Programming Assignment 1

Computer Networks

10 NOV 2024

Saifullah Mousaad

Ahmed Ashraf

1 Multi-threaded Web Server

1.1 Purpose of the Server

The server is a simple HTTP server designed to handle both GET and POST requests. It serves static files stored in a directory called root and logs information about each request, including details such as the client's IP address, port, method, and connection type.

1.2 Features and Capabilities

- **Supports GET Requests**: Serves files from the root directory based on the requested file path.
- **Supports POST Requests**: Saves incoming POST data to a file within the root directory with the specified content type.
- **Multithreaded**: Each connection is handled in a separate thread, allowing for concurrent handling of multiple connections.
- **Content-Type Handling**: Determines appropriate content type based on file extension and serves text, image, and application types.
- **Error Handling**: If a requested file is not found, the server returns a 404 Not Found response along with a default page_not_found.html page.

1.3 Components and Functions

1. **get_content_type**
    a. Determines the correct MIME (Multipurpose Internet Mail Extensions) type based on the file extension.
    b. Supports text, image, and application content types, with specific handling for common file extensions (e.g., html, css, jpg, png).
2. **prepare_response**
    a. Builds and sends HTTP responses with appropriate headers, including:
        i. **Status Code**: 200 OK or 404 Not Found.
        ii. **Date**: Set to the current GMT time.

iii. **Server**: Identifies the server as "Thunder/1.0.0 (WindowsOS)".
iv. **Content-Length**: Indicates the length of the response content.
v. **Content-Type**: Derived from the requested file's extension.

b. For images, headers and content are sent separately to handle binary data properly.

3. **parse_message**
   a. Parses incoming requests, identifies the HTTP method (GET or POST), and extracts headers such as Connection and Content-Type.
4. **process_message**
   a. **For GET requests**: Attempts to locate the file in root and serves it. If the file is not found, responds with a 404 error.
   b. **For POST requests**: Extracts and saves the request body data to a file in root, appending the appropriate extension based on Content-Type.
   c. Logs each request in a separate thread.
5. **Main Loop and Client Connections**
   a. Creates a TCP socket bound to the specified port.
   b. Listens for incoming client connections with a maximum queue of 5.
   c. Spawns a new thread to handle each connection, ensuring that multiple clients can be served concurrently.

2 HTTP Web Client

- The Client is developed to handle GET and POST requests with HTML, text and Images.

2.1 Flow of the program:

- Receive an IP address and port number as arguments to the command line
- Establish a connection with the server has the specified IP and port number entered
- Ask for a input file contains command to perform with the server
- Commands are executed back-to-back depending on the method (GET - POST) and the other parameters (File Type,...,etc

2.2 Methods and Functionalities:

1. **send_request**
    a. The executor of each command, take the request as an input
    b. Does the main steps represented in calling the following functions below in an ordered manner
    c. Handles receiving large data and it's open for the user to POST a file with any size he wants, the buffering is done dynamically based on the size of the file.
    d. After receiving responses, based upon the method (GET, POST) the function either only shows the response message in case of POST or displays the response message including the body then creates a file for saving the data received and stores it in depending on the type (Image-text).
2. **parse_command**
    a. Handles the splitting of the command line from the input file to retrieve the required information setting the port to 80 by default if the command ignores the port
3. **file_type**
    a. This function generates Content_Type header in its standard form to send with the request.
4. **read_posted_file**
    a. Takes the path of the file requested to be posted on the server and returns the data within the file and handles the error resulting from the absence of the file mentioned.
5. **form_request**
    a. responsible for forming requests based on the method (GET, POST) while putting any necessary header lines such as Content_Type and Content_Length. The requests must follow the standard format of HTTP request.

3 HTTP 1.1

1. The server by default runs persistent connections between clients and servers.
2. As Required in Part 3 of the assignment, Pipelining is considered as each client gets one thread to serve him alone, hence the server can serve many clients at the same time.
3. A timeout consideration is applied also, i.e each client has a limited amount of time for inactivity before the connection will be lost (aborted by the server). The Heuristic function is a constant number, in our case 20, divided by the number of current clients using the server at this time. Of course, the number of concurrent clients changes with each new client so clients get their allowed timeout dynamically.
4. Heuristic Function = $\frac{20}{Number\ of\ Currently\ Active\ Clients}$
5. The timeout duration of the server itself is put to None using "**server_socket.settimeout(None)**" to ensure that the server is always on.
6. For the clients, *settimeout* method is put before each client gets his service with the existence of except "**TimeoutError**" to catch the client when its connection is closed by the server to handle what happens after, which is terminating his program.

4 Bonus

3.1 Test With Browser

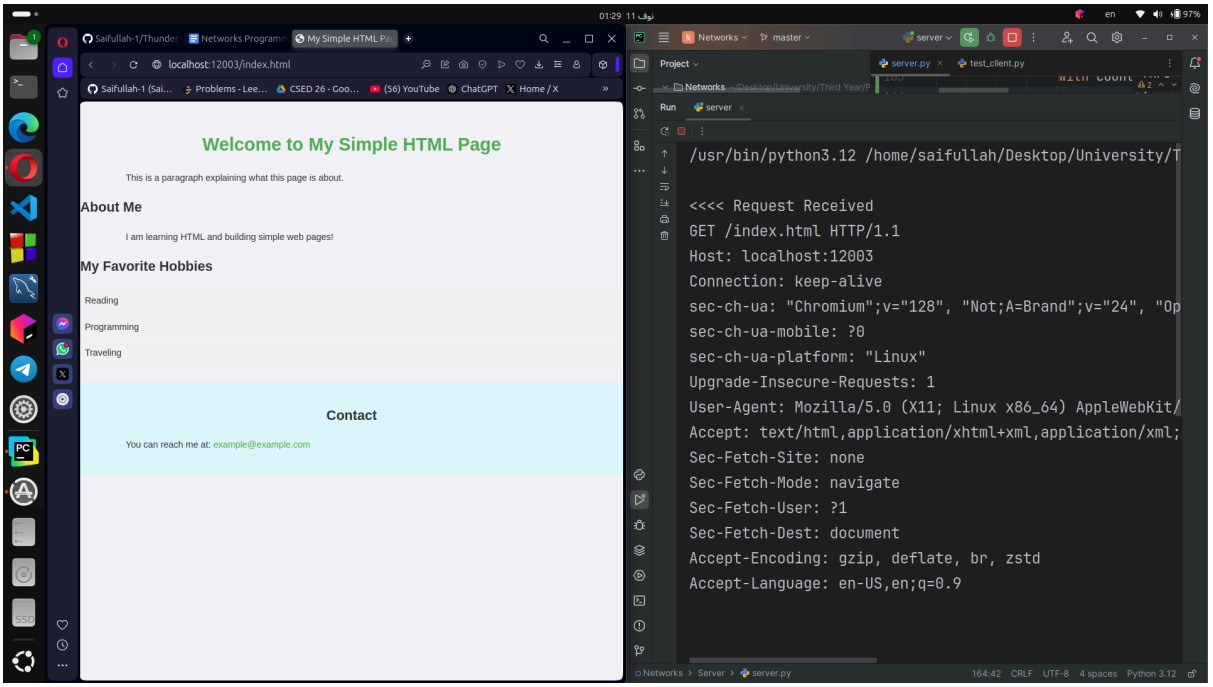

Figure 1: Test with Opera Browser

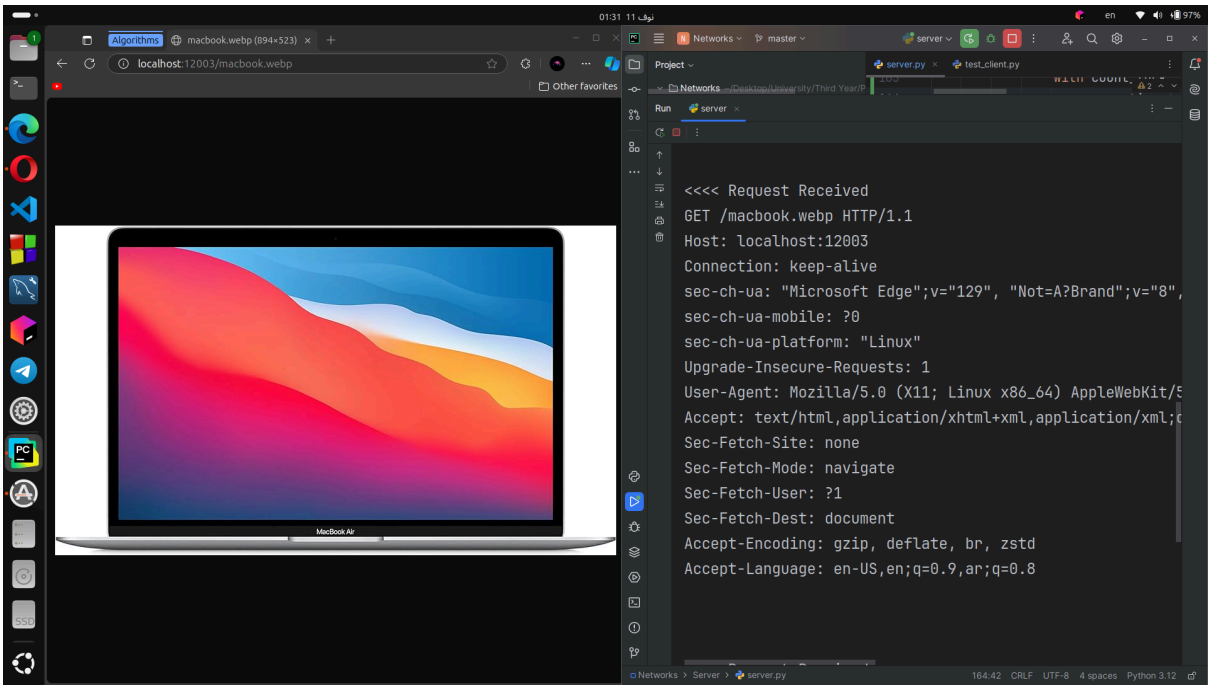

Figure 2: Test with Microsoft Edge Browser

3.2 Performance Evaluation

Figure 1: Performance Evaluation

5 Source Code

https://github.com/Saifullah-1/HTTP-Client-Ser