**NAME: DHANRAJ SUBHASH KORE**

**BATCH: B-3 ROLL NO: 60**

**CNT LAB ASSIGNMENT**

**Implementation of chat using UDP Client-Server communication**

**Server.cpp**

#undef UNICODE

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

// Need to link with Ws2\_32.lib

#pragma comment (lib, "Ws2\_32.lib")

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd, struct sockaddr\_in cliaddr)

{

char buff[MAX];

char hello = 'abcd';

char\* hellopter = &hello;

int n;

int len = sizeof(cliaddr);

// infinite loop for chat

for (;;) {

memset(&buff, 0, MAX);

// read the message from client and copy it in buffer

recvfrom(sockfd, buff, sizeof(buff), 0, (SA\*)&cliaddr, &len);

// print buffer which contains the client contents

printf("From client: %s\t To client : ", buff);

memset(&buff, 0, MAX);

n = 0;

// copy server message in the buffer

while ((buff[n++] = getchar()) != '\n');

// and send that buffer to client

//sendto(sockfd, buff, sizeof(buff), 0, (SA\*)&cliaddr,len);

sendto(sockfd, hellopter, sizeof(hellopter), 0, (SA\*)&cliaddr, len);

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

void calc(int sockfd)

{

char buff[MAX], temp[MAX];

char o;

int a, b, sol = 0;

// infinite loop for chat

for (;;) {

memset(&buff, 0, MAX);

recv(sockfd, buff, sizeof(buff), 0);

a = atoi(buff);

recv(sockfd, buff, sizeof(buff), 0);

b = atoi(buff);

recv(sockfd, buff, sizeof(buff), 0);

o = buff[0];

// print buffer which contains the client contents

//printf("From client: %s\t To client : ", buff);

if (o == '+')

{

sol = a + b;

}

else if (o == '-')

{

sol = a - b;

}

else if (o == '\*')

{

sol = a \* b;

}

else if (o == '/')

{

sol = a / b;

}

memset(&buff, 0, MAX);

// copy server message in the buffer

//while ((buff[n++] = getchar()) != '\n');

sprintf\_s(buff, MAX, "%d", sol);

// and send that buffer to client

send(sockfd, buff, sizeof(buff), 0);

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

// Function designed for chat between client and server.

void func(int sockfd)

{

char buff[MAX];

int n;

// infinite loop for chat

for (;;) {

memset(&buff, 0, MAX);

// read the message from client and copy it in buffer

recv(sockfd, buff, sizeof(buff), 0);

// print buffer which contains the client contents

printf(" Message from client  %s\t Message to client : ", buff);

memset(&buff, 0, MAX);

n = 0;

// copy server message in the buffer

while ((buff[n++] = getchar()) != '\n')

;

// and send that buffer to client

send(sockfd, buff, sizeof(buff), 0);

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

void acceptFile(int sockfd)

{

const int BUFFER\_SIZE = 256;

char buff[BUFFER\_SIZE];

size\_t datasize;

FILE\* fd;

fopen\_s(&fd, "tmp.txt", "wb");

while ((datasize = recv(sockfd, buff, sizeof(buff), 0)) > 0)

{

fwrite(&buff, 1, datasize, fd);

memset(&buff, 0, sizeof(buff));

}

fclose(fd);

}

// Driver function

int main()

{

int sockfd, connfd, len, iResult;

struct sockaddr\_in servaddr, cli, cliaddr;

WSADATA wsaData;

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

// socket create and verification

sockfd = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

memset(&servaddr, 0, sizeof(servaddr));

memset(&cliaddr, 0, sizeof(cliaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

len = sizeof(cli);

func(sockfd, cliaddr);

closesocket(sockfd);

}

**Client.cpp**

#undef UNICODE

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

#pragma comment (lib, "Ws2\_32.lib")

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd, struct sockaddr\_in servaddr)

{

int len = 12;

char buff[MAX];

int n;

for (;;) {

memset(buff, 0, sizeof(buff));

printf("PLEASE ENTER MESSAGE : ");

n = 0;

while ((buff[n++] = getchar()) != '\n');

sendto(sockfd, buff, sizeof(buff), 0, (SA\*)&servaddr, sizeof(servaddr));

memset(buff, 0, sizeof(buff));

recvfrom(sockfd, (char\*)buff, sizeof(buff), 0, (SA\*)&servaddr, &len);

printf("MESSAGE FROM SERVER : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0) {

printf("EXIT\n");

break;

}

}

}

void calc(int sockfd)

{

char buff[MAX];

int n;

for (;;) {

memset(buff, 0, sizeof(buff));

printf("Enter the first number ");

scanf\_s("%s", buff, (unsigned)\_countof(buff));

send(sockfd, buff, sizeof(buff), 0);

memset(buff, 0, sizeof(buff));

printf("Enter the second number ");

scanf\_s("%s", buff, (unsigned)\_countof(buff));

send(sockfd, buff, sizeof(buff), 0);

memset(buff, 0, sizeof(buff));

printf("Enter the operator ");

scanf\_s("%s", buff, (unsigned)\_countof(buff));

send(sockfd, buff, sizeof(buff), 0);

memset(buff, 0, sizeof(buff));

recv(sockfd, buff, sizeof(buff), 0);

printf("The answer is %s \n", buff);

if ((strncmp(buff, "exit", 4)) == 0) {

printf("Client Exit...\n");

break;

}

}

}

void sendFile(int sockfd)

{

const int BUFFER\_SIZE = 256;

char buff[BUFFER\_SIZE];

int n;

FILE\* fd;

fopen\_s(&fd, "tmp.txt", "rb");

//size\_t rret, wret;

int bytes\_read;

while (!feof(fd)) {

if ((bytes\_read = fread(&buff, 1, BUFFER\_SIZE, fd)) > 0)

send(sockfd, buff, bytes\_read, 0);

else

break;

}

fclose(fd);

}

int main()

{

int sockfd, connfd, iResult;

struct sockaddr\_in servaddr, cli;

WSADATA wsaData;

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

// socket create and varification

sockfd = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

memset(&servaddr, 0, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

InetPton(AF\_INET, (PCSTR)("127.0.0.1"), &servaddr.sin\_addr.s\_addr);

servaddr.sin\_port = htons(PORT);

func(sockfd, servaddr);

closesocket(sockfd);

}

