Name: Zhu Jin Shun
Student ID: 22101071d

COMP 2322
Project
Multi-thread Web Server

Summary of design and implementation of the server program

This program is based on the one learned in Lab 5, with additional functions such as "if modified since" and "last modified" fields, along with other response status functions.

To enable multithreading, I utilized the IP address of the POLYU WLAN, which is 172.16.141.128, and port number 80. By using this IP address, I can access the server not only from my computer but also directly from my phone or iPad which can achieve a Multi-threaded Web server.

I have created a folder named "files" where users can add the files that they want to search for using the server, users can choose to add image and text files. There are two options for searching: using the included Client.py program along with the Server.py file or using a browser.

The input of the user will be read by the program and the program will extract relevant information from the request headers, such as the request type, filename, client IP address, and access time.

When the user inputs a request type on searching the file, the program checks if the file is present in the "files" folder. The program opens a file

specified by the filename variable located in the 'files' directory. It reads the contents of the file into the content variable and then closes the file. Using open (file_path, 'rb')/content = fin.read() can also let the program read text and image files and store the information in content for GET request printing. If it is presented in folder, the server handles two request_type situations: GET and HEAD.

If the user inputs GET/HEAD and the file is found in the "files" folder, the server determines the content type based on the file extension. For example, if the file ends with '.jpeg', '.jpg', '.png', or '.gif', the content type is determined as an image. If it ends with '.html' or '.txt', it is determined as a text file. If the command is GET, the server sends back all the headers, response status, and contents of the file as a response to the client. If the command is HEAD, only the headers and response status are sent back. The server also records the hostname/IP address, access time, requested file name, and response type for each request in the server_log.txt file. The response for each message with an OK status includes the response status, response type, content length, and last modified time of the file.

If the requested file is not found in the "files" folder, the server sends back a 404 NOT FOUND response status.

If a command other than GET or HEAD is entered, such as GEET, the server sends back a 400 BAD REQUEST response status.

For the If-Modified-Since and 304 Not modified field. The function is achieved by user browser access. So, if the user tries to access the file again using the browser with commands like previous access or refreshing the web page, the browser will automatically produce an If-Modified-Since filed indicating the 304 Not modified and If-modified-since for the file execution which will be displayed in server.

For every response message sent to user, the last modified time will be updated for every execution according to the access time of the user.

After each client connection, the server program prompts the user whether they would like to continue for persistent connection and non-persistent connection. If the user enters 'Y', the server continues to receive connections, allowing users to access websites or call client programs to connect again. If the user enters 'N', the server program terminates.

To use each function, the user should input the following command after running the server program:

- GET function: To search for and retrieve the headers and contents of a specific file.

  - Client programs: GET /filename.extension

  - Website: http://IPaddress/filename.extension

- HEAD function: To search for and retrieve only the headers of a specific file.

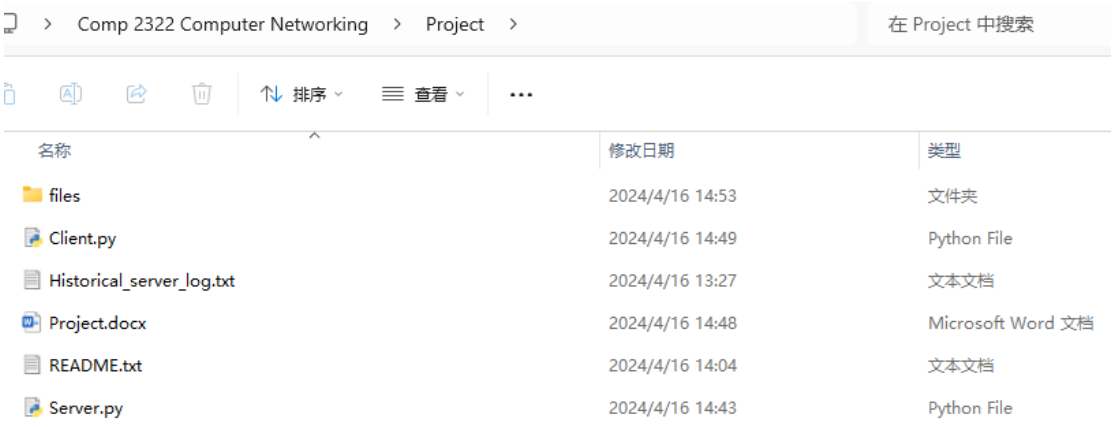
  - Client programs: HEAD /filename.extension

  - Website: http://IPaddress/filename.extension

Demonstration of executing program:

Demonstration of executing program:

To start, download the zip file and extract all the files, open the Server.py and then change the IP address of the SERVER_HOST according to the WLAN you are going to use.

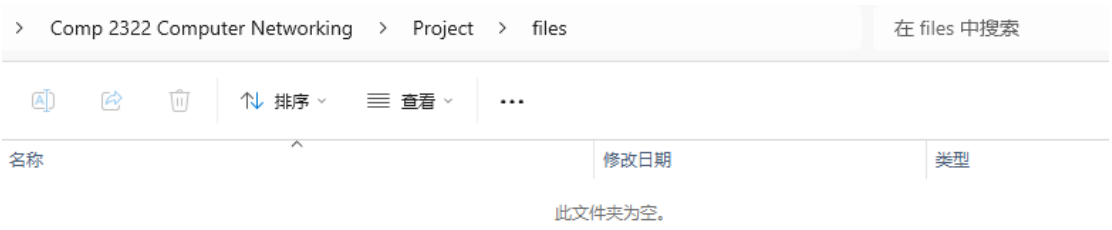


Files inside the Project.zip file

```
Server.py - C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project\Server.py (3.10.7)
File  Edit  Format  Run  Options  Window  Help
import socket
import os
import time
from datetime import datetime, timezone
# REMEMBER TO CHANGE THE SERVER HOST INTO THE IP ADDRESS OF THE WLAN USER IS GOING TO USE
SERVER_HOST = '172.16.141.128'  # The server's host IP address (In this case, is the IP address for POLYU WLAN
SERVER_PORT = 80  # The server's port number
LOG_FILE = 'server_log.txt'  # The name of the log file created containing the records
```

Reminder to change the SERVER_HOST of the Server.py program

Then input and change the IP address for the command when using browser and the SERVER_HOST when using the client.py program. After that, the user can choose to add files to the ‘files’ folder to start checking the functions.



Add files to the ‘files’ folder for searching(Initially it is empty)

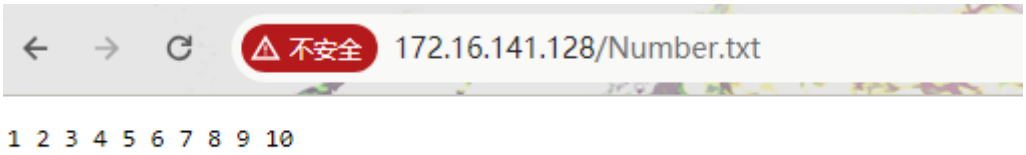
```
*Client.py - C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project\Client.py (3.10.7)*
File  Edit  Format  Run  Options  Window  Help
import socket
# REMEMBER TO CHANGE THE SERVER HOST INTO THE IP ADDRESS OF THE WLAN USER IS GOING TO USE
SERVER_HOST = '172.16.141.128'
SERVER_PORT = 80
```

Reminder to change the SERVER_HOST of the Client.py program

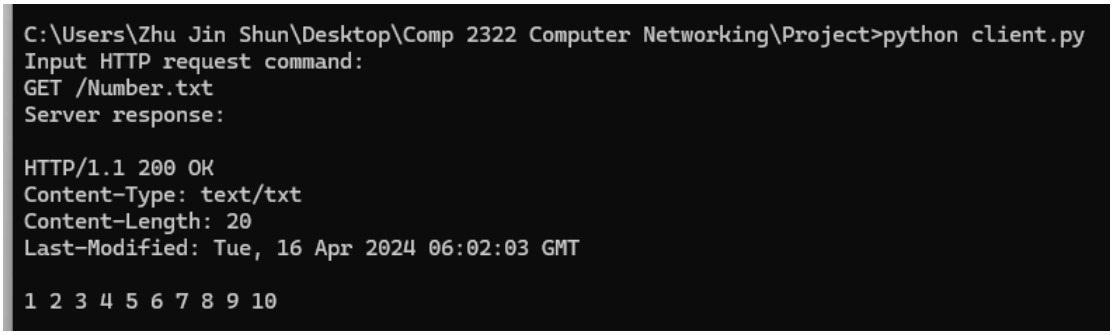
Finally, check the responses when using GET/HEAD commands and the browser access method or client program access method. A new txt file named server_log.txt will be created after the execution for the Server.py

containing the historical information about the client requests and server responses.

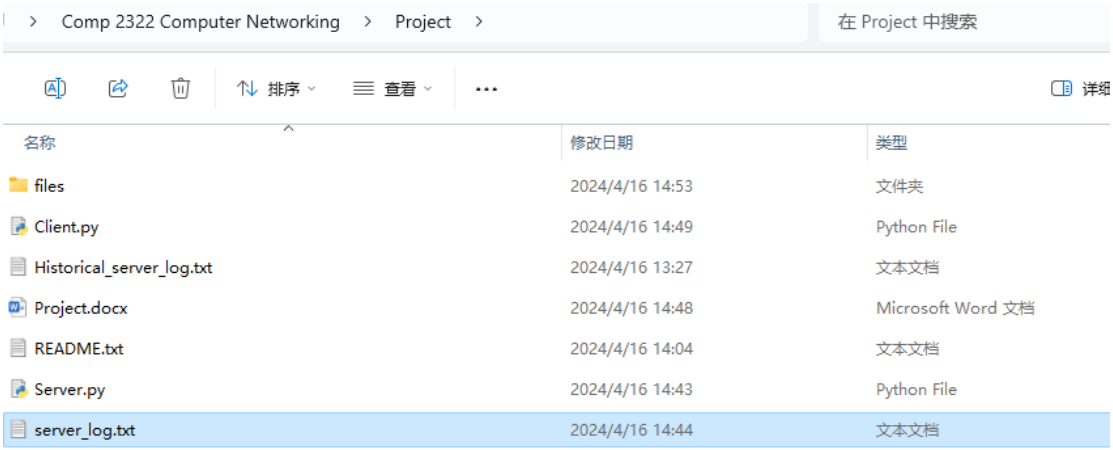
Example of a Number.txt file



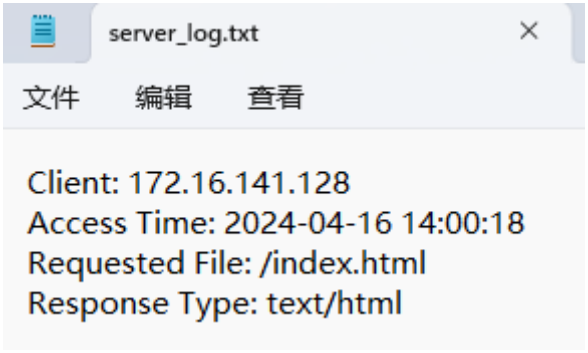
When access on website



When access on client



Server_log.txt file created after execution



Information of the access of GET message and what log file includes

For every client connection, once the client is connected, a message will be printed on server page to indicate the success connection of client and server. The client is executed by entering python Client.py in command line at the location of the client file.

```
Listening on port 80 ...
The client has successfully connected to the server
```

Successful connections of client to server

After every client connection, the server program will ask if the user would like to continue, if user enter Y, then server continues to receive connections and the user can enter the website or run the client program again for execution, if user enters N, then the server program terminates. This function is for handling both HTTP persistent connection (keep-alive) and non-persistent connection (close). Where the user can choose to close the connection with answer N or keep continuing connection with answer Y.

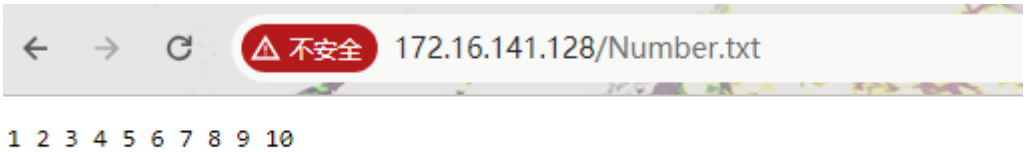
```
Listening on port 80 ...
The client has successfully connected to the server
GET /Number.txt
Do you want to continue? (Y/N): Y
The client has successfully connected to the server
GET /apple.jpg
Do you want to continue? (Y/N): N
The server has successfully closed
```

Example of continue or not function

Screen capturing of results of all functions:

Function 1 GET function for txt files:

Example of a Number.txt file



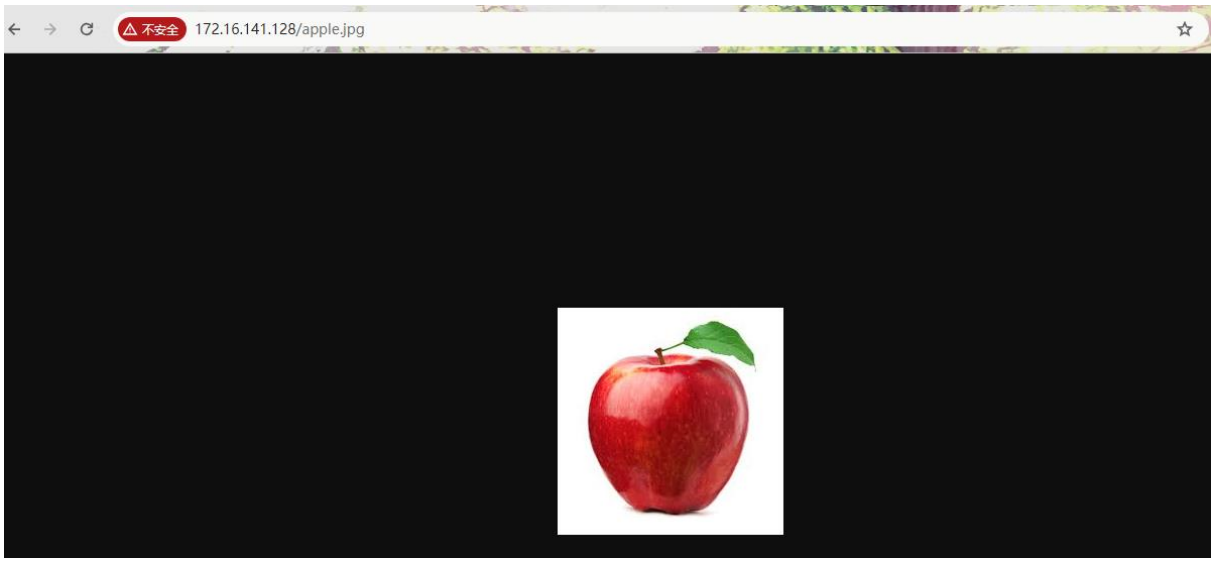
When access on website

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
GET /Number.txt
Server response:

HTTP/1.1 200 OK
Content-Type: text/txt
Content-Length: 20
Last-Modified: 2024-04-16 18:03:47

1 2 3 4 5 6 7 8 9 10
```

When access on client

Function 2 GET function for image files:

Apple.jpg file



When access on website

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
GET /apple.jpg
Server response:

HTTP/1.1 200 OK
Content-Type: image/jpeg
Content-Length: 4727
Last-Modified: 2024-04-16 18:09:02

JFIF
( ?%
!1!%)+...383-7(-.+




        !1AQ"aq2B#bRcr%C$AQ"aq2B#
? 16ïBo% !‘5nÑ\^Yb9ia(].2
c
7

j+K^Tw&jXɛJ
Txgj;OhEyTNo%{Y$.Pĵ⊖fR^>l'FF>nnnOn-id{yvYz]i6~xB{8K&7ChupW /-LLT*N,Z2}
QQĸ*.Q':qdrt+z$`
U0Z>h,D
@\ЄJ{J*i%Ż/6pn14vweʔvJtfi"K°S}.{
åmJqI7
-9gS*i◆.Bxe?ĥF;z4nKzYZjF$HĔ"-F\N_Ĵc
kasIY
iaM;,M9FQ|.i]U
W|(z.~ʔ249F%OBiE+G
S\W,u'øT\ʻHGM>xűk3hl<Du<-6n`     ;>
o{F^ Fm<        ]SQWy+^&[sʔIќe(
```

When access on client

Function 3 GET function for html files:

```html
<html>
<head>
    <link rel="icon" href="data:,">
    <title>Index</title>
</head>
<body>
    <h1>Welcome to the Kingston's web page.</h1>
    <p>Here's a link to <a href="helloworld.html">hello world</a>.</p>
    <br>
    <p>Here's a link to <a href="apple.html">apple</a>.</p>
    <br>
    <p>Here's a link to <a href="Computer.html">Computer</a>.</p>
    <br>
    <p>Here's a link to <a href="Numbers.html">Numbers</a>.</p>
    <br>
</body>
</html>
```

Index.html file

Welcome to the Kingston's web page.

Here's a link to hello world.

Here's a link to apple.

Here's a link to Computer.

Here's a link to Numbers.

When access on website

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
GET /index.html HTTP:/1.1
Server response:

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 446
Last-Modified: 2024-04-16 18:09:38

<html>
<head>
        <link rel="icon" href="data:,">
        <title>Index</title>
</head>
<body>
    <h1>Welcome to the Kingston's web page.</h1>
    <p>Here's a link to <a href="helloworld.html">hello world</a>.</p>
    <br>
    <p>Here's a link to <a href="apple.html">apple</a>.</p>
    <br>
    <p>Here's a link to <a href="Computer.html">Computer</a>.</p>
    <br>
    <p>Here's a link to <a href="Numbers.html">Numbers</a>.</p>
    <br>
</body>
</html>
```

When access on client

SAME FILES AS ABOVE FOR HEAD COMMAND

Function 4 HEAD function for txt files:

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
HEAD /Number.txt
Server response:

HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 20
Last-Modified: 2024-04-16 18:10:05
```

When access on client

Function 5 HEAD function for image files:

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
HEAD /apple.jpg
Server response:

HTTP/1.1 200 OK
Content-Type: image/jpeg
Content-Length: 4727
Last-Modified: 2024-04-16 18:10:20
```

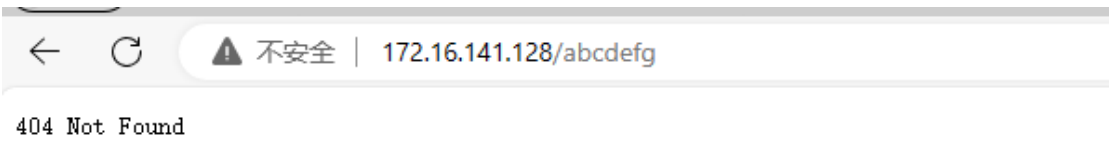
When access on client

Function 6 HEAD function for html files:

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
HEAD /index.html HTTP:/1.1
Server response:

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 446
Last-Modified: 2024-04-16 18:10:36
```

When access on client

Function 7 detecting not presenting file and return 404 not found:



404 Not Found

When access on website

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python client.py
Input HTTP request command:
GET /abcdefg.jpg
Server response:

HTTP/1.1 404 NotFound

404 Not Found
```

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python client.py
Input HTTP request command:
HEAD /abcdefg.txt
Server response:

HTTP/1.1 404 Not Found

404 Not Found
```

When access on client

Function 8 detecting wrong command and return 400 Bad request:

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python client.py
Input HTTP request command:
GEET /Number.txt
Server response:

HTTP/1.1 400 Bad Request

400 Bad Request Not Supported
```

When access on client

Function 9 If modified since when executing same file (Also indicating 304 not modified but still displaying file because using website):

`

When access on website(If modified since displayed in server on second message)

Function 10 changing last-modified time every new execution:

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
GET /Number.txt
Server response:

HTTP/1.1 200 OK
Content-Type: text/txt
Content-Length: 20
Last-Modified: 2024-04-16 18:03:47

1 2 3 4 5 6 7 8 9 10
```

```
C:\Users\Zhu Jin Shun\Desktop\Comp 2322 Computer Networking\Project>python Client.py
Input HTTP request command:
GET /Number.txt
Server response:

HTTP/1.1 200 OK
Content-Type: text/txt
Content-Length: 20
Last-Modified: 2024-04-16 18:06:16

1 2 3 4 5 6 7 8 9 10
```

Last modified time changed for new accessing files

Log file that records the historical information about the client requests and server responses:

Here are the records log file used for testing. (This File is also stored in Historical_server_log.txt file included in the folder)

Note that only response status 200 OK will have responses send back to client about client/access time/requested file/response type and create a record (Client=host/IP address), others will just send back an error message:

Client: 172.16.141.128

Access Time: 2024-04-16 14:00:18

Requested File: /index.html

Response Type: text/html

Client: 172.16.141.128

Access Time: 2024-04-16 14:00:44

Requested File: /apple.jpg

Response Type: image/jpeg

Client: 172.16.141.128

Access Time: 2024-04-16 14:02:10

Requested File: /Number.txt

Response Type: text/txt

Client: 172.16.141.128

Access Time: 2024-04-16 14:09:24

Requested File: /Number.txt

Response Type: text/txt

Client: 172.16.141.128

Access Time: 2024-04-16 18:03:47

Requested File: /Number.txt

Response Type: text/txt

Client: 172.16.141.128

Access Time: 2024-04-16 18:06:16

Requested File: /Number.txt

Response Type: text/txt

Client: 172.16.141.128

Access Time: 2024-04-16 18:09:02

Requested File: /apple.jpg

Response Type: image/jpeg

Client: 172.16.141.128

Access Time: 2024-04-16 18:09:38

Requested File: /index.html

Response Type: text/html

Client: 172.16.141.128

Access Time: 2024-04-16 18:10:20

Requested File: /apple.jpg

Response Type: image/jpeg

Client: 172.16.141.128

Access Time: 2024-04-16 18:10:36

Requested File: /index.html

Response Type: text/html

Client: 172.16.141.128

Access Time: 2024-04-16 18:17:28

Requested File: /Number.txt

Response Type: text/txt