# 聊天系统设计

**王钰翔 1200012968**

## 系统设计

本系统实现了基本的聊天系统功能，允许多组用户同时进行聊天。服务器端运行聊天协调器程序，使用 UDP 协议与聊天客户端通信，且客户端之间可以建立会话，通过 TCP 协议彼此通信。

本系统使用python3.4开发，可运行于安装了python3.4的Linux和windows环境中。项目有以下两个文件：server.py，client.py，分别对应服务端及客户端。

## 系统架构

聊天流程如下：

 具体的功能下文介绍。

### 聊天协调器（服务器端）

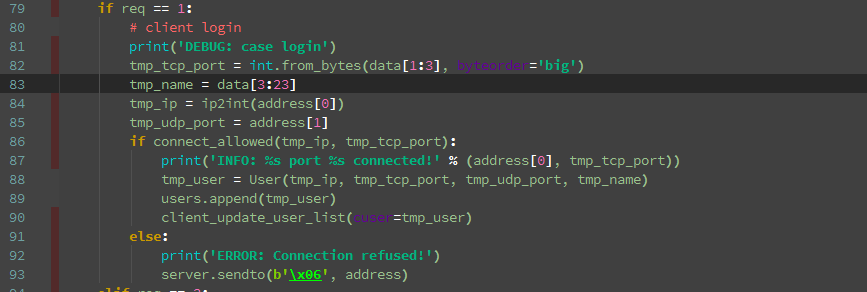
服务器端实现了以下功能：

1. 客户端登录

客户端发起登录请求。服务器端返回登录确认信息。

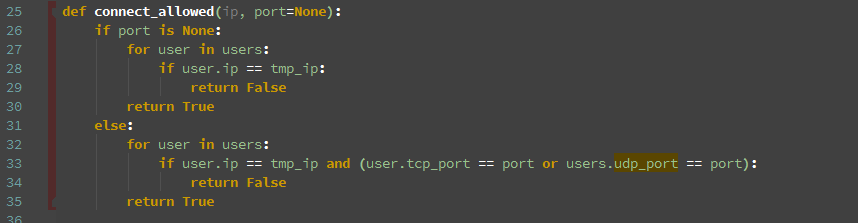
若登录成功，则更新用户列表。并更新所有用户的用户列表信息

代码如下：



其中，connect\_allowed方法用以确认用户是否满足登录条件。为方便本机调试，本系统允许同一IP不同端口登录。

connect\_allowed实现逻辑如下：



client\_update\_user\_list方法用以更新所有客户端所维护的用户列表。由于当处理某一个登录请求时，若登录成功应返回0x02，而服务器端主动更新列表则返回0x05，传参tmp\_user用以区分这两种情况。

该函数实现如下：

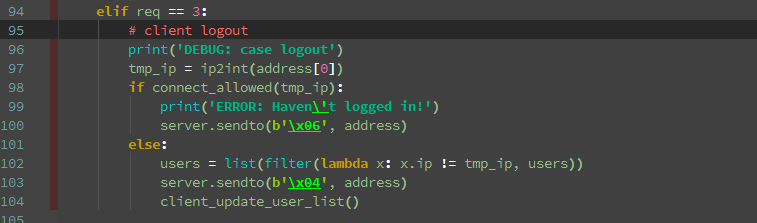


1. 客户端登出

客户端发起登出请求，服务器端接收并返回确认信息。

登出后，维护当前用户列表，并更新所有用户的用户列表。

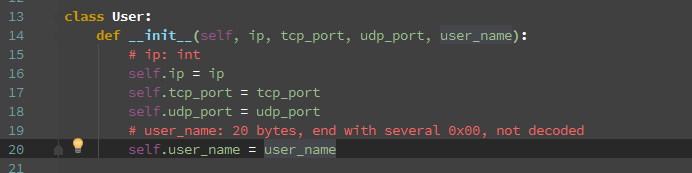
处理逻辑如下：



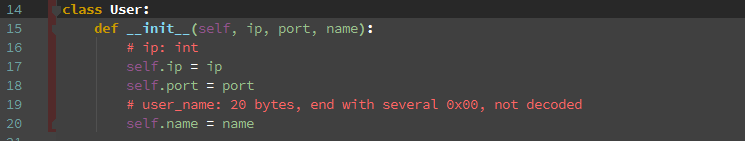
其中，调用connect\_allowed用以确认当前用户是否已经登录。若已登录则满足登出条件。此时直接返回登出确认信息，并更新所有客户端用户列表。

1. 维护当前登录用户列表，并实时更新各户端

在服务器端定义了users用户列表，其中保存User对象，User类定义如下：



在客户端同样也维护了一个users用户列表，其中保存了User对象，但User类定义不同于服务器端。



其中，客户端不需要保存用户的UDP PORT，因为客户端不需要彼此之间进行UDP通信。

主要在两种情况下维护：

1. 登入

对于符合条件的登入，新建User对象，并添加到users对象列表中。

返回登录确认信息，并更新所有各户端。

1. 登出

客户端登出之后，在users对象列表中移除对应的对象。

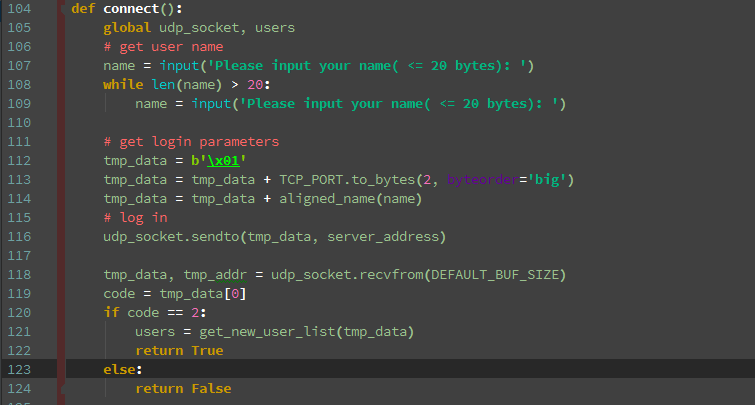
返回登出确认信息，并更新所有客户端。

### 聊天客户端（客户机端）：

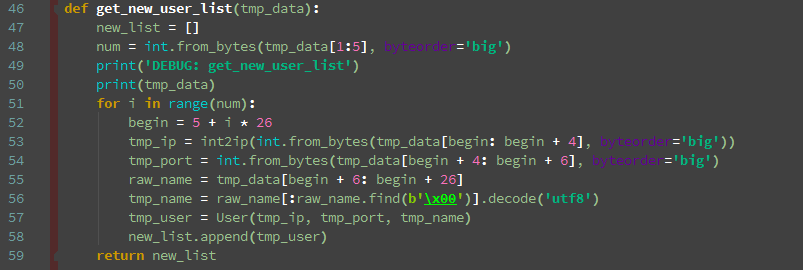
聊天客户端处理流程如下：

1. 提示输入端口及用户名。
2. 创建udp\_socket，用以和服务器之间的UDP通信。
3. 登录并接收登录确认信息。根据登录确认信息初始化users用户列表。

其中登录部分逻辑如下：

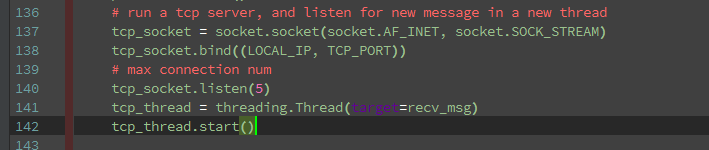


其中get\_new\_user\_list方法根据接收到的数据，更新user列表，其操作如下：

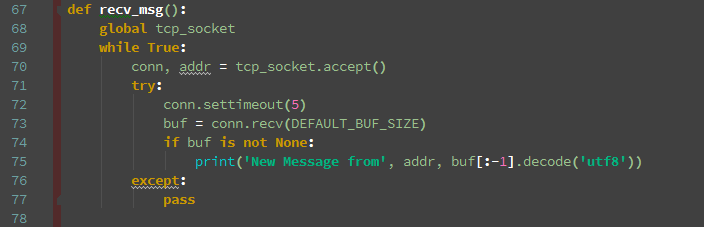


1. 启动TCP Server，新建tcp\_socket用以建立tcp连接。新建tcp\_thread线程，用以接收聊天消息。

新建线程：



其中recv\_msg定义如下：

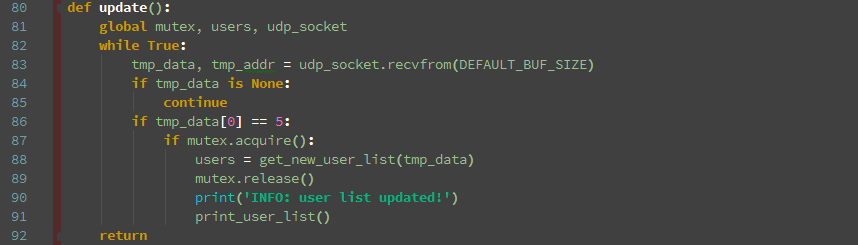


1. 新建udp\_thread线程，用以监听服务器端用户列表更新请求。

新建线程：



其中update方法如下：



1. 等待用户输入命令

命令分以下几种：

1. Help

输出帮助信息

1. list

输出用户列表

1. chat –t ip:port –m msg

向某用户发送信息。如向192.168.0.1:8000发送消息‘Hello’，则需输入命令：chat –t 192.168.0.1:8000 Hello

1. exit

登出

### 报文格式设计

客户端登录请求UDP报文

|  |  |  |
| --- | --- | --- |
| 0x01 | TCP监听端口 | 用户名 |
| 1 | 2 | 20 |

服务端登录确认UDP报文(含客户列表)

|  |  |  |  |
| --- | --- | --- | --- |
| 0x02 | 列表项数 | | |
| 1 | 4 | | |
| IP | | TCP监听端口 | 用户名 |
| 4 | | 2 | 20 |
| IP | | TCP监听端口 | 用户名 |
| 4 | | 2 | 20 |
| … | | | |

服务端更新客户列表UDP报文

|  |  |  |  |
| --- | --- | --- | --- |
| 0x05 | 列表项数 | | |
| 1 | 4 | | |
| IP | | TCP监听端口 | 用户名 |
| 4 | | 2 | 20 |
| IP | | TCP监听端口 | 用户名 |
| 4 | | 2 | 20 |
| … | | | |

TCP连接方身份信息

|  |  |
| --- | --- |
| TCP监听端口 | 用户名 |
| 2 | 20 |

TCP聊天信息格式

|  |  |
| --- | --- |
| 消息内容 | 0x0 |
|  | 1 |

客户端离开请求UDP报文

|  |
| --- |
| 0x3 |
| 1 |

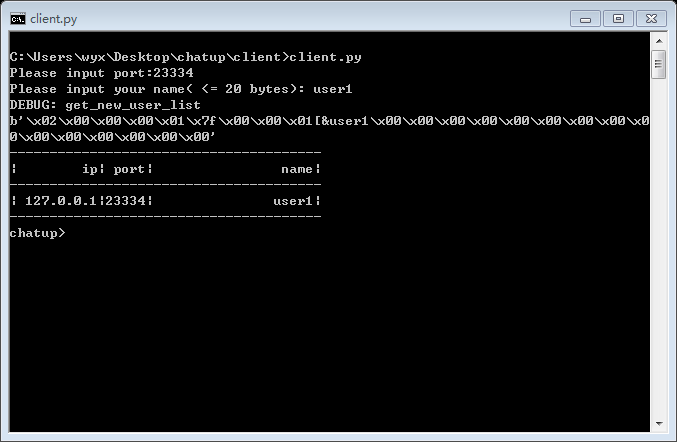
服务端离开确认UDP报文

|  |
| --- |
| 0x4 |
| 1 |

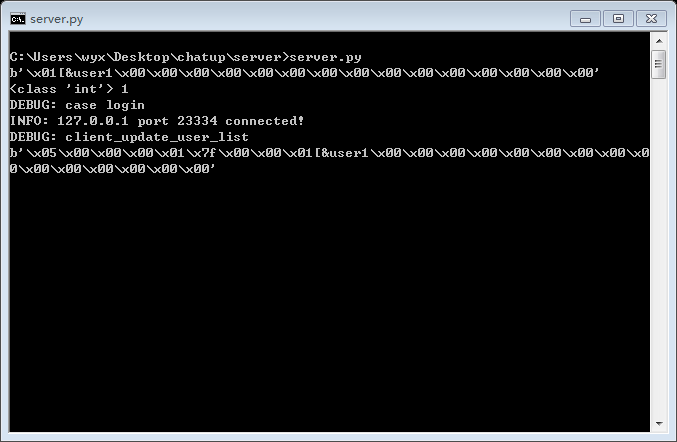
## 运行方法及演示

1. 用户1登录

用户1输出：

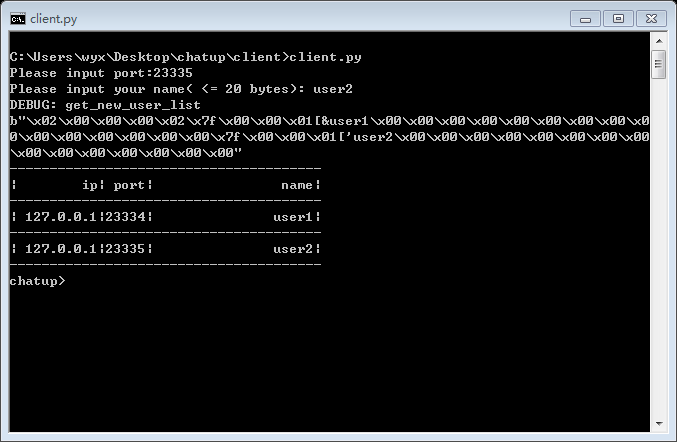


服务器端输出：

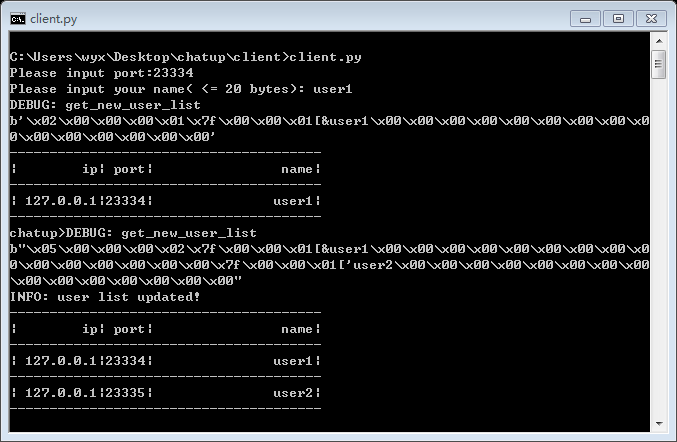


1. 用户2登录

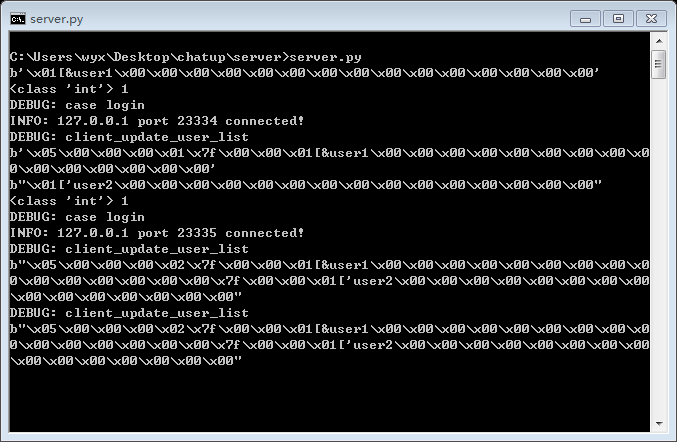
用户2端输出：



用户1端输出（收到update信息）：

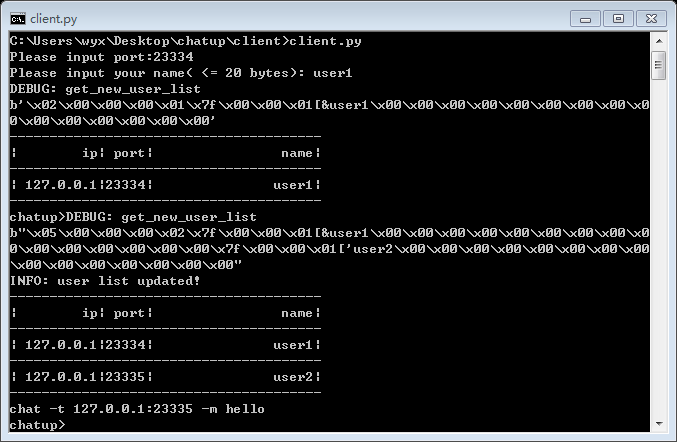


服务器端输出：

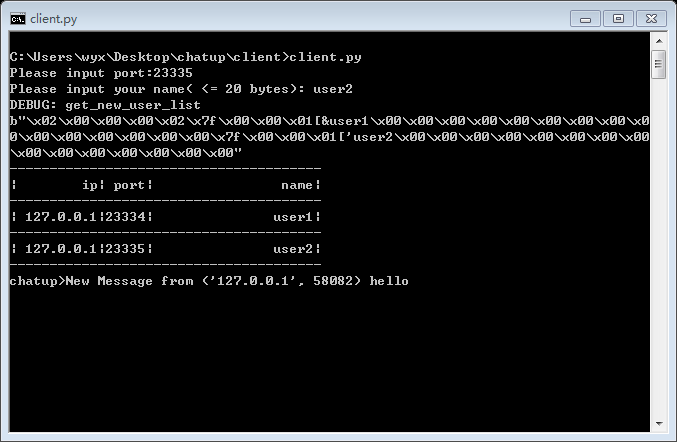


1. 用户1向用户2发送消息

用户1端命令：

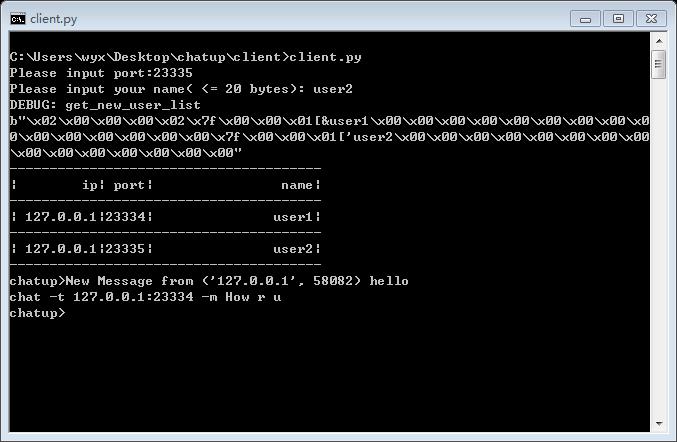


用户2端输出：

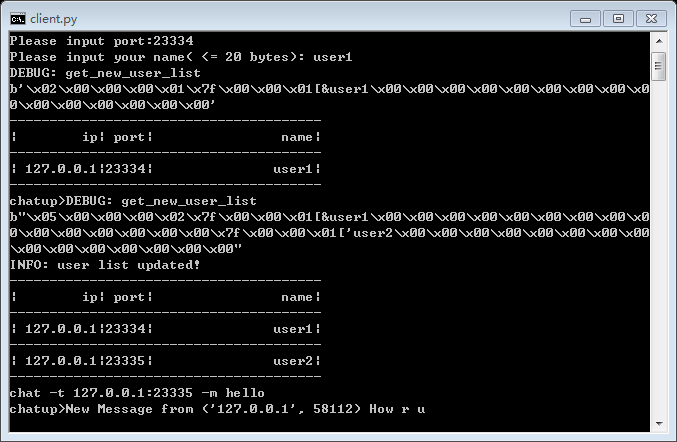


1. 用户2向用户1发送消息

用户2端命令：

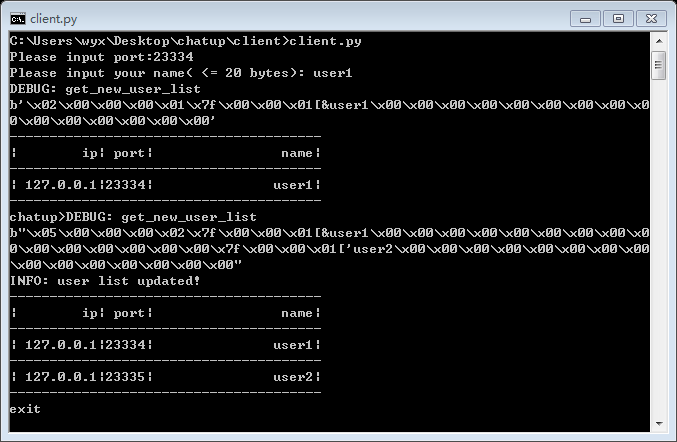


用户1端输出：

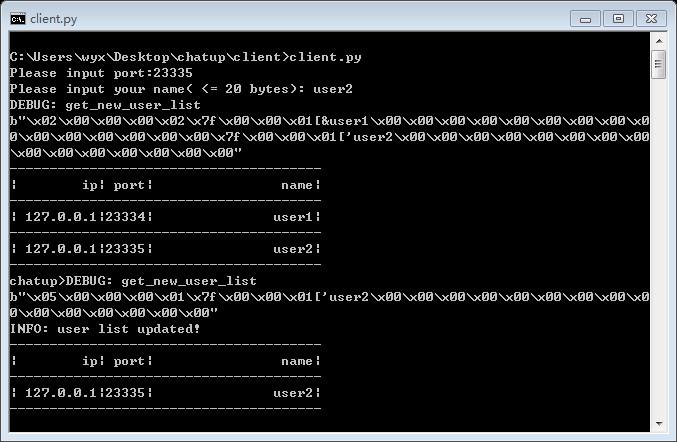


1. 用户1登出

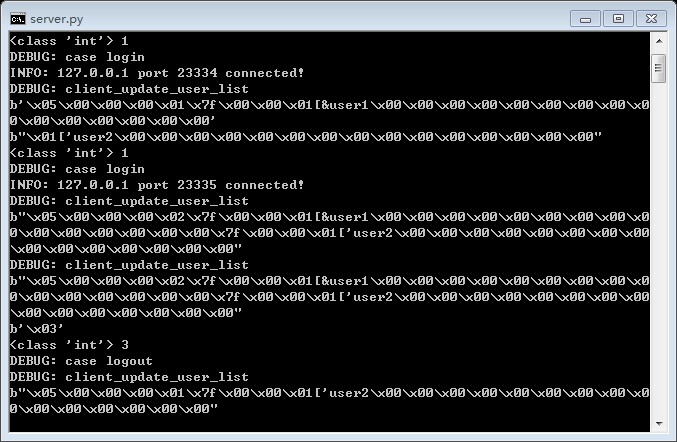
用户1端命令：



用话2端输出（接收到update信息）：



服务器端输出：



## 存在的问题

我认为，本系统最大的问题在用户交方面。

客户端一方面要时刻节后udp和tcp报文，同时要处理用户命令所以客户端逻辑但线程无法完成，只能靠多线程实现。

由于时间原因，本系统通过多线程实现了客户端的基本功能，但是交互逻辑并不是很好。