Portable File Manipulator Design Guide

Yash Gupta, Shaun Cyr, Chacko Panicker

December 3, 2019

1 Introduction

The purpose of this project is to design and implement a portable file manipulator independent of the operating system. The program uses standard C & C++ commands making it easier to execute it without downloading extra libraries. The program provides an easy interface for the user to interact with. The program uses uncomplicated commands and help section to provide ease to the user.

2 Design

The program is divided into 3 major sections

1. Operations
   - Create
   - Copy
   - Delete
   - Move
   - Rename
2. User Interface
3. File Editor
   - Add text
   - Append
   - Insert Text
   - Truncate

2.1 Operation

2.1.1 Create

This function will create a file using the standard c++ notation without the use of any system calls to ensure that its functionality is cross platform and will require no midifcation to this function to make it run on other operating systems. This will work by writing to a file that is not currently saved to disk, and as a result the c++ language implementation will create the file as needed.

2.1.2 Copy

This function will use the same functionality style as specified in 2.1.1 above but will add contents to the new file based on what the contents of the original file contained. This will be done in sections in accordance to the size of the character buffer provided to ensure security of the system and extensible file sizes.

2.1.3 Delete

This will work by checking if the file exists, and if so, calling the existing c++ method remove() with a parameter of the filepointer to delete the file. This will cause the operating system to mark the relevent parts of hard disk space as free, and will eventually write over that area with new data as needed.

2.1.4 Move

The 'move' command is used to transfer a file from one directory/folder to another directory. The user selects the file and then navigates to the desired directory through using the interface and places the file there using the 'here' command.

2.1.5 Rename

The 'rename' command is used to rename a file present in the working directory by taking the orignal filename and the new filename.

2.2 User Interface

The User Interface implements a level command line interface for the user to interact with. It takes in the command and the parameters from the user. The command is further inspected for validation and further relative functions are called to perform the operation requested or else an error is thrown along with an example on how to use the command and the user is asked to refer to the help command. The UI uses the cd command for moving throght directories and the 'ls' command to print a list the content of the working directory. The 'clear' command is used by the user to clear the entire the console of the interface for when it gets full with previous results and commands. The 'help' command is used to provide a reference guide to the user for all the commands and thier sytax.

2.3 File Editor

While using the text file editor, the user is presented a menu driven interface. The user chooses to write or append text to a file, insert text to a file, remove all contents from a file, or display contents of a file. Each function prompts the user to enter the text file they wish to work with from the directory the text editor was opened. The program

The text file editor when called presents the user with a menu driven interface with all its functions.

2.3.1 Add/Appendtext

The add/append text function prompts the user to enter the filename to write too. If the text file exists, the function asks the user to enter text and appends it to the end of the existing file. Else if the entered filename does not exist, the function creates a new text file and writes to it.

2.3.2 Display content

Display all content function asks the user to enter the file to read. If found, it asks the user the number of lines to be displayed at once. The function displays that number of lines at once and prompts the user to press enter to display the next set of lines.

2.3.3 Insert Text

Insert text function asks the user to enter the filename to insert text into. If the file is found, it displays the total number of characters in the file and waits for the user to enter which position to seek to. If an incorrect seek distance is entered, a corresponding error message is displayed. Else if everything works as expected, a Success message is displayed.

2.3.4 Remove all content

The remove all content truncates the file entered by the user if it exists, otherwise displays a file not found error.

3 Implementation

3.1 Operation

3.1.1 Create

The Create() method has little complex implementation requirements other than as outlined in the design section above.

3.1.2 Copy

The Copy() method has little complex implementation requirements other than as outlined in the design section

3.1.3 Delete

The Delete() method has little complex implementation requirements other than as outlined in the design section above.

3.1.4 Move

This 'move' command initially takes in the filename that has to be moved and throws an exception if no such file is found. In the event that the file exists in the working directory, the interface prompts the user to navigate to the desired directory. Once the user is in the desired directory, 'here' command is used to move the file.

The implementation of the move command is similar to the moving operation in a GUI. The move commands invokes the move() function. The move() function further calls deletefile(), copy() & rename(). The function checks for the right amount of parameters and throws an exception if its more than 1. The function saves the old directory of the file to be moved and creates a copy of the file in the desired directory. After the copy is created the, the file is deleted from the old directory. Once the process is complete, the funcion returns to the old directory which was initially stored.

3.1.5 Rename

The 'rename' command takes in two parameters, the orignal filename and the new filename. It then creates a copy of the orignal file using the new file name with the copy() function and deletes the orignal file using the deletefile() function.

3.2 User Interface

The UI for the system is a close implementation of command line. The UI runs in a constant loop which breaks with an 'exit' command. The UI starts the program with the directory it is executed in. The UI takes in the command and the parameters and calls the respective functions. If the command is not found, it throws an exception and prompts the user to enter the 'help' command. The UI splits the command into operation and paramters that are to be passed to the functions. The interface also provides some special commands :

1. cd

   The cd command helps the user to navigate through directories, if the derectory is a successor or the predecessor of the current directory i.e. if its up or down the current directory. The command invokes the callcd() function which further uses the POSIX library function chdir(). The command takes in the directory name to go up or '..' to go down a directory.

2. ls

   The ls command prints a list of all the files and folders present in the list. The ls command invokes the callls() function to print the list. The callls() function uses the ls() to get the list. The ls() uses a DIR pointer variable to open the directory using opendir() and parse through it using the readdir() function. The ls() returns a vector of strings to the callls().

3. help

   The 'help' command is implemented with a lot of output statements since the purpose of the command is to display the correct syntax of the commands and an example on how the command is used and what it does. The command invokes the help() function.

4. clear

   Since system calls are only specific to vertain operating systems, to implement this command a looping structure was used to print new lines clearing the console for the user. The command invokes the callclear() function.

3.3 File Editor

The file editor is called from the directory the user is currently on. When called, a do-while loop presents a menu to the user with all functions it is programmed to do. The user can

1. Add/Append file
2. Insert text in a specific position of a file
3. Remove all text in a file
4. Display all contents of a file

User can enter their choice, and the program calls the respective functions. When prompted whether they wish to continue, the user can exit by entering n or go back to the menu by entering y.

3.3.1 Add/Append File

The function when called, asks the user to enter which file they would like to open and stores the input into a string. It then checks if the filename entered by the user contains .txt at the end in order to open files correctly. If it does not, it appends the inputted string with .txt and proceeds. The function creates a new file if it does not already exist or opens the file if it already exists. The user is then prompted to enter content to write to the file. The content is written to the new file created or appended to the end of the existing file.

3.3.2 Display Content

The function asks the user which file to be displayed. If no such file is found, an error message is displayed to the user. Else, the program asks the user the number of lines to be displayed at once. Then the program starts reading the file using a for loop. The file is displayed in such a way that one line consists of thirty characters. Each time a line is displayed, an integer variable line-counter is incremented by one. When the line-counter matches the number of lines to be displayed by the user, the program prompts and waits for the user to press the Enter key to display the next set of lines. The line-counter is reset every time the set of lines is displayed.

3.3.3 Insert Text

The function asks the user which file the user would like to insert text into. It checks if the file exists and displays the appropriate error message if it done not. Else, it counts the number of characters and stores it into an integer variable count. It then uses the count and prompts the user to Enter number of charecters" and displays how many characters are in the file so the user enters an appropriate value to seek.

3.3.4 Remove all content

The function asks the user which file the user would like to remove all contents from. The program checks if the entered file string contains .txt and the end and checks if it exists. If found, it opens the file with truncate, which removes all contents from the file. A success message is displayed and the file is closed.