Development Diary

21st October 2017

Development started on the server. Server networking code is based on the operating systems assignment (threaded version); I decided to finish the OS assignment first specifically so the result could be used as the basis for this assignment.

22nd October 2017

Intended to finish the assignment today, but learned a time extension was provided if I do it in C (which I am), so all I did today was split the server code up into three different C modules to make the code cleaner.

24th October 2017

Spent an hour or two drafting a few methods to keep this server as lightweight as possible. The gist of my ideas is to store as little text info from the request as possible, primarily because string manipulation in C is so obtuse and easy to introduce bugs into. Instead I've started on a line-by-line request processor, which only reads data from headers where it is relevant to this server's operation. I've finished handling of GET requests, but I need to do more to handle paths which are not specific, and also remove the query string info – right now the server just tries to access the requested path as a file relative to the current working directory.

25th October 2017

Now I've added some server configuration which can be changed using command-line flags. The server path is now parsed more comprehensively, the query string info is removed and the URL is decoded (ie. percentage signs are replaced with the escaped character).

The server also now returns a 404 when the requested file doesn't exist, rather than just killing the request. Additionally, the server now checks if the path is a directory, and if so `/index.html` is appended to the path. The MIME type is now also deduced from the path (rather than just serving a fixed `text/html` response.

One challenge I did have today was to do with debugging. I treated one enum as a bitflag and tried to do bitwise operations on it (which turned out to be unnecessary anyway) when I had not explicitly set the enum values to be powers of two, so some of the bitwise operations were producing unexpected results. However I did become more familiar with the use of gdb as a result of this so I got something out of it.

28th October 2017

The server can now handle different request methods other than GET, as well as correctly serving a 405 Method Not Allowed if the method is not supported. Additionally, the HTTP version string (eg. HTTP/1.1) is parsed now, however it is not used other than to send the same version back, as there is nothing in the server right now which is specific to the HTTP version.

29th October 2017

Wrote the documentation file. Also added DEFLATE support which took surprisingly little time due to the way I have implemented it. The server now also frees the memory used by each client thread upon the response being completed, as I realised it was entirely pointless to wait until the server was terminated, as the OS frees the memory on termination anyway so it was doing nothing to stop memory leakage.

Finally the server now also does some rudimentary checks to ensure the path you request is not going to escape the HTTP root by returning a 400 Bad Request if the path contains the `..` directory-up metadirectory.

Reflection

In hindsight, in my attempt to avoid annoying string manipulation, I made some areas of the code quite dense and stateful. For example, `rq_parse_path` is quite complex and not very extendable. However, it all works fine and does not use any particularly ugly hacks, so this would only pose a problem if I intended to extend this project into a more functional web server in the future.

I would also actually store the request headers if I were to do this assignment more comprehensively. Right now, however, just reading the necessary data from the headers as they are read is fine, as I am not passing the headers to anything else.

While some areas of the code seem to store information in quite obtuse ways (such as use of bitflags), it did mean that as little memory is allocated as possible. This is done with the intention of making the server leak absolutely no memory; running the server via valgrind should indicate that no memory is ever left unfreed.