Hayden Kendall

Prof. Jean-Yves Hervé

CSC 412: Operating Systems and Networks

23 March 2020

Programming Assignment Two

This assignment took an introductory look into C code syntax and bash shell scripting. The combination of these two concepts resulted in a program that outputs a sequence of Fibonacci numbers given a primed input by the user. The program then generates text files containing the summations in a directory named “Results”. In addition to the base requirements, the extra credit opportunities were also fulfilled. This resulted in a program that also moves the contents of “Results” to a new directory named “Results_n”, depending on the number of n folders already in existence.

An important design choice with any program is modularity. I was instructed to create a separate function in C with the task of calculating the Fibonacci sequence at a specified number of iterations. One limitation that I did come across, being one who has not touched C in some time, was that only one value can be returned out of a function. In this scenario I took a “less is more” approach which improves overall readability. While one can return a pointer to an array, I felt that a much simpler solution would be to call the function twice. The function as such takes three parameters, the first Fibonacci number, the second Fibonacci number, and the index of which to calculate the sequence. In the main function one can call the Fibonacci function twice and redirect those outputs as now two distinct variables. I felt that this was a cleaner method because in the other scenario one would need to also do the Fibonacci loop twice within the function to get two outputs. It is more modular to call the same function twice for simplicity, ease of data manipulation, and readability as the calculations would be performed twice anyways.

That is not to say that there were not difficulties in programming. One situation that I came across in this assignment, as well as when I took the original course, was C code editing in the Ubuntu environment. At the time I was resorting to the nano text editor to write out code. While it does get the job done, the process becomes more tedious with simplicity. Breakpoints and debugging have become a great feature I have learned to take advantage of over the years. I have found that using a proper IDE for coding allows for a much easier development process where bugs and errors are easily solvable.

Another major hurdle was the use of bash shell scripting. At the time of taking the course I remember being intimidated by the character intensive language that I could never quite get a good grasp on throughout the semester. The most important ways I have found to learn new languages is to take the modularity approach. In this assignment one outcome was required which was to call the C program multiple times and write the outputs to a file. At the beginning I was clueless on how to do this. What was important was that I broke each step down to its simplest core. First learning to pass arguments to the script through terminal, calling the C function a single time, then reading the txt file for the outputs to store them in variables, and finally moving individual files into a new directory. Once I

was able to accomplish those steps the extra credit portions came easy to me and I was able to get a beginner understanding of bash scripting.

I did not take an intensive amount of time to ensure that my program would not completely crash and fail in any possible way. One portion of criteria for the assignment was to implement forms of error checking to return to the terminal if certain situations were encountered. However, a known limitation to this program is the if the user were to attempt to pass strings of text instead of numerical values as arguments. In the main function the argv[] components come in as a series of characters that are converted to int datatypes. This program will likely error out when attempting to convert a nonnumerical value or produce an incorrect output.

Another limitation that I did notice with this program is that I accidentally passed large index numbers through (151, 312). The output of the program was a large number in association to the first index that I think calculated correctly. The second index resulted in a negative number showing that the program calculated the sequence incorrectly. I have reason to believe that as Fibonacci numbers grow in a quick rate, the number eventually surpassed the data limitations of the int datatype. Using long would probably fix this limitation.

With the bash script in mind, I did mention that this was a new language that I was learning at the time of writing and something I never picked up well during my time at University. I do believe that functionally the script does what is intended however there can certainly be more sophisticated ways to accomplish it. For example, once the files are created, I essentially write the same line three times to move each individual file to the “Reports” directory. I saw mention of a wildcard notation that would have condensed the three lines into one. If in the future more output files were needed, the bash script likely would not need to be edited.

Overall, I found this assignment to be a nice reintroduction to C programming and pleased to finish it with not nearly as much stress or lack of understanding when in the course. I have taken my evolved experiences in programming and applied them in such a way that the assignment now comes easier and was done in a matter of hours. I do believe that revising more assignments will benefit my understanding of operating systems greatly.