Mikias Berhanu
2021280115
Assignment Submission V

Simple Terminal Based Chat Room with Sockets

Socket programming is a way of connecting nodes like computers and servers with each other. One packet will listen for incoming connections, in this case the server and the other socket will try to connect with the server on that specific port. **Socket** is a combination of **IP Address** and **Port Number**.

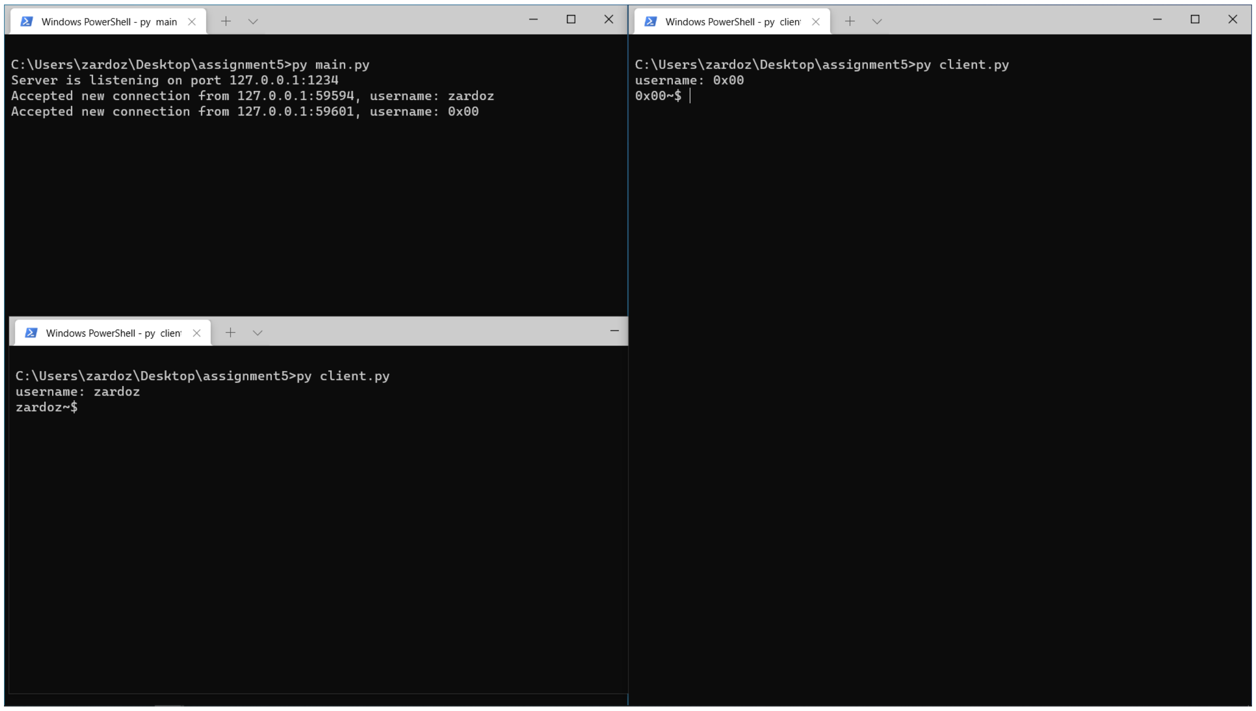
In this assignment I have used python as a tool to create this simple chat server. The program uses the **socket** module of python which helps us to create socket objects, listen for incoming connections, receive connections, accept connections etc… Another module is the **select** module which is an interface on top of the operating system which helps sockets from different operating systems communicate seamlessly.

Server program logic

```
create server socket object
bind server with ip and port number (in this case 127.0.0.1 and any other port number)
listen for incoming connections
while true:
    normal sockets, and exception sockets
    for sockets in the normal sockets:
        if socket is server socket
            get client ip address and port number
            receive client message
            if there is no user or message:
                just continue
            else append the new user to a list of other users
        else:
            directly receive message
            if there is no passage:
                drop connection and socket
            display message
            for clients in the client list:
                broadcast the message
```

Client program logic

```
create client socket
bind server with ip and port number (in this case 127.0.0.1 and any other port number)
ask user for username input
while true:
    ask user for input
    if there is message:
        encode the message to utf-8
        set the message header
        send the message using client socket
    try:
        while true:
            receive connections
            if there is nothing received:
                it means connection is closed from the server
            get the username length
            according to the username length get username
            get message header
            get message according to the message header and length
            receive messages
            display the message
    check for input output error:
        continue
    check for other exceptions:
        exit
```

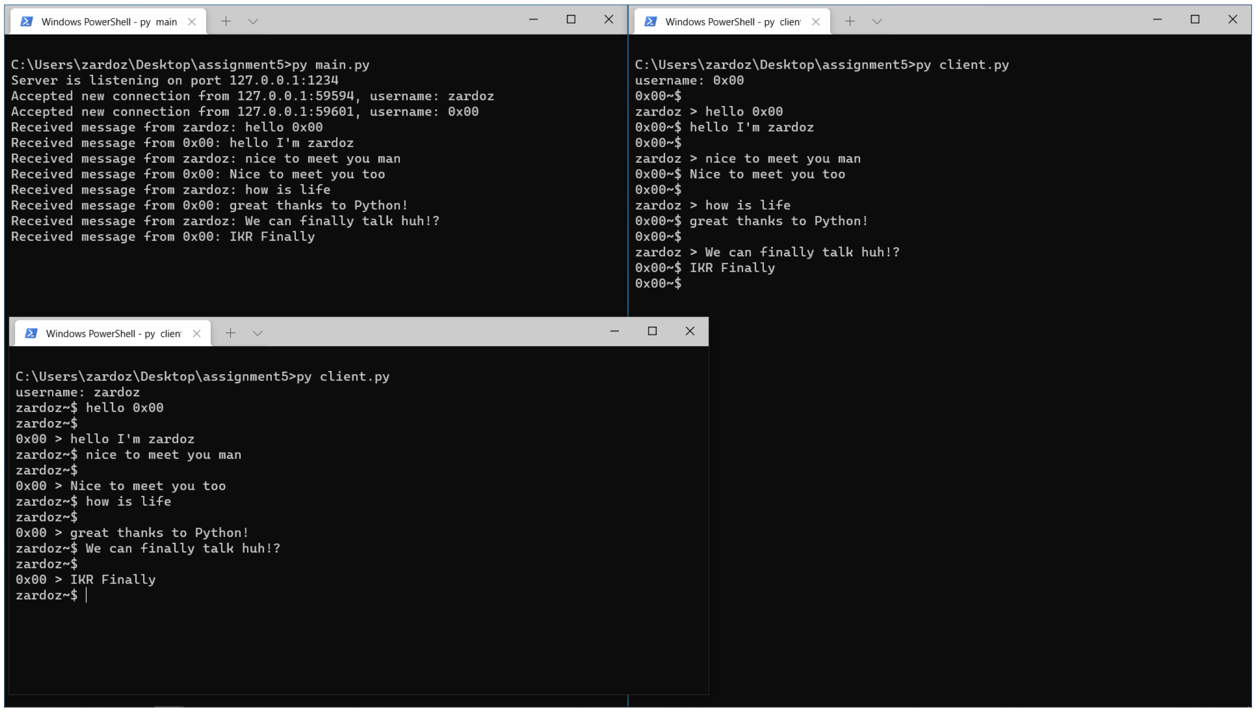


Windows PowerShell - py main
Windows PowerShell - py clien
Windows PowerShell - py clien

```
C:\Users\zardoz\Desktop\assignment5>py main.py
Server is listening on port 127.0.0.1:1234
Accepted new connection from 127.0.0.1:59594, username: zardoz
Accepted new connection from 127.0.0.1:59601, username: 0x00
```

```
C:\Users\zardoz\Desktop\assignment5>py client.py
username: zardoz
zardoz~$
```

```
C:\Users\zardoz\Desktop\assignment5>py client.py
username: 0x00
0x00~$
```



Windows PowerShell - py main
Windows PowerShell - py clien
Windows PowerShell - py clien

```
C:\Users\zardoz\Desktop\assignment5>py main.py
Server is listening on port 127.0.0.1:1234
Accepted new connection from 127.0.0.1:59594, username: zardoz
Accepted new connection from 127.0.0.1:59601, username: 0x00
Received message from zardoz: hello 0x00
Received message from 0x00: hello I'm zardoz
Received message from zardoz: nice to meet you man
Received message from 0x00: Nice to meet you too
Received message from zardoz: how is life
Received message from 0x00: great thanks to Python!
Received message from zardoz: We can finally talk huh!?
Received message from 0x00: IKR Finally
```

```
C:\Users\zardoz\Desktop\assignment5>py client.py
username: zardoz
zardoz~$ hello 0x00
zardoz~$
0x00 > hello I'm zardoz
zardoz~$ nice to meet you man
zardoz~$
0x00 > Nice to meet you too
zardoz~$ how is life
zardoz~$
0x00 > great thanks to Python!
zardoz~$ We can finally talk huh!?
zardoz~$
0x00 > IKR Finally
zardoz~$
```

```
C:\Users\zardoz\Desktop\assignment5>py client.py
username: 0x00
0x00~$
zardoz > hello 0x00
0x00~$ hello I'm zardoz
0x00~$
zardoz > nice to meet you man
0x00~$ Nice to meet you too
0x00~$
zardoz > how is life
0x00~$ great thanks to Python!
0x00~$
zardoz > We can finally talk huh!?
0x00~$ IKR Finally
0x00~$
```

Running demo of a Simple Chat Room using Socket Programming

Source code can be found here https://github.com/mikias21/NADC/tree/main/assignment_5

Source code for Server.py

```
# Author : Mikias Berhanu
# Date : 18/12/2021

import socket
import select


class Server:
   """
       This class is responsible for the server logic
       initializes the server socket
       binds the local host port with port 7777
       accepts incoming connections from other computers or networks
   """
   def __init__(self):
       self.header_length = 10
       self.ip = "127.0.0.1"
       self.port = 45678
       self.server_socket = None
       self.socket_list = [self.server_socket]
       self.users = {}

   def init_server_socket(self):
       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((self.ip, self.port))
       self.server_socket.listen()
       print(f"server is listening on port {self.ip}:{self.port}")
       return self.server_socket

   def receive_client_messages(self, client_socket: socket.socket):
       try:
           message_header = client_socket.recv(self.header_length)
           if not len(message_header):  # if there is not message
               return False
           # get the length and receive message based on the message length
not more than that
           message_length = int(message_header.decode('utf-8').strip())
           return {'header': message_header, 'data':
client_socket.recv(message_length)}
       except:
           return False
```

```
if __name__ == '__main__':
   # Program Loop
   # keep listening and accepting connections
   server = Server()
   server_socket = server.init_server_socket()
   socket_list = [server_socket]
   users = {}
   while True:
       incoming_sockets, _, exception_sockets = select.select(socket_list, [],
socket_list)
       for n_socket in incoming_sockets:
           if n_socket == server_socket:  # check if the incoming socket is
server socket
                c_socket, c_address = server_socket.accept()
               client = server.receive_client_messages(c_socket)
               if not client:
                   continue
               socket_list.append(c_socket)
               users[c_socket] = client
               print('[*][*]new connection from {}:{}, username:
{}'.format(*c_address, client['data'].decode('utf-8')))
           else: # a client from our list is sending message
                c_message = server.receive_client_messages(n_socket)
               if not c_message:  # check if there is a client message no
client connected
                    print('[-][-]unable to make connection')
                   socket_list.remove(n_socket)
                   del users[n_socket]
                   continue
               client = users[n_socket] # get the user who is sending message
                print(f'Received message from {client["data"].decode("utf-8")}:
{c_message["data"].decode("utf-8")}')
               for c_socket in users:  # for users in our list broadcast
message
                    if c_socket != n_socket:
                       c_socket.send(client['header'] + client['data'] +
c_message['header'] + c_message['data'])

       # handle socket exceptions
       for n_socket in exception_sockets:
           socket_list.remove(n_socket)
           del users[n_socket]
```

Source code for Client.py

```python
# Author : Mikias Berhanu
# Date : 18/12/2021

import socket
import errno
import sys

# constants
CLIENT_HEADER_LENGTH = 10
CLIENT_IP = "127.0.0.1"
CLIENT_PORT = 45678

# Ask user for username
r_username = input("username please: ")

# create socket connection
c_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c_socket.connect((CLIENT_IP, CLIENT_PORT))
c_socket.setblocking(False)

# encode username and create connection header
u_name = r_username.encode('utf-8')
u_header = f"{len(u_name):<{CLIENT_HEADER_LENGTH}}".encode('utf-8')
c_socket.send(u_header + u_name)


while True:
   m_input = input(f"{r_username}~$ ")
   if m_input:
       m_input = m_input.encode('utf-8')
       m_header = f"{len(m_input):<{CLIENT_HEADER_LENGTH}}".encode('utf-8')
       c_socket.send(m_header + m_input)
   try:
       while True:
           u_header = c_socket.recv(CLIENT_HEADER_LENGTH)
           if not len(u_header):
               print("[!][!] unable to make connection")
               sys.exit()
           u_length = int(u_header.decode('utf-8').strip())
           u_name = c_socket.recv(u_length).decode('utf-8')
           m_header = c_socket.recv(CLIENT_HEADER_LENGTH)
           m_length = int(m_header.decode('utf-8').strip())
           m_incoming = c_socket.recv(m_length).decode('utf-8')
           print(f"{u_name} > {m_incoming}")
   except IOError as e:
       if e.errno != errno.EAGAIN and e.errno != errno.EWOULDBLOCK:
           print("[-][-] something is wrong with the server", str(e))
           sys.exit()
       continue
   except Exception as e:
       print("[-][-]something is wrong with the server", str(e))
       sys.exit()
```