摩根项目 Design Doc

516030910293 姚子航

注：该项目使用的IDE是Visual Studio 2010，采用通讯库为WinSock2.h，如果使用Linux系统进行检测则会编译失败，使用其他IDE有可能发生未知异常，所以我将自己电脑上运行的一部分运行状况截图。

第一部分：关于服务器端与客户端的关系架构

1. 服务端

程序的主要部分，即为项目“服务器”，采用Windows Socket 借口，使用虚拟地址“127.0.0.1”，利用一个while循环来实现与多个客户端连接，每次循环连接成功后都会对当前客户端开启一个新的线程，所有客户端连接成功的socket都储存在全局变量m\_Server数组里。

1. 普通客户端

用户部分，即为项目“普通客户端”，普通客户端开启两个线程，一个用来发送订单，一个用来接受订单。用户开启客户端会有指令来操作一系列交易。客户端与服务器通过recv函数与send函数来互相发送FIX协议信息。

1. Monitor 客户端

Monitor Client，用来监测每个连接的用户进行的一系列操作。

1. 随机客户端

Random Client，用来模拟实际的交易所，该客户端会随机发送买卖订单给服务器。

第二部分：主要功能类

1. Stock类（股票）

股票类有以下几个属性：

1.averagePrice（当前价格）

1. start\_price（开盘价格）
2. id（股票代码）
3. stock\_name（股票名称）
4. end\_price（闭盘价格）
5. max\_price（最高价格）

7.min\_price（最低价格）

成员函数：

1.每个属性对应的获取接口

2.updatePrice（更新最高价格与最低价格）

1. Order类（订单）

关于Order对象有以下几个属性：

1.Order\_id（每个订单独有，由系统自动生成）

2.stock\_name股票名称

3.amount （交易数量）

4.side（买方卖方）

5.avg\_price（当前平均交易价格）

6.give\_price（订单产生时出价）

7.execute\_amount（已经交易的数量）

8.infomation（具体的交易情况）

成员函数：

1. 各个属性的获取接口
2. writeResult(string opposite\_name,int ex\_amount,double ex\_price)

该函数作用是将交易双方的一系列交易信息（交易数量、价格）写入对应的股票record.txt中。

1. FixMessage类（消息类）

关于FixMessage类有以下属性：

1.side（买方卖方，tag=“54”）

1. average\_price（平均价格，tag=“6”）
2. price（价格，tag=“44”）
3. filled\_amount（已经交易数量，tag=“14”）
4. order\_id（订单号，tag=“11”）
5. user\_name（用户名，tag=“49”）
6. open\_amount（剩余未交易数量，tag=“151”）
7. amount（总数，tag=“38”）
8. stock\_name（股票名称，tag=“1”）
9. order\_type（订单消息种类(新订单、取消、拒绝)，tag=“35”）
10. execution type（执行种类，tag=“150”）

12.order\_status（订单状态(新订单，部分交易，全部交易，取消)，tag=“39”）

成员函数：

1.各个属性的获取接口

2.在构造函数中对获取的字符串进行解析，用“；”分割出各个tag

第三部分：关于匹配机制

一、在服务器端程序中构造一个子线程共享的Order队列，对于每一个线程提出的要求，都要去Order队列里进行匹配，所以Order队列应对子线程保持互斥。这里使用**queue\_section**全局关键段来达到互斥，给orderList“上锁”。

下列是全局的订单队列：

1. buyOrderList：买方订单，按照价格由高到低排序，出价高的优先进行交易
2. SaleOrderList:卖方订单，按照价格由低到高排序，要价低的优先进行交易
3. endBuyOrder:已经结束或被取消的买方订单
4. endSaleOrder:已经结束或被取消的卖方订单

二、Order进入后，先辨别是买方Order还是卖方Order进行分流，其次判断价格是否在市场内，不在的话插入队列结束，在的话直接进行匹配完成交易。

匹配细节：优先匹配最令客户满意的Order，再进行数量匹配，出现下列三种情况：

1. 己方订单数量大于对方订单数量，则对方订单全部交易，己方继续与队列中下一个订单进行比较，循环直到不能再交易或全部交易。不能交易时如果未交易完，则放入队列buyOrderList或SaleOrderList
2. 己方订单数量等于对方订单数量，双方均全部交易
3. 己方订单数量小于对方订单数量，己方全部交易，对方部分交易不出队列，直接结束。

三、取消操作

用户发出取消操作，检查buyOrderList、SaleOrderList，如果发现要取消的订单，则将其放入endBuyOrder或endSaleOrder，如果在endBuyOrder、endSaleOrder中找到该订单，则拒绝取消操作，因为该订单已被交易完毕。

四、用户登录

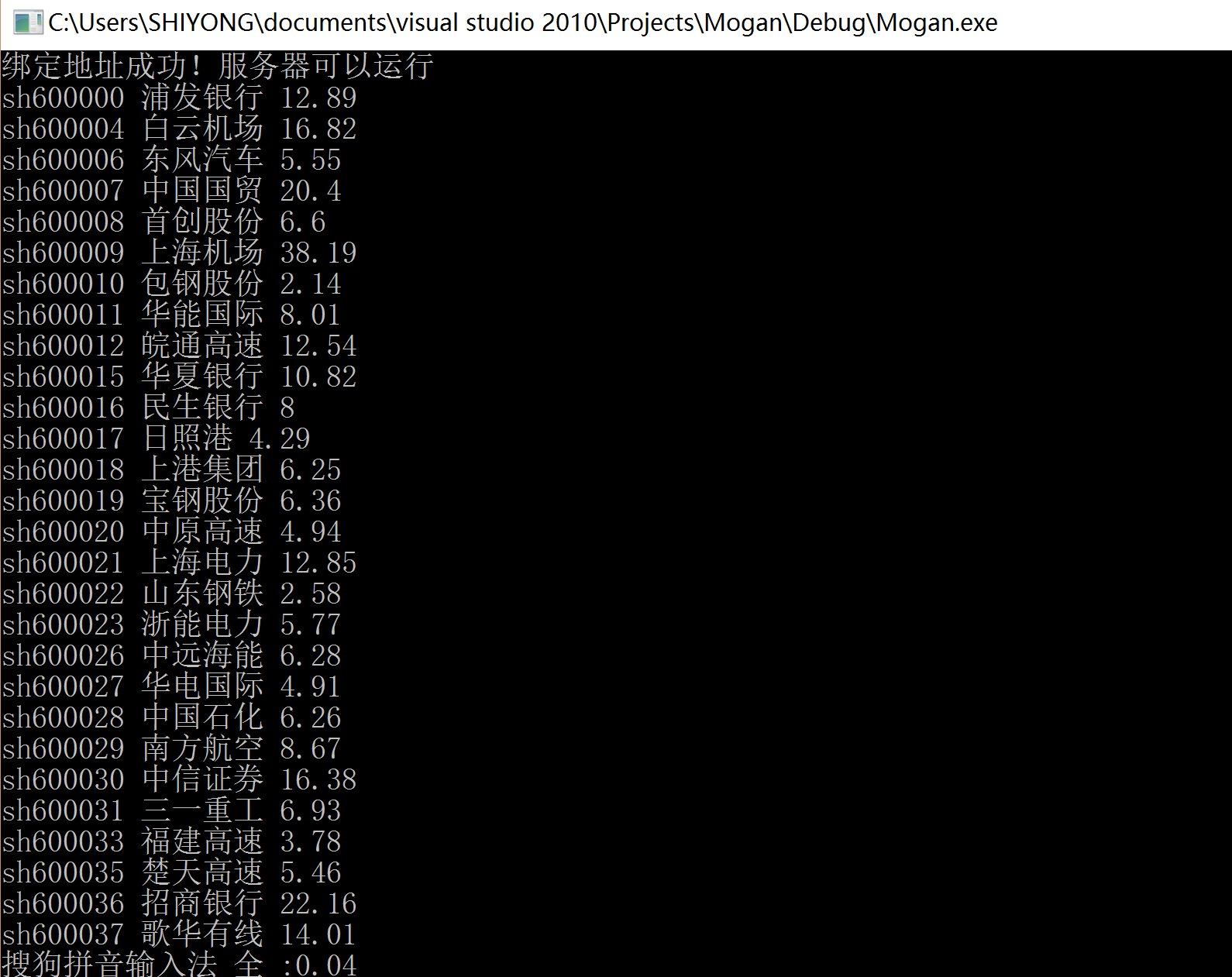
设置一个用户名密码的文件，根据是否存在来判定登陆成功、设置一个监视管理账号用户名为“Monitor Client”，当服务器识别出这个用户时，会将生成的Socket，储存在全局变量monitor\_client



登录后的交互界面如上图所示。

五、服务器开始后的信息准备

从share\_information中逐个读取股票信息，并储存在全局变量share\_map中



六、Monitor Client的orderbook打印及定期清屏

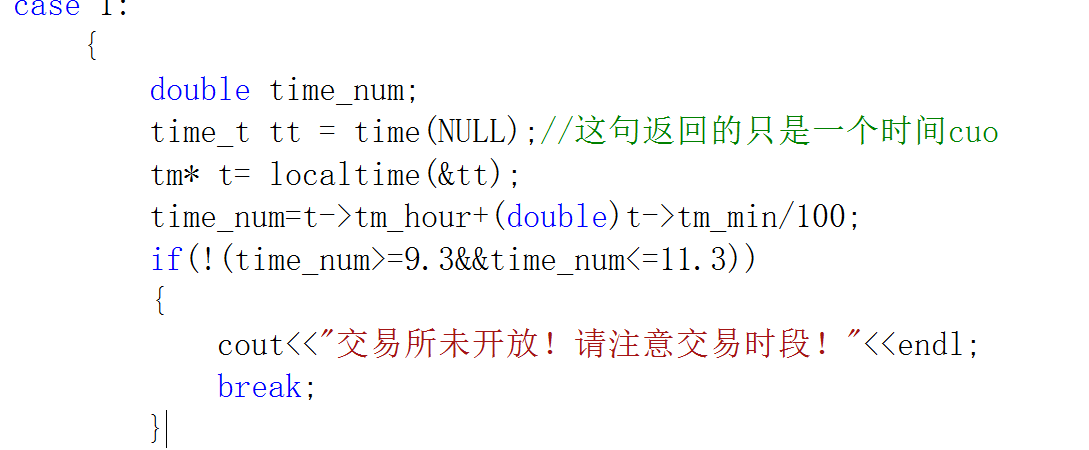
每当有一个操作发生，如果上文提到的全局变量monitor\_client存在，即已开启Monitor Client，则会向该Monitor发送交易信息，Monitor每接到一笔交易信息，就会将其解析判断转化为可读交易信息打印。

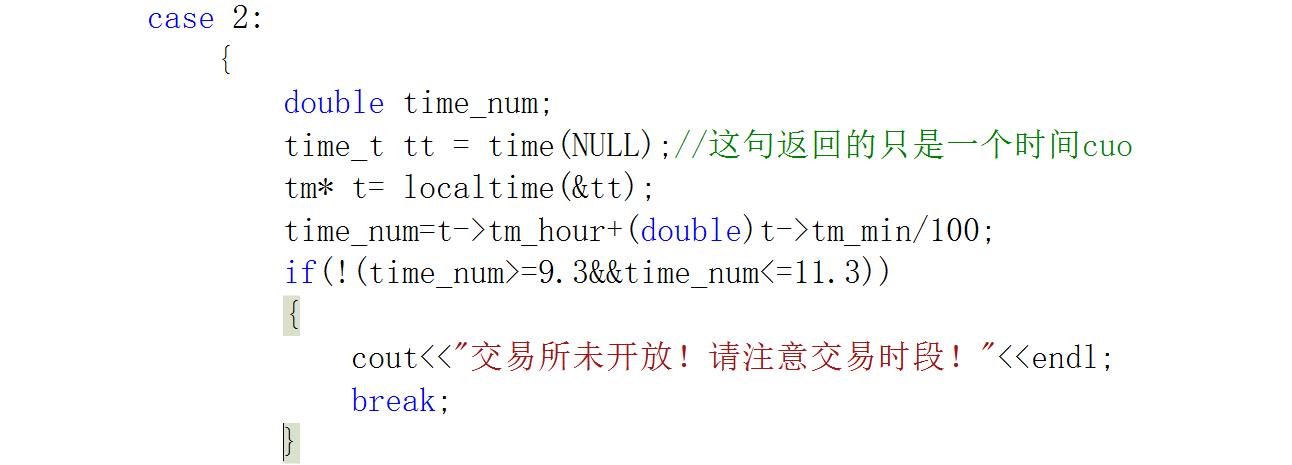


七、关于交易的一系列规范限制

1.交易价格不能超过该股票参考价格的120%（max\_price）,不能低于参考价格的80%(min\_price)

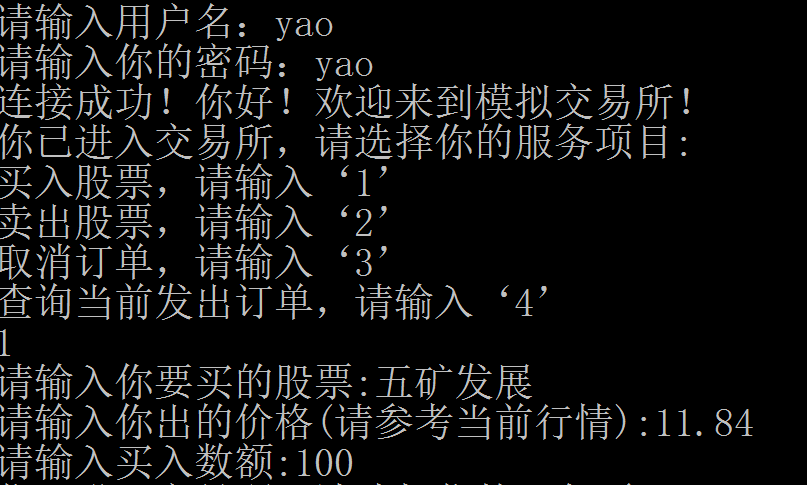
2.交易时间必须在系统时间上午九点半与十一点半之间，如果不在则不能发送信息，如果需要发送信息请将Client line43-line47&&line71-line75注释掉。



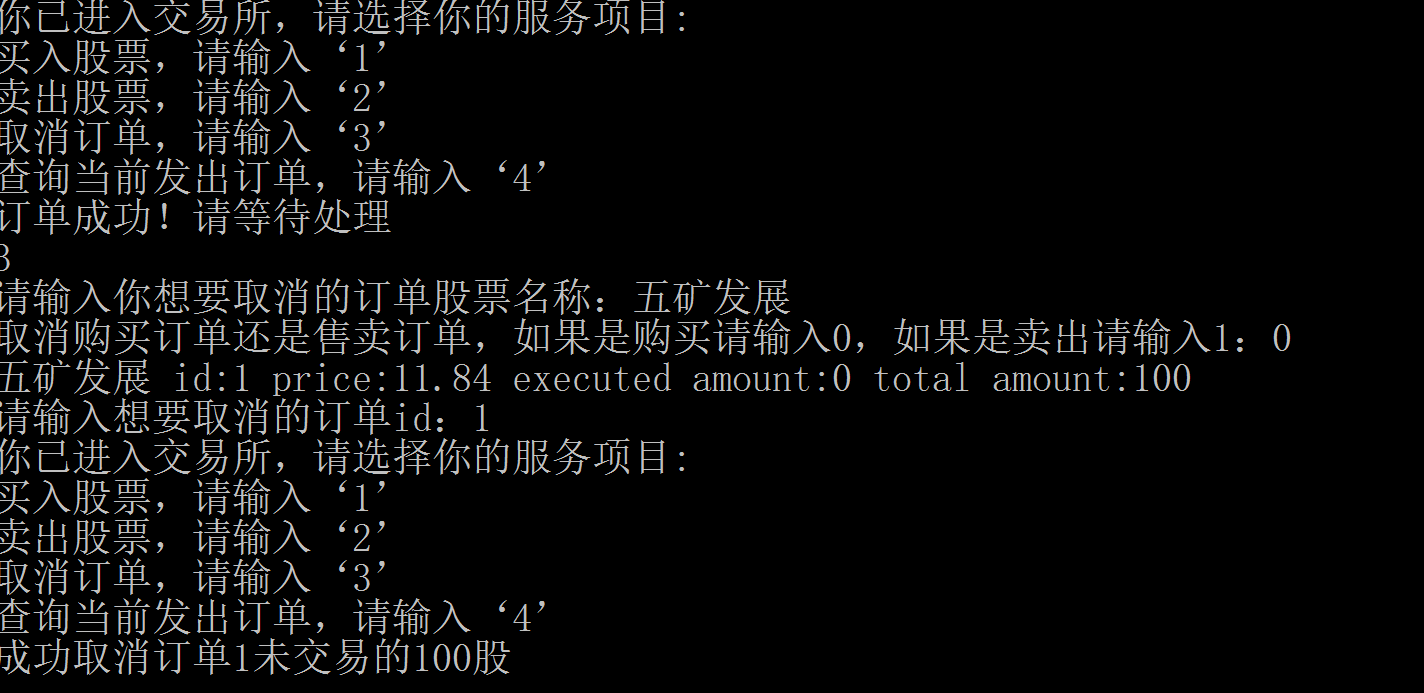


八、运行截图

1. 买卖股票



1. 取消订单



1. 查询现有订单

