Jason Jaroszewicz

Dr. John Zhao

CS-GY 6843: Computer Networking, Section INET

10-02-2023

Socket Prog.1: Web Server

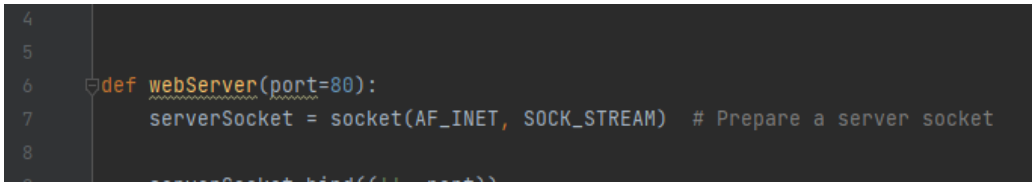
Lab Environment Details

Windows 10 *ipconfig* results:

```
Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::9d3b:12f:8535:1830%9
   IPv4 Address. . . . . . . . . . . : 192.168.1.203
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

```python
# import socket module
from socket import *
import sys  # In order to terminate the program


def webServer(port=80):
    serverSocket = socket(AF_INET, SOCK_STREAM)  # Prepare a server socket
    serverSocket.bind(('', port))
    serverSocket.listen(1)

    while True:
        # Establish the connection
        print(f'Ready to serve on port {port}...')
        connectionSocket, addr = serverSocket.accept()

        try:
            message = connectionSocket.recv(1024)
            filename = message.split()[1]

            with open(filename[1:], 'r') as f:
                outputdata = f.read()
                f.close()
                # Send one HTTP header line into socket
                outputdata = f'HTTP/1.1 200 OK\r\n\r\n{outputdata}'

                # Send the content of the requested file to the client
                for i in range(0, len(outputdata)):
                    connectionSocket.send(outputdata[i].encode())
            connectionSocket.close()
        except IOError:
            # Send response message for file not found
            error = 'HTTP/1.1 404 Not Found\r\n\r\n'
            print(f'error: {error}')

            # Close client socket
            for i in range (0, len(error)):
                connectionSocket.send(error[i].encode())
            connectionSocket.close()
    serverSocket.close()
    sys.exit()  # Terminate the program after sending the corresponding data


if __name__ == "__main__":
    try:
        port = input("Input desired port or press 'Enter' to initiate the
server...")
        if port == '':
            # port 80
            webServer()
        elif 0 <= int(port) <= 65535:
            webServer(port)
        else:
            print("Invalid input. Port must be an integer between 0 and
65535.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

- Line 6: Changed default port to 80 to match the default port expected by the browser if no other port is specified.

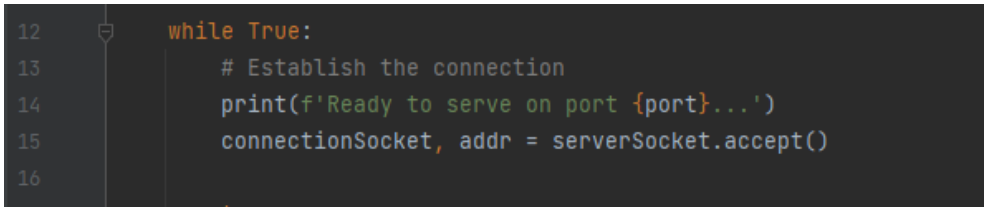


```
4
5
6   def webServer(port=80):
7       serverSocket = socket(AF_INET, SOCK_STREAM)  # Prepare a server socket
8
```

- Line 9-10: Preparing server socket. Binding the ip address and port number. Listening on the socket.

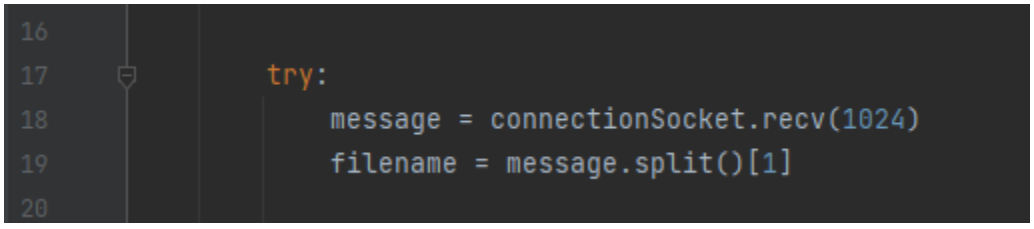
```
7       serverSocket = socket(AF_INET, SOCK_STREAM)  # Prepare a server socket
8
9       serverSocket.bind(('', port))
10      serverSocket.listen(1)
11
```

- Line 14-15: Outputting passed port number to the console. Accepting a connection with the address.



```
12      while True:
13          # Establish the connection
14          print(f'Ready to serve on port {port}...')
15          connectionSocket, addr = serverSocket.accept()
16
```

- Line 17-19: Receiving data from the socket with a 1024-byte buffer. Parsing filename from `message` variable.



```
16
17          try:
18              message = connectionSocket.recv(1024)
19              filename = message.split()[1]
20
```

- Line 21-25: Opening file using `with` to ensure file is closed even if an error is thrown. Reading file data into `outputdata` variable and closing the file. Preparing

`outputdata` with one HTTP header line.

```
20
21              with open(filename[1:], 'r') as f:
22                  outputdata = f.read()
23                  f.close()
24                  # Send one HTTP header line into socket
25                  outputdata = f'HTTP/1.1 200 OK\r\n\r\n{outputdata}'
26
```

- Line 31-39: Handling exceptions. Creating an error message with a 404 header line. Printing error to console. Closing the client socket after sending error.

```
30                  connectionSocket.close()
31              except IOError:
32                  # Send response message for file not found
33                  error = 'HTTP/1.1 404 Not Found\r\n\r\n'
34                  print(f'error: {error}')
35
36                  # Close client socket
37                  for i in range (0, len(error)):
38                      connectionSocket.send(error[i].encode())
39                  connectionSocket.close()
```

- Line 45-56 Main method. Handling user input with a try-except block to gracefully handle invalid input. Expecting a valid integer in the range of 0 through 65535, the valid port numbers. If there is no port entered, port 80 will be passed as default.

```
45  if __name__ == "__main__":
46      try:
47          port = input("Input desired port or press 'Enter' to initiate the server...")
48          if port == '':
49              # port 80
50              webServer()
51          elif 0 <= int(port) <= 65535:
52              webServer(port)
53          else:
54              print("Invalid input. Port must be an integer between 0 and 65535.")
55      except ValueError:
56          print("Invalid input. Please enter a valid integer.")
57
```