CA2: Socket Programming Assignment

Submission Instructions

- CA2 is due on **Sunday, April 14, 2024, at 23:59:59**.
- Submit your homework to Canvas folder 'CA2 submission' as a single zip file containing,
    1. A server file,
    2. A client file,
    3. Three .jpg files as described below,
    4. A brief report (PDF) for your responses to the questions below.
- The zip file should be named "STUDENT-NO_NAME.zip".

Rules

- You are **not** allowed to use any API/libraries to construct the HTTP messages (e.g., Python Request library). This is a socket programming assignment. You have to create sockets and construct messages manually. Most other libraries are okay to use.
- You can use online resources, including software blogs, and (although discouraged) generative AI tools. Appropriately cite the source if you adopted a part of your code from somewhere. Adopting a part of your code or report without appropriate citations is a plagiarism offence.
- Comment your code.
- In your report, only include the responses to the questions asked in the assignment. Lengthy explanations are unnecessary and discouraged.
- Feel free to ask for clarifications after the lectures. E-mail questions may not be attended to. No clarification will be provided after **April 10, 2023.**

Tasks

In this assignment, you will implement a simple HTTP server and a client program that requests objects from the server. You may use any programming language to implement the server and client.

First, create a simple website for a fictional character. Your website must have a block of text, an embedded profile picture in .jpg format, and references to two other pictures in .jpg format. You may use the following template with your own .jpg files and text.

```
<html>
    <head>
        <title>Grogu's Website</title>
    </head>
    <body>
        <header>
            <h1>Grogu's Website</h1>
        </header>
        <img src="photo.jpg" alt="my_photo">
        <h1>About</h1>
        <p>Grogu, colloquially referred to as Baby Yoda…</p>
        <p>Here are some pictures of me: <br>
            <a href="pic1.jpg">Picture 1</a> <br>
            <a href="pic2.jpg">Picture 2</a>
        </p>
    </body>
</html>
```

Opening the HTML file using your browser should display the website you built.

Optional: Add some style to your website. Websites use Cascading Style Sheets (CSS) to enhance their visual appeal. You can define elements (e.g., using) in your HTML, insert the pictures and text under these elements and define their layout in the CSS file.

HTTP Server:

Implement an HTTP server program capable of serving the following over TCP:

a. An HTML file with the name “index.html”.
b. Any .jpg file referenced in the HTML file (create your own files).
c. A dummy favicon.ico file*.

Your server program should be able to parse the incoming HTTP request for the requested object name and reply with an HTTP response message that contains the requested object.

Your server program should also print the incoming HTTP requests to the console/terminal.

Use the standard loopback interface (localhost) address for the server's address. Also, use a nonstandard port number ([PORT_NUMBER]) instead of using the standard port of HTTP.

Next, run your server program and type http://localhost:[PORT_NUMBER]/ into the URL field of your browser. What do you see? Take a screenshot of the browser window and paste it into your report. Observe the terminal output for the incoming HTTP requests. Copy the screenshot of the terminal to your report.

Finally, click on both .jpg images (your own versions of pic1.jpg and pic2.jpg) to fully test your server program.

HINT: .jpg files are in binary format, therefore, do not require encoding, unlike text.

*Typically, browsers request a "favicon.ico" object. This is the small icon found in the tabs of browsers. Your server program should also serve this file when requested. Create a dummy file with the same name in your project directory to meet this request.

HTTP Client:
So far, you have used your browser as the HTTP client. Now, implement an HTTP client program to perform the following sequentially.

a. Request the base HTML file from a server,
b. Parse the HTML for references to .jpg objects,
c. Request the .jpg objects referenced in the HTML from the server.

You can use either persistent or non-persistent (the request for every item creates a new TCP connection) HTTP connections.

Your client program should be able to request .jpg files with any filename. You should find a way to parse the HTML, find references to the images, and scrape the filenames of .jpg objects before sending HTTP requests for them.

Run your server and client programs simultaneously. Save the screenshot of the terminal output from your client and server programs.

Make sure that your code works with any .jpg filenames. Test your server and client programs with different .jpg filenames.

Improvements:
Besides using persistent HTTP, what improvement can you make on your server and client programs for your client program to download images faster? Explain in your report. You do **not** have to implement it.