COMP2322 Project Report

Name: Cheng Ho Wing 29 Mar 2022
Student number: 21089537d

Summary of design and implementation of the server program:

My multi-thread HTTP webserver is written in python with the standard socket and threading library. The *HTTPServer.py* has three major function which are: logging, threading class, and main run function.

Main function (run):

When the main function starts, it will first initialize the logger by calling the **Logging** function. After that, it will bind the server IP and port and start to listen to that port for HTTP connection. The default IP is local IP (127.0.0.1), and default port is 80 for HTTP protocol. The port can be changed if user input command line argument as port number, but the IP is fixed to local IP and not changeable. Once a HTTP client initiate the connection, a new thread will be created to handle the client. The created thread is created from the **Threading** class and the client IP, port and socket will be passed to the thread for handling the client.

Logging function (initLogger):

This function will create a logger to write log file named with today's date and other timing information of the logged status.

Threading class (clientThread):

This threading class serve the most important function to the client request. This class is inherited from the python threading library and override the **initialize** and **run** function to serve the client request. The class **initialize** with three parameters (client IP, port and socket) comes from the main connection and one status flag (self.live: default is true), one timer session flag (self.threadLiveTime: default is 5 seconds). Once it created, it will go to **run** function. In **run** function, it will keep listening to the client until timeout for the session or bad request/ file not found, etc.

During the listening period, the server will decode the client's HTTP request and validate the request by the header value (e.g., not accept the POST request). If the request is not valid, the thread will response 400 Bad request, end the connection and kill the thread. If the request is valid, it will check the requested file/ type and check the server local file status then response the client with 200 OK/ 404 not found/ 304 not modified depends on file status compared to the client and server status. Also, it will extend the session (lifetime) of server and client upon request from client ()

The request status is got from **getRequestedConnectionStatus** function inside the **Threading** class.

This function will accept the client request string as parameter. After is get the client request string, it will decode it to find the key string "Connection" and "Keep-Alive: ". Once it gets the requested status, it will return the connection status Str, keepAliveTime Str and keepAliveTime.

The local file status is got from **getLocalFileStatus** function inside the **Threading** class.

This function will get the local requested file size, last modified time and file type (text/image file). Once it gets the requested status, it will return the lastModifiedTimeStr, contentLengthStr and fileTypeStr.

Other files beside *HTTPServer.py* attached in the zip file together with the program and report:

abc.txt

Testing text file.

test.png

Testing image file.

index.html

Testing html file.

TCPClient.py

Simulation client request program (for bad requset).

readme.md

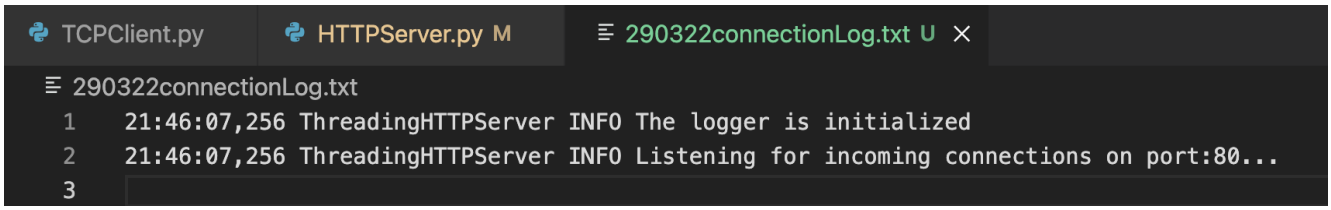
Readme file of instruction to run the program.

290322connectionLog.txt

Connection log created when running the server and used in this report.

Demonstration of executing program & Screen capturing of results of all functions

When the server started, the server will first init the logger by **initLogger**() function and run (port=80) function to open port to listen for requests.



When a client request is initiated, then the server will first get the request sent from browser (clientRequestStr) and determine the response message (the sample response on browser & log status)

Welcome to the index.html web page.

```
21:46:39,728 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63100
21:46:39,729 ThreadingHTTPServer INFO clientRequestStr is:
GET / HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, lik
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7


21:46:39,730 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
21:46:39,732 ThreadingHTTPServer INFO Response headerStr is:
HTTP/1.1 200 OK
Server: Python 2.7
Last-Modified: Tue, 29 Mar 2022 20:40:02 GMT
Content-Length: 145
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

```

Browser request

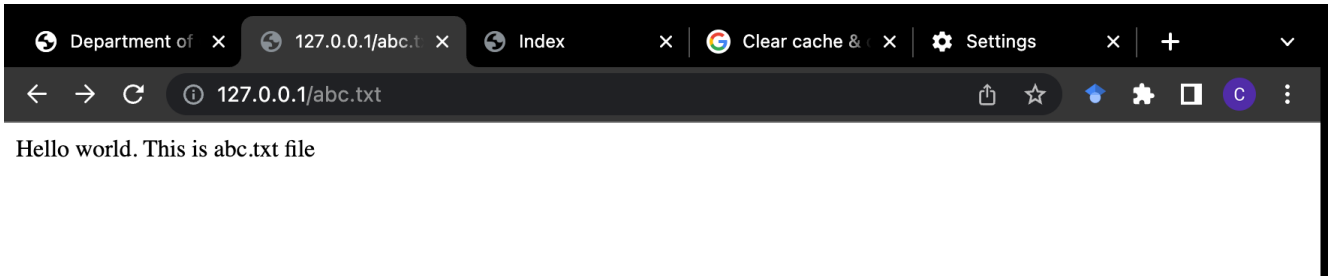
Server response header

If the user send the same request again, the server will get the last modified time and only response the header, the same sitution happen in log below because I have tested the program serval time in browser without clear the browsing history (cached files).

```
21:50:07,753 ThreadingHTTPServer INFO clientRequestStr is:
GET / HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gec
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
If-Modified-Since: Tue, 29 Mar 2022 20:40:02 GMT


21:50:07,754 ThreadingHTTPServer INFO Response headerStr is:
HTTP/1.1 200 OK

```

```
21:57:03,965 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63375
21:57:03,965 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
21:57:03,965 ThreadingHTTPServer INFO clientRequestStr is:
GET /abc.txt HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gec
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
If-Modified-Since: Sat, 19 Mar 2022 17:41:04 GMT


21:57:03,965 ThreadingHTTPServer INFO Response headerStr is:
HTTP/1.1 200 OK

```

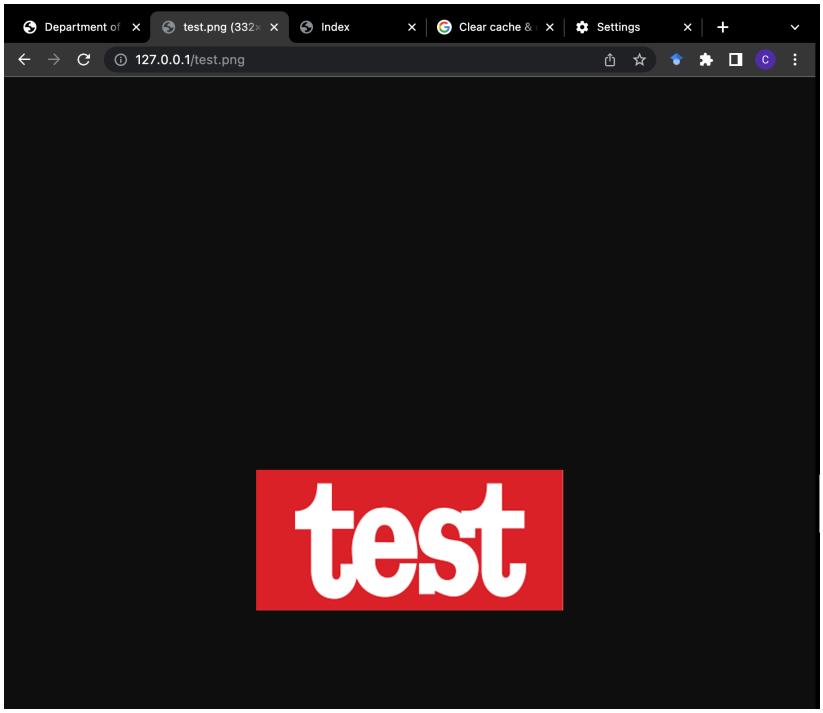
Another type of Request (text file)



After I manualy clear the browser history, the server response normally.

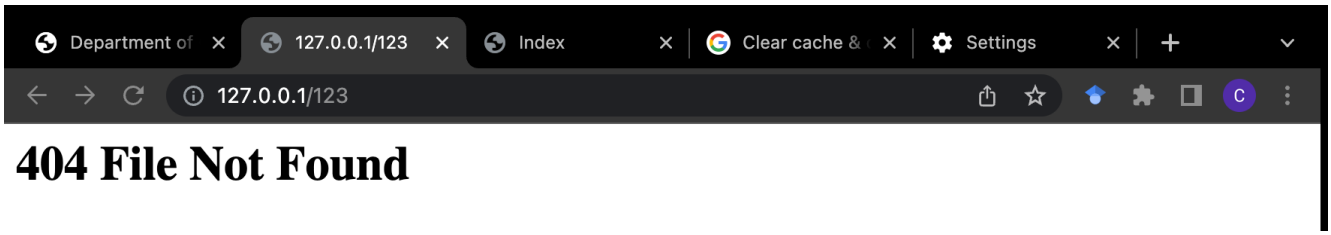
```
22:00:13,134 ThreadingHTTPServer INFO clientRequestStr is:

22:00:13,137 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63467
22:00:13,137 ThreadingHTTPServer INFO clientRequestStr is:
GET /abc.txt HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gec
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7


22:00:13,137 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
22:00:13,138 ThreadingHTTPServer INFO Response headerStr is:
HTTP/1.1 200 OK
Server: Python 2.7
Last-Modified: Sat, 19 Mar 2022 17:41:04 GMT
Content-Length: 33
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Another type of Request (image file)




```
22:07:02,163 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63654
22:07:02,163 ThreadingHTTPServer INFO clientRequestStr is:
GET /test.png HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Ge
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7


22:07:02,163 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
22:07:02,164 ThreadingHTTPServer INFO Response headerStr is:
HTTP/1.1 200 OK
Server: Python 2.7
Last-Modified: Sat, 12 Mar 2022 19:50:26 GMT
Accept-Ranges: bytes
Content-Length: 3408
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: image/png
```

Another type of Request (not existing file)




```
22:08:43,178 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63703
22:08:43,178 ThreadingHTTPServer INFO clientRequestStr is:
GET /123 HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7


22:08:43,179 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
22:08:43,179 ThreadingHTTPServer INFO HTTP/1.1 404 Not Found

<html><body><h1>404 File Not Found</h1></body></html>
22:08:43,179 ThreadingHTTPServer INFO Disconnected to client ..., killing thread for: 127.0.0.1:63703
```

Another type of request (POST) bad request simulation

TCPClient.py × HTTPServer.py M 290322connectionLog.txt U

TCPClient.py > ...

```python
from socket import *
serverName = '127.0.0.1'
serverPort = 80
serverPath = "/bookstore"
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))

request = "POST /test/ HTTP/1.1\r\n\r\n"
#getRequest = "GET / HTTP/1.1\r\n" + serverName + ":"+ str(serverPort) + serverPath  + \
#            + "\r\n\r\n"
clientSocket.send(request.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server (last):', modifiedSentence.decode())
```

```
[Running] python -u "/Users/howingcheng/Documents/COMP2322/Project/TCPClient.py"
From Server (last): HTTP/1.1 400 Bad Request

<html><body><h1>400 Bad Request</h1></body></html>

[Done] exited with code=0 in 0.108 seconds
```

Server log

```
22:10:05,619 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63756
22:10:05,620 ThreadingHTTPServer INFO clientRequestStr is:
POST /test/ HTTP/1.1


22:10:05,620 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
22:10:05,620 ThreadingHTTPServer INFO HTTP/1.1 400 Bad Request

<html><body><h1>400 Bad Request</h1></body></html>
22:10:05,620 ThreadingHTTPServer INFO Disconnected to client ..., killing thread for: 127.0.0.1:63756
```

References:

https://www.tutorialspoint.com/python/python_multithreading.htm

https://stackoverflow.com/questions/17453212/multi-threaded-tcp-server-in-python

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

https://reqbin.com/Article/HttpHead

https://docs.python.org/3/library/socket.html

https://stackoverflow.com/questions/57882042/keep-tcp-socket-connection-alive-and-read-write-coordination

https://stackoverflow.com/questions/10847157/handling-if-modified-since-header-in-a-php-script

https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types