

**SHANGHAI JIAO TONG UNIVERSITY**

**实验报告**

LAB REPORT



**数据通信**

Winsock ex2

姓 名: 裘炜程

学 号: 516030910287

班 级: F1603602

一、实验要求

1、Write a stream based echo server printing out the message received from the client, echoing it back, until the client closes the connection.

Write a stream based echo client sending messages to the echo server, receiving each message returned by the server. Terminate the connection when “quit” is entered.

2、Modify your solution to Exercise 2 to write a stream based echo server, which can simultaneously handle multiple clients connecting to it. No modification of the client code is necessary, but multiple instances of the client should be started. Hint: use Windows threads functions.

二、实验原理

1、EchoClient基本步骤

（1）初始化Winsock：WSAStartup( )

（2）创建socket:socket( )

（3）接受地址信息，result以链表储存该信息，每一个节点存放一个ip及其信息

（4）根据result连接server：connnect( )

（5）发送接收数据：send( )、receive( )

（6）关闭连接：shutdown( ), closesocket( ), WSACleanup( )

2、EchoServer基本步骤

（1）初始化Winsock：WSAStartup( )

（2）创建socket:socket( )

（3）接受地址信息，result以链表储存该信息，每一个节点存放一个ip及其信息

（4）绑定socket：bind( )

（5）在socket上监听：listen( )

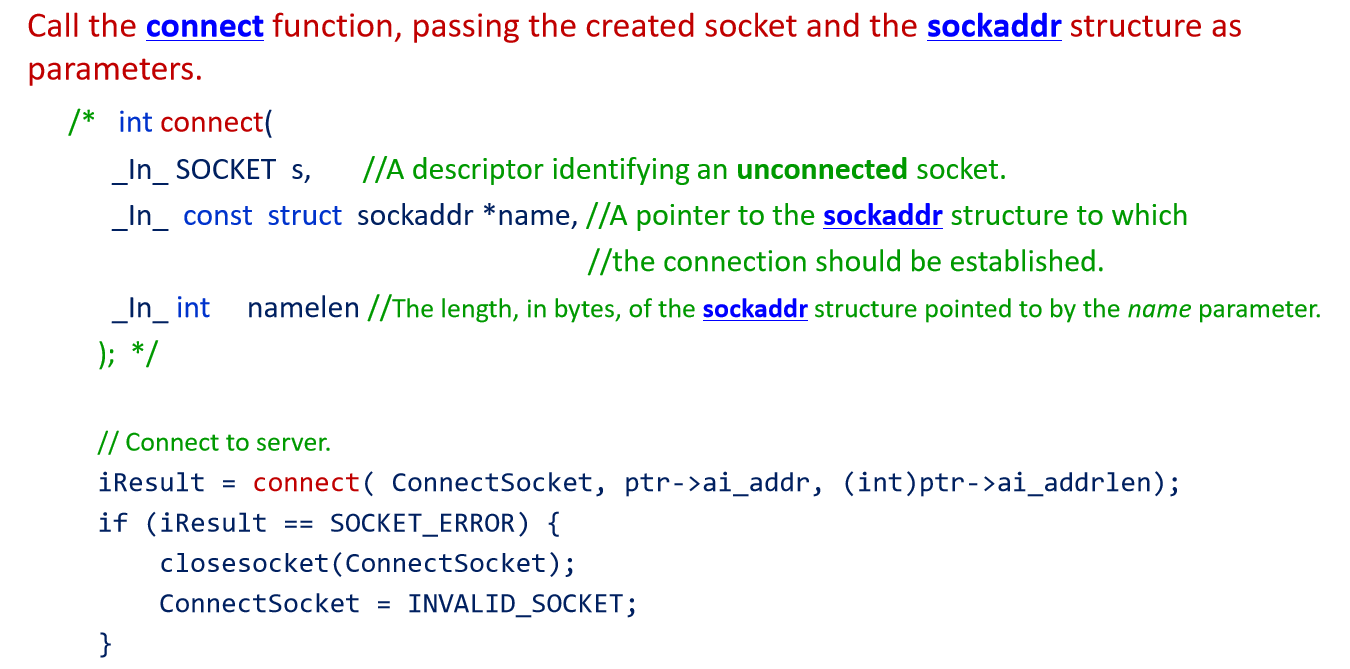
（6）接受客户端发来的请求：accept( )，并创建一个ClientSocket

（7）用ClientSocket发送接收数据：send( )、receive( )

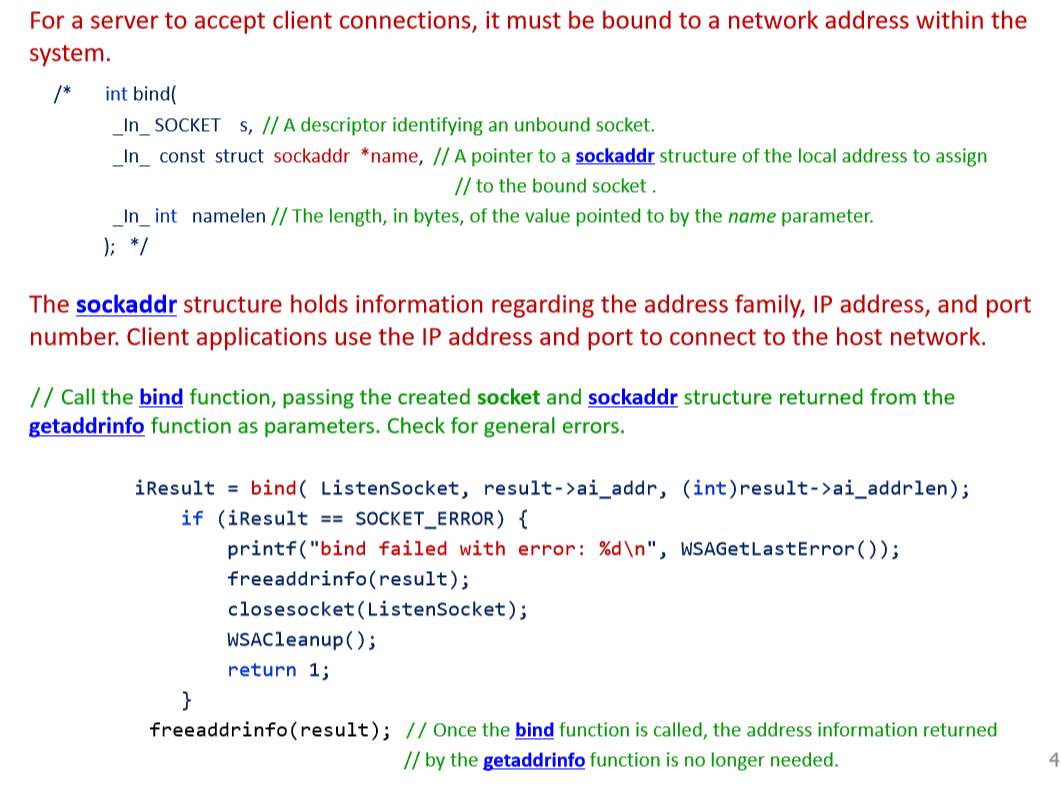
（8）关闭连接：shutdown( ), closesocket( ), WSACleanup( )

3、核心函数

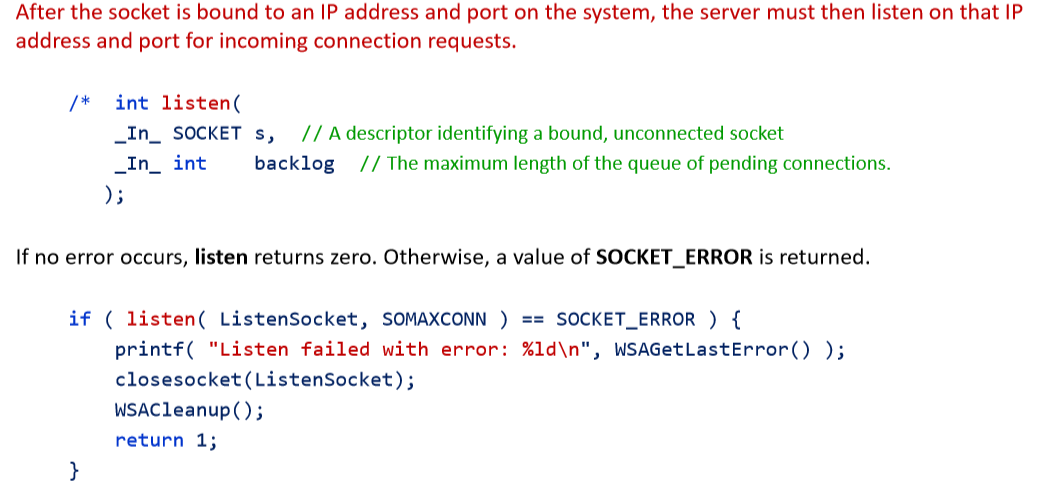
（1）connect( )：客户端，传入创建好的socket和地址信息



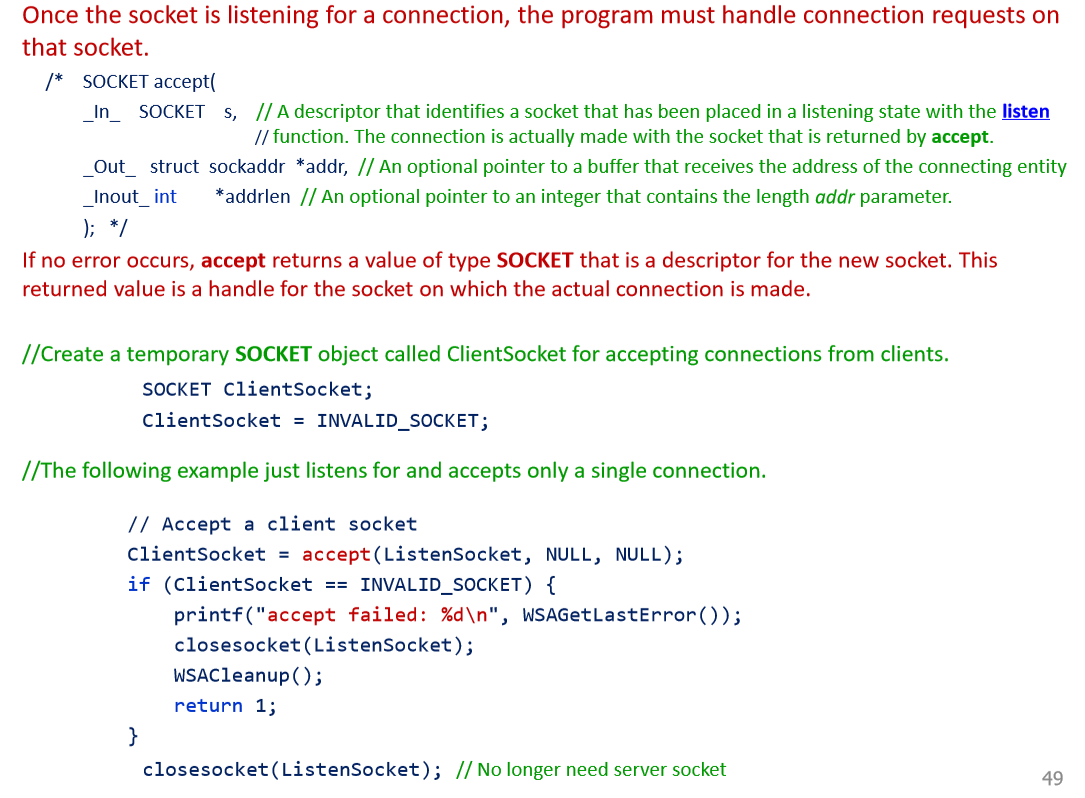
（2）bind( )：服务器，传入创建好的socket和地址信息



（3）listen( ):服务器监听，传入上一步创建的socket和最大连接数



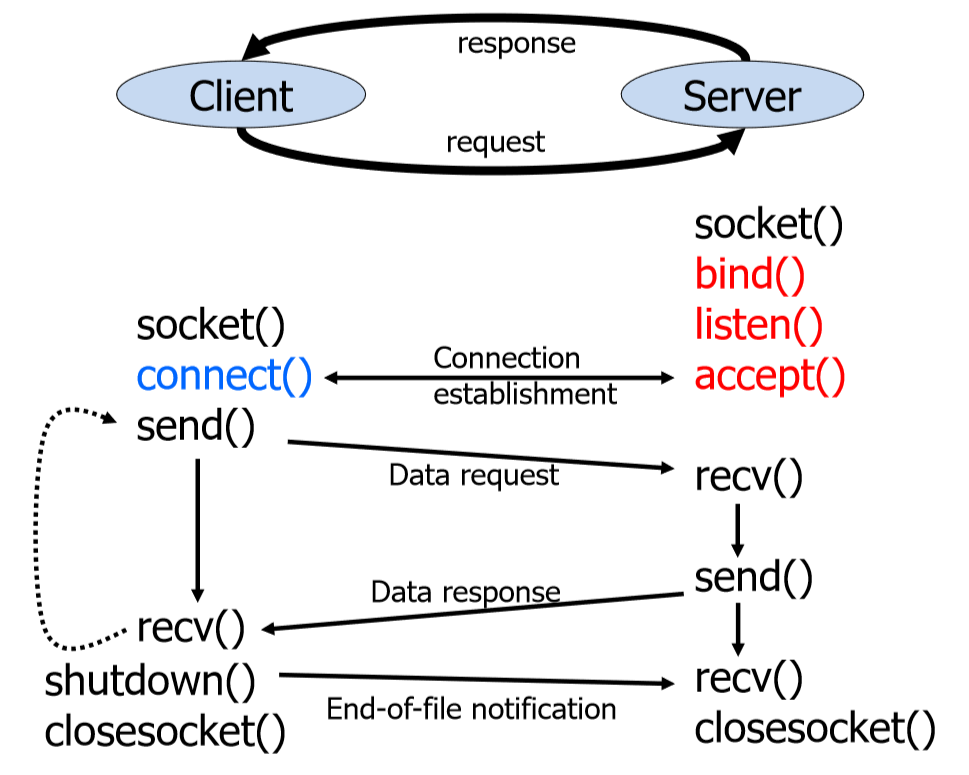
（4）accept( )：服务器，传入监听的socket，传出收到的地址，返回一个客户端socket结构

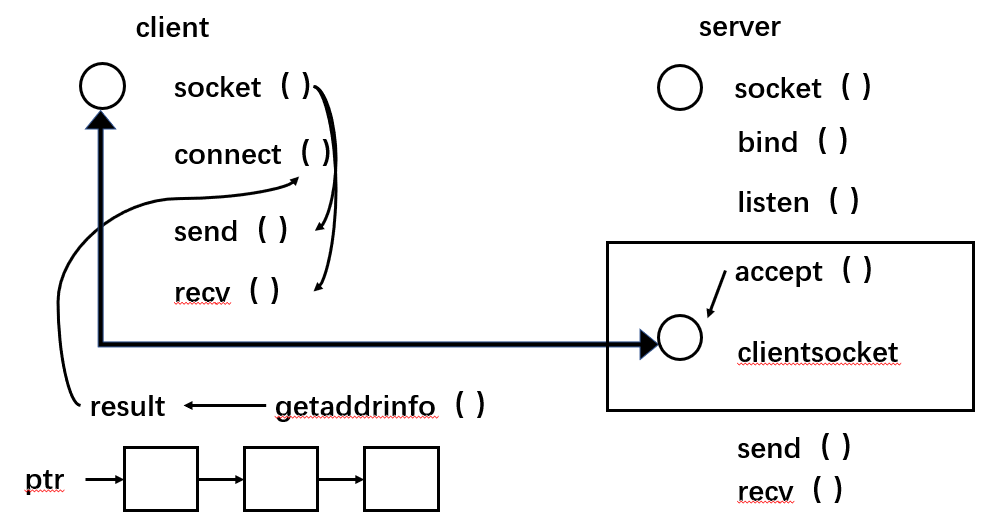


（5）send( )、recv( )：客户端和服务器接受发送数据，输入发送的字符串和长度



其他的函数已经在第一次实验报告中分析过了，这里就不再说明。总体来说服务器和客户端的连接过程和函数连接为：



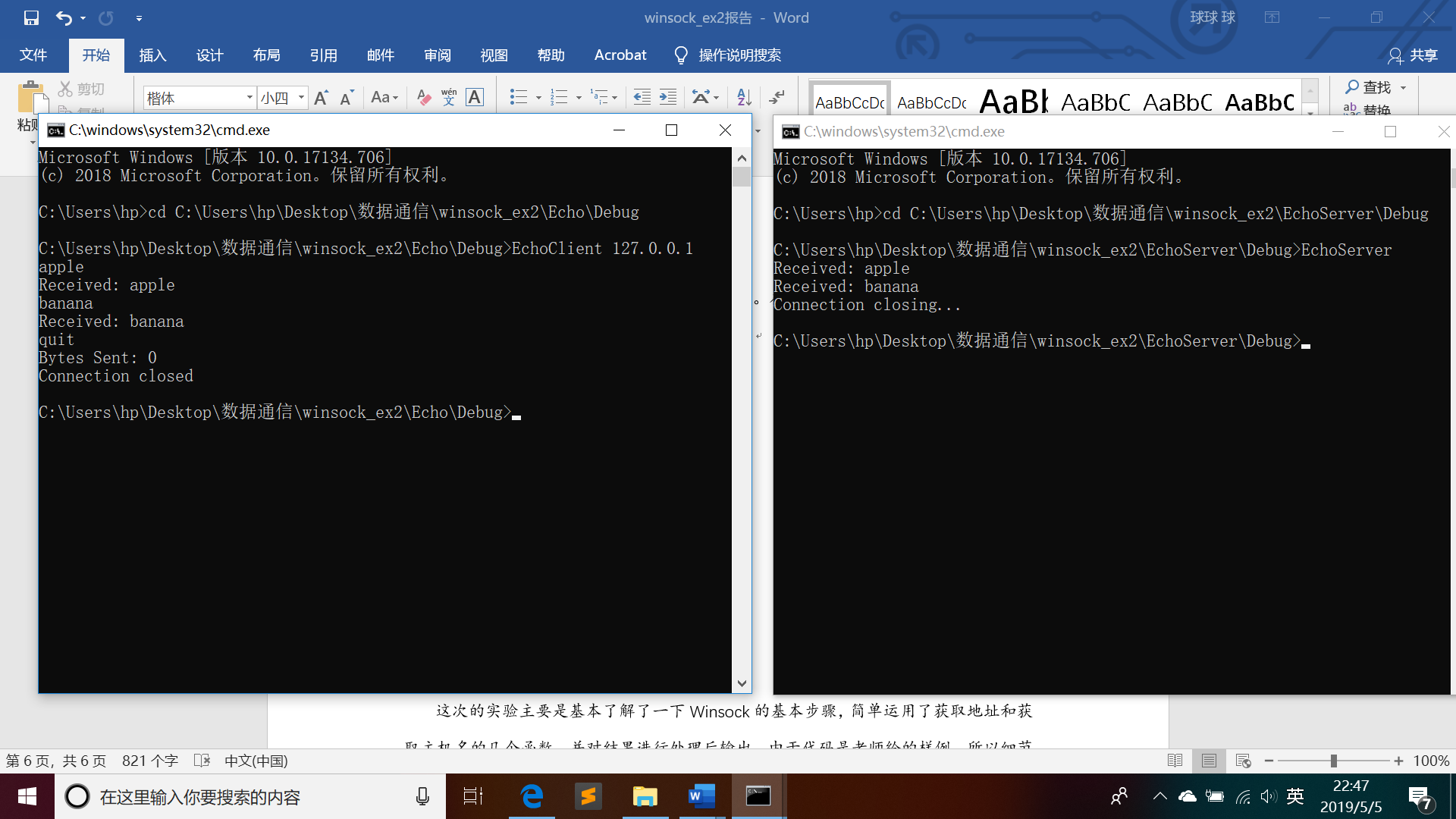


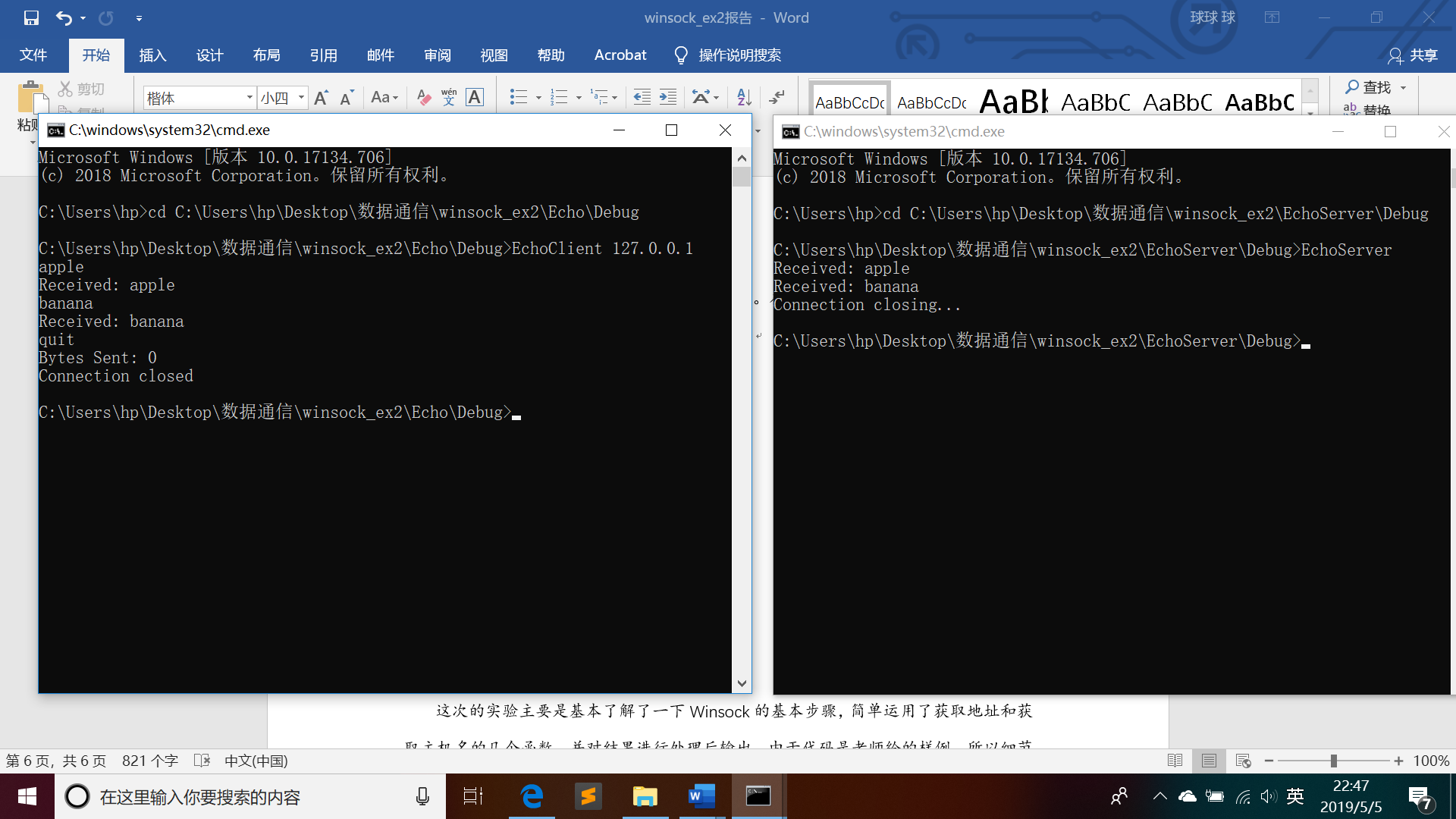
4、多线程处理

不断地接受客户端的请求，直到客户端关闭。在循环中不断创建clientsocket，然后对于每一个线程进行send和recv的处理。

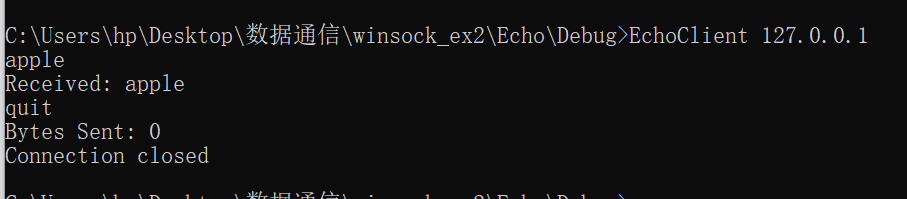
三、实验结果

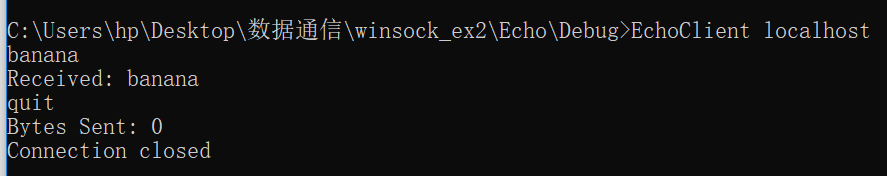
1、EchoClient和EchoServer





2、EchoClient和多线程的EchoServer2







四、实验思考

在代码中出现一个问题：在每一次send时总要多发送一位，如：send(ConnectSocket, sendbuf, int(strlen(sendbuf)+1), 0)。原因是每一次发送的数据放在char中，并固定了位数，而char存放数据时会在字符串最后加一位终结符，因此在发送时需要将这一位一起发送，表示在传到终结符一位时数据结束，接收端收到结束信息就可以打印。如果不发送终结符，接收端会将整个char包括缓冲区的东西全部打印，因而出现“烫烫烫”。

在实验中，其他同学发现如果在客户端输入结束命令CTRL+C，服务器端会显示异常，说明本来代码中并没有对CTRL+C进行处理。因此我的解决方法是分析CTRL+C发送时的情况，实际上在发送这个的时候会先发一个空串，然后返回一个-1，而quit是返回0，因此在接受时将recv((unsigned int)client\_s, in\_buf, sizeof(in\_buf), 0) == 0的判断改为<=0即可，但是还存在的问题是空串依然会被打印出来，这个问题如何解决还没有想到。