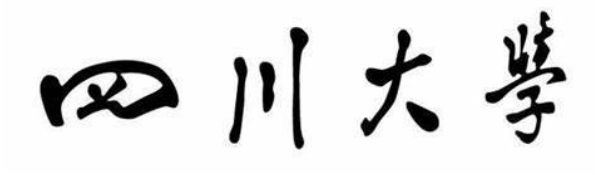


计算机网络与分布式系统

开发过程文档

选　　题　即时通讯系统-ChatRoom

姓　　名　徐晓龙

学　　号　2018141424178

年　　级　2019

指导老师　宋万忠

1.说明

1.1. 项目名称

即使通讯系统

1.2 项目需求

1） 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。

2） 通讯系统至少可以满足三个客户端同时登录、通讯的需求。

3） 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、端口号与主机名写进 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

用户在输入账号后，客户端向服务端发送请求完成以下功能：点击登录，如果账号和密码都正确则进入主界面，如果其中有一项错误则直接弹出提示框，提示错误。

2.2.2 注册功能

注册功能允许用户通过网页注册相应的账号，注册时可以对个人基本信息进行填写，注册成功后将用户信息添加进 user 库中。

2.2.3 文字聊天功能

用户进入聊天室后左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，还可以清空输入栏中的信息。

2.2.4 图片聊天功能

用户进入聊天室后左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，大小最多为 5MB。发送的消息会显示消息的日期与时间。在有成员下线后，左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。在用户输入账号和密码信息后，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。

系统注册界面如下图所示

登录实现源码如下图所示

```
def login():
    print("点击登录按钮")
    user, key = login_frame.get_input()
    # 密码传md5
    key = MD5.gen_md5(key)
    if user == "" or key == "":
        messagebox.showwarning(title="提示", message="用户名或者密码为空")
        return
    print("user: " + user + ", key: " + key)
    if client.check_user(user, key):
        # 验证成功
        goto_main_frame(user)
    else:
        # 验证失败
        messagebox.showerror(title="错误", message="用户名或者密码错误")
```

注册实现方法如下图

```
# 登陆界面前往注册界面
def register():
    print("点击注册按钮")
    login_frame.close()
    global reg_frame
    reg_frame = RegisterPanel(close_reg_window, register_submit, close_reg_window)
    reg_frame.show()


# 提交注册表单
def register_submit():
    print("开始注册")
    user, key, confirm = reg_frame.get_input()
    if user == "" or key == "" or confirm == "":
        messagebox.showwarning("错误", "请完成注册表单")
        return
    if not key == confirm:
        messagebox.showwarning("错误", "两次密码输入不一致")
        return
```

```
    # 发送注册请求
    result = client.register_user(user, MD5.gen_md5(key))
    if result == "0":
        # 注册成功，跳往登陆界面
        messagebox.showinfo("成功", "注册成功")
        close_reg_window()
    elif result == "1":
        # 用户名重复
        messagebox.showerror("错误", "该用户名已被注册")
    elif result == "2":
```

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

```
# 处理消息发送请求
def handle_message(_conn, addr):
    content = recv_all_string(_conn)
    # 发送给所有在线客户端
    for c in online_conn:
        # 先发一个字符串告诉客户端接下来是消息
        send_string_with_length(c, "#!message#!")
        send_string_with_length(c, conn2user[_conn])
        send_string_with_length(c, content)

    return True
```

客户端发送消息

```
# 发送消息
def send_message(self, message):
    self.sk.sendall(bytes("3", "utf-8"))
    self.send_string_with_length(message)
```

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。

实现：服务器端处理图片请求如下所示：

```
# 处理图片发送请求
def handle_image(_conn, addr):
    filename = recv_all_string(_conn)
    filepath = os.path.join("upload", filename)
    print("handle_image.filename", filename)
    recv_file(_conn, filename)
    # 发送给所有在线客户端
    for c in online_conn:
        # 先发一个字符串告诉客户端接下来是消息
        send_string_with_length(c, "#!image#!")
        # 发送用户名
        send_string_with_length(c, conn2user[_conn])
        # 发送文件名
        send_string_with_length(c, filename)
        # 发送文件
        send_file_with_length(c, filepath)
    return True
```

客户端发送图片请求如下：

```
# 发送图片
def send_image(self, filepath):
    self.sk.sendall(bytes("5", "utf-8"))

    (file_path, filename) = os.path.split(filepath)
    self.send_string_with_length(filename)
    self.send_file_with_length(filepath)
```

聊天室界面对于图片消息的处理:

```
            main_frame.recv_message(user, content)
        # 获取显示图片
        elif _type == "#!image#!":
            print("获取新图片消息")
            user = client.recv_all_string()
            print("user: " + user)
            filename = client.recv_all_string()
            print("filename: " + filename)
            filepath = client.recv_file(filename)
            main_frame.recv_images(user, filepath)
```

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('config.ini')
5  IP = cf.get('ip', 'chat_server_ip')
6  PORT = int(cf.get('port', 'chat_server_port'))
7
```

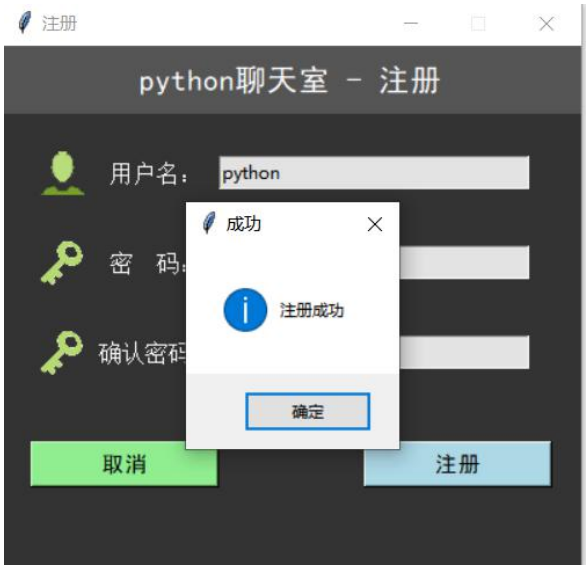
Ip 地址以及端口号的配置文件如下图:

```
1  import configparser
2
3  cf: ConfigParser = ConfigParser()
4  cf.read('config.ini')
5  IP = cf.get('ip', 'chat_server_ip')
6  PORT = int(cf.get('port', 'chat_server_port'))
7
```

4.项目展示

1. 运行 sever.py 文件启动服务器，开始监听。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，可以选择进行注册还是登录。



目前已注册的用户有：

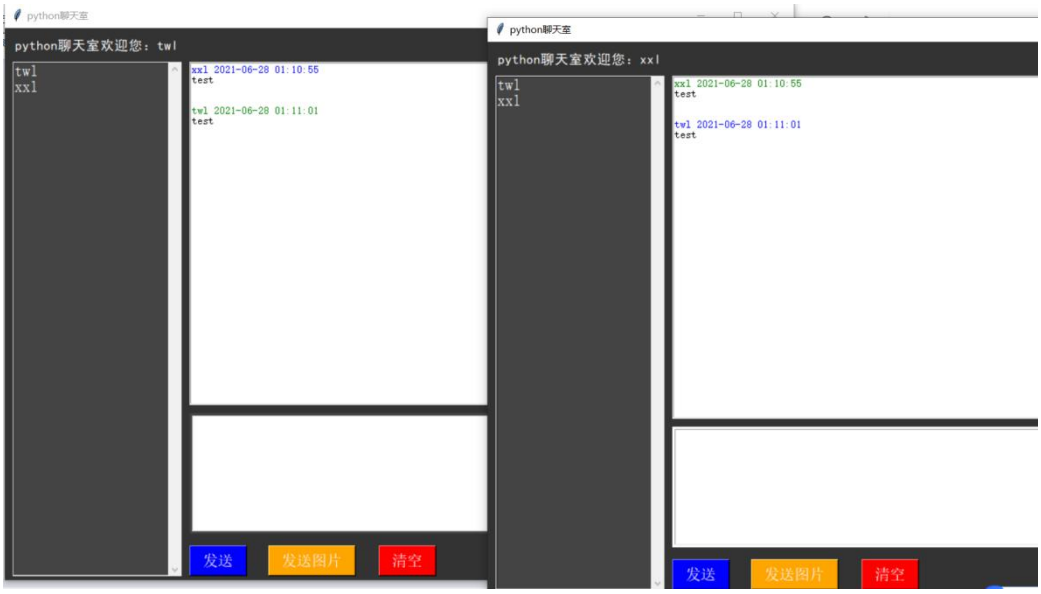
User：xxl　password：123　　user：twl　password：123

User：python　password：123

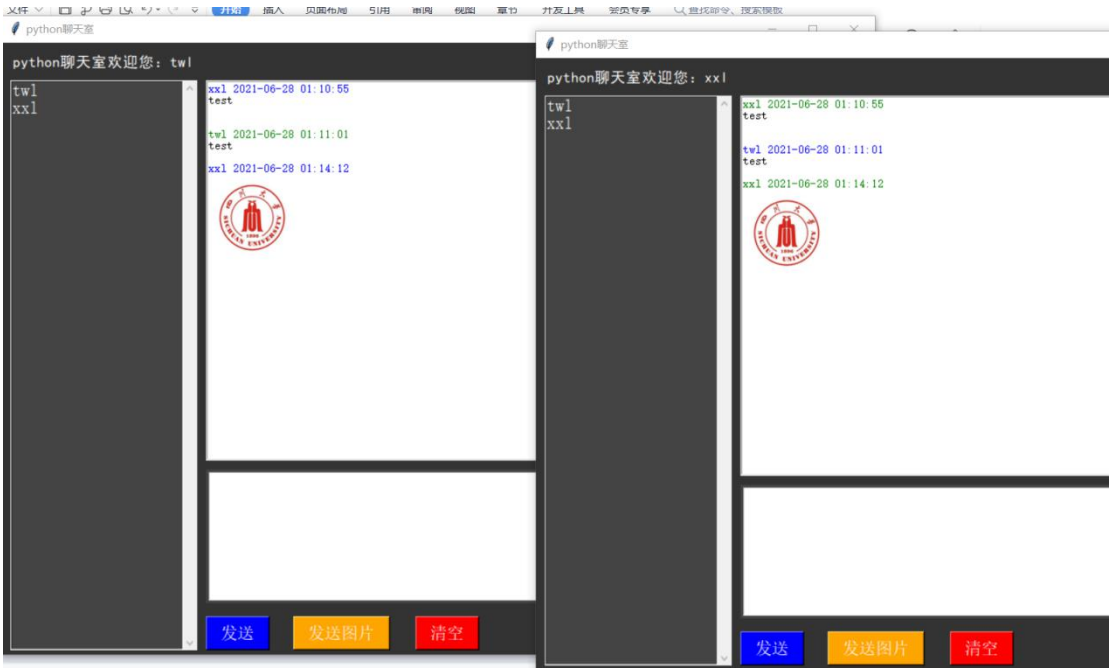
3.　用户进入聊天室后左侧会显示在线的用户，用户可以在聊天室内进行文本消息与图片的即使通讯，图片支持 jpg，gif，png 等格式，大小最多为 5MB。发送的消息会显示消息的日期与时间。在有成员下线后，左侧列表的在线用户也会相应更新。

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。

文本消息发送：



图片发送 jpg 格式测试：



图片发送（png 格式）：

4. 两台主机都可以发送消息，并且 socket 通信连接稳定无中断，还有对于异常输入的处理，对于不支持的图片类型和大小会进行报错，点击清空按钮可以将输入在发送框中的消息清空。

图片类型异常处理：

5. 发送的图片文件会缓存在服务器和用户独立的文件夹中，方便用户可以查看

五、开发过程中遇到的问题及解决方法

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. 在进行 socket 通信时，一段时间后客户端无法进行正常的通讯，用户 1 发送了消息，用户 2 未能正常接收。

解决方法：

资源耗尽，运行时占用了随即分配的随机本地端口，而我一直没有关闭，导致所有的 port 都被占用，在我重新分配的时候就得到了有效的端口了，要 closesocket。

4. 同一网络内，设定了静态 IP 的两台 PC 之间，可以互相 ping，但是 socket 连接的时候，总是报错 WinError10061。报错的代码是客户端：s.connect((host, port))。

解决方法：

尝试了关闭防火墙，发现没有解决，随后修改了代码：把 host = socket.gethostname() 改成 host = ‘服务器的静态 IP’，问题解决。

5. 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。

6. 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。