Indian Institute of Technology Jodhpur

CSL6010 – Cyber Security
LAB-2 Report

Name- Aman Srivastava
Roll No.- B20CS100

Date-2nd Feb 2023

- **INTRODUCTION:**

  This lab aims to build a simple client-server system where the client program connects to a server and sends arithmetic expressions to be evaluated by the server. Two versions of the server have been developed in this project: "server1" and "server2". The "server1" is a single process server that can handle only one client at a time. The "server2" is a multi-process or multi-threaded server that forks a process for every new client it receives.

- **SERVER 1 PROGRAM:**

  The server1 program is implemented as a single process server that listens on a specified port and handles a single client at a time. If a second client tries to connect to the server while one client's session is already in progress, the second client's socket operations will see an error.

  client.py (in text)

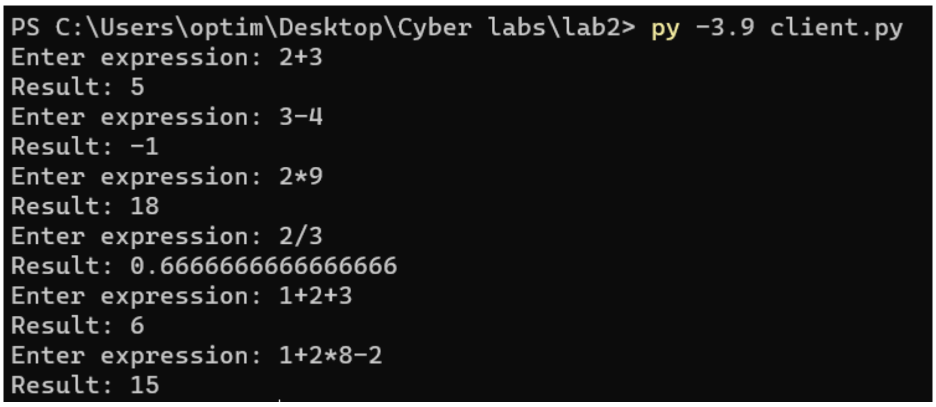
```
import socket

def main():
    host = 'localhost'
    port = 5000

    client_socket = socket.socket()
    try:
        client_socket.connect((host, port))
        while True:
            expr = input("Enter expression: ")
            client_socket.send(expr.encode())
            data = client_socket.recv(1024).decode()
            print("Result: " + data)
    except ConnectionRefusedError:
        print("Another client is already connected to the server. Please try
again later.")
    client_socket.close()

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

The client program connects to the server, sends arithmetic expressions to be evaluated, and displays the results received from the server. The client also handles the case where another client tries to connect to the server while the first client's session is in progress and prints an error message in such cases.

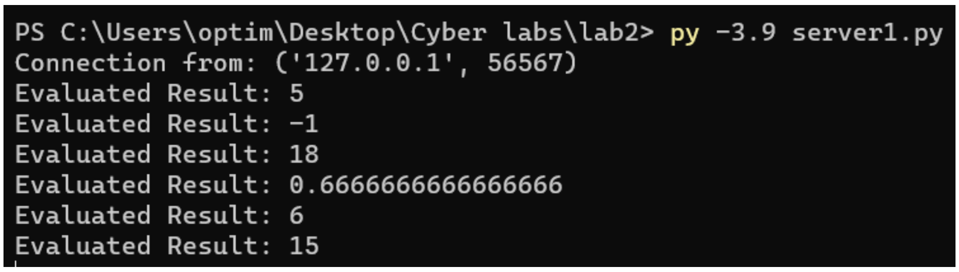
server1.py (in text)

```python
import socket
def handle_client(conn, addr):
    print("Connection from: " + str(addr))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break
        try:
            result = eval(data)
            print("Evaluated Result:",result)
            conn.send(str(result).encode())
        except:
            conn.send("Invalid Expression".encode())
    conn.close()
def main():
    host = 'localhost'
    port = 5000
    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(1)
    conn, addr = server_socket.accept()
    handle_client(conn, addr)
    conn.close()
    print("Connection closed")
    try:
        conn, addr = server_socket.accept()
    except:
        print("Error: Another client is already connected")
if __name__ == '__main__':
    main()
```

Output:



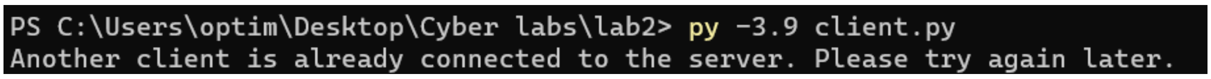
```
PS C:\Users\optim\Desktop\Cyber labs\lab2> py -3.9 client.py
Enter expression: 2+3
Result: 5
Enter expression: 3-4
Result: -1
Enter expression: 2*9
Result: 18
Enter expression: 2/3
Result: 0.6666666666666666
Enter expression: 1+2+3
Result: 6
Enter expression: 1+2*8-2
Result: 15
```



```
PS C:\Users\optim\Desktop\Cyber labs\lab2> py -3.9 server1.py
Connection from: ('127.0.0.1', 56567)
Evaluated Result: 5
Evaluated Result: -1
Evaluated Result: 18
Evaluated Result: 0.6666666666666666
Evaluated Result: 6
Evaluated Result: 15
```

The client connects to the server and evaluated expression is also shown to the server side along with the client side.

But when another client tries to connect to the server it gives an error message as below:



```
PS C:\Users\optim\Desktop\Cyber labs\lab2> py -3.9 client.py
Another client is already connected to the server. Please try again later.
```

- **SERVER 2 PROGRAM:**

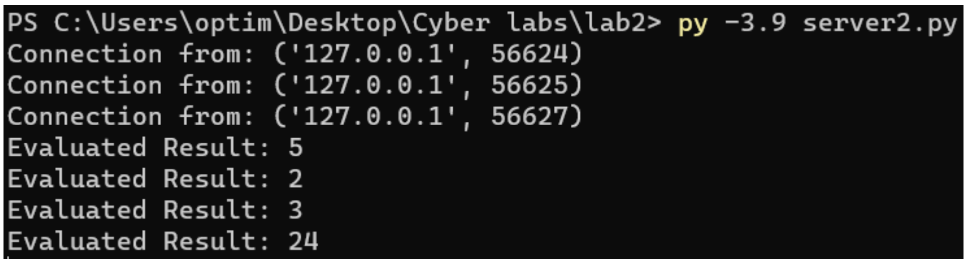
  The server2 program is implemented as a multi-process or multi-threaded server that forks a process for every new client it receives. Multiple clients can simultaneously chat with the server.

  server2.py (in text)

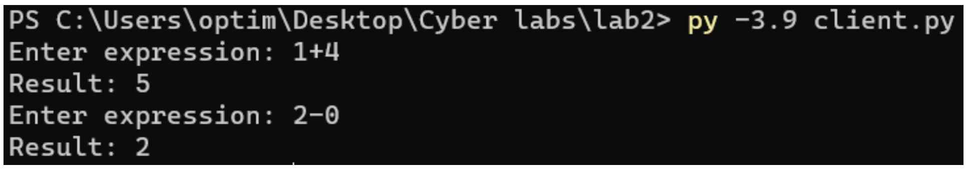
```python
import socket
from multiprocessing import Process
def handle_client(conn, addr):
    print("Connection from: " + str(addr))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break
        try:
            result = eval(data)
            print("Evaluated Result:",result)
            conn.send(str(result).encode())
        except:
            conn.send("Invalid Expression".encode())
    conn.close()
def main():
    host = 'localhost'
    port = 5000

    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(5)
    while True:
        conn, addr = server_socket.accept()
        p = Process(target=handle_client, args=(conn, addr))
        p.start()
        conn.close()
if __name__ == '__main__':
    main()
```

Output:

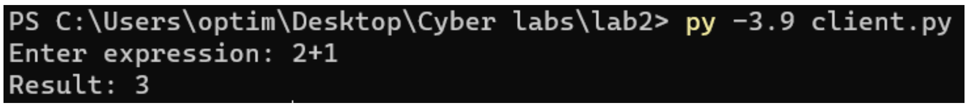
The server connects to different client at the same time.



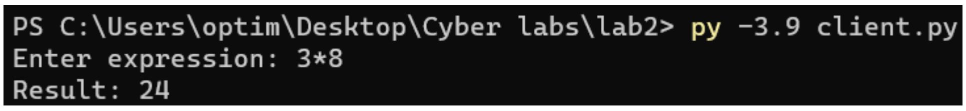
```
PS C:\Users\optim\Desktop\Cyber labs\lab2> py -3.9 server2.py
Connection from: ('127.0.0.1', 56624)
Connection from: ('127.0.0.1', 56625)
Connection from: ('127.0.0.1', 56627)
Evaluated Result: 5
Evaluated Result: 2
Evaluated Result: 3
Evaluated Result: 24
```



```
PS C:\Users\optim\Desktop\Cyber labs\lab2> py -3.9 client.py
Enter expression: 1+4
Result: 5
Enter expression: 2-0
Result: 2
```



```
PS C:\Users\optim\Desktop\Cyber labs\lab2> py -3.9 client.py
Enter expression: 2+1
Result: 3
```



```
PS C:\Users\optim\Desktop\Cyber labs\lab2> py -3.9 client.py
Enter expression: 3*8
Result: 24
```

- **CONCLUSION:**

This lab demonstrates the implementation of a simple client-server system using socket programming in Python. The two versions of the server (server1 and server2) handle different numbers of clients and provide a basis for further exploration of socket programming and concurrent client servers. The project also provides a good starting point for exploring more complex client-server systems with multiple clients and advanced features.