PROGRAM 8

Board & Environment Setup

Platform: Raspberry Pi / PC
OS: Raspberry Pi OS / Linux / Windows
Language: Python 3 (ONLY)

> **Note:** The `socket` module is a built-in Python library and does NOT require installation.

Program Title

Client–Server Communication using Socket Programming (Single Document)

Aim

To establish client–server communication using socket programming and exchange data between a server and a client.

Description / Concept

Client–server communication is a network model where one program (server) waits for requests and another program (client) initiates communication. The server listens on a specific port, accepts client connections, receives data, sends a response, and then closes the connection. The client connects to the server using its IP address and port number, sends a message, receives a reply, and terminates the connection.

Algorithm / Logic

Server Side Logic

1. Create a TCP socket
2. Bind the socket to an IP address and port number
3. Listen for incoming client connections
4. Accept client connection
5. Receive data from client
6. Send response to client
7. Close the connection

Client Side Logic

1. Create a TCP socket
2. Specify server IP address and port number
3. Connect to the server
4. Send message to server
5. Receive response from server
6. Close the connection

Program Code

Server Program (Run FIRST)

```
# ------------------------------------------
# PROGRAM 8: Server Program
# Client-Server Communication using Sockets
# Language : Python 3
# ------------------------------------------

import socket

# Create TCP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind socket to IP address and port
server_socket.bind(("0.0.0.0", 12345))

# Listen for incoming connections
server_socket.listen(1)
print("Server is listening on port 12345...")

# Accept client connection
client_socket, client_address = server_socket.accept()
print("Connected to client:", client_address)

# Receive data from client
data = client_socket.recv(1024)
message = data.decode()
print("Message from client:", message)

# Send response to client
client_socket.send("Message received by server".encode())

# Close connections
client_socket.close()
```

```
server_socket.close()
print("Server connection closed.")
```

Client Program (Run AFTER server)

```
# ------------------------------------------
# PROGRAM 8: Client Program
# Client-Server Communication using Sockets
# Language : Python 3
# ------------------------------------------

import socket

# Server IP address and port number
# Replace SERVER_IP with the server machine IP
SERVER_IP = "127.0.0.1"
SERVER_PORT = 12345

# Create TCP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server
client_socket.connect((SERVER_IP, SERVER_PORT))
print("Connected to server")

# Send message to server
message = "Hello from Client"
client_socket.send(message.encode())

# Receive response from server
response = client_socket.recv(1024)
print("Reply from server:", response.decode())

# Close connection
client_socket.close()
print("Client connection closed.")
```

Output / Result

- Server displays the message received from the client
- Client receives acknowledgment message from the server
- Successful client–server communication is established

Important Notes

- Server program must be executed before the client program
- Both server and client must be connected to the same network
- Port number must be the same in both programs

Possible Viva / Exam Questions

1. What is client–server communication?
2. What is a socket?
3. Difference between TCP and UDP?
4. Why must the server run before the client?
5. What is the role of IP address and port number?

Conclusion

The program demonstrates basic client–server communication using socket programming, enabling reliable data exchange between two networked systems.