Name: Shivani Suresh

Register Number:2021503050

Semester:5

Batch:2

Course: CS6111- Computer Networks

Lab – Experiment 5-Implementation of echo server over TCP socket programming

Code:

Server:

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <cerrno>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

using namespace std;

const int PORT = 8080;

const int BUFFER\_SIZE = 1024;

int main() {

int serverSocket, clientSocket;

struct sockaddr\_in serverAddr, clientAddr;

socklen\_t clientAddrLen = sizeof(clientAddr);

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

if (serverSocket == -1) {

perror("Error creating socket");

exit(EXIT\_FAILURE);

}

serverAddr.sin\_family = AF\_INET;

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

serverAddr.sin\_port = htons(PORT);

if (bind(serverSocket, (struct sockaddr \*)&serverAddr, sizeof(serverAddr)) == -1) {

perror("Error binding");

close(serverSocket);

exit(EXIT\_FAILURE);

}

if (listen(serverSocket, 5) == -1) {

perror("Error listening");

close(serverSocket);

exit(EXIT\_FAILURE);

}

cout << "Server listening on port " << PORT << endl;

while (1) {

clientSocket = accept(serverSocket, (struct sockaddr \*)&clientAddr, &clientAddrLen);

if (clientSocket == -1) {

perror("Error accepting connection");

close(serverSocket);

exit(EXIT\_FAILURE);

}

char buffer[BUFFER\_SIZE];

ssize\_t bytesRead;

while ((bytesRead = recv(clientSocket, buffer, sizeof(buffer), 0)) > 0) {

buffer[bytesRead] = '\0';

cout << "Client: " << buffer << endl;

if (strcmp(buffer, "Exit") == 0) {

cout << "Client has terminated the conversation." << endl;

break;

}

send(clientSocket, buffer, bytesRead, 0);

}

if (bytesRead == -1) {

perror("Error receiving data");

close(clientSocket);

exit(EXIT\_FAILURE);

}

close(clientSocket);

}

close(serverSocket);

return 0;

}

Client:

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <cerrno>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

using namespace std;

const char\* SERVER\_IP = "127.0.0.1";

const int SERVER\_PORT = 8080;

const int BUFFER\_SIZE = 1024;

int main() {

int clientSocket;

struct sockaddr\_in serverAddr;

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

if (clientSocket == -1) {

perror("Error creating socket");

exit(EXIT\_FAILURE);

}

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(SERVER\_PORT);

if (inet\_pton(AF\_INET, SERVER\_IP, &(serverAddr.sin\_addr)) <= 0) {

perror("Error converting IP address");

close(clientSocket);

exit(EXIT\_FAILURE);

}

if (connect(clientSocket, (struct sockaddr \*)&serverAddr, sizeof(serverAddr)) == -1) {

perror("Error connecting to server");

close(clientSocket);

exit(EXIT\_FAILURE);

}

char message[BUFFER\_SIZE];

ssize\_t bytesRead;

while (1) {

cout << "Enter a message to send to the server (type 'Exit' to end the conversation): ";

cin.getline(message, BUFFER\_SIZE);

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

if (strcmp(message, "Exit") == 0) {

cout << "You have terminated the conversation." << endl;

break;

}

bytesRead = recv(clientSocket, message, sizeof(message), 0);

if (bytesRead == -1) {

perror("Error receiving data");

close(clientSocket);

exit(EXIT\_FAILURE);

}

message[bytesRead] = '\0';

cout << "Server: " << message << endl;

}

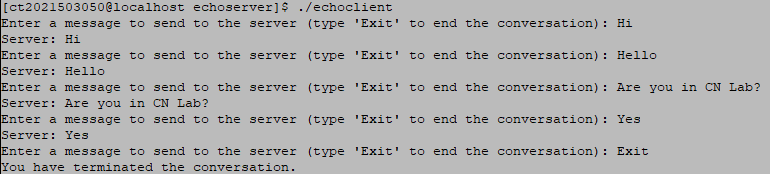
close(clientSocket);

return 0;

}

Output:

Client side:



Server side:

