A Web Server in C

A web server in C

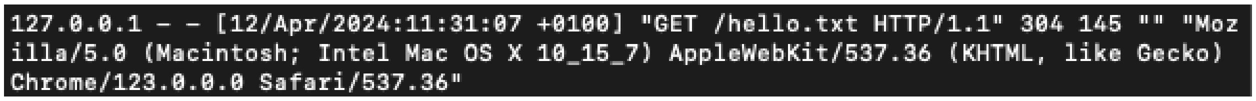
On your server machine, clone the repository `https://github.com/emikulic/darkhttpd` which contains a single-file web server in just under 3000 lines of C. You do not need to understand this code in detail, but it can be educational to try to understand some of what it does internally.

Compile the program either with `make` or simply `gcc darkhttpd.c -o darkhttpd`. The server's job is to serve files within a folder, so make a subfolder called `web` and then run it with `./darkhttpd web --port 8000`. You can stop it again at the end of the exercise with `Control+C`.

You can now experiment with the following:

- Create some files (plain text, HTML, image etc.) in `web/`.
- Access them from your browser with `localhost:8000/FILENAME`, for example `web/hello.txt` would become `localhost:8000/hello.txt`.
- Observe the HTTP headers in your browser's developer tools, and the server logs printed in the server terminal.

Server logs:



```
127.0.0.1 - - [12/Apr/2024:11:31:07 +0100] "GET /hello.txt HTTP/1.1" 304 145 "" "Moz
illa/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.0.0 Safari/537.36"
```

This particular server is written so that if you try and access a folder instead of a file, for example just `localhost:8000` which has the implicit path `/` which maps to the `web` folder, then it shows a list of files in this folder as clickable links.

Pay particular attention to how the `Content-type` header is sent depending on the file extension. From the browser's point of view, that header is what counts: if you send a file with `.html` extension but set `Content-type: text/plain`, the browser would not interpret it as HTML. This makes it possible to serve URLs with no "extension" at all like `/students` for a database application, and still have the browser understand what to do.

From the server's point of view, this server (like most other servers that simply serve files from a folder) has chosen to use a file's extension to determine what `Content-type` to send; this is implemented in the `default_extension_map` starting at line 333 of the source file.

You can try this out for yourself if you want: make a HTML file called `one.html5` in the web directory and access it with the browser. (`.html5` is not an official extension, it's something I just made up. The correct one to use for proper websites is `.html`.) The server won't send a `Content-type` header at all, since it doesn't know this extension, so the browser will either display it as a plain text file or download the file instead of showing it.

Edit the map in the source file and change the entry for `text/html` to read `" html htm html5"`, keeping the space at the start of the string. Recompile the server.

Rename the file to `two.html5` (I'll explain why in a second) and restart the server and try and open the file in the browser. This time, the file will display as a HTML page.

Why did you have to rename the file? Because the browser will otherwise try and be clever if it thinks you're trying to access a file that you've just downloaded already. The server sends a `Last-modified` header, and if the browser notices you're asking for a file you have just downloaded, with the same timestamp as the HTTP header indicates, then your browser might ignore the rest of the request and not see that another header has changed. Changing the file name forces the browser to look again, as it thinks it's a different file now. Deleting the downloaded file, or editing the file on the server, would both have the same effect.

There is a moral here: if you're doing web development, and you are trying to find out why a change you made isn't showing in the browser (even if you refresh the page), it might be a cache problem. On the network tab of the developer tools window, there should be a checkbox to "disable cache" which you might want to turn on in this case and reload the page again, to exclude the cache as the possible source of the problem.