

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

一、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。

二、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。

三、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。

四、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

一、Stock 类（股票）

股票类有以下几个属性：

- 1.averagePrice（当前价格）
- 2.start_price（开盘价格）
- 3.id（股票代码）
- 4.stock_name（股票名称）
- 5.end_price（闭盘价格）

6.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）

二、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。

三、FixMessage 类（消息类）

关于 FixMessage 类有以下属性：

1.side（买方卖方，tag=“54”）

2.average_price（平均价格，tag=“6”）

3.price（价格，tag=“44”）

4.filled_amount（已经交易数量，tag=“14”）

5.order_id（订单号，tag=“11”）

6.user_name（用户名，tag=“49”）

7.open_amount（剩余未交易数量，tag=“151”）

8.amount（总数，tag=“38”）

9.stock_name（股票名称，tag=“1”）

10.order_type（订单消息种类(新订单、取消、拒绝)，tag=“35”）

11.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`

 C:\Users\SHIYONG\documents\visual studio 2010\Projects\Client\Debug\Client.exe

```
sh600039 四川路桥 3.98
sh600048 保利地产 9.73
sh600050 中国联通 7.47
sh600051 宁波联合 8.99
sh600052 浙江广厦 4.97
sh600053 九鼎投资 32.9
sh600054 黄山旅游 16.93
sh600055 DR万东医 11.09
sh600057 象屿股份 9.67
sh600058 五矿发展 11.84
请输入用户名: yao
请输入你的密码: yao
连接成功! 你好! 欢迎来到模拟交易所!
你已进入交易所, 请选择你的服务项目:
买入股票, 请输入 '1'
卖出股票, 请输入 '2'
取消订单, 请输入 '3'
查询当前发出订单, 请输入 '4'
1
交易所未开放! 请注意交易时段!
你已进入交易所, 请选择你的服务项目:
买入股票, 请输入 '1'
卖出股票, 请输入 '2'
取消订单, 请输入 '3'
查询当前发出订单, 请输入 '4'
-
```

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

```
C:\Users\SHIYONG\documents\visual studio 2010\Projects\Mogan\Debug\Mogan.exe
绑定地址成功！服务器可以运行
sh600000 浦发银行 12.89
sh600004 白云机场 16.82
sh600006 东风汽车 5.55
sh600007 中国国贸 20.4
sh600008 首创股份 6.6
sh600009 上海机场 38.19
sh600010 宝钢股份 2.14
sh600011 华能国际 8.01
sh600012 皖通高速 12.54
sh600015 华夏银行 10.82
sh600016 民生银行 8
sh600017 日照港 4.29
sh600018 上港集团 6.25
sh600019 宝钢股份 6.36
sh600020 中原高速 4.94
sh600021 上海电力 12.85
sh600022 山东钢铁 2.58
sh600023 浙能电力 5.77
sh600026 中远海能 6.28
sh600027 华电国际 4.91
sh600028 中国石化 6.26
sh600029 南方航空 8.67
sh600030 中信证券 16.38
sh600031 三一重工 6.93
sh600033 福建高速 3.78
sh600035 楚天高速 5.46
sh600036 招商银行 22.16
sh600037 歌华有线 14.01
搜狗拼音输入法 全 :0.04
```

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。

```
C:\Users\SHIYONG\documents\visual studio 2010\Projects\MonitorClient\Debug\MonitorClient.exe
你好！欢迎来到模拟交易所！
用户的购买订单成功！订单号为0, 股票名称为五矿发展, 订单价格为11.84, 数量为100.
用户取消了未交易完的0订单中的100股五矿发展
用户的购买订单成功！订单号为1, 股票名称为五矿发展, 订单价格为11.84, 数量为100.
用户取消了未交易完的1订单中的100股五矿发展
```

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`)

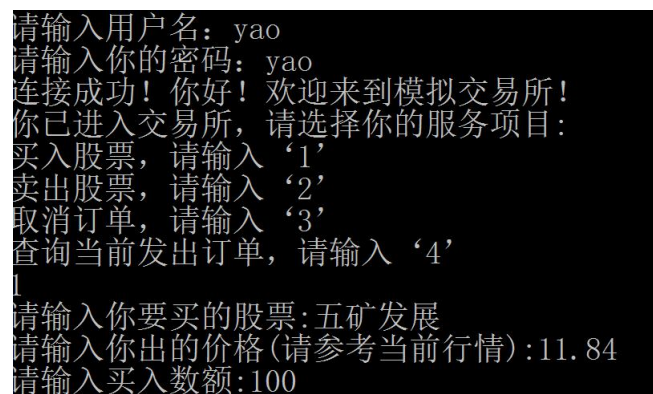
2. 交易时间必须在系统时间上午九点半与十一点半之间, 如果不在则不能发送信息, 如果需要发送信息请将 Client line43-line47&&line71-line75 注释掉。

```
case 1:
{
    double time_num;
    time_t tt = time(NULL); //这句返回的只是一个时间戳
    tm* t= localtime(&tt);
    time_num=t->tm_hour+(double)t->tm_min/100;
    if(!(time_num>=9.3&&time_num<=11.3))
    {
        cout<<"交易所未开放! 请注意交易时段!"<<endl;
        break;
    }

case 2:
{
    double time_num;
    time_t tt = time(NULL); //这句返回的只是一个时间戳
    tm* t= localtime(&tt);
    time_num=t->tm_hour+(double)t->tm_min/100;
    if(!(time_num>=9.3&&time_num<=11.3))
    {
        cout<<"交易所未开放! 请注意交易时段!"<<endl;
        break;
    }
}
```

八、运行截图

1. 买卖股票



```
请输入用户名: yao
请输入你的密码: yao
连接成功! 你好! 欢迎来到模拟交易所!
你已进入交易所, 请选择你的服务项目:
买入股票, 请输入 '1'
卖出股票, 请输入 '2'
取消订单, 请输入 '3'
查询当前发出订单, 请输入 '4'
1
请输入你要买的股票: 五矿发展
请输入你出的价格(请参考当前行情): 11.84
请输入买入数额: 100
```

2. 取消订单

```
你已进入交易所，请选择你的服务项目：
买入股票，请输入 '1'
卖出股票，请输入 '2'
取消订单，请输入 '3'
查询当前发出订单，请输入 '4'
订单成功！请等待处理
3
请输入你想要取消的订单股票名称：五矿发展
取消购买订单还是售卖订单，如果是购买请输入0，如果是卖出请输入1：0
五矿发展 id:1 price:11.84 executed amount:0 total amount:100
请输入想要取消的订单id: 1
你已进入交易所，请选择你的服务项目：
买入股票，请输入 '1'
卖出股票，请输入 '2'
取消订单，请输入 '3'
查询当前发出订单，请输入 '4'
成功取消订单1未交易的100股
```

3.查询现有订单

```
买入股票，请输入 '1'
卖出股票，请输入 '2'
取消订单，请输入 '3'
查询当前发出订单，请输入 '4'
订单成功！请等待处理
4
当前购买的订单有：
五矿发展 id:0 price:11.84 executed amount:0 total amount:100
当前售卖的订单有：
你已进入交易所，请选择你的服务项目：
买入股票，请输入 '1'
卖出股票，请输入 '2'
取消订单，请输入 '3'
查询当前发出订单，请输入 '4'
```