# 多线程交易算法TradeExecute说明

1. **程序说明**

本交易算法的主要思路为：分别登陆行情与交易服务器后，等待交易指令来临，对于每一个交易指令分配单独的一个交易线程，在常规的交易时间内按照市价进行报价等待成交；常规时间结束若还有未成交订单则进入最后成交时间，按照市价+I ticks价格进行试探性报价，然后马上撤单，I 值随着报价次数线性增长，直至成交或超时。

1. **开发环境**

本交易算法基于【金仕达投资管理系统交易接口IDMP C++版V2.6.0】交易接口与【宏惠交易行情TDF C++版 V1.2】行情接口进行交易算法开发，开发环境为Microsoft Visual Studio2010版，测试环境为Intel i5 3210 64位CPU + Windows 8.1 64位专业版+boost1.55

Boost1.55安装方法：

解压缩boost\_1\_55\_0.zip

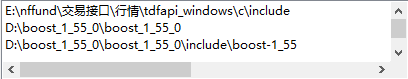
执行bootstrap.bat

执行b2.exe

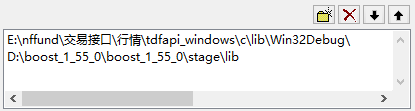
VS中工程配置方法：

TradeExecute工程——>右键属性——>VC++目录

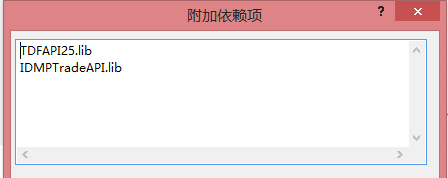
【包含目录】中添加tdfapi的include目录与boost的include目录，例如：



【库目录】中添加tdfapi的lib目录(根据debug版和release版选择)与boost的lib目录，例如：



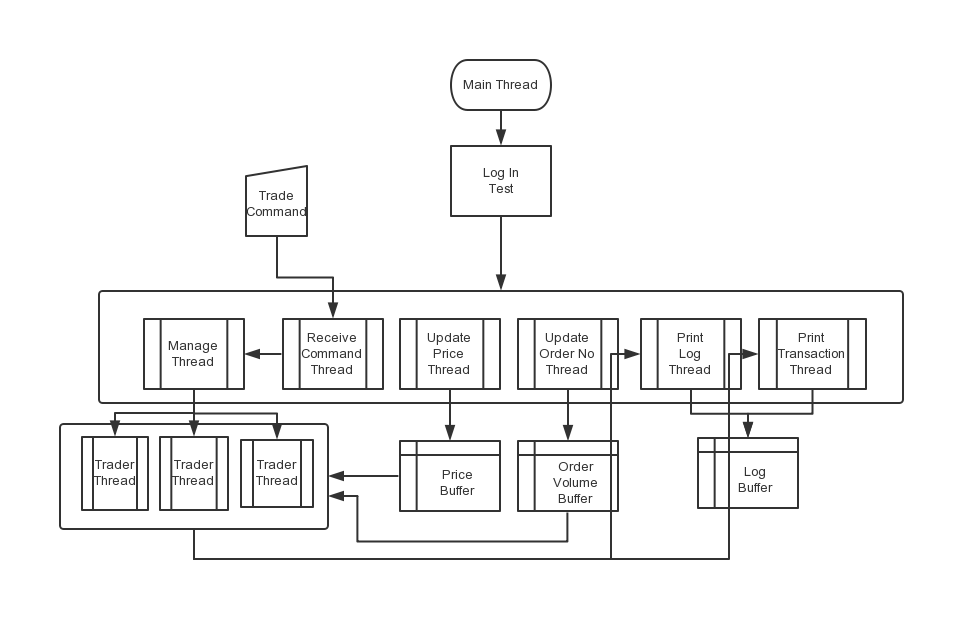
属性——>链接器——>输入——>附加依赖项中增加TDFAPI25.lib和IDMPTradeAPI.lib，例如：



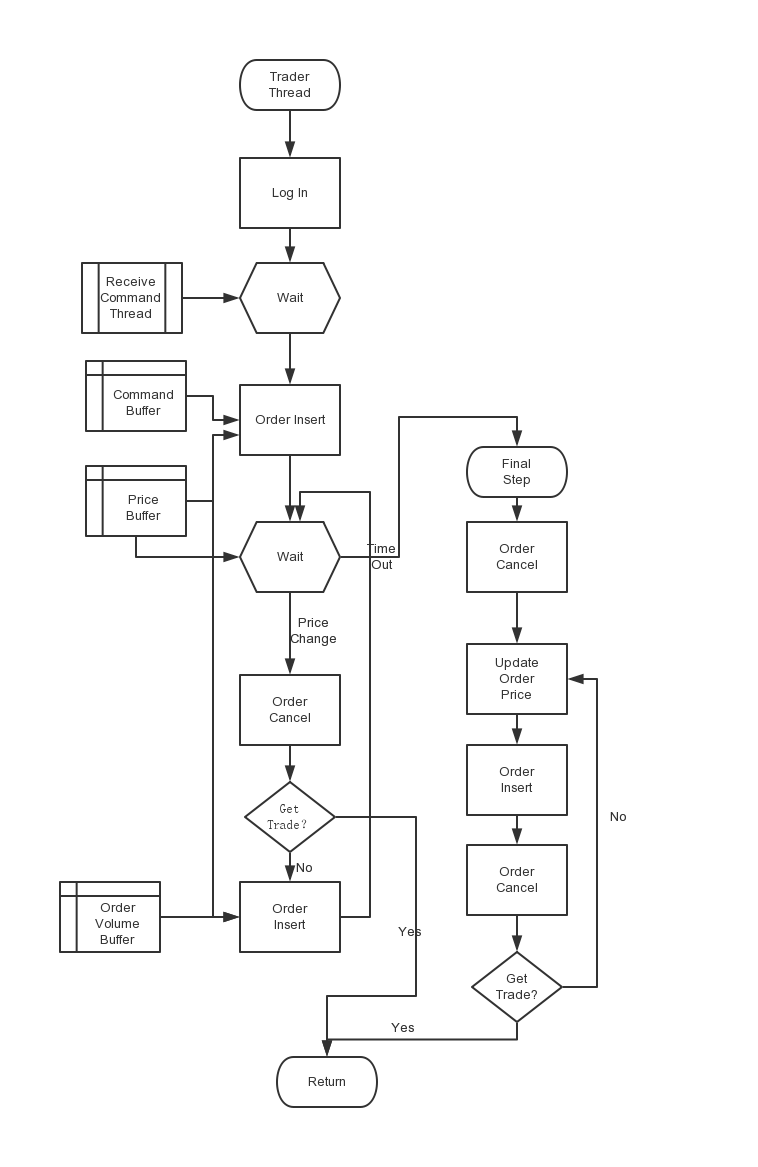
1. **程序结构**

本交易算法程序主要分为主线程、交易线程管理、交易线程、行情线程、log记录几个部分组成。

程序的线程结构如图所示



* 1. 主线程：主线程作为所有逻辑的最上层，分成两个部分。首先使用预设定的用户名密码测试交易服务器登录是否正常，然后分别启动线程管理(ManageThread)、交易命令接收(ReceiveCommandThread)、行情更新(TDFUpdatePriceThread)、订单信息更新(UpdateOrderNoThread)、交易日志信息(PrintLogThread)、成交日志信息(PrintTransLogThread)等6个线程，然后等待所有线程都运行完毕。
  2. 线程管理(ManageThread)：开启MAX\_THREAD\_NUM个交易线程(TraderThread)，登陆到服务器然后等待，若有一个交易命令，则意味着线程池中有一个线程被唤醒，需要新建一个交易线程登陆到服务器然后继续等待。
  3. 交易命令接收(ReceiveCommandThread)：通过用户输入判断是否继续交易，若继续交易则等待用户输入订单信息并写入cmd\_buffer中；否则标记程序结束标志，线程结束。
  4. 交易线程(TraderThread):登陆后按照指令下单，然后等待价格变化更新订单信息，直至交易成功或者超时进入最后阶段，是本程序的核心部分，其逻辑结构如下图所示：



* 1. 行情更新(TDFUpdatePriceThread)：连接宏汇的行情服务器，按照预读取的合约列表(暂时只支持股指期货)，并在回调函数中更新price buffer的内容。
  2. 订单信息更新(UpdateOrderNoThread)：根据现在活跃在内存中的订单号，定时(500ms)更新其订单未成交订单数量信息。
  3. 交易日志信息(PrintLogThread)：在每一次发出订单、撤销订单、更新订单信息时同时记录到log.txt文件中。
  4. 成交日志信息(PrintTransLogThread)：在每次成交(全部成交或部分成交)时记录并计算本次命令的已成交订单部分均价。

**4. 程序参数定义**

MAX\_THREAD\_NUM: 交易线程池中等待线程最大个数。

N\_TICKS: 交易最后阶段每次价格改变tick个数

LOG\_PATH: log文件绝对路径

TRANS\_LOG\_PATH: 交易log文件

ACTION\_PATH： 本地编号记录文件绝对路径

SYMBOL\_PATH: 需要检测价格变化的合约列表文件绝对路径