**SOCKET编程实验报告——简单的聊天系统**

**摘要**

首先简要介绍实现的聊天系统的文件组织，接着从服务器和用户程序两个部分介绍聊天系统的具体设计，然后介绍相关的错误处理，最后说明该聊天系统还可继续改进的方面。

**一、聊天系统简介**

我选择的是参考题目四，简单的聊天系统，使用Python3编程，该聊天系统主要分为服务器和用户程序两个部分：

服务器部分包含：

server.py文件，该文件负责对接入的用户进行响应和通知操作。

用户程序部分包含：

login.py文件，该文件实现了用户登录的界面；

main.py文件，该文件表现了用户登录聊天系统的主界面，用户可以在主界面使用

聊天系统的各项功能。

chatting.py文件，该文件表现了用户聊天时的窗口，用户在该窗口下进行聊天操作。

**二、服务器部分**

由于服务器需要不断地对客户端的连接进行监听，并同时对每个客户端的连接的消息进行接收和处理，所以一个线程是不够的，这里用到了多线程库threading。

**多线程处理**：

服务器的主进程会不断地接收客户端的连接请求，然后服务器为每个连接开辟一个对应的线程来处理该连接的消息。

每个socket连接对应的线程会不断地尝试接收客户端发送来的消息，然后将接收到的消息转发给该消息的所有目的客户端。

服务器如何转发该消息给其对应的所有目的客户端呢？首先，服务器为相互通信的各客户端建立一个列表，当该列表中的某个客户端要与列表中的其他客户端通信时，服务器直接遍历该列表，将信息转发给列表中所有的客户端。

**协议的设计和解析**如下：

socket连接上传送的消息包含头部和数据两部分：头部有六个，分别表示数据长度(DATALEN)，日期(DATE)，客户端(HOST)，消息类型(TYPE)，代理(AGENT)，编码语言(LANGUAGE)。实际上，头部可以根据服务器的需要来灵活地调整，头部旨在为服务器的工作提高效率和安全性。这里，服务器根据头部中的数据长度，通过连接进一步获取该消息所包含的全部数据。

这里的数据并不只是连接上传送的信息。数据部分首先封装了传送信息的格式，即聊天会话的字体和颜色设置。字体设置包含字体和字的大小，颜色设置则用RGB的3个数字表示，字体和颜色设置最后都转换成字符串来传送。然后才是传送的信息。

**三、用户程序部分**

用户程序部分包含login.py，main.py，chatting.py三个文件，这三个文件的界面都是用python的库pyqt5来做的，login.py是用户登录界面，main.py是聊天程序主界面，chatting.py负责用户聊天窗口和用TCP传输数据。

login.py，**用户登录界面**：

登录界面需要用户输入账号，密码，然后才能登录，点击登录之后，界面将转到聊天程序主界面(main.py)。

使用pyqt做界面时，py文件的主线程会停留在UI的显示上，直到UI被关闭。主函数调用mywindow()类来创建UI界面。在mywindow()类中，先调用setupUi()函数来绘制界面，然后将界面中按钮的信号和响应函数连接起来。

登录界面如下：

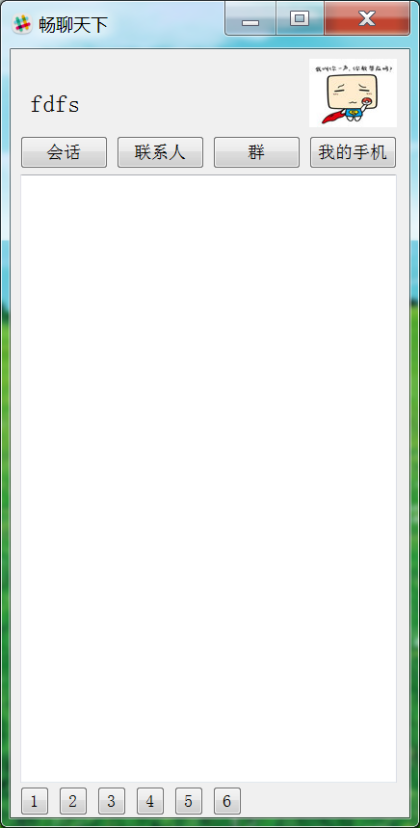


main.py，**聊天程序主界面**：

UI和事件的交互方式跟登录界面是相同的：主函数调用mywindow()类来创建UI界面。在mywindow()类中，先调用setupUi()函数来绘制界面，然后将界面中按钮的信号和响应函数连接起来。

主界面显示了会话，联系人，群，我的手机四个模块，这里只实现了会话模块，该模块可以进行多人聊天，另外三个模块可根据需要再另行扩展。在点击会话之后，界面将会转到用户聊天窗口(chatting.py)。

聊天程序主界面如下：



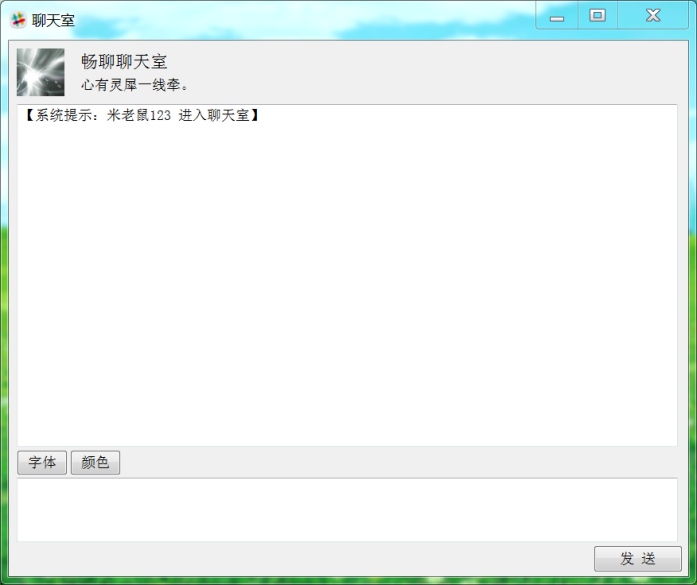
chatting.py，**用户聊天窗口**：

UI和事件的交互方式跟登录界面和主界面是相同的：主函数调用mywindow()类来创建UI界面。在mywindow()类中，先调用setupUi()函数来绘制界面，然后将界面中按钮的信号和响应函数连接起来。

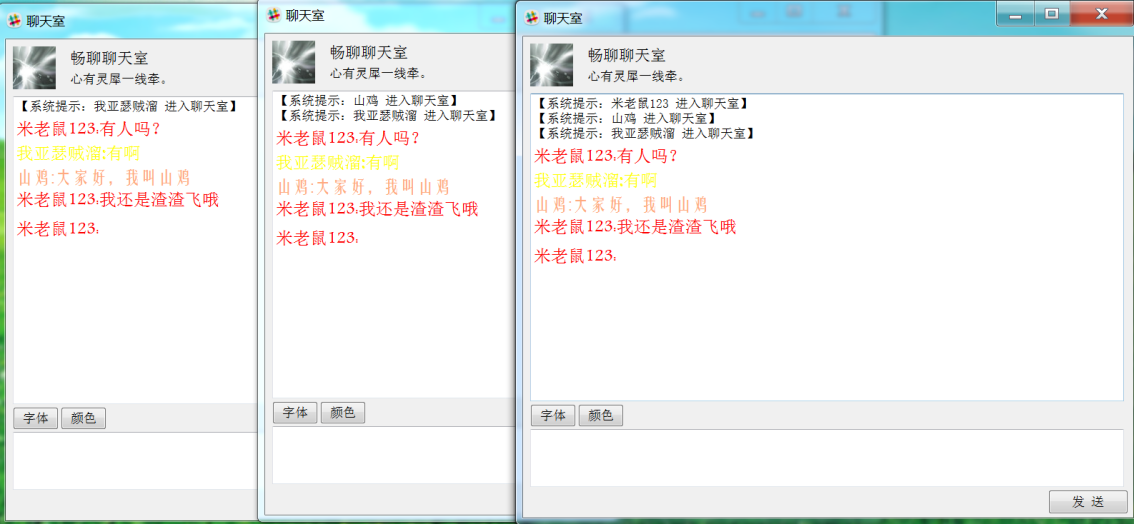
在chatting.py文件中，主线程会绘制UI界面，然后创建一个子线程来不断地接收服务器发送过来的消息。而在UI界面中，按钮信号有四个，分别是设置字体，设置字的颜色，发送消息和各种快捷键，其对应的三个响应函数是choicefont()，choicecolor()，sendmessage()函数。启动快捷键时，直接调用该快捷键对应的按钮的响应函数即可。

在发送消息时，首先收集各个头部字段合并成头部块，然后将字体和颜色设置和需要传送的信息合并成数据块，最后将头部块和数据块合并发送给服务器。

用户聊天窗口如下：



字体设置，颜色设置和多人在线聊天测试：



**四、错误处理**

在客户端连接上服务器之后，客户端会向服务器发送一个特定的字符串来表示已建立连接，此时服务器会检查建立连接后第一个收到的字符串是否是该特定的字符串，如果不是，表明该连接发生了错误，需要进行处理，这里只输出了错误，也可强制关闭该连接。

客户端向服务器发送和接受消息时，若服务器关闭了该连接，会产生ConnectionAbortedError错误，若服务器已关闭，会产生ConnectionResetError错误。这里仅仅打印错误发生的原因来提示用户，并不关闭连接或者关闭客户端。

还可能出现的问题：

1.数据传送过程中可能会出现不可抗力导致连接故障，可以建立一个缓冲区来保存最近的数据，防止数据丢失。

2.服务器只是一个py文件，稳定性还不够，可以给服务器配置数据库和本地文件来增加稳定性。

3.服务器可能遭受各种攻击，可以给服务器增加一些安全和防御模块。

4.服务器和客户端之间的通信可能被篡改，可以对连接进行加密和认证处理。

**五、聊天系统需要的改进**

1.可在各个界面上实现更多的功能。

2.可以增加对各类信息的支持，比如图片，文件，实时视频，VR等。

3.可以在消息传递时使用合适的头部以提高效率和安全性。