Braden Burriss, Ethan Harness, Harper Hill

Dr. Rojas

Programming Principles II

8/1/2021

Final Report Document

Objectives- In our project the goal was to be able to access, search through, and even edit files on a server machine from a client. We did this by connecting our machines to a server using the winsock library. Then using many helper functions to navigate directories, search files, and pull files.

Inputs and Outputs to the System- You would input the server’s IP on the client and then use numbers to select options for what you want to do in a particular directory. You would be able to go into a new directory, pull a file, or search a file.

Structure Chart-

Detailed description of all functions-

int isExactMatch(char fileName[], char searchFor[]);

This function checks if there is an exact match for a particular word in a file. It takes in a “filename[]” which holds the file name, and “searchFor[]” which holds the word the function searches for. It returns a 1 if the word exist, and 0 if the word does not.

int isRoughMatch(char fileName[], char searchFor[],int characters);

This function checks if there is a rough match for a particular word in a file with “characters” amount of consecutive matching characters. It takes in a “fileName[]” which holds the file name, “searchFor[]” which holds the word the function searches for, and “characters” which holds the required amount of consecutive matching letters for a word to be a rough match. It returns a 1 if there is a word that fits the requirements, and a 0 if there is not.

int getWord(FILE \*searchThrough, word \*currentWord);

This function gets the next word in a file. It takes in a file pointer “\*searchThrough” and scans the next word to “\*currentWord’s” “thisWord” inside the structure. The int return is only used as a way to exit the function when it is done scanning a word.

void makeLowercase(char word[]);

This function makes a word all lowercase. It takes in a “word[]” and edits it so that it is all lowercase. Being void, there is no return.

void cleanString(char string[]);

This function makes a string into something comparable by removing everything that is not a space. It takes in a “string[]” and edits it to be all characters. Being void, there is not return.

int closeWord(char keyWord[],char currentWord[],int characters);

This function is a helper to the isRoughMatch function. Ever word scanned is sent through this function to see if it fits the requirments. It takes in a “keyword[]” or the word the user is looking for, “currentWord[]”, the most recent word from the file being compared, and “characters” the amount of characters that need to be consecutively matching. It returns 1 if the consecutive matching letters are greater than the required “characters” and 0 they are not.

int continueCheck(char keyword[],char currentWord[]);

This function is a helper function for closeWord. When closeWord finds a matching character in the words, it passes the two words from the character they are at to the end. This allows for more easy comparison of the rest of words. It takes “keyWord[]” which holds the rest of the word the user is searching for and “currentWord[]” which is the rest of the current word. It returns the number of consecutive matching characters from the indices passed in.

Test Plan- For our test plan we planned to use different machines on both the server and client side. Then navigate through directories to find files and scan, pull, or edit said files. This would allow us to test the connection capabilities of the program, directory navigation, and file handling.

Test Results-

Division of Labor- Braden: File scanning algorithm, Harper: Functions for pulling files, Ethan: User interface and directory navigation.

Observations- The searching through files was something that worked well, having rough match and exact match functions allow for high variability in scanning files. Sending and receiving information was something that worked well.

There were many things that did not work quite as planned. Everything involving sockets was somewhat difficult when it came to speeds of data transmission. Data would send too slow or too fast and cause our program to break. This forced us to use sleep function to make the program wait. There were issues with memory corruption when navigating directories. One thing we thought we would be able to do was scan word documents. However, reading “.docx” files turned out to be something that would be much more complicated to do in C then we anticipated.

To improve in the future, we could broaden the capabilities of our program. Allow the user to scan other document types like “.docx” files or maybe even PDF documents. The user interface would be better if it had more error catching. We could also improve data transmission between the server and the client.