COMP 307 Assignment 1

Introduction

Create a web server using Python 3 that serves resources in a local directory (document root) using the HTTP 1.1 protocol.

Suppose we run your web server using the following command:

python3 webserver.py 192.168.0.100 8888 /path/to/document/root

The first two arguments are the IP and port your server should listen on for incoming HTTP requests. The third argument is the document root. Your web server should accept any valid IP, port, and path.

Assume your web server is run with the argument in the example command above, and suppose a browser requests a resource from your webserver with the URL:

http://192.168.0.100:8888/images/cat.jpg

Then, your web server should look for the following file on the local file system:

/path/to/document/root/images/cat.jpg

Once found, it should construct a correct HTTP response message with the correct headers specifying the media type and send it to the browser.

Notice how the part of the URL after the IP and port is appended to the document root in order to locate the file. This is how web servers typically work.

As another example, if the browser requests:

http://192.168.0.100:8888/tutorials/python3/sockets.html

Your web server should serve the following file on the local file system:

/path/to/document/root/tutorials/python3/sockets.html

Again, take care to specify the correct media type in the header.

Media Type Support

To keep things simple, you only need to support the following 3 file types: html files, jpg images, and png images. Simply use the file extension to determine the media type, and then

construct the response header accordingly. You can assume that the document root will only contain files of these types.

Special 404 page

If the requested resource is not found inside the document root, send a response with a body that has some simple html that says 404 File not found.

Request Parsing

Your server should handle request messages of any length. Use recv properly as discussed in class to ensure you properly read the whole request message.

Rejecting Firefox

Let us pretend that recently a critical security issue has been found in the Firefox browser, and our web server needs to deny access to any requests coming from Firefox. After parsing each request, use the request headers to determine if the request is coming from Firefox. If it is, respond with a special HTML body that tells the user to switch browser. The 404 page does not apply to Firefox users – they will always get the same response, regardless of what resource is being requested.

Concurrency

You do not need to worry about handling concurrent requests. In other words, you do not need to spawn individual threads for each request as we discussed in the lecture.