

socket编程实验报告

学号：2013622 姓名：罗昕珂

一、实验要求：

1. 使用流式Socket，设计一个两人聊天协议，要求聊天信息带有时间标签。请完整地说明交互消息的类型、语法、语义、时序等具体的消息处理方式。
2. 对聊天程序进行设计。给出模块划分说明、模块的功能和模块的流程图。
3. 在Windows系统下，利用C/C++对设计的程序进行实现。程序界面可以采用命令行方式，但需要给出使用方法。编写程序时，只能使用基本的Socket函数，不允许使用对socket封装后的类或架构。
4. 对实现的程序进行测试。
5. 撰写实验报告，并将实验报告和源码提交。

二、实验原理

socket实现的过程

- 服务端：建立socket，声明自身的端口号和地址并绑定到socket，使用listen打开监听，然后不断用accept去查看是否有连接，如果有，捕获socket，并通过recv获取消息的内容，通信完成后调用closeSocket关闭这个对应accept到的socket，如果不再需要等待任何客户端连接，那么用closeSocket关闭掉自身的socket。
 - 客户端：建立socket，通过端口号和地址确定目标服务器，使用Connect连接到服务器，send发送消息，等待处理，通信完成后调用closeSocket关闭socket。
1. listen：服务器在准备接受客户端的握手请求之前，需要准备半连接队列和全连接队列，准备好之后才能接收握手请求。
 2. connect：客户端选择了一个可用端口，然后向服务器发起握手请求了。同时自己还开了个定时器，如果逾期收不到反馈会重试。
 3. 客户端发起连接请求之后，三次握手的工作就由双方的内核完成了。三次握手成功之后，服务器端会创建一个sock对象，在它上面保存好tcp连接的四元组信息，然后放在接收队列中。
 4. accept：从这个接收队列中获取一个握手就绪连接来用。
 5. 连接之上的读和写：用户流程只需要发起读写请求，放到接收缓存或者发送缓存中。真正的读写，重试都由内核从缓存中取数据，或者写入。

三、实验编程步骤

服务端

1. 加载套接字库(WSAStartup());

```
//初始化WSA ,加载Winsock库
WORD sockVersion = MAKEWORD(2,2); //声明使用socket2.2版本
WSADATA wsaData; //存放被WSAStartup函数调用后返回的Windows Sockets数据
//初始化socket资源
if(WSAStartup(sockVersion, &wsaData)!=0)
{
    return 0; //初始化失败
}
```

2. 创建套接字 (socket())

```
//创建流式监听套接字
SOCKET ServerSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
//地址类型为AD_INET, 服务类型为流式(SOCK_STREAM), 协议采用TCP
if(ServerSocket == INVALID_SOCKET)
{
    printf("socket创建失败 !\n");
    return 0;
}
```

3. 绑定套接字到一个IP地址和一个端口上 (bind());

```
//绑定IP和端口
sockaddr_in listenAddr;
listenAddr.sin_family = AF_INET; //地址类型为AD_INET, 即IP格式
listenAddr.sin_port = htons(7777); //绑定本地监听端口: 7777
listenAddr.sin_addr.S_un.S_addr = INADDR_ANY;
if(bind(ServerSocket, (LPSOCKADDR)&listenAddr, sizeof(listenAddr)) ==
SOCKET_ERROR)
{
    printf("端口绑定失败 !\n");
    closesocket(ServerSocket);
    return 0;
}
```

4. 将套接字设置为监听模式等待连接请求 (listen());

```
//绑定成功就开始监听
if(listen(ServerSocket, 5) == SOCKET_ERROR)
{
    printf("监听失败 !\n");
    return 0;
}
```

5. 请求到来后, 接受连接请求, 返回一个新的对应于此连接的套接字 (accept());

```

//SOCKET Command_Sock = accept(Listen_Sock,...)
ClientSocket = accept(ServerSocket, (SOCKADDR *)&acceptAddr,
&ClientAddrlen);
if(ClientSocket == INVALID_SOCKET)
{
    printf("服务器接收请求失败 !\n");
    //重新等待连接
    continue;
}

printf("成功接受到一个连接\n");

```

6. 用返回的套接字和客户端进行通信 (send()/recv()) ;

```

//接收数据
//等待客户接入 revData[1024].
//接收数据: recv(Command_Sock, buf, ...)

//memset(revData, 0, sizeof(revData));
int retlen = recv(ClientSocket, revData, 1024, 0);
if(retlen > 0)
{
    revData[retlen] = 0x00;
    time_t nowtime;
    struct tm *sysTime;
    nowtime = time(NULL); //获取日历时间
    sysTime=localtime(&nowtime); //转换为系统的日期
    printf("%d-%d-%d  %d:%d:%d: %s\n", 1900+sysTime->tm_year,sysTime-
>tm_mon+1,sysTime->tm_mday,sysTime->tm_hour,sysTime->tm_min,sysTime-
>tm_sec,revData);
}

//发送数据
//发送数据: send(Command_Sock, buf, ...)
const char * sendData = "请输入: ";
send(ClientSocket, sendData, strlen(sendData), 0);

```

7. 返回, 等待另一个连接请求;

```

//关闭客户端socket
closesocket(ClientSocket);

```

8. 关闭套接字, 关闭加载的套接字库 (closesocket()/WSACleanup()) ;

```

//关闭服务端socket
closesocket(ServerSocket);

```

```
//释放Winsock库, 释放socket库  
WSACleanup();
```

客户端

1. 加载套接字库, 创建套接字 (WSAStartup()/socket()) ;

```
WORD sockVersion = MAKEWORD(2, 2);  
WSADATA data;  
//存放被WSAStartup函数调用后返回的Windows Sockets数据的数据结构  
if(WSAStartup(sockVersion, &data)!=0)  
{  
    return 0;  
}
```

2. 创建套接字 (socket())

```
//创建流式通讯socket  
SOCKET ClientSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);  
//地址类型为AD_INET, 服务类型为流式(SOCK_STREAM), 协议采用TCP  
if(ClientSocket == INVALID_SOCKET)  
{  
    printf("socket创建失败 !\n");  
    return 0;  
}
```

3. 向服务器发出连接请求 (connect()) ;

```
//连接目的IP地址和端口  
sockaddr_in ServerAddr;  
ServerAddr.sin_family = AF_INET;  
ServerAddr.sin_port = htons(7777);  
ServerAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");  
//客户端: 请求与服务端连接  
//int ret = connect(Client_Sock, ...)  
if(connect(ClientSocket, (sockaddr *)&ServerAddr, sizeof(ServerAddr)) ==  
SOCKET_ERROR)  
{ //连接失败  
    printf("连接失败 !\n");  
    closesocket(ClientSocket);  
    return 0;  
}
```

4. 和服务器进行通信 (send()/recv()) ;

设0

```
//发送数据
//string data;
//cin>>data;
const char * sendData;
sendData = data.c_str();
//string转const char*
//请求与服务端连接char buf[1024].
send(ClientSocket, sendData, strlen(sendData), 0);
//send()用来将数据由指定的socket传给对方主机
//int send(int socket, const void * msg, int len, unsigned int flags)
//socket为已建立好连接的socket, msg指向数据内容, len则为数据长度, 参数flags一般

//成功则返回实际传送出去的字符数, 失败返回-1, 错误原因存于error

//等待服务端处理以后返回数据
//接收数据
char recData[1024];
int ret = recv(ClientSocket, recData, 1024, 0);
if(ret>0){
    recData[ret] = 0x00;
    printf("%s ", recData);
}
```

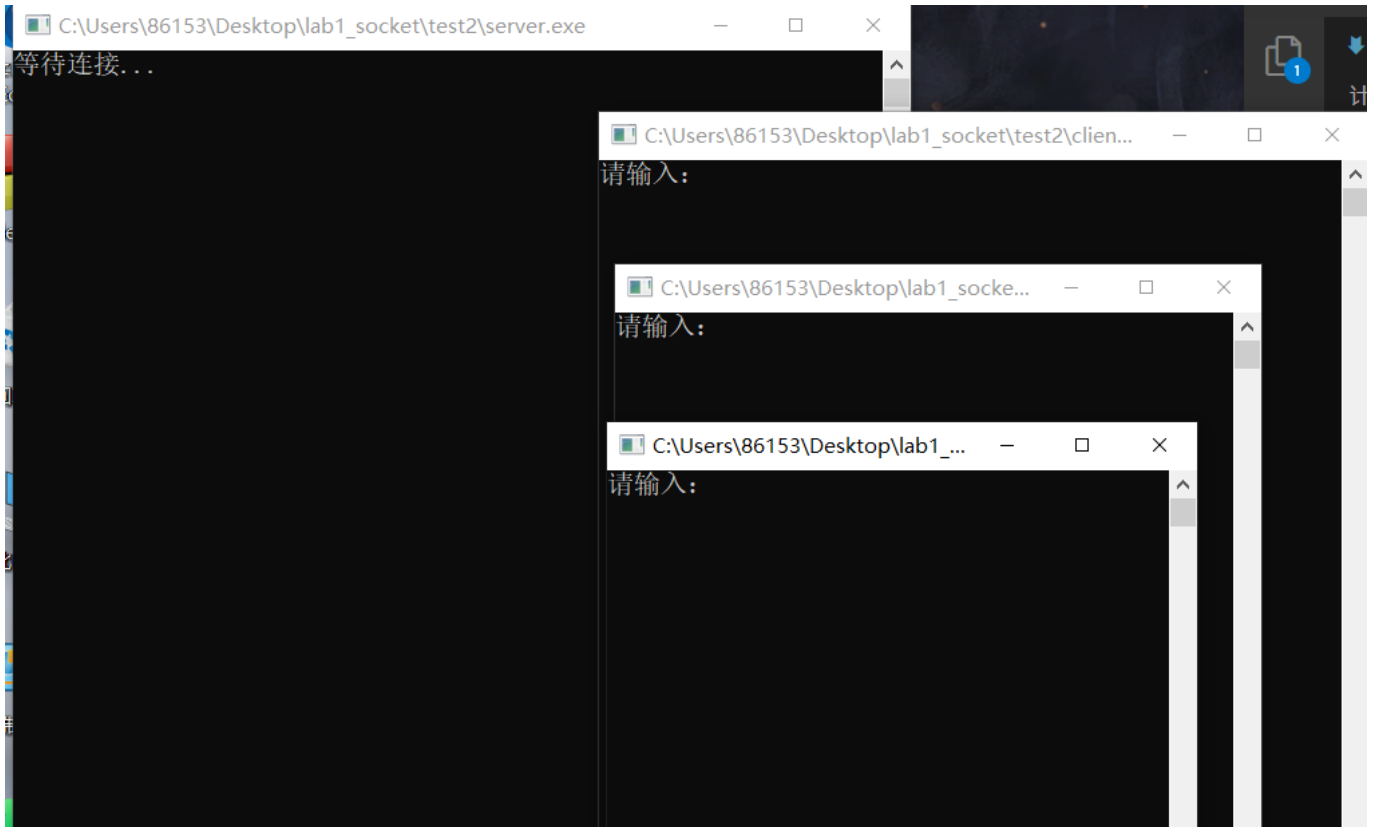
5. 关闭套接字, 关闭加载的套接字库 (closesocket()/WSACleanup()) ;

```
//关闭客户端socket
closesocket(ClientSocket);
//释放Winsock库, 释放socket库
WSACleanup();
```

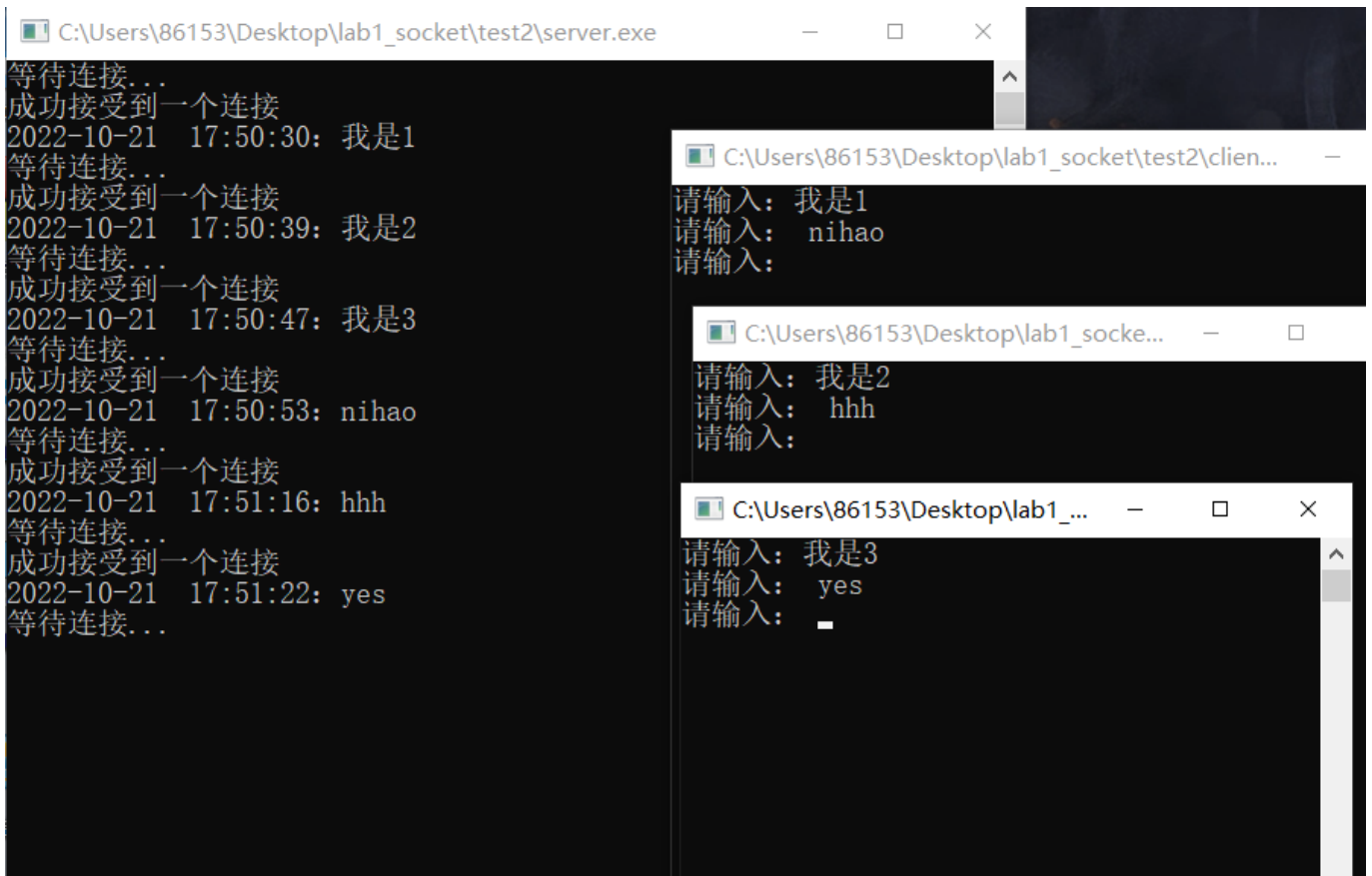
四、实验结果

- 在windows环境下用c++实现了socket编程。
- 客户端与服务器建立连接后, 由客户端先发送消息开启对话。
- 支持中英文聊天, 一次最多发送1024个字节的数据。
- 建立多个客户端的话只需要点开多个client.exe进程。
- 发送的消息全部显示在服务端, 并带有时间标签信息。

初始界面



聊天界面



五、实验中遇到的问题

1、自动在终端执行编译运行的指令，一直编译错误

```
g++ Untitled-1.cpp -o Untitled-1 ;
```

错误原因：

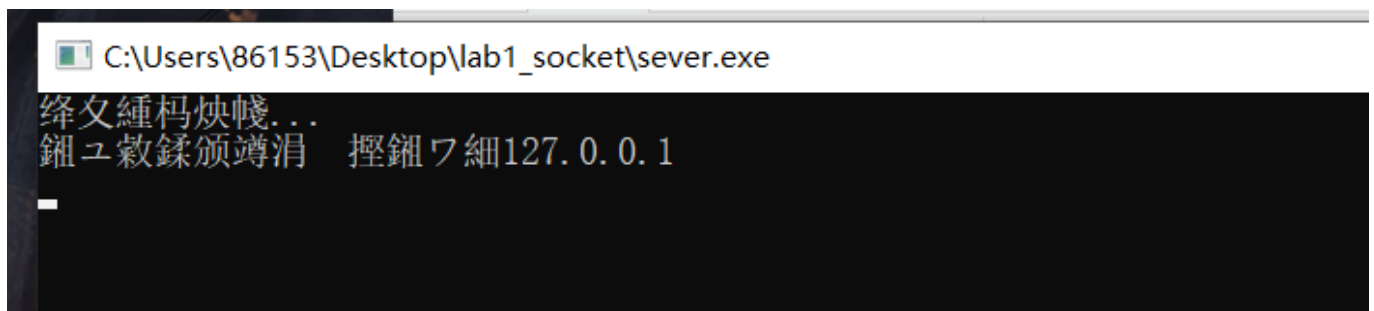
如果是使用了winsock2.h，同时又是使用gcc/g++编译，在编译时我们应该在编译指令中额外添加-lwsck32指令，而Code Runner默认下并不会添加这条指令

解决方法：

正确的编译指令应该是：

```
g++ xxx.cpp -o xxx.exe -lwsck32
```

2、命令行下显示为乱码



错误原因：



因为cpp文件是UTF-8格式的，但它编译的时候原封不动地抄进二进制文件里。中文的Windows默认用GB2312编码，这样就会产生乱码了。

解决方法：

```
g++ -finput-charset=GBK -fexec-charset=GBK client.cpp -o client.exe -lwsck32
```

-finput-charset 指定源文件的编码，默认UTF-8 -fexec-charset 指定可执行程序的编码，默认UTF-8

3、在命令行无法输入数据

错误原因：

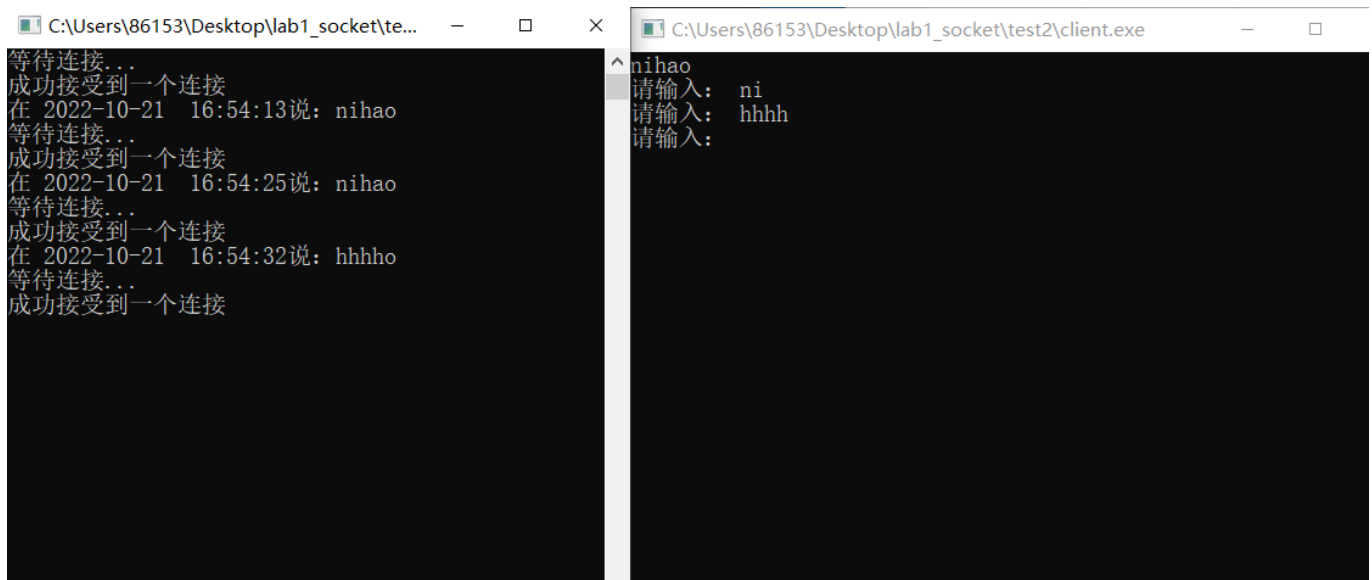
输入数据类型无法被socket识别

解决方法：

用string和cin输入数据，转化为char*

```
string data;  
cin>>data;  
const char * sendData;  
sendData = data.c_str();
```

4、再次发送数据，发生了数据覆盖的问题



错误原因：

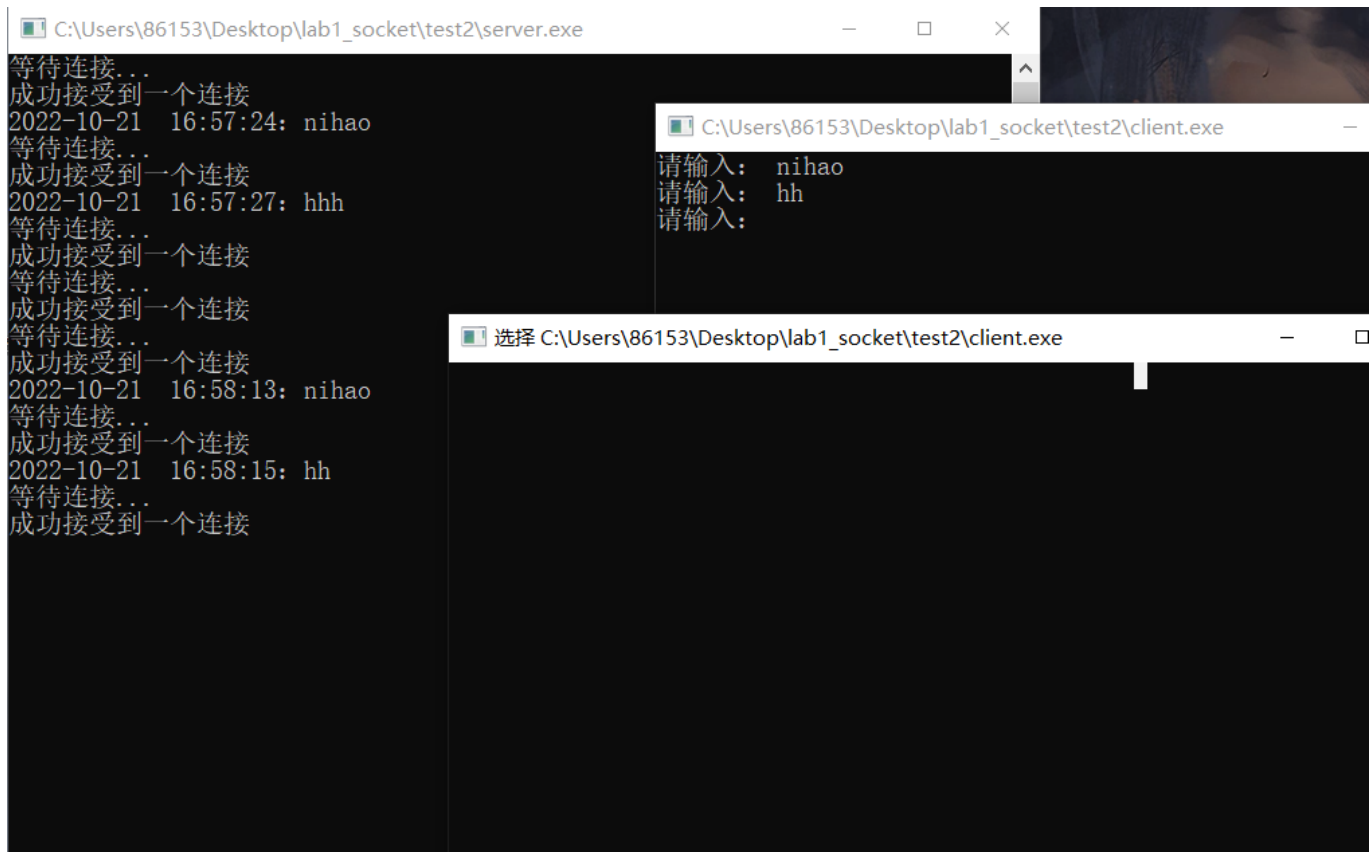
输入了一次数据以后，之前的buf数据内容还在，造成缓冲区冗余

解决方法：

在第二次交互发送信息前，把之前的数据清空，避免缓冲区冗余

```
memset(s, 0, sizeof(buffer));
```

5、一个窗口在发送消息，另一个不能发



```
C:\Users\86153\Desktop\lab1_socket\test2\server.exe
等待连接...
成功接受到一个连接
2022-10-21 16:57:24: nihao
等待连接...
成功接受到一个连接
2022-10-21 16:57:27: hhh
等待连接...
成功接受到一个连接
等待连接...
成功接受到一个连接
等待连接...
成功接受到一个连接
2022-10-21 16:58:13: nihao
等待连接...
成功接受到一个连接
2022-10-21 16:58:15: hh
等待连接...
成功接受到一个连接

C:\Users\86153\Desktop\lab1_socket\test2\client.exe
请输入: nihao
请输入: hh
请输入:

选择 C:\Users\86153\Desktop\lab1_socket\test2\client.exe
```

错误原因：

在socket连接里面，编写的socket程序是客户端先进行接收数据，再发送数据。每次输入数据以后，自动进入while（1）的循环里面，开启下一个socket连接，导致服务端无法与其他进程再连接。

解决方法：

先输入数据，之后再建立socket，就可以保证程序在“cin”处手动停下来，输入数据以后再进行连接socket。