HW#1_SocketProgramming Report

B09901080 電機三 吳宣逸

■ p1 – Socket Programming – TCP

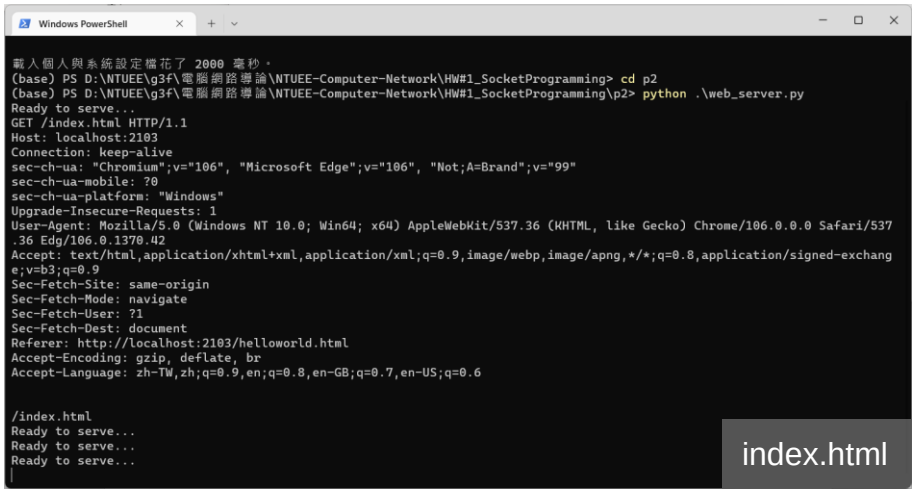
- ✓ Default (HOST, PORT) = (‘127.0.0.1’, 2103)

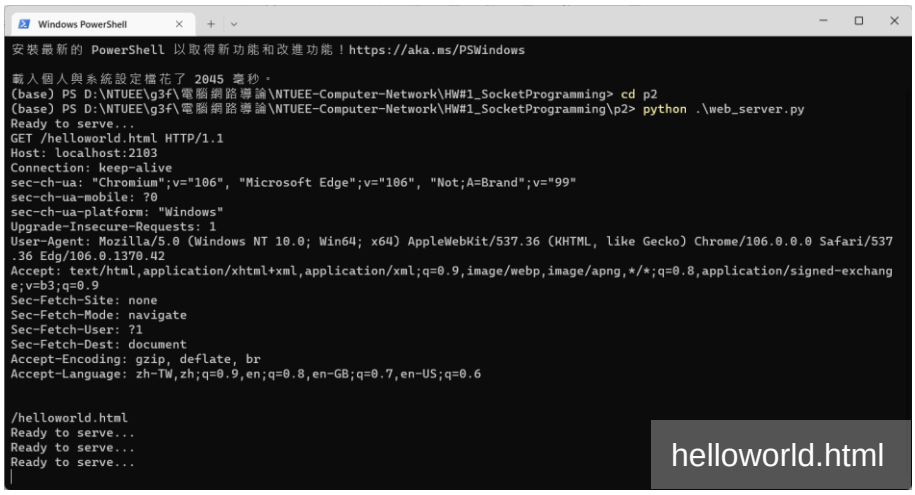
```
# HOST, PORT = '140.112.42.104', 7777  # TA's server
HOST, PORT = '127.0.0.1', 2103  # localhost server
```

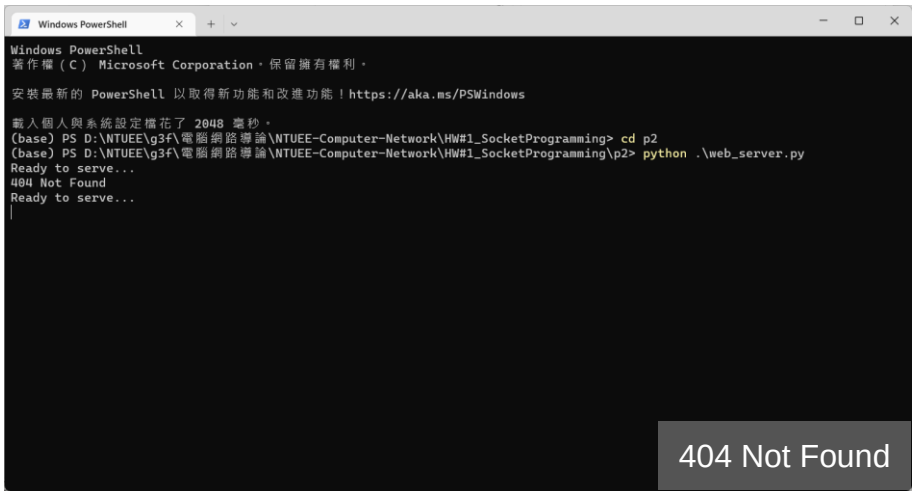
- ✓ No extra function.
- ✓ Read data from p1_testcase, and write to b09901080_p1_client_log.txt after receiving HTTP response.

■ p2 – Web Server

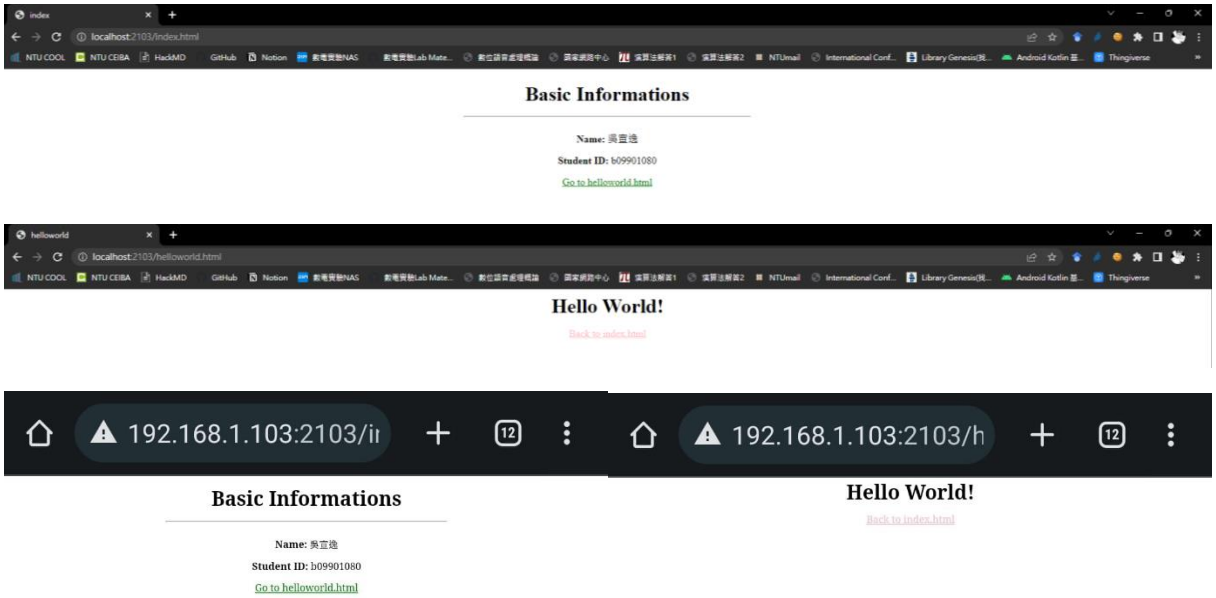
- ✓ Default (HOST, PORT) = (‘127.0.0.1’, 2103)
- ✓ Output Results







- ✓ Try to avoid the request for favicon.ico, but still cause duplicate output of “Ready to **serve…**”. However, it’s not a matter in comparison with trouble from trial to transporting favicon.ico, which is not a HTML file.
- ✓ Access HTML files from local machine and smartphone. Besides, users can access helloworld.html through index.html, and vice versa.



- ✓ Available to change default localhost IP address to current IP address to connect from other devices.

```
# HOST, PORT = str(gethostbyname(gethostname())), 2103    # current IP
HOST, PORT = '127.0.0.1', 2103  # localhost
print(f'(HOST, PORT) = ({HOST}, {PORT})\n')
```

- ✓ Todos
  1. Create socket ServerSocket, bind to localhost:2103, and listen to 10 sockets.

```
ServerSocket = socket(AF_INET, SOCK_STREAM)
HOST, PORT = '127.0.0.1', 2103  # localhost
ServerSocket.bind((HOST, PORT))
ServerSocket.listen(10)
```

  2. Establish connection with client socket ConnectionSocket.

```
ConnectionSocket, Addr = ServerSocket.accept()
```

3. Receive HTTP request (max.4096 characters) from client and decode in UTF-8 codec. Extract the filename of requested html file. For example, split “/index.html” from “GET /index.html HTTP/1.1\r\n”.

```
RecvMessage = ConnectionSocket.recv(4096).decode('utf-8')  # HTTP request
FileName = RecvMessage.split()[1]
```

4. Open and read html files in UTF-8 codec.

```
f = open(os.path.join(os.getcwd(), FileName[1:]), encoding='utf-8')
DataInFile = f.readlines()
f.close()
```

5. Send HTTP response to client with status, header, and data.

```
ConnectionSocket.send('HTTP/1.1 200 OK\r\n'.encode('utf-8'))
ConnectionSocket.send('Content-Type: text/html\r\n\r\n'.encode('utf-8'))
for i in range(0, len(DataInFile)):
    ConnectionSocket.send(DataInFile[i].encode('utf-8'))
ConnectionSocket.send("\r\n".encode('utf-8'))
```

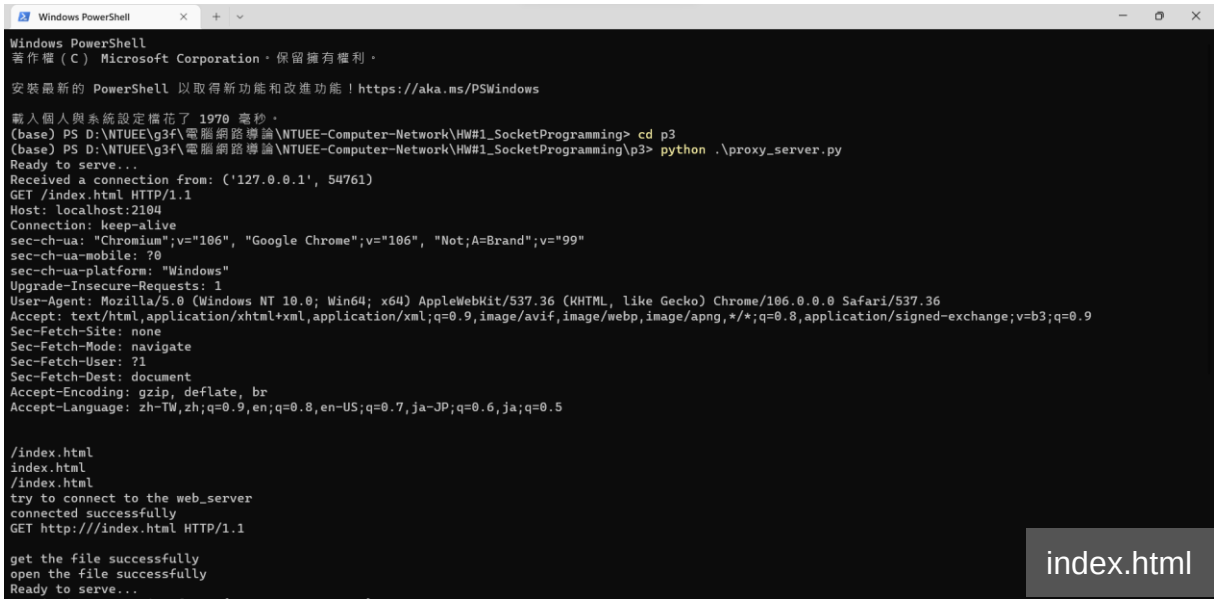
After finishing transmission, close client socket.

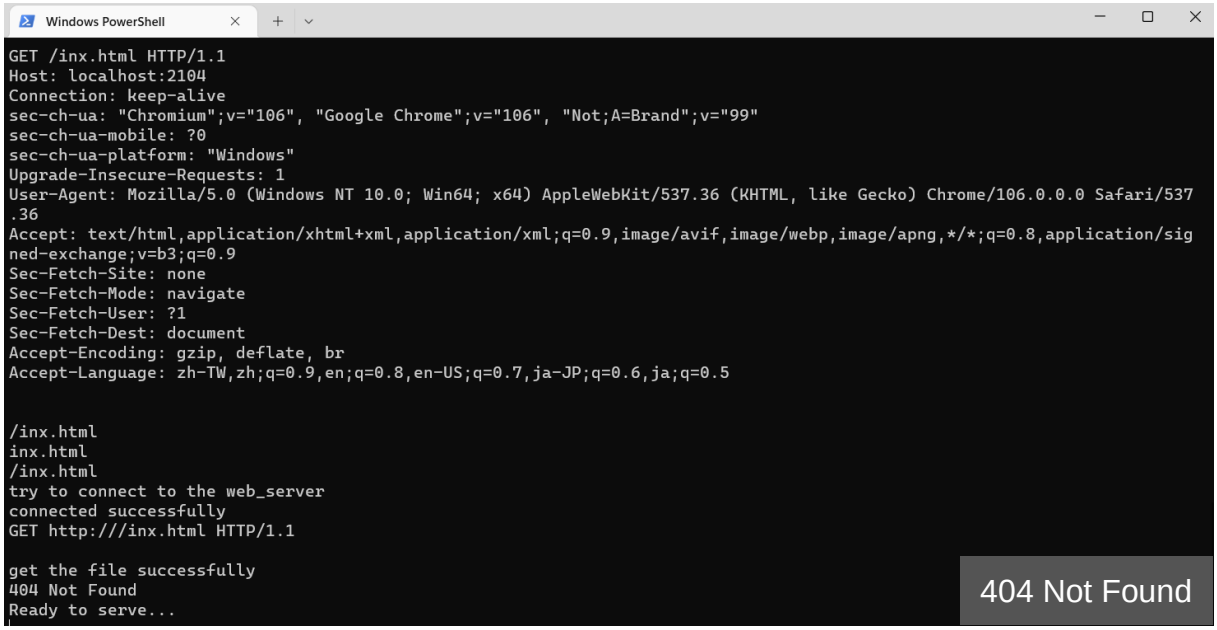
```
ConnectionSocket.close()
```

6. If error occurs while receiving or sending HTTP code and it is has nothing to do with ‘favicon.ico’, then print “404 Not Found” on command, send 404 HTTP status, and close client socket.

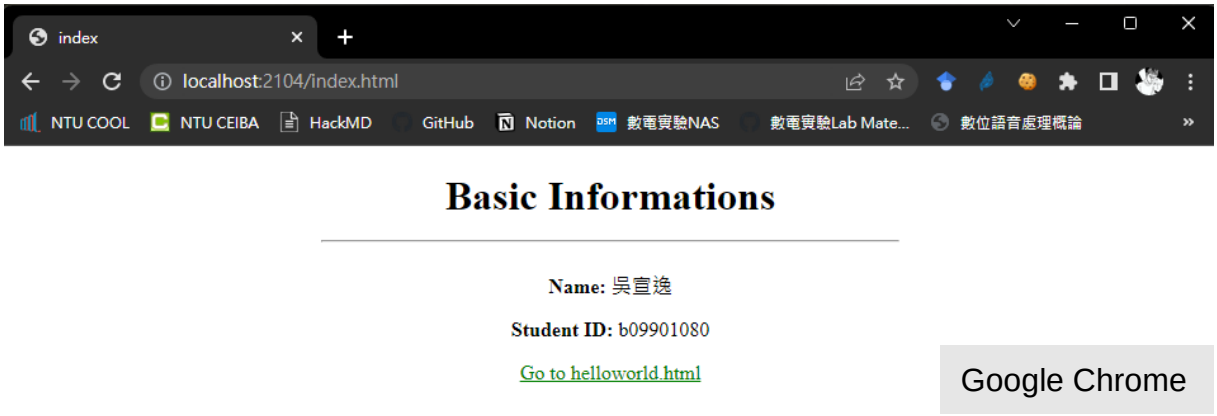
```
if 'favicon.ico' not in str(e):
    print('404 Not Found')
    ConnectionSocket.send('HTTP/1.1 404 NotFound\r\n'.encode('utf-8'))
    ConnectionSocket.close()
```

- p3 – Proxy Server
  - ✓ Default (HOST, PORT) = (‘127.0.0.1’, 2104) for proxy server
  - ✓ Default (HOST, PORT) = (‘127.0.0.1’, 2103) for web server
  - ✓ Output Results





✓ Access HTML files through Google Chrome and Microsoft Edge.



✓ Todos

1. Run web_server of p2 beforehand.
2. Create proxy server socket TCPServerSocket, bind to localhost:2104, and listen to 10 sockets.

```
TCPServerSocket = socket(AF_INET, SOCK_STREAM)
HOST, PORT = '127.0.0.1', 2104  # default HOST:PORT
TCPServerSocket.bind((HOST, PORT))
TCPServerSocket.listen(10)
```

3. Establish connection with client socket TCPClientSocket.

```
TCPClientSocket, Addr = TCPServerSocket.accept()
```

4. Receive HTTP request (max.4096 characters) from client and decode in UTF-8

codec, just like what web_server in p2 did.

```
RecvMessage = TCPClientSocket.recv(4096).decode('utf-8')
Filename = RecvMessage.split()[1].partition("/")[2]
FileToUse = "/" + Filename  # path of file
```

5. If the request html file is in cache, open and read html files in UTF-8 codec.

```
f = open(FileToUse[1:], "r", encoding='utf-8')
DataInFile = f.readlines()
f.close()
```

Then send HTTP response to client with status, header, and data.

```
TCPClientSocket.send(("HTTP/1.1 200 OK\r\n").encode('utf-8'))
TCPClientSocket.send(("Content-Type:text/html\r\n\r\n").encode('utf-8'))
for i in range(len(DataInFile)):
    TCPClientSocket.send(DataInFile[i].encode('utf-8'))
TCPClientSocket.send("\r\n".encode('utf-8'))
```

After transmission, close client socket.

```
TCPClientSocket.close()
```

6. If the html file is not in cache, create a socket SocketOnProxyServer and connect to localhost:2103 in order to send HTTP request to web_server(on localhost:2103) for corresponding html file.

```
SocketOnProxyServer = socket(AF_INET, SOCK_STREAM)
SocketOnProxyServer.connect(('127.0.0.1', 2103))
```

Use TextIOWrapper FileObject as a file-like object, which will receive and store the HTTP response in UTF-8 codec after sending HTTP request to web server.

```
FileObject = SocketOnProxyServer.makefile('rw', None, encoding='utf-8 ')
FileObject.write("GET " + FileToUse + " HTTP/1.1\r\n")
FileObject.flush()
```

Read received HTTP response into string list Buffer from FileObject, also extract status and Data from Buffer. If status code is 404, print “404 Not Found” on command.

```
Buffer = FileObject.readlines()
pivot = [i for i, val in enumerate(Buffer) if val == '\n']  # index of blank
status = Buffer[0].split()[1]
if status == '404':
    print('404 Not Found')
    continue
Data = Buffer[pivot[0]+1:]
```

If status code is 200, save Data to cache TmpFile, and send Buffer (HTTP response) back to client socket TCPClientSocket.

```
TmpFile = open(Filename, "w", encoding='utf-8')
for i in range(len(Buffer)):
    TCPClientSocket.send(Buffer[i].encode())
for j in range(len(Data)):
```

```
    TmpFile.write(str(Data[j]))
TmpFile.close()
SocketOnProxyServer.close()
```