CS132 Coursework 2 Report

Donatas Jacinavicius u2001524

February 2021

Contents

1 Introduction

The aim of this report is to analyse the requirements of question 2 of CS132 Coursework 2, discuss system design in high level terms, implement the command-line editor, record the testing done on the implementation, evaluate the implementation, conclude the report, and reference the resources I have used in this question.

2 Requirements Analysis

The requirements for the command-line editor are as follows:

- File Operations
  - Create File - Create a new file with a specified name

  - Copy File - Create a new file with a specified name and identical contents to an existing file
  - Delete File - Delete an existing file with a specified name
  - Show File - Display the contents of an existing file with a specified name

- Line Operations
  - Append Line - Create a new line of content at the end of a specified file
  - Delete Line - Delete a line of content at a particular line number in a specified file
  - Insert Line - Create a new line of content at a particular line number in a specified file
  - Show Line - Display the contents of a file at a particular line number in a specified file

- General Operations
  - Show Change Log - Display the sequence of operations performed on all files created by the program, including the number of lines following each operation
  - Show Number of Lines - Show the number of lines in a specified file

The editor must also be capable of two extra operations, I have decided the two extra operations will be:

- List Files - List all files in the current directory with their permissions, and also give the option to change permissions of any files in the directory
- Rename File - Change the name of a file with a specified name

List Files is a reasonable function to implement because with the original specification, a user would require a separate window to the command-line editor to be able to see files in the directory or change their permissions. This function will make the editor easier to use.
Rename File is a reasonable function to implement because with the original specification, to rename a file a user would need to copy a file using the Copy File function, then delete the original file with the Delete File function. This function will allow the user to rename a file with a lower amount of operations.

Additionally, I have decided to add a Menu function which will allow the user to choose between all other operations, and will do so until the option to quit is chosen.

3 System Design

3.1 Menu

The Menu function will begin by printing the choice of every operation to the screen, numbered from 1 to 13. These choices will be:

1. Create a file
2. Copy a file
3. Delete a file
4. Show a file
5. Append a line to end of file
6. Delete a line from file
7. Insert a line to file
8. Show a line from file
9. Rename a file

10. List all files in current directory and/or change permissions
11. Show change log
12. Show number of lines in a file
13. Quit

Next, the function will ask for a user input, that being the number from the corresponding operation. The correct operation will be performed, and the user will be prompted for a new selection. The user will be continuously given this prompt until the Quit choice is chosen.

3.2 Create File

The function will begin by asking the user for the name of a file, and giving them the prompt for an input. A file with that name will then be created. The change log is written to, the file is closed, and the function returns.

3.3 Copy File

The function will begin by asking the user for the name of a file to be copied, and the name of the new file. A prompt will be given for each of the inputs separately. If the file to be copied exists, it is opened, else, the function returns. The new file is created and opened. Characters are written sequentially from the original file to the new file until EOF is reached, and both files are closed once this has finished. The change log is written to, the files are closed and the function returns.

3.4 Delete File

The function will begin by asking the user for the name of a file to be deleted, and giving them the prompt for an input. The file with that name will be deleted if it exists in the directory. The change log is written to, and the function returns.

3.5 Show File

The function will begin by asking the user for the name of a file to be displayed, and giving them the prompt for an input. The file will be opened if it exists, else, the function returns. Characters will be printed to the terminal screen from the file until EOF is reached. The file is closed and the function returns.

3.6 Rename File

The function will begin by asking the user for the name of the file to rename, and the new name of this file. A prompt will be given for each of the inputs separately. If the file exists, it will be renamed, else, the function returns. The change log is written to, and the function returns.

3.7 List Files

The function will begin by listing all files in the current directory, along with their permissions. The user will then be asked if any permissions are to be changed. A prompt will be given for input of Y/N. If N, the function returns. Else, the user is asked for the name of the file to change permissions for, and the new permissions to be given. A prompt will be given for each of these inputs separately. If the file exists, the permissions of the file will be changed, else, the function returns. The change log is written to, and the function returns.

3.8 Append Line

The function will begin by asking the user for the name of the file to append the line to, and the content to append. A prompt will be given for each of the inputs separately. If the file exists, it is opened, else, the function returns. The content is appended to the file. The change log is written to, the file is closed, and the function returns.

3.9 Delete Line

The function will begin by asking the user for the name of the file to delete the line from, and the line number to delete. A prompt will be given for each of the inputs separately. If the file exists, it is opened, else, the function returns. A loop goes through each line of the file and writes the line to a temporary file until the specified line for deletion is found. The deletion line is not written to the temporary file. After each line apart from the deletion line has been written, the original file is deleted and the temporary file is renamed to the original file. The change log is written to, the file is closed, and the function returns.

3.10 Insert Line

The function will begin by asking the user for the name of the file to insert content into, the line number to insert the content, and the content to be inserted. A prompt will be given for each of the inputs separately. If the file exists, it is opened, else, the function returns. A loop goes through each line of the file and writes the line to a temporary file until the line to insert content into is reached. The content is written to the file on this line and the loop continues writing the remaining lines into the file. Once this loop has finished, the original file is deleted and the temporary file renamed to the original. The change log is written to, the file is closed, and the function returns.

3.11 Show Line

The function will begin by asking the user for the name of the file to show the line from, and the line number to show. A prompt will be given for each of the inputs separately. If the file exists, it is opened, else, the function returns. A loop goes through each line until the line to be shown is found, this line is printed to the terminal. The file is closed and the function returns.

3.12 Show Number Of Lines

The function will begin by asking the user for the name of the file to count lines from, and a prompt for the input will be given. If the file exists, it is opened, else, the function returns. A loop goes through the file reading each character, upon finding a newline character the amount of lines increases by 1. The amount of lines is printed to the terminal. The file is closed and the function returns.

3.13 Show Change Log

The function begins by opening the change log text file. A loop goes through the file printing characters from the file to the terminal until EOF is reached. The file is closed and the function returns.

3.14 Append Change Log

The function takes 3 parameters: operation, file1, file2. The change log text file is opened, and a new line is appended to the end of the file containing the operation performed and the file(s) involved. The file is closed and the function returns.

4 Implementation

The implementation of the editor can be found in the attached C file with the submission. Code annotations explain the implementation of each function on a line-by-line basis.

My implementation does not use scanf, instead I choose to only use getline. Using only getline means none of my strings used have a cap on the string length, I use getline(&name, &length, stdin) (that's my general usage of it). This ensures a user never types in a file name or contents of a line that are too long for the program to handle.
The disadvantage of getline is that it always assigns a string to the first argument, whereas scanf can immediately assign to an integer. This means I need an extra line with strtol if I want to take numbers as user input.
Additionally, strtok needs to be used to remove the newline character from the end of the string obtained by using getline

I used the following sources while implementing the design:

- Documentation for C library <string.h> used to find strtok and strcat[4]
- Article on strings in C, used to find getline and other options to getline that were considered, such as fgets [1]
- Documentation for C library <stdio.h> used to find file operations such as fopen and many others, helped with file operations [2]
- Documentation for C library <stdlib.h> used to find strtol and system [3]

5 Testing

Test	Expected Outcome	Outcome	Result
Program should compile with no errors	Terminal does not display any warnings or errors	Terminal does not display any warnings or errors	Pass
Menu function should display options and take user input	All options should be printed to terminal upon running the program. User input should be taken	All options printed to terminal, user input taken with no error	Pass
Menu function should correctly call the corresponding function based on user input	Selecting any option 1-13 will run the correct function or quit in the case of 13	Selecting any option does run the correct function	Pass
CreateFile function should take user input and create file with that name	Entering any file name into the input prompt should result in a new file being created in the current directory with that name	Entering a file name does result in a new file being created in the current directory with that name	Pass
CreateFile function should return to menu if invalid input given	Entering no input will cause the program to return to menu	Entering no input causes program to return to menu	Pass

CopyFile function should take two user inputs and copy the first input with the name of the second	Entering two file names into the input prompts should result in a new file being created with the name of the second input but contents of file of first input	Entering two names does result in a new file with the contents of the first file being created	Pass
CopyFile function should return to menu if invalid input(s) is/are given	Entering no input in either of the two prompts returns the program to the menu	Entering no input does return the program to the menu	Pass
DeleteFile function should take user input and delete file with that name	Entering any file name that exists in directory results in that file being deleted	Entering any file name in directory does result in deletion	Pass
DeleteFile function should return to menu if invalid input is given	Entering no input in the prompt returns program to the menu	Entering no input does return program to the menu	Pass
ShowFile function should take user input and display file in terminal with that name	Entering any file name in current directory should result in contents of file being displayed in terminal	Entering a file name in directory does result in contents of file being displayed in terminal	Pass
ShowFile function should return to menu if invalid input is given	Entering no input in the prompt returns program to the menu	Entering no input does return program to the menu	Pass
AppendLine function should take user input of file name and line contents and add the contents of the line to the end of the given file	Entering inputs for file name and line contents should result in that line being appended to the file	Entering inputs for file name and line contents does result in line being appended	Pass
AppendLine function should return to menu if invalid input(s) is/are given	Entering no input in either of the two prompts returns the program to the menu	Entering no input does return program to the menu	Pass

DeleteLine should take user input of file name and line number to delete chosen line in chosen file	Entering inputs for file name and line number should result in that line being deleted from file	Entering inputs for file name and line number does result in line being deleted	Pass
DeleteLine function should return to menu if invalid input(s) is/are given	Entering no input in either of the two prompts should result in the program returning to the menu	Entering no input does result in the program returning to the menu	Pass
InsertLine function should take user input of file name, line number, and line contents and add the line contents to the line number in specified file	Entering inputs for file name, line number, and line contents should result in a line being added in the specified line in the text file	Entering inputs does result in line being added in correct place in file	Pass
InsertLine function should return to menu if invalid input(s) is/are given	Entering no input in any of the three prompts should result in program returning to the menu	Entering no input does result in the program returning to the menu	Pass
ShowLine function should take user input of file name and line number and print the line to terminal	Entering input for file name and line number should result in the specified line from the file being printed to the terminal	Entering inputs does result in correct line being displayed	Pass
ShowLine function should return to menu if invalid input(s) is/are given	Entering no input in either of the two prompts should result in program returning to menu	Entering no input does result in the program returning to the menu	Pass
RenameFile function should take user input of old file name and new file name and rename the file	Entering input for both old and new file name should result in the file being renamed to the new file name	Entering inputs does result in file being renamed correctly	Pass
RenameFile function should return to menu if invalid input(s) is /are given	Entering no input in either of the two prompts should result in program returning to menu	Entering no input does result in the program returning to the menu	Pas

ListFiles function should display all files in current directory, display their permissions, and ask the user if they wish to change any permissions	Selecting 10 in menu should display all files and their permissions, and terminal should ask user if they wish to change any permissions	Selecting 10 does display files and their permissions, and give user option to change permissions	Pass
ListFiles function should take user input Y/N, react accordingly	Entering 10 in the menu then Y should result in a new prompt to enter file name and new permissions, Entering 10 in the menu then N should result in a return to the menu	Entering 10 in menu then Y does result in prompt to enter file name and new permissions, and entering 10 in menu then N does result in return to the menu	Pass
ListFiles function after taking input Y should take user input of file name and new permissions, then change file permissions accordingly	Entering 10 in menu then Y then giving inputs to file name and new permissions should result in file having changed permissions as described by new permissions	Entering 10 in menu then Y then giving inputs does result in file changing permissions	Pass
ListFiles function should return to menu if at any input an invalid input is given	Entering no input for any prompt in the function should result in a return to the menu	Entering no input does result in a return to the menu	Pass
ShowChangeLog function should print the change log to the terminal when called	Selecting 11 in menu should result in change log being printed in terminal	Selecting 11 does result in change log being printed in terminal	Pass
ShowNumberOfLines function should take input of file name and print number of lines in that file to terminal	Entering a file name should result in the correct number of lines being printed in the terminal	Entering a file name does result in correct number of lines being printed in terminal	Pass
ShowNumberOfLines function should return to menu if invalid input is given	Entering no input should result in program returning to the menu	Entering no input does result in a return to the menu	Pass

Selecting option 13 (Quit) in menu should result in program terminating	Entering 13 when prompted for a choice in menu should stop the menu from being displayed again and stop the program	Entering 13 does stop the program	Pass

6 Evaluation

The testing conducted in the previous section demonstrates that the program fits the requirements and specification initially provided. The testing shows correct implementation of the following operations:

- Menu displaying other operations - Called recursively until user decides to exit program
- Create File - Creates file or returns to menu if invalid input
- Copy File - Copies file or returns to menu if invalid input
- Delete File - Deletes file or returns to menu if invalid input
- Show File - Shows file or returns to menu if invalid input
- Rename File - Renames file or returns to menu if invalid input
- List Files - Lists files and their permissions, asks if user wishes to change permissions, changes permissions. Returns to menu if invalid input
- Append Line - Appends line or returns to menu if invalid input
- Delete Line - Deletes line or returns to menu if invalid input
- Insert Line - Insert line or returns to menu if invalid input
- Show Line - Shows line or returns to menu if invalid input
- Show Number Of Lines - Shows number of lines or returns to menu if invalid input
- Show Change Log - Shows change log then returns to menu
- Append Change Log - Appends change log

7 Conclusion

My implementation meets the design criteria, but there can be some improvements, for example:

- Append Line - Allow the user to append multiple lines at once
- Delete Line - Allow the user to delete multiple lines at once
- Insert Line - Allow the user to insert multiple lines at once
- Show Line - Option of displaying lines in file from line number x to number y
- Append Change Log - Include the date of the change being made
- Append Change Log - Include every line that was removed/added
- Show Change Log - Could be altered to show only up to a 50 recent changes, currently the entire file is printed to terminal and this may be overwhelming once the file reaches a certain size
- Initially had idea of implementing an encryption function which could lock files behind passwords, however this proved too difficult to implement so ultimately the function was never implemented

References

[1] Studymite, https://www.studymite.com/blog/strings-in-c. *Article on strings in C.*

[2] tutorialspoint, https://www.tutorialspoint.com/c_standard_library/stdio_h.htm. *C <stdio.h> library.*

[3] tutorialspoint, https://www.tutorialspoint.com/c_standard_library/stdlib_h.htm. *C <stdlib.h> library.*

[4] tutorialspoint, https://www.tutorialspoint.com/c_standard_library/string_h.htm. *C <string.h> library.*