CO324 Lab 07

HTTP server implementation

This lab is based on the given HTTP server. We will learn the usage of reader-writer model in real life scenario of thread pools.

HTTP servers

HTTP (Hypertext Transfer Protocol) is the basic protocol used to communicate data around the Internet. HTTP servers (or web servers) are the servers that offer the service of processing HTTP requests and responding.

HTTP protocol

You already have learned about the HTTP protocol. We will just look at an example here. As you know when you open your browser and type www.google.com/index.html on the address bar, the browser sends a HTTP GET request to the HTTP server at www.google.com requesting index.html page. When the HTTP server receives the request, it responds with the content of the index.html file that it has. In addition to the content of the index file, the response also contains a 'status' number, indicating the status of the response.

The detailed functionality of the given HTTP is as follows:

When a client is connected the server expects a request in the following format from the client:

"GET <file_path> HTTP/1.1\r\n"

For an example:

GET /home/index.html HTTP/1.1\r\n

requests the /home/index.html file. For now assume there are no spaces in the file path.

The response of the server could be one of the follows:

"HTTP/1.1 403 forbidden\r\n" – the file requested is not allowed to be read.

"HTTP/1.1 404 Not found\r\n" – the file requested does not exist.

"HTTP/1.0 200 OK\r\n\r\n <text in the file>" - everything is fine. Sending the file

The implementation of the above specification is provided.

Exercises:

Read and understand the HTTP server implementation.

1. Fill in the missing code in run() method in the given code.
2. test the code with wget as follows:

   ```
   wget -d 'http://localhost:8080/<file_path>'
   ```

   where <file_path> is the path for your file. The -d flag provides debug information and will show you the HTTP headers used in communication. Set the file path such that:

   1. It requests a non-existing file
   2. it requests a file that does not have permission

   and see if the output is correct.

3. Now test it with your browser. Open your favorite web browser and type the following in the address bar:

http://localhost:8080/<file_path>

This should display your file in the browser.

4. A real http response will have more headers than given in the example. For instance, it might look like:

```
HTTP/1.0 200 OK

Content-Type: text/html

Content-Length: 1354

<content>
```

Where content length is the length of the file.

Add these headers to the given implementation.

5. Test the new server with wget. Do you see any difference in wget output?