Name: Daniel E. Ems
Date: 06/24/2017
Current Module: Networking in C
Project Name: ashti

Project Goals:
Write a tcp web server that will return information from a directory defined by the user. The web server must be able to return html docs, and return values from executables.

Considerations:

1. What elements of tcp communication will you need to consider?
2. How will you connect to the server?
3. How will the server process requests?
4. How will the server identify valid and invalid requests?
5. How will the server send back the information requested?
6. What functions will be needed in order to access the appropriate information?
7. What will be the best architecture for this project?

Initial Design:
The original design of this program revolved around accomplishing what I have accomplished before, nc. This forced me to cross many bridges that I assumed would be relevant when it comes to the browser. So, as a result, solving the nc requirements was the initial goal. Once that was completed, the next step would be to make it so that the browser worked as well. However, this ultimately never happened. A reason for this may be the design chosen to implement the nc portion of the project. A function was written that would take a buffer of request information and parse it accordingly. This was set up for single receipts from recv, however, it was later discovered that a web browser requires multiple receipts. A crucial flaw in the code.

Data Flow:
The program begins by accepting a directory as a command line argument and firing up the server. The server listens on 127.0.0.1:uid. The server will sit in a n accept block until a connection is established and then it will fork a process to service the connection. The child will receive a single request from the client, send it to a custom function that will check the information passed, and parse it accordingly. The parsed information is stored in a custom struct and returned to the child process. The child process would then check certain values within the struct and send information accordingly. This worked in regards to nc, however, not so much the browser. I am still uncertain as to why. One theory is not reading in enough information. Another is not properly formatting headers (although that was checked thoroughly). However, ultimately, unfortunately, I can not explicitly state the issue.

Communication Protocol:
TCP

Potential Pitfalls:
The only pitfall, which was the missing cornerstone of this project unfortunately, was understanding how to work with browsers over tcp. This ignorance created a slew of problems. First, it prevented me from ever developing a true understanding of the issues I faced. Second, it forced me to do something no developer should ever do, and that is to completely rewrite large sections of code to fix and issue, that could potentially make things worse. I should have branched and then merged. Finally, it led me to developing an architecture that was incompatible with the browser. Again, this was done by creating a function that required a single buf, and not the potential of many. This pitfall, while very

simple. Created numerous issues throughout the program that even now I still do not understand. This really became an issue in not understanding, or knowing what was creating the issue. Trouble shooting was performed with numerous print statements in different areas. However, not knowing what was and was not working was difficult to identify.

Test Plan:
User Test:

All of the user tests will be taking place after the server is up and running. This is because the process of accepting a command line argument is rather standard at this point. With the exception of checking the directory for appropriate access rights. This however, was tested by entering both valid and invalid directories relevant to my local system and printing the results.

1. Attempt to connect to the server via nc
2. printf(connected)
3. attempt to send information to the server
4. printf(received information %s, received information)
5. attempt to access files by sending the nc requests on the requirements sheet.
6. Print at every step in accessing the file for error checking.
7. If errors occur, address them.
8. If successful, print to server.
9. Attempt to send back to the client.
10. Print at every step for debugging purposes.
11. If successful in returning html, attempt to return cgi.
12. Repeat steps 5-9 until successful.
13. Once nc is successful, repeat steps 1-10 from browser.
14. ***** print everything to server for debugging purposes.
15. Ensure you are compliant with all headers
16. repeat steps 10-15 until timeline or completion.

Test Cases:

The test cases that were implemented were all taken from the safety dome project. The first step after successfully connecting was to retrieve the index.html file from the safetydome directory. After 3 hours of debugging a \n at the end of my file path, I had successfully done so. After that, accessing the cgi-bin was rather easy as well over nc, just use popen. The entire time, I was printing in order to ensure I was gettng what I was expecting. These were all of my successful test cases.

My unsuccessful test cases were in regards to my browser. They were much more difficult to track down because everything was printing and appearing to work fine. The error arose when trying to access both the html and the cgi bins. Essentially attempting to recreate my nc test cases. However, after the headers got squared away, I was able to view my first page. However, the test cases in regards to the browser became more and more buckshot like as we came closer to the deadline. As things began to work less and less, it felt as if I was shooting arrows in the dark trying to fix the issues. This created far more problems than it fixed. A better approach would have been to change single items, and then check one by one.

Conclusion:

This project kicked my ass. A large part was self inflicted frustration from dealing with the browsers. If I were to grade my self I would say that I failed. I feel as if I struggled on this far more than many others and I can not help but think it is because I am missing some fundamental understanding of the concepts in place. However, the frustrating part is that I do not know what else I could have understood. I do think however, I learned a lot from this project. About the pitfalls of frustration, as well as HTTP. I would say that my understanding of the materials is better now than it was at the beginning of the project. However, I am still nervous that I have missed some fundamental pieces. Ones that If I feel I know I would have been able to breeze through this project rather that struggle as greatly as I did.