实验1：利用Socket，设计和编写一个聊天程序

# 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至本网站。

# 实验内容

## 协议设计

* **语法**（怎么做，字段分割，哪些字段）

发送的信息支持中英文字符，内容发送格式为：发送时间+“ ”+信息的具体内容

* **语义**（做什么，信息具体含义）

发送方发送“over”时，意味着发送方想发的消息发送完成；

发送方发送“exit”，意味着发送方结束对话。

* **处理**（何时发送，收到消息的动作）

服务器端从发送方接收到消息，处理信息，获得发送信息的最后四个字符，然后发送给所有接收方；

服务器端收到“over”消息时，结束从发送方接收消息，开始从接收方接收消息；

服务器端收到“exit”消息时，关闭连接的套接字；

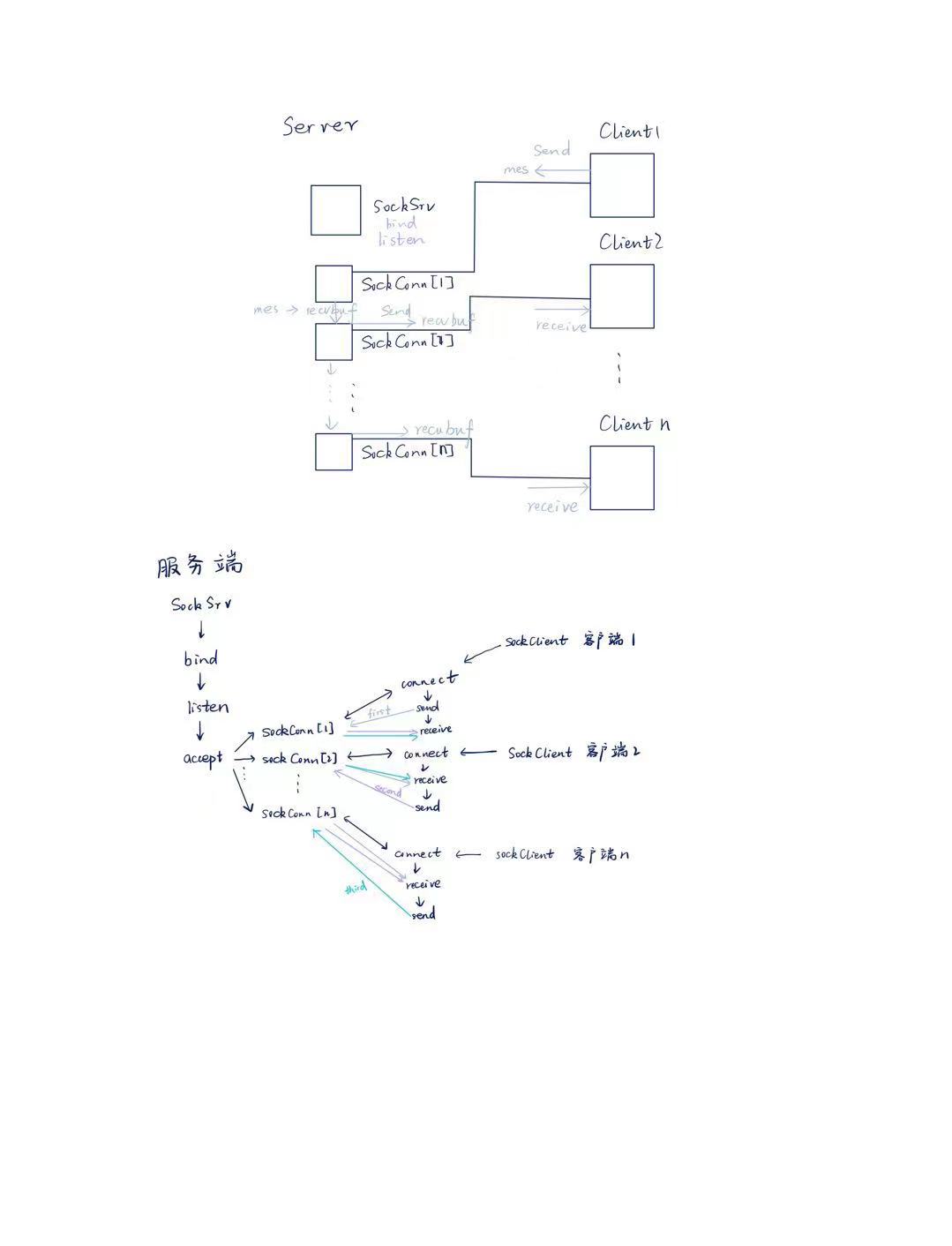
接收方收到消息时，进行处理，获得接收信息的最后四个字符；

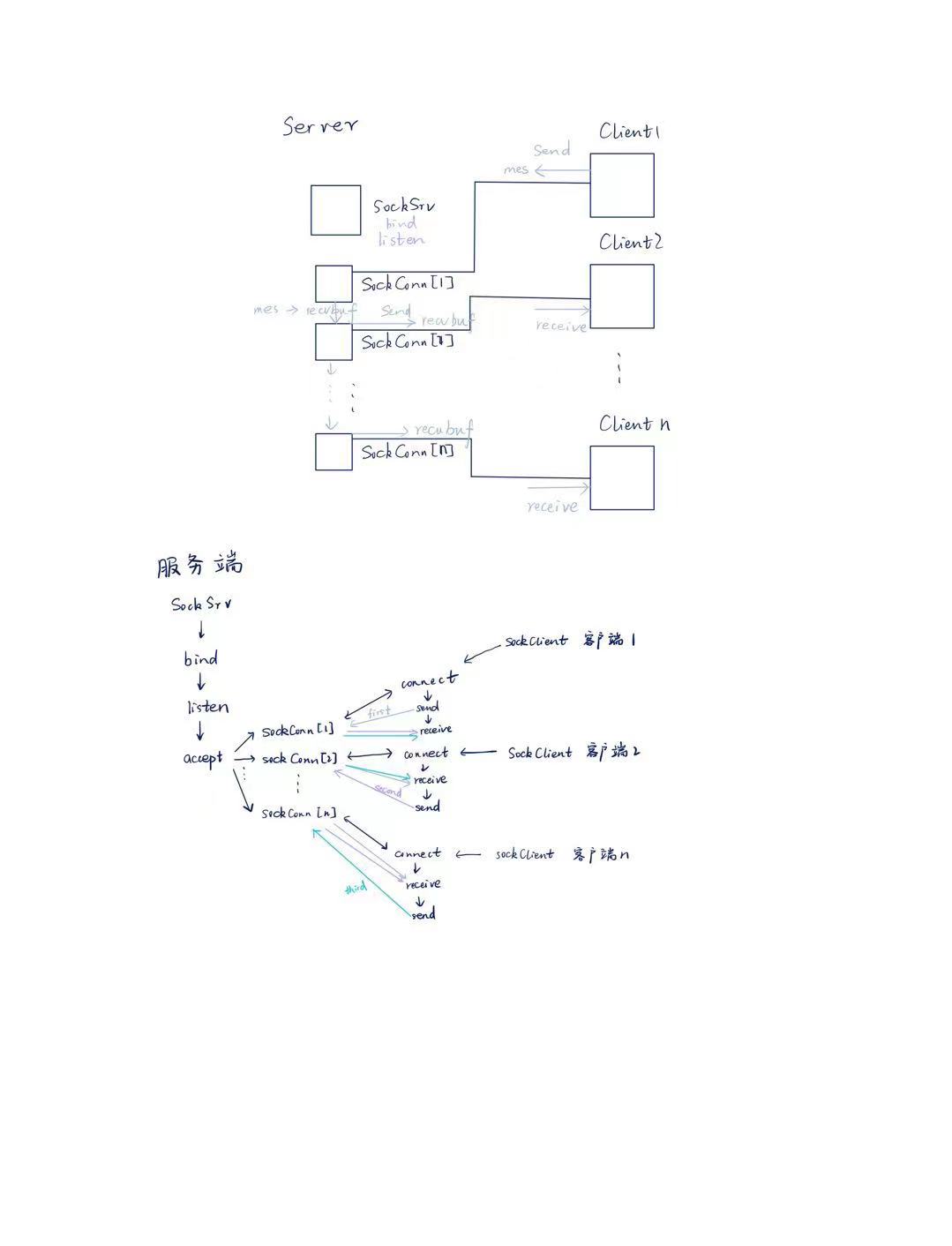
接收方收到普通消息时，正常打印；

接收方收到“over”消息时，不显示消息内容，确定下一个需要发送消息的发送方，若是自己，则开始发送消息功能，否则继续接收消息；

接收方收到“exit”消息时，打印“对方结束对话”，结束对话；

## 程序设计





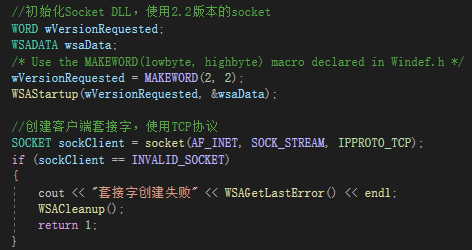
## 程序实现

### 实现基本功能

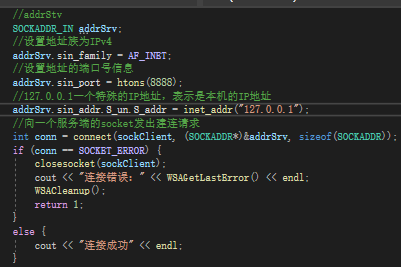
完成客户端与服务端的连接，完成客户端之间发送和接受消息。

#### 客户端代码

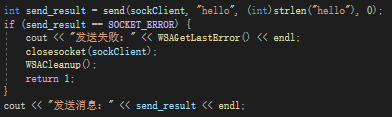
**创建客户端套接字**

****

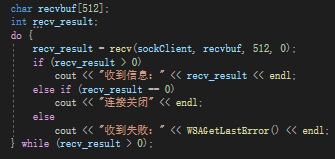
**向服务端发出建连请求**

****

**发送消息**

****

**接收消息**

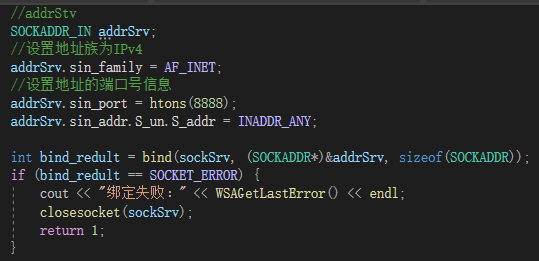
****

#### **服务端代码**

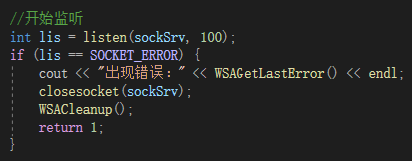
**创建服务端套接字**

****

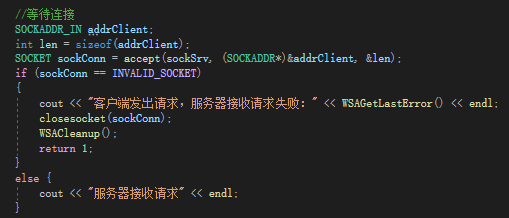
**绑定**

****

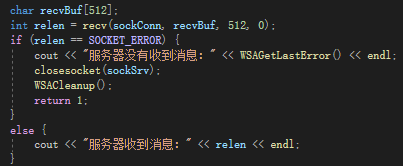
**开始监听**

****

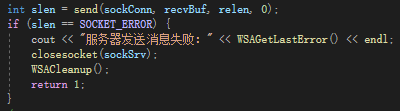
**连接客户端**

****

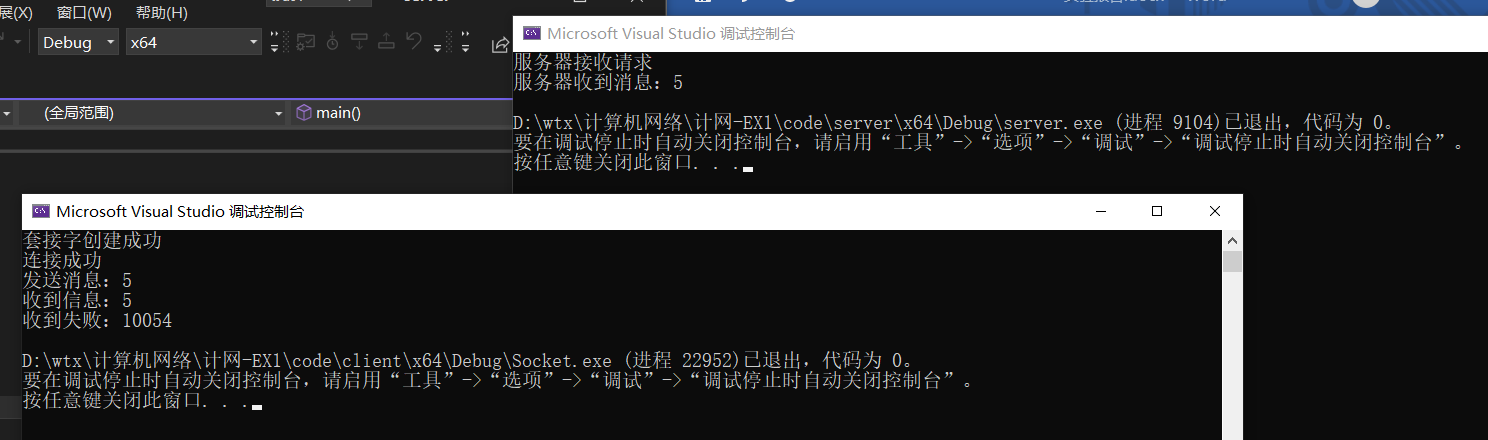
**接收消息**

****

**发送消息**

****

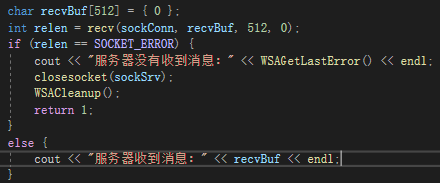
#### 运行结果

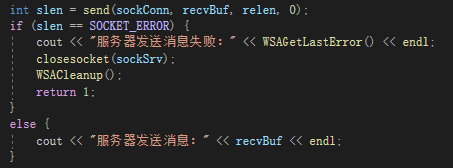
****

此时服务端和客户端可以进行连接，可以互发消息，收到的消息和发送的消息是一致的，但是客户端发送的消息并不是所希望的“Hello“，而且还未实现多轮对话的功能

### 完善功能

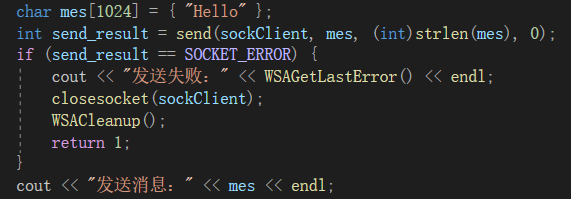
#### 消息发送正确

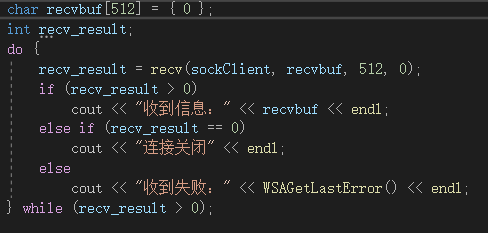




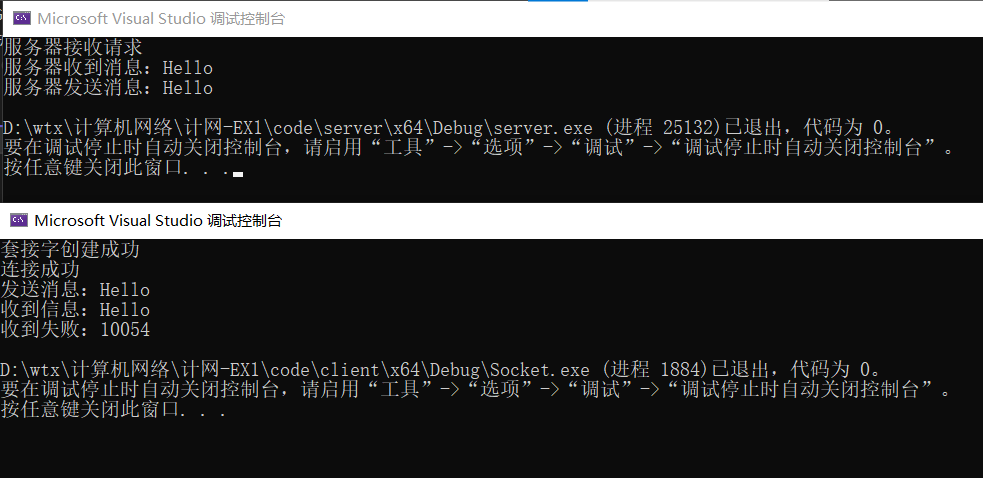
修改部分

服务端：recvBuf初始化；接收消息和发送消息成功时输出recvBuf



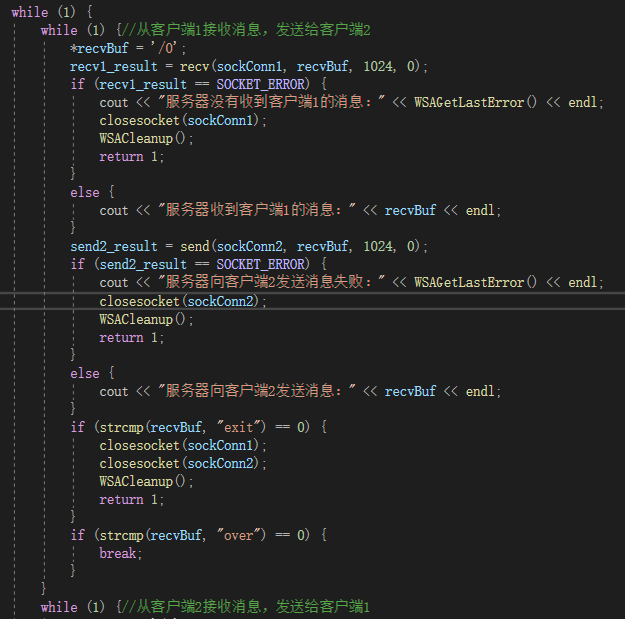


客户端：增加mes表示发送消息；recvbuf初始化；接收消息和发送消息成功时输出recvBuf

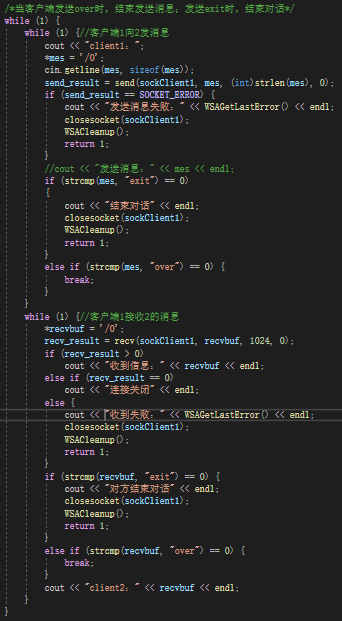


运行结果达到预期目的

#### 多轮对话



服务器端接收消息信息如图



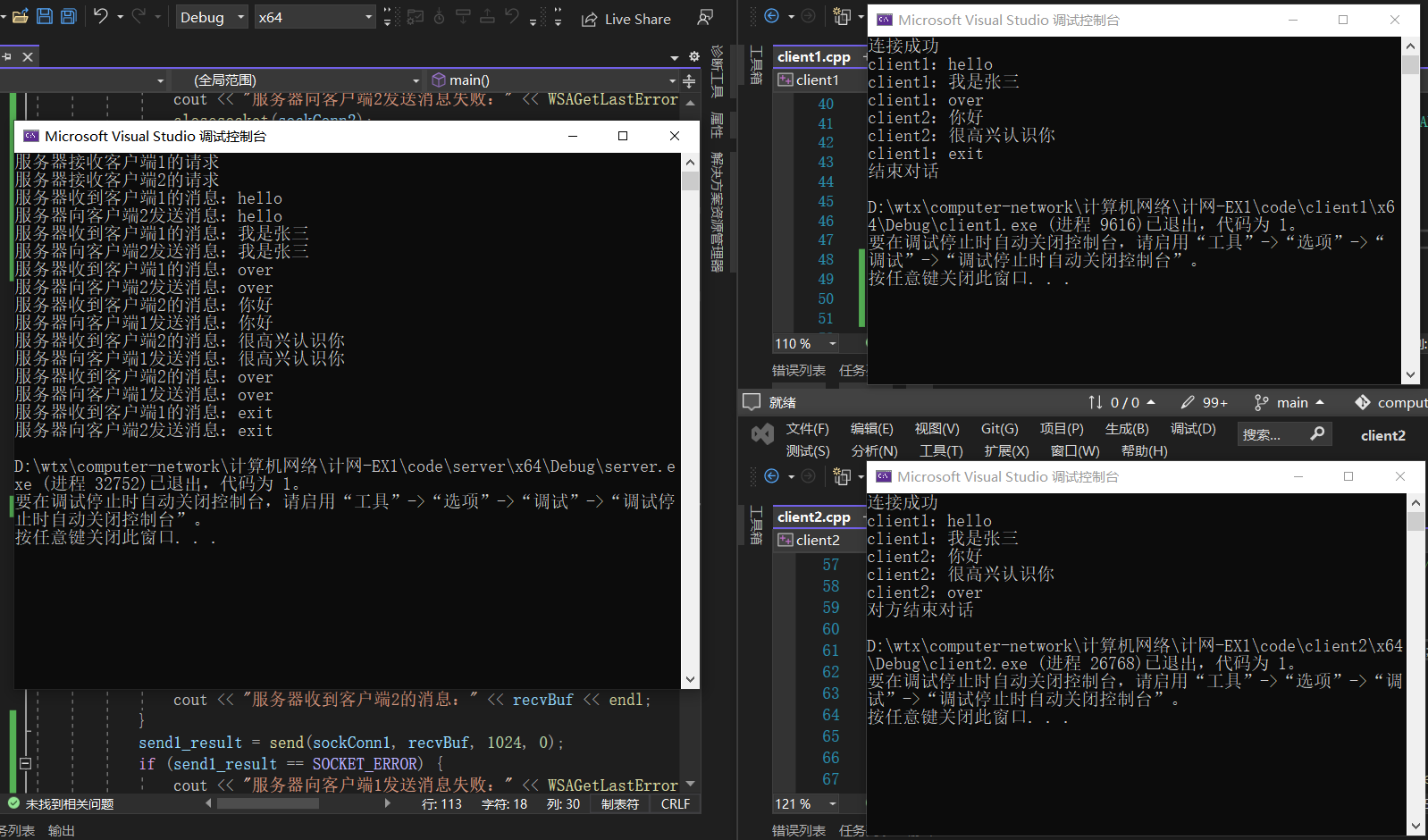
客户端1收发消息代码如图，此时默认客户端1先发消息，客户端2后发消息，客户端2相关代码即将client1和client2关键词调换。并且将接收、发送代码顺序调换。

运行后发现遇到问题：客户端1输出over后客户端2不能进行输出

原因：char型数组内容清空时代码



使用memset函数更改后成功，此时运行结果如下图



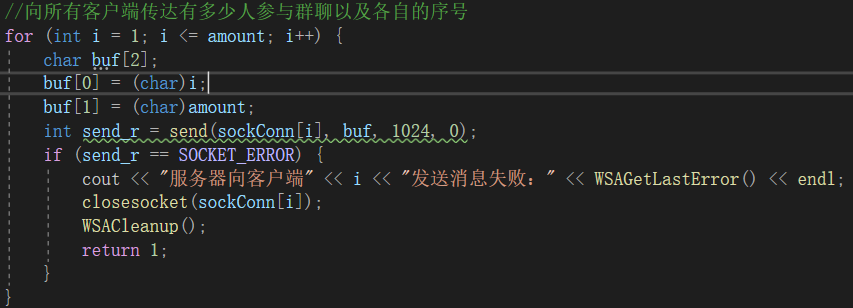
## 增加功能：群聊

按照连接服务端的顺序进行标记id，根据id以自小到大的顺序依次发言。

### 服务端代码修改



以数组形式存储与客户端连接产生的套接字



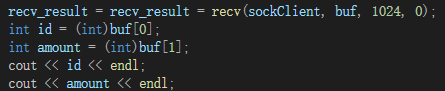
序号依据连接服务端顺序产生



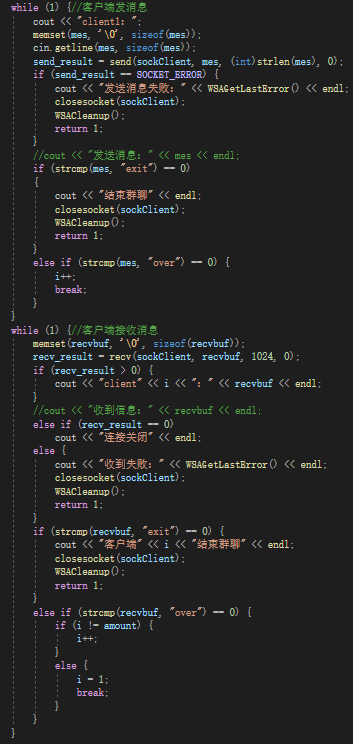
不再有从具体某个客户端接收消息以及向具体某个客户端发送消息。服务端从任意一个客户端接收到消息，转发给其余所有客户端。

### 客户端代码修改

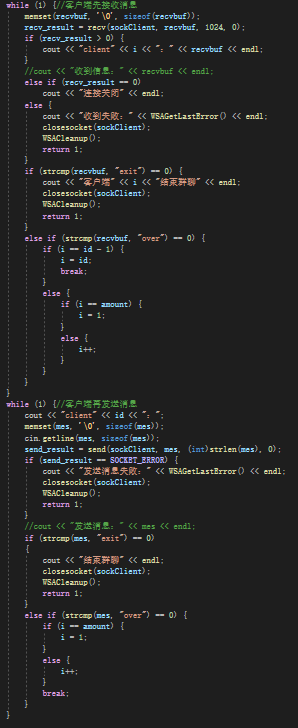
客户端实际上不区分客户端1、2、3，即可以client3程序先连接服务端，被认作是客户端1。



连接后由服务端对于客户端进行id赋值，并告知连接的总客户端数



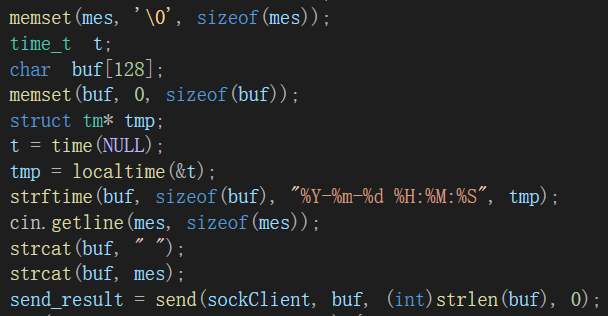
客户端1（即最先连接的客户端）先发送消息，再接收消息。所以当它发送“over”消息后，跳出发送消息的循环，进入到接收消息的循环中。每接收到一次“over”，发送消息的客户端id为i+1，直到发送消息的客户端id为amount时，下一次发送消息的客户端为客户端1，进入发送消息的循环。



其余客户端，先接收消息，轮到自己时发送消息。所以当它收到“over”消息时，需要进行判断，如果下一个发消息的客户端是自己，那么跳出接收消息的循环，进入到发送消息的循环。

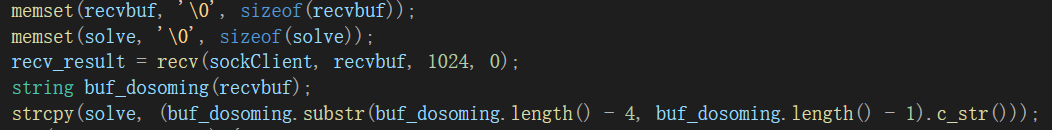
## 带时间发送消息

### 客户端发送信息代码



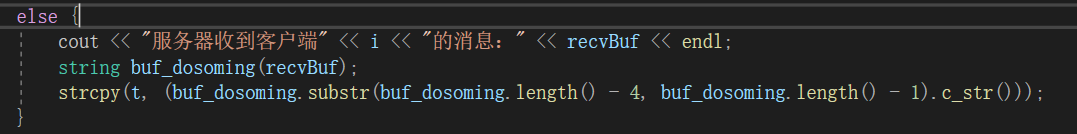
发送到服务端的消息格式为：时间消息+“ ”+发送消息内容

### 客户端接收信息代码



处理得到接收消息的最后四个字符组成的字符串solve，用solve判断是否到下一个客户端进行发送或者是否结束群聊。

### 服务端接收消息代码



处理得到接收消息的最后四个字符组成的字符串solve，用solve判断是否到下一个客户端进行发送或者是否关闭套接字。

## 运行结果

