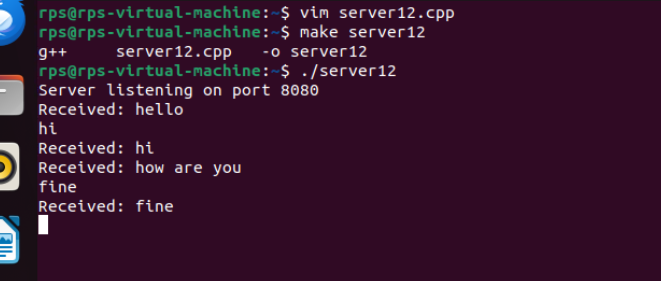
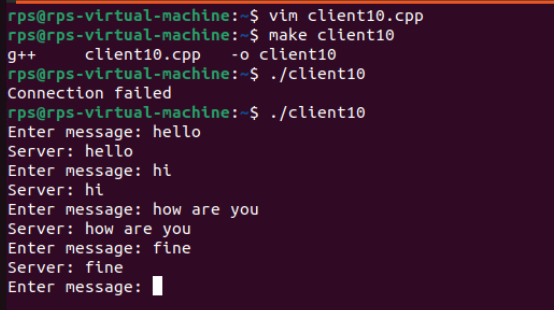
**Problem Statement: Socket Programming in C**

**Design and implement a reliable and efficient network communication system using socket programming in C to enable data exchange between two or more processes running on different machines over a network.**

****

**SERVER CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <pthread.h>

#define PORT 8080

#define BUFFER\_SIZE 1024

void\* handle\_client(void\* arg) {

int client\_socket = (int)arg;

char buffer[BUFFER\_SIZE];

int bytes\_received;

while ((bytes\_received = recv(client\_socket, buffer, BUFFER\_SIZE, 0)) > 0) {

buffer[bytes\_received] = '\0';

printf("Received: %s\n", buffer);

send(client\_socket, buffer, bytes\_received, 0); // Echo back to client

}

close(client\_socket);

free(arg);

return NULL;

}

int main() {

int server\_socket;

int \*client\_socket;

struct sockaddr\_in server\_addr, client\_addr;

socklen\_t client\_addr\_len = sizeof(client\_addr);

pthread\_t thread\_id;

// Create socket

server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if (server\_socket == -1) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

// Bind socket to address

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

server\_addr.sin\_port = htons(PORT);

if (bind(server\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("Bind failed");

close(server\_socket);

exit(EXIT\_FAILURE);

}

// Listen for incoming connections

if (listen(server\_socket, 5) == -1) {

perror("Listen failed");

close(server\_socket);

exit(EXIT\_FAILURE);

}

printf("Server listening on port %d\n", PORT);

// Accept and handle incoming connections

while (1) {

client\_socket = (int\*)malloc(sizeof(int));

if (client\_socket == NULL) {

perror("Failed to allocate memory for client socket");

continue;

}

client\_socket = accept(server\_socket, (struct sockaddr)&client\_addr, &client\_addr\_len);

if (\*client\_socket == -1) {

perror("Accept failed");

free(client\_socket);

continue;

}

if (pthread\_create(&thread\_id, NULL, handle\_client, client\_socket) != 0) {

perror("Thread creation failed");

close(\*client\_socket);

free(client\_socket);

continue;

}

pthread\_detach(thread\_id);

}

close(server\_socket);

return 0;

}

**CLIENT CODE:**

#include <iostream>

#include <cstring>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define BUFFER\_SIZE 1024

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

char buffer[BUFFER\_SIZE] = {0};

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

std::cerr << "Socket creation error" << std::endl;

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

std::cerr << "Invalid address / Address not supported" << std::endl;

return -1;

}

if (connect(sock, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) < 0) {

std::cerr << "Connection failed" << std::endl;

return -1;

}

while (true) {

std::cout << "Enter message: ";

std::string message;

std::getline(std::cin, message);

send(sock, message.c\_str(), message.size(), 0);

int valread = read(sock, buffer, BUFFER\_SIZE);

std::cout << "Server: " << buffer << std::endl;

memset(buffer, 0, BUFFER\_SIZE);

}

close(sock);

return 0;

}