****

《网络程序设计》实验报告

**实验二 使用流式套接字实现网络通信**

【实验目的】

学会使用流式套接字编写基于C/S模型的两个通信程序

熟悉流式套接字的常用接口函数

掌握服务器端流式套接字编程的基本流程

掌握客户端流式套接字编程的基本流程

【实验过程】

1)使用基本的服务器编程模式，创建流式套接字，根据指定的端口号绑定服务，建立监听队列，并接受来自客户端的连接请求，收发数据；

使用基本的客户端编程模式，创建流式套接字，通过用户指定的服务器地址向指定服务器发起连接请求，与服务器之间实现收发数据，然后关闭连接。

本机测试：本机回送地址（127.0.0.1），测试本机进程间通信。

不同电脑测试：服务器（192.168.43.27）,客户端（192.168.43.40）。

2)通过改进服务器端，实现服务器循环为多个客户端提供服务；

要求服务器端能够显示对应客户端的ip地址；客户端实现用户输入信息方式与服务器对话。

本机测试：本机回送地址（127.0.0.1），测试本机进程间通信。

不同电脑测试：服务器（192.168.43.27）,客户端（192.168.43.40）。

3)采用 Windows环境下多线程开发方法改进服务器端，对每个客户连接请求独立创建通信线程,实现并发服务器。

本机测试：本机回送地址（127.0.0.1），测试本机进程间通信。

不同电脑测试：服务器（192.168.43.27）, 客户端1（192.168.43.19），客户端2（192.168.43.40）。

**实验结果截图**

1.服务器与客户端一对一通信

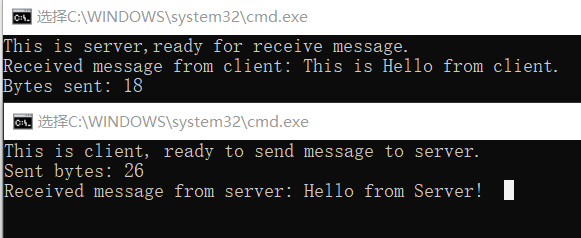
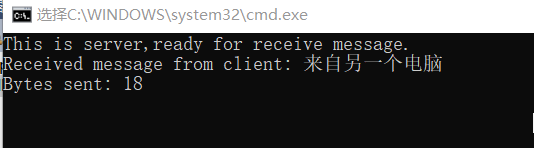
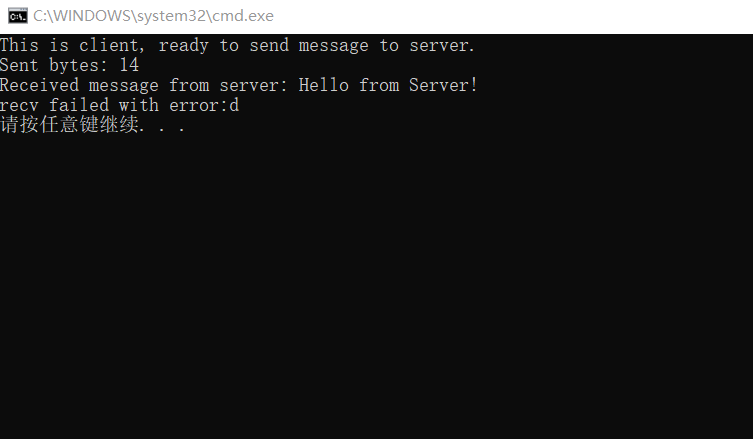


图1 本机服务器与客户端一对一



服务器（192.168.43.27）



客户端（192.168.43.40）

图 2 不同电脑服务器与客户端一对一

2.服务器循环提供服务

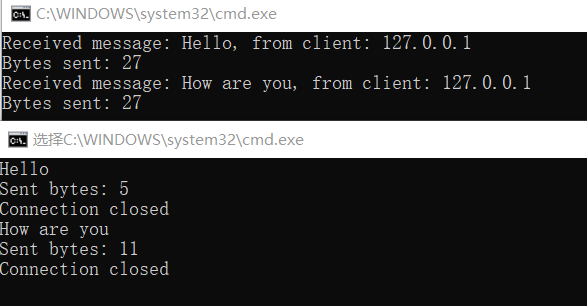
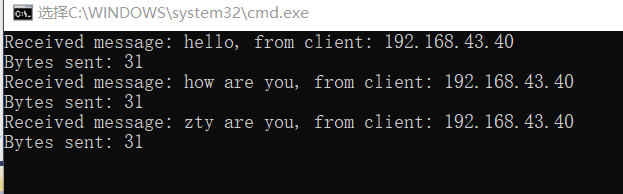
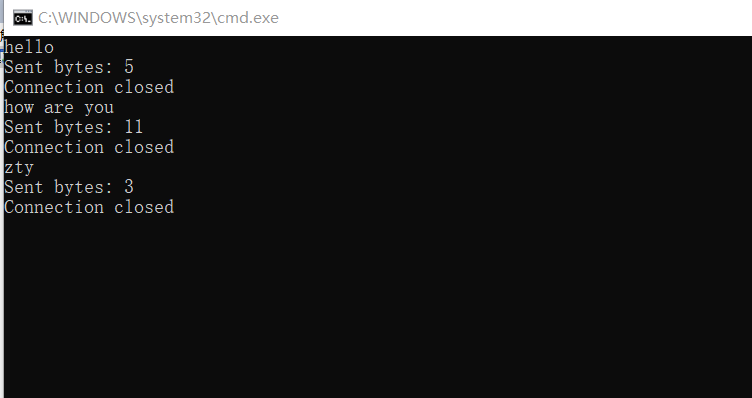


图3 本机



服务器（192.168.43.27）



客户端（192.168.43.40）

图4 不同电脑

3.多线程并发服务器

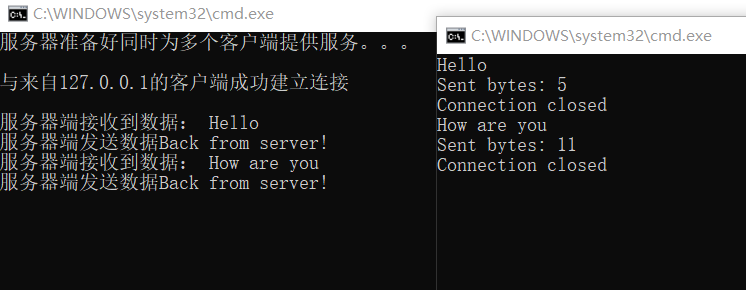
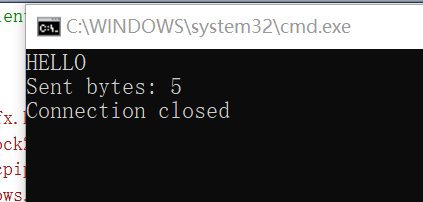


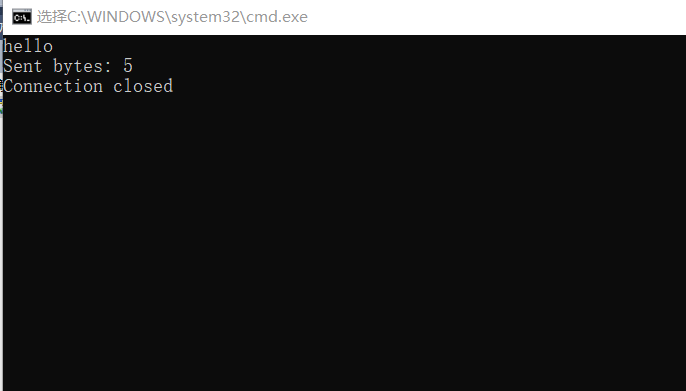
图5 本机



服务器（192.168.43.27）



客户端1（192.168.43.19）



客户端2（192.168.43.40）

图6 不同电脑

**调试中主要存在问题****总结：**

问题1：



图7 问题1截图

解决： 项目->SimpleHelloClient属性页->配置属性->常规->字符集，将“使用Unicode字符集”改为“使用多字节字符集”。

问题2：

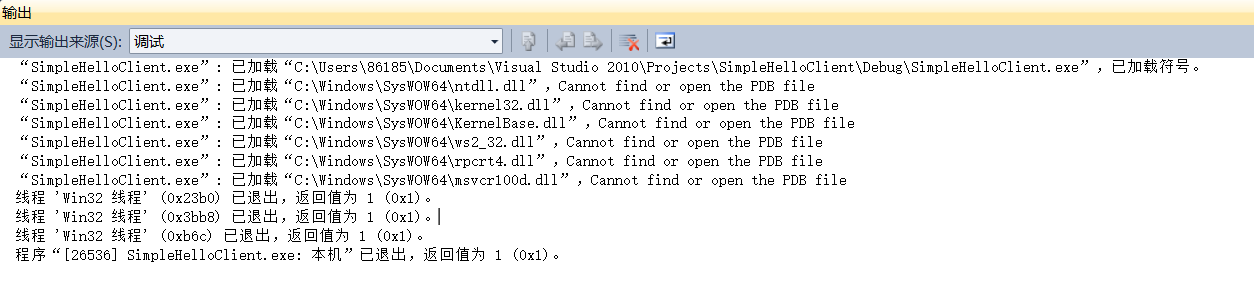


图8 问题2截图

解决：工具->选项->调试->符号,勾选“Microsoft符号服务器”。

问题3：

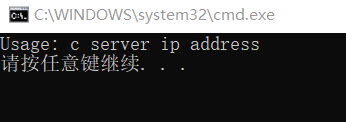


图9 问题3截图

解决：项目->SimpleHelloClient属性页->配置属性->调试,在命令参数中输入127.0.0.1（本机回送地址，用来测试本机进程间通信）。

问题4：

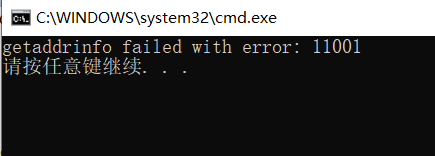


图10 问题4截图

解决：解决方法同问题1。

问题5：注意端口号保持一致！

**代码**

1. 服务器与客户端一对一

1.1服务器：

//SimpleHelloServer.cpp : 这是一个用流式套接字实现的一次性接收来自客户端信息并返回一个信息的服务器端程序。

#include "stdafx.h"

#include <WinSock2.h>

#include <Windows.h>

#include <WS2tcpip.h>

//连接到WinSock 2对应的lib文件：Ws2\_32.lib

#pragma comment (lib,"ws2\_32.lib")

//定义默认的端口号和缓冲区长度

#define SERVER\_PORT "27015"

#define BUFFER\_LEN 512

int \_tmain(int argc, \_TCHAR\* argv[])

{

WSADATA wsaData;

int iResult;

SOCKET ListenSocket=INVALID\_SOCKET;

SOCKET ClientSocket=INVALID\_SOCKET;

struct addrinfo \*result=NULL;

struct addrinfo hints;

int isendResult;

char \*sendbuf="Hello from Server!";

char recvbuf[BUFFER\_LEN];

memset(recvbuf,0,BUFFER\_LEN\*sizeof(char));

//初始化WinSock

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

if(iResult!=0)

{

printf("WSAStartup failed with error: &d\n",iResult);

return 1;

}

ZeroMemory(&hints,sizeof(hints));

//声明IPv4地址簇，流式套接字，TCP协议

hints.ai\_family=AF\_INET;

hints.ai\_socktype=SOCK\_STREAM;

hints.ai\_protocol=IPPROTO\_TCP;

hints.ai\_flags=AI\_PASSIVE;

//解析服务器地址和端口号

iResult=getaddrinfo(NULL,SERVER\_PORT,&hints,&result);

if(iResult!=0){

printf("getaddrinfo failed with error %d\n",iResult);

WSACleanup();

return 1;

}

//为面向连接的服务器创建套接字

ListenSocket=socket(result->ai\_family,result->ai\_socktype,result->ai\_protocol);

if(ListenSocket==INVALID\_SOCKET){

printf("socket failed with error %d\n",WSAGetLastError());

freeaddrinfo(result);

WSACleanup();

return 1;

}

//为套接字绑定地址和端口号 iResult=bind(ListenSocket,result->ai\_addr,(int)result->ai\_addrlen);

if(iResult==SOCKET\_ERROR){

printf("bind failed with error %d\n",WSAGetLastError());

freeaddrinfo(result);

closesocket(ListenSocket);

WSACleanup();

return 1;

}

freeaddrinfo(result);

//监听连接请求

iResult=listen(ListenSocket,SOMAXCONN);

if(iResult==SOCKET\_ERROR){

printf("listen failed with error %d\n",WSAGetLastError());

freeaddrinfo(result);

closesocket(ListenSocket);

WSACleanup();

return 1;

}

printf("This is server,ready for receive message.\n");

//接受客户端的连接请求，返回连接套接字ClientSocket

ClientSocket=accept(ListenSocket,NULL,NULL);

if(iResult==INVALID\_SOCKET){

printf("accept failed with error %d\n",WSAGetLastError());

freeaddrinfo(result);

closesocket(ListenSocket);

WSACleanup();

return 1;

}

//在必须要监听套接字的情况下释放该套接字

closesocket(ListenSocket);

//持续接收数据直到对方关闭连接

do{

iResult=recv(ClientSocket,recvbuf,30,0);

if(iResult>0){

//情况1：成功接收到数据

printf("Received message from client: %s\n",recvbuf);

//将缓冲区的内容回送给客户端

isendResult=send(ClientSocket,sendbuf,strlen(sendbuf),0);

if(isendResult==SOCKET\_ERROR){

printf("send failed with error %d\n",WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

return 1;

}

printf("Bytes sent: %d\n",isendResult);

}

else if(iResult==0){

//情况2：连接关闭

printf("Connection closing...\n");

}

else{

//情况3：接收发生错误

printf("recv failed with error:d\n",WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

return 1;

}

}while(iResult>0);

//关闭套接字，释放资源

closesocket(ListenSocket);

WSACleanup();

return 0;

}

1.2客户端：

// SimpleHelloClient.cpp : 这是一个只向服务器发送一次信息的简单客户端程序。

#include "stdafx.h"

#define WIN32\_LEAN\_AND\_MEAN

#include <Windows.h>

#include <WinSock2.h>

#include <WS2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <WinBase.h>

//连接到WinSock 2对应的lib文件：Ws2\_32.lib,Mswsock.lib,AdvApi32.lib

#pragma comment (lib,"ws2\_32.lib")

#pragma comment (lib,"Mswsock.lib")

#pragma comment (lib,"AdvApi32.lib")

//定义默认的端口号和缓冲区长度

#define SERVER\_PORT "27015"

#define BUFFER\_LEN 512

int \_tmain(int argc, \_TCHAR\*argv[])

{

WSADATA wsaData;

SOCKET ConnectSocket;

struct addrinfo \*result=NULL,\*ptr=NULL,hints;

char \*sendbuf="This is Hello from client.";

char recvbuf[BUFFER\_LEN];

int iResult;

memset(recvbuf,0,BUFFER\_LEN\*sizeof(char));

//验证参数的合法性

if(argc!=2){

printf("Usage: %s server ip address\n",argv[0]);

return 1;

}

//初始化套接字

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

if(iResult!=0){

printf("WSAStartup failed with error: d%\n",iResult);

return 1;

}

ZeroMemory(&hints,sizeof(hints));

hints.ai\_family=AF\_INET;

hints.ai\_socktype=SOCK\_STREAM;

hints.ai\_protocol=IPPROTO\_TCP;

//解析服务器地址和端口号

iResult=getaddrinfo(argv[1],SERVER\_PORT,&hints,&result); //将输入参数argv[1]中指定的服务器信息写入result

if(iResult!=0){

printf("getaddrinfo failed with error: %d\n",iResult);

WSACleanup();

return 1;

}

//创建套接字

ConnectSocket=socket(result->ai\_family,result->ai\_socktype,result->ai\_protocol); //使用result指定的信息创建套接字

if(ConnectSocket==INVALID\_SOCKET){

printf("socket failed with error: %ld\n",WSAGetLastError());

WSACleanup();

return 1;

}

printf("This is client, ready to send message to server.\n");

//向服务器请求连接

iResult=connect(ConnectSocket,result->ai\_addr,result->ai\_addrlen); //使用套接字ConnectSocket向result中指定的服务器请求连接

if(iResult==SOCKET\_ERROR){

printf("connect failed with error: %ld\n",iResult);

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

freeaddrinfo(result); //释放动态分配的地址信息结构体result

//发送缓冲区中的测试数据

iResult=send(ConnectSocket,sendbuf,strlen(sendbuf),0);

if(iResult==SOCKET\_ERROR){

printf("send failed with error: %ld\n",iResult);

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

printf("Sent bytes: %d\n",iResult);

//持续接收数据，直到服务器关闭连接

do{

iResult=recv(ConnectSocket,recvbuf,sizeof(recvbuf),0);

if(iResult>0){

printf("Received message from server: %s\n",recvbuf);

}

else if(iResult==0){

printf("Connection closed\n");

}

else{

printf("recv failed with error:d\n",WSAGetLastError());

}

}while(iResult>0);

//关闭套接字

closesocket(ConnectSocket);

//释放资源

WSACleanup();

return 0;

}

1. 服务器循环提供服务

2.1服务器

// DuokehuServer.cpp：为多客户提供服务的服务器端程序。

#include "stdafx.h"

#include <WinSock2.h>

#include <Windows.h>

#include <WS2tcpip.h>

#pragma comment (lib,"ws2\_32.lib")

#define SERVER\_PORT "8888"

#define BUFFER\_LEN 512

int \_tmain(int argc, \_TCHAR\* argv[])

{

WSADATA wsaData;

SOCKET ListenSocket=INVALID\_SOCKET;

SOCKET ClientSocket=INVALID\_SOCKET;

struct addrinfo hints,\*result=NULL;

struct sockaddr\_in clientaddr;

char sendbuf[BUFFER\_LEN];

char recvbuf[BUFFER\_LEN];

int iResult,isendResult;

memset(recvbuf,0,BUFFER\_LEN\*sizeof(char));

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

if(iResult!=0)

{

printf("WSAStartup failed with error: &d\n",iResult);

return 1;

}

ZeroMemory(&hints,sizeof(hints));

hints.ai\_family=AF\_INET;

hints.ai\_socktype=SOCK\_STREAM;

hints.ai\_protocol=IPPROTO\_TCP;

hints.ai\_flags=AI\_PASSIVE;

iResult=getaddrinfo(NULL,SERVER\_PORT,&hints,&result);

if(iResult!=0){

printf("getaddrinfo failed with error %d\n",iResult);

WSACleanup();

return 1;

}

ListenSocket=socket(result->ai\_family,result->ai\_socktype,result->ai\_protocol);

if(ListenSocket==INVALID\_SOCKET){

printf("socket failed with error %d\n",WSAGetLastError());

freeaddrinfo(result);

WSACleanup();

return 1;

}

iResult=bind(ListenSocket,result->ai\_addr,(int)result->ai\_addrlen);

if(iResult==SOCKET\_ERROR){

printf("bind failed with error %d\n",WSAGetLastError());

freeaddrinfo(result);

closesocket(ListenSocket);

WSACleanup();

return 1;

}

freeaddrinfo(result);

iResult=listen(ListenSocket,SOMAXCONN);

if(iResult==SOCKET\_ERROR){

printf("listen failed with error %d\n",WSAGetLastError());

freeaddrinfo(result);

closesocket(ListenSocket);

WSACleanup();

return 1;

}

for(;;){

int addrlenth=sizeof(clientaddr);

ClientSocket=accept(ListenSocket,(sockaddr\*)&clientaddr,&addrlenth);

if(iResult==INVALID\_SOCKET){

printf("accept failed with error %d\n",WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return 1;

}

char \*peeraddr= inet\_ntoa(clientaddr.sin\_addr);

do{

iResult=recv(ClientSocket,recvbuf,BUFFER\_LEN,0);

if(iResult>0){

printf("Received message: %s, from client: %s\n",recvbuf,peeraddr);

//显示收到的信息及对方的IP地址

strcpy(sendbuf,"Hello from server!"); //给发送缓冲区sendbuf赋初值

strcat(sendbuf,peeraddr);

//将服务器问候字符串后跟该客户端的IP地址，作为新的sendbuf内容。 isendResult=send(ClientSocket,sendbuf,strlen(sendbuf),0);

if(isendResult==SOCKET\_ERROR){

printf("send failed with error %d\n",WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

break;

}

printf("Bytes sent: %d\n",isendResult);

}

else if(iResult==0){

printf("Connection closing...\n");

iResult=shutdown(ClientSocket,SD\_SEND);

if(iResult==SOCKET\_ERROR){

printf("shutdown failed with error %d\n",WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

break;

}

closesocket(ClientSocket);

WSACleanup();

break;

}

else{

printf("recv failed with error:%d\n",WSAGetLastError());

iResult=shutdown(ClientSocket,SD\_SEND);

if(iResult==SOCKET\_ERROR){

printf("shutdown failed with error %d\n",WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

break;

}

}

}while(iResult>0);

}

closesocket(ListenSocket);

WSACleanup();

return 0;

}

2.2客户端

// ShuruxinxClient.cpp : 客户端程序，用户可以从键盘输入信息并发送给服务器。

//

#include "stdafx.h"

#include <WinSock2.h>

#include <WS2tcpip.h>

#include <Windows.h>

#include <WinBase.h>

#pragma comment (lib,"ws2\_32.lib")

#define SERVER\_PORT "27015"

#define BUFFER\_LEN 512

int \_tmain(int argc, \_TCHAR\* argv[])

{

struct addrinfo \*result=NULL,\*ptr=NULL,hints;

WSADATA wsaData;

SOCKET ConnectSocket;

char sendbuf[BUFFER\_LEN];

char recvbuf[BUFFER\_LEN];

int iResult;

if(argc!=2){

printf("Usage: %s server ip address\n",argv[0]);

return 1;

}

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

if(iResult!=0){

printf("WSAStartup failed with error: d%\n",iResult);

return 1;

}

ZeroMemory(&hints,sizeof(hints));

hints.ai\_family=AF\_INET;

hints.ai\_socktype=SOCK\_STREAM;

hints.ai\_protocol=IPPROTO\_TCP;

iResult=getaddrinfo(argv[1],SERVER\_PORT,&hints,&result); //将输入参数argv[1]中指定的服务器信息写入result

if(iResult!=0){

printf("getaddrinfo failed with error: %d\n",iResult);

WSACleanup();

return 1;

}

ConnectSocket=socket(result->ai\_family,result->ai\_socktype,result->ai\_protocol);

//使用result指定的信息创建套接字

if(ConnectSocket==INVALID\_SOCKET){

printf("socket failed with error: %ld\n",WSAGetLastError());

WSACleanup();

return 1;

}

iResult=connect(ConnectSocket,result->ai\_addr,result->ai\_addrlen); //使用套接字ConnectSocket向result中指定的服务器请求连接

if(iResult==SOCKET\_ERROR){

printf("connect failed with error: %ld\n",iResult);

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

freeaddrinfo(result); //释放动态分配的地址信息结构体result

while(gets(sendbuf)!=NULL){ //从键盘获取输入字符串

if(\*sendbuf=='Q'){

closesocket(ConnectSocket);

return 0;

}

iResult=send(ConnectSocket,sendbuf,strlen(sendbuf),0);

if(iResult==SOCKET\_ERROR){

printf("send failed with error: %ld\n",iResult);

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

printf("Sent bytes: %d\n",iResult);

do{

memset(recvbuf,0,BUFFER\_LEN\*sizeof(char));

iResult=recv(ConnectSocket,recvbuf,strlen(recvbuf),0);

if(iResult>0){

printf("Received message from server: %s\n",recvbuf);

}

else if(iResult==0){

printf("Connection closed\n");

}

else{

printf("recv failed with error:d\n",WSAGetLastError());

}

}while(iResult>0);

}

closesocket(ConnectSocket);

WSACleanup();

return 0;

}

1. 多线程并发服务器

3.1服务器

// DuoxianchengServer.cpp : 采用多线程机制实现并发为多个客户端服务。

//

#include "stdafx.h"

#include "DuoxianchengServer.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

#define SERVER\_PORT "27015"

#define BUFFER\_LEN 512

#include <WinSock2.h>

#include <Windows.h>

#include <WS2tcpip.h>

#pragma comment (lib,"ws2\_32.lib")

// 唯一的应用程序对象

CWinApp theApp;

using namespace std;

UINT duoxiancheng\_server\_fun( LPVOID pParam );

int \_tmain(int argc, TCHAR\* argv[], TCHAR\* envp[])

{

int nRetCode = 0;

int iResult = 0;

SOCKET ListenSocket, ConnectSocket;

CWinThread \*pThread= NULL;

WSADATA wsaData;

struct addrinfo hints,\*result=NULL;

struct sockaddr\_in clientaddr;

HMODULE hModule = ::GetModuleHandle(NULL);

if (hModule != NULL)

{

// 初始化 MFC 并在失败时显示错误

if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))

{

// TODO: 更改错误代码以符合您的需要

\_tprintf(\_T("错误: MFC 初始化失败\n"));

nRetCode = 1;

}

else

{

// // TODO: 在此处为应用程序的行为编写代码。

// //Windows Sockets Dll初始化

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

if(iResult!=0){

printf("WSAStartup failed with error %d\n",iResult);

return 1;

}

ZeroMemory(&hints,sizeof(hints));

hints.ai\_family=AF\_INET;

hints.ai\_socktype=SOCK\_STREAM;

hints.ai\_protocol=IPPROTO\_TCP;

hints.ai\_flags=AI\_PASSIVE;

iResult=getaddrinfo(NULL,SERVER\_PORT,&hints,&result);

if(iResult!=0){

printf("getaddrinfo failed with error %d\n",iResult);

closesocket(ListenSocket);

WSACleanup();

return 1;

}

ListenSocket= socket(result->ai\_family,result->ai\_socktype,result->ai\_protocol);

if(ListenSocket==INVALID\_SOCKET){

printf("socket failed with error %ld\n",GetLastError());

WSACleanup();

return 1;

}

iResult=bind(ListenSocket,result->ai\_addr,(int)result->ai\_addrlen);

if(iResult==SOCKET\_ERROR){

printf("bind failed with error %d\n",WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

freeaddrinfo(result);

return 1;

}

freeaddrinfo(result);

iResult=listen(ListenSocket,SOMAXCONN);

if(iResult==SOCKET\_ERROR){

printf("listen failed with error %d\n",WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return 1;

}

printf("服务器准备好同时为多个客户端提供服务。。。\n");

//开始

for(;;){

int addrlenth=sizeof(clientaddr);

ConnectSocket=accept(ListenSocket,(sockaddr\*)&clientaddr,&addrlenth);

if(ConnectSocket==INVALID\_SOCKET){

printf("accept failed with error %d\n",WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return -1;

}

char \*peeraddr= inet\_ntoa(clientaddr.sin\_addr);

//建立连接成功

printf("\r\n与来自%s的客户端成功建立连接\n\n",peeraddr);

//启动回射线程

pThread = AfxBeginThread( duoxiancheng\_server\_fun, &ConnectSocket );

}

printf("accept failed with error %d\n",WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return -1;

}

}

else

{

// TODO: 更改错误代码以符合您的需要

\_tprintf(\_T("错误: GetModuleHandle 失败\n"));

nRetCode = 1;

}

return nRetCode;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名：duoxiancheng\_server\_fun

输入参数：SOCKET s:服务器的连接套接字

输出参数：0：成功，-1：失败

功能：与客户端的具体通信功能实现函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

UINT duoxiancheng\_server\_fun( LPVOID pParam )

{

int iSendResult=0;

char sendbuf[BUFFER\_LEN];

char recvbuf[BUFFER\_LEN];

int iResult=0;

int err=0;

//将输入参数转换为连接套接字

SOCKET s =\*( (SOCKET \*)pParam);

do {

memset( recvbuf, 0, BUFFER\_LEN );

//接收数据

iResult = recv( s, recvbuf, BUFFER\_LEN, 0);

if (iResult > 0){

printf("服务器端接收到数据： %s\n", recvbuf);

//向客户端发送数据

memset( sendbuf, 0, BUFFER\_LEN );

strcat(sendbuf,"Back from server!");

iResult = send( s,sendbuf,BUFFER\_LEN, 0 );

if(iResult == SOCKET\_ERROR)

{

printf("send 函数调用错误，错误号:: %ld\n", WSAGetLastError());

err = closesocket(s);

if (err == SOCKET\_ERROR){

printf("closesocket函数调用错误，错误号：%d\n", WSAGetLastError());

}

iResult = -1;

}

else

printf("服务器端发送数据Y%s\n", sendbuf);

}

else{

if (iResult == 0)

printf("对方连接关闭，退出\n");

else{

printf("recv 函数调用错误，错误号: %d\n", WSAGetLastError());

iResult = -1;

}

err = closesocket(s);

if (err == SOCKET\_ERROR){

printf("closesocket 函数调用错误，错误号：%d\n", WSAGetLastError());

}

break;

}

} while (iResult > 0);

return iResult;

}

3.2客户端

代码同2.2客户端

【实验总结】

1.“使用Unicode字符集”改为“使用多字节字符集”。

2.勾选“Microsoft符号服务器”。

3. 端口号保持一致

4.注意本机进程通信与不同主机通信传参不同

5.仅实现服务器循环为一个客户端提供服务，并没有实现循环为多个客户端提供服务。

6.编写多线程并发服务器代码，由于没能解决调试时的错误，所以没有使用含csocketframe类的代码，而是用了老师提供的另一个代码。