

LAB REPORT 02

CYBERSECURITY

RITIK TIWARI (B21CS098)

Link for the code files: <https://github.com/testgithubtiwari/CyberSecurity-Labs/tree/main/lab02>

Question a). Your server program "server1" will be a single process server that can handle only one client at a time. If a second client tries to chat with the server while one client's session is already in progress, the second client's socket operations should see an error.

Solution: For this question we must check only one condition if server is already connected to a client and any other client is trying to connect to the server then it refuses. I have done with the initialization of a bool variable which will take over the conditions and initially it was false and if a client has successfully connected then it changes to true which stops to connect with another client. I have attached both the codes screenshot client.py and serve.py file and the result which is an error message for the second client if any other client is already connected.

Screenshot of the codes for both client.py and server.py file and the Output of the screenshot is shown below

Result

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS SEARCH ERROR GITLENS COMMENTS
Enter the port: 123
Enter the server ip address: 127.0.0.1
Traceback (most recent call last):
  File "D:\cybersecurity\lab02\q1\server.py", line 57, in <mod
    ule>
    main()
  File "D:\cybersecurity\lab02\q1\server.py", line 32, in main
    server_socket.bind((localhost, port))
TypeError: 'str' object cannot be interpreted as an integer
PS D:\cybersecurity\lab02\q1> python .\server.py
Enter the port: 123
Enter the server ip address: localhost
Server listening on port 123
Connection from ('127.0.0.1', 61646)

Traceback (most recent call last):
  File "D:\cybersecurity\lab02\q1\client.py", line 32, in <mod
    ule>
    main()
  File "D:\cybersecurity\lab02\q1\client.py", line 11, in main
    client_socket.connect((localhost_server, int(port_server))
)
socket.gaierror: [Errno 11001] getaddrinfo failed
PS D:\cybersecurity\lab02\q1> python .\client.py
Enter the port: 123
Enter the server ip address: localhost
Connected to server.
Enter arithmetic expression: 1 + 2
Result: 3
Enter arithmetic expression: []

PS D:\cybersecurity\lab02\q1> python .\client.py
Enter the port: 123
Enter the server ip address: localhost
Server is busy. Please try again later.
PS D:\cybersecurity\lab02\q1> []
```

Client.py

```
You, 5 minutes ago | Editor (100) | Click here to ask Blackbox to help you code faster
import socket
import sys

def main():

    port_server = input("Enter the port: ")
    localhost_server=input("Enter the server ip address: ")

    try:
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_socket.connect((localhost_server, int(port_server)))
    except ConnectionRefusedError:
        print("Server is busy. Please try again later.")
        return

    print("Connected to server.")
    try:
        while True:
            expression = input("Enter arithmetic expression: ")
            client_socket.send(expression.encode())
            result = client_socket.recv(1024).decode()
            print("Result:", result)

    except KeyboardInterrupt:
        print("Client terminated.")
    except Exception as e:
        print("Error:", e)
    finally:
        client_socket.close()

if __name__ == "__main__":
    main()
```

server.py

```
import socket
import sys

server_busy_message = "Server busy now. Try again later!"
server_busy = False

You, 4 minutes ago | 1 author (You)
class ServerBusyError(Exception):
    pass

def handle_client_connection(client_socket):
    while True:
        try:
            data = client_socket.recv(1024)
            if not data:
                break
            result = eval(data.decode())
            client_socket.send(str(result).encode())
        except ServerBusyError:
            client_socket.send("ServerBusyError".encode())
            client_socket.close()
            break
        except Exception as e:
            print("Error:", e)
            break
    client_socket.close()

def main():
    global server_busy
    port = input("Enter the port: ")
    localhost=input("Enter the server ip address: ")
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((localhost, int(port)))
    server_socket.listen(1)
    print(f"Server listening on port {port}")

    while True:
        try:
            client_socket, client_address = server_socket.accept()
            print(f"Connection from {client_address}")
            if server_busy:
                raise ServerBusyError
            else:
                server_busy = True
                handle_client_connection(client_socket)
                server_busy = False
        except KeyboardInterrupt:
            print("Server terminated.")
            break
```

Q2) Question b. Your server program "server2" will be a multi-process or multi-threaded server that will fork a process for every new client it receives. Multiple clients should be able to simultaneously chat with the server.

Solution: For this we must allow as many as client with the server and perform the arithmetic expression to achieve this I have used the process of multiprocessing where each time a client join, I have just started the new process which is like the fork of the process which makes the ongoing process break into two which allowed the other client to join successfully. I have attached the screenshot of the code and the result where I have run the 4 clients simultaneously and gives the correct evaluation of the given arithmetic expression and more thing to evaluate the expression, I have used the python math library eval function which eval a given mathematical expression and return the result as integer.

client.py

```

1  import socket
2  import sys
3
4  def main():
5      server_ip = input("Enter the server ip address: ")
6      server_port = input("Enter the port: ")
7
8      try:
9          client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10         client_socket.connect((server_ip, int(server_port)))
11         print("Connected to server.")
12         while True:
13             expression = input("Enter arithmetic expression: ")
14             client_socket.send(expression.encode())
15             result = client_socket.recv(1024).decode()
16             print("Result:", result)
17
18         except KeyboardInterrupt:
19             print("Client terminated.")
20         except Exception as e:
21             print("Error:", e)
22         finally:
23             client_socket.close()
24
25 if __name__ == "__main__":
26     main()

```

Result

<pre> PS D:\cybersecurity> cd .\lab02\ PS D:\cybersecurity\lab02> cd .\q2\ PS D:\cybersecurity\lab02\q2> python .\server.py Enter the port: 123 Enter the server ip address: localhost Server listening on port 123 Connection from ('127.0.0.1', 57341) Connection from ('127.0.0.1', 57365) Connection from ('127.0.0.1', 57379) </pre>	<pre> PS D:\cybersecurity\lab02\q2> python .\client.py Enter the server ip address: localhost Enter the port: 123 Connected to server. Enter arithmetic expression: 2 * 4 * 4 Result: 32 Enter arithmetic expression: </pre>	<pre> PS D:\cybersecurity\lab02\q2> python .\client.py Enter the server ip address: localhost Enter the port: 123 Connected to server. Enter arithmetic expression: 9 + 19 Result: 28 Enter arithmetic expression: </pre>	<pre> PS D:\cybersecurity\lab02\q2> python .\client.py Enter the server ip address: localhost Enter the port: 123 Connected to server. Enter arithmetic expression: 1 + 2 Result: 3 Enter arithmetic expression: </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

server.py

```
import socket
import sys
import signal
import multiprocessing

def handle_client_connection(client_socket):
    while True:
        try:
            data = client_socket.recv(1024)
            if not data:
                break
            result = eval(data.decode())
            client_socket.send(str(result).encode())
        except Exception as e:
            print("Error:", e)
            break
    client_socket.close()

def handle_client(client_socket, client_address):
    print(f"Connection from {client_address}")
    handle_client_connection(client_socket)

def main():
    port = int(input("Enter the port: "))
    localhost_server = input("Enter the server ip address: ")
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((localhost_server, port))
    server_socket.listen(10)
    print(f"Server listening on port {port}")

    while True:
        try:
            client_socket, client_address = server_socket.accept()
            process = multiprocessing.Process(target=handle_client, args=(client_socket, client_address))
            process.start()
            client_socket.close()
        except KeyboardInterrupt:
            print("Server terminated.")
            break
        except Exception as e:
            print("Error:", e)
            break
    server_socket.close()

if __name__ == "__main__":
    main()
```