TCP Server-Client Communication:

Problem Statement: Write a TCP server and client program in C where the server listens for incoming connections and echoes back any message it receives from the client. The client should be able to send a message to the server and display the echoed message.

Requirements:

The server should run indefinitely, waiting for client connections.

The client should take a message as input from the user, send it to the server, and display the response.

Implement proper error handling and cleanup (e.g., closing sockets).

Ans:

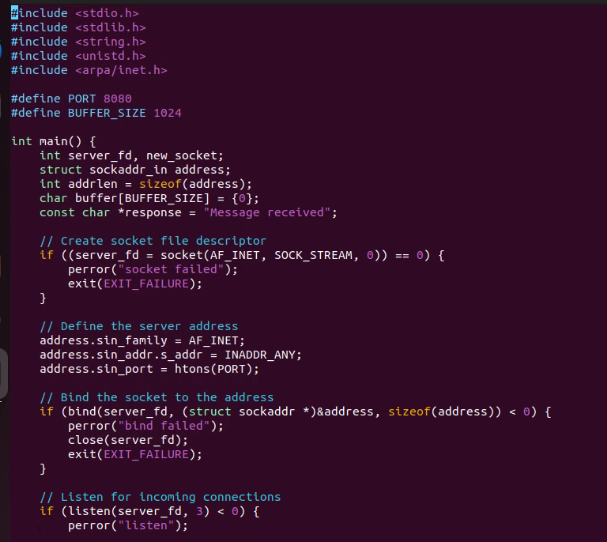
Server Code:

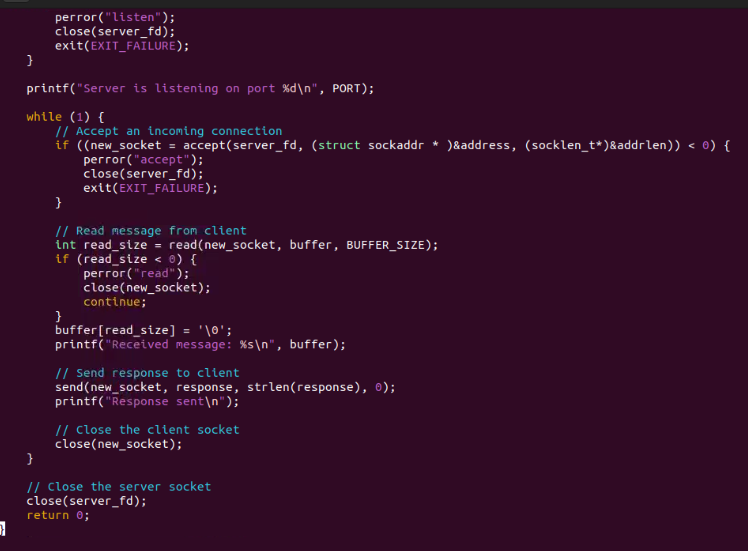
Creates a socket and binds it to an IP address and port.

Listens for incoming connections and accepts them.

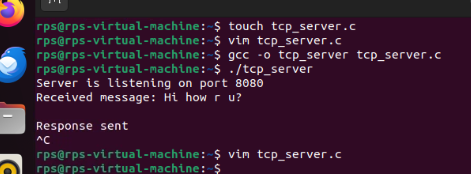
Reads data from the client, echoes it back, and closes the connection.

Server code:





Output:

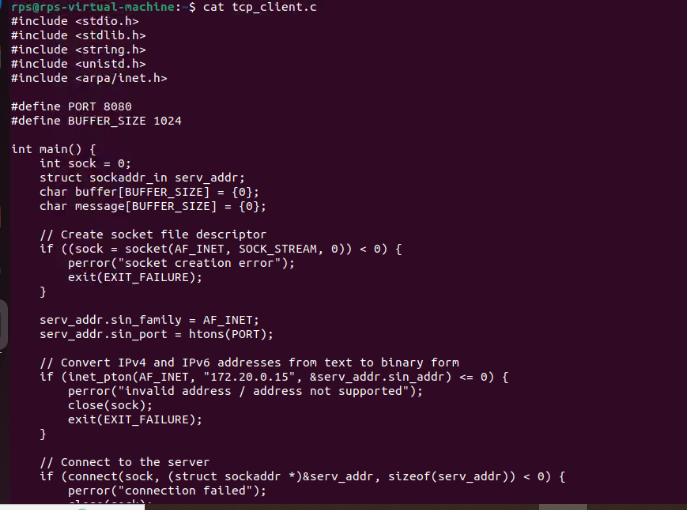


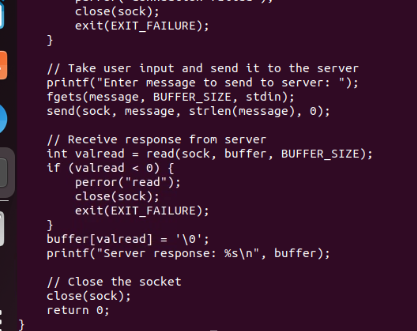
Client part:

Creates a socket and connects it to the server.

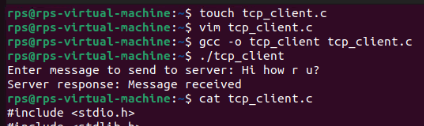
Sends a message to the server, reads the echoed message, and prints it.

Code:





Output:



UDP Server-Client Communication:

Problem Statement: Write a UDP server and client program in C where the server listens on a specific port and responds with "Hello, Client!" whenever it receives a message. The client should send a message to the server and print the response.

Requirements:

The server should run indefinitely, waiting for incoming messages.

The client should send a predefined message (e.g., "Hello, Server!") and display the server's response.

Implement proper error handling.

Ans:

Udp server:

The server will:

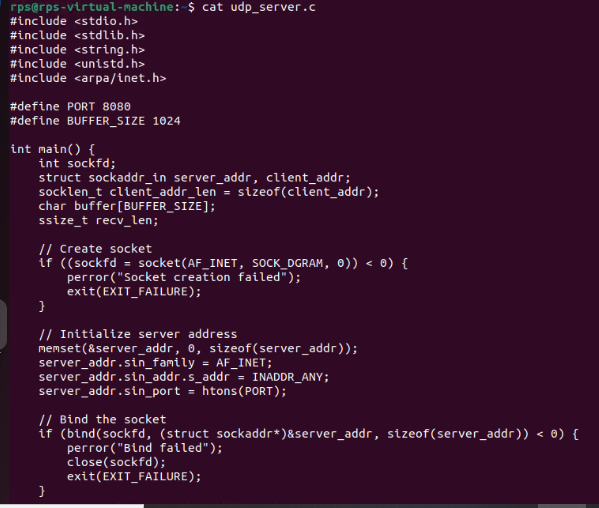
Create a UDP socket.

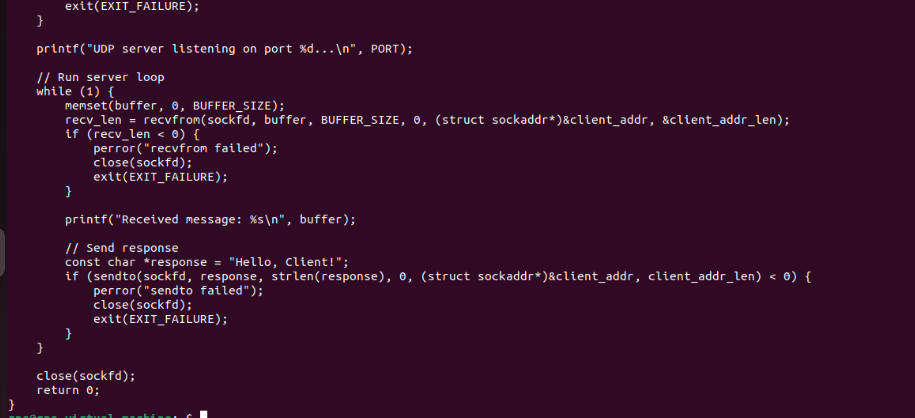
Bind the socket to a specific port.

Wait for incoming messages.

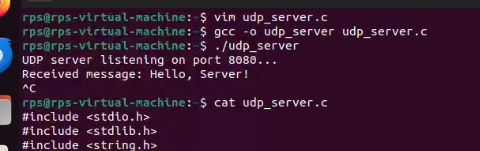
Send a response "Hello, Client!" to the client.

Code:





Output:



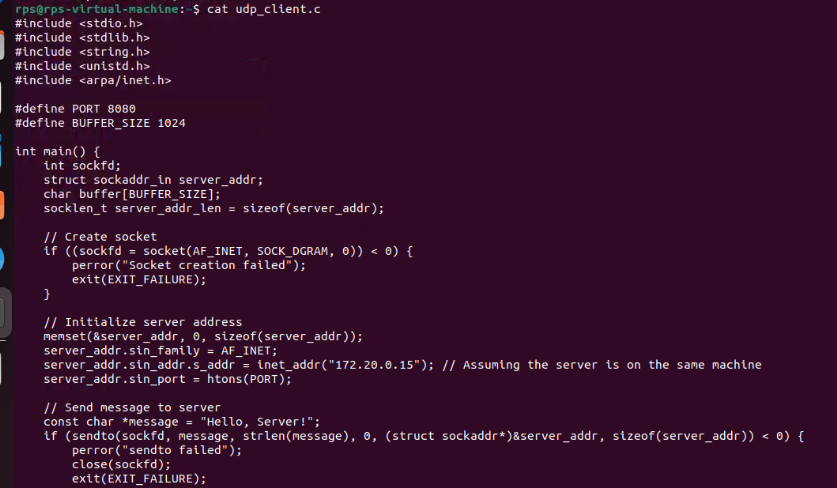
Udp client:

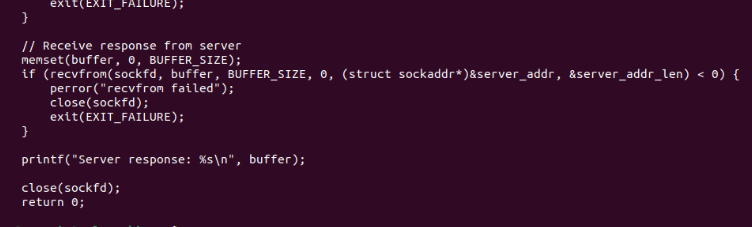
Create a UDP socket.

Send a predefined message "Hello, Server!" to the server.

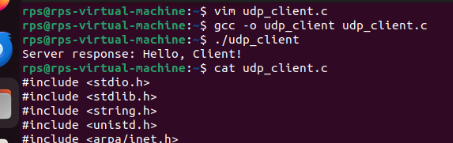
Receive and print the response from the server.

Code:





Output:



File Transfer using TCP:

Problem Statement: Write a TCP server and client program in C to transfer a file from the client to the server. The server should save the received file with the same name, and the client should specify the file to be sent.

Requirements:

The server should run indefinitely, waiting for file transfer requests.

The client should prompt the user for a file path, read the file, and send its contents to the server.

Server setup:

Create a TCP socket.

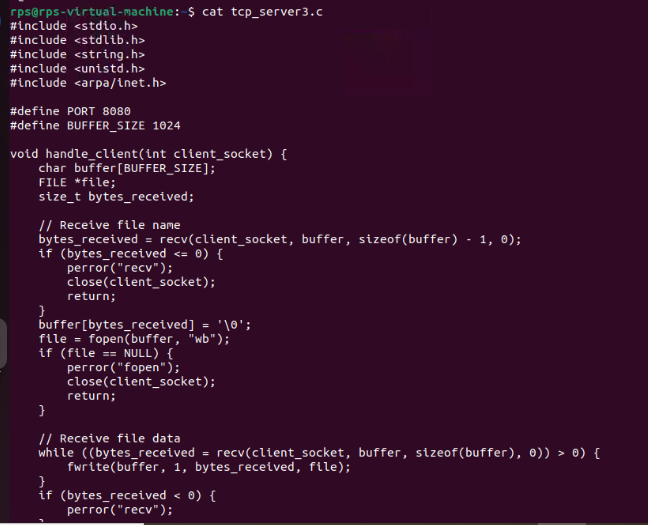
Bind the socket to the server address and port.

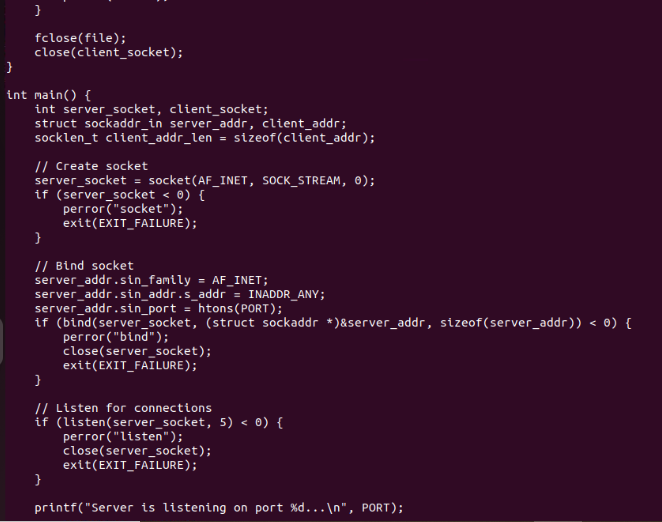
Put the server in listen mode to accept incoming connections.

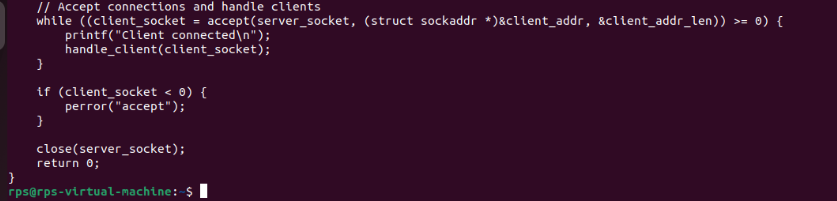
Accept an incoming connection from the client.

Read the filename first, then receive the file data and save it with the same name.

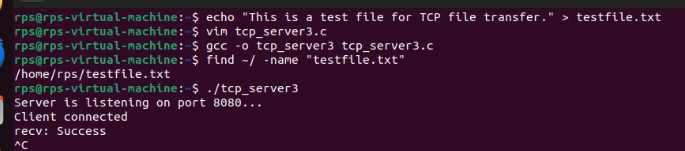
Code:







Output:



Client part:

Prompt the user for the file path to send.

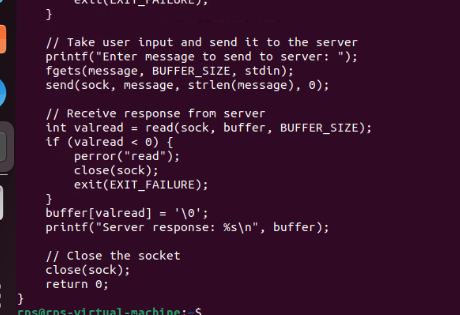
Create a TCP socket.

Connect to the server using its IP address and port.

Send the filename first, then read the file data and send it to the server in chunks.

Code:





Broadcast Messaging using UDP:

Problem Statement: Write a UDP server and client program in C to implement a simple broadcast messaging system. The server should broadcast a message to all clients in the network, and each client should display any broadcast messages it receives.

Requirements:

The server should send a broadcast message to a specific port.

Each client should listen on the same port and display any messages it receives.

Implement proper error handling and use UDP broadcast mechanisms.

Server part:

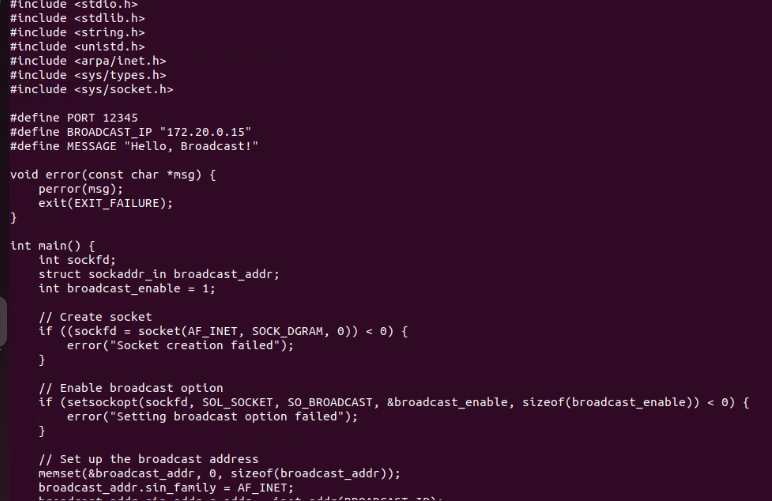
The server creates a UDP socket.

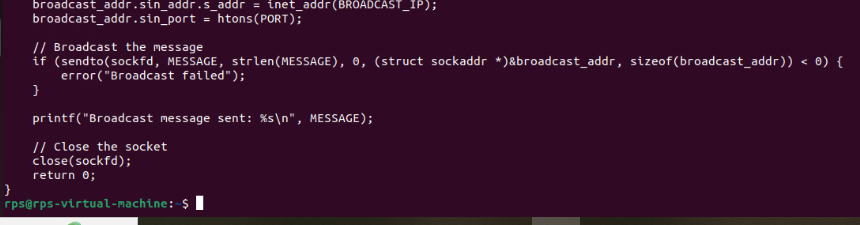
It sets the socket option to allow broadcast messages.

It constructs the broadcast address and port.

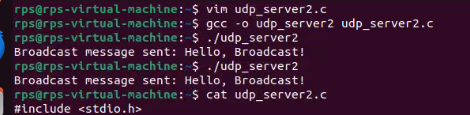
It sends the broadcast message to the broadcast address.

Code:





Output:



Client part:

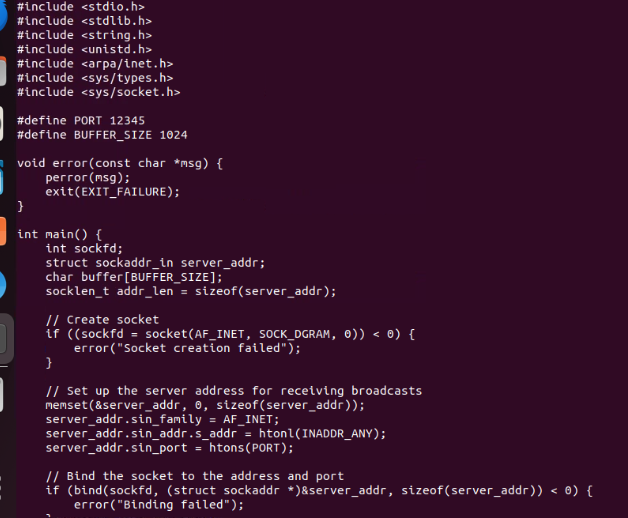
Each client creates a UDP socket.

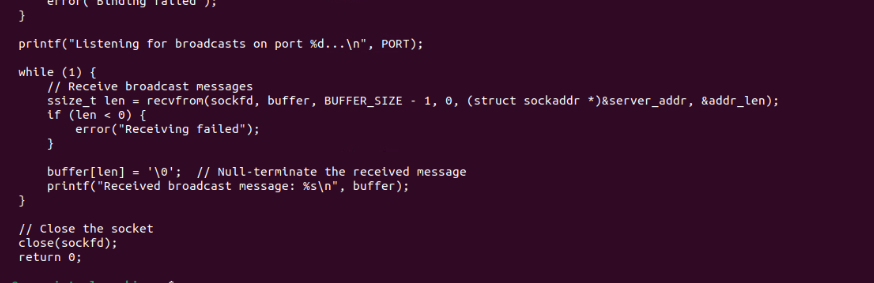
It sets the socket option to allow address reuse (so multiple clients can bind to the same port).

It binds the socket to the specific port to listen for incoming broadcast messages.

It receives and displays any broadcast messages it receives.

Code:





Output:

