Post-project Writeup - ashti

James Viner

Project Summary

The objective was to create a simplistic, multiplexing TCP web server that accepts HTTP requests for files and returns them to the client.

Challenges

Absolute biggest challenge was implementing HEAD. You might have noticed, you might not have noticed, but HEAD doesn't actually work completely as-intended when used with the contents of a cgi-bin script. I struggled with this problem with it appending an incorrect HTTP header for a minimum of five hours, banging my head against a metaphorical wall before I literally gave up. I'm thinking it probably has something to do with the forking done in the function that executes the script, resulting in two programs that send the client their own responses, but I'm not at all sure. All I know is that chasing this bug cost me hours. This time around, my architecture was poor. I felt like I was in a time scramble for much of the project and the depth of things required meant that I definitely have spent much of the 48 hours in my chair, drinking Monster and trying to implement features. Breaking things up into their own separate functions and files would've been nice, but I really felt as if I didn't have the time to do so, as refactoring existing code felt like a waste of time when I could instead be working on the next requirement. If I had more time, I would've written a shorter program.

Successes

Going to be honest, watching the first text document appear in the browser felt like a dream. It was super neat seeing that all it really takes is a properly-formatted HTTP header telling it how to display the file and it just figures the rest of it out. It's nice. It's refreshing. It just worked without any fanfare necessary besides the initial validation of the file and permissions, etc. I also think that my validation functions worked pretty nicely, on that note. The one that validates a request's "legality" based on what directory it lands in felt pretty clever, using the absolute path to ensure the directory is at minimum as high as the server root. Maybe there's a better way of doing it, but it does work, and that's the important bit to me.

Lessons Learned

I still have trouble wrapping my head around forking and thread-usage sometimes. The idea that two processes or threads are running concurrently makes it difficult to really place what my program is doing at any one particular moment since it can be in two different places if measures aren't put in place to keep things synchronized. I also learned that certain protocols are actually not that difficult once you get to know the syntax and a little bit of the nuance. Managing the HTTP requests and sending back the correct codes and pages was pretty fun once I got in to it, and if I had more time to actually flesh things out, I'm pretty sure I could end up with a competent structure for a basic webserver. Though, at the end of the day, I'd probably still use Apache if I needed to actually host

something. This was a very fun project though, as a learning exercise.