Ali Emre Öz 213950785

Ali Süheyl Ersözlü 212071104

Fatih Çağatay Gülmez 213962062

**CS 352 FINAL REPORT**

**Introduction**

In this project, Portable File Manipulator (POFM) is designed. Following report explains the C implementation of the project. He corresponding program lets the users do some operation on file system in any Operating System, files can be created, deleted, copied, moved and renamed, also text files can be manipulated by adding text and removing text from that file.

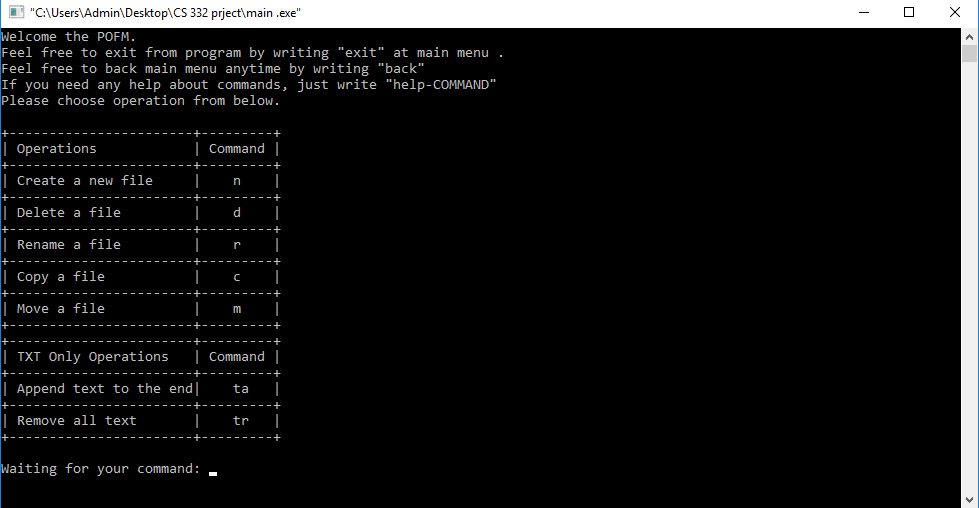
**Contents**

1) Design Guide and User’s Manual

* 1. Main
  2. CreateFile
  3. DeleteFile
  4. CopyFile
  5. RenameFile
  6. MoveFile
  7. AppendText
  8. RemoveText

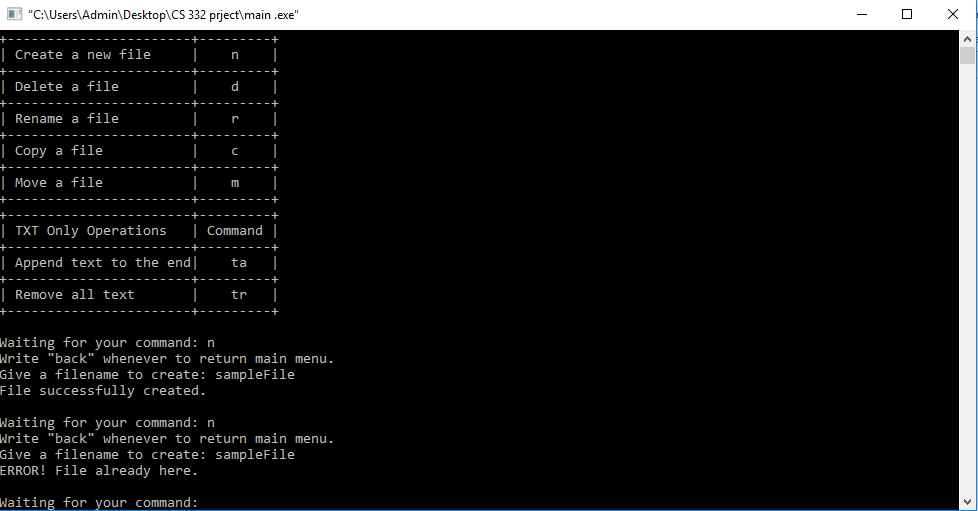
**Design Guide**

1. **Main**

The main file of the project. The aim is creating main menu of the program and taking an input command from users. After some instructions and information, the code starts an endless loop till the “exit” command. If the command is recognized, the functions calls corresponding header file which includes the command’s code. The recognized commands can be seen on the picture below, also “help” commands can be written at the main menu. For instance, if user wants to get help about create a file function, s/he should write the command “help-n”. User just need to write the command keyword into screen, and program will redirect the user to corresponding function’s menu.

1. **CreateFile**

This function can be found at createFile.h file. Firstly, if program enters that function that means if user type “n” command at main menu, a file name is expecting by user input to create a file. If user types a file name which is already exist, the function give an error and returns. If user’s input is valid, file will be created if the operation permitted and program will return to main menu. If user wants to cancel the operation, s/he can write “back” at any time to return main menu. For example, let’s create the file called “sampleFile” and then try to re-create it. The screenshot below is showing outputs and program interface for createFile function.

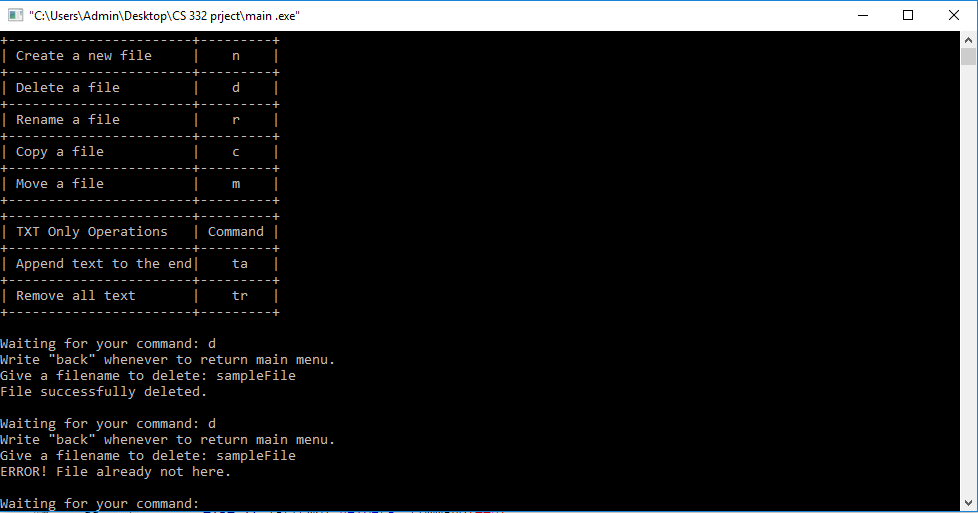


For this function, the code implementation is done by fopen() function. That creates a file without system calls and the most basic code for file creation. Also strcmp() used for command recognitions.

1. **DeleteFile**

This function can be found at deleteFile.h file. Firstly, if program enters that function that means if user type “d” command at main menu, a file name is expecting by user input to delete a file. If user types a file name which is already not exist, the function give an error and returns. If user’s input is valid, the file will be deleted if the operation permitted and program will return to main menu. If user wants to cancel the operation, s/he can write “back” at any time to return main menu. For example, let’s delete the file called “sampleFile” which is already created and exist and then try to re-delete it. The screenshot below is showing outputs and program interface for deleteFile function.

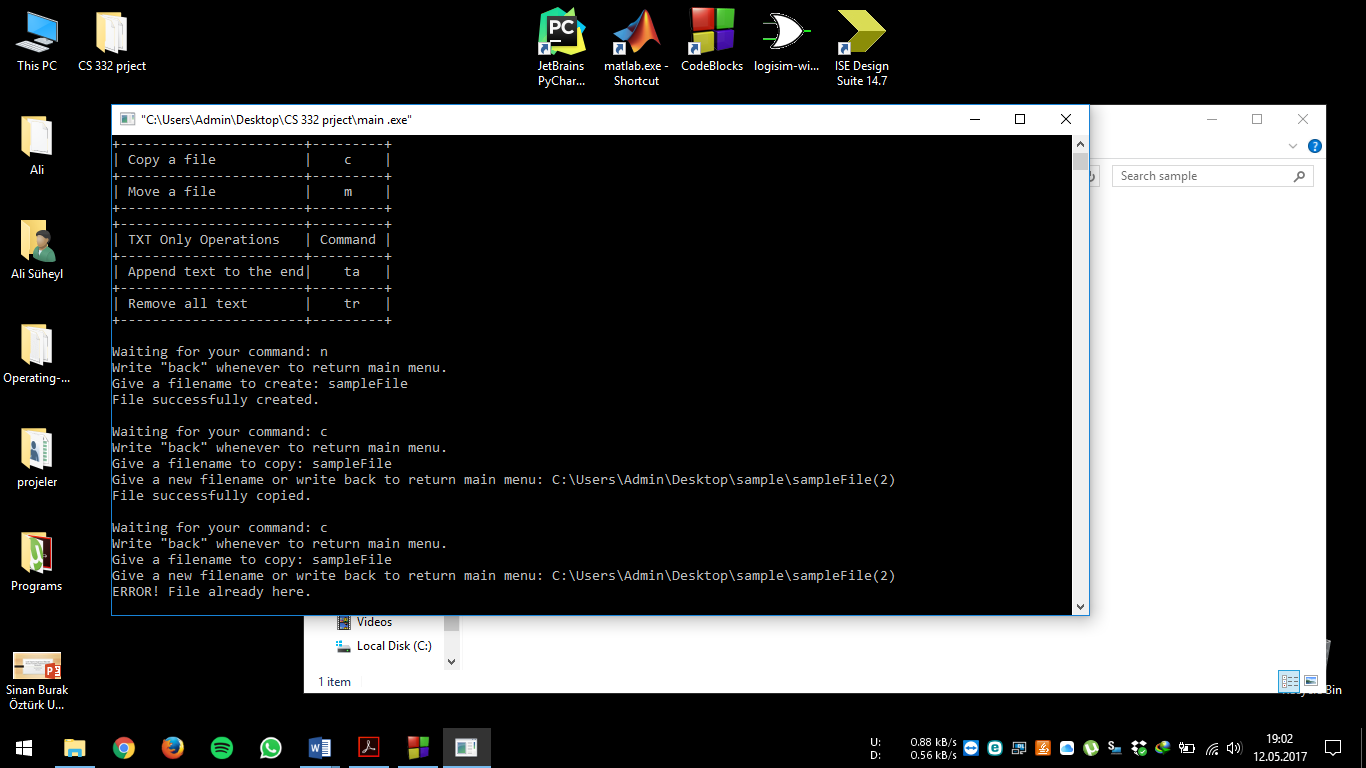
For this function, the code implementation is done by remove() function. That deletes a file without system calls and the most basic code for file deletion. Also strcmp() used for command recognitions.



1. **CopyFile**

This function can be found at copyFile.h file. Firstly, if program enters that function that means if user type “c” command at main menu, a file name is expecting by user input to copy a file. If user types a file name which is already not exist, the function give an error and returns. If user’s input is valid, the program wants another input by user which is the path includes new file name or just name. The file exist at given path, function gives an error and holds for another directory or file name. If both input is valid, the file will be copied if the operation permitted and program will return to main menu. If user wants to cancel the operation, s/he can write “back” at any time to return main menu. For example, let’s copy the file called “sampleFile”. The screenshot below is showing outputs and program interface for copyFile function.

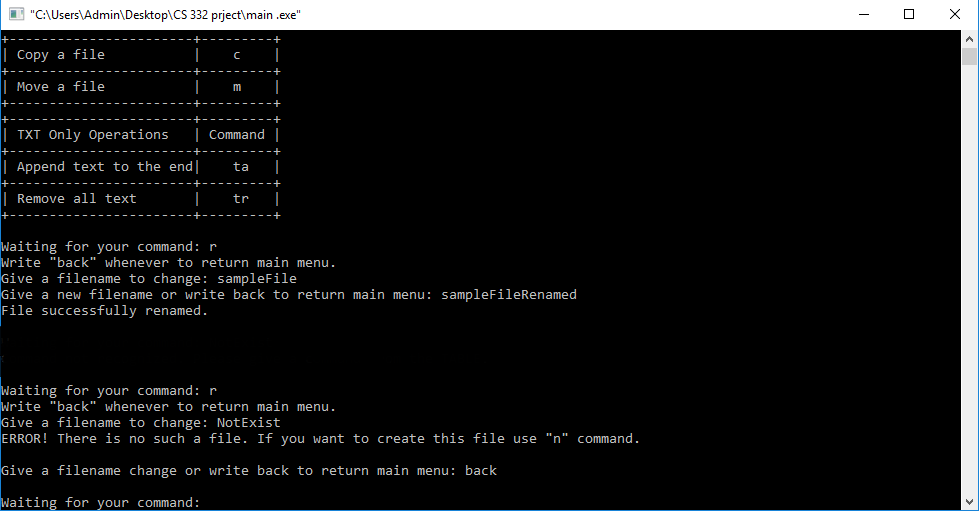
For copying a file, the code reads the source file character by character with fgetc() function and then rewrites it to new file with fputc(). fgetc is good for reading a file because strings are not useful in C, chars are much more easier to work on.



1. **RenameFile**

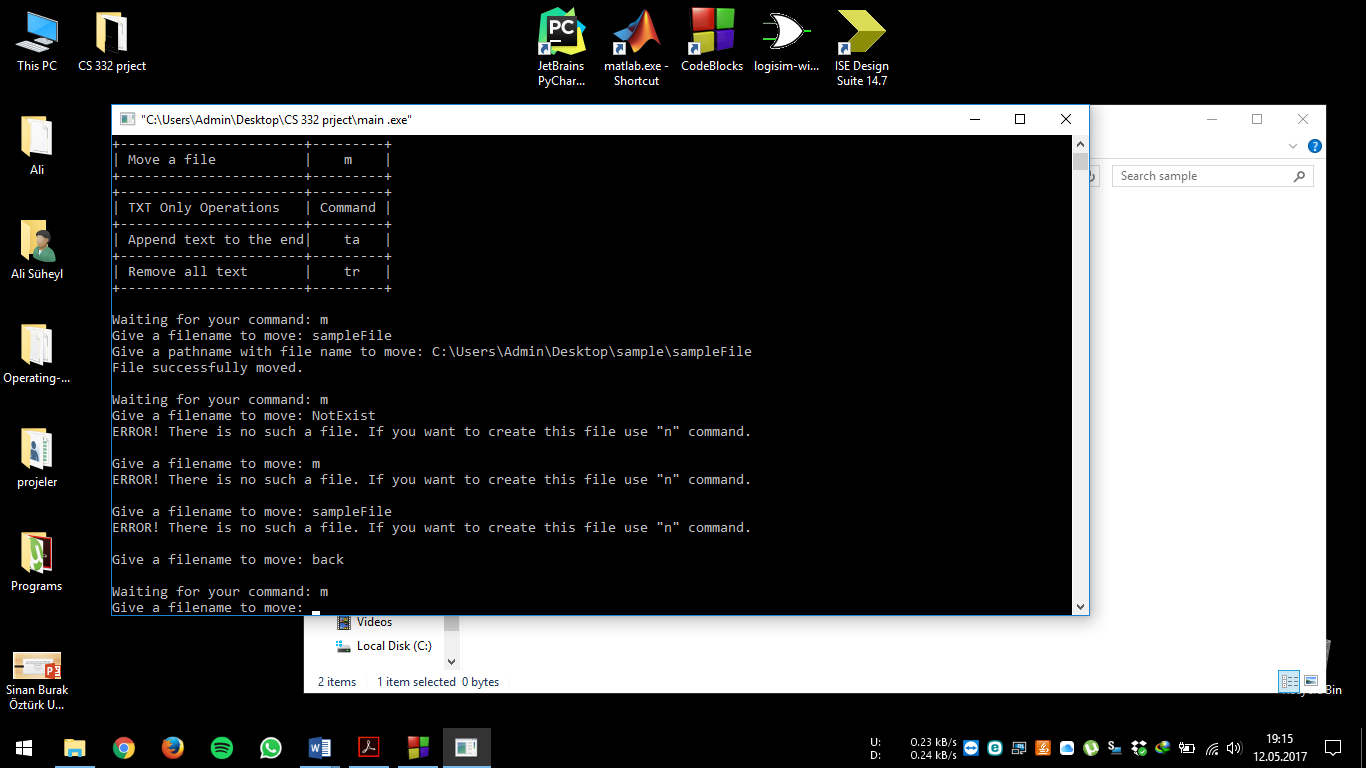
This function can be found at renameFile.h file. Firstly, if program enters that function that means if user type “r” command at main menu, a file name is expecting by user input to rename a file. If user types a file name which is already not exist, the function gives an error and returns. If user’s input is valid, the program wants another input by user which is the new file. The file exist at given path, function gives an error and holds for another file name. If both input is valid, the file will be renamed if the operation permitted and program will return to main menu. If user wants to cancel the operation, s/he can write “back” at any time to return main menu. For example, let’s rename the file called “sampleFile” as “sampleFileRenamed”. The screenshot below is showing outputs and program interface for renameFile function.

The code of rename file is just uses rename() function. This implementation is good and easier.



1. **MoveFile**

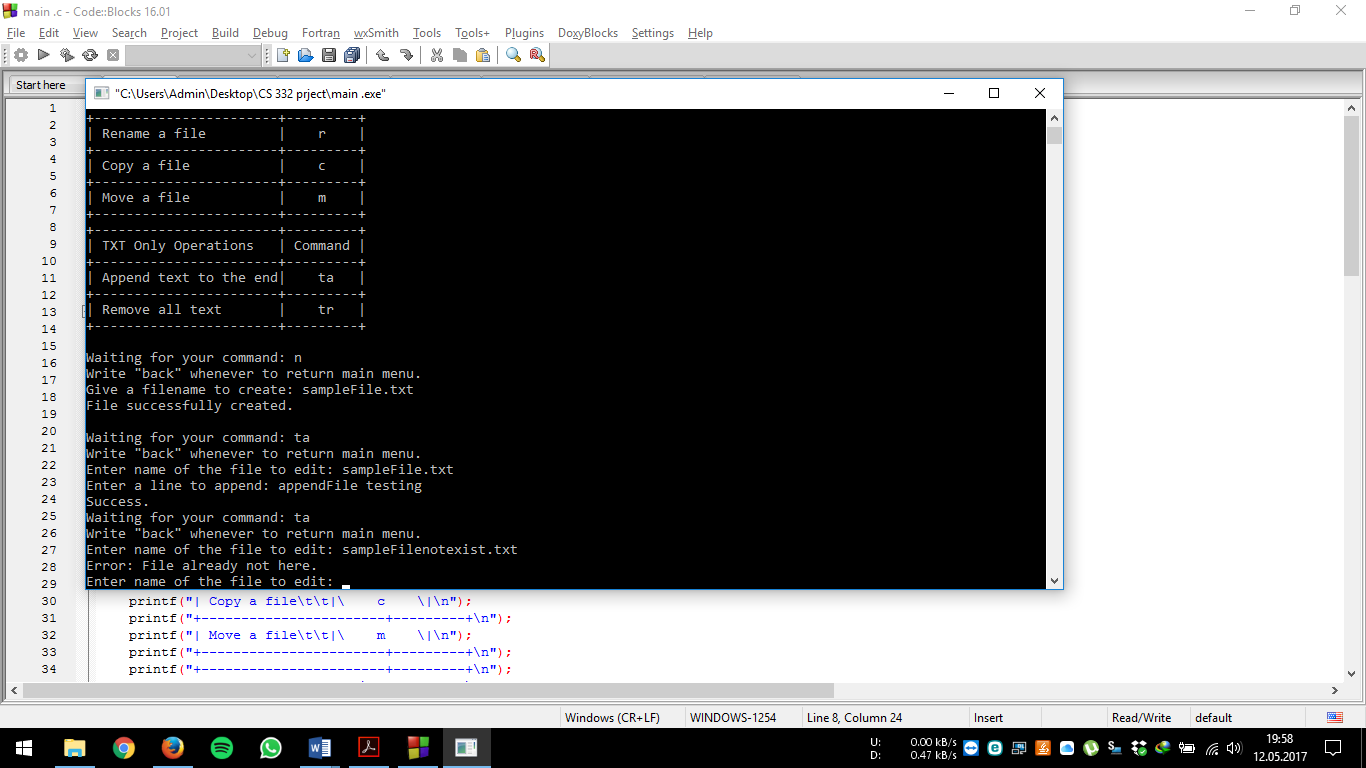
This function can be found at moveFile.h file. Firstly, if program enters that function that means if user type “m” command at main menu, a file name is expecting by user input to copy a file. If user types a file name which is already not exist, the function gives an error and returns. If user’s input is valid, the program wants another input by user which is the path includes new file name or just name. The file exist at given path, function gives an error and holds for another directory or file name. If both input is valid, the file will be copied if the operation permitted and program will return to main menu. Basically, function uses copyFile and deleteFile functions’ codes. If user wants to cancel the operation, s/he can write “back” at any time to return main menu. For example, let’s move the file called “sampleFile”. The screenshot below is showing outputs and program interface for moveFile function.



The code implementation inspired from copyFile and deleteFile codes, a file is copying to new direction and then source file is deleted.

1. **AppendText**

This function can be found at appendText.h file. Firstly, if program enters that function that means if user type “ta” command at main menu, a file name is expecting by user input to insert a text into a file. If user types a file name which is already not exist, the function gives an error and returns. If user’s input is valid, the program wants a text to append into the file. After that program will return to main menu. If user wants to cancel the operation, s/he can write “back” at any time to return main menu. For example, let’s append a text into the file called “sampleFile.txt”. The screenshot below is showing outputs and program interface for appendText function. The implementation is done by functions called fgets() and strcat(). A buffer used in this implementation to get keyboard inputs from user as string variables. Fgets is saves keyboard inputs to buffer and then we take text from buffer, open file in append mode and then append text to the file.



1. **RemoveText**

This function can be found at removeText.h file. Firstly, if program enters that function that means if user type “tr” command at main menu, a file name is expecting by user input to remove the text in the file. If user types a file name which is already not exist, the function gives an error and returns. If user’s input is valid, all text in the file is removing. After that program will return to main menu. If user wants to cancel the operation, s/he can write “back” at any time to return main menu. For example, let’s remove a text in the file called “sampleFile.txt”. The screenshot below is showing outputs and program interface for removeText function. The implementation of the functions is inspired by deleteFile and createFile. Simply the function first deletes the file and then creates it again.

