DAY-30:

File Manipulation using System Calls in C++ on Linux

Objective:

Create a C++ program that performs file manipulation using Linux system calls. The program should be able to:

Create a new file.

Write a specified string to the file.

Read the contents of the file and display them on the console.

Append additional text to the file.

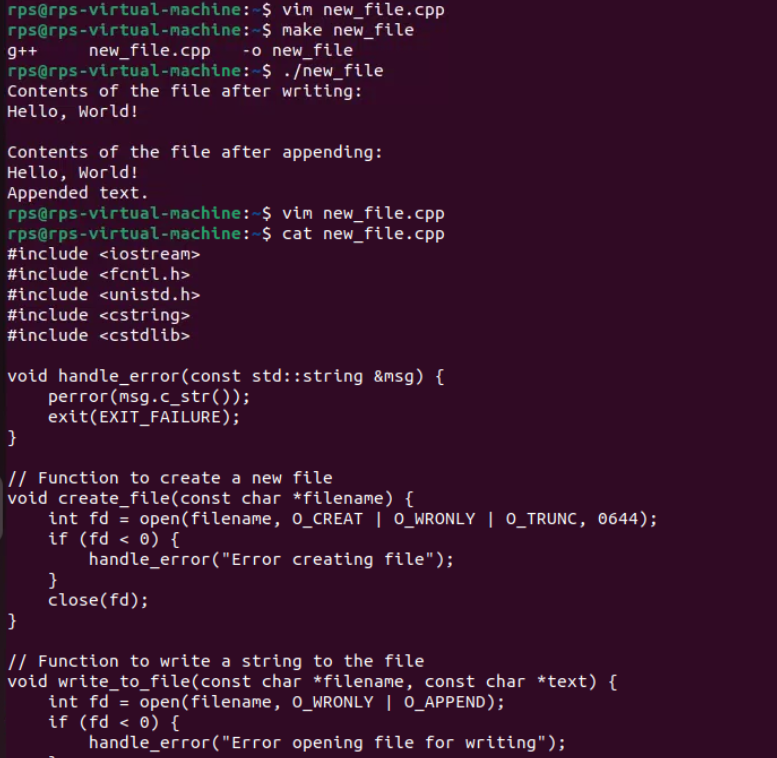
Delete the file.

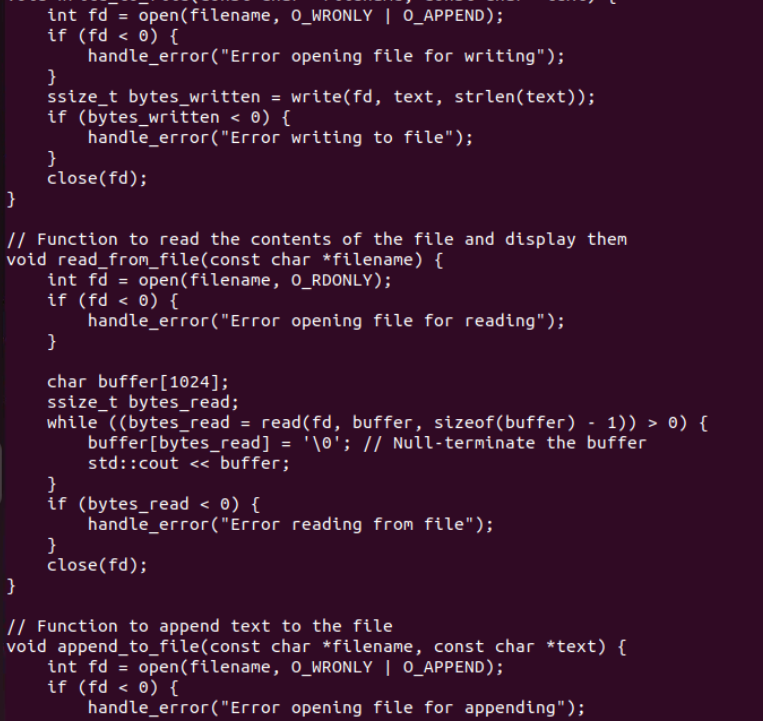
Requirements:

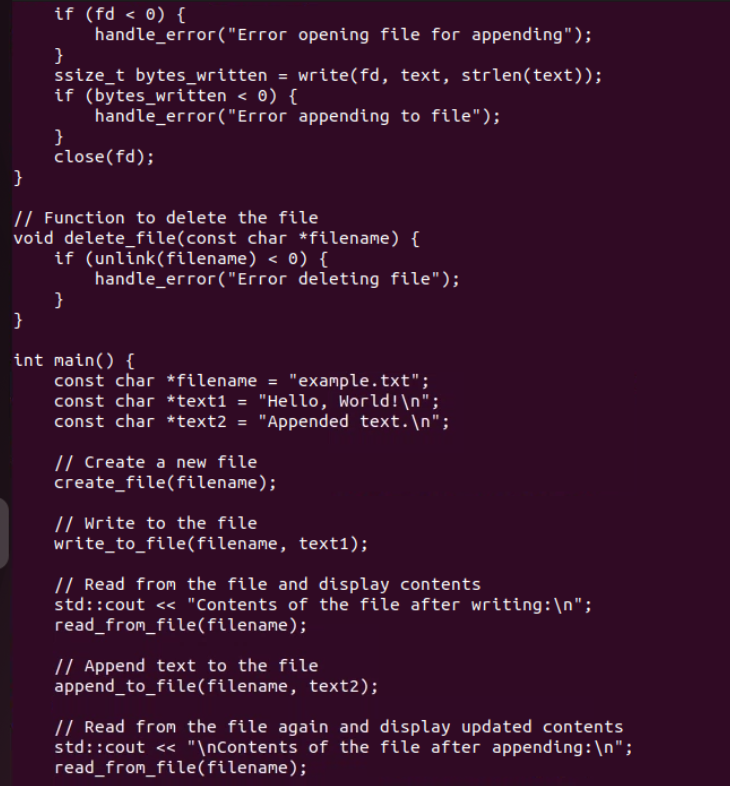
Use system calls like open, read, write, close, and unlink.

Handle errors appropriately by checking the return values of system calls and using perror to print error messages.

Ensure the program is modular with separate functions for each file operation (create, write, read, append, delete).

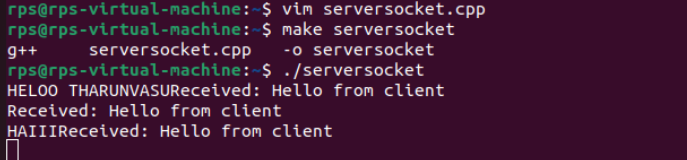




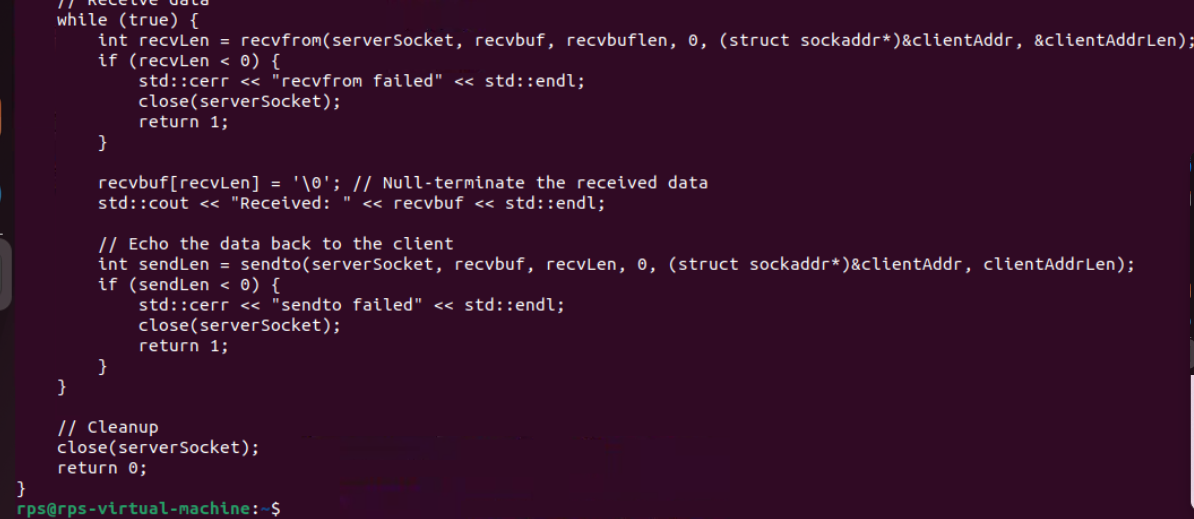




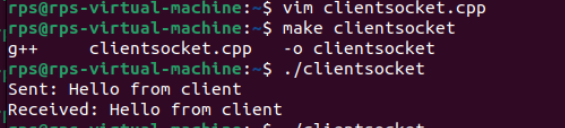
Server socket:



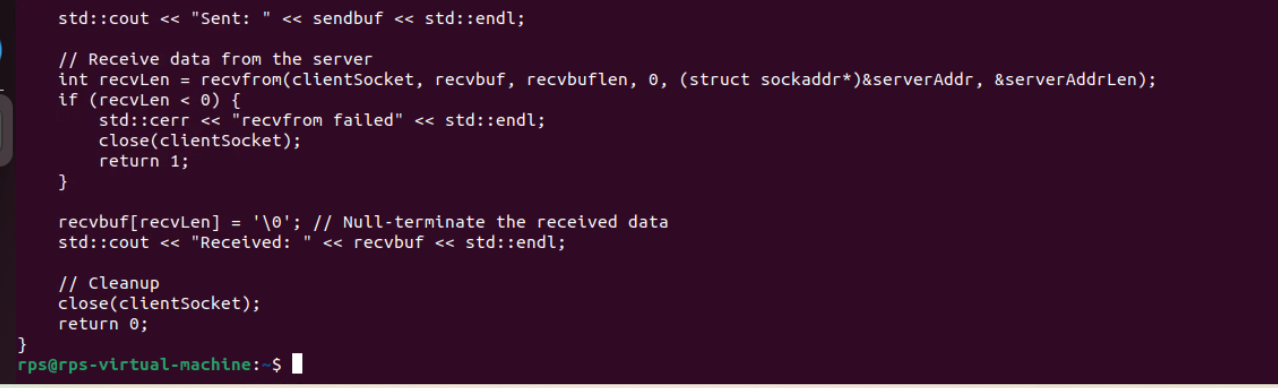




Client socket:

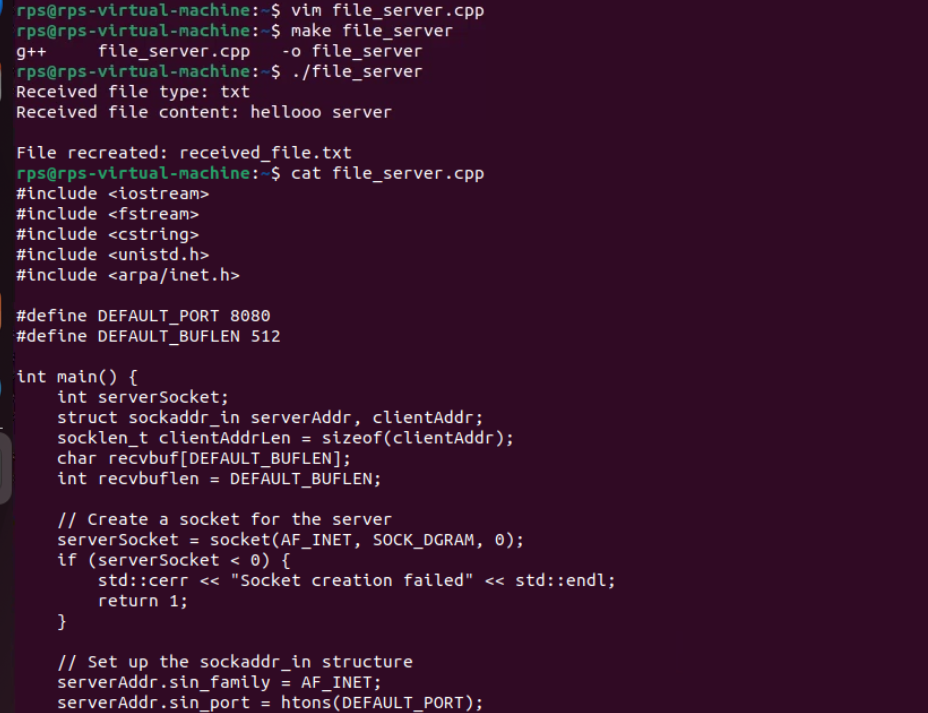


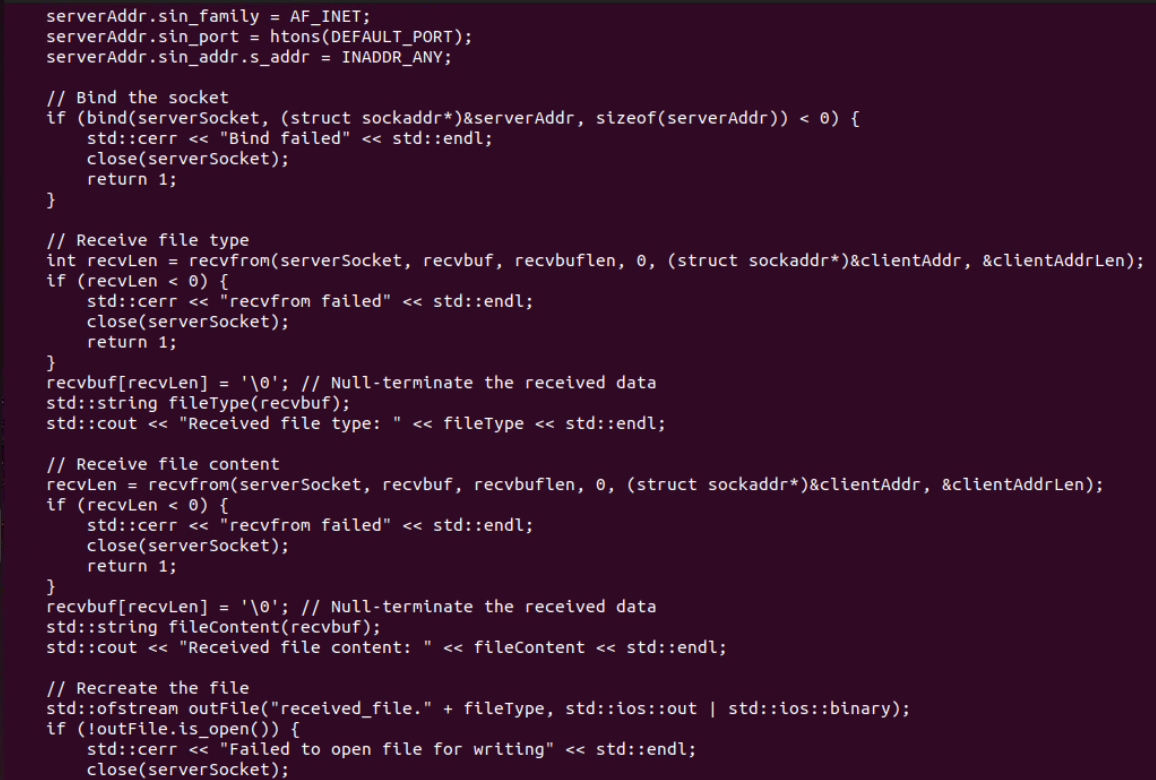


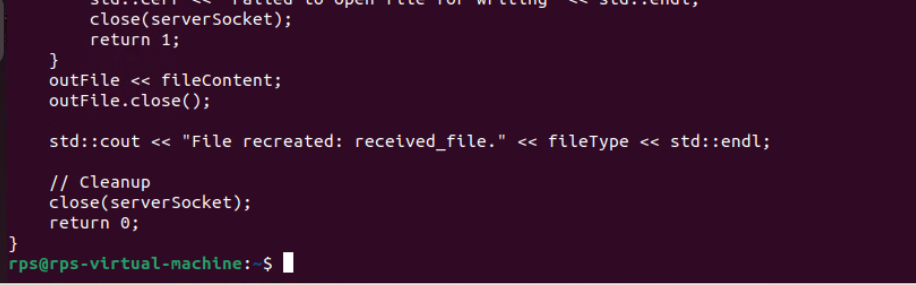


Sending file client to server in UDP:

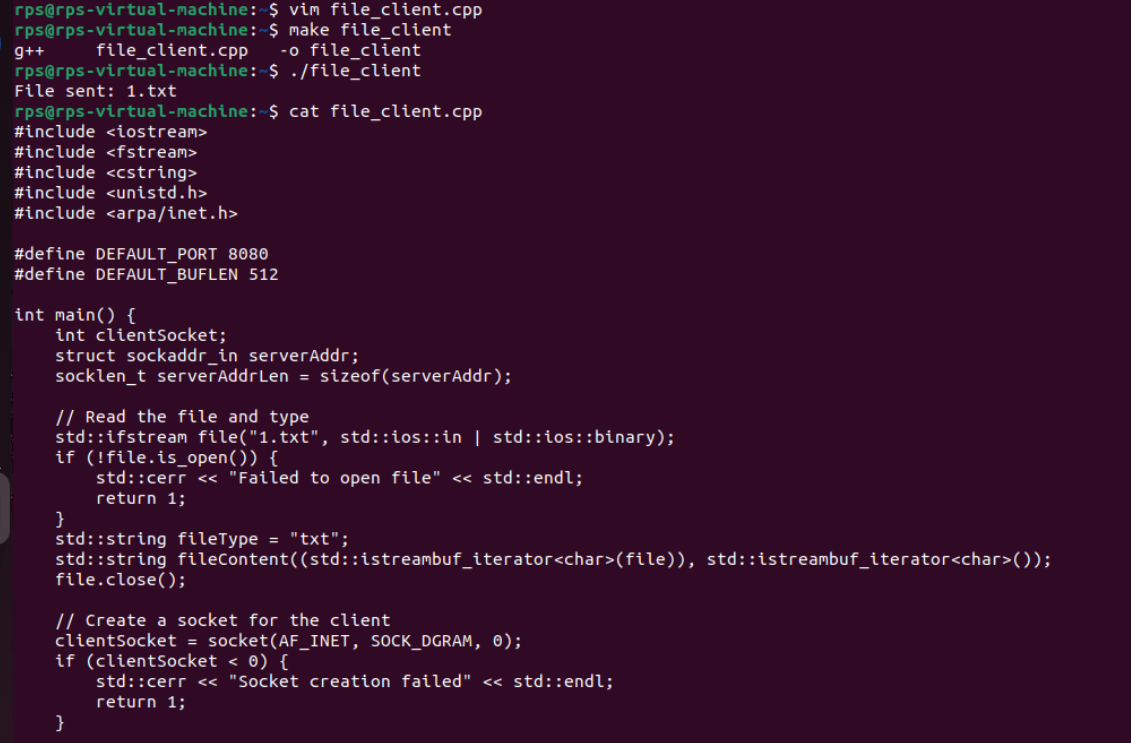
Server part:

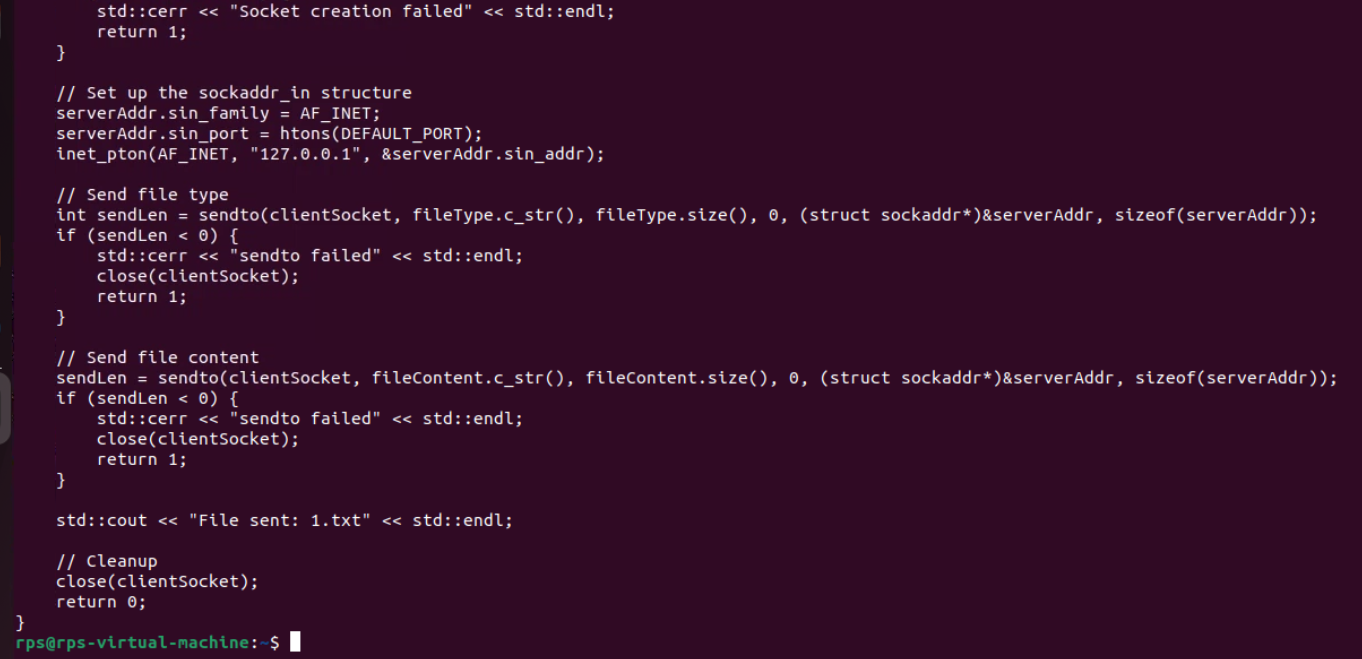






Client part:





UDP Server Implementation:

Create a UDP socket.

Bind the socket to a specified port.

Implement a loop to continuously listen for incoming messages.

Upon receiving a message:

Print the received message along with the client’s address and port.

Send an acknowledgment message ("Message received") back to the client.

Ensure proper error handling and resource cleanup.

2. UDP Client Implementation:

Create a UDP socket.

Allow the user to input the server’s IP address and port number.

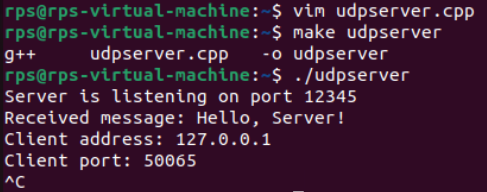
Send a predefined message (e.g., "Hello, Server!") to the server.

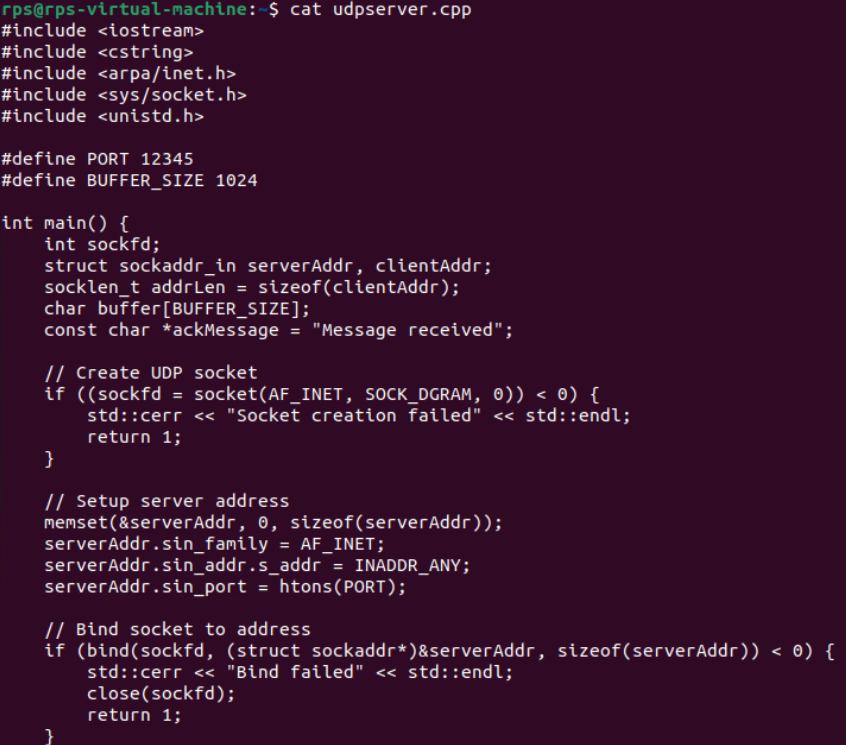
Wait for an acknowledgment from the server.

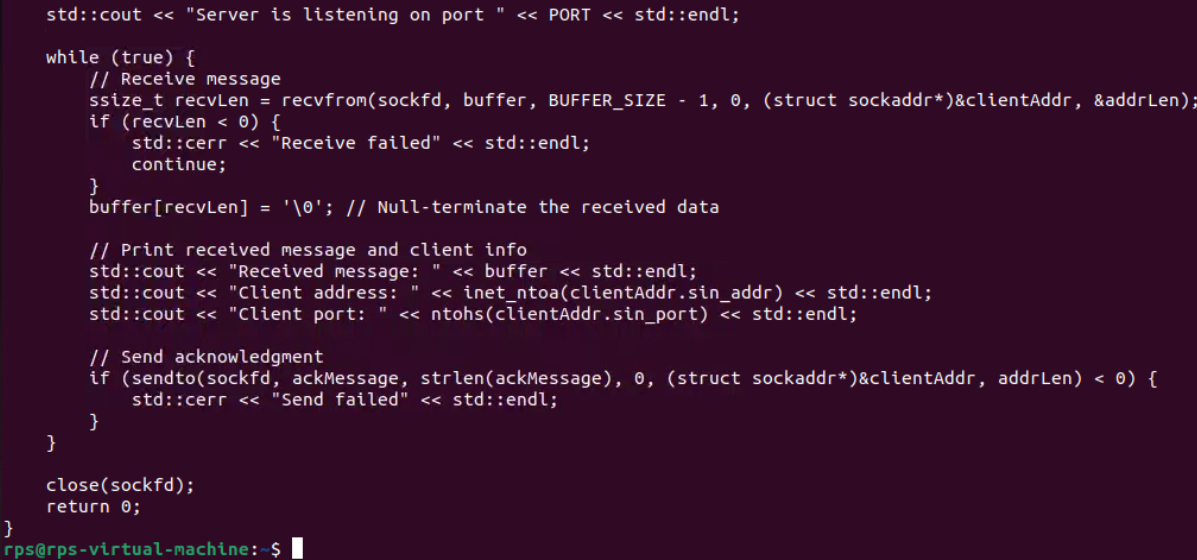
Print the acknowledgment message to the console.

Ensure proper error handling and resource cleanup.

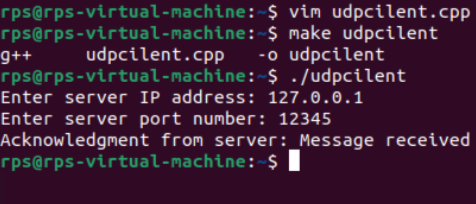
Server:

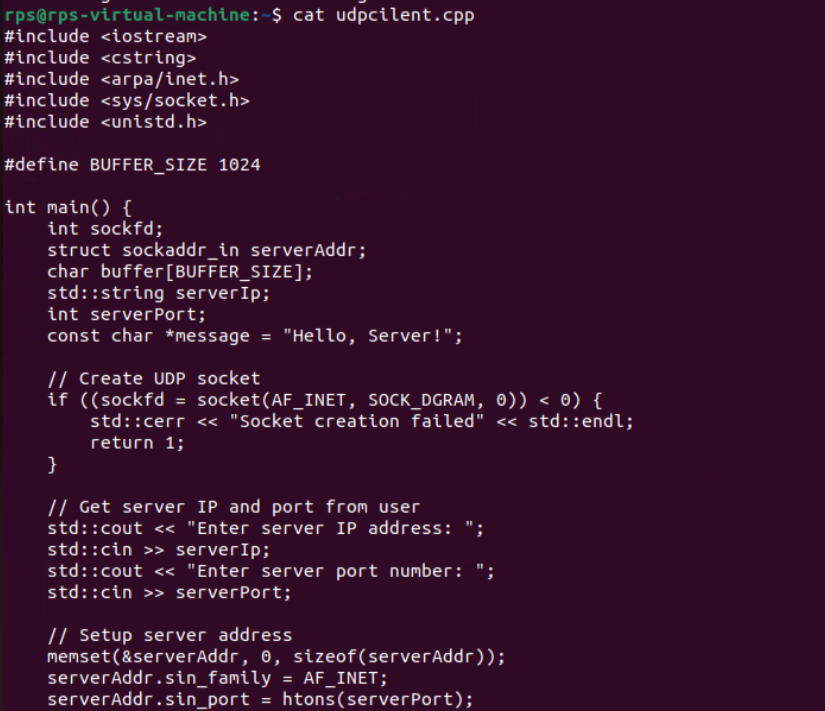
****

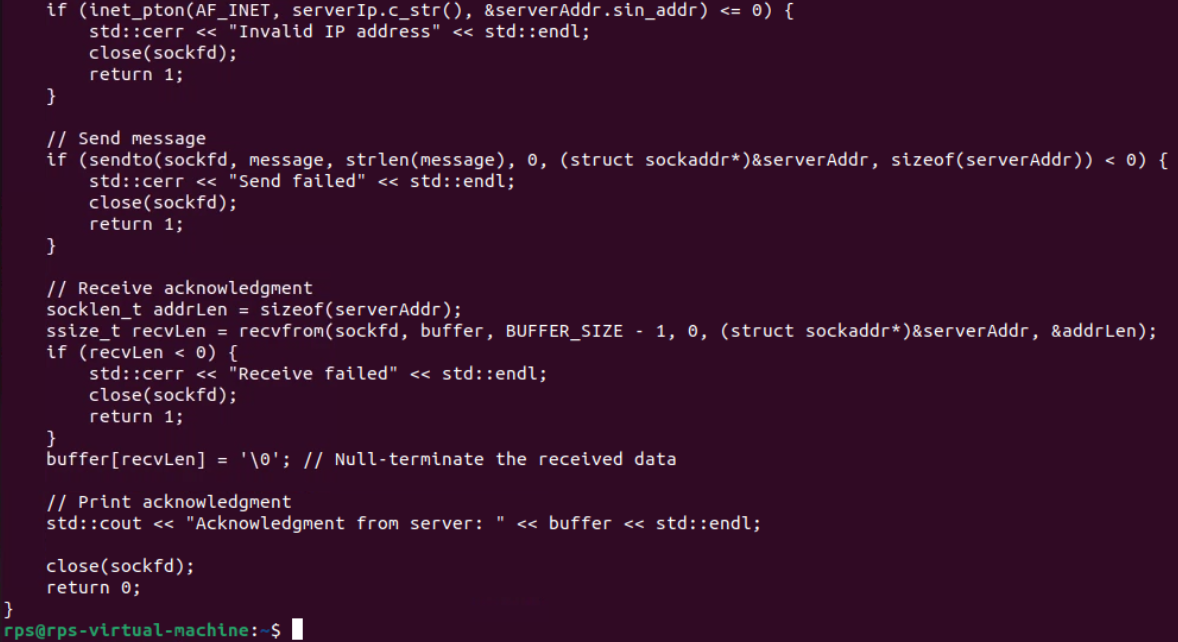
****

****

Client:

****

****

****