基于UDP SOCKET实现一个简单的QQ客户端

# 1、理解协议

QQ作为一个应用层的程序，其实现即时通信。本次实习要求同学们实现一个基于服务器转发的客户端程序。服务器程序老师已经编写并部署完成，具体如下：

服务器的IP地址：202.114.196.97

端口：21568

要实现的的核心功能，就是与服务器进行通信。并通过服务器作为中转，来实现与其他好友收发消息。通信。是程序，实现双向聊天功能，及客户端和服务器都能收发信息。但此时是有序收发。

为了理解代码，同学们可以修改老师上课PPT中给的UDP双向无限循环通信代码（仅修改IP和端口应该就可以了）。接下来根据协议，与服务器进行各种通信。请做以下尝试：

1）用我们帮你建立的帐户登陆

2）修改密码

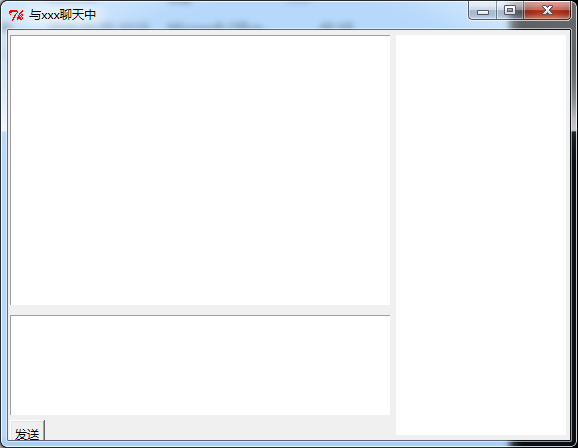
3）与其他同学通信等。

要求：把整个协议都试一次。包括各种出错信息，并写出测试文档！

理念1：专业程序员写程序，都不用MOUSE，只用键盘。正如专业打字员，打字不看键盘，不看屏幕，只看文稿。此部分设计的事实上是只有专业程序员才会使用的QQ。

# 2、图形用户界面（GUI）的聊天程序

## 2.1、图形用户界面（GUI）

阅读并理解本次大作业提供的“QQ聊天界面参考”，该代码是基于PYTHON的TK库实现的。其样式如图：

如果有必要，对此代码做极小的修改（也可能不修改就可以），使其实现与第一部分同样的功能。并重做之前的协议试验。

## 2.2基于面向对象的图形用户界面（GUI）聊天程序

思路：修改2.1的代码为面向对象的形式。

设计一个类，该类中包含二个方法（函数），如下：

1）发送消息的方法（函数）：可以直接把之前的代码复制过来

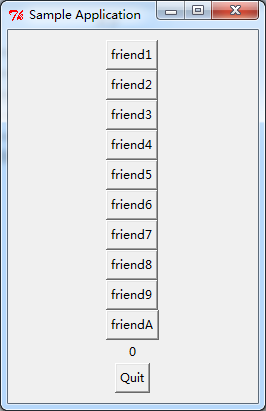
2）初始化的方法（函数）：实现图形界面的布局，以及ip、端口号的设置。

注意：如果代码调试出错，在相关的一些地方加上SELF。SELF的意思是类自己了，就是调用类自己的数据。

# 3、主界面程序

## 3.1、简单的主界面程序

阅读并理解本次大作业提供的“QQ主界面编写参考”，该代码是基于PYTHON的TK库实现的。其样式如图：



要求实现下面功能：

1）点击其中一个好友（friend?）按钮，即开启一个新的聊天窗口，并对该窗口进行若干初始设置。

2）进一步修改之前编写的“面向对象聊天代码”，使得其中仅显示正常的聊天数据，而在窗口标题显示与某个好友如七号好友（friend7）聊天中。并仅在网络出错时，给出相关的提示（如对方已离线等）

提示：在这一步中，可以在聊天窗口的对象中，直接发送UDP消息。

## 3.2、高级的主界面程序

要求修改上一节的代码，进一步实现。QQ仅通过主界面与服务器通信。每个聊天窗口不对外通信。

1）主界面作为信息传输的代理，将每个聊天窗口的信息发送到服务器，并将从服务器接收到的信息转发给合适的聊天窗口。

2）主界面与聊天窗口界面之间的信息交换，可用网络通信实现，也可以用其它方式实现（如果大家会其他方式的话）。

3）主界面在作用上：即是服务器，也是客户机。其作为服务器与各个聊天窗口通信，其作为客户机，与老师在公网上提供的服务器进行通信。

# 4、其他（暂时可以不看）

**4.1、QQ登陆界面**



上面这个是QQ登陆，可以仿这个做一个界面，并实现其中的逻辑：

1. 错误，清空输入并提示错误，用户重新输入
2. 正确，登陆成功了，就新启一个聊天应用。然后关闭当前登录应用。

进一步可以做一个注册页面。

**4.2、GUI+多线程的全双向聊天程序**

其实这个也是很简单的，就是将原来的接收数据模块分离出来，做成一个专门的接收方法。然后，再写一个启动多线程的代码，就是一句话，启一个新的线程，而这个线程指明是调用上面的接收程序。最后在主程序中，加一句话，来调用上面写的类中的那个启动调用多线程的方法。

在python 中，可以通过thread模块中的start\_new\_thread(function,args[])函数来启动一个新线程，其中function参数是你将要调用的线程函数，args[]是传递给你的线程函数的参数。eg:

import thread

def Func1():

print('hello')

def Func2():

thread.start\_new\_thread(Func1, ()) //启用单个线程

if \_\_name\_\_=='\_\_main\_\_':

Func2()

这里需要注意的是：其实也没有什么太多的东西了，就是用UDP，可以起一个服务端来，接收服务器发送的消息，也就是要绑定一个新的端口。

理念2：从小到大一步步构建。

# 附录：协议及其说明

## FU1用户注册

仿照QQ的注册页面，进行注册功能的实现(此处规定，仅进行账号、密码和确认密码的填写，**账号统一使用学号**)，在用户完成信息填写后，将所填信息以

01#账号#密码#确认密码# (注：此设计中，使用#作为分隔符，所以帐号，密码中都不可含有#字符)

的消息形式发送至服务器，服务器端确认所填信息无误后完成注册，并返回注册的回馈消息:

1. 用户注册成功：01:01
2. 确认密码错误：01:02
3. 用户已存在： 01:03

客户端需根据服务器端的反馈消息进行客户端相关功能的实现，如收到确认密码错误/用户已存在的消息时在客户端界面显示错误信息，提醒用户重新输入；在收到用户注册成功的消息时，客户端跳转到登陆页面(也可不自动跳转)。



## FU2 用户登录

登录指令的格式如下：

02#账号#密码#

按上面格式把相关信息发送至服务器后，服务器端进行密码验证，并返回如下形式的回应信息：

1. 登陆成功：02:01
2. 密码错误：02:02
3. 用户不存在：02:03
4. 用户已登录：02:04

客户端根据服务器返回的信息，进行客户端登陆信息的错误显示以及正确填写后的页面跳转。

## FU3与多好友进行单窗口聊天

添加一个好友，基于作业一的聊天窗口与单个好友进行聊天:

1. 消息的发送：在向好友发送消息后，客户端需将所发消息以 03#账号1(接收者)#消息# 的形式发送至服务器，服务器端对消息进行转发，并对转发的消息是否发送成功进行消息反馈：
2. 发送成功：03:01
3. 对方离线：03:02
4. 指定好友未读消息的接收：客户端需每隔30秒向服务器发送消息询问某个好友所发的未读消息条数，所发消息协议为04#账号1(发送者)#(该步骤中的发送者账号均为指定的好友账号)，服务器会将未读条数以 04:条数的形式进行消息的反馈，若获取条数非0，客户端需再向服务器端发送05#账号1(发送者)#的消息进行指定好友所发送的单条未读消息的接收。若想接收好友全部未读消息，可不断进行该消息的发送。服务器端会将未读信息以

05:账号1(发送者)：发送时间:消息的消息形式发送至客户端，客户端根据消息进行聊天窗口消息的显示。若无未读消息，服务器端将向客户端发送05:01的消息。

1. 所有未读消息的接收：客户端需每隔30秒向服务器发送消息询问未读消息条数，所发消息协议为 08#，服务器会将所有未读消息条数以 08:条数的形式进行消息的反馈，若获取条数非0，客户端需再向服务器端发送09#的消息进行单条未读消息的接收。若想接收全部未读消息，可不断进行该消息的发送。服务器端会将未读信息以

09:账号1(发送者)：发送时间:消息的消息形式发送至客户端，客户端根据消息进行聊天窗口消息的显示。若无未读消息，服务器端将向客户端发送09:01的消息。

*注：2)，3)步骤的区别在于,2)中的未读消息只来自于单个好友，3)中的未读消息来自于多个好友。*

## FU4与多个好友进行聊天

在3功能完成的基础上，添加多个好友，每个好友拥有独立的聊天窗口，在切换不同好友聊天时，客户端需获取当前好友的账号，并以3中所说的消息形式向服务器端发送消息。同样的，在接受消息时，客户端需要根据服务器端发送的消息判断不同好友发送的消息，并且将每个好友发送的消息分别显示在与该好友的聊天窗口中。

1. 离线

用户离线时，客户端需向服务器端发送离线消息，消息形式为

06# , 服务器接收到消息后，进行消息的反馈：

1. 离线成功：06:01
2. 已处于离线状态：06:02

并关闭与客户端的通信。

**注：**

1. **当客户端发送的消息格式错误时，服务器均以00:00的消息进行反馈。**
2. **当客户端未登陆就进行3,4,5操作时，会被认定为越权操作，服务器均以07:01的消息进行反馈，客户端需向用户进行提醒。**

## FU5协议小结

|  |  |  |
| --- | --- | --- |
| **协议动作** | **发送、返** | **格式** |
| 注册 | 发 | 01#账号#密码#确认密码# |
|  | 返回值 | 成功：01:01  确认密码错误：01:02  用户已存在： 01:03 |
| 用户登录 | 发 | 02#账号#密码# |
|  | 返回值 | 登陆成功：02:01  密码错误：02:02  用户不存在：02:03  用户已登录：02:04 |
| 与单个好友进行聊天 |  |  |
|  |  |  |
| 离线 |  |  |
|  |  |  |