Advanced OS
Assignment #1 - File Explorer
Deadline - 9/10/2020 (11:55pm)

Objective:

Build a fully functional File Explorer Application, albeit with a restricted feature set.

Prerequisites:

1. Basic usage and architectural know-how of file explorer features
2. Preliminaries such as C/C++ code compilation, execution & debugging

Requirements:

Your File Explorer should work in two modes -

1. Normal mode (default mode) - used to explore the current directory and navigate the filesystem
2. Command mode - used to enter shell commands

The root of your application should be the directory where the application was started.

The application should display data starting from the top-left corner of the terminal window, line-by-line. You should be able to handle text rendering if the terminal window is resized. The last line of the display screen is to be used as a status bar.

Normal mode:

Normal mode is the default mode of your application. It should have the following functionalities -

1. Display a list of directories and files in the current folder
    a. Every file in the directory should be displayed on a new line with the following attributes for each file -
        i. File Name
        ii. File Size
        iii. Ownership (user and group) and Permissions
        iv. Last modified

        All of this should be displayed in human readable format
    b. The file explorer should show entries "." and ".." for current and parent directory respectively
    c. The file explorer should handle scrolling in the case of vertical overflow using keys **k** & **l**
    d. User should be able to navigate up and down in the file list using the corresponding up and down arrow keys

2. Open directories and files
   When enter key is pressed -
   a. Directory - Clear the screen and navigate into the directory and show the directory contents as specified in point 1
   b. File - Open the file in vi editor

3. Traversal
   a. Go back - Left arrow key should take the user to the previously visited directory
   b. Go forward - Right arrow key should take the user to the next directory
   c. Up one level - Backspace key should take the user up one level
   d. Home - **h** key should take the user to the home folder (the folder where the application was started)

Command Mode:

The application should enter the Command button whenever ":" (colon) key is pressed. In the command mode, the user should be able to enter different commands. All commands appear in the status bar at the bottom.

1. Copy - 'copy <source_file(s)> <destination_directory>'
   Move - 'move <source_file(s)> <destination_directory>'
   Rename - 'rename <old_filename> <new_filename>'
   a. Eg -
   ```
   copy foo.txt bar.txt baz.mp4 ~/foobar
   move foo.txt bar.txt baz.mp4 ~/foobar
   rename foo.txt bar.txt
   ```
   b. Assume that the destination directory exists and you have write permissions.
   c. Copying/Moving directories should also be implemented
   d. The file ownership and permissions should remain intact

2. Create File - 'create_file <file_name> <destination_path>'
   Create Directory - 'create_dir <dir_name> <destination_path>'
   a. Eg -
   ```
   create_file foo.txt ~/foobar
   create_file foo.txt .
   create_dir foo ~/foobar
   ```

3. Delete File - 'delete_file <file_path>'
   Delete Directory - 'delete_dir <dir_path>'
   The file/dir path should be relative to the root from where the application is run

4. Goto - 'goto <location>'
   a. Eg - `goto <directory_path>`
   b. Absolute path wrt application root will be given

5. Search - ‘search <file_name>’ or ‘search <directory_name>’
    a. Search for a given file or folder under the current directory recursively
    b. Output should be True or False depending on whether the file or folder exists

6. On pressing **ESC** key, the application should go back to Normal Mode

Submission format:

The main file of your application should be named main.c or main.cpp. We will run your code as follows -

```
$ gcc main.c / g++ main.cpp
$ ./a.out
```

Put all your C/C++ files in a folder named <roll_num> and create a zip folder by the name <roll_num>.zip and submit this zip file on moodle. For example, if your roll number is 20201000, then you should submit a folder `20201000.zip` which on extracting gives a folder `20201000/` containing all the necessary files.

Guidelines:

1. Languages allowed: C/C++. Use of STL is allowed.
2. Use of system() library function for command mode functions is not allowed.
3. Use of system commands like ls, cp, mv, mkdir etc are not allowed. You have to implement your own versions of these commands.
4. Use of ncurses.h is strictly prohibited.
5. Modularize your code. Indent and comment your code properly.
6. Handle error cases wherever required.
7. Make sure that you test your code well. Compilation/runtime errors will be penalised.
8. Do not upload any executables.
9. **ZERO** tolerance towards any kind of plagiarism. **DO NOT copy or share code/code-snippets** (even a few lines of copied code would be detected and punished) - **both the parties will get zero**.