Government College of Engineering, Jalgaon

Department of Computer Engineering

(CO309-Computer Network Technology Lab)

Name: Pallavi Govardhan Patil. PRN: 2241046. Class: TY.

Practicale No: 1. Session: V Subject Teacher: Ms.prajakta Swale.

Date of Performance: 02/09/2024. Date Of completation:09/09/2024.

Write Up	Correctness of Program	Documentation of Program	Viva	Attendance for Practical	Timely Completion	Total	Dated sign of Subject Teacher
4	2	2	5	2	5	20	

Practical No.1

❖ **AIM :-**

1. UNIX Sockets: WAP program in C/C++ /Python/Java sockets API.
   a. TCP sockets.
   b. UDP sockets.

❖ **SOFTWARE REQUIRED :-**

Operating System: - Ubuntu Python312 .

❖ **THEROY:-**

- ## What is Unix Socket

A **Unix socket** (or **Unix domain socket**, **UDS**) is an inter-process communication (IPC) mechanism that allows data exchange between processes running on the same host (computer). Unlike network sockets, which communicate over a network (e.g., TCP/IP), Unix sockets operate only within the local machine, providing a faster and more efficient means of communication between processes.

Here are some key characteristics of Unix sockets:

1. **Local-only Communication**: They are limited to processes on the same system, making them different from network sockets, which can communicate across different systems.
2. **Types**:
   - **Stream Sockets** (SOCK_STREAM): Provide reliable, connection-based communication similar to TCP. E.g **Like a Telephone Line**: Once a connection is established, it acts like a continuous phone call. Data flows back and forth smoothly and reliably
   - **Datagram Sockets** (SOCK_DGRAM): Provide connectionless, message-based communication similar to UDP. Eg.**Like Sending Letters**: You send individual messages (like letters) without making a permanent connection. Each message is sent independently and might arrive out of order or not at all.
3. **File System Representation**: Unix sockets are represented as files in the file system. These files allow processes to connect to the socket by referencing the file's path.

4. **File System Representation**: Unix sockets are represented as files in the file system. These files allow processes to connect to the socket by referencing the file's path.
5. **Performance**: Since they bypass the network stack, Unix sockets tend to have lower latency and higher throughput compared to network sockets.
6. **Common Use Cases**:
    - Communication between system services (like a database server and client).
    - Communication within containerized environments (e.g., Docker containers on the same host).
    - APIs like **gRPC**, which can use Unix sockets for communication between microservices on the same machine

❖ TCP Sockets (Stream Sockets)

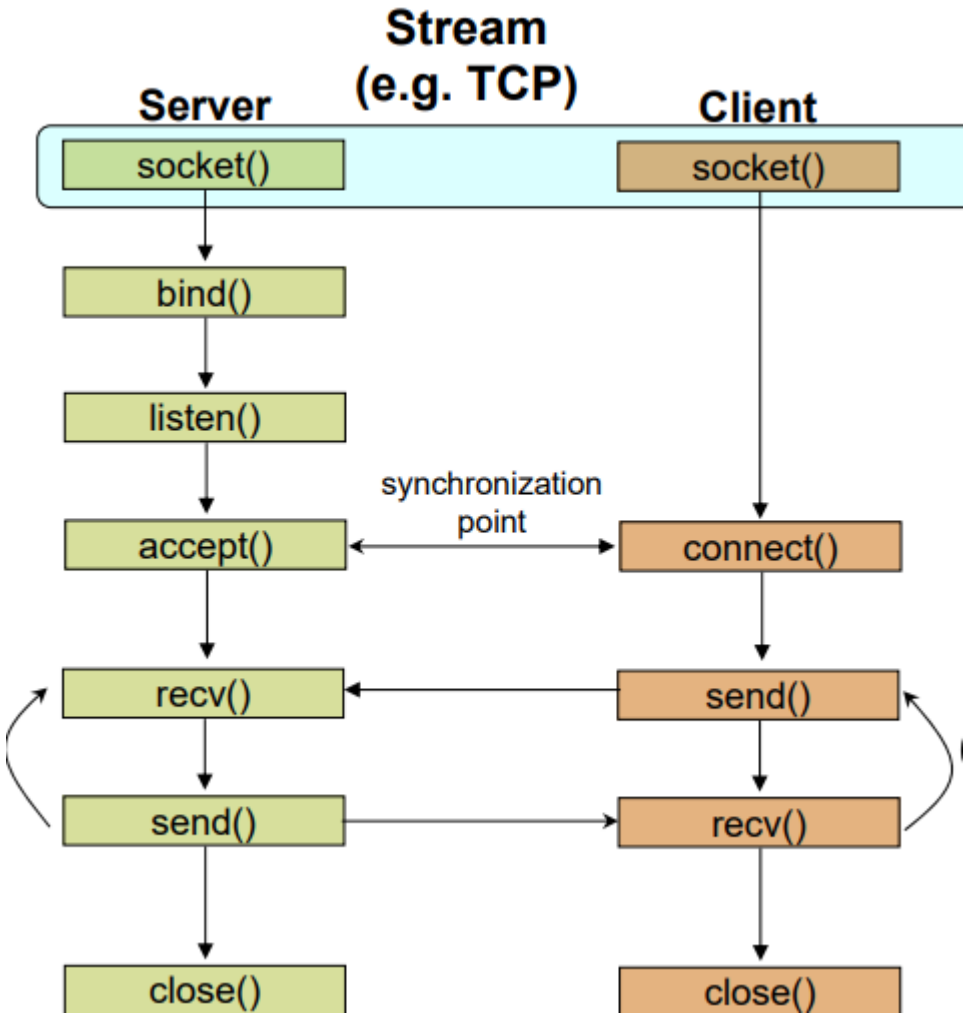
Definition: TCP is a connection-oriented protocol that ensures reliable and ordered data transfer between two endpoints. When using TCP sockets, a persistent connection is established, and data is transmitted after a successful connection handshake.

- **TCP communication involves a three-way handshake**:

    1. SYN: The client sends a synchronization (SYN) message to the server.
    2. SYN-ACK: The server responds with an acknowledgment (SYN-ACK).
    3. ACK: The client sends a final acknowledgment (ACK), establishing the connection.
- **Key Features of TCP**:

    1. Reliable Transmission: TCP ensures that data is delivered to the destination. If packets are lost or corrupted, TCP automatically retransmits them.
    2. Connection-Oriented: A connection must be established between the client and server before data transmission can occur.
    3. Ordered Data Delivery: TCP ensures that packets are received in the order they were sent, making it ideal for applications requiring sequential data.
    4. Error Checking: TCP provides mechanisms to check for errors in transmitted data using checksums.

- Diagram :- (Client - Server Communication – Unix)



❖ **TCP Server:**

A **TCP server** is a program or system that listens for incoming connection requests from **TCP clients** and engages in reliable, two-way communication using the **Transmission Control Protocol (TCP)**. Unlike a TCP client, the server waits for clients to connect and manages multiple client connections as needed.

Here's a step-by-step breakdown of the operations of a TCP server:

Steps for a TCP Server:

1. **socket()**: The server creates a **socket** to establish a communication endpoint.
2. **bind()**: The server assigns (binds) the socket to a specific **IP address** and **port number**.

3. **listen()**: The server **listens** for incoming connection requests from clients.

4. **accept()**:The server **accepts** an incoming connection from a client

5. **recv()** and **send()**: Once the connection is established, the server and client can exchange data using recv() to receive and send() to transmit.

6. **close()**: Once the server is finished communicating with the client, it closes the dedicated client socket, freeing up resources.

❖ TCP Client:

A **TCP client** is a program or system that initiates a connection to a **TCP server** to send and receive data using the **Transmission Control Protocol (TCP)**. TCP provides reliable, ordered, and error-checked delivery of data between applications running on different hosts on a network.

Here's a step-by-step breakdown of the typical operations of a TCP client:

1. **socket()**: The client first creates a **socket**. A socket is an endpoint for sending or receiving data over a network.

2. **connect()**: The client establishes a connection to the **server** by specifying the server's IP address and port number.

3. **send()**: After the connection is established, the client can start **sending data** to the server.

4. **recv()**: The client waits to **receive a response** from the server.

5. **close()**: Once the client has finished communicating with the server, it **closes** the connection.

➢ TCP Advantages

1. **Reliability**: Ensures accurate and complete data delivery with error detection and retransmissions.
2. **Order Preservation**: Maintains the sequence of data packets.
3. **Flow Control**: Prevents overwhelming the receiver.

➢ TCP Disadvantages

1. **Overhead**: Adds extra data and processing due to its control features.
2. **Latency**: Introduces delays due to connection setup and error recovery.
3. **Resource Intensive**: Consumes more system resources for connection management.
4. **Not Ideal for Real-Time**: Can cause delays in real-time applications like video streaming.
5. **Congestion Control**: Adjusts data transmission based on network congestion.

❖ Conclusion :

To conclude, sockets provide a highly efficient method for establishing strong communication. Inter-process communication (IPC) within the same host has strengthened Unix socket programming in C. It enables the creation of reliable, high-performance communication channels for processes running on the same machine using Unix domain sockets. The client starts a TCP socket, sets up the server address, and connects to receive messages.

Name & Sign of Course Teacher
Mrs.Prajakta DSawle