### 实验4 迷你区块链系统

1.背景

在实验2中，我们从文件中读取数据，构造了由一组区块构成的链表。本实验要求在此基础上，实现一个迷你区块链系统。包括多个节点之间通过“随机”共识来维护一条一致的链，完成消息“收发”处理，以及查询功能。

2．目的

* 练习队列操作和文件操作。
* 为后续实验做准备。

1. 问题描述和具体要求

* 搭建一个由三个“节点”（程序）构成的微型区块链系统。

**节点1：模拟一个终端客户**，每隔一定时间（比如100毫秒）随机向另外两个程序发送交易请求或查询请求。交易数据从实验2的数据集中生成。每条请求将“插入”到对应节点的“客户消息队列”尾部。

**节点2和节点3：模拟两个区块链节点**，独立实现下面的逻辑：

while (1) {

IF（中奖）{ //中奖调用一个随机数来实现，中奖几率在0.01-0.1之间

从“交易池”中取出n (>=1)个交易，组成一个区块newBLK。

newBLK的prevHash等于本节点区块链表最后一个区块lastBLK的hash值。newBLK的hash值可以采用一个随机函数来生成；height值为lastBLK的height+1； merkleRoot和nonce都为空。newBLK中的交易集合由上述n个交易构成。

将newBLK插入本节点的区块链表末尾。

将newBLK以某个格式（比如JSON）组成字符串“发送”给另一个区块链节点的“区块消息队列”。

} ELSE { //没有中奖

IF （“区块消息队列”不为空）{

从“区块消息队列”头部取出一个消息（内容为区块）；

判断该区块是否与已有区块冲突（即是否存在一个区块，与新区块的preHash相同。）

IF（冲突） 丢弃该区块；

ELSE {

将该区块插入到本节点区块链表末尾；

从“交易池”中删除该区块中包含的交易；

}

} ELSE { //“区块消息队列”为空

从“客户消息队列”头部取出一个消息MSG；

IF（MSG 是交易）{

IF（“交易池”不包含该交易） 将该交易加入“交易池”；

ELSE 丢弃该交易；

}

ELSE IF（MSG 是查询）{

在本节点维护的区块链表中执行查询；

将查询结果输出在屏幕上；

}

}

}

Sleep(200)；// 休眠200毫秒

}

* 消息收发方法：鉴于同学们没学过网络或多线程，本实验中采用文件读写方式来进行消息的收发。每个节点维护自己的一个文件夹。若节点A给节点B发消息，则在节点B的文件夹中写入一个文件。
* 消息队列实现参考方法：
  + 队列以同一文件夹下面多个文件的形式保存在磁盘上。
  + 队列中每个节点（包括头节点）是一个文件，可以采用链表方法来实现。
  + 每个节点的地址为对应文件的文件名。
  + 每个节点的next 域为下一个节点的地址（文件名）

例子：

文件head.txt的内容如下所示：

Next: 1.txt

Data: NULL

文件1.txt的内容如下所示：

Next: 2.txt

Data: { msg: ……

type: …… }

* + 发送节点所发送的消息会在相应文件夹下面新建一个文件，并插入队列尾部。
  + 接收节点每次从队列头部删去一个节点（文件）。
* 查询功能：支持按照height或hash查找一个区块；或按照交易id 查询一个交易；以及按顺序展示当前链表中所有区块的高度和hash值。

1. 展示内容

* 多个节点能正常通信，并能维护一个一致的区块链。在两个节点分别按顺序展示链表所有区块，比较是否一致。
* 能正确执行查询。

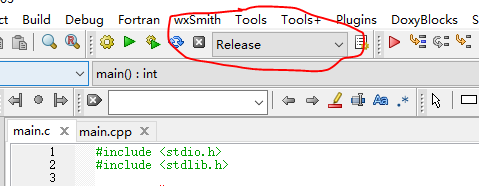
1. 进阶要求

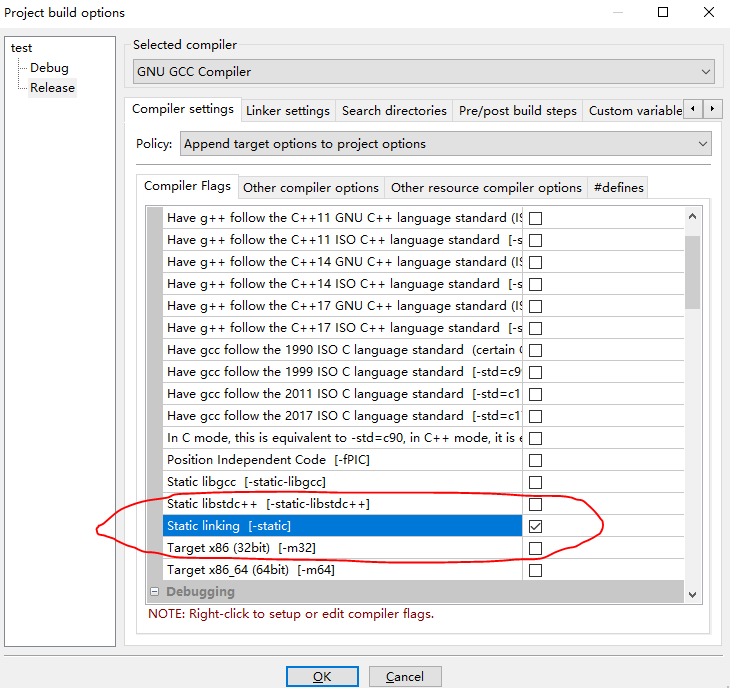
* 实现KMP算法来进行字符串查找。

1. 注意事项

* 控制消息的发送速度，小心磁盘容量！
* 本次实验不需要检查交易的合法性！
* 可能出现两个程序同时去操作同一个文件的情况，要处理此类冲突

1. 以下是在Code Blocks（windows环境）中发布独立运行程序的步骤。供参考。若使用其他IDE，请自行查询解决方案。也可尝试同时打开多个IDE来运行不同节点。

第一步：将此处由 “Debug”改为 “Release”。

第二步：在菜单Project/Build Options 中，勾选Static linking [-static]

第三步：在菜单Build中执行Rebuild

第四步：打开项目文件所在目录，在Release目录下可以看到一个exe文件，双击执行即可。

提示：为了看到运行效果，可以在程序结束前加一个 getchar()。