计算机网络与分布式系统

开发过程文档

选　　题　即时通讯系统

姓　　名　唐尉淋

学　　号　2018141424142

年　　级　2019

指导老师　宋万忠

1.说明

1.1. 项目名称

1） 即使通讯系统

1.2 项目需求

2） 系统要正确响应用户发出的消息，将其上传到服务器，分发给客户端。

3） 通讯系统至少可以满足三个客户端同时登录、通讯的需求。

4） 用户可以对即时通讯系统的工作端口以及 ip 进行设置。

1.4 开发环境

操作系统：Windows 10

开发工具：VScode

Python 版本：3.6

实现语言：python

2.系统需求分析

2.1 系统非功能需求

2.1.1 用户需求

系统的设计建立在用户的需求之上，通过对用户需求的分析，可以更好的实现系统，明确系统的设计方向。随着互联网的不断发展，人与人之间的交流也逐渐开始越来越依赖网络，在这种形势下，我们开发出了这款即时通信系统。本着用户至上的原则，我们对用户需求进行了分析调研。结果显示，用户除了对基本的通信功能的要求外，还希望能够拥有个性化的展示，同时希望能和好友进行更多的交互。

2.1.2 性能需求

本系统在保持界面的美观、易用的同时，尽量为用户提供及时的通讯响应，保证通讯过程无卡顿。

2.1.3 易用性需求

将 ip 地址、端口号写进 ini 配置文件中，方便用户进行更改和查看服务器客户端的 ip。

2.2 系统功能需求

2.2.1 登录功能

用户在输入用户名后，客户端向服务端发送请求，点击登录，进入主界面。

2.2.2 传输文件

用户可以选择本机的文件（大小有限制）发送到聊天室，其他用户可以在聊天室下载上传的文件，用户点击文件后文件开始上传，文件上传完成后会出现相应的提示。用户点击文件开始下载，下载成功后也会有相应提示。

2.2.3 文字聊天功能

用户进入聊天室后右侧会显示在线的用户，用户可以在聊天室内进行文本消息即时通讯。

2.2.4 表情聊天功能

用户在聊天室中可以选择已经缓存好的表情进行发送，发送速率较快。

2.2.5 图片聊天功能

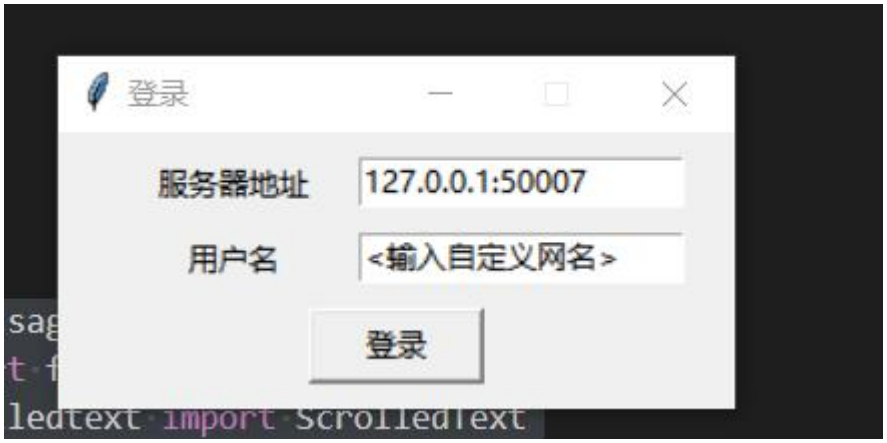
用户进入聊天室后右侧会显示在线的用户，用户可以在聊天室内进行图片的即时通讯，图片支持 jpg，gif，png 等格式。在有成员下线后，右侧列表的在线用户也会相应更新。

点击上传图片后可在本地电脑上选择相应的图片进行上传。

3.项目实现

3.1 登录功能

描述：本项目采用客户端/服务器端架构（C/S 架构），用户发送的请求都将传送到服务器端进行处理。服务器端包括数据库以及服务端后台软件。在用户输入用户名信息后，客户端通过对象流的方式向服务端发送一次请求，这是的链接是一次性的链接，服务端在接受到这个请求后就通过事先实现的相关协议对请求进行处理。处理完成后服务器端在将处理结果返还给客户端发起请求的用户。至此，一次请求结束。在登录成功后，在客户端开一个线程和服务器端保持通信。系统登录界面如图所示。



登录实现源码如下图所示

```
## 登录窗口
root1 = tkinter.Tk()
root1.title('登录')
root1['height'] = 110
root1['width'] = 270
root1.resizable(0, 0)  # 限制窗口大小

IP1 = tkinter.StringVar()
IP1.set('127.0.0.1:50007')  # 默认显示的ip和端口
User = tkinter.StringVar()
User.set('<输入自定义网名>')
```

```
# 登录按钮
def login(*args):
    global IP, PORT, user
    IP, PORT = entryIP.get().split(':')  # 获取IP和端口号
    PORT = int(PORT)  # 端口号需要为int类型
    user = entryUser.get()
    root1.destroy()  # 关闭窗口


root1.bind('<Return>', login)  # 回车绑定登录功能

but = tkinter.Button(root1, text='登录', command=login)
but.place(x=100, y=70, width=70, height=30)

root1.mainloop()

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.connect((IP, PORT))
# 发送用户名,没有输入用户名则标记no
socketUtils.send_string(conn, user if user else "no")
```

3.2 文字聊天功能实现

描述： 本项目支持简单的文字聊天。在发送文字信息时，客户端向服务端发送请求，服务端在接受到请求后将消息转发聊天室，消息的展示则通过对界面的时时更新来完成。在接受到服务器端发来的信息后通过取出发送者和接收者组成一个唯一标识。

实现：服务器端接收到消息后，处理消息请求的实现，如下图

```
    # 将接收到的信息(ip,端口以及发送的信息)存入que队列
    def update_que(addr, data):
        lock.acquire()
        try:
            que.put((addr, data))
        finally:
            lock.release()
```

```python
# 将队列que中的消息发送给所有连接到的用户
def sendData():
    while True:
        if not que.empty():
            data = ''
            message = que.get()  # 取出队列第一个元素
            if isinstance(message[1], str):  # 如果data是str则返回Ture
                for i in range(len(users)):
                    # user[i][1]是用户名, users[i][2]是addr, 将message[0]改为用户名
                    for j in range(len(users)):
                        if message[0] == users[j][2]:
                            data = ' ' + users[j][1] + ':' + message[1]
                            break
                    socketUtils.send_string(users[i][0], data)
                    # users[i][0].send(data.encode())
            data = data.split(':;')[0]
            if isinstance(message[1], list):  # 同上
                # 如果是list则打包后直接发送
                for i in range(len(users)):
                    socketUtils.send_json(users[i][0], message[1])
                    # users[i][0].send(data.encode())
```

客户端发送消息

```python
# 创建输入文本框和关联变量
a = tkinter.StringVar()
a.set('')
entry = tkinter.Entry(root, width=120, textvariable=a)
entry.place(x=5, y=348, width=570, height=40)


def send(*args):
    # 没有添加的话发送信息时会提示没有聊天对象
    # user_list.append()
    if chat not in user_list and chat != '----------群聊----------':
        tkinter.messagebox.showerror('发送失败', message='没有聊天对象!')
        return
    if chat == user:
        tkinter.messagebox.showerror('发送失败', message='不能私聊自己!')
        return
    mes = entry.get() + ':;' + user + ':;' + chat  # 添加聊天对象标记
    print("mes", mes)
    socketUtils.send_string(conn, mes)
    a.set('')  # 发送后清空文本框
```

3.3 图片聊天功能实现

描述：图片聊天是文件传输的一种体现，客户端读取本地文件并通过字节数组流将文件转换为字节数组进行传输。服务器将信息转发给相应的客户端即可。

客户端发送图片请求如下：

```
###### 图片功能代码部分
pictureName = './客户端图片缓存/' + '缓存图.jpg'
pho = ImageTk.PhotoImage(file=pictureName)

file_conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
file_conn.connect((FILE_SERVER_IP, FILE_SERVER_PORT))


# 从图片服务端的缓存文件夹中下载图片到客户端缓存文件夹中
def fileGet(name):
    message = 'get ' + name
    print("fileGet", message)
    socketUtils.send_string(file_conn, message)
    file_save_dir = os.path.join("客户端图片缓存", user)
    filepath = socketUtils.recv_file(file_conn, file_save_dir)
    # 打开图片然后贴到聊天框
    photo = Image.open(filepath)
    photo = photo.resize((100, 100), Image.ANTIALIAS)  # 规定图片大小
    # 如果pho和窗口作用域不一样就只能显示空白图片
    if filepath in IMAGE_POOL:
        tk_image = IMAGE_POOL.get(filepath)
    else:
        tk_image = ImageTk.PhotoImage(photo)
        IMAGE_POOL[filepath] = tk_image
    listbox.image_create(tkinter.END, image=tk_image)
    listbox.insert(tkinter.END, "\n")


# 将图片上传到图片服务端的缓存文件夹中
def filePut(fileName):
    # 截取文件名
    name = fileName.split('/')[-1]
    message = 'put  ' + name
    # 延时确ss.send(message.encode())
    socketUtils.send_string(file_conn, message)
    socketUtils.send_file(file_conn, fileName)
    time.sleep(0.1)
    # 上传成功后发一个信息给所有客户端
    mes = '``#' + name + ':;' + user + ':;' + chat
    socketUtils.send_string(conn, mes)
```

```
# 创建发送图片按钮
pBut = tkinter.Button(root, text='图片', command=picture)
pBut.place(x=65, y=320, width=60, height=30)
```

3.4 config 系统配置实现

描述:系统从 config.ini 中获取设定的服务器端的 ip 地址与端口号。

实现：config 如下图所示：

```
1   import configparser
2
3   cf = configparser.ConfigParser()
4   cf.read('requirement.ini')
5   CHAT_SERVER_IP = cf.get('ip', 'chat_server_ip')
6   FILE_SERVER_IP = cf.get('ip', 'file_server_ip')
7   PICTURE_SERVER_IP = cf.get('ip', 'picture_server_ip')
8   CLIENT_IP = cf.get('ip', 'client_ip')
9
10  CHAT_SERVER_PORT = int(cf.get('port', 'chat_server_port'))
11  FILE_SERVER_PORT = int(cf.get('port', 'file_server_port'))
12  PICTURE_SERVER_PORT = int(cf.get('port', 'picture_server_port'))
13  CLIENT_PORT = int(cf.get('port', 'client_port'))
14
```

Ip 地址以及端口号的配置文件如下图:

```
1   [ip]
2   chat_server_ip = 127.0.0.1
3   file_server_ip = 127.0.0.1
4   picture_server_ip = 127.0.0.1
5   client_ip = 127.0.0.1
6
7   [port]
8   chat_server_port = 50007
9   file_server_port = 50008
10  picture_server_port = 50009
11  client_port = 50010
```

3.5 文件传输的实现

文件传输与图片传输实现的方法类似，同样是通过传输字节数组来完成的。

服务器端传输文件实现如下：

```
# 传输当前目录列表
def sendList(conn):
    listdir = os.listdir(os.getcwd())
    listdir = json.dumps(listdir)
    socketUtils.send_string(conn, listdir)

# 发送文件函数
def sendFile(message, conn):
    name = message.split()[1]  # 获取第二个参数(文件名)
    print("发送文件", conn)
    for _ in range(10):
        if not FILE_WRITE_LOADING.get(name) is False:
            time.sleep(0.5)
        else:
            break

    socketUtils.send_file(conn, name)
```

客户端下载文件实现如下图所示:

```
# 接收下载文件(get)
def get(message):
    name = message.split()
    name = name[1]  # 获取命令的第二个参数(文件名)
    # 选择对话框，选择文件的保存路径
    fileName = tkinter.filedialog.asksaveasfilename(title='保存文件到', initialfile=name)
    # 如果文件名非空才进行下载
    if fileName:
        socketUtils.send_string(conn, message)
        file_save_path = socketUtils.recv_file(conn, filepath=fileName)
        if os.path.isfile(file_save_path):
            tkinter.messagebox.showinfo(title='提示', message='下载成功!')
```

客户端上传文件具体实现如下图所示:

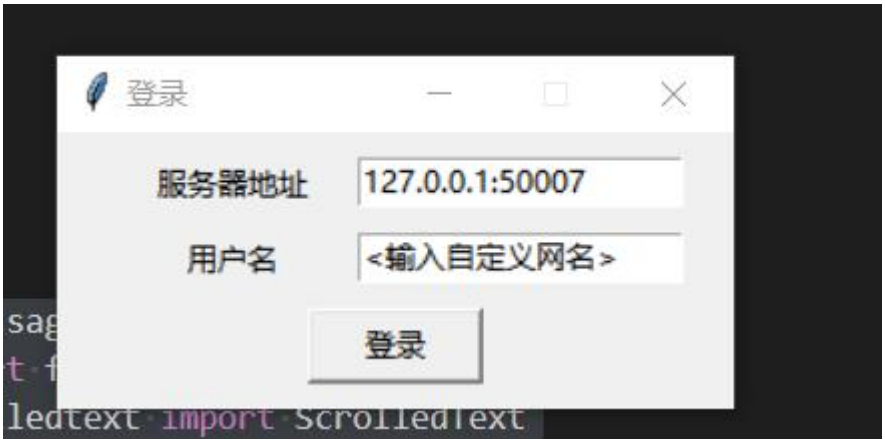
```
# 上传客户端所在文件夹中指定的文件到服务端，在函数中获取文件名，不用传参数
def put():
    # 选择对话框
    fileName = tkinter.filedialog.askopenfilename(title='选择上传文件')
    # 如果有选择文件才继续执行
    if fileName:
        name = fileName.split('/')[-1]
        message = 'put ' + name
        socketUtils.send_string(conn, message)
        socketUtils.send_file(conn, fileName)
        tkinter.messagebox.showinfo(title='提示', message='上传成功!')
    cd('cd same')
    lab()  # 上传成功后刷新显示页面
```

4.项目展示

1. 运行 sever.py 与 pictureSever.py 文件启动服务器。涉及的 ip 和主机名写入了 ini 配置文件中.

2. 首先实现了多用户登录，可以同时在三台主机上进入聊天室，运行程序后，会显示用户登录界面。

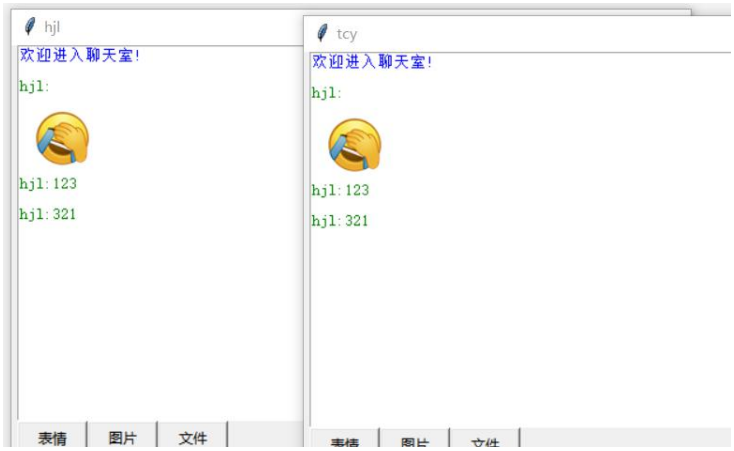
注意：用户名不要带特殊符号，否则被默认当作文件路径，会产生冲突。



3. 用户进入聊天室后右侧会显示在线的用户，用户可以在聊天室内进行文本消息与图片的即时通讯，图片支持 jpg，gif，png 等格式，在有成员下线后，右侧列表的在线用户也会相应更新。

点击上传图片后可在本地电脑上选择相应的图片进行发送。

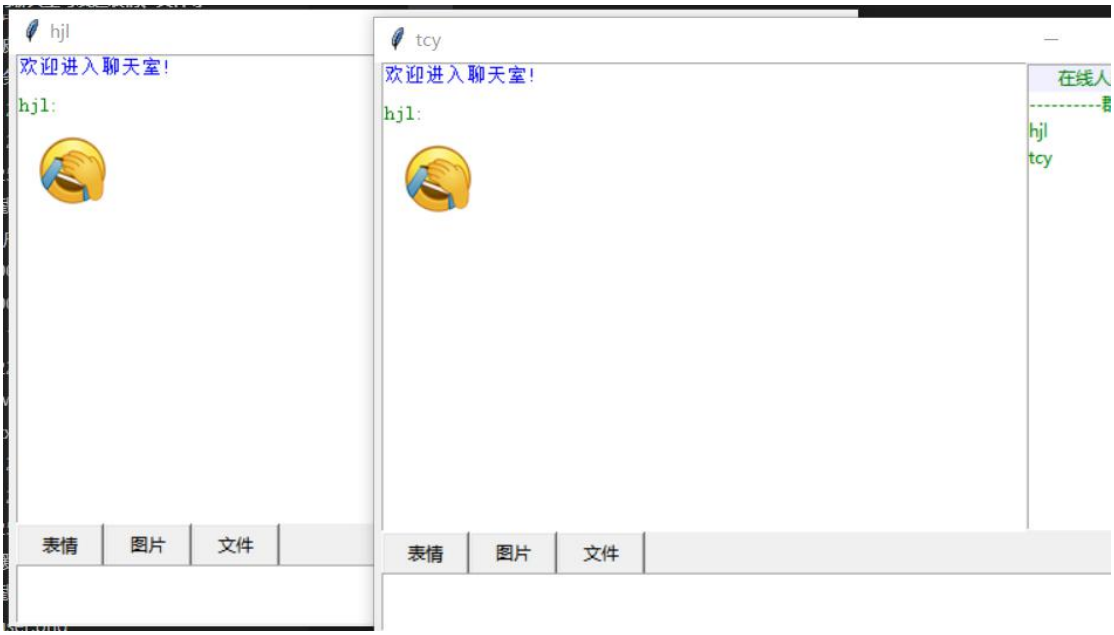
文本消息发送：



图片发送测试：

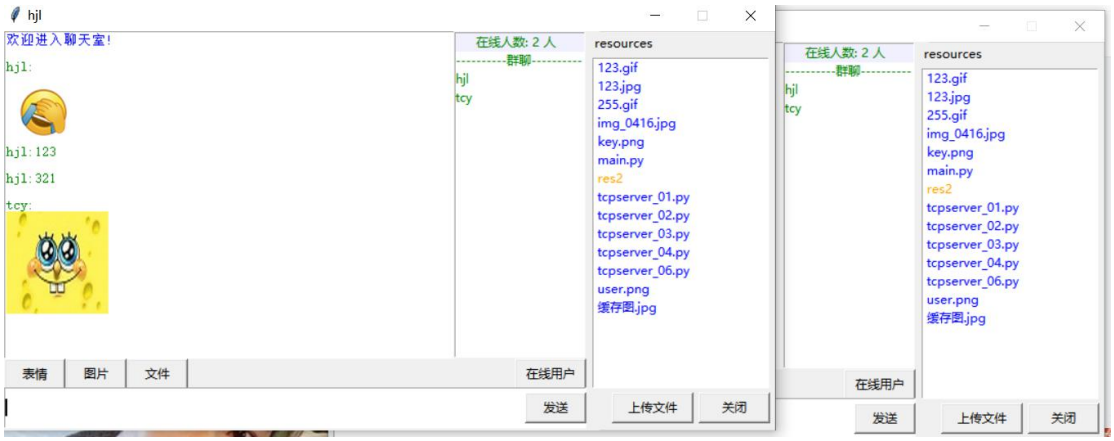
4. 点击表情按钮可以显示出系统中已有的表情并选择想要发送的表情，点击进行发送。

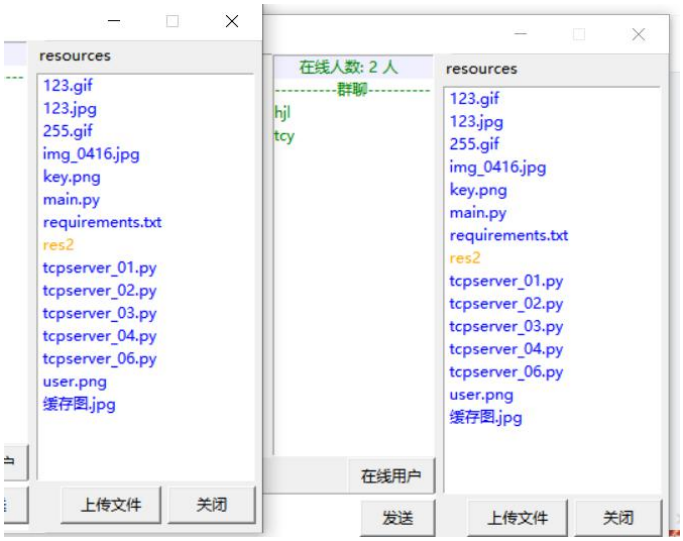
表情发送：



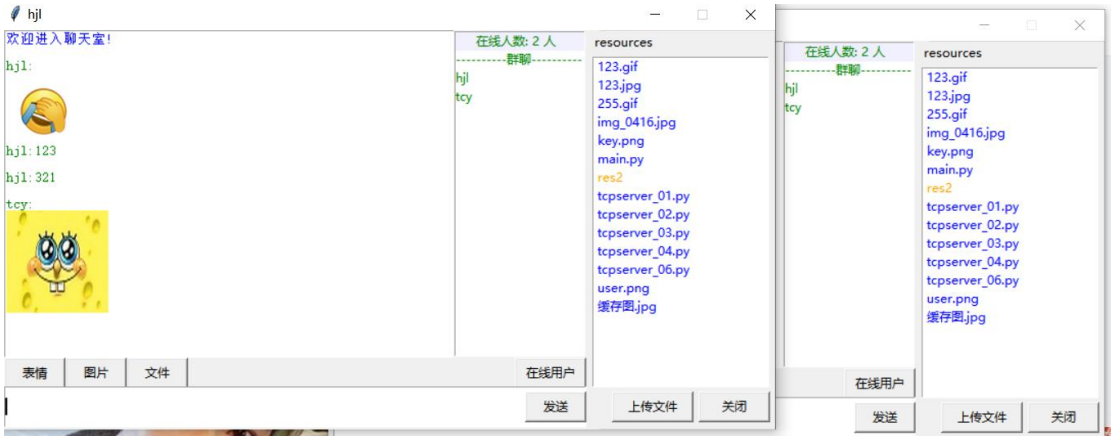
5.　传输与下载文件功能使用：用户点击文件按钮可以看到聊天室中所存在的文件资源，点击想要下载的文件可以将文件下载到本地计算机上，用户点击文件上传按钮也可以将本地想要上传的文件，上传至聊天室中。

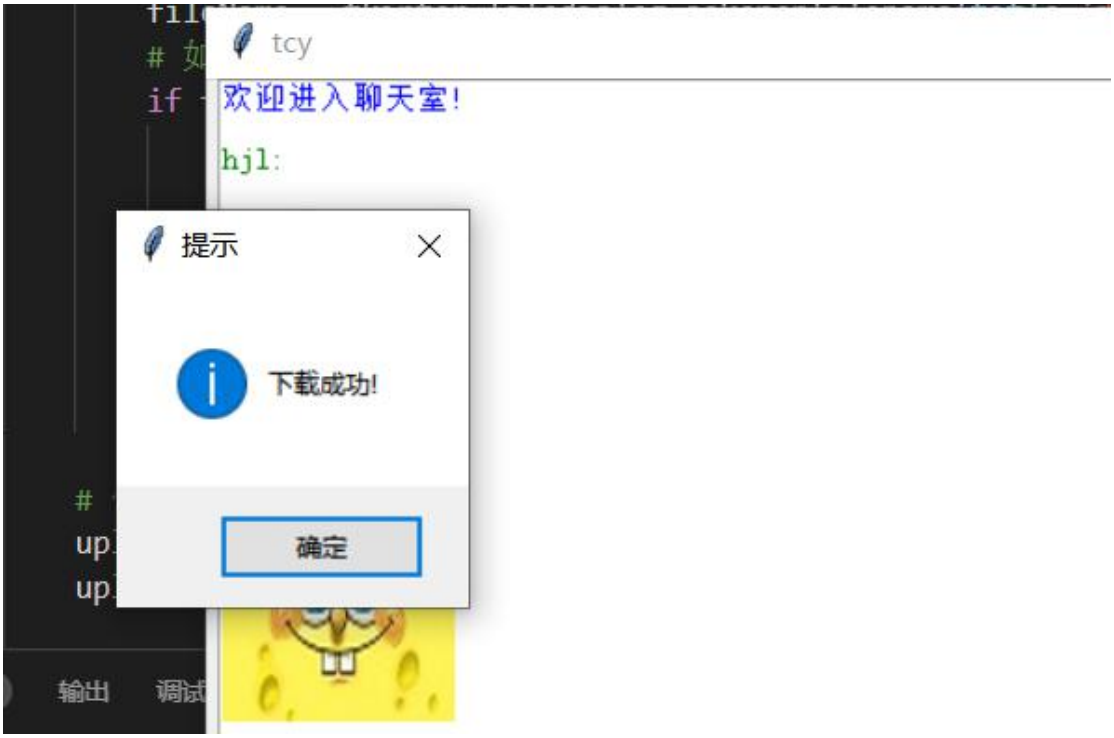
传输文件（requirements.txt）：


hjl
欢迎进入聊天室!
hjl:
hjl: 123
hjl: 321
tcy:
在线人数: 2 人
群聊
hjl
tcy
resources
123.gif
123.jpg
255.gif
img_0416.jpg
key.png
main.py
res2
tcpserver_01.py
tcpserver_02.py
tcpserver_03.py
tcpserver_04.py
tcpserver_06.py
user.png
缓存图.jpg
表情
图片
文件
在线用户
发送
上传文件
关闭


resources
123.gif
123.jpg
255.gif
img_0416.jpg
key.png
main.py
requirements.txt
res2
tcpserver_01.py
tcpserver_02.py
tcpserver_03.py
tcpserver_04.py
tcpserver_06.py
user.png
缓存图.jpg
在线人数: 2 人
群聊
hjl
tcy
在线用户
发送
上传文件
关闭

下载文件


hjl
欢迎进入聊天室!
hjl:
hjl: 123
hjl: 321
tcy:
在线人数: 2 人
群聊
hjl
tcy
resources
123.gif
123.jpg
255.gif
img_0416.jpg
key.png
main.py
res2
tcpserver_01.py
tcpserver_02.py
tcpserver_03.py
tcpserver_04.py
tcpserver_06.py
user.png
缓存图.jpg
表情
图片
文件
在线用户
发送
上传文件
关闭



五、开发过程中遇到的问题

1. 首先遇到了发送图片消息时发送端发送了图片，但接收端未显示在窗口上。用户 1 发送了图片，但在用户 2 的聊天室中无法显示。

解决办法：

上述问题，我发现是由于每个用户的接收端文件夹是共享的，也就是说用户 1 与用户 2 同用了一个文件夹产生了冲突，每个用户应该独立文件夹。随后更改了相应的用户文件夹目录，问题就解决了。

2. 在编写 client 端时用了 from socket import *语句，随后出现了报错。

解决方法：from socket import *它引用的 socket 不一定就是我想要的 socket 文件，用 import socket 就解决了，模块和函数是重名的。

3. 我在测试发送两条连续图片时，最终会拼接成一条进行发送，并不是分开显示日期时间，造成了黏包问题。

解决方法：设定接收数据大小，发送端每次发送需要发送的数据的数据大小，接收端只接收确定的大小问题得以解决。

4. 在进行测试中，遇到了聊天通信卡顿，未响应，图片发送不成功等情况。

解决方法：登录时的用户名不能存在特殊字符，不然会被默认当作文件路径，发送图片时会产生冲突。