Diary

Networks Assignment 1

1575286

Development

I have used Ian's sample code and built up on it. I started first by understanding what was going on in the sample code. I learned that service_socket_client.c contains the methods that are going to be called per thread, so I have modified its functions and added lots of other useful functions that helped me complete the exercise.

I learned that it is crucial to understand the format of HTTP Response and Request Headers before trying to do anything, so printing the response/request buffers really helps. Requests contains GET followed by /[page], so I created a two For loops to extract the [page] string, and store it. Then it checks if the request contains a Range request. If it did, then the server would return the contents delimited by the specified range after parsing it. If not, then the server would just return the whole webpage to the client based on the requested [page]. Any requests for a non-existing page would result in the server returning a 404 error page back to the client. Any invalid range requests would result in a 416 Range Not Satisfiable response returned to the client, without giving back any content.

I also tried to make sure that the server remains robust with a long input. The size of request buffer is only 1024. A usual request would not be this long too. Any request would only be longer than this if the user requests for an arbitrary long webpage, which results in a request that looks like this: GET /thisisareallylongstringsothiskindofrequestissomethingthatanormalpersonwouldnotnormallyaskforasyoucanseeitisnotagoodideatohaveawebpagenamedthislong … and so on. Upon getting a bad request like these, the server would just return a 404.

Problems

I do not think I encountered many problems in this assignment, but it is just very time consuming. I wrote lots of bad code design to begin with, where I try to fit everything (parsing, file read, etc) inside the while loop, which made code very difficult to read and bugs more invisible. I had to rewrite the whole file once, and learned that design really matters. So, I broke down my code into several functions, each helping one part of the process, making it way clearer to read and debug. I have also added variables that contains headers for responses for different HTTP codes. This was a huge improvement from before.

I have attempted gzip compression, but I found it very hard and complex to do. Parsing the was not the problem, but for some reason I just have troubles understanding how these compressions work. Especially when you need to use a different library to perform the operation. I gave up on it and attempted byte ranges instead. This involved lots of string parsing and I did it with a couple of helper methods. One of them was to determine whether a given range is valid for the requested content, another is to return the partial content if the range is valid.

I have encountered a beginner's mistake bug that costs me more than 4 hours of my sleep. I have error switches as a global variable in my code (these are int error_404; int error_416) and these are initialised as 0 for "false", which reads "error_404 is false". Whenever the process encounters any of these errors, then it would be assigned to 1. So, I have made this working and all. The problem is

after I encountered an error, every other page I navigate to (even valid ones) will still be an error. Turns out I forgot to flip all the error switches to 0 at the end of each loop.

At the end of the day, I have enjoyed working on the assignment. If I was going to do this again, I would make sure to understand HTTP protocol better before attempting it. Or else I would be wasting a lot of time trying on the go like this time.