Project: Simulated File System

CMSC 216.001

Due Date: March 10, 2023, 11:59pm

1 Overview

Each student will build a simulated filesystem with a hierarchical set of directories that can contain multiple files and subdirectories.

1.1 Objectives

- Build and traverse arbitrarily complex data-structures in memory.
- Understanding dynamic memory allocation.
- Understanding the use of pointers.
- Learning about file systems.

2 Detailed Description

2.1 Introduction

This project is an implementation of a simulated hierarchical file system in C using dynamic memory allocation and pointers. A hierarchical file system is a system that organizes files and directories in a tree-like structure, with a root directory at the top and subdirectories branching out from there.

The project includes the following functionalities:

- Create a new file
- Delete an existing file
- Rename a file
- Create a new directory
- Delete an existing directory
- Change the current working directory

- Print the contents of a directory

To implement the file system, you will need to use dynamic memory allocation to create new directories and files, as well as to store the names of directories and files. You will also need to use pointers to traverse the linked list of directories and files, and to perform operations such as creating, deleting, and renaming files and directories.

2.2 What to do

Create C program called "fs" that implements a simulated filesystem. Your program should operate as an interactive "shell" that accepts filesystem commands and performs the associated functions in the simulated filesystem.

2.3 Quick Details

- possibly infinite files and directories (only limited by available memory)
- start with an initial state that includes just a root directory
- files/directories have the following properties
  - name - provided at creation (max: 256 chars)
  - creation time - determined from system time
  - size - random number for files, determined by the number of subdirectories for directories

Since there is no limit to the number of files or directories, students will have to allocate memory for files and directories dynamically. The selection of the appropriate dynamic memory data structures is left as an exercise for the student.

2.4 Filesystem Commands/Operations

The simulated filesystem shell will support seven possible commands:

`touch [filename]` - Create a file in the current working directory with the provided name. To create a file, you will need to allocate memory for a new file object and set its attributes (name, a random size, and creation time based on the system time). You will also need to add the new file to the appropriate place in your filesystem hierarchy based on the current working directory. An attempt to `touch` an existing file or directory, will merely update its creation time.

`mkdir [filename]` - Create a subdirectory in the current working directory with the provided name. To create a directory, you will need to allocate memory for a new directory object and set its attributes (name, initial size, and creation time based on the system time). You will also need to add the new directory to the appropriate place in your filesystem hierarchy based on the current working directory. An attempt to `mkdir` a directory or file that already exists should return an error message.

`rename [old-name] [new-name]` - Rename a file or directory in the current working directory. To rename a file or directory, you will need to traverse the the data of the current directory, find the file or directory with the given name, and update its `name` attribute. An attempt to `rename` a file that does not exist should return an error message.

`rm [filename]` - Delete a file or directory in the current working directory. To delete a file or directory, you will need to traverse the data of the current directory, find the file or directory with the given name, and remove it from the data structure. You will also need to free the memory allocated for the file or directory object. An attempt to `rm` a file that does not exist should return an error message.

`ls` - Print the contents of the current directory, including the name, size, and creation time of each file and subdirectory. To print the contents of the current directory, you will need to traverse the data for the current directory and print the `name`, `size`, and `creation time` of each object.

`cd [directory]` - Change the current directory to a specified subdirectory. To change the current directory, you will need to traverse the data for the current directory, find the subdirectory with the specified name, and set it as the new current directory. You will also need to provide a way to return to the parent directory (i.e., the directory that contains the current directory), this is usually done with `cd ..` Attempts to `cd` to a directory that does not exist should return an error.

`exit` - Exit the program Free all data structures and allocated memory, and exit the program. No filesystem information needs to be saved.

Other unknown commands, incorrect inputs, or otherwise invalid input should generate an appropriate error message. Your interactive shell should be relatively robust to common mistakes and errors.

3 Submission

Project should be submitted by **March 10, 2023, 11:59pm**. Follow these instructions to turn in your project.

You should submit the following files:

- fs.c
- fs.h *(optional)*
- Makefile *(optional)*
- *any other source files your project needs*

The following submission directions use the command-line `submit` program on the class server that we will use for all projects this semester:

- Log into the VDI: `http://desktop.montgomerycollege.edu`
- If necessary, transfer your source code onto the VDI
- Use *Bitvise SSH* client to log into `tpaclinux`
- If necessary, use the *Bitvise SFTP transfer* to upload your source code to the server
- Finally, use the command-line `submit` program to turn in your code

An example command line submission of files: `fs.c`, `fs.h`, `util.c`, `util.h` and `Makefile` for project 3, would look like:

```
submit proj3 fs.c fs.h util.c util.h Makefile
```

Late assignments will not be given credit.

4 Grading

While this rubric is subject to change based on class performance, the current grading rubric for this assignment is as follows:

component	value
Basic Dynamic Structures	20
Files, Directories, Subdirectories	30
Other Functions	30
Efficiency	10
Code Quality	10