Steps to Write Client-Server Code for TCP and UDP

TCP Client-Server

Server-Side

1. Create a Socket (Outside while loop): Use socket() to create a TCP socket.

2. Bind the Socket (Outside while loop): Use bind() to bind the socket to an IP address and port.

3. Listen for Connections (Outside while loop): Use listen() to allow the server to listen for incoming connections.

4. Accept Connections (Inside while loop): In the while loop, use accept() to accept new connections. A new socket for communication with the specific client is created here.

5. Receive and Send Data (Inside while loop): Use read() or recv() to receive data from the client. Process the data and use write() or send() to send a response.

6. Close Client Connection (Inside while loop): After communication, close the client socket. The main listening socket remains open.

7. Close Listening Socket (Outside while loop): Close the listening socket when the server shuts down.

Client-Side

1. Create a Socket (Outside while loop): Use socket() to create a TCP socket.

2. Connect to the Server (Outside while loop): Use connect() to establish a connection to the server's IP and port.

3. Send and Receive Data (Inside while loop): Use write() or send() to send data to the server. Use read() or recv() to receive data from the server.

4. Close Connection (Outside while loop): Close the socket after communication.

Concurrent TCP Server

1. Create a Socket (Outside while loop): Use socket() to create a TCP socket.

2. Bind and Listen (Outside while loop): Bind the socket using bind() and start listening using listen().

3. Accept Connections (Inside while loop): Use accept() to accept new connections in the while loop.

4. Fork a New Process (Inside while loop): For each accepted client, fork() a new child process. The child process handles the client, while the parent process continues listening for more connections.

5. Handle Client (Inside the child process): Receive data from the client, process it, and send the response back.

6. Prevent Zombie Processes: Use signal(SIGCHLD, SIG_IGN) to prevent zombie child processes by ignoring the SIGCHLD signal.

7. Close Client and Parent Sockets: The child process closes the client socket when done, and the parent process closes the new client socket after forking.

UDP Client-Server

Server-Side

1. Create a Socket (Outside while loop): Use socket() to create a UDP socket.

2. Bind the Socket (Outside while loop): Use bind() to bind the socket to an IP address and port.

3. Receive Data (Inside while loop): Use recvfrom() to receive data from a client. This function provides both the data and the client's address.

4. Process Data (Inside while loop): Perform any necessary processing on the received data.

5. Send Data Back (Inside while loop): Use sendto() to send the processed data back to the client using the client's address.

6. Close the Socket (Outside while loop): Close the UDP socket after the server is done.

Client-Side

1. Create a Socket (Outside while loop): Use socket() to create a UDP socket.

2. Send Data to Server (Inside while loop): Use sendto() to send data to the server's IP and port.

3. Receive Data from Server (Inside while loop): Use recvfrom() to receive data from the server.

4. Close the Socket (Outside while loop): Close the socket after communication.