# Socket套接字编程

|  |  |  |
| --- | --- | --- |
| **实验课程**：计算机网络 | **年级**：2023 | **实验成绩**： |
| **实验名称**：Socket Programming | **姓名**：张建夫 |  |
| **实验编号**： | **学号**：10235101477 | **实验日期**：2024/12/30 |
| **指导教师**：章玥 | **组号**： | **实验时间**：8：00~9：30 |

**一、实验目的**

1.掌握Socket编程接口；掌握基于Socket编程接口编写网络通信程序的方法和典型程序构成；

2.根据指定的通信需求，使用Socket编程接口编写网络通信程序的服务器端程序和客户端程序。

**二、实验内容与实验步骤**

实验需求分析：

\* Server:

1.在标准输出上打印客户端发送的文本消息，但不应该打印任何其他消息，如调试信息。

2.服务器应该能够接收来自多个客户端的文本消息。

3.当多个客户端同时尝试向服务器发送文本信息时，服务器应该一次只打印一条（顺序不限）。

4.服务器应该把要监听的端口作为它的第一个参数。

\* Client:

1.客户端应该从标准输入中接收文本信息。

2.信息的结束由一个控制序列来标记，该序列对应于按两次ENTER（这也是客户端退出的指令）。

3.如果你的客户端通过管道从文件中读取文本消息，并在没有看到控制序列的情况下到达EOF，那么客户端仍然应该传输消息并退出。

4.客户端应该把服务器运行的主机名称和服务器监听的端口作为第一个参数。

\* 错误处理：

1.如果服务器不能绑定到你指定的端口，应该打印出一条标准错误信息，程序应该退出。

2.你不应该假设你的服务器将运行在一个特定的IP地址上，或者客户端将来自一个预先确定的IP地址。

3.客户端和服务器都应该产生一个适当的错误信息，并在给出无效参数时终止。

\* 测试要求：

1.实现长文件的传输（至少20KB）。

2.至少五个客户端同时运行。

**三、实验环境**

1.操作系统：Linux ubuntu22.04

2.内核版本：6.8.0-49-generic

3.架构：x86\_64

4.使用语言：python

5.python版本号：3.10.12

**四、实验设计与实现：**

实验设计所用的类：

本实验采用python语言的socket库和threading库进行实现，使用的主要类有：

1.socket：用于实现服务器端和客户端的套接字，套接字是两台机器间通信的端点。

2.Thread：线程类，实现多线程编程，提高服务器处理效率。

3.lock：锁类，用于多线程之间的同步。

实验亮点：

1.实现了要求功能：大文件传输/5个客户端同时运行。

2.可以同时满足**多个客户端**（理论上最大数量为100，而不仅仅是5个）。

3.实现了**多线程编程**，服务器创建多个线程来服务不同客户端。

4.实现了客户端与服务器之间的**双工通信**, 而且服务器可以指定向哪些客户端发送消息。

5.实现了错误处理，在不能建立连接时发送超时错误。

6.实现了文件传输**缓冲**机制，按页（1KB）输出。

使用方法(linux环境下)：

Client：在client.py文件所在目录中输入命令python3 client.py <ip address> <port>启动。

1.传输短文本：根据启动后的提示直接在标准输入 输入你想传给服务器端的内容即可

2.传输文件：需要使用管道，具体用法为cat <file> | python3 client.py <ip address> <port>.

3.退出方式：按两次回车即可

Server：在server.py文件所在目录中输入命令python3 server.py <port>启动。

1.向指定客户端传输消息：根据提示在标准输入中输入：<用户编号> <你的消息>

(用户编号会在用户连接成功时告知服务端，所有用户编号为正整数)

(用户编号0表示广播，向所有客户端发送消息)

2.退出方式：Ctrl+c

设计的大致思路：

Server端：

基础思路是实现多线程处理多用户请求，然后单独建立一个线程负责向用户发送消息，文件传输使用管道。

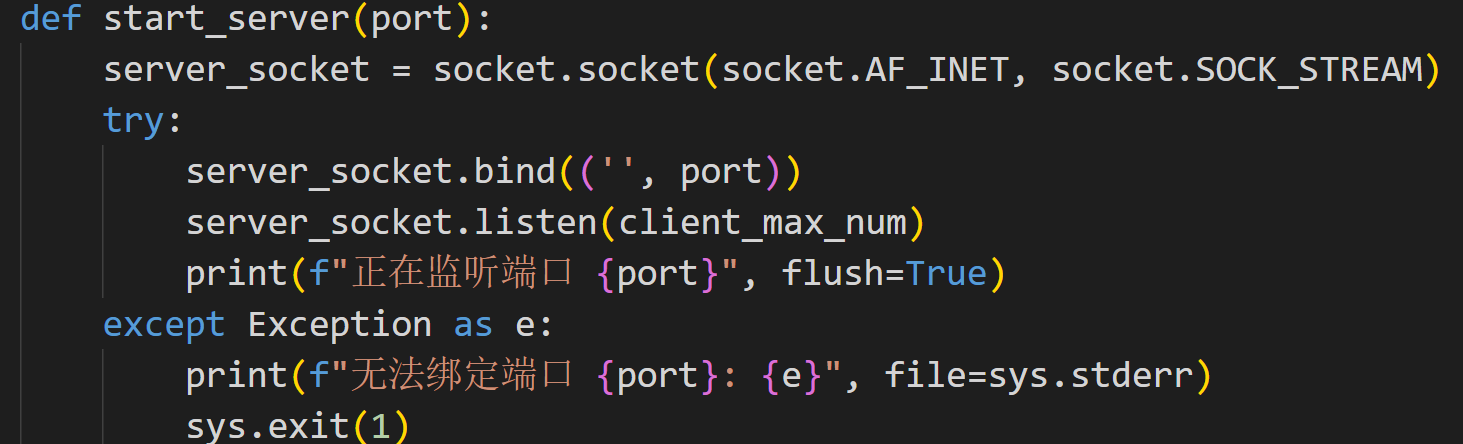
Client端：

基础思路是通过区分用户是管道传输（文件传输）还是标准输入分别进行处理，然后在初始化的时候创建一个额外的的线程接收服务器端得到的消息。

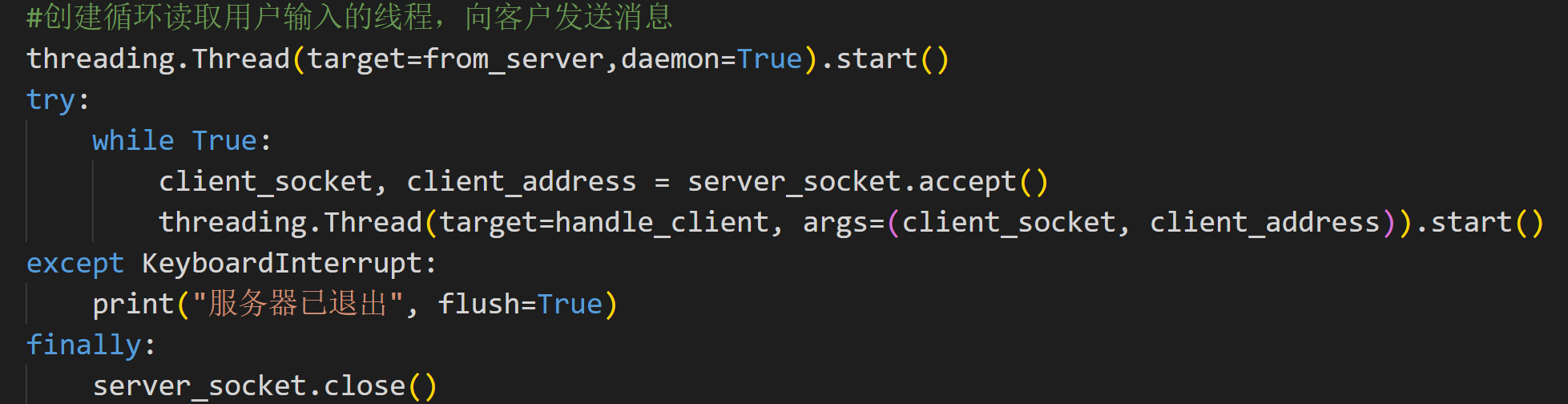
设计与实现的具体思路：

Server端：

首先定义函数start\_server，该函数初始化server的socket ，绑定用户传入的端口并监听，如果无法绑定端口则退出：

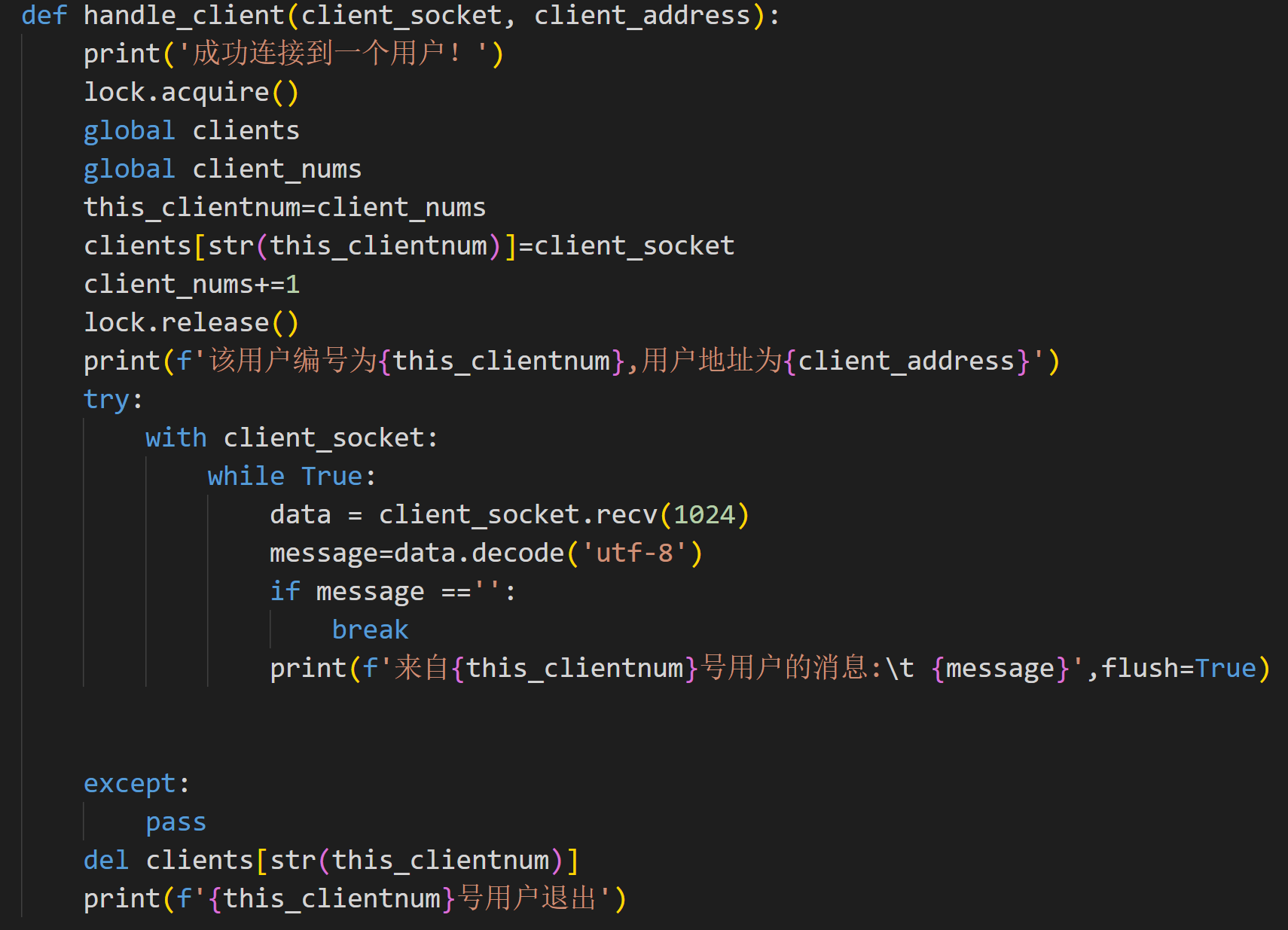


接着该函数开启守护线程from\_server（负责传输消息给客户端，并在主线程退出时一并退出），然后为每个client创建一个专属的线程函数handle\_client：



当按下Ctrl+c时，服务器退出并关闭套接字

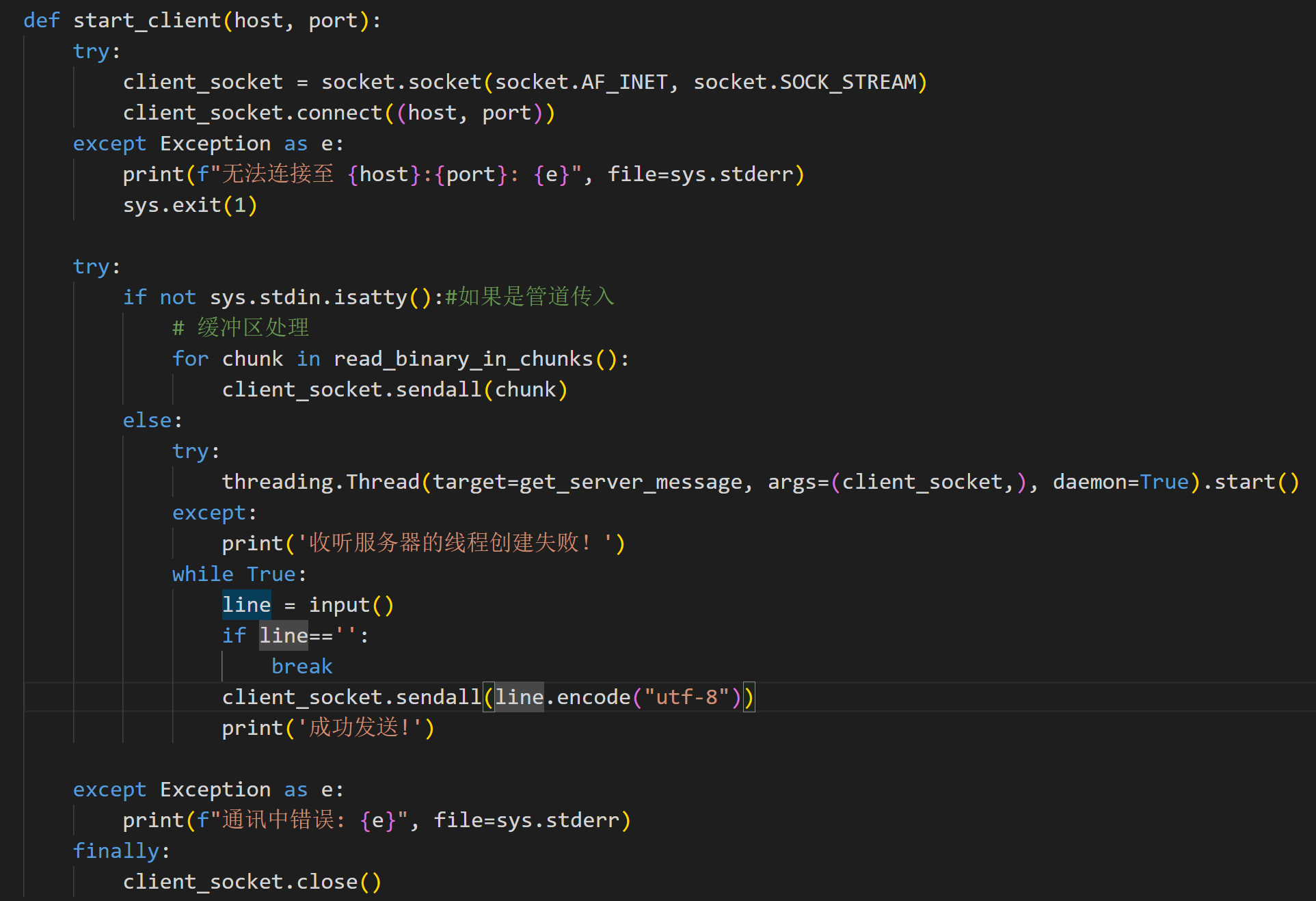
handle\_client：该函数是处理用户信息的主函数



先将client加入一个字典方便服务器端管理（此处需要进行同步处理，防止多个客户端同时连接时出错），之后重复接收该用户的消息直到用户退出。

Client端：

首先定义stay\_client函数用于处理客户端输入并将其发送给服务器端，同时负责接收服务器端发来的信息（该功能会创建另外一个线程处理）：



该函数首先试图连接服务器ip和端口，如果连接失败，告知用户。

如果用户选择文件传输，使用管道，其中的read\_binary\_in\_chunks函数负责分页（默认1KB）读取文件内容，sendall方法负责发送文件。

如果不是文件传输，则重复接收用户的消息，两次回车退出客户端。

**五、实验测试：**

在实验课上已经实现了本机的双工通信和两个机器间的双工通信。满足了实验的所有要求。（下面的截图是按照评分标准进行截图的，会提示截图对应哪些评分标准）

评分标准：

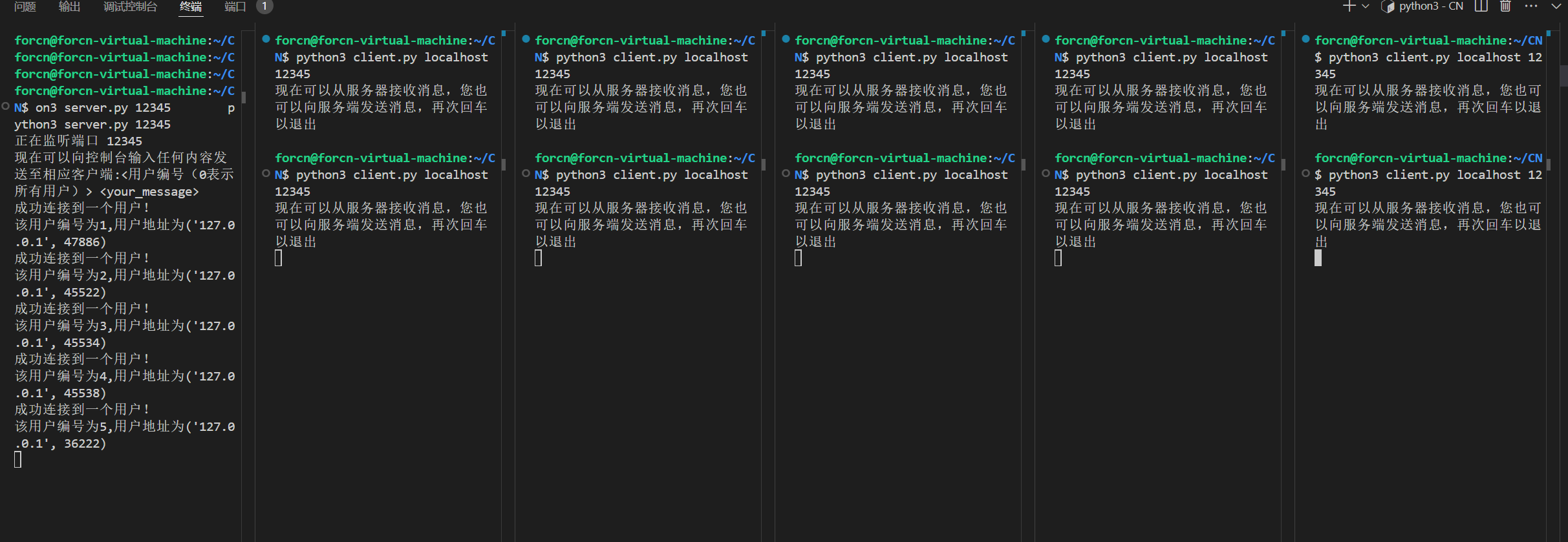
Server端：

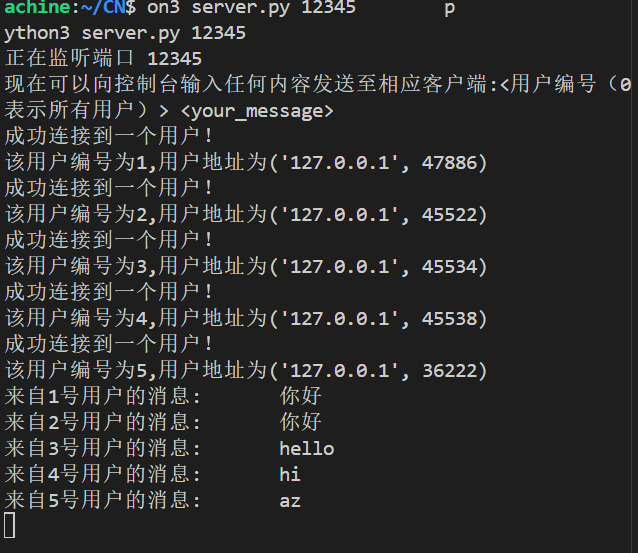
能在标准输出打印客户端发送的消息

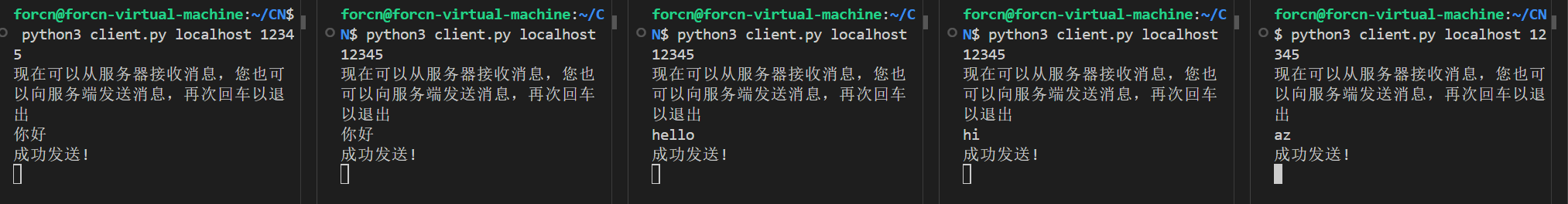
支持5个以上客户端同时发送消息并逐一打印（课上已验证）

Client端：

能从标准输入或文件接收消息







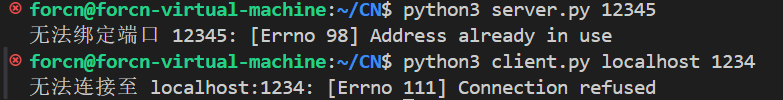
评分标准：

Server端：

绑定至错误的端口号时能提示出错信息

Client端：

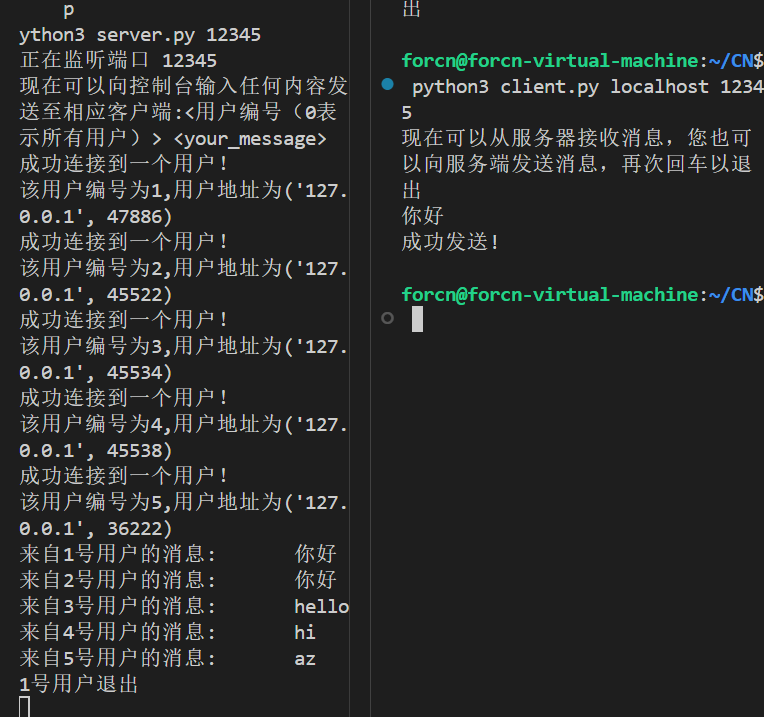
连接至错误的IP地址/端口号时能提示出错信息



评分标准：

Client端：

标准输入消息以两次回车作为结束标志



可以看到在两次回车后（“你好”发送完一次回车，再一次回车，总共两次）1号用户退出

评分标准：

整体：

支持长文本消息 （不少于 20KB），有缓冲区管理

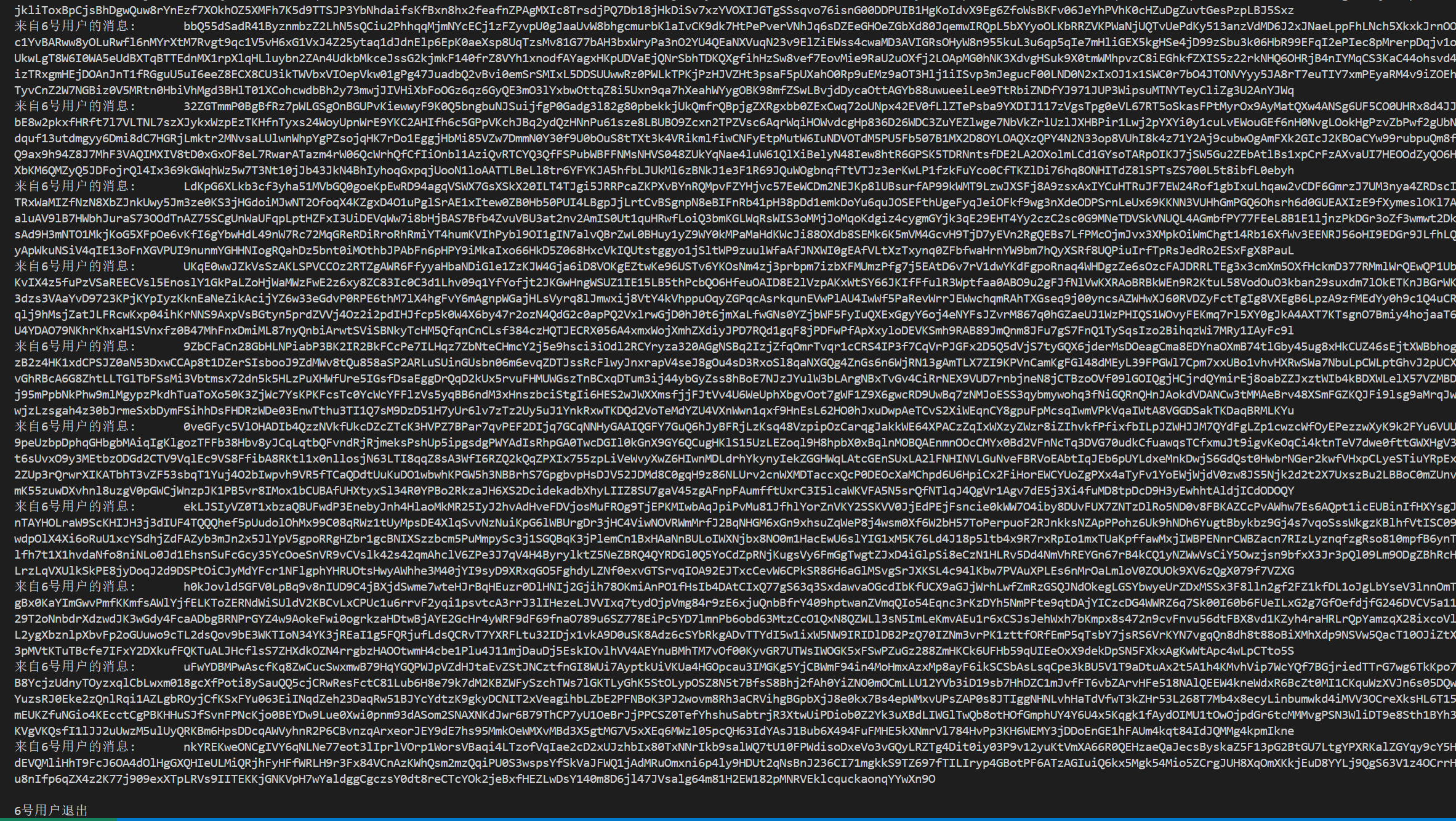
支持在localhost及在两台不同机器上运行

容错性好，无闪退

传输文件（不同机器上传输）：



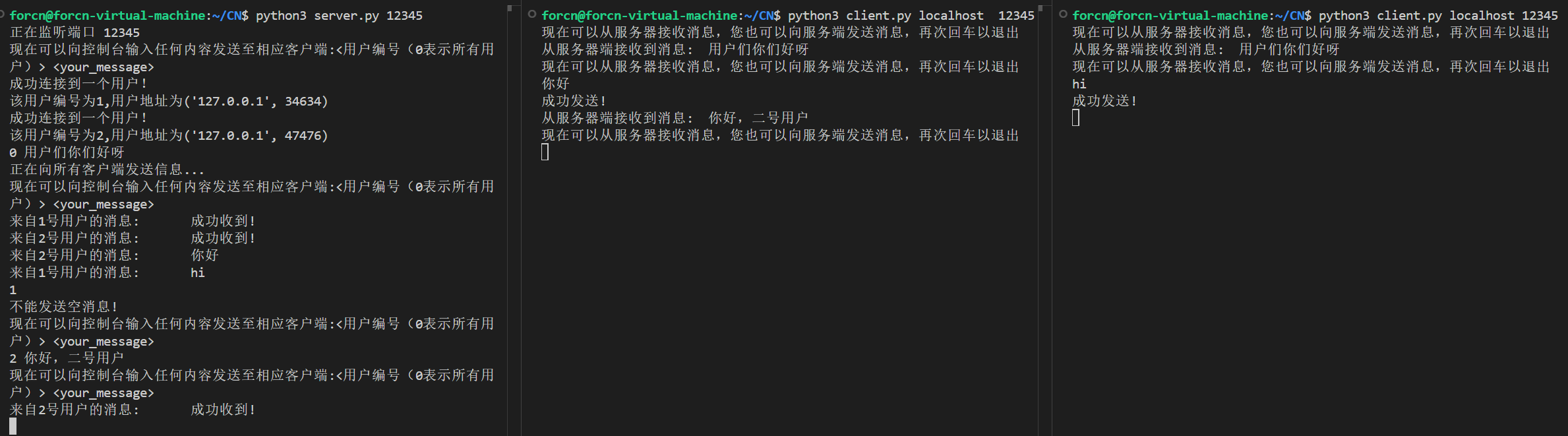
此处ip地址为我另外一个虚拟机的ip地址



此处选取的文件是课程测试文件test\_5里面的iput\_2.txt(4MB大小，缓冲区为1KB)

评分标准：

双工通信



在上图中，服务器端先向所有用户发送：“用户们你们好呀”（用户编号0表示所有用户）

可以看到所有用户都接收到了消息

接下来，服务器端单独向2号用户发送消息：“你好，二号用户”

可以看到二号用户成功接收到了消息（中间的为2号用户）

成功实现了双工通信，并且指定用户编号通信

**六、实验结果总结**

这是本学期的最后一次实验，也是我们的第一个编程实验。本次实验将理论更加落实于实践，让我们学习了理论原理之后能够进一步认识到Socket编程和信息传递的具体步骤。

**七、附录**

实现代码：见我的github仓库（<https://github.com/saydontgo/computer-network-proj>）