Name : Mintara Gabriel Sean

Student ID : 32229173

Documentation Homework Making Your Own Server

Link GitHub : https://github.com/InfinityFross/32229173_WebServer

INTRODUCTION

In today's digital landscape, cloud computing has transformed how we engage with technology which provides scalable solutions to businesses and individuals. However, beneath this convenience lies the complex world of networking where data flows across the internet. Moving from the abstract concept of cloud computing to the practical reality of low-level socket programming offers a unique opportunity to delve deeper into networking fundamentals. In my opinion, creating my own server using low-level socket programming is more than simply a technical exercise, but it is a journey of understanding and skill development. It helps me to learn the basics of networking firsthand which empowers me to provide solutions to specific needs. This initiative is personally noteworthy since it reflects my passion and enthusiasm for exploring new technology and my commitment to continual learning and growth.

CONCEPTS/BACKGROUND

I am not really familiar with creating my own server using low-level socket programming APIs because of my past learning of the introduction of cloud infrastructure course at my home university. However, my prior experience with Python and HTML provided a solid foundation to leverage my skills to navigate the complexities of networking. In essence, the server-client model is the foundation of network communication, wherein servers await incoming connections from clients and respond to their requests. This fundamental idea applies to web servers which are specialized servers that handle and process HTTP requests and deliver web content to clients. Server sockets listen for incoming connections, whereas client sockets initiate connections to servers. This basic understanding of networking theory laid the framework for my exploration into creating a server from scratch which bridges the gap between theory and practical application in the field of networking.

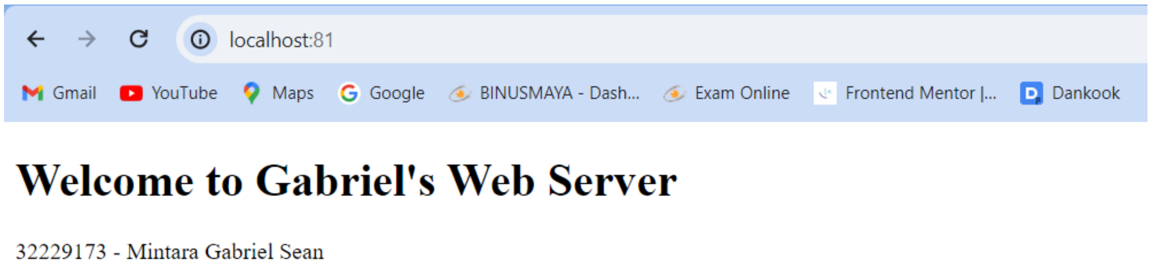
DESIGN

I plan to make a simple web page which consists of a server in a Python file and an HTML file (index). In the Python file, the first thing I do is import necessary modules, such as socket for socket programming and threading for handling multiple clients concurrently. After that, use the “handle_client” function to manage each client connection. It receives a client socket as an argument, reads the request from the client, processes it, generates an appropriate response, and sends it back to the client. The main function will be the entry point of the program which creates a socket and binding it to a specified address (localhost) and port 80, then listening for incoming connections. When a client connects, it creates a new thread to handle that client's request. Inside the handle_client function, the incoming HTTP request is parsed. If it is a GET request, it checks the path. If the path is “/”, it reads the content of index.html and sends it back to the client with a 200 OK response. If the path is something else, it responds with a 404 Not Found status code. For any other request method other than GET, it responds with 501 Not Implemented. For the response handling, the response is created with appropriate HTTP headers and content based on the request and then sent back to the client via the client socket. I also use threading to handle several client connections at the same time. Each client connection is processed in a separate thread which allows the server to handle and manage multiple clients simultaneously without blocking.

IMPLEMENTATION

Here are the steps that I have taken to create my own server:

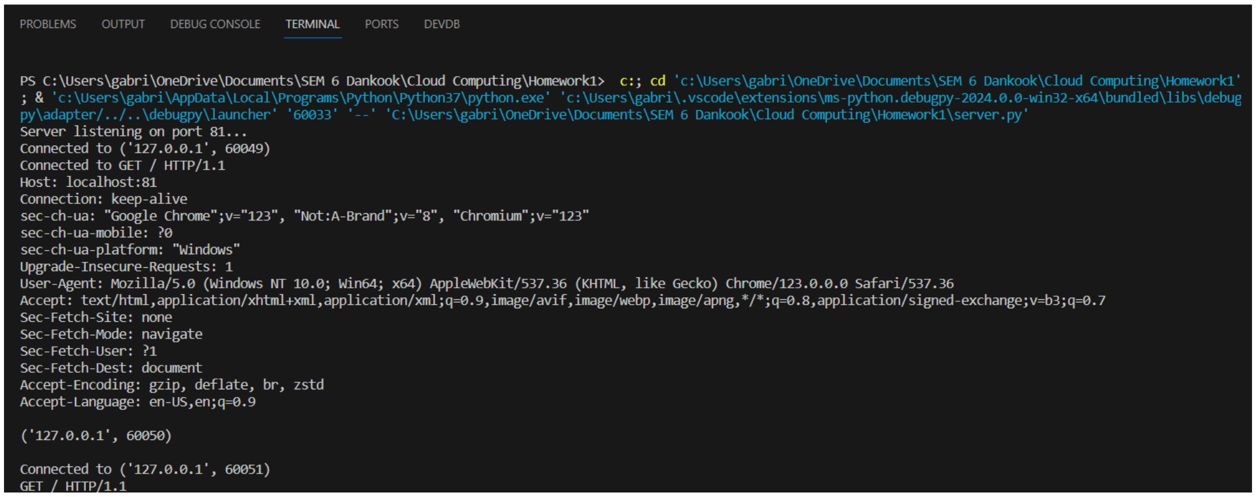
1. Prepare the HTML file which in my case I named it “index.html” to display the website page of my web server
2. Create a Python file for the HTTP server file using low-level socket programming
3. Run the server from Visual Studio Code
4. Check the website page using “localhost:80” (but I changed to localhost:81)



Welcome to Gabriel's Web Server

32229173 - Mintara Gabriel Sean

5. Check the information log in the terminal

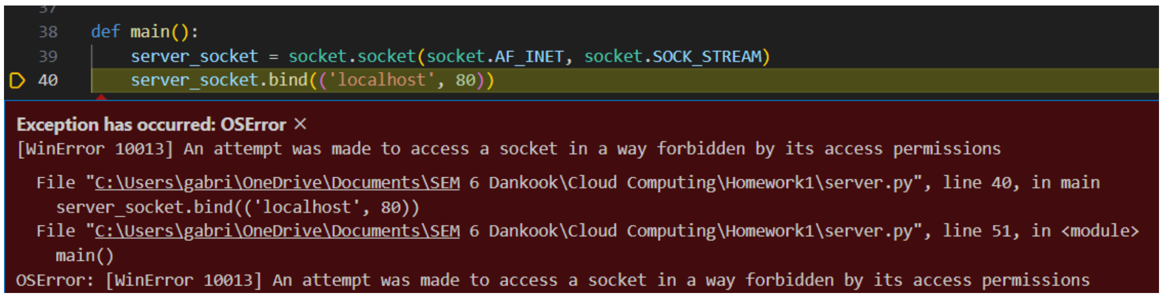


```
PS C:\Users\gabri\OneDrive\Documents\SEM 6 Dankook\Cloud Computing\Homework1>  c:; cd 'c:\Users\gabri\OneDrive\Documents\SEM 6 Dankook\Cloud Computing\Homework1'
; & 'c:\Users\gabri\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\gabri\.vscode\extensions\ms-python.debugpy-2024.0.0-win32-x64\bundled\libs\debug
py\adapter/../..\debugpy\launcher' '60033' '--' 'C:\Users\gabri\OneDrive\Documents\SEM 6 Dankook\Cloud Computing\Homework1\server.py'
Server listening on port 81...
Connected to ('127.0.0.1', 60049)
Connected to GET / HTTP/1.1
Host: localhost:81
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="123", "Not:A-Brand";v="8", "Chromium";v="123"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

('127.0.0.1', 60050)

Connected to ('127.0.0.1', 60051)
GET / HTTP/1.1
```

The problem I experienced after trying to run the code was an error appeared where I could not access port 80 and was required to change the port to a higher number.



```
def main():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 80))
```

```
Exception has occurred: OSError ×
[WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissions
  File "C:\Users\gabri\OneDrive\Documents\SEM 6 Dankook\Cloud Computing\Homework1\server.py", line 40, in main
    server_socket.bind(('localhost', 80))
  File "C:\Users\gabri\OneDrive\Documents\SEM 6 Dankook\Cloud Computing\Homework1\server.py", line 51, in <module>
    main()
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissions
```

I tried to fix the problem, but it still gives some errors. Therefore, I decided to change the port to a number higher than 80 which is port 81.

EVALUATION

After I had done the implementation of the web server, I had some evaluation point:

1. The code is easier to understand and maintain because there are separate functions for handling client connections (handle_client) and the main server loop (main)
2. The code utilizes threading to handle multiple client connections concurrently which is a good approach for scalability. However, threading can lead to potential issues such as resource contention, especially in more complex server applications
3. The server implementation supports basic HTTP request methods (GET) and status codes, but it's limited in functionality. I had some ideas to use the POST request, but I worried about the length of the process and the difficulty of the code so that it adds to the existing errors made me decide to try a simple one
4. The decision to change the port from 80 to 81 due to the error that constantly occurred

CONCLUSION

Creating my own web server using low-level socket programming felt like a leap into the unknown. I had limited understanding of web servers and their processes. However, as I immersed myself in the process, I gained invaluable insights into the process or interaction between server and client sockets. Building the server from scratch provides a hands-on education and experience with HTTP communication, threading, and handling multiple client connections. This experience provided me with not only practical skills, but also a deeper understanding of the mechanisms underlying web development. In summary, the process of creating my web server served as a transformative learning experience that revealed the underlying workings of the web architecture.

REFERENCES

https://www.geeksforgeeks.org/socket-programming-cc/