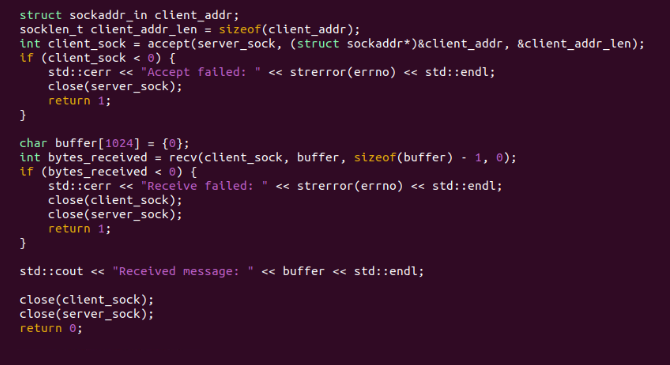
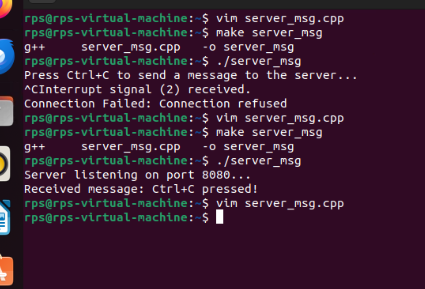
Try to write a Code where when user put ctrl c

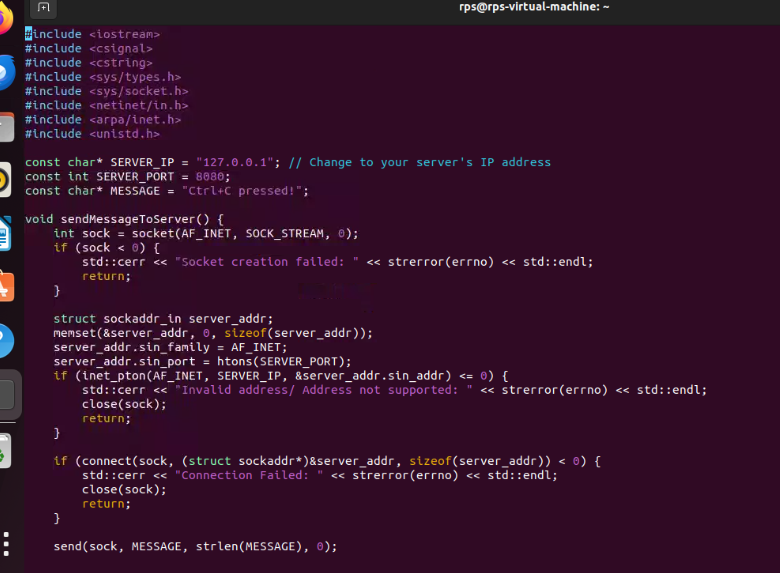
Message delivered to server

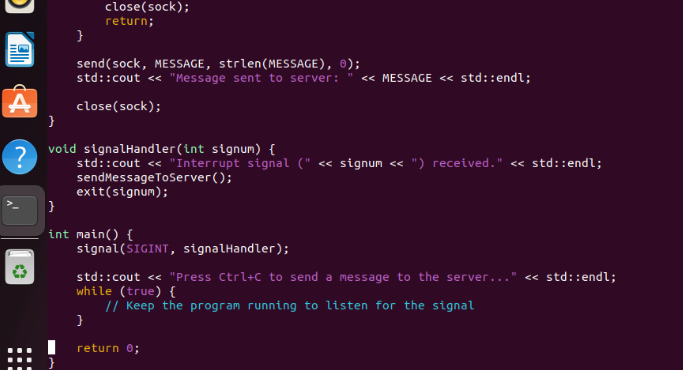


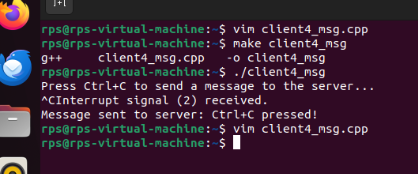




Server part.

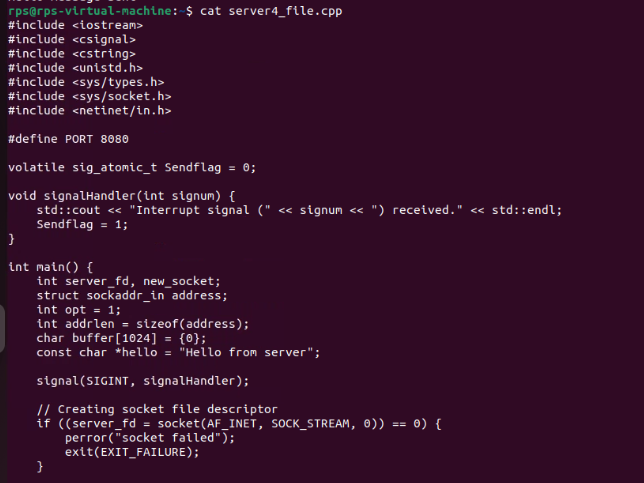


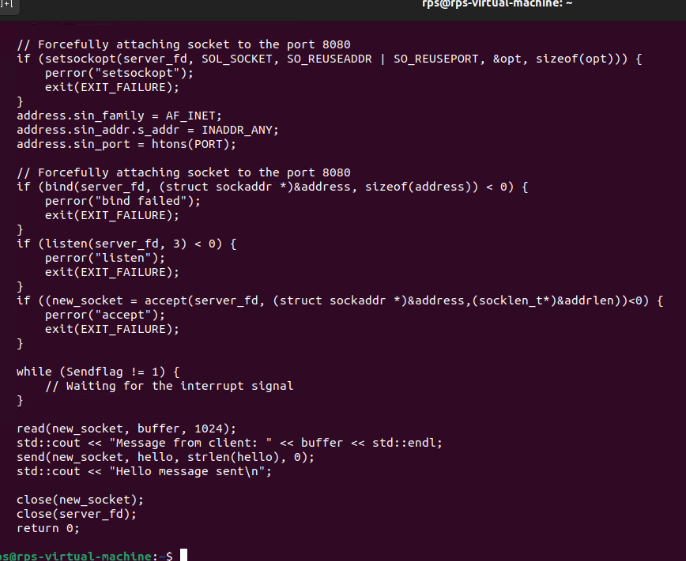


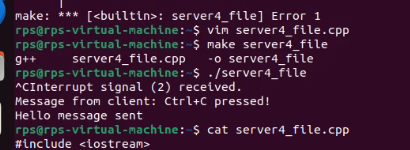


Client part.

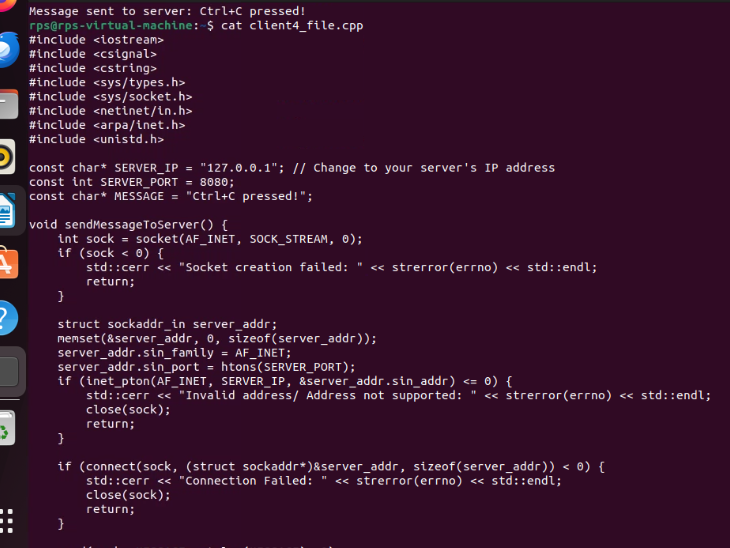
Serever part:

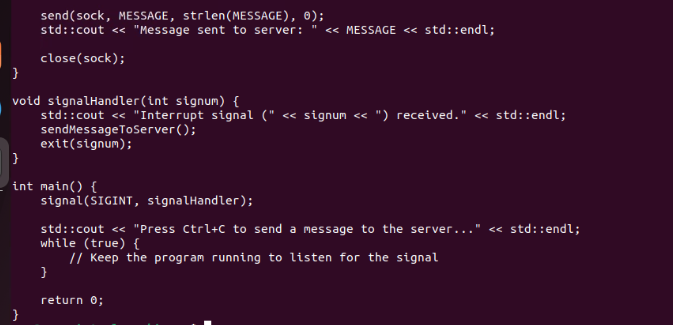


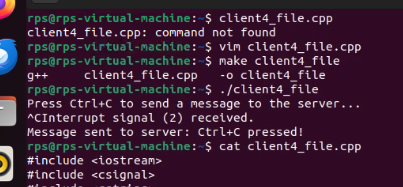




Client part:







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:

Socket Programming:

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.

Data Exchange:

The client should be able to send a message to the server.

The server should echo the received message back to the client.

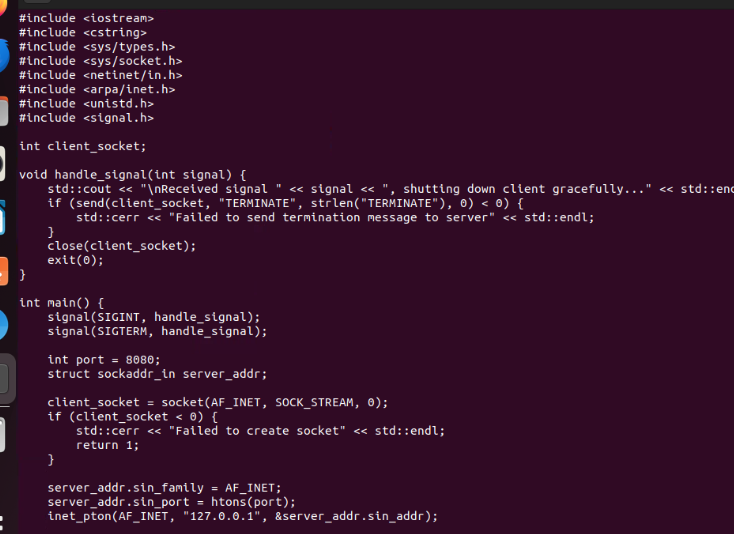
Graceful Shutdown:

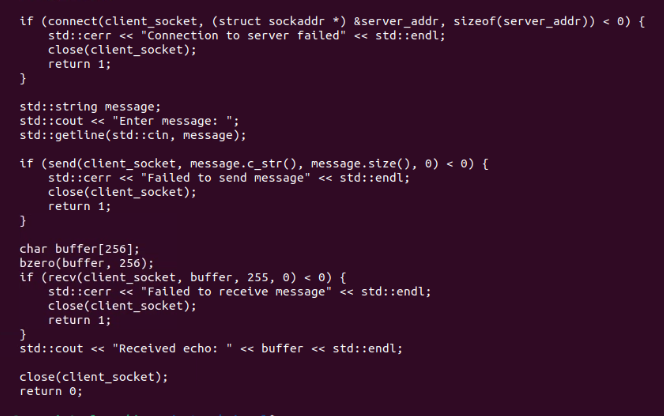
When the server receives a SIGINT or SIGTERM signal, it should close all active connections and free resources before terminating.

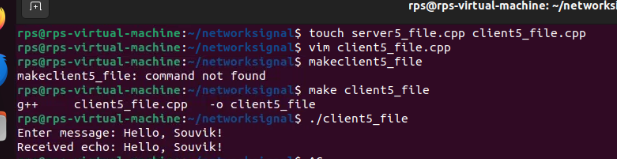
When the client receives a SIGINT or SIGTERM signal, it should inform the server before terminating.

Answer:

Client Part:

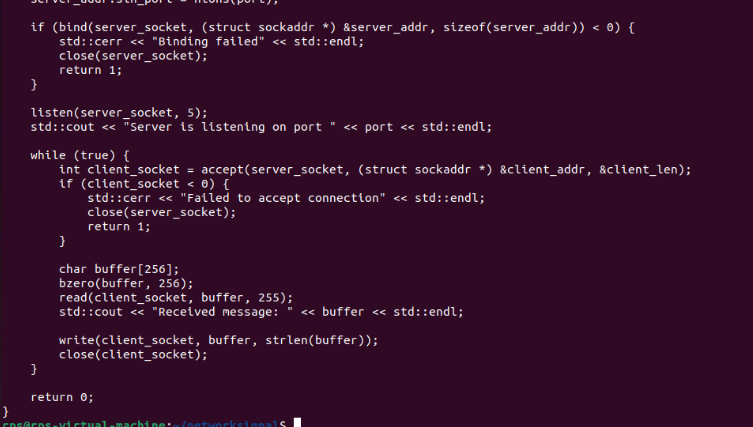


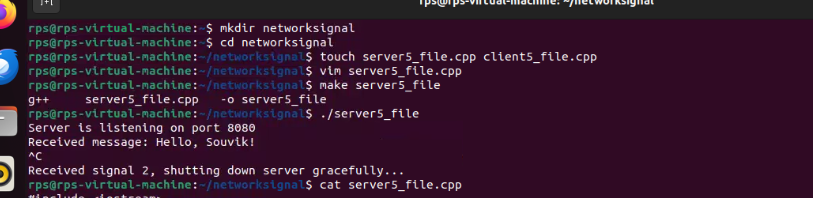




Server Part:



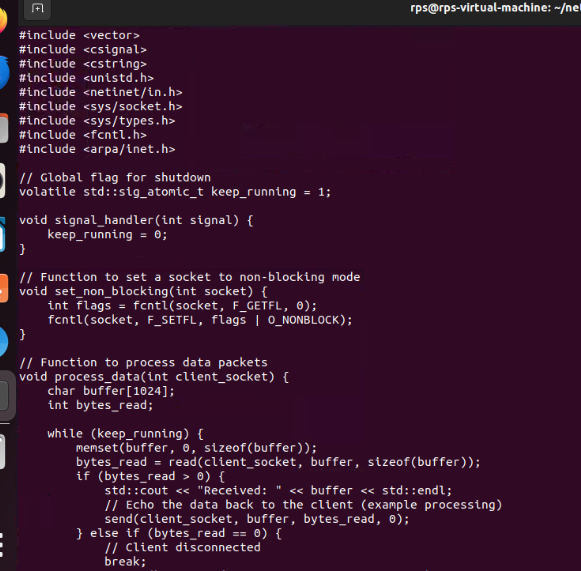


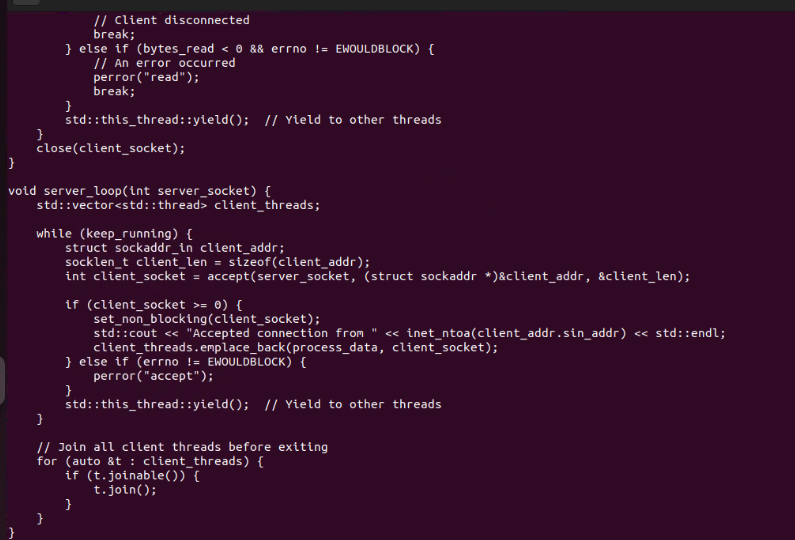


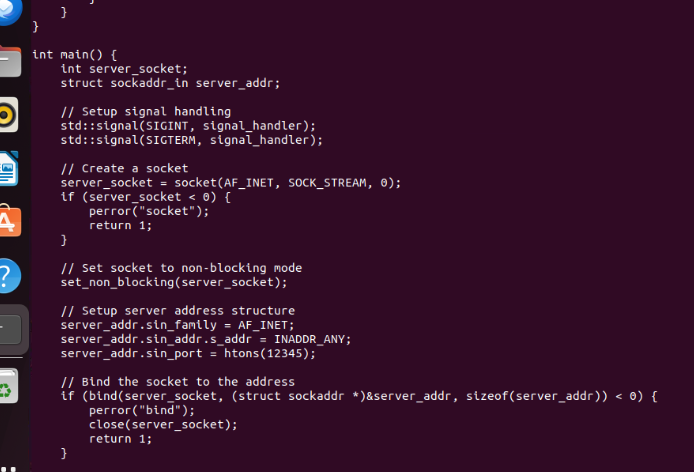
Problem 4: Real-time Data Processing Server

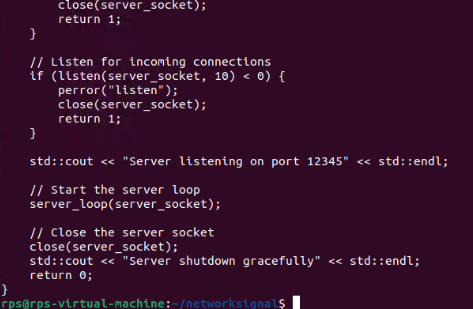
Implement a real-time data processing server in C++ that receives data packets from multiple clients over TCP sockets. The server should process the data in real-time and send back responses. Handle signals to gracefully terminate the server and ensure that all in-flight data processing is completed before shutdown.

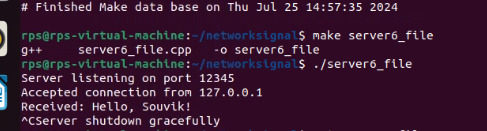
Server Part:



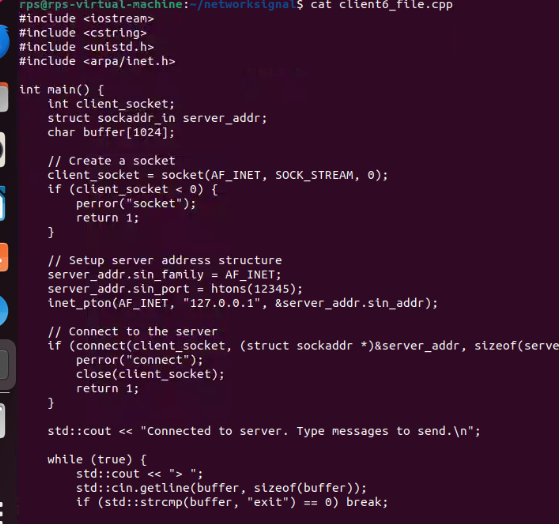


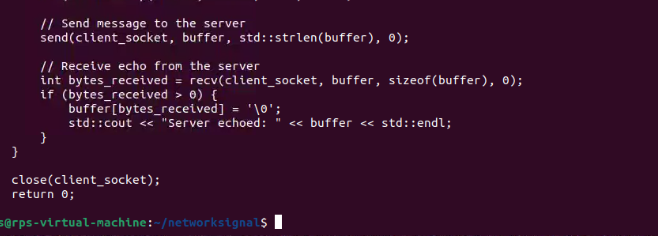


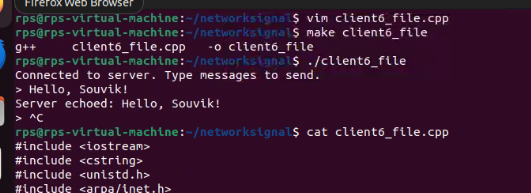




Client Part:



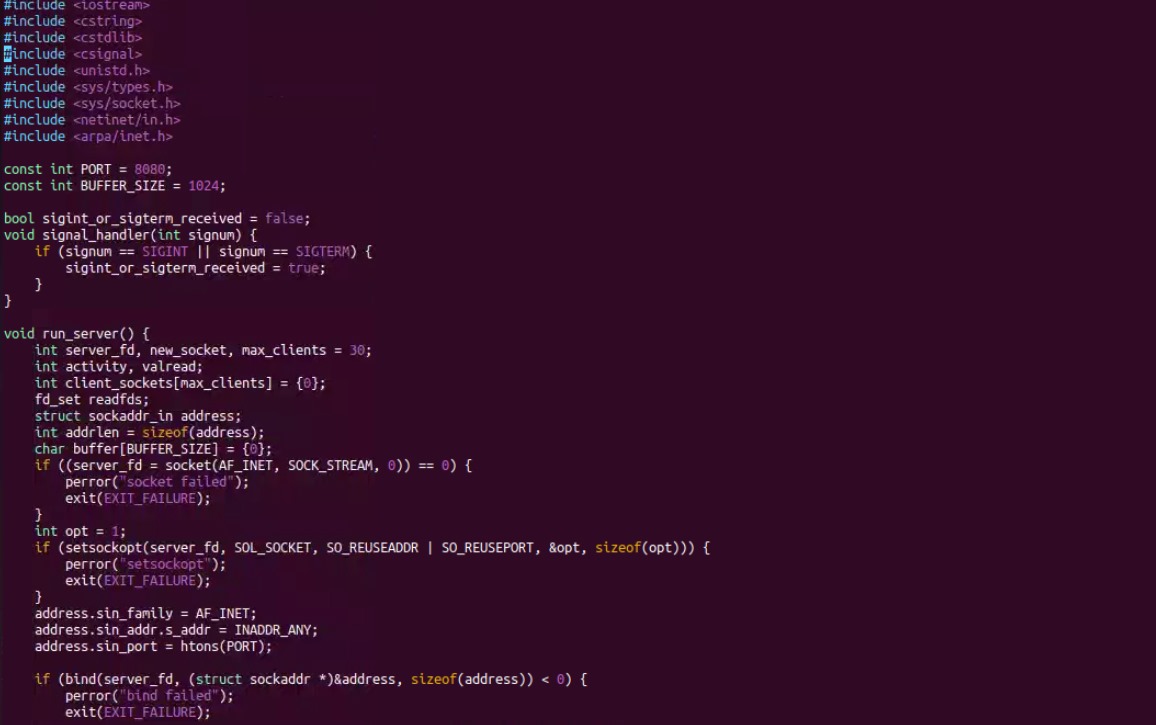


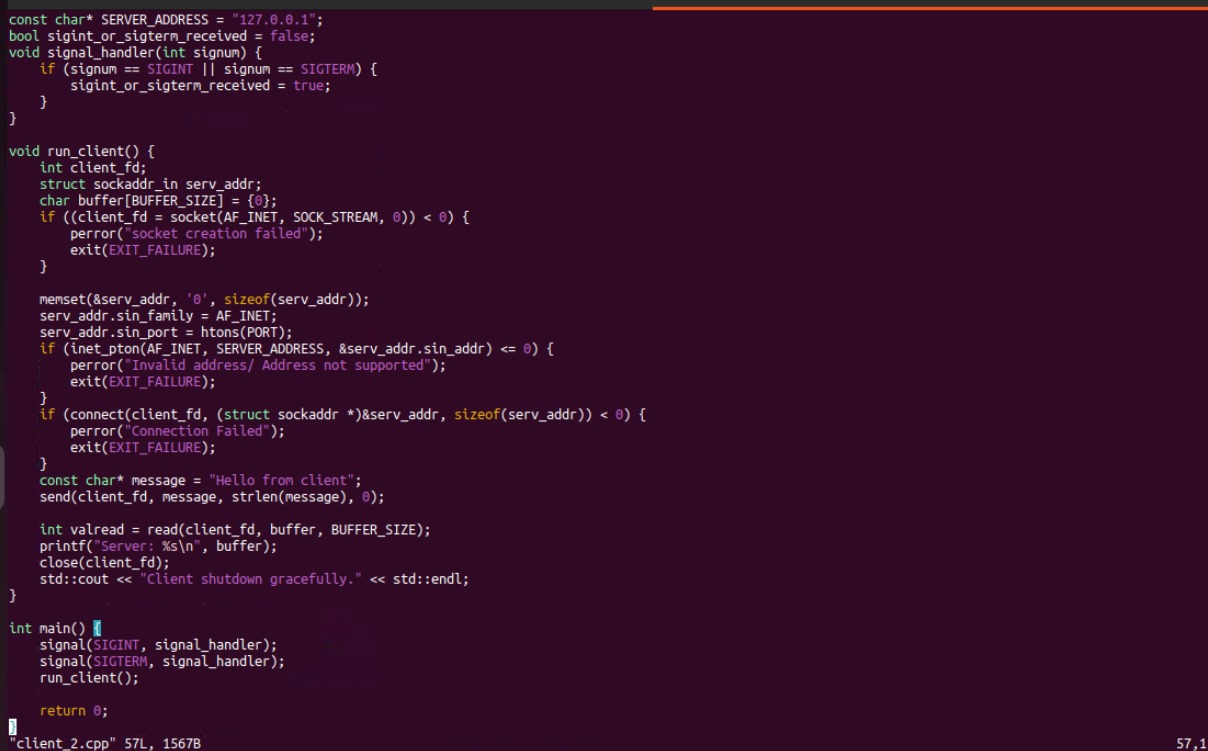


Problem 3: Asynchronous I/O with Signals

Create a C++ program that uses asynchronous I/O operations for reading from and writing to a socket. Implement signal handling to manage program interruptions and ensure that all pending I/O operations are completed or properly canceled before the program exits.

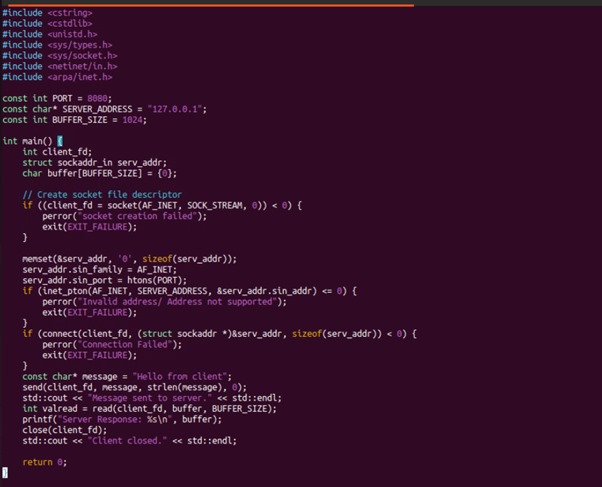
Server part:

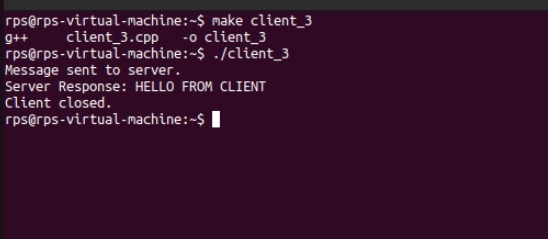






Client Part:





CP Echo Server:

Implement a TCP server that:

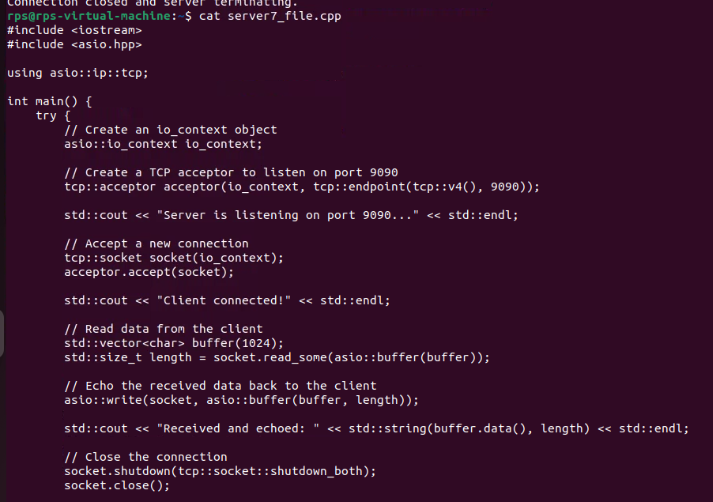
Binds to port 9090.

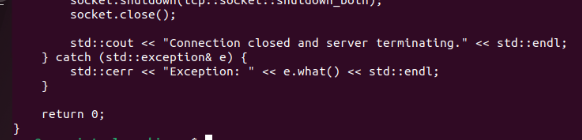
Listens for incoming connections.

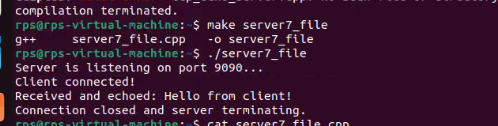
Accepts a connection from a client.

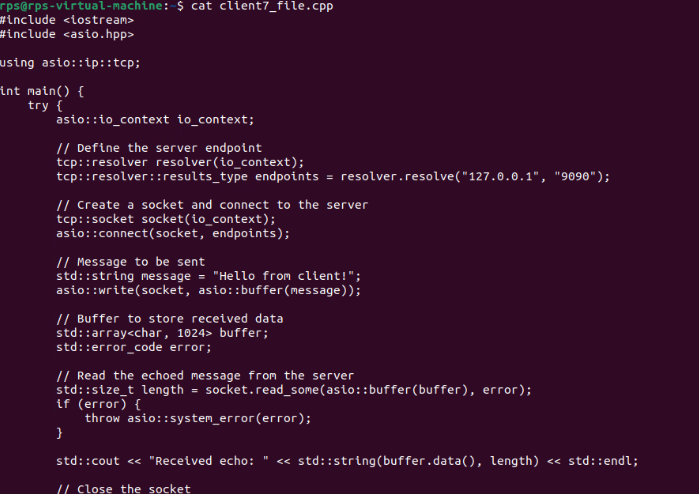
Receives a message from the client and echoes the same message back to the client.

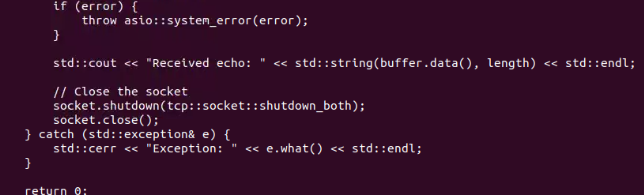
Closes the connection and terminates.

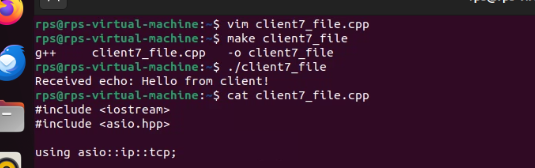












UDP Client-Server Communication:

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:

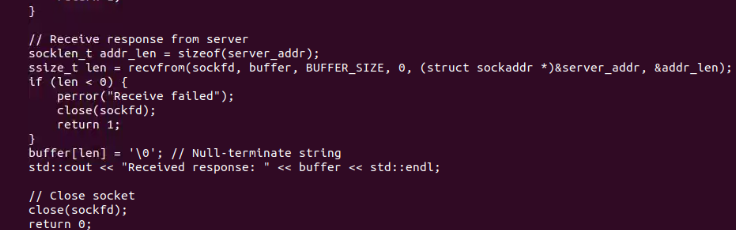
Sends a message to the server on port 7070.

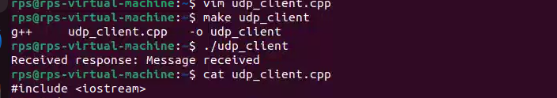
Receives and prints the response message from the server.

Closes the socket and terminates.

Client Part:







Server Part:

