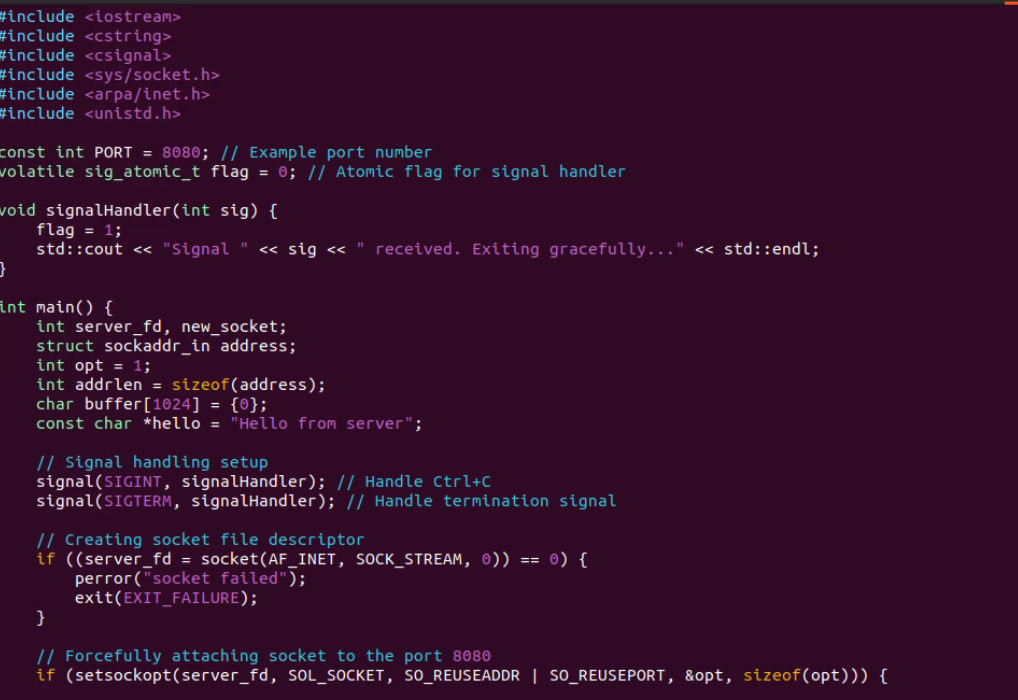
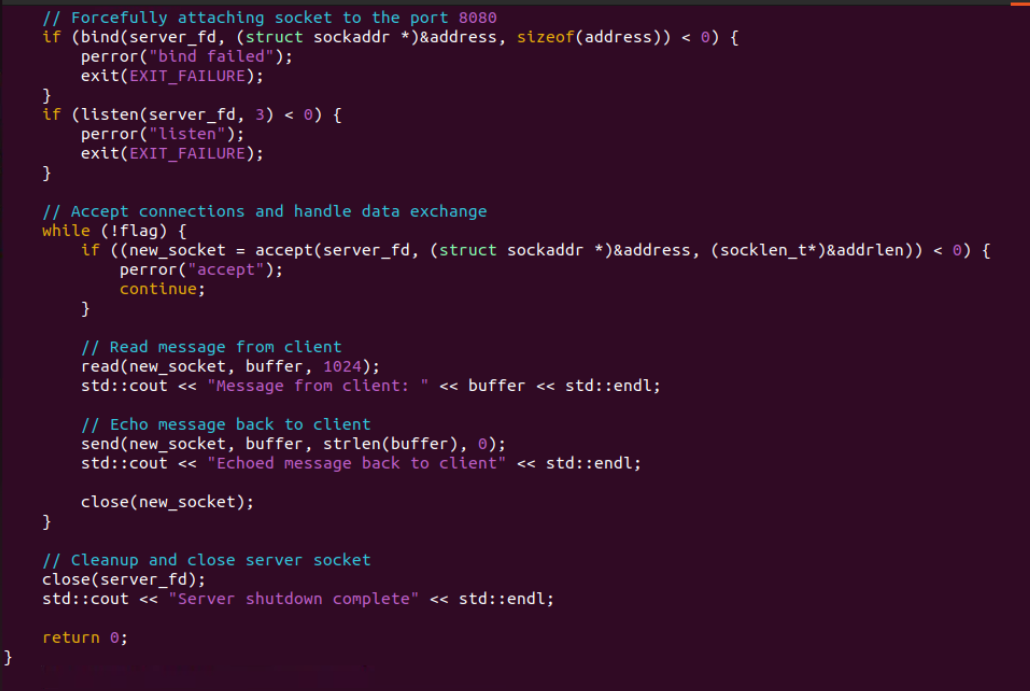
1.Objective:

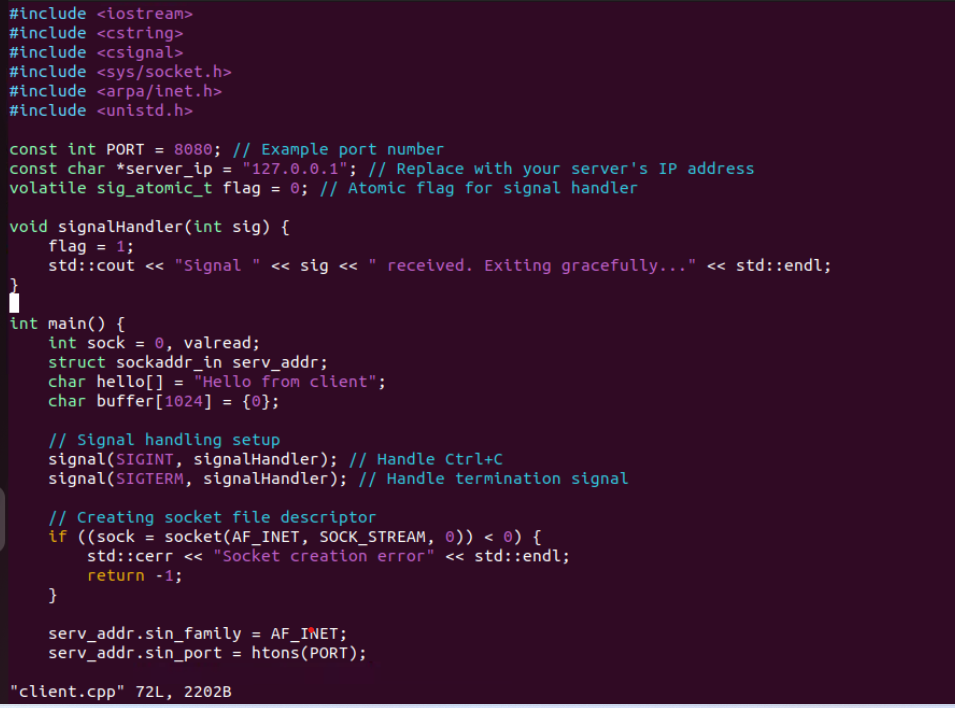
Create a C++ application that combines signal handling and socket programming to manage network communication while gracefully handling interruptions (e.g., SIGINT for program termination). The application should be capable of sending and receiving messages over a network while responding appropriately to system signals.

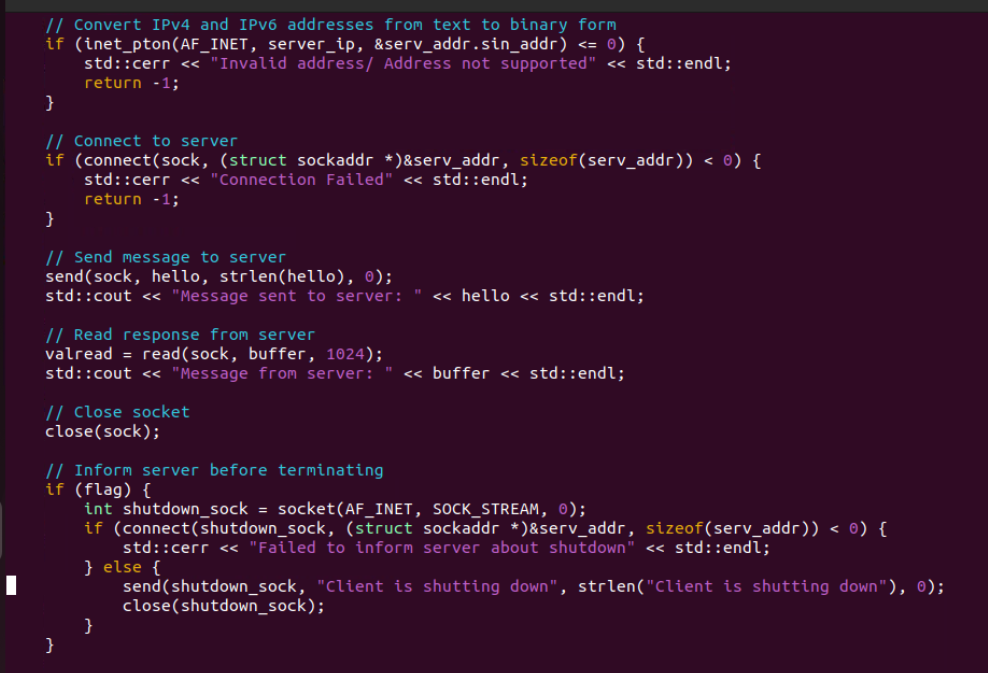
Requirements:SocketProgramming:Implement a TCP server that listens for incoming connections on a specified port. Implement a TCP client that connects to the server and exchanges messages.Signal Handling:

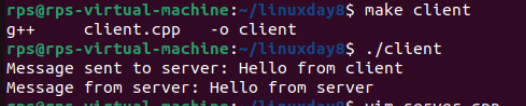
Implement signal handlers for SIGINT (Ctrl+C) and SIGTERM to gracefully shut down the server and client.Ensure that the program can handle interruptions without crashing or leaving resources unfreed.DataExchange:The client should be able to send a message to the server.The server should echo the received message back to the client.GracefulShutdown:When the server receives a SIGINT or SIGTERM signal, it should close all active connections and free resources before terminating.When the client receives a SIGINT or SIGTERM signal, it should inform the server before terminating.

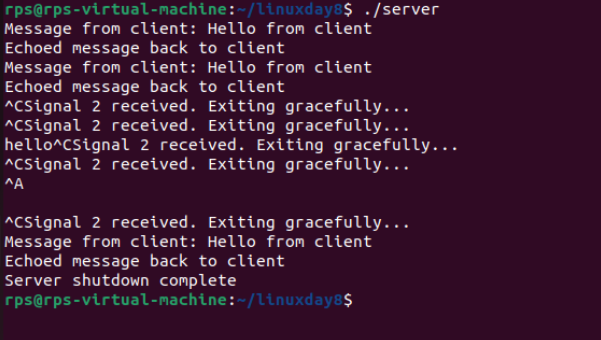






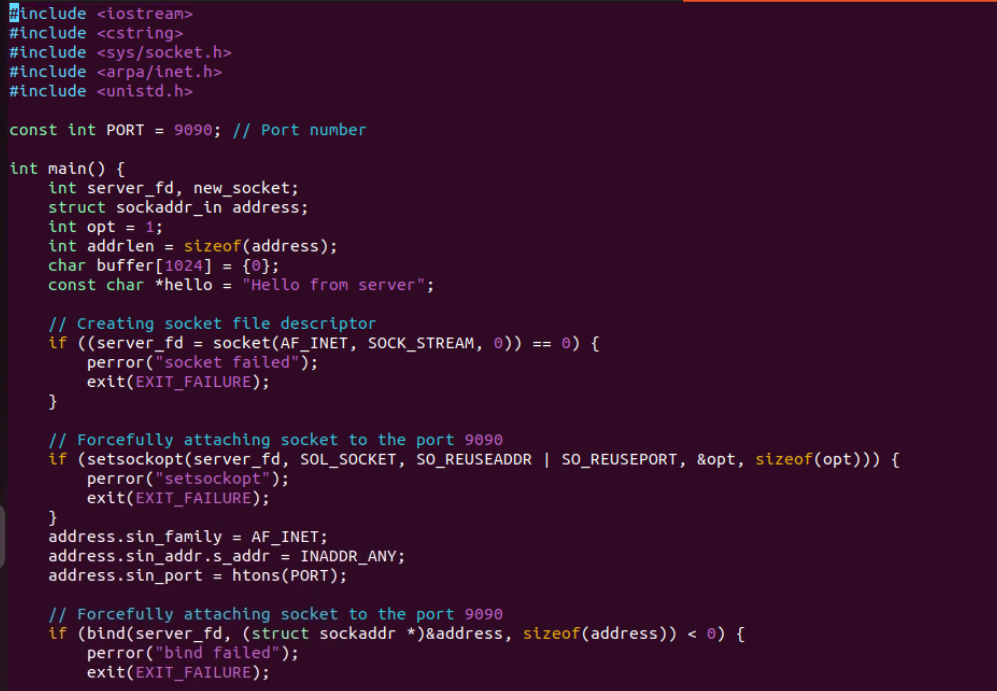


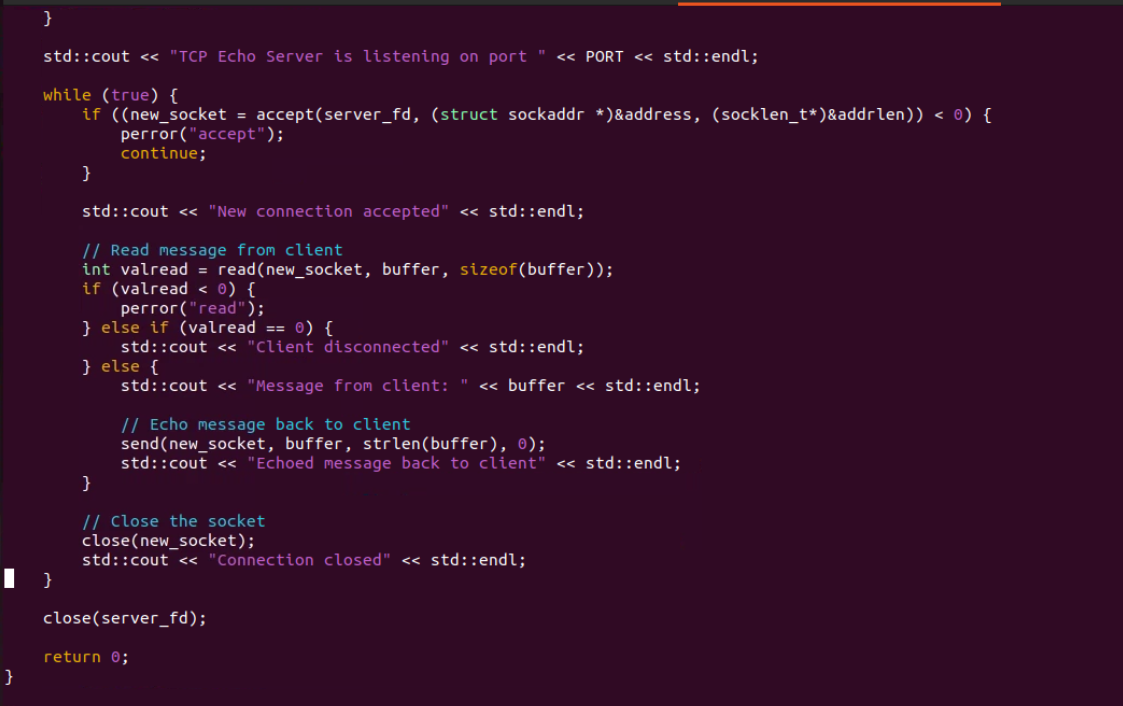




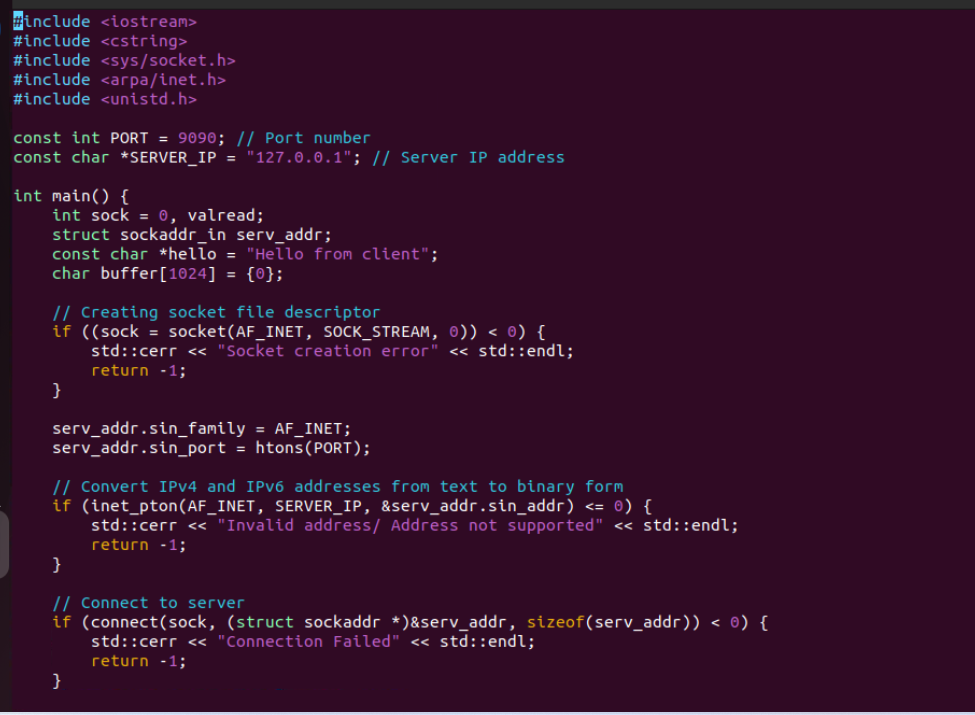
2.CP Echo Server:Implement a TCP server that:Binds to port 9090.Listens for incoming connections.Accepts a connection from a client.Receives a message from the client and echoes the same message back to the client.Closes the connection and terminates.UDP Client-Server Communication:

Server:





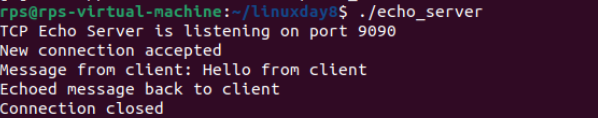
Client:





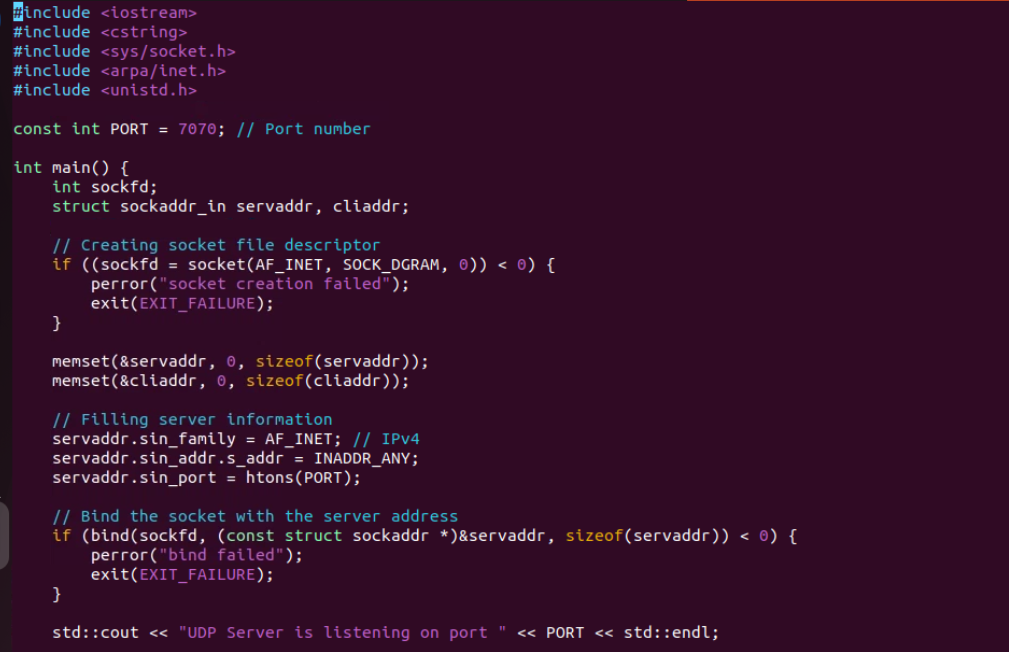
Output:

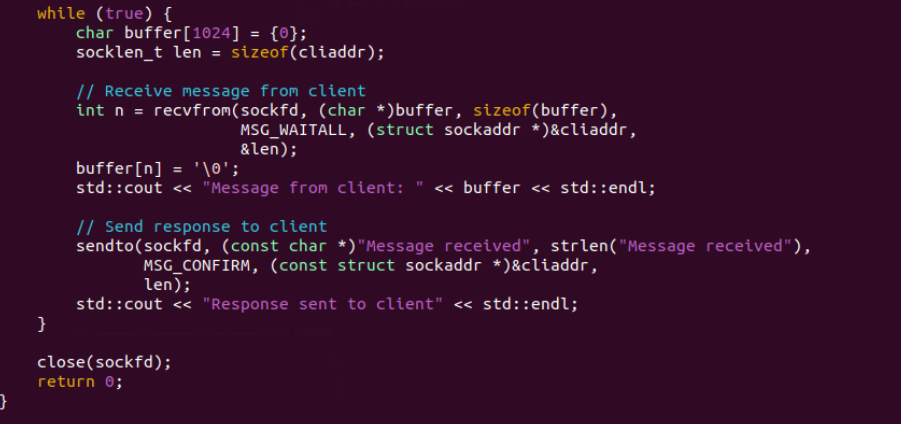




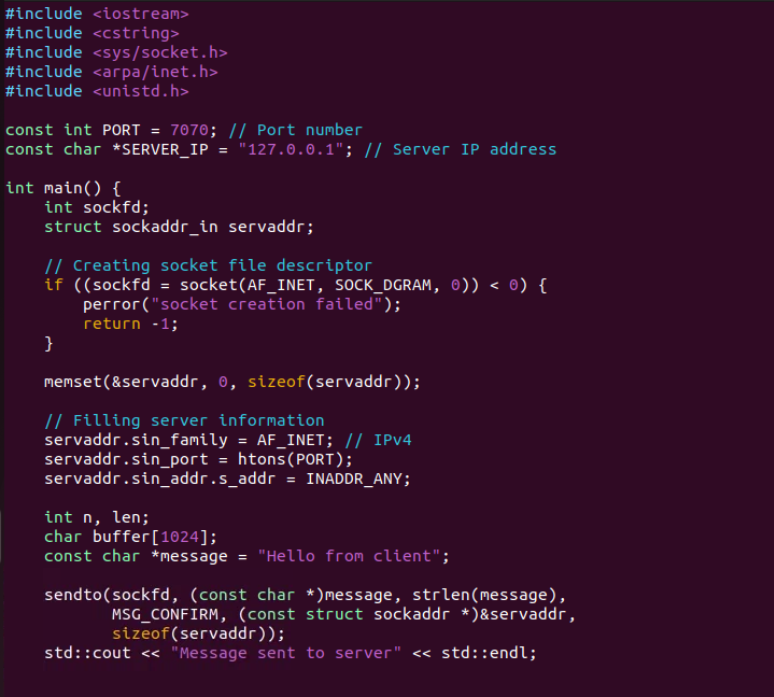
3.Create a UDP server that:Binds to port 7070.Receives a message from a client.Sends a response message back to the client.Closes the socket and terminates.Create a UDP client that:Sends a message to the server on port 7070.Receives and prints the response message from the server.Closes the socket and terminates.Multi-Client TCP Server:

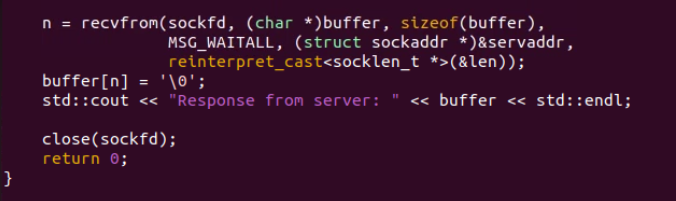
Server:



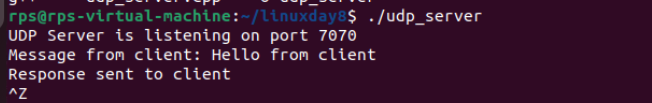


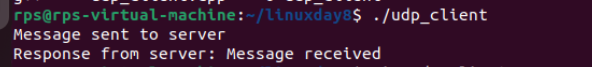
Client:





Output:





4.Implement a TCP server that:

Binds to port 6060.

Listens for incoming connections.

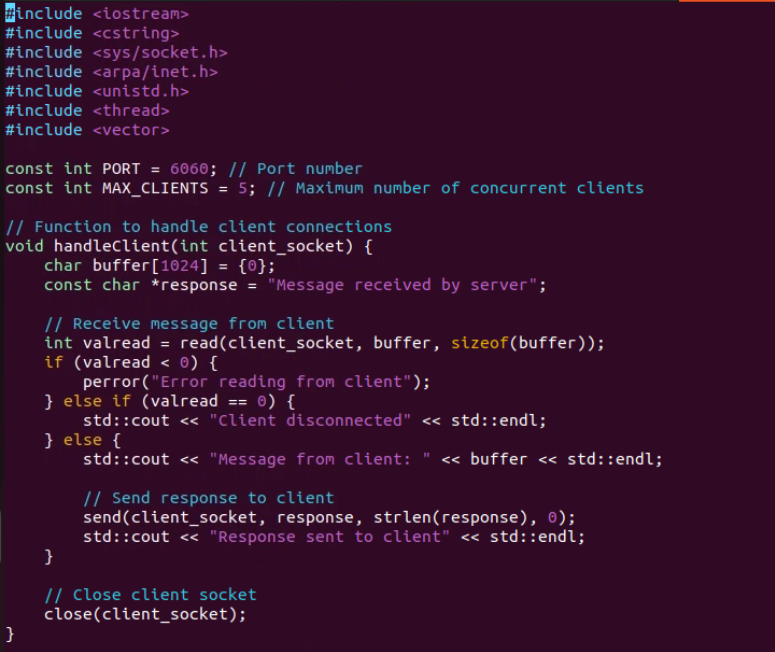
Accepts multiple client connections concurrently.

Receives a message from each client and sends a unique response back.

Closes the connections and terminates after handling all clients.

TCP Client with Error Handling:

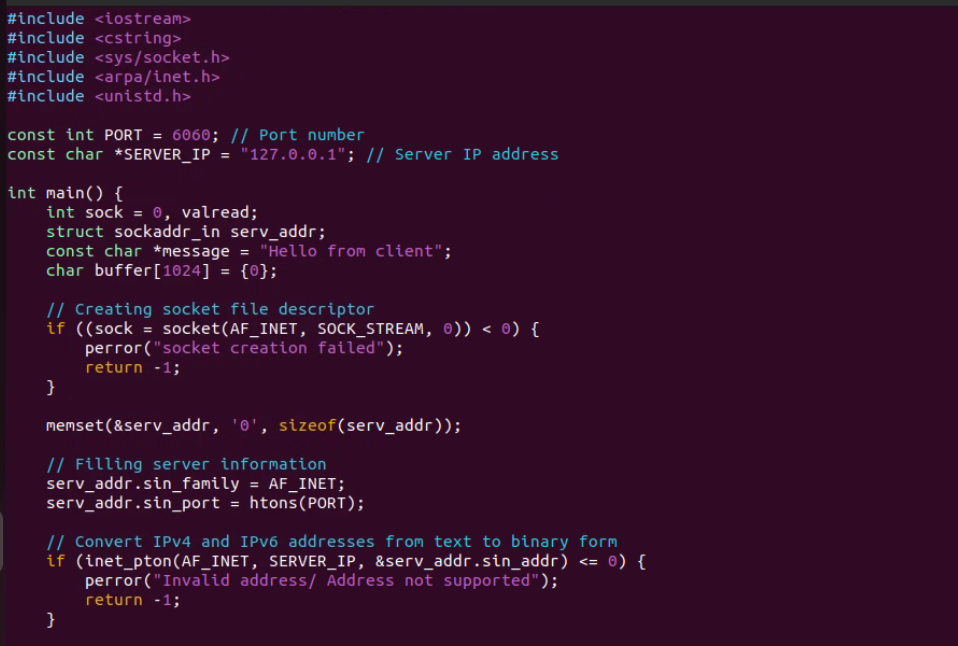
Tcp\_server:

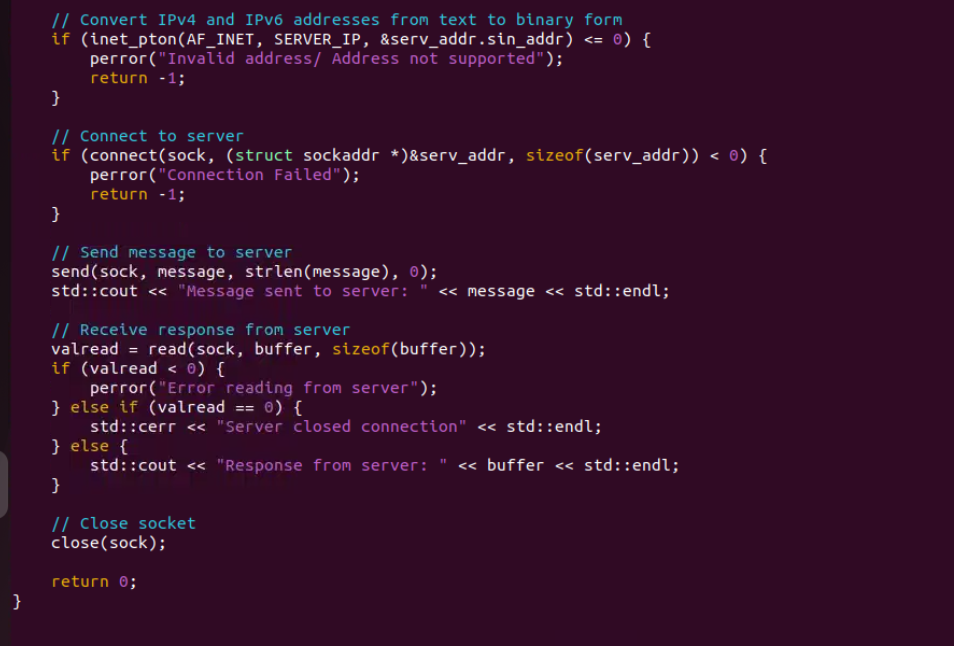






Tcp\_client:





Output:

