

**北 京 科 技 大 学**

**计算机网络课程设计报告**

学　　院： \_\_计算机与通信工程学院\_\_

班　　级： \_\_\_\_\_\_\_\_信安182\_\_\_\_\_\_\_\_\_

姓　　名： \_\_\_\_\_\_\_\_\_梁逸飞\_\_\_\_\_\_\_\_\_\_

学　　号： \_\_\_\_\_\_\_\_41824141\_\_\_\_\_\_\_\_\_

成　　绩： \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

指导教师签字： \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**年 月**

**北京科技大学实验报告**

学院：计算机与通信工程学院 专业：信息安全 班级： 信安182

姓名：梁逸飞 学号：41824141 实验日期：2021年 4月

**实验名称：**计算机网络课程设计 实验任务一——基于SOCKET开发简单的C/S文本传输程序

**实验目的：**

将书本上抽象的概念与具体实现技术结合，通过网络软件编程的实践，深入理解理论课上学习到的计算机网络基本原理和重要协议，通过自己动手编程封装与发送数据分组，加深对网络协议的理解，掌握协议数据单元的结构和协议工作原理及其对协议栈的贡献。

自己动手编写一个基于SOCKET的简单C/S本文传输程序，理解TCP/IP协议的工作过程，掌握服务器与客户端搭建与连接的流程。

**实验仪器：**

实验硬件设备：华硕飞行堡垒， Intel(R) Dual Band Wireless-AC 8265

实验软件要求：win10， VS2010， C语言

**实验原理：**

实验原理：

基于SOCKET开发简单的C/S文本传输程序，socket是通信系统双方建立的一个标准约定，即“接头暗号”，其内部封装有各种网络通信所需要的功能函数。系统中的应用层功能的实现是需要通过传输层进行服务中转的，分布式的系统决定了网络系统中同一时间内应用程序可能是多个的，但是传输层只能有一个，此时传输层就需要同时为多个应用程序进程提供并发服务。socket接口能够为不同的应用程序进程和网络连接创建各自的身份标识，规范应用程序与TCP/IP协议之间的数据交互。应用程序进程间的网络通信和连接的身份主要通过目标IP地址、传输层协议和端口号三个参数标识。每个socket都绑定有IP地址、端口、端口号这3个参数。传输层借助socket服务，区分不同的应用程序进程或网络连接，并分别对其提供对应的数据传输服务。

**实验内容与步骤：**

**题目：基于SOCKET开发简单的C/S文本传输程序**

（1）需求分析

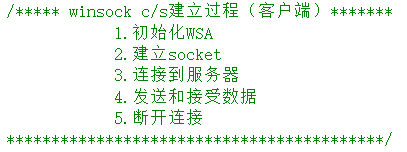
选择使用C语言开发，基于TCP/IP协议SOCKET套接字，开发一个简单的C/S文本传输程序，客户端能够发送由标准输入得到的文本，服务器端能够接收并将其显示在标准输出上。

（2）设计方案

将基于SOCKET编写服务器和客户端两个程序，服务器开发流程如下图所示。



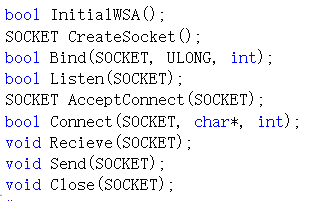
客户端开发流程如下图所示。



其中WSA是Windows Sockets Asynchronous，用于创建套接字、连接关闭等等。

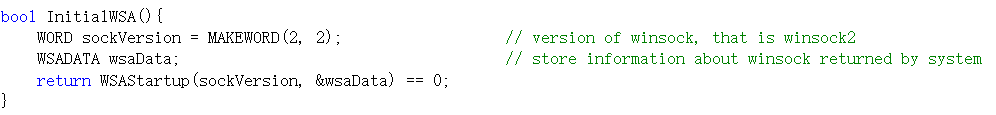
服务器和客户端建立过程中有很多相同的地方，所以在开发过程当中我们将SOCKET建立过程单独写入一个头文件，然后在服务器与客户端中的主函数进行调用。

头文件中包含初始化WSA、建立socket、绑定、监听、接收连接、连接、发送和接受数据、断开连接函数，囊括了客户端和服务端建立连接过程。以下是对于头文件中各函数的说明，在主函数中只需进行调用即可。

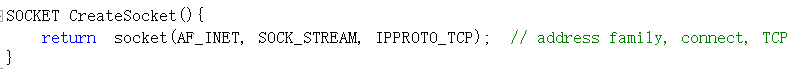


①初始化WSA

通过调用winsock库来建立起socket连接。



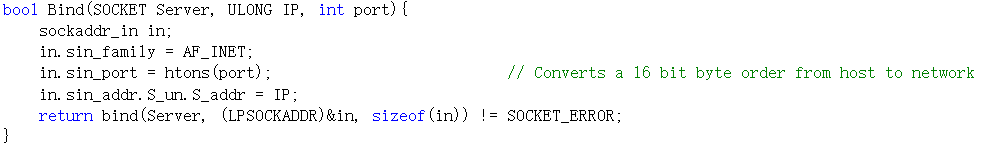
②创建socket



返回一个SOCKET对象，在主函数中创建一个SOCKET变量来接收返回值。

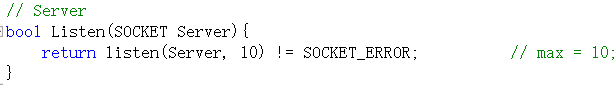


③绑定端口和IP



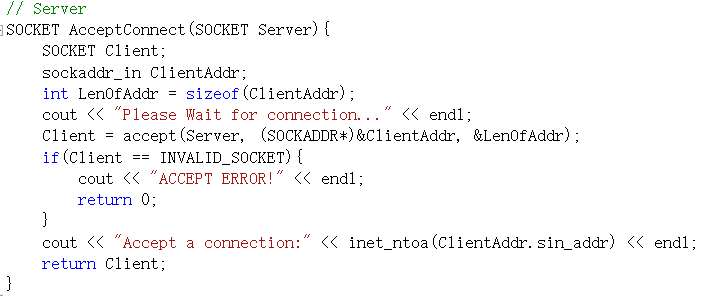
在主函数中传入SOCKET、IP和port，调用socket库将服务器SOCKET与设置好的IP与端口port进行绑定，绑定成功的话会返回true。

④监听



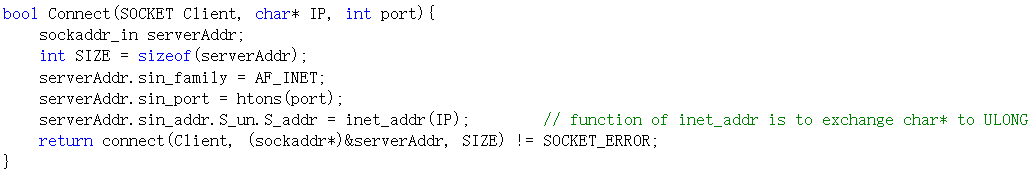
同样调用socket库中的listen函数进行监听，监听成功返回true。

⑤服务器接受连接



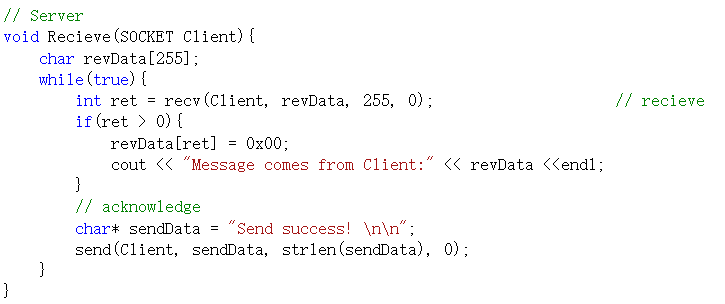
传入的参数是服务器的SOCKET，通过调用accept函数对客户端的连接请求进行响应，返回一个客户端的SOCKET。

⑥客户端请求连接



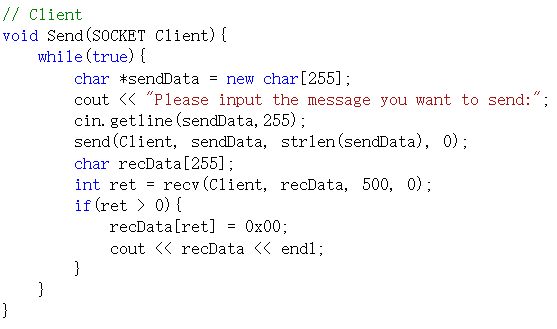
主函数传入的参数是客户端的SOCKET，客户端所要连接的IP和端口port，通过调用connect函数对对应IP与端口上的服务器进行连接，连接成功返回true。

⑦服务器接收数据



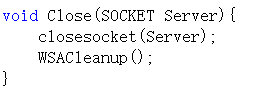
主函数传入客户端SOCKET，对客户端传送的数据通过recv函数进行接收，接收大小为255字节，接收完毕后最后通过send函数对客户端进行应答。

⑧客户端发送数据



主函数中传入客户端的SOCKET，首先通过getline，输入需要发送的消息，再通过send函数将消息发送给服务器。

⑨关闭连接

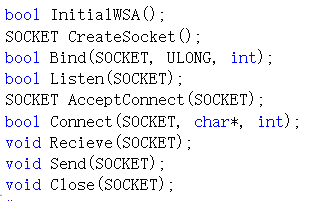


通过调用closesocket函数关闭SOCKET，然后要调用WSACleanup清除WSA。

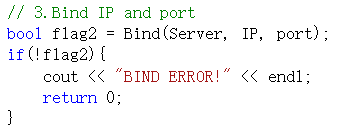
（3）异常处理

C/S建立的每个步骤都会进行一次异常检测，如有出错则会及时报错并退出程序，以防止程序运行造成的恶性后果。

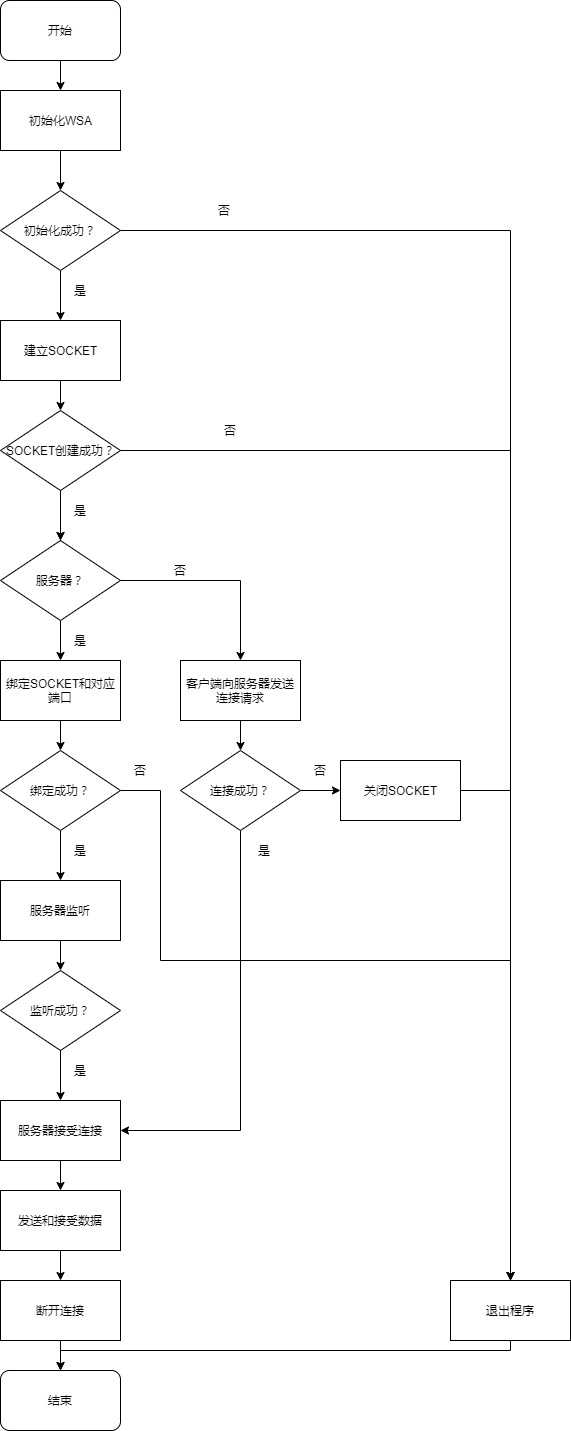
在初始化WSA、绑定SOCKET和端口、监听、连接过程中，主函数中调用其函数，函数调用后会返回bool类型的参数，用于判断该过程是否执行成功。如果是真，则说明该步骤成功，可以继续往后执行，如果为假，说明该步骤失败，需及时退出程序。



以服务器绑定IP和端口为例，调用头文件中的绑定函数后会返回一个bool值，对其进行判断，如果为假，说明绑定步骤出现问题，输出提示，然后退出程序。



（4）程序结构和流程图



（5）测试方案

直接启动客户端不启动服务器，观察程序运行结果。

先启动服务器程序，再启动客户端程序，然后在客户端中输入”Hello Server!”，观察服务器程序中的运行结果。

**实验数据：**

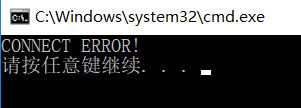
输入数据：在客户端中输入”Hello Server!”。

预期输出数据：服务器中显示”Hello Server!”信息。

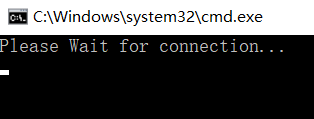
**实验结果与分析：**

（1）程序运行结果

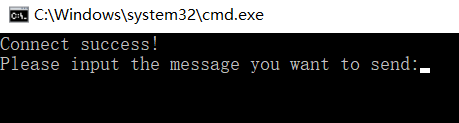
直接启动客户端不启动服务器，结果如下图所示。



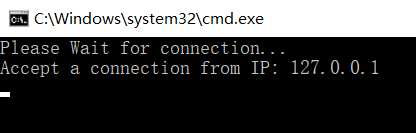
先启动服务器，结果如下图所示。



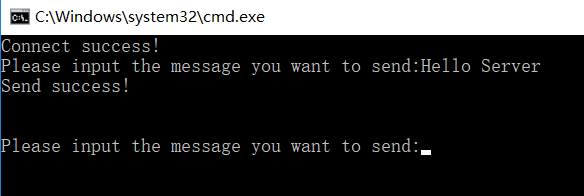
再启动客户端，结果如下图所示。



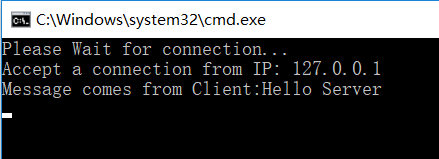
再查看服务器。



在客户端中输入”Hello Server”，运行后结果如下。

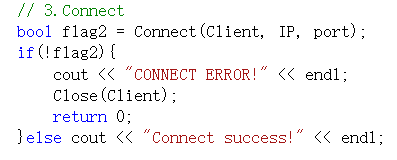


然后查看服务器。

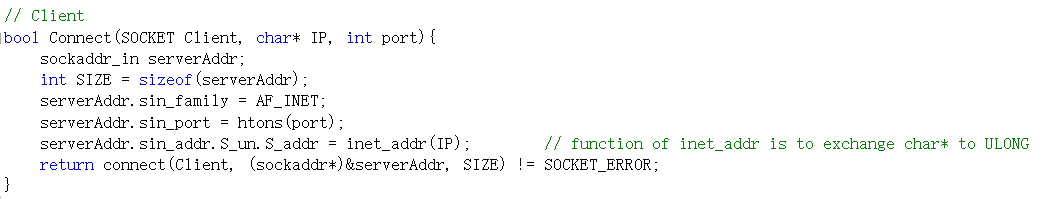


（2）运行结果分析

直接启动客户端不启动服务器会出现连接失败的情况，分析程序可以发现是在下图出现了错误，即连接失败。



其Connect函数是在头文件中进行声明实现的，如下所示。



之所以会出现连接失败，是因为没有启动服务器，在对应的端口port上没有服务器接受客户端发出的连接请求，所以会连接失败。

先启动服务器，通过运行结果可以看出，此时说明服务器正在等待客户端的连接请求，然后再运行客户端，通过服务器和客户端的运行结果可以看出，此时客户端已经成功连接上服务器。

在客户端中输入”Hello Server”后的运行结果，可以发现服务器成功接收了客户端发送的信息并显示，并且还应答回复了客户端，此时实验任务要求的功能已经实现。

（3）实验结论

通过本次实验了解了C/S结构的通信，即客户端-服务器通信方式，利用基于TCP/IP协议的SOCKET，了解了SOCKET的搭建过程和服务器与客户端之间的连接过程，同时也认识了他们之间的通信方式。

（4）讨论

本次实验开发的程序只适用于一对一的C/S架构，如果有多个客户端同时请求连接的话只能成功连接一个。在之后实验会实现一对多的C/S架构体系。

**重要程序代码：**

|  |
| --- |
| // Server.cpp : 定义控制台应用程序的入口点  //  /\*\*\*\*\* winsock c/s建立过程（服务器）\*\*\*\*\*\*\*  1.初始化WSA  2.建立socket  3.绑定socket绑定对应端口  4.监听  5.接受连接  6.发送和接受数据  7.断开连接  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include "stdafx.h"  #include <iostream>  #include <stdio.h>  #include <WinSock2.h>  #include "Socket.h"  #pragma comment(lib, "ws2\_32.lib") // 手工导入链接ws2\_32.lib库，ws2\_32.lib是winsock2的库文件  using namespace std;  int \_tmain(int argc, \_TCHAR\* argv[])  {  int port = 8080; // set port  ULONG IP = INADDR\_ANY; // set accept All ip  // char\* IP = 127.0.0.1;  // 1.initial WSA  bool flag1 = InitialWSA();  if(!flag1) return 0;  // 2.Create socket  SOCKET Server = CreateSocket();  if(Server == INVALID\_SOCKET){  cout << "SOCKET ERROR!" << endl;  return 0;  }  // 3.Bind IP and port  bool flag2 = Bind(Server, IP, port);  if(!flag2){  cout << "BIND ERROR!" << endl;  return 0;  }    // 4.listin  bool flag3 = Listen(Server);  if(!flag3){  cout << "LISTEN ERROR!" << endl;  return 0;  }    // 5.Accept Connect  SOCKET Client = AcceptConnect(Server);    // 6.recieve data  Recieve(Client);    // 7.Close  Close(Server);  return 0;  } |

服务器主函数Server.cpp

|  |
| --- |
| // Client.cpp : 定义控台应用程序的入口点。  //  /\*\*\*\*\* winsock c/s建立过程（客户端）\*\*\*\*\*\*\*  1.初始化WSA  2.建立socket  3.连接到服务器  4.发送和接受数据  5.断开连接  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include "stdafx.h"  #include "stdafx.h"  #include "Socket.h"  #include <iostream>  #include <stdio.h>  #include <WinSock2.h>  #pragma comment(lib, "ws2\_32.lib")  using namespace std;  int main(){  int port = 8080; // set port  char\* IP = "127.0.0.1"; // set IP  // 1.initial WSA  bool flag1 = InitialWSA();  if(!flag1) return 0;  // 2.Create socket  SOCKET Client = CreateSocket();  if(Client == INVALID\_SOCKET){  cout << "SOCKET ERROR!" << endl;  return 0;  }  // 3.Connect  bool flag2 = Connect(Client, IP, port);  if(!flag2){  cout << "CONNECT ERROR!" << endl;  Close(Client);  return 0;  }else cout << "Connect success!" << endl;  // 4.send and recieve  Send(Client);  // 5.close  closesocket(Client);  WSACleanup();  return 0;  } |

客户端主函数Client.cpp

|  |
| --- |
| #ifndef \_\_SOCKET\_\_  #define \_\_SOCKET\_\_  #include "stdafx.h"  #include "Socket.h"  #include <iostream>  #include <WinSock2.h>  using namespace std;  bool InitialWSA();  SOCKET CreateSocket();  bool Bind(SOCKET, ULONG, int);  bool Listen(SOCKET);  SOCKET AcceptConnect(SOCKET);  bool Connect(SOCKET, char\*, int);  void Recieve(SOCKET);  void Send(SOCKET);  void Close(SOCKET);  #endif |

Socket头文件Socket.h

|  |
| --- |
| #include "stdafx.h"  #include "Socket.h"  #include <iostream>  #include <string>  #include <WinSock2.h>  using namespace std;  bool InitialWSA(){  WORD sockVersion = MAKEWORD(2, 2); // version of winsock, that is winsock2  WSADATA wsaData; // store information about winsock returned by system  return WSAStartup(sockVersion, &wsaData) == 0;  }  SOCKET CreateSocket(){  return socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP); // address family, connect, TCP  }  // Server  bool Bind(SOCKET Server, ULONG IP, int port){  sockaddr\_in in;  in.sin\_family = AF\_INET;  in.sin\_port = htons(port); // Converts a 16 bit byte order from host to network  in.sin\_addr.S\_un.S\_addr = IP;  return bind(Server, (LPSOCKADDR)&in, sizeof(in)) != SOCKET\_ERROR;  }  // Server  bool Listen(SOCKET Server){  return listen(Server, 10) != SOCKET\_ERROR; // max = 10;  }  // Server  SOCKET AcceptConnect(SOCKET Server){  SOCKET Client;  sockaddr\_in ClientAddr;  int LenOfAddr = sizeof(ClientAddr);  cout << "Please Wait for connection..." << endl;  Client = accept(Server, (SOCKADDR\*)&ClientAddr, &LenOfAddr);  if(Client == INVALID\_SOCKET){  cout << "ACCEPT ERROR!" << endl;  return 0;  }  cout << "Accept a connection:" << inet\_ntoa(ClientAddr.sin\_addr) << endl;  return Client;  }  // Client  bool Connect(SOCKET Client, char\* IP, int port){  sockaddr\_in serverAddr;  int SIZE = sizeof(serverAddr);  serverAddr.sin\_family = AF\_INET;  serverAddr.sin\_port = htons(port);  serverAddr.sin\_addr.S\_un.S\_addr = inet\_addr(IP); // function of inet\_addr is to exchange char\* to ULONG  return connect(Client, (sockaddr\*)&serverAddr, SIZE) != SOCKET\_ERROR;  }  // Server  void Recieve(SOCKET Client){  char revData[255];  while(true){  int ret = recv(Client, revData, 255, 0); // recieve  if(ret > 0){  revData[ret] = 0x00;  cout << "Message comes from Client:" << revData <<endl;  }  // acknowledge  char\* sendData = "Send success! \n\n";  send(Client, sendData, strlen(sendData), 0);  }  }  // Client  void Send(SOCKET Client){  while(true){  char \*sendData = new char[255];  cout << "Please input the message you want to send:";  cin.getline(sendData,255);  send(Client, sendData, strlen(sendData), 0);  char recData[255];  int ret = recv(Client, recData, 500, 0);  if(ret > 0){  recData[ret] = 0x00;  cout << recData << endl;  }  }  }  void Close(SOCKET Server){  closesocket(Server);  WSACleanup();  } |

Socket头文件的实现Socket.cpp

**指导教师评语：（空白，半页A4）**