Computer Network Lab
CSE-325

Assignment - 3

Submitted by -
Gyanendra Shukla
CSE 1
191112040

To write a Socket Program to implement CHAT between client & server

I implemented a program where multiple clients can connect to the server and send messages to each other. The Server listens to any incoming connection and sends the message to all the clients.

Chat Server

```
import socket
import select

IP = "127.0.0.1"
PORT = 12345

class ChatServer:
    HEADER_LENGTH = 10
    def __init__(self, ip, port) -> None:
        self.server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.server_socket.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
        self.server_socket.bind((ip, port))
        self.server_socket.listen()

        self.socket_list = [self.server_socket]
        self.clients = {}
        print("Server started on {}:{}".format(ip, port))

    def receive_message(self, client_socket):
        try:
            message_header = client_socket.recv(self.HEADER_LENGTH)

            if not len(message_header):
                return False

            message_length = int(message_header.decode("utf-8").strip())

```

```
            return {'header': message_header, 'data':
                    client_socket.recv(message_length)}

        except:
            return False

    def start(self):
        while True:
            read_sockets, _, exception_sockets =
select.select(self.socket_list,
                                                                  [],
self.socket_list)
            for notified_socket in read_sockets:
                if notified_socket == self.server_socket:
                    client_socket, client_address =
self.server_socket.accept()

                    user = self.receive_message(client_socket)

                    if user is False:
                        continue

                    self.socket_list.append(client_socket)
                    self.clients[client_socket] = user

                    print("Accepted new connection from {}:{}, username: {}"
                            .format(*client_address,
user['data'].decode("utf-8")))

                else:
                    message = self.receive_message(notified_socket)

                    if message is False:
                        print("Closed connection from: {}"
                              .format(self.clients[notified_socket]['data']
                                      .decode("utf-8")))

                        self.socket_list.remove(notified_socket)
                        del self.clients[notified_socket]
                        continue

                    user = self.clients[notified_socket]
                    print("Received message from {}: {}"
                          .format(user['data'].decode("utf-8"),
message['data']
                                  .decode("utf-8")))

                    for client_socket in self.clients:
                        if client_socket != notified_socket:
                            client_socket.send(user['header'] + user['data']
                                               + message['header'] +
message['data'])

            for notified_socket in exception_sockets:
                # If we've got exceptional socket, probably it's broken one
                # so we need to remove it from socket list
                self.socket_list.remove(notified_socket)
                del self.clients[notified_socket]
```

```
80
81
82  if __name__ == "__main__":
83      server = ChatServer(IP, PORT)
84      server.start()
```

Chat Client

```
1   import socket
2   import select
3   import errno
4   import sys
5
6
7   IP = "127.0.0.1"
8   PORT = 12345
9
10  class ChatClient:
11      HEADER_LENGTH = 10
12      def __init__(self, ip, port, username) -> None:
13          self.client_socket = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
14          self.client_socket.connect((ip, port))
15          self.client_socket.setblocking(False)
16
17          self.username = username.encode("utf-8")
18          self.username_header = f"{len(self.username):
    <{self.HEADER_LENGTH}}".encode("utf-8")
19          self.client_socket.send(self.username_header + self.username)
20
21      def start(self):
22          while True:
23              message = input(f"{self.username}> ")
24              if message:
25                  message = message.encode("utf-8")
26                  message_header = f"{len(message):
    <{self.HEADER_LENGTH}}".encode("utf-8")
27                  self.client_socket.send(message_header + message)
28
29              try:
30                  while True:
31                      username_header =
    self.client_socket.recv(self.HEADER_LENGTH)
32                      if not len(username_header):
33                          print("Connection closed by the server")
34                          sys.exit()
35
36                      username_length = int(username_header.decode("utf-
    8").strip())
37                      username =
    self.client_socket.recv(username_length).decode("utf-8")
38
39                      message_header =
    self.client_socket.recv(self.HEADER_LENGTH)
40                      message_length = int(message_header.decode("utf-
    8").strip())
```

```
                    message =
self.client_socket.recv(message_length).decode("utf-8")

                    print(f"{username}> {message}")
            except IOError as err:
                if err.errno != errno.EAGAIN and err.errno !=
errno.EWOULDBLOCK:
                    print(f"Reading error: {str(err)}")
                    sys.exit()

                continue

            except Exception as e:
                print(f"Reading error: {str(e)}")
                sys.exit()

if __name__ == "__main__":
    chat_client = ChatClient(IP, PORT, str(sys.argv[1]))
    chat_client.start()
```

Output

```
In D:\Books\sem 6\networks\labs\lab4 107ms python .\ChatServer.py
Server started on 127.0.0.1:12345
Accepted new connection from 127.0.0.1:1076, username: gyan
Accepted new connection from 127.0.0.1:1246, username: al
Received message from al: Hello Gyan
Received message from gyan: Hi, al! This is some message.
Received message from al: some
Received message from al: more
Received message from al: messages
Received message from al: from al
Received message from gyan: ok bye!
Received message from al: bye!!
```

Fig: Chat Server

```
In D:\Books\sem 6\networks\labs\lab4 223ms python .\ChatClient.py gyan
b'gyan'>
al> Hello Gyan
b'gyan'> Hi, al! This is some message.
b'gyan'>
al> some
al> more
al> messages
al> from al
b'gyan'> ok bye!
b'gyan'>
```

Fig: Chat Client 1

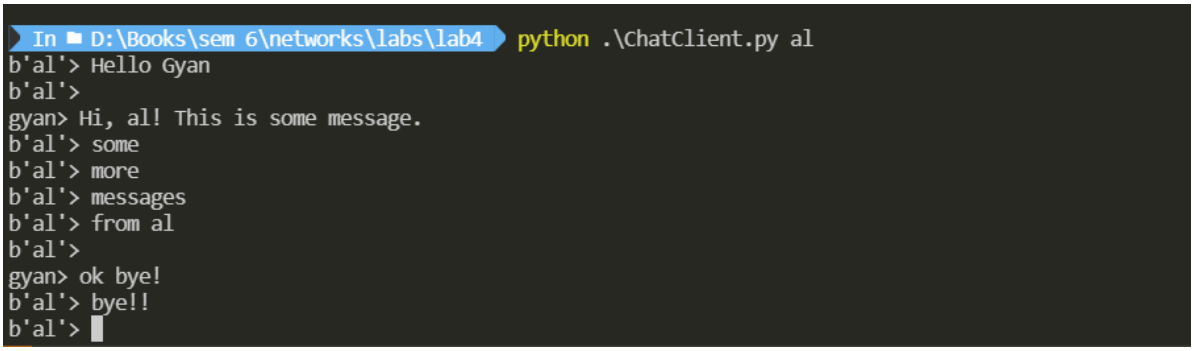

Fig: Chat Client 2

To write a Socket Program to implement File Transfer between client & server

I wrote a program to implement file transfer between client and server. The server receives the file from the client. The client sends the file it has to send to the server through command line args.

The received file has a `recv-` prefix.

File Transfer Server

```python
import socket
import os

IP = "127.0.0.1"
PORT = 12345

class FileServer:
    SEPARATOR = "<SEPARATOR>"
    BUFFER_SIZE = 4096

    def __init__(self, ip, port) -> None:
        self.sock = socket.socket()
        self.sock.bind((ip, port))
        self.sock.listen()
        print(f"Listening on {ip}:{port}")

    def receive(self):
        client_socket, address = self.sock.accept()
        received = client_socket.recv(self.BUFFER_SIZE).decode()
        filename, filesize = received.split(self.SEPARATOR)

        filename = "recv-" + os.path.basename(filename)
        filesize = int(filesize)

        with open(filename, "wb") as f:
            print(f"Incoming file, saving as {filename}")
            while True:
                bytes_read = client_socket.recv(self.BUFFER_SIZE)
                if not bytes_read:
                    # we've completed receiving files
                    break
```

```
                f.write(bytes_read)
        print(f"Done receiving {filename}")

        client_socket.close()
        self.sock.close()


if __name__ == "__main__":
    server = FileServer(IP, PORT)
    server.receive()

```

File Transfer Client

```
import socket
import sys
import os

class FileClient:
    SEPARATOR = "<SEPARATOR>"
    BUFFER_SIZE = 4096

    def __init__(self, ip, port) -> None:
        self.sock = socket.socket()
        print(f"Connecting to {ip}:{port}")
        self.sock.connect((ip, port))
        print(f"Connected to {ip}:{port}")

    def send(self, filename):
        if not os.path.isfile(filename):
            print(f"{filename} does not exist!")
            return
        filesize = os.path.getsize(filename)
        self.sock.send(f"{filename}{self.SEPARATOR}{filesize}".encode())

        with open(filename, "rb") as f:
            print(f"Sending {filename}")
            while True:
                bytes_read = f.read(self.BUFFER_SIZE)
                if not bytes_read:
                    # we've completed sending files
                    break
                self.sock.sendall(bytes_read)
        print(f"Done sending {filename}")

        self.sock.close()


if __name__ == "__main__":
    client = FileClient("127.0.0.1", 12345)
    filename = sys.argv[1]
    client.send(filename)
```

Output

```
In D:\Books\sem 6\networks\labs\lab4 Get-ChildItem

    Directory: D:\Books\sem 6\networks\labs\lab4

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          2/7/2022 10:57 AM           8712 191112040.md
-a---          2/7/2022 10:49 AM          12311 cclient1.png
-a---          2/7/2022 10:48 AM          11421 cclient2.png
-a---          2/7/2022  9:29 AM           2165 ChatClient.py
-a---          2/7/2022  9:28 AM           2999 ChatServer.py
-a---          2/7/2022 10:49 AM          22012 cserver.png
-a---          2/7/2022  9:06 AM           1119 FileClient.py
-a---          2/7/2022 10:56 AM           1168 FileServer.py
```

Fig: Files before sending

```
In D:\Books\sem 6\networks\labs\lab4 python .\FileServer.py
Listening on 127.0.0.1:12345
Incoming file, saving as recv-ChatClient.py
Done receiving recv-ChatClient.py
In D:\Books\sem 6\networks\labs\lab4 15.895s
```

Fig: File Transfer Server

```
In D:\Books\sem 6\networks\labs\lab4 python .\FileClient.py .\ChatClient.py
Connecting to 127.0.0.1:12345
Connected to 127.0.0.1:12345
Sending .\ChatClient.py
Done sending .\ChatClient.py
In D:\Books\sem 6\networks\labs\lab4 156ms
```

Fig: File Transfer Client

```
In D:\Books\sem 6\networks\labs\lab4 156ms Get-ChildItem

    Directory: D:\Books\sem 6\networks\labs\lab4

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          2/7/2022 10:57 AM           8712 191112040.md
-a---          2/7/2022 10:49 AM          12311 cclient1.png
-a---          2/7/2022 10:48 AM          11421 cclient2.png
-a---          2/7/2022  9:29 AM           2165 ChatClient.py
-a---          2/7/2022  9:28 AM           2999 ChatServer.py
-a---          2/7/2022 10:49 AM          22012 cserver.png
-a---          2/7/2022  9:06 AM           1119 FileClient.py
-a---          2/7/2022 10:56 AM           1168 FileServer.py
-a---          2/7/2022 11:00 AM           2165 recv-ChatClient.py
```

Fig: Files after sending (recv-ChatClient.py at the bottom)