Name: Shivani Suresh

Register Number:2021503050

Semester:5

Batch:2

Course: CS6111- Computer Networks

Lab – Experiment 3- Chat Application

1. Using TCP

Code:

Client:

#include <iostream>

#include <cstdlib>

#include <cstring>

#include <unistd.h>

#include <cstdio>

#include <arpa/inet.h>

#include <sys/socket.h>

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(2000);

serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

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

perror("Error connecting to server");

close(clientSocket);

exit(EXIT\_FAILURE);

}

std::cout << "Connected to server" << std::endl;

char buffer[1024];

while (true) {

std::string message;

std::cout << "Client : ";

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

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

memset(buffer, 0, sizeof(buffer));

int bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);

if (bytesReceived == -1) {

perror("Error receiving data");

close(clientSocket);

exit(EXIT\_FAILURE);

}

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

if (message == "BYE") {

break;

}

}

close(clientSocket);

return 0;

}

Server:

#include <iostream>

#include <cstdlib>

#include <cstring>

#include <cstdio>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

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\_port = htons(2000);

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

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

perror("Error binding socket");

close(serverSocket);

exit(EXIT\_FAILURE);

}

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

perror("Error listening");

close(serverSocket);

exit(EXIT\_FAILURE);

}

std::cout << "Server listening on port 2000..." << std::endl;

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

if (clientSocket == -1) {

perror("Error accepting client connection");

close(serverSocket);

exit(EXIT\_FAILURE);

}

char buffer[1024];

while (true) {

memset(buffer, 0, sizeof(buffer));

int bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);

if (bytesReceived == -1) {

perror("Error receiving data");

close(clientSocket);

close(serverSocket);

exit(EXIT\_FAILURE);

}

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

std::cout << "Server : ";

std::string response;

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

send(clientSocket, response.c\_str(), response.size(), 0);

if (response == "END") {

break;

}

}

close(clientSocket);

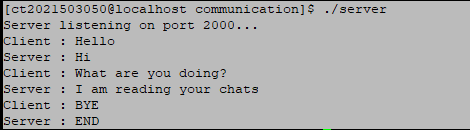
close(serverSocket);

return 0;

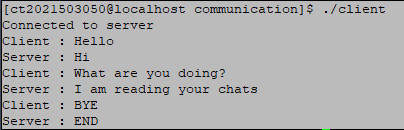
}

Output:

Server side:



Client side:



1. Using UDP

Code:

Client:

#include <iostream>

#include <cstdlib>

#include <cstdio.h>

#include <cstring>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

int main() {

int clientSocket;

struct sockaddr\_in serverAddr;

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

if (clientSocket == -1) {

perror("Error creating socket");

exit(EXIT\_FAILURE);

}

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(2000);

serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

std::string message;

while (true) {

std::cout << "You: ";

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

sendto(clientSocket, message.c\_str(), message.size(), 0, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));

}

close(clientSocket);

return 0;

}

Server:

#include <iostream>

#include <cstdlib>

#include <cstring>

#include <cstdio.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

int main() {

int serverSocket;

struct sockaddr\_in serverAddr, clientAddr;

socklen\_t clientAddrLen = sizeof(clientAddr);

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

if (serverSocket == -1) {

perror("Error creating socket");

exit(EXIT\_FAILURE);

}

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(2000);

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

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

perror("Error binding socket");

close(serverSocket);

exit(EXIT\_FAILURE);

}

char buffer[1024];

while (true) {

memset(buffer, 0, sizeof(buffer));

int bytesReceived = recvfrom(serverSocket, buffer, sizeof(buffer), 0, (struct sockaddr\*)&clientAddr, &clientAddrLen);

if (bytesReceived == -1) {

perror("Error receiving data");

close(serverSocket);

exit(EXIT\_FAILURE);

}

std::cout << "Client (" << inet\_ntoa(clientAddr.sin\_addr) << ":" << ntohs(clientAddr.sin\_port) << ") : " << buffer << std::endl;

}

close(serverSocket);

return 0;

}

Output:

