Gebze Technical University
Department of Computer Engineering
CSE 312 /CSE 504
Operating Systems
Spring 2020

Midterm Exam Project
Due Date: May31st
2020
File Systems

In this project, you will design and implement a simplifies UNIX like file system in C or C++. This project does not use MIPS or SPIM, so it is completely independent of your other homework assignments. This will count as midterm exam for this course.

Part 1

Design a file system that uses i-nodes blocks and data blocks to keep your files. Your file system will use, a i-node structure like Fig 4.33 of your textbook (single link, double link and triple link i-nodes). Your directory, i-node block, data block structure will be similar to Fig 4.34 of the textbook. Your file attributes will include size, last modification date and time, and name of the file. No permissions or owner attributes will be kept. Write a design report that specifies the following

- Define your directory structure and directory entries;
- Define how you keep the free blocks and free i-nodes;
- Define your i-node structure
- Define your superblock that contains crucial information about the file system such as the block size, i-node positions, 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 1 MB Linux file. This file will include all the information about your file system including the i-nodes, data blocks, free blocks and i-nodes, directories, data, etc. The sample run of the program will be like

```
makeFileSystem 4 400 mySystem.dat
```

where 4 is the block size of the file system in KB for both data blocks and i-node blocks, and 400 is the number of free i-nodes for an empty file system. **`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 1 MB all the time 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
list	Path	Lists the contents of the directory shown by path on the screen.	**fileSystemOper fileSystem.data list "/"** lists the contents of the root directory. The output will be like ls –l
mkdir rmdir	Path and dir name	Makes or removes a directory	**fileSystemOper fileSystem.data mkdir "/usr/ysa"** makes a new directory under the directory "ysa" if possible. These two works exactly like mkdir and rmdir commands of Linux 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, i-node count, free block and i-nodes, number of files and directories, and block size. Different from regular dumpe2fs, this command lists all the occupied i-nodes, 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 "/usr/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. This works very similar to Linux copy command.
read	Path and file name	Reads data from the file	**fileSystemOper fileSystem.data read "/usr/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 "/usr/ysa/file"** Deletes the file named **file** under **"/usr/ysa"** in your file system. This again works very similar to Linux del command.
Operations below are bonus (20 points)			
ln	Source and destination path and file names	Hard linking between 2 files	**fileSystemOper fileSystem.data ln "/usr/ysa/file1" "/usr/ysa/file2"** Allows more than one filename to refer to the same file Linux ln command
lnsym	Source and destination path and file names	Symbolic linking between 2 files	**fileSystemOper fileSystem.data lnsym "/usr/ysa/file1" "/usr/ysa/file2"** Linux ln-s command

fsck	none	Simplified File system check	**fileSystemOper fileSystem.data fsck** Just print the two table of Fig 4.27 of your textbook for both i-nodes and blocks

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 4 400 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 list "/"                            ; 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 ln "/usr/file2" "/usr/linkedfile2"
fileSystemOper fileSystem.data list "/usr"
fileSystemOper fileSystem.data write "/usr/linkedfile2" linuxFile.data
fileSystemOper fileSystem.data dumpe2fs
fileSystemOper fileSystem.data lnsym "/usr/file2" "/usr/symlinkedfile2"
fileSystemOper fileSystem.data list "/usr"
fileSystemOper fileSystem.data del "/usr/file2"
fileSystemOper fileSystem.data list "/usr" ; Should see likedfile2 is there, symlinkedFile2 is there but..
fileSystemOper fileSystem.data write "/usr/symlinkedfile2" linuxFile.data ;  Should print error!
fileSystemOper fileSystem.data dumpe2fs
```

Notes

1. Always be careful about the errors, such as bad block sizes, bad file names, non-existent files or directories, etc.
2. Run experiments that uses up all of your i-nodes and data blocks.
3. Try to get fragmentation and show your fragmented file system using the **`dumpe2fs`** command.
4. Do not use any code from any other source even a single line!

What to submit

Follow all instructions below, otherwise you will get penalty on your grade for each missing instruction.

- ✓ **Name all files as they specified below exactly.**
- ✓ **Zip all files (6 files below) into only one file and name it like below**

  CSE312_Midterm_StudentNumber_Surname_Name.zip

  - Do not use other than .zip extensions
- ✓ **If you want to specify problems such as run time errors, compilation error, or etc., please use only one ReadMe file to write those into.**

  - Do not put them into your Part_1.pdf report

1. Part_1.pdf: This will contain your report as the output of Part 1.
2. Part_2_Program.zip : This will contain your program as the output of Part 2.
   a. All the files that are required to run your program of Part 2 must in this zip file.
3. Part_3_Program.zip : This will contain your program as the output of Part 3.
   a. All the files that are required to run your program of Part 3 must in this zip file.
4. fileSystem.data : This file is your file system as the output of Part2.
5. commands.txt : This is the file containing the file system commands that are developed and tested.
6. readMe.txt: Described above.

General Homework Guidelines

1. No cheating, No copying, No peaking to other people homework
2. Follow the instructions very carefully.
3. Send required files only. Do not share your whole file system with us.
4. If you fail to implement one of the requirements, leave it be. Do not send an empty file
5. Respect the file names! Our HW grading is case-sensitive.
6. Failing to comply any of the warnings above will result in getting a **0** for your current homework.

Homework Instructions

1. Download and Install Vmware Player from Official site.
2. Download and install our virtual machine from
https://drive.google.com/open?id=1YppX3lNkyTsHV_lvA4w9TomNCUkpLeEg