User Input Program

Stephan Peters

Colorado State University Global

CSC450-1 23WD: Programming III

Reginald Haseltine

17 March 2024

User Input Program

For this assignment, the students were asked to create a console application in C/C++ that appends data from the user to a text file, reads the contents of that text file, reverses the contents of that text file, and saves them to a new text file.

The main() Runner Engine

Since the directory path may defer when compiling from different Integrated Development Environments (IDEs), I created separate variables for the directory path and file name which are merged into a file path variable. My IDE compiles the executable into a cmake-build-debug directory, so I set the `directoryPath` variable to `"../"`. This may need to be modified (for example, an empty string; `""`) to compile using a different build system. The program will throw an exception if the `CSC450_CT5_mod5.txt` file is not found where expected.

I wrote each section as a C function; the main program runs these functions sequentially. Those functions are:

- `append_string` – Gets data from the user console and appends the data to a file. Allows for multiple lines. Terminated by entering `q`.
- `exists_file` – Checks if a file exists. Used by `append_string` to throw an exception if the file to append to does not exist.
- `read_file` – Reads the entire contents of a file and returns them as a string. Used by `main()` to read the contents of the appended file.
- `reverse_string_to_file` – Reverses a string and saves the result to a file.

Source Code 1 shows the `main()` function.

Source Code 1

Source code for the main function.

```
int main() {
    // Add a directory path here if necessary. I use "../" for my IDE.
    string directoryPath = "";

    string fileName = "CSC450_CT5_mod5.txt";
    string reverseFileName = "CSC450-mod5-reverse.txt";

    append_string(directoryPath + fileName);
    fprintf(stdout,
            "Data appended to %s successfully. \n",
            fileName.c_str());

    string fileContents = read_file(directoryPath + fileName);
    reverse_string_to_file(fileContents, directoryPath + reverseFileName);
    fprintf(stdout,
            "Contents of %s successfully reversed and stored in %s. \n",
            fileName.c_str(),
            reverseFileName.c_str());

    return 0;
}
```

Running the Program

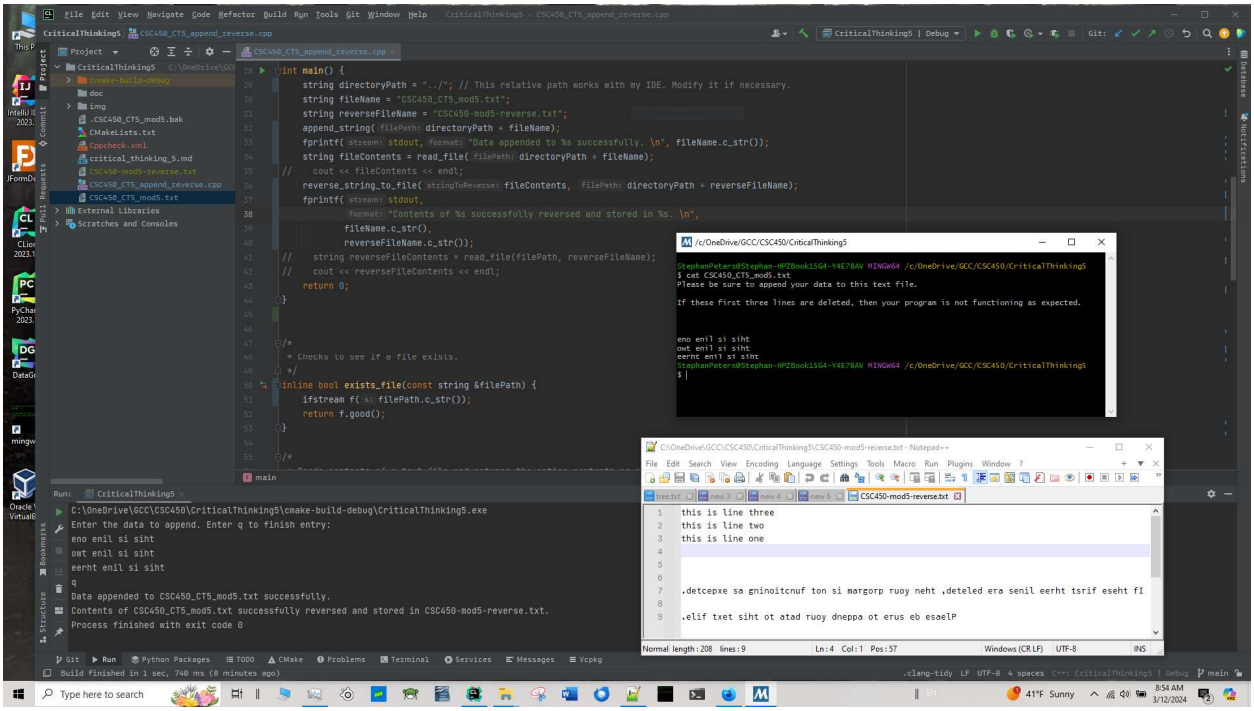
I ran the program appending the following lines to the file, then viewed the contents of the two files with MSYS2 shell and Notepad++.

```
eno enil si siht
owt enil si siht
eerht enil si siht
```

Figure 1 shows execution after compilation, and displays the revised contents of the original file as well as the contents of the reversed file:

Figure 1

Program Execution



Vulnerability Assessment

The program has user console input and several streams, which can introduce vulnerabilities to the program. All open streams are closed at the end of the functions, and the functions are isolated as well. When the user is asked for input, each line is stored in the stream as a string until written to the file, then the stream is closed. Attacks from malicious modification of the files are mitigated by reading them into strings. The source file has been checked for vulnerabilities using Cppcheck.

GitHub Repository

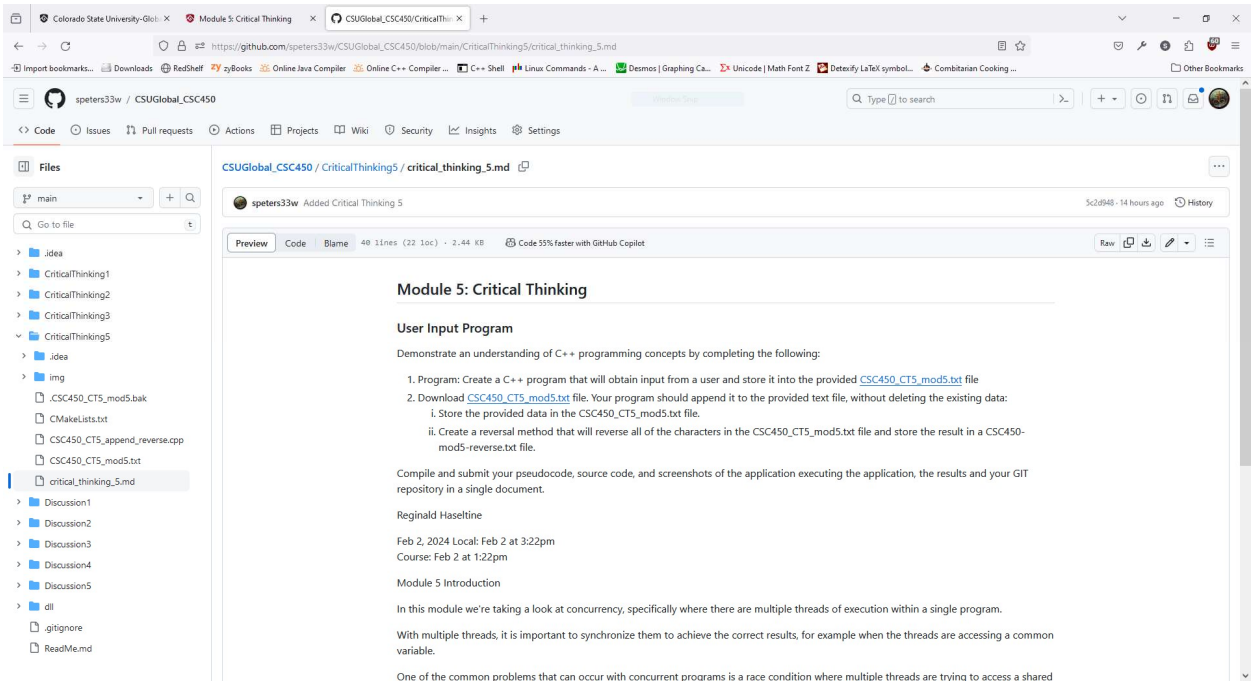
In addition to the solutions above, I was tasked to create a GitHub repository for the project. This repository is located at

https://github.com/speters33w/CSUGlobal_CSC450/tree/main/CriticalThinking5

Figure 2 shows a screenshot of the main page of this repository.

Figure 2

Image of the main page of my CSC 450 Critical Thinking 5 GitHub repository



Appendix A – Full Source Code as Submitted

Source Code 2

Final source code as submitted for compilation.

```
/*
 * A simple C++ program that obtains input from a user
 * and appends the user input into a file
 * Then reads the contents of the file,
 * reverses the contents,
 * then stores them in another file.
 * (CSC450_CT5_append_reverse.cpp)
 */
#include <algorithm>
#include <fstream>
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

inline bool exists_file(const string &filePath);

string read_file(const string &filePath);

void append_string(const string &filePath);

void reverse_string_to_file(string stringToReverse, const string &filePath);

/*
 * This is the main runner engine for the program.
 */
int main() {
    // Add a directory path here if necessary. I use "../" for my IDE.
    string directoryPath = "";

    string fileName = "CSC450_CT5_mod5.txt";
    string reverseFileName = "CSC450-mod5-reverse.txt";

    append_string(directoryPath + fileName);
    fprintf(stdout,
            "Data appended to %s successfully. \n",
            fileName.c_str());
    string fileContents = read_file(directoryPath + fileName);
//    cout << fileContents << endl;

    reverse_string_to_file(fileContents, directoryPath + reverseFileName);
    fprintf(stdout,
            "Contents of %s successfully reversed and stored in %s. \n",
            fileName.c_str(),
            reverseFileName.c_str());
//    string reverseFileContents = read_file(directoryPath +
reverseFileName);
//    cout << reverseFileContents << endl;
```

```
    return 0;
}


/*
 * Checks to see if a file exists.
 */
inline bool exists_file(const string &filePath) {
    ifstream fileExistStream(filePath.c_str());
    bool good = fileExistStream.good();
    fileExistStream.close();
    return good;
}

/*
 * Reads contents of a text file and returns the entire contents as a
string.
 */
string read_file(const string &filePath) {
    ifstream fileContents(filePath);
    stringstream buffer;
    buffer << fileContents.rdbuf();
    fileContents.close();
    string returnString = buffer.str();
    buffer.clear();
    return returnString;
}

/*
 * Appends user input from the console to a file.
 * Allows multiple lines.
 * The user indicates the input is complete by typing q.
 */
void append_string(const string &filePath) {
    bool done = false;
    string userString;
    ofstream fileStream;
    if (!exists_file(filePath)) {
        fprintf(stdout,
                "Could not open file for editing: %s.",
                filePath.c_str());
        throw std::ios_base::failure("Could not open file for editing: " +
                                     filePath);
    }
    fileStream.open(filePath, ios::out | ios::app);
    cout << "Enter the data to append. Enter q to finish entry: " << endl;
    while (!done) {
        getline(cin, userString);
        if (userString == "q") {
            done = true;
        } else {
            fileStream << endl << userString;
        }
    }
    fileStream.close();
}
```

```
/*
 * Reverses a string and saves it to a file.
 * The string may include line breaks.
 */
void reverse_string_to_file(string stringToReverse, const string &filePath)
{
    ofstream fileStream;
    reverse(stringToReverse.begin(), stringToReverse.end());
    fileStream.open(filePath);
    fileStream << stringToReverse;
    fileStream.close();
}
```

References

Du, W. (Kevin). (2014). Format String Vulnerability printf (user input). In *Department of Electrical Engineering and Computer Science, Syracuse University*. https://web.ecs.syr.edu/~wedu/Teaching/cis643/LectureNotes_New/Format_String.pdf

LePage, G. (2017, November 3). *How do I clear an int pointer (int *pointer) from memory on C++?* Quora. https://www.quora.com/How-do-I-clear-an-int-pointer-int-*pointer-from-memory-on-C++

Reid (ModShop), B. (2012, October 18). *How to delete a variable. - C++ Forum*. Cplusplus.com. https://cplusplus.com/forum/beginner/82290/

Stieber et. al., C. (2017, May 23). *What does delete command really do for memory, for pointers in C++?* Stack Overflow. https://stackoverflow.com/questions/11603005/what-does-delete-command-really-do-for-memory-for-pointers-in-c