**开发过程总结**

本系统实现了一个简易的即时通讯工具，整个软件遵循C/S（client/server）的架构。

其中，服务器承担的主要工作则是信息类型的判断和转发。即服务器保存着一个当前所有在线客户端的IP和端口号列表（之后的迭代可以用数据库保存这一部分信息，还可以扩展出用户昵称等个人信息），根据接收到的讯息的标识符判断：

- 如果是群聊模式，则服务器会将接收到的讯息转发至每一个在线客户端（完善后可设计仅转发至某个特定的小群聊中，现在的设计更像是全网广播）；

- 如果是获取客户列表，则服务器向发出请求的客户端返回当前所有在线客户端的IP和端口号列表；

- 如果是私聊模式，则服务器会将讯息转发至客户端双击选择的聊天对象，同时基于人性化的信息呈现考量，还会返回一份讯息至发出该讯息的主机；

而客户端系统分群聊和私聊两种模式，具体的使用步骤演示已经在“软件使用说明文档.docx”中给出。

本文档对软件测试过程中发现的主要问题、及解决问题的具体方法作总结，其中具体包括：

1.（1）问题1：有多个客户端同时运行时，如何保证服务器端信息的正确返回？以及如何维持客户端与服务器端的长连接（这对于即时通讯工具而言至关重要）？

即原本单线程的通讯逻辑是：至多支持客户端和服务器端之间的来回通信，无法再多加哪怕一个客户端（存在堵塞）。

（2）解决方法：我觉得无论是多个客户端的情况，还是依托服务器的信息转发等，都必须要实现多线程-即为每一个客户端与服务器之间的连接建立一个线程，这样顺带着也就实现了长连接，即不用每次发消息都重新建立一次连接，同时也就不必为每次连接都重新分配一个端口号。具体的解决过程为：通过在线程内部设置循环的方式，即只要该客户端还有信息要发/发送的信息不为空等，则循环就不会终止，也就保持了长连接：



上图代码的逻辑是，服务器端始终监听着是否有新的客户端建立连接，对于每一个新加入的客户端，都为其分配一个线程来维持会话，以此实现了客户端与服务器端的长连接。

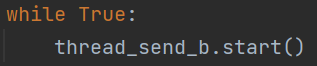
2.（1）问题2：就客户端程序而言，如何实现信息发送和接收的异步IO操作？

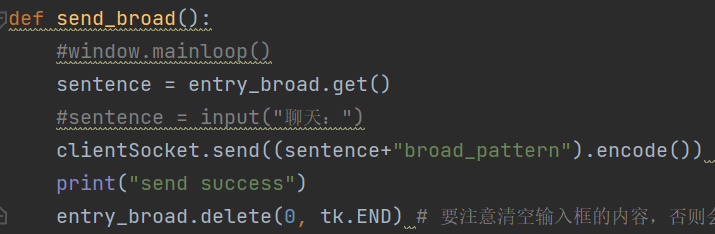
客户端程序最开始的信息发送和接收函数是同步的，即要逐个执行，如此常常会导致这样一个问题：send()函数执行完毕后，下面的recv()等待着服务器端的回应，在还没有接收到服务器响应的情况下，客户端是无法继续向下执行的，也就无法执行下一次的send()，这种同步运行的逻辑会导致即时通讯的效率极低。

（2）解决方法：将客户端信息的发送和接收写成两个线程，这样发送和接收就可以并发执行，send()函数也就不会被recv()函数堵塞住：

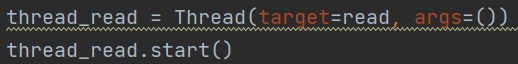
讯息的发送线程：







讯息的接收线程：



3.（1）问题3：python中的thread线程是无法被中止或暂停的，因此在通讯系统分为多个模块的情况下，服务器端传回的讯息只会被最早创建的那个接收线程接收到，其他模块的信息接收线程是无法正常接收讯息的。

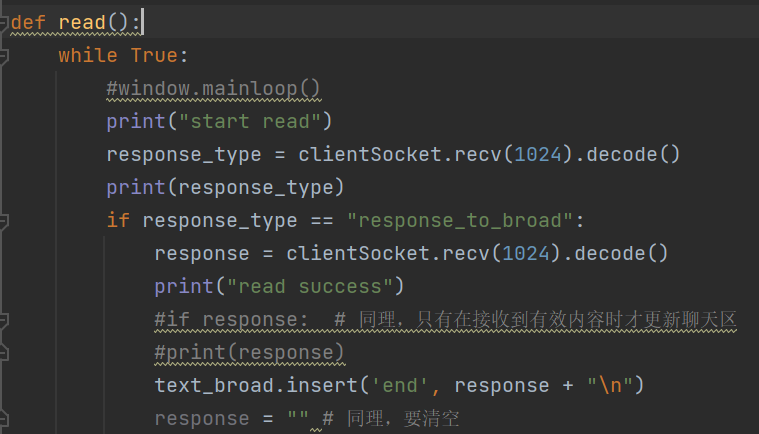
程序分为三个模块，即群聊模块、选择私聊对象模块和私聊模块，原本为每个模块都写了各自的信息接收进程，但之后发现了一个问题：即python中的thread线程是无法中止或暂停的，因此当从一个模块切换至另一个模块时，前一个模块的信息接收线程等仍存活着。

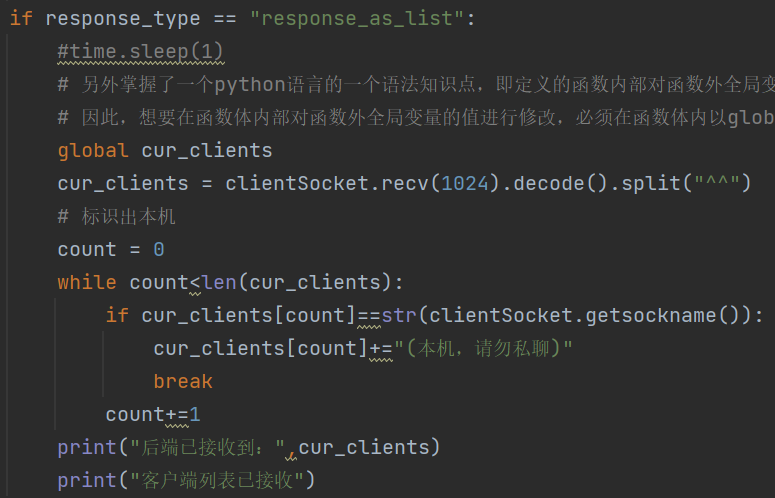
为了避免讯息接收发生混乱，我在三个模块中分别设置了三种信息标识符，并对服务器做了对应的修改，甚至尝试当不符合条件的接收线程在接收到信息时time.sleep几秒、在前一个线程的基础上调用后一个线程、或是使用thread实例的join()函数对前一个线程进行堵塞。

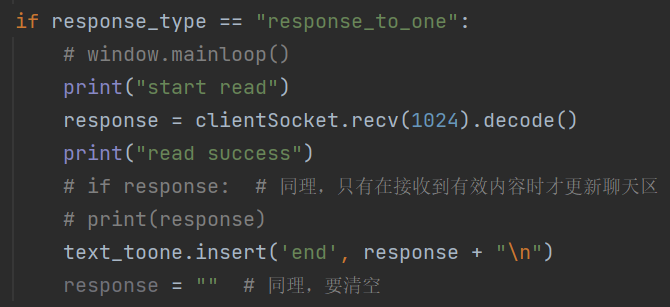
但上述举措都无法解决一个不易被察觉的问题（由多线程异步IO导致的）：即当客户端同时有两个以上的接收（socket.recv()）线程存在时，服务器端传回的讯息只会被最早创建的那个接收线程接收到，其他线程无论如何都无法接收（即便最早的那个线程已被暂停或尚未执行至其中的recv步，服务器也会持续等待这个线程接收而不是其他的，像是进入了一种堵塞状态）

（2）解决方法：对此在网上也并没有找到很好的解决方法，因此我最终决定将三个模块的接收线程合并为一（即下面这个read函数），由此客户端讯息接收的步骤或逻辑是：

- 每次向服务器发送讯息时，都附加上该讯息所属模块的标识符，服务器根据这个标识符判断应该执行哪一部分的代码；而在服务器端，每次向客户端返回数据时，同样会附带其所属模块的标识符；同时整个read函数被分为三部分（根据功能的扩展还可以分出更多），通过数据的标识符判断应该由哪个模块来接收这个讯息：





  
 - 简而言之，整个客户端的各线程共享的是同一个接收进程，这一函数承载了客户端讯息的识别和接收工作（因为函数中实际上包含了3个socket.recv函数，但通过if语句和讯息标识符区别和并行开来，这样前面的问题就得到了很好的解决。  
（ps. 不想彻底杀死线程，主要是考虑到各模块之间的切换可能会很频繁，有些模块之间还存在较大关联-比如选择私聊对象和私聊。另外参考error信息，python中一个thread好像只能被唤醒一次，重新创建同名线程的代价好像比较大）

4.（1）问题4：客户端的在线用户列表是作为全局变量存在的，这种情况下，服务器发给客户端的当前在线用户列表无法被正确或及时地传参、更新至对应的界面展示函数中。

（2）解决方法：掌握了一个python语言的一个语法知识点，即定义的函数内部对函数外全局变量的赋值和引用操作是无效的（默认只是和全局变量重名的函数内局部变量）。

因此，想要在函数体内部对函数外全局变量的值进行修改，必须在函数体内以global特殊字符按相同的变量名进行二次定义，这样在函数内部对全局变量值的修改才会定向至对应的函数外全局变量并被保存下来：



此外，在选取私聊对象模块的主函数中，在相应的私聊模式标识符发送完毕后，加入1秒延迟，使得讯息接收线程对数据列表的更新先进行，界面的生成后进行，否则更新后的用户列表数据无法正常显示：

