

**简易微信实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **学号** | **班级** | **负责工作** |
| **李渤涟** | **3018205377** | **腾讯一班** | **通信协议，搭建客户端，搭建服务器，完善图形界面，实验报告** |
| **高子龙** | **3018001663** | **腾讯一班** | **设计图形界面，实验报告** |

1. **简易微信系统的总体设计**
2. 服务器：

服务器负责管理数据库与处理接收到的指令。服务器在启动时将保存在硬盘上的数据库备份读入内存，开始等待接收指令。

对于每一个处理指令的周期，服务器首先按照相同的IP地址与端口号创建一个UDP套接字，并在对数据库的操作执行完毕后，向本条指令的发送方发送返回信息，最后关闭套接字。

针对数据库的操作可分为三层：其一是对数据表在内存层面上进行的数据项的获取、插入、删除以及判断存在性的底层操作；其二是针对微信的具体功能设计的对底层操作的封装，这一层使用存在性判断函数为底层操作规避了内存访问的潜在错误；其三是为具体指令服务，完成功能层面的数据库访问的服务器顶层。

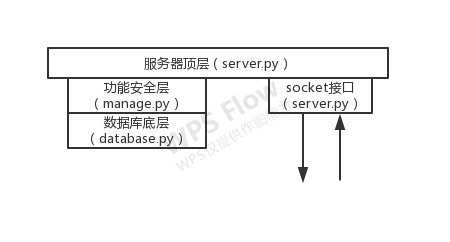


图1 服务器结构示意图

如图1所示，服务器顶层与socket接口均在server.py文件中实现，而功能安全层和数据库底层分别在manage.py和database.py文件中实现。

1. 客户端：

客户端是由多个由点击事件所触发的窗口所组成的。在启动客户端之初，首先将全部窗口初始化，并打开登录窗口。此外的全部窗口以及所有通信行为都是由现有窗口中的按钮点击所触发的。

在点击有关发送数据按钮时，首先根据所要求的功能和数据生成指令，然后创建UDP套接字，向已指定位置的服务器发送指令，接收返回信息并关闭UDP套接字，最后显示之或是根据之进行判断。

客户端的窗口都是系统窗口的实例，点击右上角的关闭按钮，就可以在任何时候强制地关闭窗口，当最后一个窗口关闭时，系统窗口的实例化进程结束，客户端程序结束。

1. **协议设计和系统设计**
2. 通信协议：

为规范客户端向服务器发送指令以请求服务的处理过程，定义如下指令集。（原文档见cmd\_format\_v6.txt）

regu\_账号\_\*\*\*\_密码\_名字\_\*\*\*：

要求服务器在数据库中添加一条用户信息，包括账号、密码和名字三部分。

lgin\_账号\_\*\*\*\_密码\_\*\*\*\_\*\*\*：

要求服务器比对（账号，密码）是否存在，若存在则返回‘1’，否则返回‘0’。

mkfa\_账号1\_账号2\_\*\*\*\_\*\*\*\_\*\*\*：

要求服务器在通知列表中添加一条信息，表示账号1向账号2发送了好友请求。

mkf\_账号1\_账号2\_\*\*\*\_\*\*\*\_\*\*\*：

要求服务器在好友列表中添加（账号1，账号2）和（账号2，账号1）两条信息。

dlf\_账号1\_账号2\_\*\*\*\_\*\*\*\_\*\*\*：

要求服务器在好友列表中删除（账号1，账号2）和（账号2，账号1）两条信息。

sdm\_账号1\_账号2\_\*\*\*\_\*\*\*\_文字信息：

要求服务器在消息列表中添加一条信息，表示账号1向账号2发送了文字信息，并设置其时间戳为当前时间。

swm\_账号\_\*\*\*\_\*\*\*\_\*\*\*\_\*\*\*：

要求服务器将消息列表中所有时间戳距现在不超过30分钟的数据项返回。

swf\_账号\_\*\*\*\_\*\*\*\_\*\*\*\_\*\*\*：

要求服务器将好友列表中所有左项是账号的数据项返回。

swfa\_账号\_\*\*\*\_\*\*\*\_\*\*\*\_\*\*\*：

要求服务器将通知列表中所有左项是账号的数据项返回。

1. 系统设计：

在数据库底层，定义了获取用户列表数据项、获取好友列表数据项、获取消息列表数据项、获取通知列表数据项、插入用户列表、插入好友列表、插入消息列表、插入通知列表、删除用户列表数据项、删除好友列表数据项、删除通知列表数据项、判断指定用户列表数据项是否存在以及判断指定好友列表数据项是否存在的函数。（见database.py）

在功能安全层，定义了注册、登录、添加好友、删除好友、显示好友、发送消息、发送通知、获取消息以及获取通知的函数。（见manage.py）

在服务器顶层，设置一个循环，循环的每一次迭代是指令处理的一个周期，并设置每经历20个周期将数据库备份一次。（见server.py）

在客户端中，将每一个窗口定义一个类，并分离类中的构造函数与启动函数（见GUI.py），早客户端启动时运行全部实例的构造函数，但只在特定点击事件发生时才运行实例的启动函数。（见client.py）

1. **协议和系统实现**
2. 协议实现：

在server.py文件中，服务器首先获取指令的第0字段，并据此进入不同的程序分支，调用功能安全层的函数并处理数据，然后将结果添加于返回值变量，最后由UDP套接字发送。

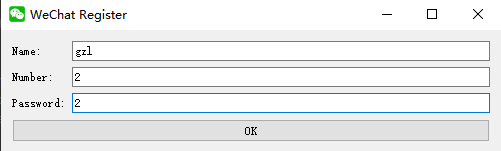
1. 系统实现：

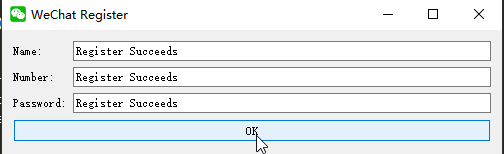
在系统正常运行的过程中，多个客户端可同时保持运行状态，每一个客户端的信息都来自客户端向服务器发送的指令的回复信息，如好友列表和消息列表。在服务器端，服务器根据指令到达的先后顺序依次处理指令所要求的操作，并返回数据。

由于服务器的指令处理流程十分扁平化，加之使用UDP协议可以减少一部分时间开销，在实验室条件下完全没有必要担心指令发送冲突，因为服务器有能力在极短的时间内区分出两条几乎同时发送的指令。

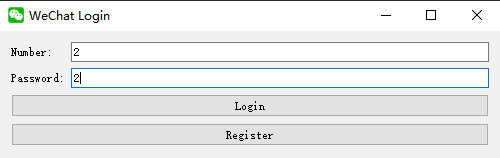
如此一来，本系统就使用了单个线程实现了多个用户的异步通信。

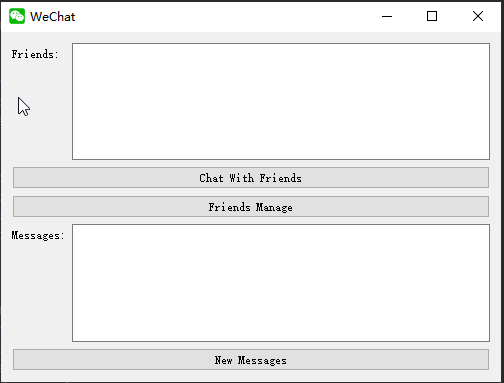
1. **实现结果**
2. 注册：

****

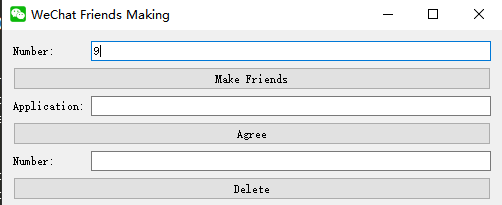
****

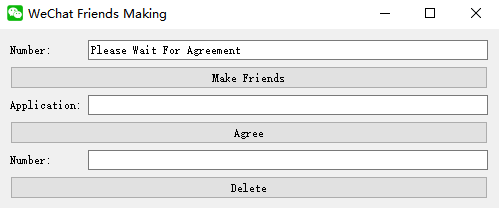
1. 登录：

****

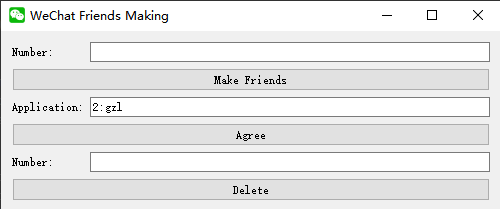
****

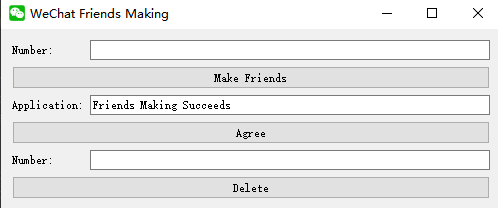
1. 发出好友申请：

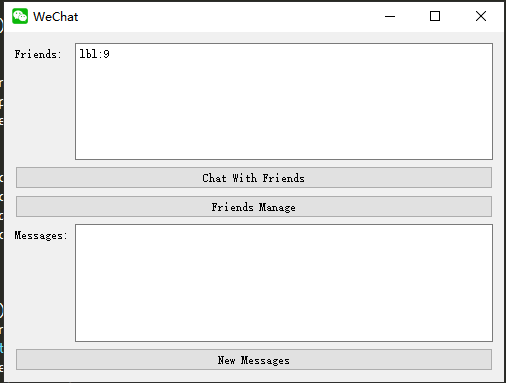
****

****

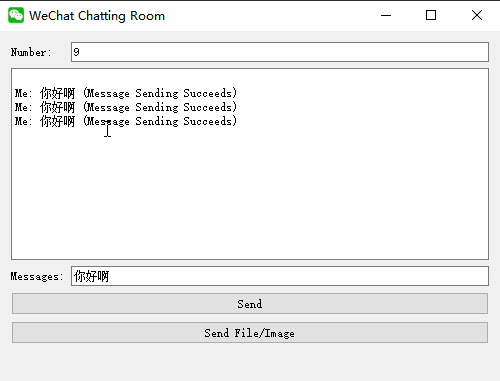
1. 同意添加好友：

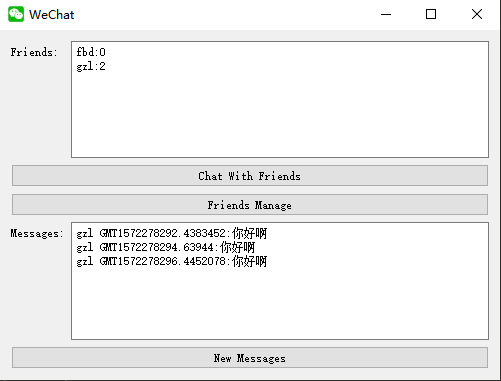
****

****

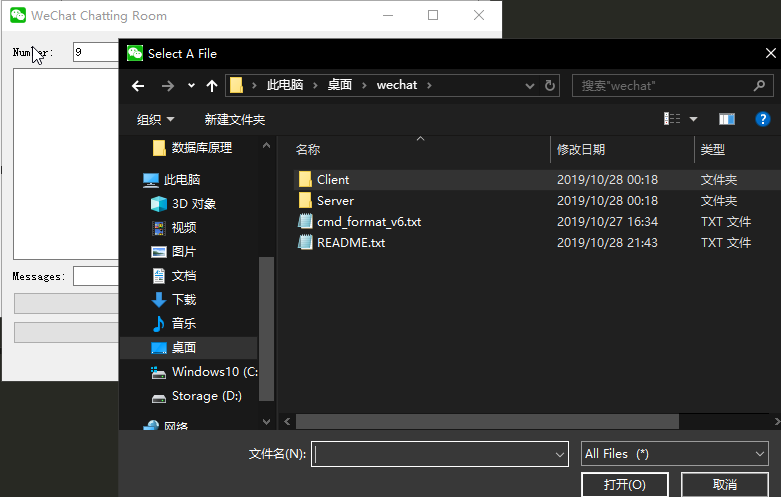
****

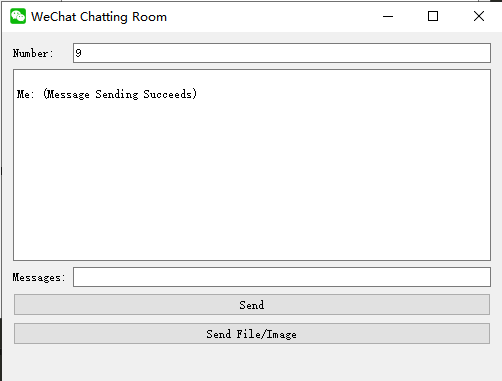
1. 发送消息：

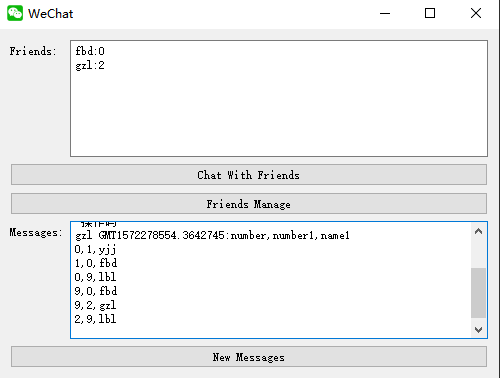
****

****

1. 发送文件或图片：

****

****

****

1. **附录**

本系统的所有代码都在压缩包3018205377-李渤涟-wechat.zip中，其根目录下有运行说明README.txt、指令文档cmd\_format\_v6.txt、文件夹Server和文件夹Client。

如需验证本系统的正确性，请参照README.txt进行操作。更多信息见https://github.com/lblaoke/simple-wechat