Task

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.

**Server**

#include <iostream>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

#define PORT 8080

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char buffer[1024] = {0};

// Creating socket file descriptor

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Forcefully attaching socket to the port 8080

address.sin\_family = AF\_INET;

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

address.sin\_port = htons(PORT);

if (bind(server\_fd, (struct sockaddr \*)&address, sizeof(address)) < 0) {

perror("bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_fd, 3) < 0) {

perror("listen");

exit(EXIT\_FAILURE);

}

std::cout << "Server is listening on port " << PORT << std::endl;

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen)) < 0) {

perror("accept");

exit(EXIT\_FAILURE);

}

read(new\_socket, buffer, 1024);

std::cout << "Message delivered to server: " << buffer << std::endl;

close(new\_socket);

close(server\_fd);

return 0;

}

**Client:**

#include <iostream>

#include <csignal>

#include <cstring>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

int sock = 0;

void signalHandler(int signum) {

const char\* message = "Ctrl+C pressed!";

send(sock, message, strlen(message), 0);

std::cout << "Message delivered to server.\n";

exit(signum);

}

int main() {

struct sockaddr\_in serv\_addr;

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

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

return -1;

}

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

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

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

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

return -1;

}

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

std::cout << "Connection Failed" << std::endl;

return -1;

}

signal(SIGINT, signalHandler);

std::cout << "Press Ctrl+C to send message to server...\n";

while (true) {

// Keep the client running

}

return 0;

}