\usr\file2" linuxFile2.data test1234 ;Should be OK
```