上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

计算机网络大作业项目文档

——聊天程序

学生姓名：吴 小 茜

学生学号：518021910628

专　　业：信 息 安 全

指导老师：姚 立 红

日　　期：2020.12.26

一、 概述

本项目实现了一个聊天程序，其功能有：多用户之间的一对一聊天，多用户之间的群组聊天，多用户之间的文件传输。本项目实现了一个用户友好的 GUI。

本项目基于 Python3 环境，使用 socket 库实现网络通信，使用 PyQt5 实现 GUI。

1.1 运行环境

运行平台：windows10

环境需求：python3.6/anaconda

说明：由于本项目用 python 实现，在生成可执行文件时由于包依赖生成了一些问题，所以为提供可执行文件。运行 Source/ChatServer.py，为聊天服务器程序；运行 Source/ChatClient.py，为聊天客户端程序。

1.2 编译工具

IDE：pycharm

Interpreter: anaconda3/python3.6

1.3 程序文件列表

程序文件	功能
Source/ChatServer.py	实现聊天服务器功能
Source/ChatClient.py	实现聊天客户端功能，运行多次可开启多个客户端
Source/server.ui	服务器程序界面，使用 QtDesign 设计 UI
Source/server.py	对 server.ui 文件运行 pyuic，生成的对应 py 文件，用于辅助界面设计
Source/client.ui	客户端程序界面，使用 QtDesign 设计 UI
Source/client.py	对 client.ui 文件运行 pyuic，生成的对应 py 文件，用于辅助界面设计
Source/icon/	存放界面设计图标
test_img.jpg	用于测试文件传输功能的文件

1.4 使用的库

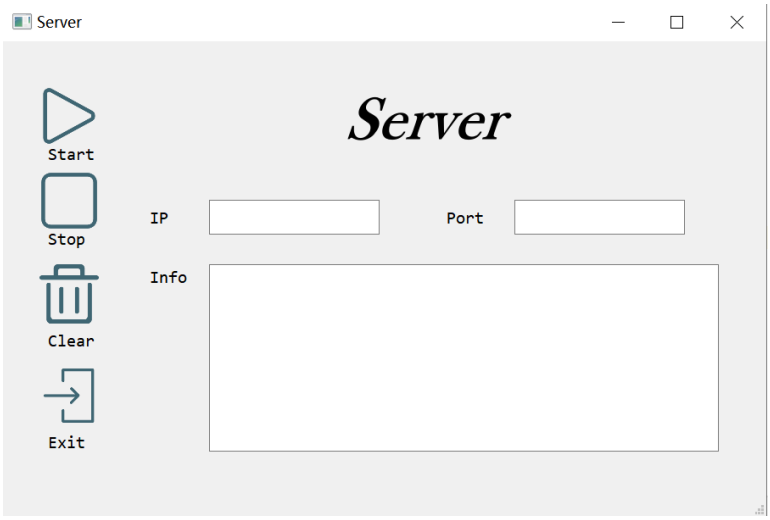
```
import sys
import socket
import threading
import struct
import os
from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog
from PyQt5 import QtCore, QtGui, QtWidgets
```

二、 主要数据结构

2.1 服务器端

数据结构	说明
app	应用程序对象，管理主事件循环 mainloop
Server	QMainWindows 对象，顶级窗口
Ui_Server	pyQt 的 UI 类，定义了窗口的布局和槽函数

Ui_Server 的界面如下：



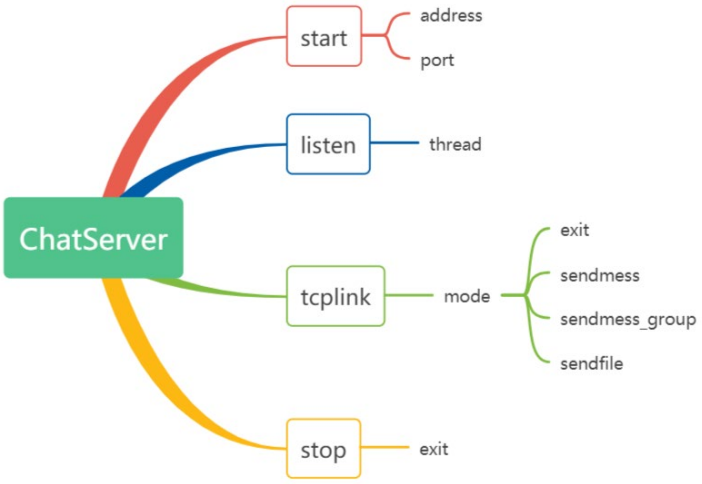
Ui_Server 中与界面设计相关的变量及对应的信号函数如下。（信号函数的说明见后表）

变量	类名称	信号函数
start_button	QtWidgets.QPushButton	start
stop_button	QtWidgets.QPushButton	stop
clear_button	QtWidgets.QPushButton	info.clear
exit_button	QtWidgets.QPushButton	exit
ip_edit	QtWidgets.QLineEdit	
port_edit	QtWidgets.QLineEdit	
info	QtWidgets.QTextBrowser	

Ui_Server 中定义的函数和功能变量列举如下：

函数/变量	功能
setupUi	类的初始化函数，定义窗口界面中的各个组件，并设置具体功能
retranslateUi	在将 xxx.ui 文件转换为 ui_xxx.h 文件的系统，uic 工具为.h 文件添加了 retranslateUi(QWidget *)函数，就是专门做的一个重新设置翻译文件的操作，不需要关闭或者隐藏任何一个窗体
exit	关闭窗口并停止监听
stop	关闭已有线程并停止监听
start	创建一个新的线程来执行监听，等待客户端的连接
listen	接收新连接并创建新线程来处理与客户端的连接
tcplink	创建套接字借助 tcp 协议来处理与客户端的连接
socketDict	字典类型，存放在线用户信息，在线用户的用户名用字符串存储

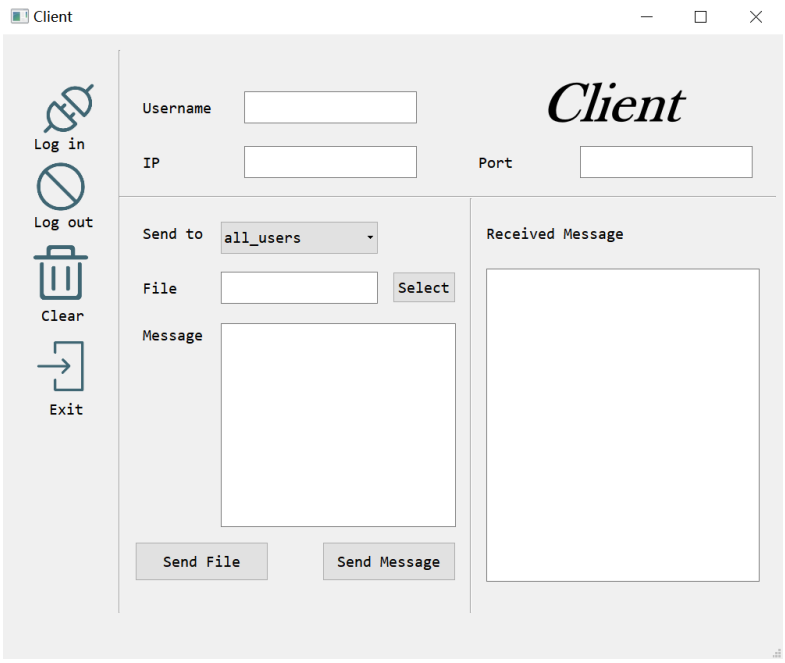
服务器端的主要结构如下图所示：



2.2 客户端

数据结构	说明
app	应用程序对象，管理主事件循环 mainloop
Client	QMainWindows 对象，顶级窗口
Ui_Client	pyQt 的 UI 类，定义了窗口的布局和槽函数

Ui_Client 的界面如下：



Ui_Client 中与界面设计相关的变量及对应的信号函数如下。（信号函数的说明见后表）

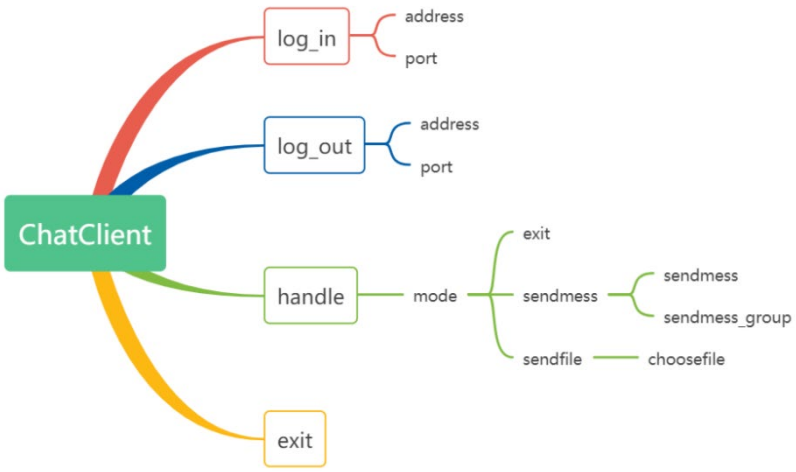
变量	类名称	信号函数/说明
login_button	QtWidgets.QPushButton	log_in
logout_button	QtWidgets.QPushButton	log_out
clear_button	QtWidgets.QPushButton	info.clear

exit_button	QtWidgets.QPushButton	Client.close
ip_edit	QtWidgets.QLineEdit	
username_edit	QtWidgets.QLineEdit	
port_edit	QtWidgets.QLineEdit	
receiver_choose	QtWidgets.QComboBox	选择消息接收者(Send to 一栏)，可选项包括“all_users”(群发消息)和其他所有在线用户
file_edit	QtWidgets.QLineEdit	存储待发送的文件路径
message_edit	QtWidgets.QTextEdit	编辑待发送的消息
file_button	QtWidgets.QPushButton	slot_btn_chooseFile
send_file_button	QtWidgets.QPushButton	sendfile
send_message_button	QtWidgets.QPushButton	send
info	QtWidgets.QTextBrowser	显示通信信息(Received Message)

Ui_Client 中定义的函数和变量列举如下：

函数/变量	**功能**
setupUi	类的初始化函数，定义窗口界面中的各个组件，并设置具体功能
retranslateUi	在将 xxx.ui 文件转换为 ui_xxx.h 文件的系统，uic 工具为.h 文件添加了 retranslateUi(QWidget *)函数，就是专门做的一个重新设置翻译文件的操作，不需要关闭或者隐藏任何一个窗体
log_in	用户登录，建立线程，发送给服务器并更新在线用户信息
log_out	用户登出，关闭线程，发送给服务器并更新在线用户信息
exit	关闭客户端，关闭线程，发送给服务器并更新在线用户信息
send	给指定用户发送消息
sendfile	给指定用户发送文件
handle	发送需执行的功能，用 mode 存储，判断 mode 的值并完成相应操作
chooseFile	选择本机的文件并上传等待发送

客户端的主要结构如下图所示：



三、 主要算法

本项目是基于客户端与服务端的信息传输设计的聊天程序项目。

整个项目的服务器端的设计包括两个部分：（1）启动关闭服务器（2）开始停止监听。客户端设计包括四个部分：（1）登录退出（2）多用户之间的一对一聊天（3）多用户之间的发送文件（4）多用户之间的群组聊天。下面对这几种模式的算法分别进行介绍。

3.1 服务器端算法

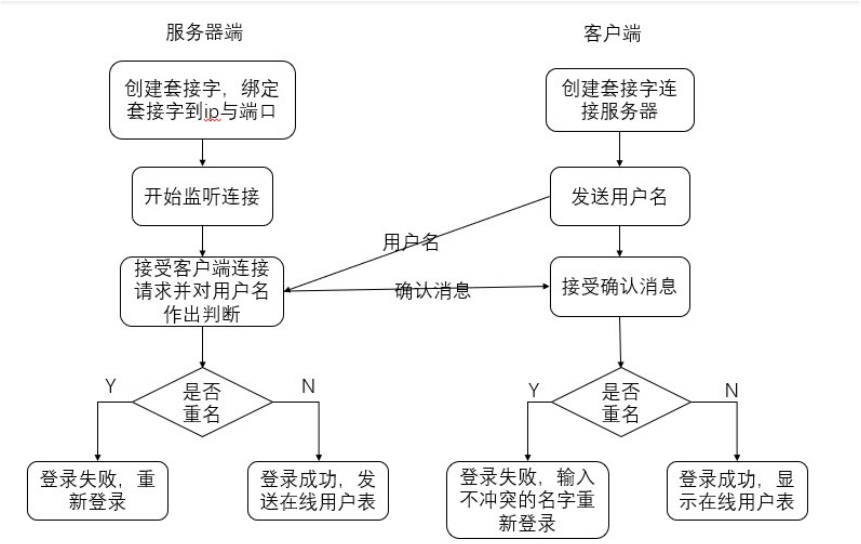
服务端启动的实现主要是通过创建线程监听客户端连接请求即可启动。停止监听和关闭服务器即结束线程。与客户端的连接主要由函数 tcplink 实现，借助 tcp，创建一个套接字来处理与客户端的连接，首先用字符串存储各在线用户的 id，并用字典类型存放所有在线的用户。并当有新用户登录或者在线用户退出登录后，更新字典 socketDict 并向其他所有的在线用户发送新的在线人信息。

接收客户端传来的信息并用 utf-8 解码后用字符串 mode 存储，对 mode 的判断可以确定客户端所要执行的具体操作。“exit”代表用户下线，“sendmess”代表一对一聊天，“sendmess_group”代表群组聊天，“sendfile”代表发送文件。

3.2 客户端的登陆退出

服务器开启监听后，若客户端进行登录操作，则会建立客户端与服务器端监听端口的连接，服务器可以监听到客户端的登录，并更新在线用户表发送其他在线用户。为了更加清晰地显示我们要展现的功能，将用户的用户名作为唯一标识指标，所以成功建立连接后，用户之间的搜索是基于服务器端的用户列表文本框进行查找。特别地，为增加可用性，定义布尔变量 connectFlag，若为 1 则代表已登录或者重名登录，会返回“you are already inline”提示信息。若为 0 则代表已退出或者在离线状态。当用户退出登录或关闭窗口，connectFlag 变为 0 且更新在线用户字典表。

登录过程流程图表示如下：



3.3 客户端一对一聊天

首先，客户端 A 执行 send 函数将 mode 更改为“sendmess”并向服务器端发送，用 utf- 8 编码解码，生成字符串：接收消息用户的用户名和发送的消息，中间用“-*-”隔开并用 utf-8 编码之后,发送给服务器端。

服务器端接收到 mode 值为“sendmess”，执行相应函数，将得到的第二条消息用 utf-

8 解码，遍历在线用户列表，当接收消息用户名匹配时，将消息编码并向客户端 B 发送。客户端 B 执行 handle 函数，判断 mode 为“sendmess”，解码相应消息并在对话框中显示。

3.4 客户端群聊

因为是基于 tcp 完成的该项目，所以都是有连接传输，故群发的实现与私人聊天类似，客户端 A 向服务器发送“sendmess_group”的 mode 指令，服务器对所有在线用户发现消息内容，所有在线用户客户端收到消息后，解码并在消息窗口中显示。

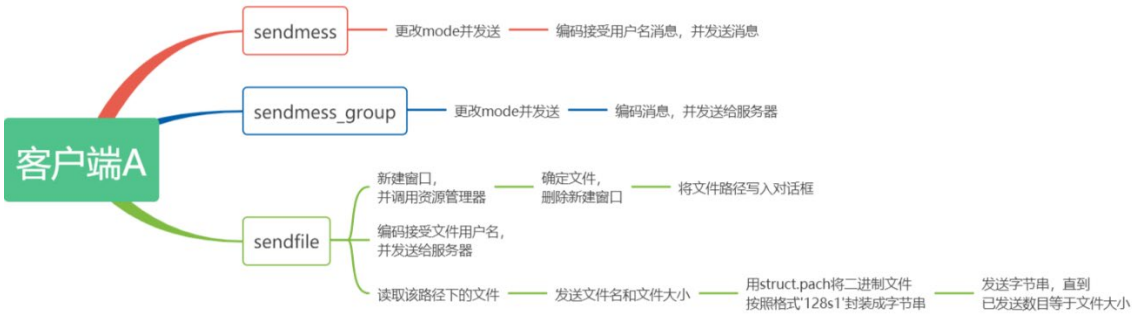
3.5 客户端的文件发送

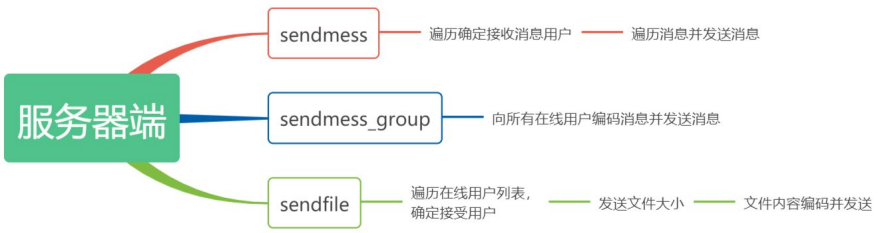
客户端 A 首先需要选择文件，利用 QFileDialog.getOpenFileName()方法，调用资源管理器，选择指定文件，并读取文件路径存入 file_edit，此时调用 sendfile 函数，首先发送 mode 为“sendfile”的指令和接收用户的用户名，然后读取该路径文件，读取文件大小，利用 struct.pack 将数据按照格式 128s1 把数据封装成字符串，将该字符串用 utf-8 编码后和文件大小用套接字传输给服务器。

服务器接收到“sendfile”的指令后，首先用遍历确定接收文件客户端的用户名，然后用 struct.calcsize 计算用给定的格式占用多少字节的内存并转发给客户端 B。将得到的字符串用 struct.unpack 解析，转发所得字节流，直至发送字节数等于文件大小。

客户端 B 收到“sendfile”的指令并调用 handle 函数，首先接收发送方用户名，下面的过程同服务器收到字节流的处理，计算要接收的文件的大小和名称，在根目录下，若接受的文件名已经存在，则新建文件用“new+源文件名称”的命名方式命名，接收字节流并用 struct.unpack 解析写入文件，直至接收到的字节数等于文件大小。接收文件成功，在对话框中返回相应信息。

客户端执行算法的流程图如下：





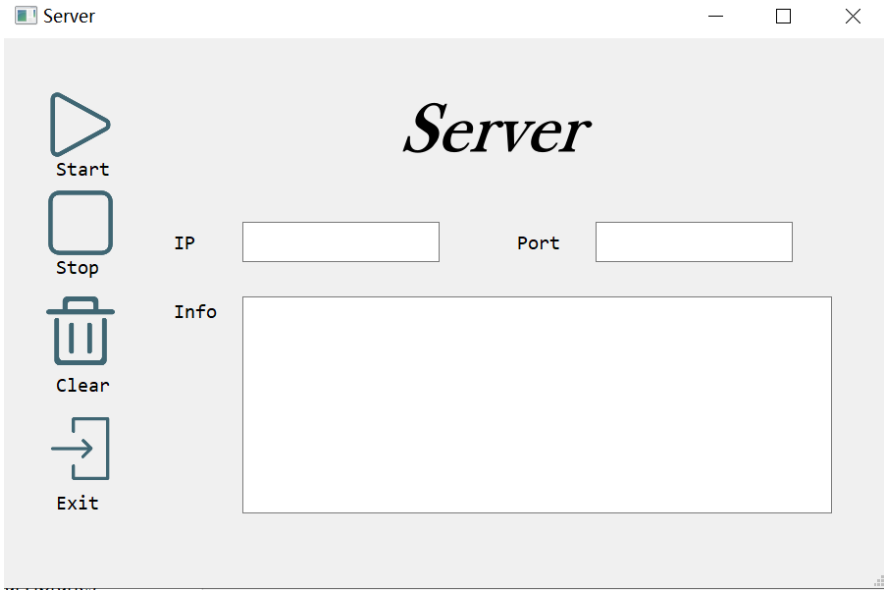
四、 程序测试截图及说明

运行平台：windows10

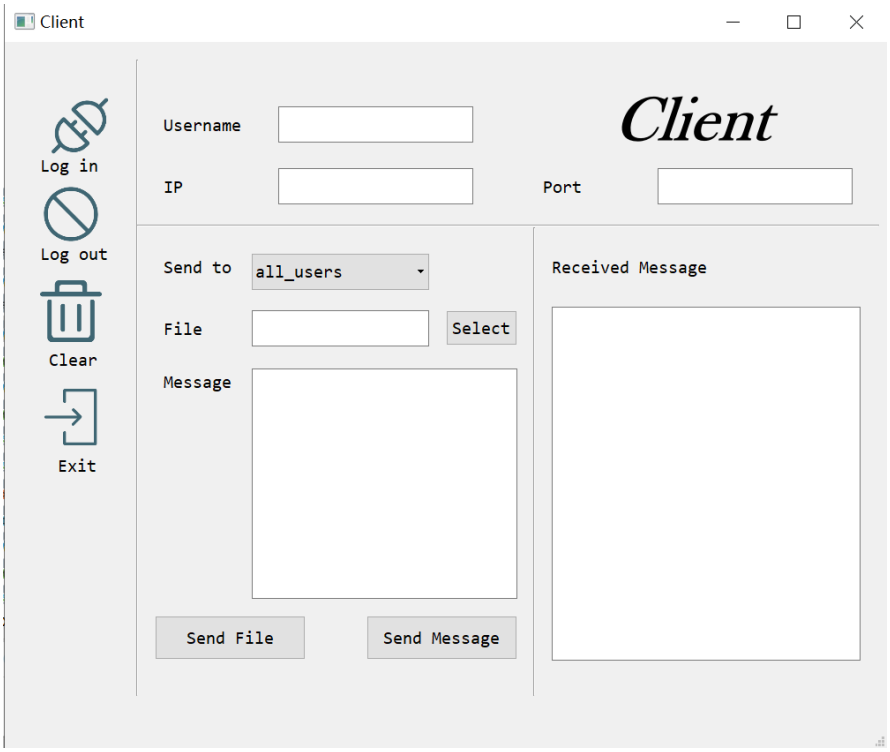
4.1 界面演示

运行 ChatServer.py 服务端，运行 ChatClient.exe 客户端，运行一次即代表一名用户，运行多次则代表多个用户登录。

服务器端：

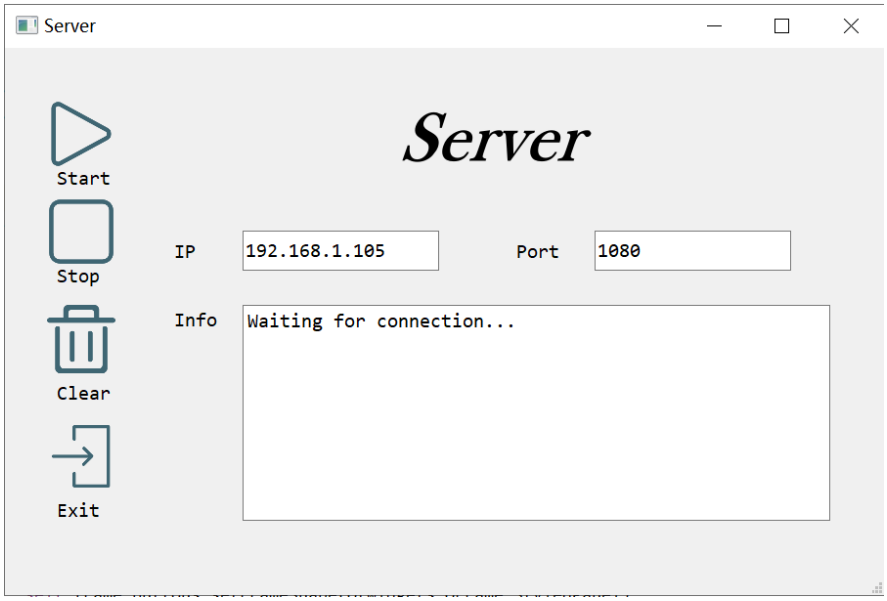


客户端：



4.2 服务器端开始监听

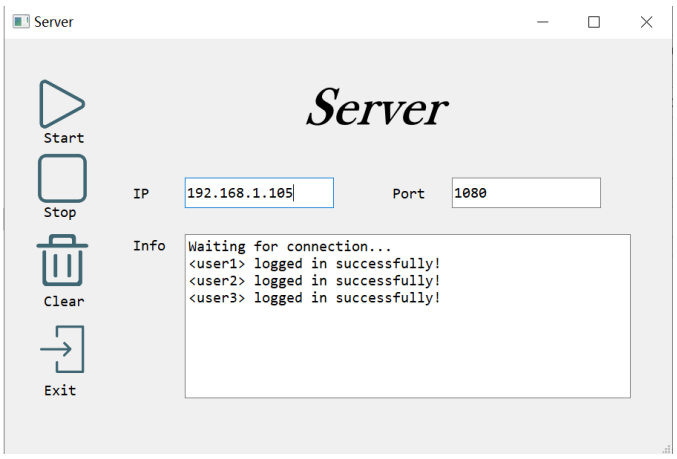
以物理本机作为服务器，填入本机 ip 地址，端口设置为 1080，点击 Start 开始监听。

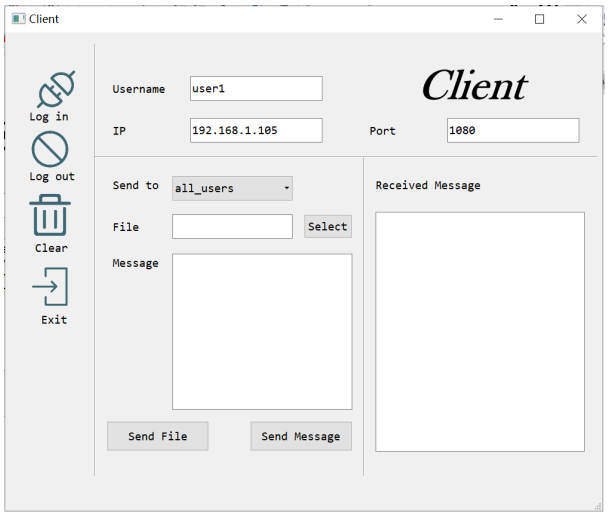


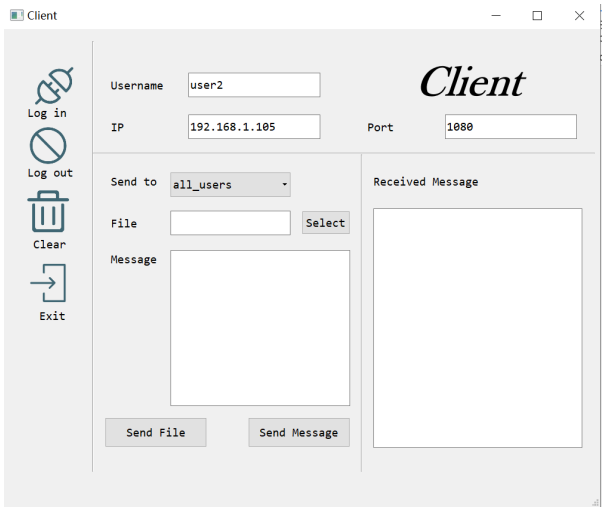
4.3 在客户端登录用户

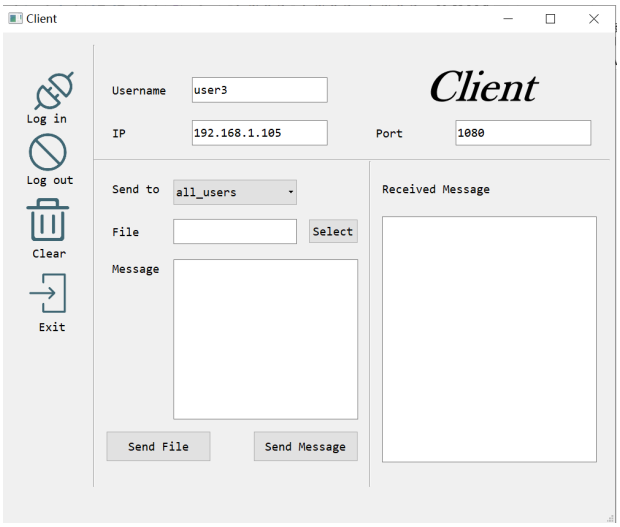
输入 ip 地址、端口号和 username 之后，点击 Log in 登录，保持 ip 地址和端口号与服务器端监听的 ip 地址和端口号相同即可。

此时服务器成功监听到三位用户的登录，且客户端的在线用户列表也已完成更新。如下所示：


Server
Start
Stop
Clear
Exit
Server
IP
192.168.1.105
Port
1080
Info
Waiting for connection...
<user1> logged in successfully!
<user2> logged in successfully!
<user3> logged in successfully!


Client
Log in
Log out
Clear
Exit
Username
user1
IP
192.168.1.105
Port
1080
Client
Send to
all_users
File
Select
Message
Received Message
Send File
Send Message

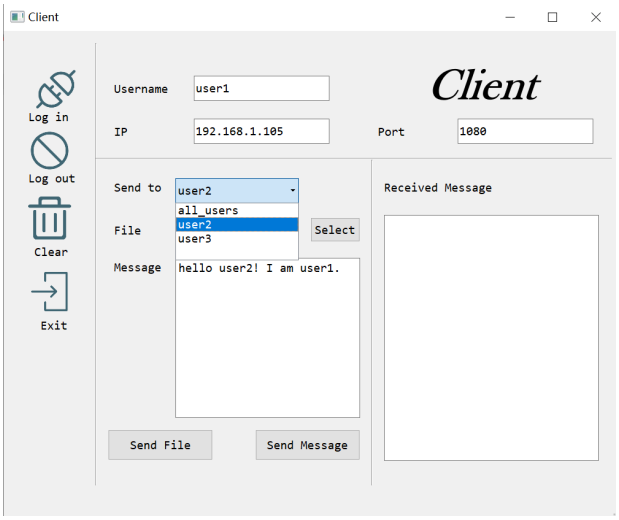

Client
Log in
Log out
Clear
Exit
Username
user2
IP
192.168.1.105
Port
1080
Client
Send to
all_users
File
Select
Message
Received Message
Send File
Send Message

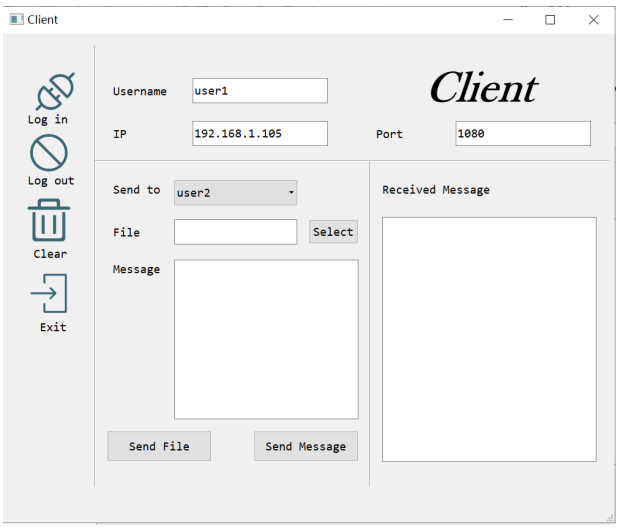


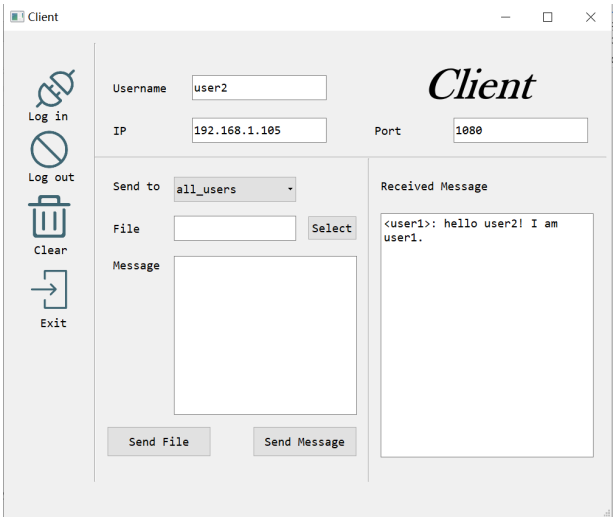
4.4 由用户 1 向用户 2 发送消息

在登录 user1 的客户端的发送消息窗口输入该消息，并在接收人一栏选中“user2”，点击“Send Message”即可发送消息。

用户 user2 接收到该消息，并显示在客户端的消息窗口中。user1 客户端发送消息窗口自动清空。

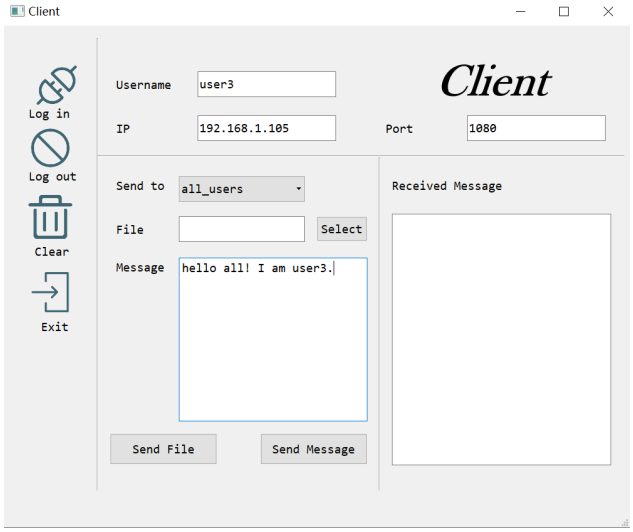


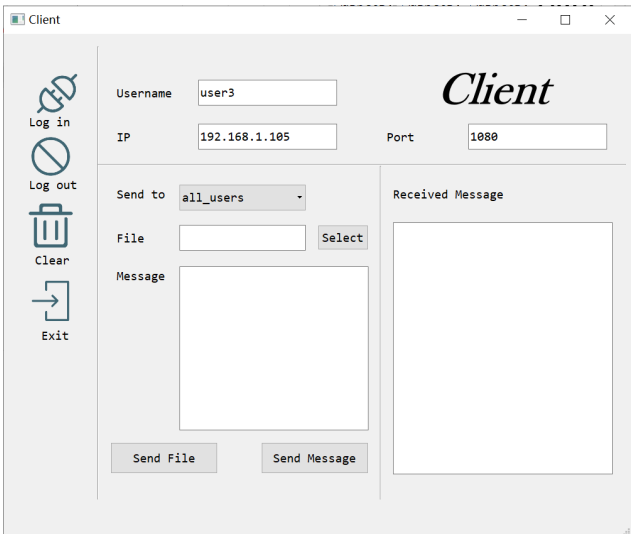


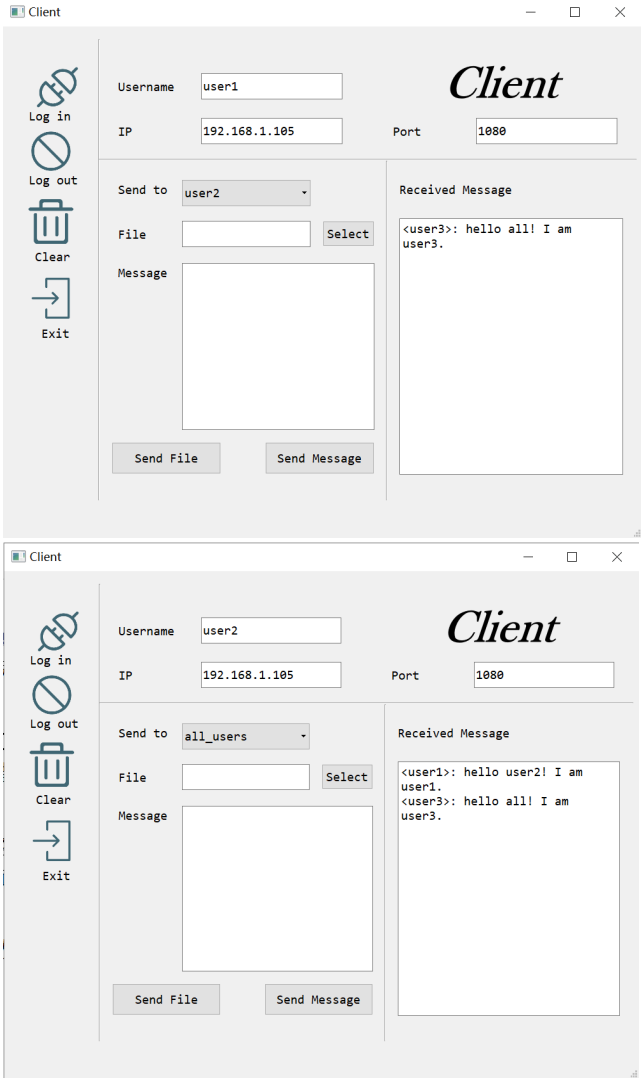


4.5 由用户 3 群发消息

在消息输入框中输入消息，并在接收人一栏选中“all_users”，即可完成群发消息。各用户在消息框中均能看到该消息。

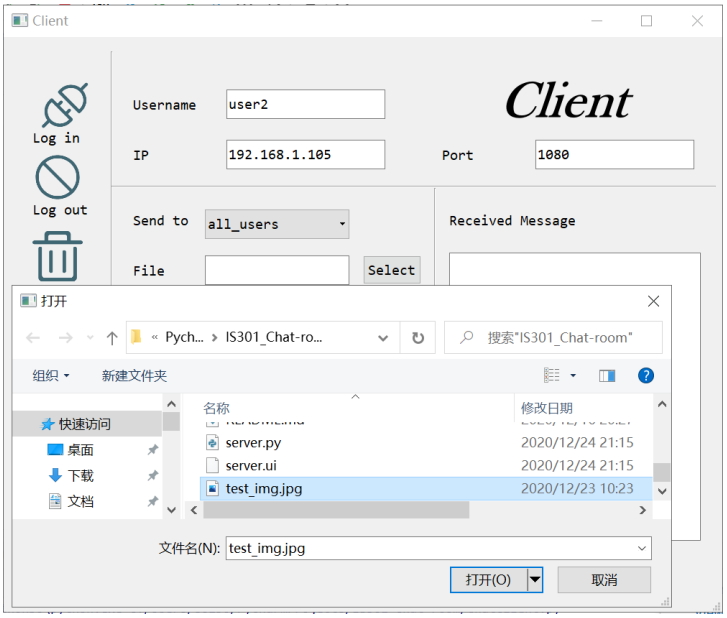






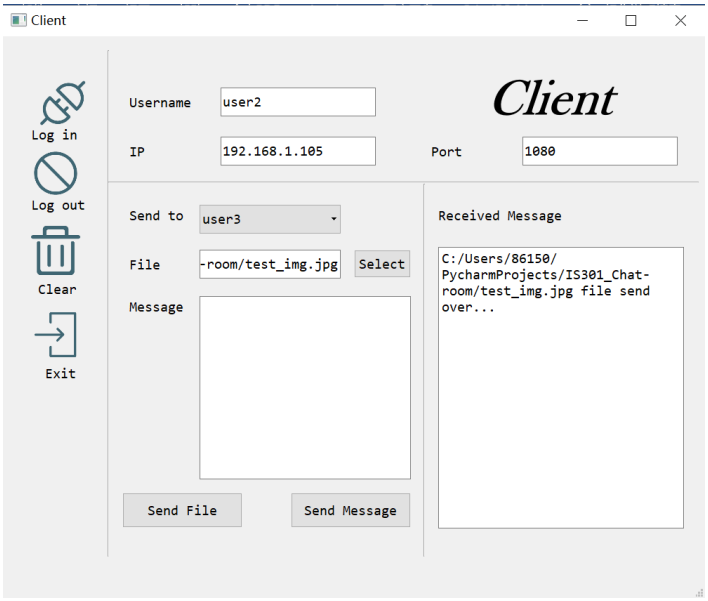
4.6 由用户 2 向用户 3 传送文件

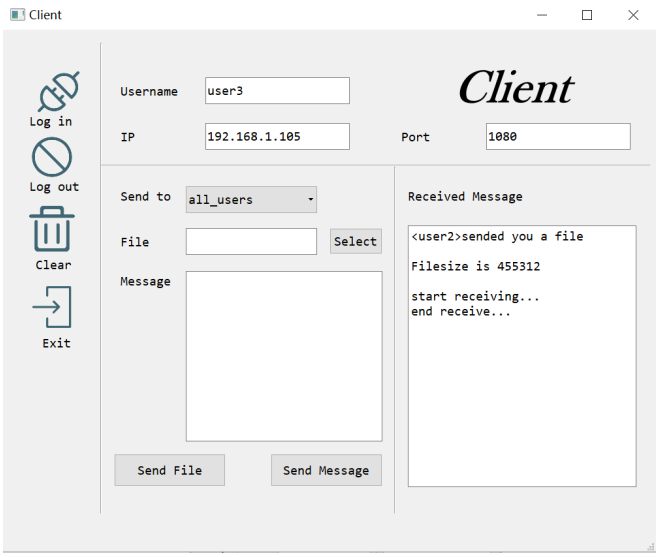
单击 File 一栏的 select，选择传输文件。选中文件 test_img.jpg，并确认，指定接收者为 user3,单击按钮“Send File”即可发送文件。



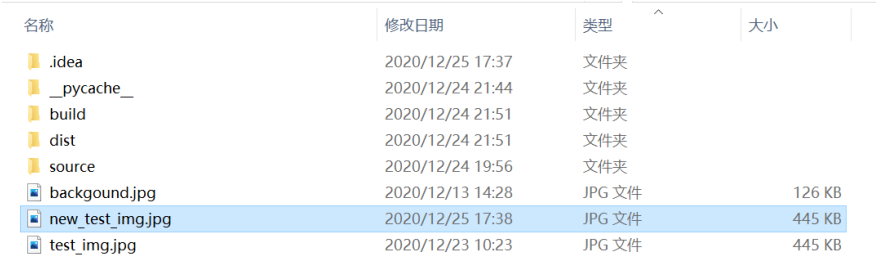
发送方显示“file send over”，接收方显示发送人和发送文件以及文件大小，并提示“start receiving”“end receive”。

说明：这里“Send to”一栏不能选择 all_users，选择 all_users 无法成功发送，因为对应群发文件功能在，做过尝试后有一些问题没有解决（具体见第五节 5.4 说明“群发文件功能的尝试”）。





查看文件列表，可以看到 user3 接受到了文件 test_img.jpg。接受到的文件默认保存在同级目录下，若已经存在同名文件，则在文件名首部加上“new_”前缀以示区别。由于此时 user2 和 user3 运行在同级目录下，user3 接受到的文件保存为 new_test_img.jpg。

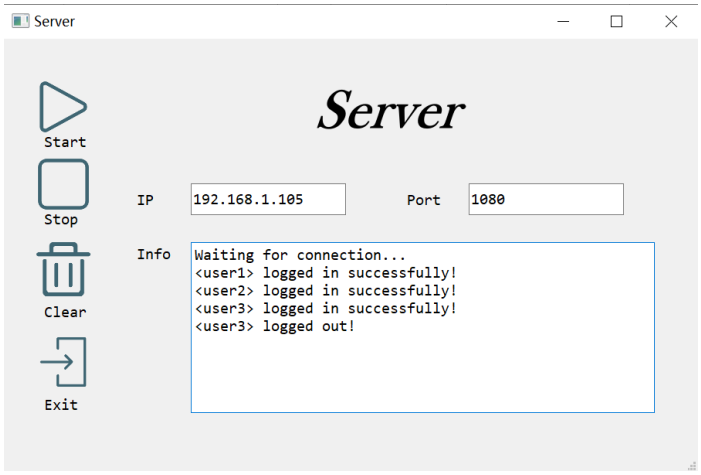


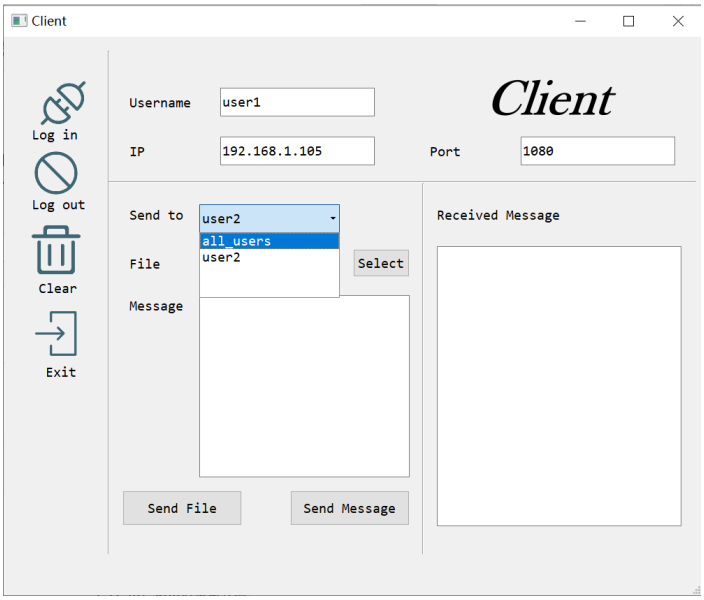
名称	修改日期	类型	大小
.idea	2020/12/25 17:37	文件夹	
__pycache__	2020/12/24 21:44	文件夹	
build	2020/12/24 21:51	文件夹	
dist	2020/12/24 21:51	文件夹	
source	2020/12/24 19:56	文件夹	
backgound.jpg	2020/12/13 14:28	JPG 文件	126 KB
new_test_img.jpg	2020/12/25 17:38	JPG 文件	445 KB
test_img.jpg	2020/12/23 10:23	JPG 文件	445 KB

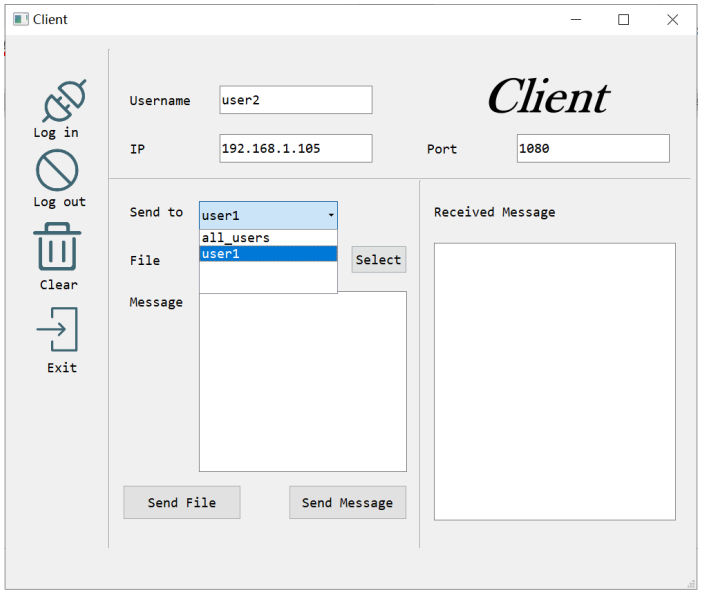
4.7 用户退出登录

单击按钮“log out”即可退出登录，单击“Clear”可清空消息栏，单击“Quit”可退出客户端界面。

退出 user3 用户并清空消息，服务器成功检测并显示“ming3 logged out!”，其他两个客户端的在线列表更新，发送消息的接受方列表更新，不再有 user3。

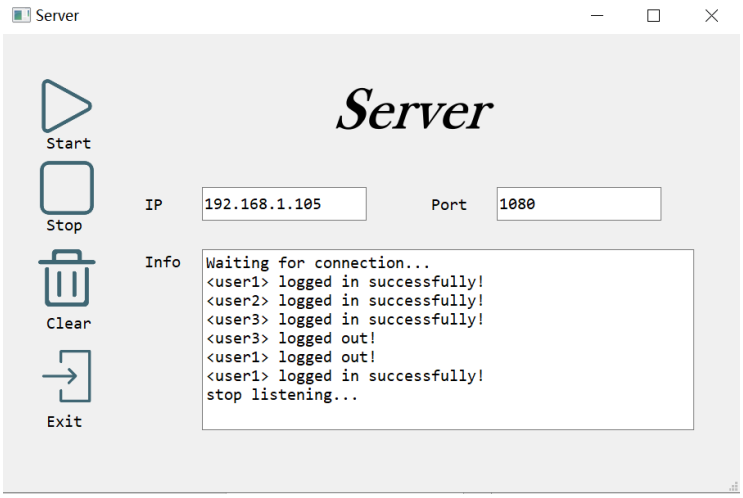






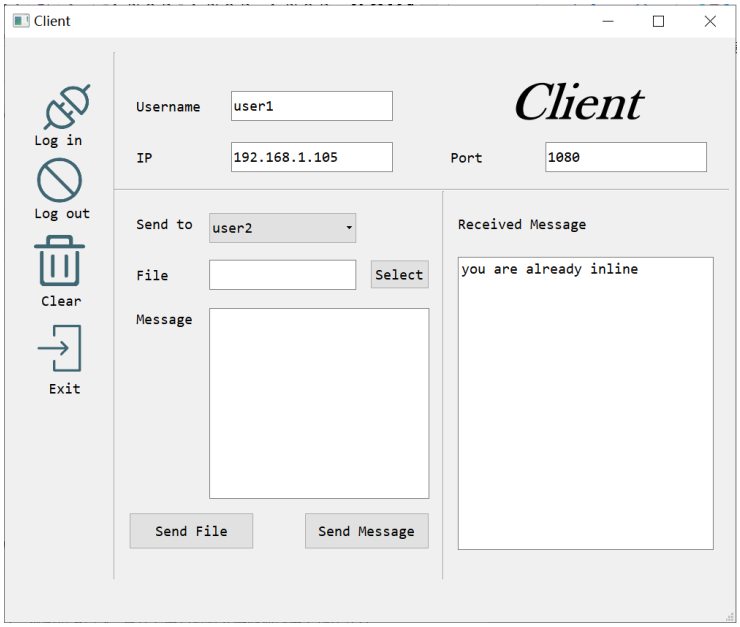
4.8 停止监听

在服务器端，单击按钮 Stop， 即可停止监听，消息窗口提示“stop listening”。

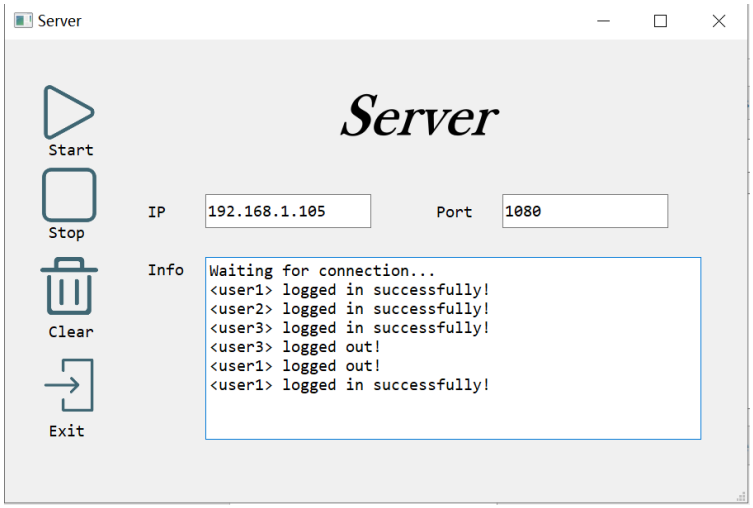


4.9 重复登入

客户端重复登录时，显示“you are already inline”。



4.10 用户退出后重新登陆



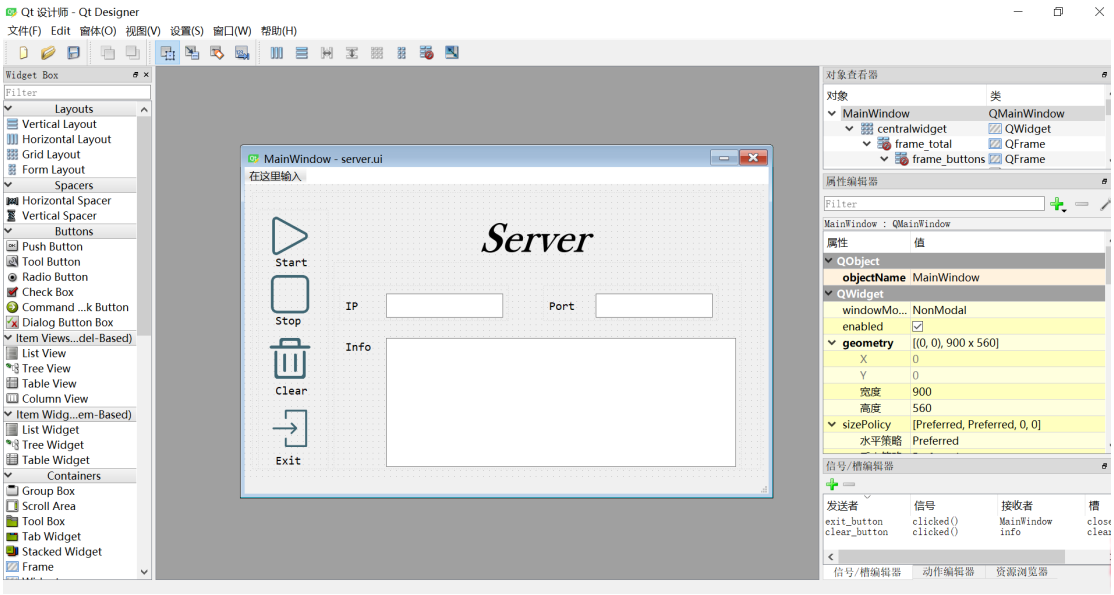
五、 遇到的问题及解决方法

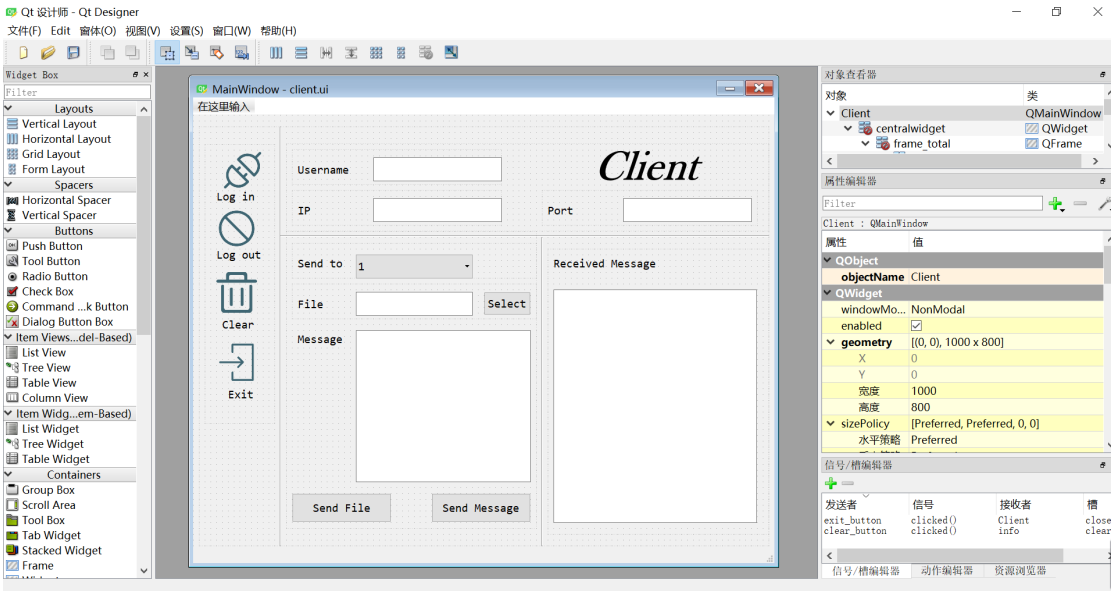
5.1 网络数据传输构思

开始选定题目时，我学习了 python socket 编程相关知识，对于用户之间的聊天，可以是服务器收到用户请求后，为发送者和接收者创建一对一的连接，也可以是通过存储转发，发送者的消息发送给服务器后通过服务器转发，考虑到拓展功能中有一对多即群发消息的要求，我采取了后一种设计即存储-转发的设计。

5.2 pyQt5 界面设计

开始时，我尝试用 Python 的 wx，tkinter 等模块设计 GUI，但这些模块设计功能有限，所以我尝试了 pyQt 的设计方式，具体来说，对于界面设计，我使用 QtDesigner，这是一种可视化的设计方式，能够方便的设计美化界面，通过 pyuic，可以使 UI 设计文件转变为 py 文件，基于这个 py 文件，我设计了其他聊天相关功能函数。





5.3 接收者列表设计

为了实现功能友好的界面，我用 QcomboBox 组件，设计了接收者列表的下拉菜单，可以通过这个组件查看当前在线的用户名，以及方便地选择消息的接受者，特别地，all_users 一项代表群发。而 QcomboBox 中的 item 需要根据服务器发来的用户列表实时更新，在实际操作中，若当前选中的用户 item 需要删除，客户端会报错，而且会出现某一 item 为空的情况。经过查阅资料和调试之后，我设计了 while 循环，从第一项开始覆盖文本，解决了这些问题。

```
439                 # 维护目前在线用户信息，并更新可供选择的接收者列表
440                 i = 0
441                 j = 1
442                 while i < len(messtext_split) and j < self.receiver_choose.count():
443                     user = messtext_split[i]
444                     if user != self.name:
445                         user = user.strip('<')
446                         user = user.strip('>')
447                         self.receiver_choose.setItemText(j, user)
448                         j += 1
449                     i += 1
450                 while i < len(messtext_split):
451                     user = messtext_split[i]
452                     if user != self.name:
453                         user = user.strip('<')
454                         user = user.strip('>')
455                         self.receiver_choose.addItem(user)
456                     i += 1
```

5.4 群发文件功能的尝试

在项目中，我实现了群发消息的功能，自然地，我希望实现群发文件的功能，和群发消息的逻辑基本一致，代码实现如下：

服务器：

```
308     # sendfile_group 在尝试之后，有bug没有解决
309     elif mode == "sendfile_group":
310         towho = sock.recv(1024).decode('utf-8')
311         fileinfo_size = struct.calcsize('128sl')
312         buf = sock.recv(fileinfo_size)
313 
314         if buf:
315             for onlinename in self.socketDic.keys():
316                 if onlinename != towho:
317                     self.socketDic[onlinename].send("sendfile".encode('utf-8'))
318                     self.socketDic[onlinename].send(sock_name.encode('utf-8'))
319                     self.socketDic[onlinename].send(buf)
320 
321             filename, filesize = struct.unpack('128sl', buf)
322             recvd_size = 0  # 定义已接收文件的大小
323             while not recvd_size == filesize:
324                 if filesize - recvd_size > 1024:
325                     data = sock.recv(1024)
326                     for onlinename in self.socketDic.keys():
327                         if onlinename != towho:
328                             self.socketDic[onlinename].send(data)
329                     recvd_size += len(data)
330                 else:
331                     data = sock.recv(filesize - recvd_size)
332                     for onlinename in self.socketDic.keys():
333                         if onlinename != towho:
334                             self.socketDic[onlinename].send(data)
335                     recvd_size = filesize
```

客户端：

```
403             '''
404             # 群发文件，在尝试之后，有bug没有解决
405             else:
406                 # 选择消息接收者为“all_users”，群发文件
407                 self.socketDict[self.name].send("sendfile_group".encode('utf-8'))
408                 self.socketDict[self.name].send((self.name).encode('utf-8'))
409                 filepath = self.file_edit.text()
410                 if os.path.isfile(filepath):
411                     fhead = struct.pack('128sl', os.path.basename(filepath).encode('utf-8'),
412                                         os.stat(filepath).st_size)
413                     self.socketDict[self.name].send(fhead)
414 
415                     fp = open(filepath, 'rb')
416                     while 1:
417                         data = fp.read(1024)
418                         if not data:
419                             self.info.append('{0} file send over...\n'.format(filepath))
420                             break
421                         self.socketDict[self.name].send(data)
422             '''
```

但是，在测试时遇到了一些问题，接收者一方无法正确接收被群发的文件，在解码消息时会报错“UnicodeDecodeError: 'utf-8' codec can't decode byte 0x90 in position 128: invalid start byte”。这个问题我查找不少资料，也调试了几天，但遗憾的是并没有成功解决，由于期末考试时间较紧，我希望在考试后 review 代码，弄懂这个 bug 的原因，实现这个功能，让项目更加完整。

六、 体会与建议

通过本次大作业的完成，我对课程中设计的理论知识包括套接字以及 C/S 模式等有了更加具体而深刻的认识，在完成大作业的同时，我查阅了很多资料，进一步了解了客户端服务器的交互是如何实现的，以及聊天程序的实现原理。同时，我熟练掌握了用 pyQt5 进行 GUI 设计，为了实现用户友好的界面，做出了不少尝试，并不断优化界面样式和控制逻辑。

我认为这次大作业是让自己的代码能力和项目能力得到提升的一次非常好的机会，在具体实现该项目从无到有的过程中，遇到很多困难，但是在查阅资料并不断调试直至解决的过程是充满成就感和愉悦感的。这是我为数不多的运用 python 具体实现工程项目，通过大作业的完成，我再一次训练了自己的编程技能和逻辑思维能力，不断地调试和编译使我得到一定进步。但我深知，项目也有一些可供完善优化之处，这也激励我不断学习，深化对计算机网络知识的理解，提高动手解决实际问题的能力。

最后，感谢老师和助教的教导！