

计算机网络课程报告：socket编程

史一哲 17300290027

目录

- [任务介绍](#)
 - [开发环境](#)
 - [运行方法](#)
 - [功能简介](#)
 - [设计思路](#)
- [协议设计说明](#)
- [数据库设计](#)
- [加密算法简介](#)
- [功能展示](#)
 - [注册](#)
 - [登录](#)
 - [群聊](#)
 - [私聊](#)
 - [登出](#)
 - [错误处理](#)
- [文件结构](#)

任务介绍

开发环境

Windows 10 + VScode + Python 3.7 + Python包 (socket, PyQt5, threading, time, sys, crypto等)

其中图形化界面使用的 PyQt5

本实验通过：SERVER = socket(AF_INET, SOCK_STREAM) 使用TCP协议

运行方法

```
python server.py
python client\client.py
```

功能简介

本实验完成了socket聊天室的设计。具体功能包含：用户登录，好友登录提醒，群发消息，私发消息，用户登出，好友离开提醒。其中用户信息以及聊天信息使用轻量级数据库 `sqlite` 进行存储，这样不仅可以对用户登录/注册进行重复性以及正确性的验证，并且可以在用户登录后显示历史消息，使得用户体验更加友好，并且对用户密码使用 `Pycrypto` 进行 `RSA` 加密以确保用户登录的安全性。

由于我采用的是 `PyQt5` 实现图形化界面，其中的 `QTextBrowser` 可以较好的显示 `CSS` 样式的内容，所以我利用 `QTextEdit` 实现了富文本框，可以对聊天内容的字体样式、大小、粗体、颜色等进行设置，同时增加了下载聊天内容以及清空聊天内容的选项。

快捷键设置

快捷键	界面	功能
ENTER	登录界面	登录
CTRL + R	登录界面	进入注册界面
ENTER	注册界面	注册
CTRL + ENTER	聊天界面	发送消息
ESC	聊天界面	退出登录

设计思路

新用户首先需要进行注册，其中会把用户名、公钥以及签名存入数据库，然后输入用户名、密码进行登录。用户登录后与服务器建立链接，并向服务器端发送登录指令。服务器端接受到登录指令后，向其它所有用户发送系统消息-好友登录实现好友登录提醒功能。客户端向服务器端发送两种格式的消息实现群发功能与私发功能，服务器端解析消息，并确定发送对象，向目标客户端发送消息，完成群发与私发任务。用户登出时，向服务器端发送登出消息，服务器断开该客户端链接，并向其它客户端发送系统消息-用户离开。

协议设计

协议头部如下

1 客户端到服务器端：

- 1. {LOGIN} 登录指令
- 2. {ALL} 群发指令
- 3. {username} 私发指令 注：username 为变量，表示私发对象的用户名
- 4. {LOGOUT} 登出指令

2 服务器端到客户端：

- 1. {CLIENTS} 更新用户列表指令
- 2. {MESSAGE} 消息指令：包括群发消息/系统消息
- 3. {PRIVATE } + username 私发消息指令 注：username 为变量，表示私发对象的用户名
- 4. {NEW} 新用户加入聊天室
- 5. {LEFT} 用户离开聊天室

数据库设计

数据库在 socketdb.db 中，结构如下：

```
CREATE TABLE users(
    username TEXT PRIMARY KEY NOT NULL,
    public_key TEXT NOT NULL,
    signature TEXT NOT NULL
);

CREATE TABLE messages(
    username TEXT NOT NULL,
    date TEXT NOT NULL,
    message TEXT NOT NULL,
    type TEXT NOT NULL
);
```

`TEXT` 就是字符串类型，为了保证登录的安全性，没有把密码直接存入数据库，而是使用 `Pycrypto` 进行加密，`username` 是 `users` 的主键，确保没有相同名字的用户。

加密算法简介

加密算法位于 `secret.py`

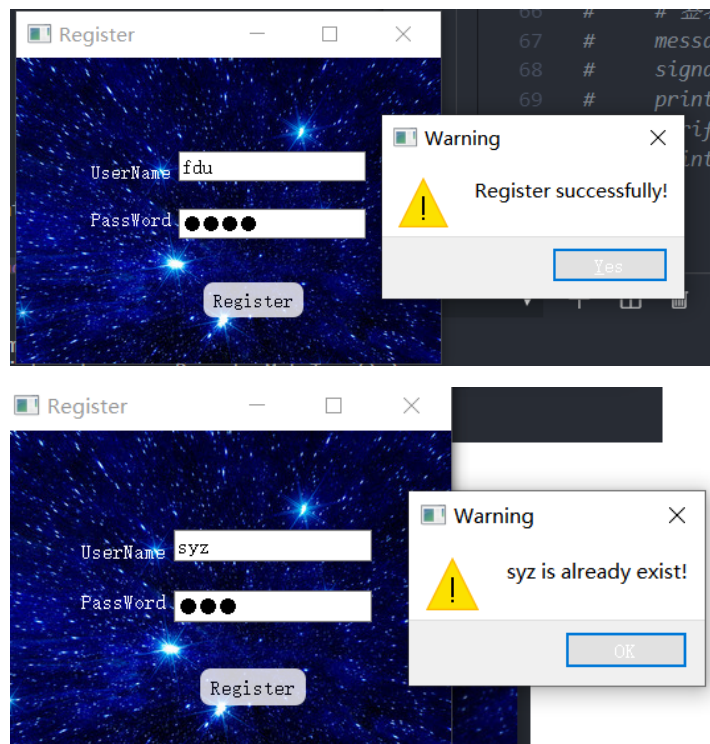
`RSA` 是一种公钥密码算法，`RSA` 的密文是对代码明文的数字的E次方求mod N的结果，也就是将明文和自己做E次乘法，然后再将其结果除以 N 求余数，余数就是密文。`RSA` 是一个简洁的加密算法。E 和 N 的组合就是公钥（public key）。

对于 `RSA` 的解密，即密文的数字的 D 次方求mod N 即可，即密文和自己做 D 次乘法，再对结果除以 N 求余数即可得到明文。D 和 N 的组合就是私钥（private key）。

算法的加密和解密还是很简单的，可是公钥和私钥的生成算法却不是随意的。本项目在于使用，对生成密钥对的算法就暂时忽略。

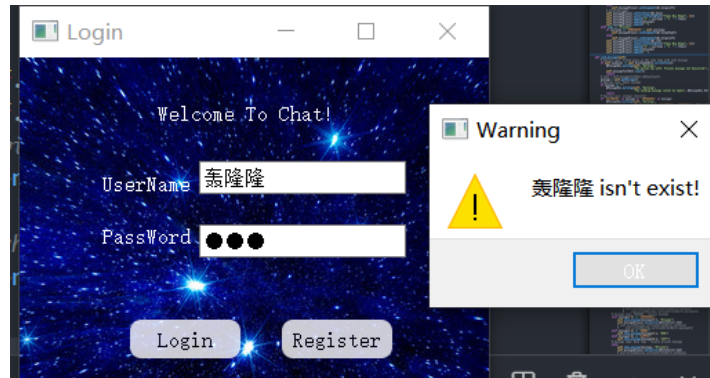
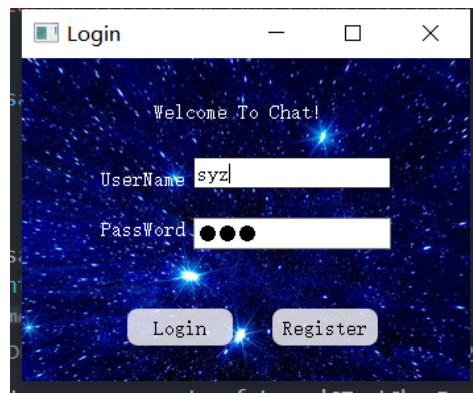
功能展示

注册



用户已存在的话会报错，注册成功之后会对密码进行加密存入数据库中。

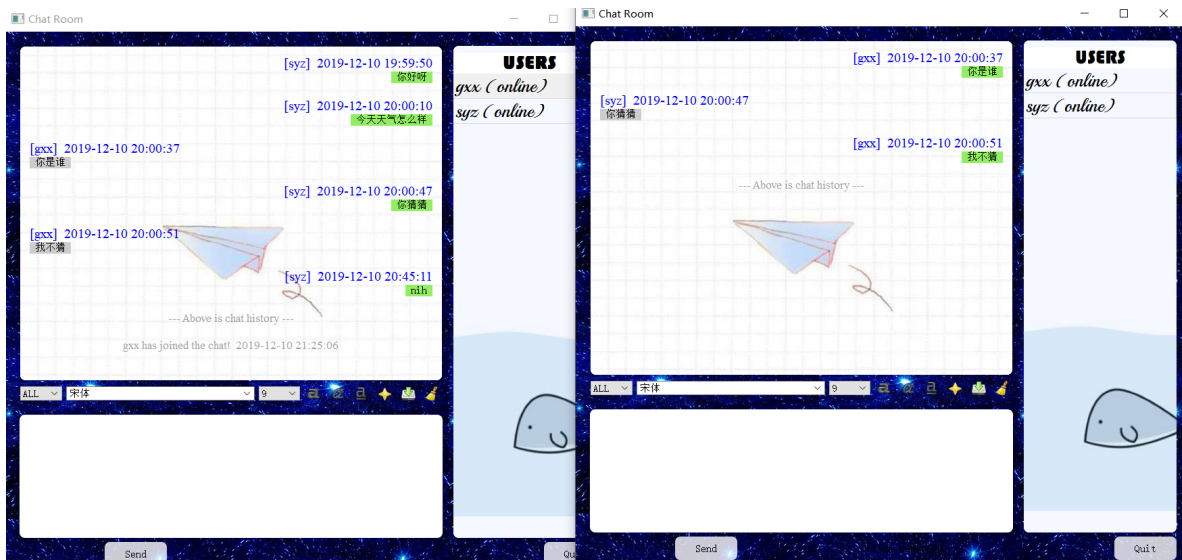
登录

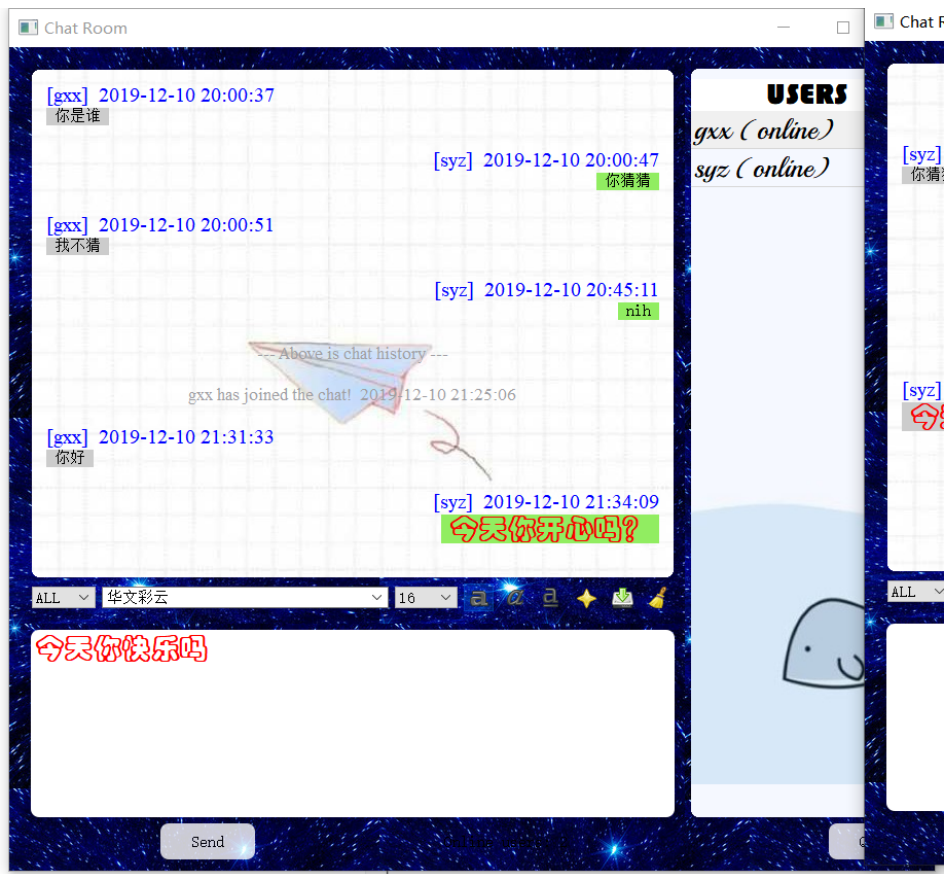


登录的时候会先判断用户是否已注册，用户不存在的话会报错，登录成功后会跳转到聊天界面

群聊

聊天界面会显示历史消息记录，成员的加入情况，右侧显示用户的在线状况，左下方是消息输入框，功能栏可以对字体样式、大小、颜色等进行修改。其中功能栏中的最左边的下拉菜单可以选择群聊或者私聊，默认是 `ALL` 即群聊。对于新登录的用户，服务器发送系统消息-欢迎 `{MESSAGE} + welcome username !`，对于已登录用户，系统放送系统消息-用户登录提醒 `{MEASAGE} + username has joined the chat!` 与 用户登录/登出消息 `{CLIENTS} + userlist`，实现好友登录提醒功能。同时，用户登录/登出消息会使得用户界面中朋友列表和发送列表根据 `userlist` 更新。





发送列表选择 `ALL`，输入发送内容，点击 `Send` 或者 `CTRL+ENTER`，通过向服务器端发送消息 `{ALL} + message`，实现消息发送功能第一阶段。服务器接收并解析这一消息后，向所有用户端发送消息 `{MESSAGE} + username + : + message`，实现消息发送功能第二阶段。用户端接收到消息后，解析，并在消息框中显示。

私聊

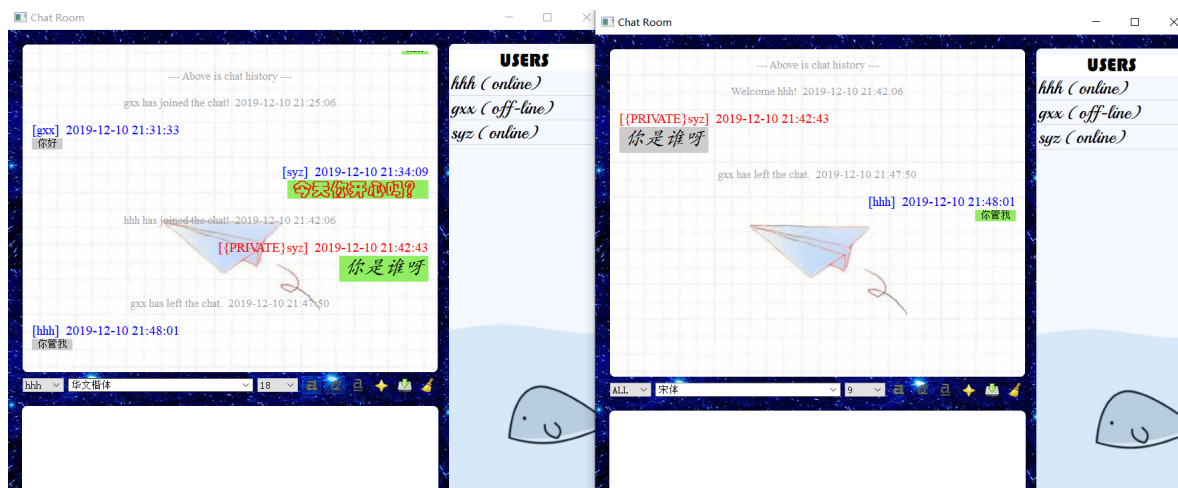
私聊需要在下拉框中选择要私聊的对象。



通过向服务器端发送消息 `{username} + message` 与消息 `{self.name} + message`，实现私聊消息发送功能第一阶段，其中后者是为了实现私聊消息在本地显示。服务器接收并解析这一消息后，解析 `username` 与 `self.name` 向 `username` 端与 `self.name` 端分别发送消息 `{PRIVATE} + self.name + : + message`，实现私聊消息发送功能第二阶段。用户端接收到消息后，解析，并在消息框中显示。

结果如图所示，用户 `syz` 私聊 `hhh`，所以只有 `hhh` 收到消息，而 `gxx` 收不到消息。其中为了进行区分，私聊消息日期和用户名显示为红色。

登出



用户登出前，会向服务器端发送消息 `{LOGOUT}`，服务器接收到该消息后，断开该客户端的链接。服务器接收到用户登出消息之后，会向其他所有用户发送系统消息 `{MESSAGE} + username + has left the chat`。用户端接收到消息后在消息框中显示用户离开消息，实现用户离开提醒功能。同时更新界面中朋友列表和发送列表。如果发送列表正好选择为该退出用户，则将发送列表设置为默认的 `ALL`，并且把登出用户的状态设置为 `off-line`

其中保存的聊天信息是从html中取出plainText，效果如下：

```
chat_records.odt
1  [syz] 2019-12-10 19:59:50
2  你好呀
3
4  [syz] 2019-12-10 20:00:10
5  今天天气怎么样
6
7  [gxx] 2019-12-10 20:00:37
8  你是谁
9
10 [syz] 2019-12-10 20:00:47
11 你猜猜
12
13 [gxx] 2019-12-10 20:00:51
14 我不猜
15
```

错误处理

服务器端解析用户端信息时，能够解析信息，但后续转发目标不再列表中，则发送提示信息——用户已离开。如果无法解析用户端信息，会发送提示信息——无法解析用户信息。同时对用户端发送消息的内容做了例如不能发送空消息等的处理。

文件结构

```
| chat_records.odt # 保存的聊天记录
| README.md
| server.py # 服务端代码
| socketdb.db # sqlite数据库
|
└─client
    | chat.py # 聊天功能代码
    | client.py # 登录+连接服务端+初始化数据库
```

```
| register # 注册
| secret # 加密算法
|
├─DataBase
|   __init__.py # 构建数据库代码
|
├─Ui_Form
|   images_rc.py # 图片资源
|   images.qrc # 图片资源目录
|   Ui_chat.py # 聊天界面
|   Ui_login.py # 登录界面
|   Ui_register.py # 注册界面
|
└─image # 图片
```