СЕРВЕР

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

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <winsock2.h>

#include <stdio.h>

#include <iostream>

#include <sstream>

#include <WS2tcpip.h>

#include <vector>

#include <string>

#include <stack>

using namespace std;

int main(void)

{

WORD sockVer;

WSADATA wsaData;

int retVal;

sockVer = MAKEWORD(2, 2);

WSAStartup(sockVer, &wsaData);

//Создаем сокет

SOCKET servSock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (servSock == INVALID\_SOCKET)

{

printf("Unable to create socket\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

//in\_addr ip\_to\_num;

//retVal = inet\_pton(AF\_INET, "10.241.107.91", &ip\_to\_num);

SOCKADDR\_IN sin;

sin.sin\_family = PF\_INET;

sin.sin\_port = htons(2001);

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

retVal = bind(servSock, (LPSOCKADDR)&sin, sizeof(sin));

if (retVal == SOCKET\_ERROR)

{

printf("Unable to bind\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

printf("Server started at %s, port %d\n", inet\_ntoa(sin.sin\_addr), htons(sin.sin\_port));

// cout << inet\_ntoa(sin.sin\_addr);

while (true)

{

//Пытаемся начать слушать сокет

retVal = listen(servSock, 10);

if (retVal == SOCKET\_ERROR)

{

printf("Unable to listen\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

//Ждем клиента

SOCKET clientSock;

SOCKADDR\_IN from;

int fromlen = sizeof(from);

clientSock = accept(servSock, (struct sockaddr\*)&from, &fromlen);

if (clientSock == INVALID\_SOCKET)

{

printf("Unable to accept\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

printf("New connection accepted from %s, port %d\n", inet\_ntoa(from.sin\_addr), htons(from.sin\_port));

char szReq[256];

retVal = recv(clientSock, szReq, 256, 0);

//Пытаемся получить данные от клиента

if (retVal == SOCKET\_ERROR)

{

printf("Unable to recv\n");

system("pause");

return SOCKET\_ERROR;

}

printf("Data received\n");

string s = (const char\*)szReq;

if ((s[0] == 's')and(s[1]=='t')and(s[2]=='o')and(s[3]=='p'))// Команда на выключение сервера

{

const char\* szResp = "Server shutdown";

retVal = send(clientSock, szResp, 256, 0);

closesocket(clientSock);

break;

}

else

{

stack <int> stack;

char szResp[256];

int i = 0;

int j = 0;

int c = 0;

while (i != retVal)

{

if (szReq[i] == '.')

{

szResp[j] = szReq[i];

j++;

i++;

if (c == 0) {

szResp[j] = 48;

j++;

}

else

{

while (c != 0)

{

stack.push(c % 10 + 48);

c = c / 10;

}

while (!stack.empty())

{

szResp[j] = stack.top();

stack.pop();

j++;

}

c = 0;

}

}

else

{

szResp[j] = szReq[i];

i++;

j++;

c++;

}

}

szResp[j] = '\0';

//Пытаемся отослать данные клиенту

printf("Sending response from server\n");

retVal = send(clientSock, szResp, 256, 0);

memset(&szReq[0], 0, sizeof(szReq));

memset(&szResp[0], 0, sizeof(szResp));

if (retVal == SOCKET\_ERROR)

{

printf("Unable to send\n");

system("pause");

return SOCKET\_ERROR;

}

closesocket(clientSock);

printf("Connection closed\n");

}

}

//Закрываем сокет

closesocket(servSock);

WSACleanup();

return 0;

}

КЛИЕНТ

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

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <winsock2.h>

#include <string>

#include <iostream>

using namespace std;

int main()

{

WORD ver = MAKEWORD(2, 2);

WSADATA wsaData;

int retVal = 0;

WSAStartup(ver, (LPWSADATA)&wsaData);

LPHOSTENT hostEnt;

hostEnt = gethostbyname("localhost");

if (!hostEnt)

{

printf("Unable to collect gethostbyname\n");

WSACleanup();

system("pause");

return 1;

}

//Создаем сокет

SOCKET clientSock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (clientSock == SOCKET\_ERROR)

{

printf("Unable to create socket\n");

WSACleanup();

system("pause");

return 1;

}

string ip;

int port;

cout << "ip>";

cin >> ip;

cin.ignore();

cout << "\nport>";

// while ((getchar()) != '\n');

cin >> port;

cin.ignore();

SOCKADDR\_IN serverInfo;

serverInfo.sin\_family = PF\_INET;

serverInfo.sin\_addr.S\_un.S\_addr = inet\_addr(ip.c\_str());

serverInfo.sin\_port = htons(port);

//Пытаемся присоединится к серверу по ip и port

retVal = connect(clientSock, (LPSOCKADDR)&serverInfo, sizeof(serverInfo));

if (retVal == SOCKET\_ERROR)

{

printf("Unable to connect\n");

WSACleanup();

system("pause");

return 1;

}

printf("Connection made sucessfully\n");

printf("Enter sentences or 'stop' to shutdown server\n");

char pBuf[256];

gets\_s(pBuf);

printf("Sending request from client\n");

//Отсылаем данные на сервер

retVal = send(clientSock, pBuf, strlen(pBuf), 0);

if (retVal == SOCKET\_ERROR)

{

printf("Unable to send\n");

WSACleanup();

system("pause");

return 1;

}

char szResponse[256];

//Пытаемся получить ответ от сервера

retVal = recv(clientSock, szResponse, 256, 0);

if (retVal == SOCKET\_ERROR)

{

printf("Unable to recv\n");

WSACleanup();

system("pause");

return 1;

}

char\* Resp;

Resp = szResponse;

printf("%s\n", Resp);

closesocket(clientSock);

WSACleanup();

system("pause");

return 0;

}