实现功能：

服务器端：1+1+n个线程

第一个1：代表的是为服务器端创建输入环境，检测到quit就关闭所有连接。

第二个1：代表的是一个线程用于管理新用户的连接（主线程）。若没有客户端发起连接请求会一直阻塞。

n：代表的是为每个客户单独建立一个线程，用于接收来自该客户的信息（同时进行广播，发送信息到所有客户处）。同时如果检测到”quit“，将其与服务器端断开连接。

为了进行广播，需要维护”在连接列表“，这个过程的重点之一在于利用互斥锁保证列表一致（即：如果一个线程在进行广播访问连接列表，有另一个客户与服务器端建立了连接、修改连接列表，可能导致访问出错）。同理，将客户端挪出在连接列表（可能有两种触发方式：客户端请求断开或服务器关闭）

服务器端关闭连接：

running\_server标记清空。

1.注意为接收/广播信息线程添加server\_running条件，检测到变化全部停止工作。注意此时不能直接close，因为其它线程可能还在工作，close可能出现死锁等问题；比较简单的实现方式是在最后（一切线程结束后）close掉连接。

2.对于管理连接的主线程，由于没有accpet到新的连接，while循环内部会一直阻塞在accpet处，running\_server也无法中断连接。一种解决方法是，在command线程中，在将server\_running置0后发起一个新的连接请求，跳过accpet阻塞过程：发起请求后什么都不需要做，仍然让主线程创建新的连接进程、又挪出客户端即可。

对一些可能的bug的解释：

1.如果短时间内有连续两个客户申请连接，receive慢慢添加进去，不会有逻辑错误；

即使在shutdown = 0后、服务器未实际close前有人申请了连接，未来得及处理就关闭了while循环，最后服务器close后的客户端也会自动提示连接失败。

客户端：1+1个线程

客户端有关套接字的逻辑处理就很简单了，发起连接之后使用两个线程（包括主线程），一个用于给服务器端发送消息，一个用于接收服务器端消息。

难点主要在于对显示的处理：

1. 为了enter输入消息的地方始终位于最下方，我们需要在接收到消息时将已经打印在屏幕上的“enter”回退，将接收消息打印后重新打印“enter”。这一步骤在接收消息的线程中完成，需要用到和光标有关的处理方法。
2. 为了缓存enter中的信息（即在接收到的消息打印后，enter区域看起来只是换了个位置，信息都还在），我们不能够使用fgets、scanf等函数直接输入：这些函数都同样以回车为结束标志，与我们想要缓存的目标矛盾。

考虑将终端修改为非规范模式后（此时输入的信息不再自动打印在屏幕上，这个过程需要上网查阅相关调用），使用getchar对输入的字符一个个处理，同时对输入的“回退”键进行额外处理：getchar到正常的字符就打印出来、存入缓冲区（后续在receive线程和enter一起打印出来），检测到回退键就删除缓冲区内容、取消打印的最后一个字符。

3.保存用户名的逻辑：在客户端启动的最开始，由用户输入自己的用户名，后续将message传给服务器端时在输入massage的最前面加上自己的用户名即可；服务器端处理消息时，额外添加对“消息来源和消息目标一致”的处理逻辑，在信息最开头再添加一个“(You)”。