**计算机网络实验**

**实验1：利用Socket，设计和编写一个聊天程序**

**姓名： 蒋浩南 学号: 2012948**

# 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。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。

# 二、协议设计

## （一）报文格式



如上图所示，协议所发送的报文格式包含了发送端信息和接收端信息，

报文还包括记录报文时间的时间戳和要发送的内容。

## （二）语义和语法

### 1、语法

（1）发送端存有发送端客户端的名称，而接收端存有接收端客户端的名称。为char数组。

（2）时间戳为记录报文生成的时间。

（3）内容为发送的消息，或者是供接收端处理的命令类消息。

### 2、语义

（1）当服务器端接受一个客户端的连接时，客户端会向服务器发送内容为”init”的报文，服务器端，会识别并完成该客户端名称与对应连接socket的键值对的初始化。

（2）当客户端向服务器发送内容为”exit”的报文，服务器端，会识别该客户端下线。

## （三）时序



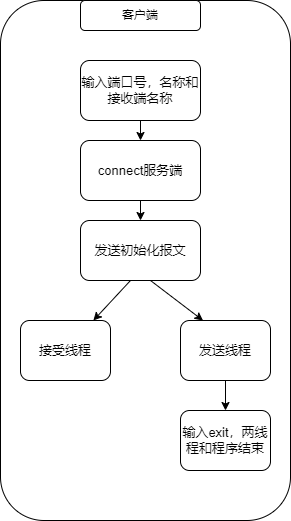
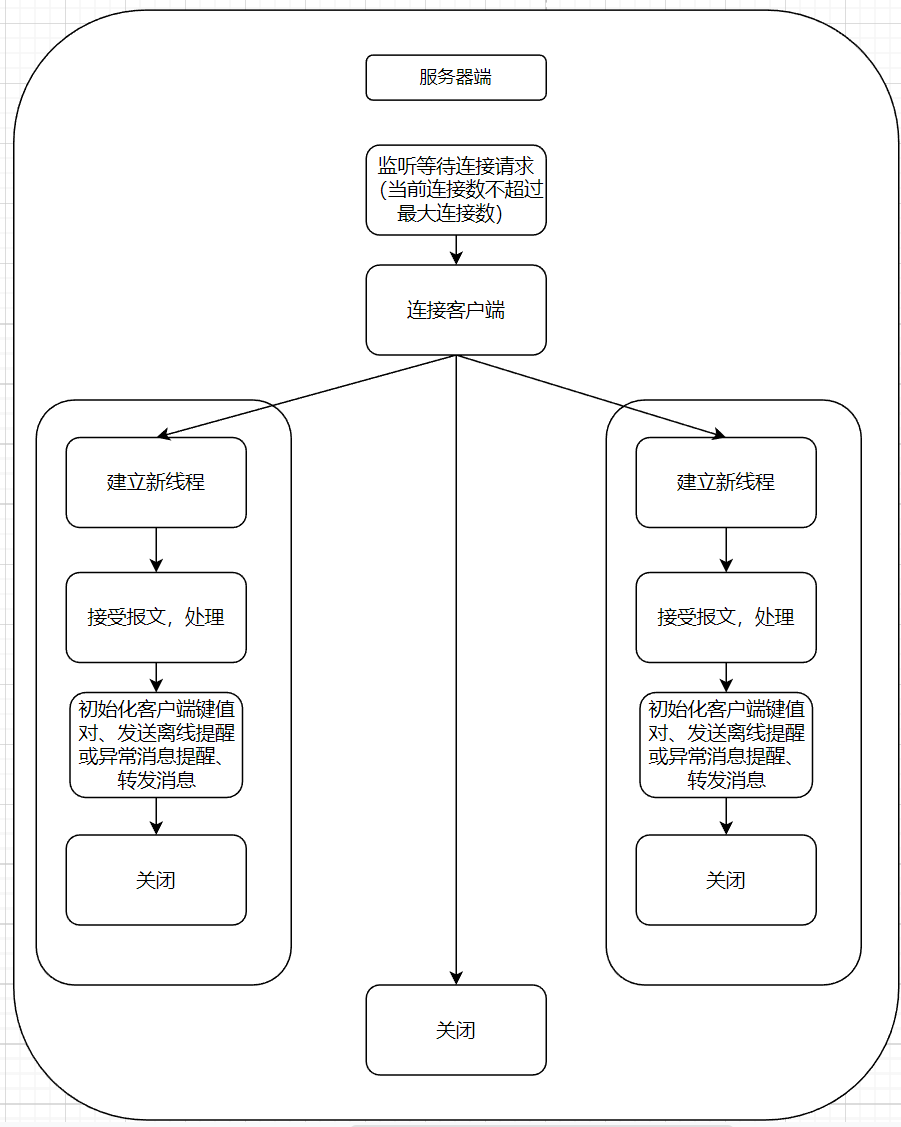
1. 服务器端启动后会进入监听，在连接数不超过最大连接数的时候，等待客户端的连接。
2. 当有一个客户端连接上客户端后，服务器端会新开一个线程建立与客户端的socket连接。而客户端在与服务器端建联成功后会发送一个注册报文，在服务器端将客户端名称和于其连接的socket建立键值对。同时客户端会开启两个线程，一个用来发送报文，一个用来接收报文。
3. 当客户端发送消息，服务器端接收消息，并根据报文中指定的接收端在存储的键值对中寻找接收端，如果没能找到，会向客户端发送消息，显示“对方未上线或不存在用户”；

如果找到，服务器端会转发接收到的消息。

1. 客户端发送“exit”报文，客户端的发送和接收报文线程结束，服务关闭。

服务器端接收，并向接收端发送“对方已下线”的消息。

步骤图示如下：

# 三、各模块功能

## （一）服务器端

### 1、监听接受客户端连接

1. **while** (1) {
2. **if** (OnConnectNum < maxClientNum) {
3. SOCKET\* clientSocket = **new** SOCKET;
4. \*clientSocket = accept(serverSocket, (SOCKADDR\*)&addrC[OnConnectNum], &len);
5. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
6. cout << endl;
7. cout << "accept Client  " << inet\_ntoa(addrC[OnConnectNum].sin\_addr) << "   " << ntohs(addrC[OnConnectNum].sin\_port) << endl;
8. cout << endl;
9. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
10. cout << endl;
11. OnConnectNum++;
12. **if** (OnConnectNum == maxClientNum)
13. {
14. cout << "达到最多连接数量" << endl;
15. }
16. **HANDLE** hThread;
17. hThread = CreateThread(NULL, NULL, &SeverThread, (**LPVOID**)clientSocket, 0, NULL);
18. CloseHandle(hThread);
19. }
20. }

当当前连接的客户端数不超过最大的时候，服务器会接受客户端的连接，并建立新的线程与该客户端进行socket连接。

### 2、服务器端服务线程

1. //服务线程
2. **DWORD** WINAPI SeverThread(**LPVOID** lpParameter)
3. {
4. //新建一个SOCKET用于通信
5. SOCKET\* ClientSocket = (SOCKET\*)lpParameter;
6. **bool** init = 1;
7. //缓冲区
8. **char** RecvBuf[MaxBufSize];
9. **char** SendBuf[MaxBufSize];
11. //开始接收与转发
12. **while** (1)
13. {
14. memset(RecvBuf, 0, 1024);
15. **if** (recv(\*ClientSocket, RecvBuf, **sizeof**(RecvBuf), 0) > 0)
16. {
17. myMSG\* receMSG=(myMSG\*)RecvBuf;
18. //与客户端初始化信息
19. **if** (strcmp(receMSG->message, "init") == 0&&init==1) {
20. memcpy(ClientMap[OnConnectNum - 1].name, receMSG->name, 10);
21. ClientMap[OnConnectNum - 1].sockClient= ClientSocket;
23. cout << endl;
24. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
25. cout << endl;
26. cout << "用户： " << receMSG->name << " 完成初始化";
27. cout << endl;
28. cout << "from:            " << receMSG->name << endl
29. << "to:              " << receMSG->recname << endl
30. << "time:            " << receMSG->time << endl
31. << "receive message: " << receMSG->message << endl;
32. cout << endl;
33. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
34. cout << endl;
35. init = 0;
36. **continue**;
38. }
40. //非初始化，即转发消息
41. **if** (init != 1) {
43. SOCKET\* ToSOCKET=NULL;
44. **bool** find = 0;
46. //在map中寻找聊天对象
47. **for** (**int** i = 0; i < maxClientNum; i++) {
48. **if** (strcmp(receMSG->recname, ClientMap[i].name) == 0) {
49. ToSOCKET = ClientMap[i].sockClient;
50. find = 1;
51. **break**;
52. }
53. }
55. //聊天用户在线
56. **if** (find == 1) {
57. //客户端退出
58. **if** (strcmp(receMSG->message, "exit") == 0) {
60. cout << endl;
61. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
62. cout << endl;
63. cout << "from:            " << receMSG->name << endl
64. << "time:            " << receMSG->time << endl
65. << "客户端退出" << endl;
66. //<< "receive message: " << receMSG->message << endl;
67. cout << endl;
68. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
69. cout << endl;
71. //向聊天对象发送对方退出的信息
72. memset(SendBuf, 0, 1024);
73. memcpy(SendBuf, RecvBuf, 1024);
74. memcpy(SendBuf+84, "对方已退出", **sizeof**("对方已退出"));
75. **int** k = 0;
76. k = send(\*ToSOCKET, SendBuf, **sizeof**(SendBuf), 0);
77. //关闭与当前用户的socket
78. OnConnectNum--;
79. closesocket(\*ClientSocket);
80. **return** 0;
81. }
83. //复制要转发的消息
84. memset(SendBuf, 0, 1024);
85. memcpy(SendBuf, RecvBuf, 1024);
86. **int** k = 0;
87. k = send(\*ToSOCKET, SendBuf, **sizeof**(SendBuf), 0);
88. cout << endl;
89. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
90. cout << endl;
91. cout << "from:            " << receMSG->name << endl
92. << "to:              " << receMSG->recname << endl
93. << "time:            " << receMSG->time << endl
94. << "receive message: " << receMSG->message << endl;
95. cout << endl;
96. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
97. cout << endl;
98. }
99. //聊天用户不在线
100. **else** {
102. memset(SendBuf, 0, 1024);
104. **char** name[10];
105. **char** recname[10];
106. memcpy(name, RecvBuf, 10);
107. memcpy(recname, RecvBuf+10, 10);
108. memcpy(SendBuf, "server", **sizeof**("server"));
109. memcpy(SendBuf+10, name, 10);
110. **time\_t** t = time(0);
111. strftime(SendBuf+20, 64, "%Y/%m/%d %X %A ", localtime(&t));
112. memcpy(SendBuf+84, "对方未上线或不存在该用户",**sizeof**("对方未上线或不存在该用户") );
114. **int** k = 0;
115. k = send(\*ClientSocket, SendBuf, **sizeof**(SendBuf), 0);
117. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
118. cout << endl;
119. cout << "发送信息方：" << name << endl;
120. cout <<"接收信息方："<< recname << "  不存在用户或未上线" << endl;
121. cout << endl;
122. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
123. cout << endl;
124. }
125. }
126. }
127. }
128. }

1、线程会建立与单个客户端通信的socket。和接受数据和发送数据的缓冲区。

2、进入循环，当接收到客户端的消息：

如果为内容为‘init’,则为初始化消息，初始化客户端名称与对应socket的键值对。

如果不是初始化消息，根据报文中的接收方在键值对中寻找对应客户端：

如果没找到，则像发送端发送消息“对方未上线或不存在该用户”；

如果找到则向接收端转发消息。

3、当客户端退出时，会向服务器发送内容为‘exit’的报文，服务器会像接收端发送“对方已退出”消息。

## （二）客户端

### 1、接收线程

1. **void** recMsg() {
3. //缓冲区
4. **char** RecvBuf[MaxBufSize];
5. memset(RecvBuf, 0, MaxBufSize);
6. **while** (recflag)
7. {
8. **int** n;
9. memset(RecvBuf, 0, MaxBufSize);
10. **if** (recv(sockClient, RecvBuf, MaxBufSize, 0) > 0) {

13. myMSG\* receMSG = (myMSG\*)RecvBuf;
14. cout << endl;
15. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
16. cout << endl;
17. cout << "name: " << receMSG->name << endl
18. << "time :" << receMSG->time << endl
19. << "receive message :" << receMSG->message << endl;
20. cout << endl;
21. cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;
22. cout << endl;


26. cout << "请输入发送信息:" << endl;
27. }
28. **else**
29. **break**;
30. }
31. }

接受输入，将发送端、接收端、时间戳和消息写入缓冲区，像服务器发送。如果输入为exit，则关闭该线程和发送线程，程序关闭。

### 2、发送线程

1. **void** sendMsg() {
3. //缓冲区
4. **char** SendBuf[MaxBufSize];
5. memset(SendBuf, 0, MaxBufSize);
6. **while** (sendflag)
7. {
9. memset(SendBuf, 0, MaxBufSize);
10. cout << "请输入发送信息:" << endl;
11. myMSG\* sendMSG = (myMSG\*)SendBuf;
13. memcpy(sendMSG->name, name, 10);
14. memcpy(sendMSG->recname, recname, 10);
16. **time\_t** t = time(0);
17. strftime(sendMSG->time, 64, "%Y/%m/%d %X %A ", localtime(&t));

20. scanf\_s("%s", sendMSG->message, MaxBufSize - 10-10 - 64);
22. //发送数据
23. send(sockClient, SendBuf, MaxBufSize, 0);
24. **if** (strcmp(sendMSG->message, "exit") == 0) {
25. sendflag = 0;
26. recflag = 0;
27. **break**;
28. }
29. }
30. }

接受来自服务器的消息，写入缓存，将缓存转为报文格式，打印显示。

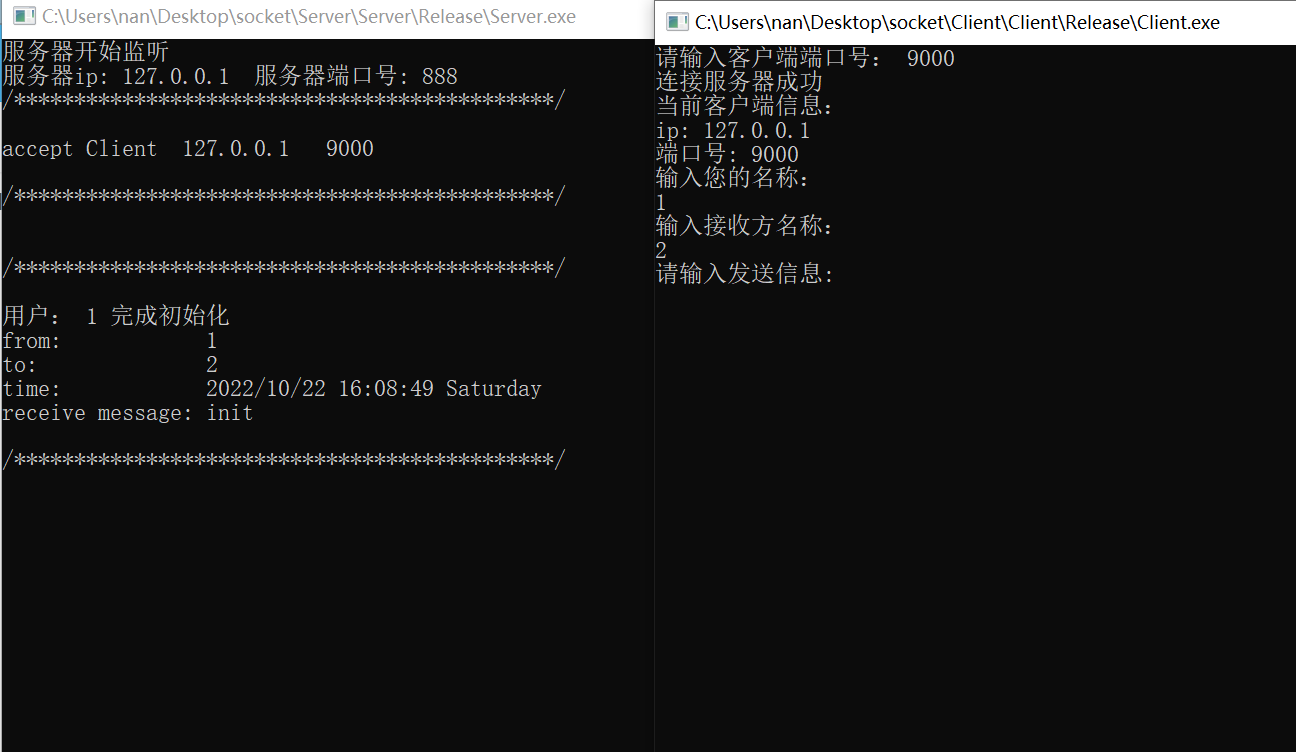
# 四、程序展示

1、服务器ip和端口写死。

客户端需输入端口号、本客户端名称和接收方客户端名称。

与服务器连接成功后客户端会发送一个初始化报文。

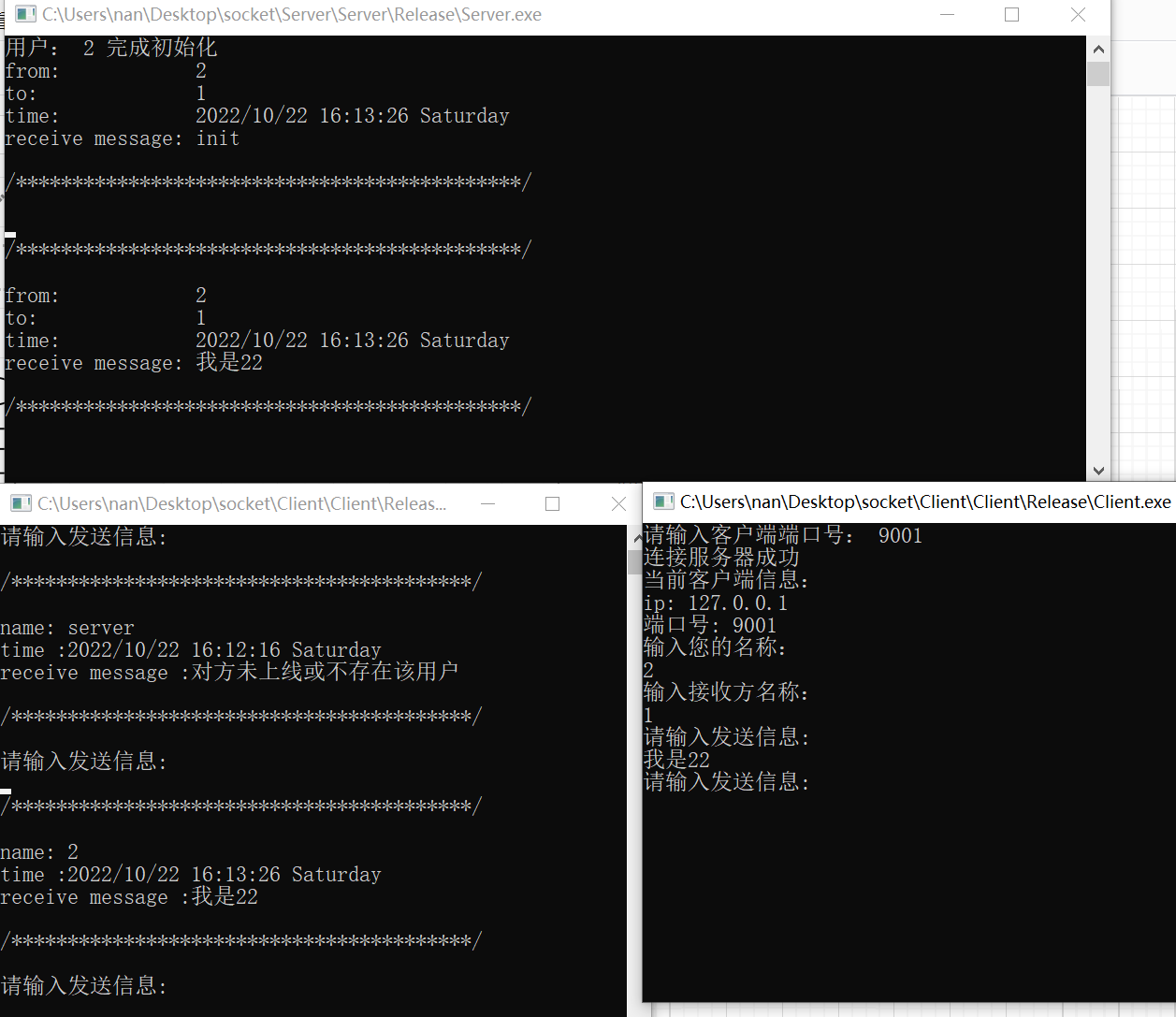
如图，服务器完成连接和客户端的初始化。



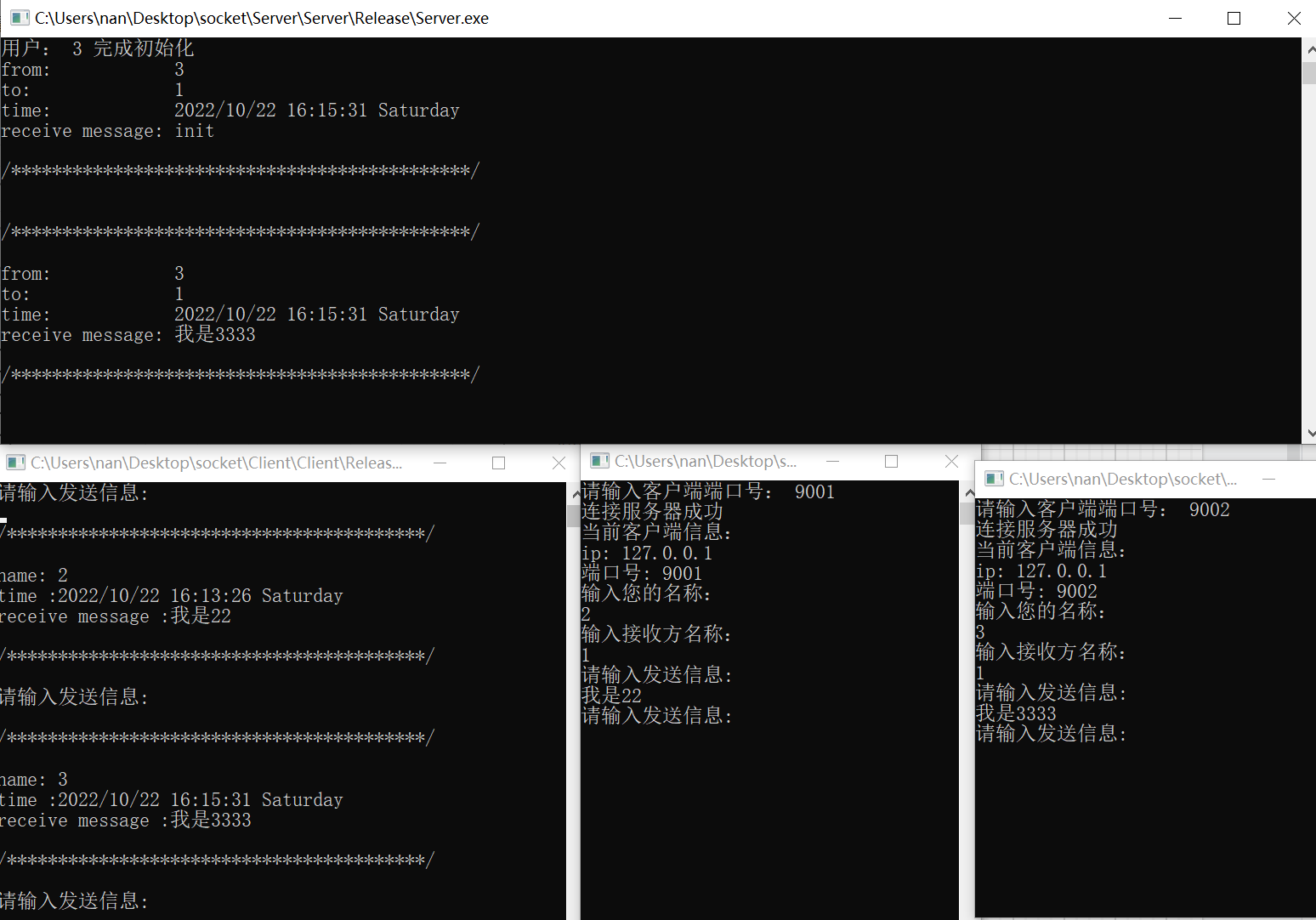
2、此时客户端发送消息，由于此时接收方未上线，客户端会受到来自服务器的消息。服务器也会显示。



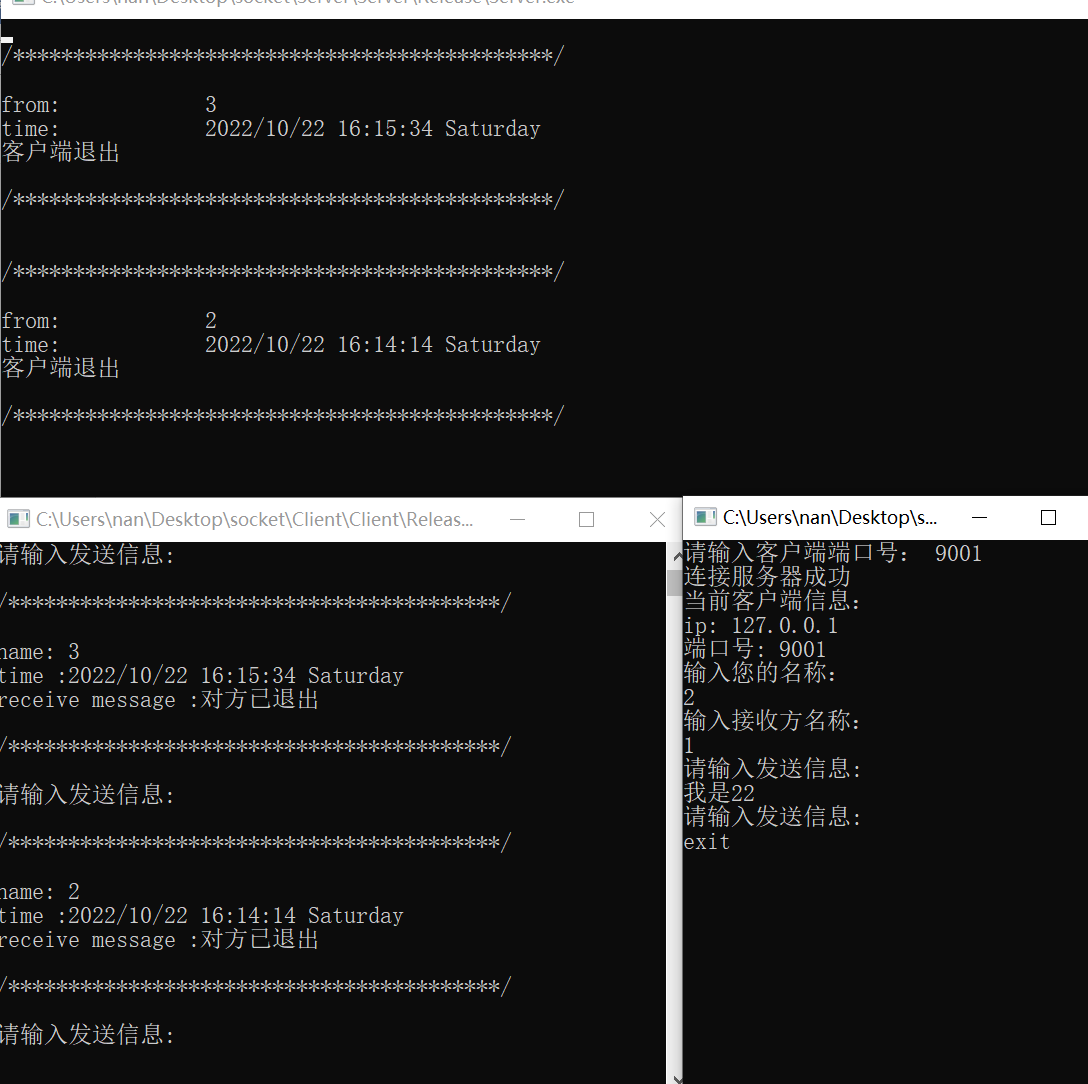
3、当接收端上线，就可以正常发送消息，服务器也会有转发消息的记录。



4、再连接一个客户端3，他的接受端也是1，也可以正常发送消息。



5、当客户端输入exit，会向服务器发送结束报文，服务器会向接收端发送报文，消息为“对方已退出”。同时服务器会有记录。



# 五、实验遇到的问题

1、在建立缓冲区的时候，使用new分配堆上内存在进行结构转换会出错，所以使用分配在栈上的连续内存。

2、在设计报文时要根据要实现的功能，并考虑未来的拓展性。