计算机网络大作业：聊天软件开发

生命科学学院 计算生物学

2019141480195 郭新宇

1.开发过程：

1.1 socket 编程：

参考老师所发的 TCPClient、TCPServer 代码，学习 python 中的 socket 与 TCP 相关方法，编制程序，此阶段实现了客户机与服务器的通信，通过引入线程使得能够服务多台客户端，并能通过客户端发送消息进行消息对象识别，从而转发消息至相应客户端，实现了私聊功能。

代码示例（详细请参见代码文件）

```
def sendNickname():
    # 登陆函数，从输入框中获取服务器ip与端口号
    host = eip.get()
    port = eport.get()
    # 创建全局变量s，并创建socket
    global s
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, int(port)))
    # 获取昵称，创建接收服务器消息的线程，并向服务器发送登陆消息
    nickName = euser.get()
    t=threading.Thread(target=readFromServer, args=(s,))
    # 守护线程，为了使主线程结束时关闭子线程
    t.daemon=1
    t.start()
    s.send(nickName.encode())
```

1.2 界面开发：

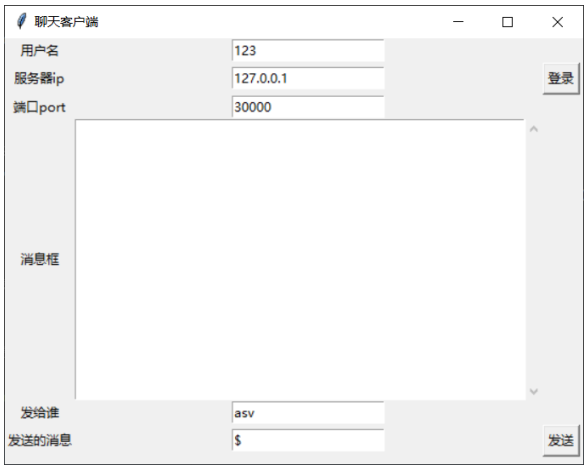
参考网络学习了 python 中的界面开发，使用 python 自带的 tkinter 完成了界面，此阶段进行了将登陆、消息显示、消息输入在界面上整合，并根据实际需要，引入了 time，以提示消息收到的时间，更符合现在使用的聊天软件。

客户端代码示例：

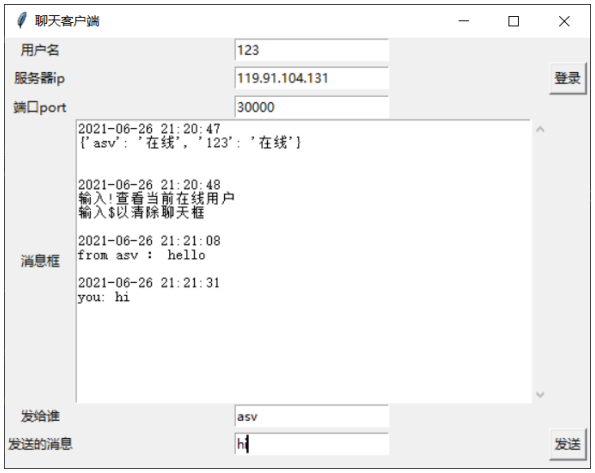
```
win.title("聊天客户端")
win.geometry("545x400+600+400")
win.protocol('WM_DELETE_WINDOW', my_close)
labelUse = tkinter.Label(win, text="用户名").grid(row=0, column=0)
euser = tkinter.Variable()
entryUser = tkinter.Entry(win, textvariable=euser).grid(row=0, column=1)

labelIp = tkinter.Label(win, text="服务器ip").grid(row=1, column=0)
eip = tkinter.Variable()
entryIp = tkinter.Entry(win, textvariable=eip).grid(row=1, column=1)

labelPort = tkinter.Label(win, text="端口port").grid(row=2, column=0)
eport = tkinter.Variable()
entryPort = tkinter.Entry(win, textvariable=eport).grid(row=2, column=1)

button = tkinter.Button(win, text="登录", command=sendNickname).grid(row=1, colu

text = scrolledtext.ScrolledText(win, height=20, width=60)
labeltext = tkinter.Label(win, text="消息框").grid(row=4, column=0)
text.grid(row=4, column=1)
```

1.3 本地服务器与云服务器搭建：

首先将服务器设定为本机 127.0.0.1 进行通信，



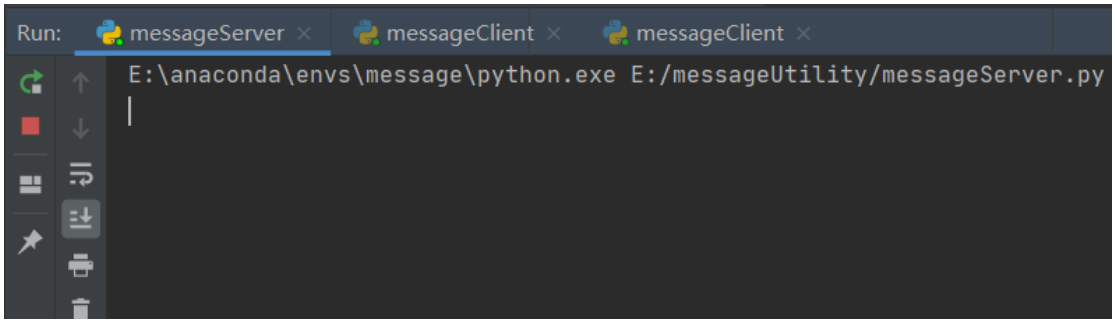
测试完成后在笔者的云服务器上安装 python3 并开启了服务器，最终本聊天室实现了跨机器之间的通信。



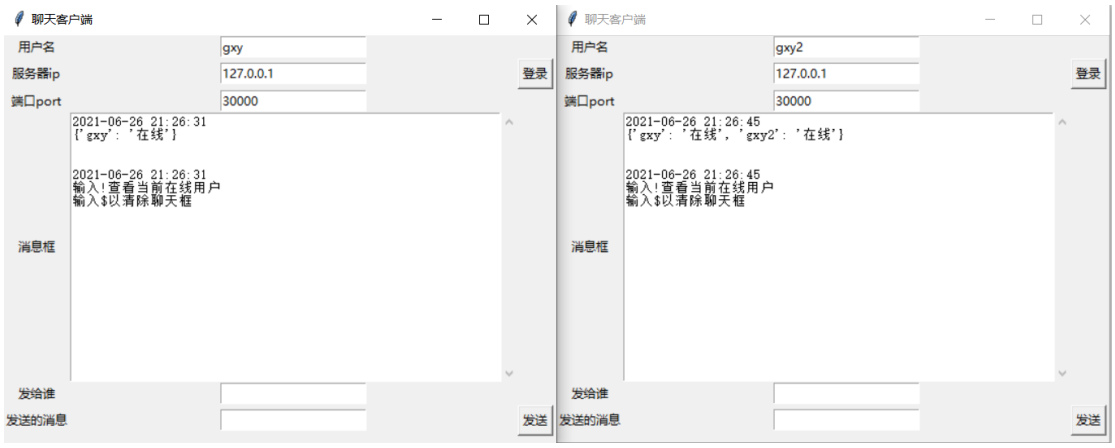
2.测试结果：

首先在将本机作为服务器进行测试：

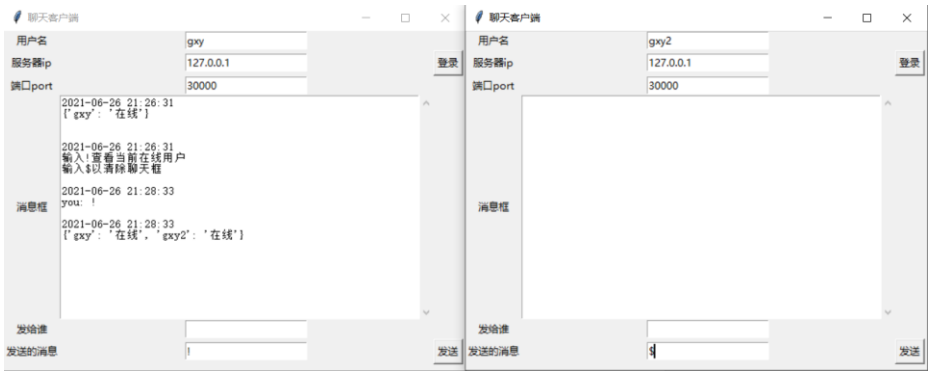
运行服务器：



设置允许并行运行 MessageClient 后，运行两客户端进行测试



由上图，登陆功能正常，登陆时显示在线用户并系统提示

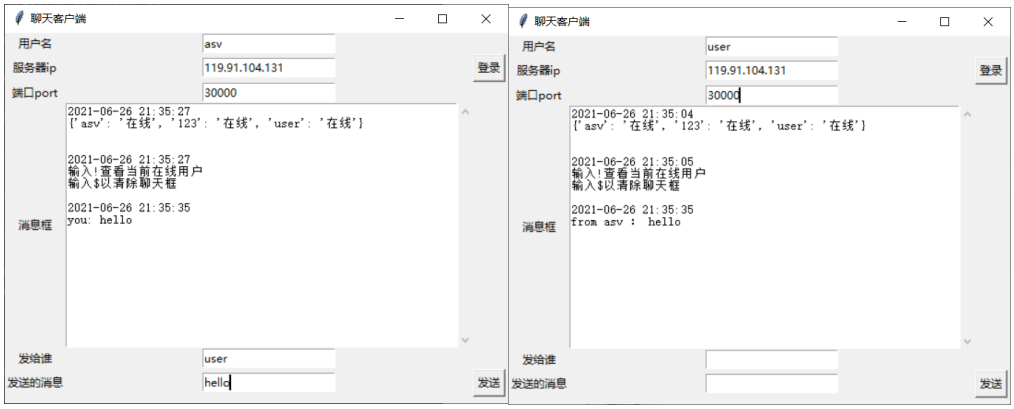


由上图，测试！获取在线用户列表与清除消息框也成功。

接来下进行云服务器测试

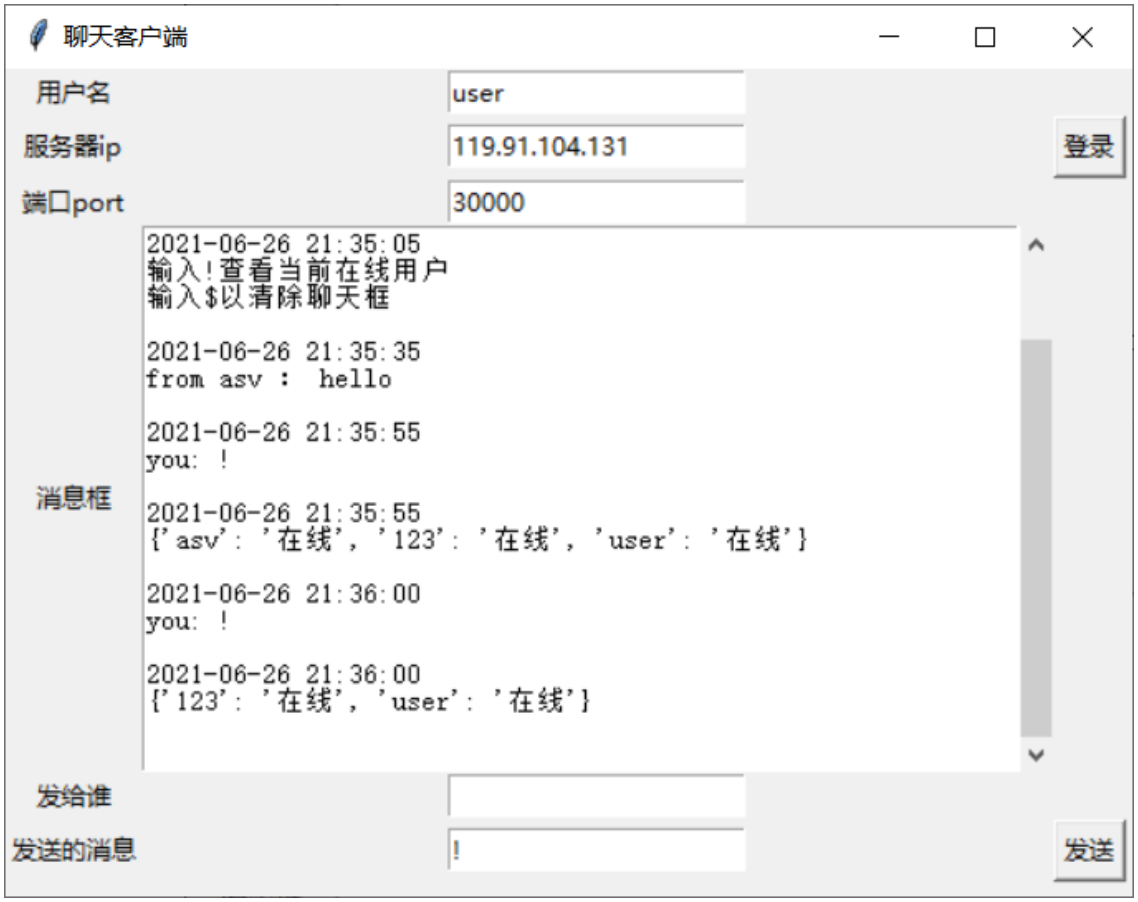
```
* Socket connection established *
Last failed login: Sat Jun 26 21:26:20 CST 2021 from 128.199.145.23 on ssh:notty
There were 70 failed login attempts since the last successful login.
Last login: Sat Jun 26 11:13:59 2021 from 119.28.22.215
(base) [root@VM-0-7-centos ~]# screen -ls
There is a screen on:
        21221.client    (Detached)
1 Socket in /var/run/screen/S-root.

(base) [root@VM-0-7-centos ~]#
```

在云服务器中使用 screen 工具使得 MessageServer.py 持续运行



测试结果如上，均成功。

下线情况测试：
一台客户端下线后，使用另一台客户端查询在线列表，发现下线成功。



3.遇到的问题以及其解决

首次使用 python 编程，虽然相较其他语言简单但也困难重重，在此列举几处问题。

1.关闭窗口后进程不退出

在关闭窗口后进程并未像主观所想会进行退出，查阅得知界面在进行 mainloop 的死循环，需要给 X 按钮也绑定退出事件，

```
win.protocol('WM_DELETE_WINDOW', my_close)
```

在一开始时直接绑定 sys.exit(0)发现这样会直接导致程序卡住，因此自定义了 my_close

```
# 窗体关闭时结束进程
def my_close():
    # True or Flase
    res = messagebox.askokcancel('提示', '是否关闭窗口')
    if res == True:
        if res == True:
            win.destroy()
```

以进行关闭。

但仅此还未结束，因为创建子线程原因，关闭主线程时子线程并不会随之关闭，因此程序继续与服务器保持通信，查阅资料后修改子线程参数后解决

```
t=threading.Thread(target=readFromServer, args=(s,))
# 守护线程，为了使主线程结束时关闭子线程
t.daemon=1
t.start()
s.send(nickName.encode())
```

2.创建 socket 问题

2.1

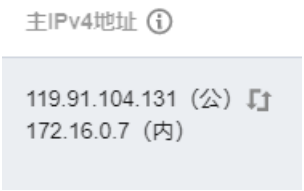
首先是创建 socket 时提示端口无效，因为端口传入的参数需要 int 型，进行强制类型转换后解决。

2.2

由于在函数中创建了 socket，在其他函数中调用时报错，设定创建的 socket 为 global 后问题解决

```
global s
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, int(port)))
```

3.云端配置问题



首次在云端运行服务器代码时，将 ip 绑定为公网 ip，结果发生错误，查询后了解到需要绑定内网 ip，修改后问题解决。