

# 区块链lab4

本实验思路简单，但是由于没有学过网络和多线程，所以实现起来比较吃力。

## 运行结果

### 按顺序展示区块链

```
id:1.txt
next block: 2.txt
PrevHash:
Hash:
Transactions:
id:2.txt
next block: 3.txt
PrevHash:
Hash: 15319383121940947389
Transactions:
20d09edd4e8d60ce32fbe041cc1f8fcc0c2e5a2a4dc97d336ab36595bf850356
id:3.txt
next block: 4.txt
PrevHash: 15319383121940947389
Hash: 11890619108523121686
Transactions:
fc32a1564e78082f64477efe6bfedaef13f3c0cea57f7b73ec1daffd598e7aef
c8b60292a8c72eaf7184bc6ed770076527a5960d0647ae9fe896be0c5a1fb2cc
e95374ae8887a0b66b2dc6ff431ef7f2c63a05b778f4eff982ea00569a5e88c4
id:4.txt
next block: 5.txt
PrevHash: 11890619108523121686
Hash: 3763088872136251199
Transactions:
110aabef9271f5ffac5009adffa117e28e74341c07e1ffffae903fa511716ab9
id:5.txt
next block: 6.txt
PrevHash: 3763088872136251199
Hash: 6851962763838322581
Transactions:
d8f7a498e129f79b80caf644d16685a13c770566f81b9560179afc54488c9b10
```

### 查找特定高度或哈希值








































```
no such Height:12
id:1.txt
next block: 2.txt
PrevHash: 83121940947389
Hash: 11890619108523121686
Transactions:
fc32a1564e78082f64477efe6bfedaef13f3c0cea57f7b73ec1daffd598e7aef
c8b60292a8c72eaf7184bc6ed770076527a5960d0647ae9fe896be0c5a1fb2cc
e95374ae8887a0b66b2dc6ff431ef7f2c63a05b778f4eff982ea00569a5e88c4
id:2.txt
```

```
id:15.txt
next block: 16.txt
PrevHash: 3857802244375008160
Hash: 14131760101097714382
Transactions:
0dfe3a66e6aed76ae97b31329863a69f1fd4873de0169610d8386f516f2cc43d
5b3f0a651529bde1ea8f2368d631752be0e28d1fd5f287cc86d63ae0983a1ac2
20f8e545f8c135f3373ae2b4437c06ab8b4e4605b28d91779b5c862c790a1388
```

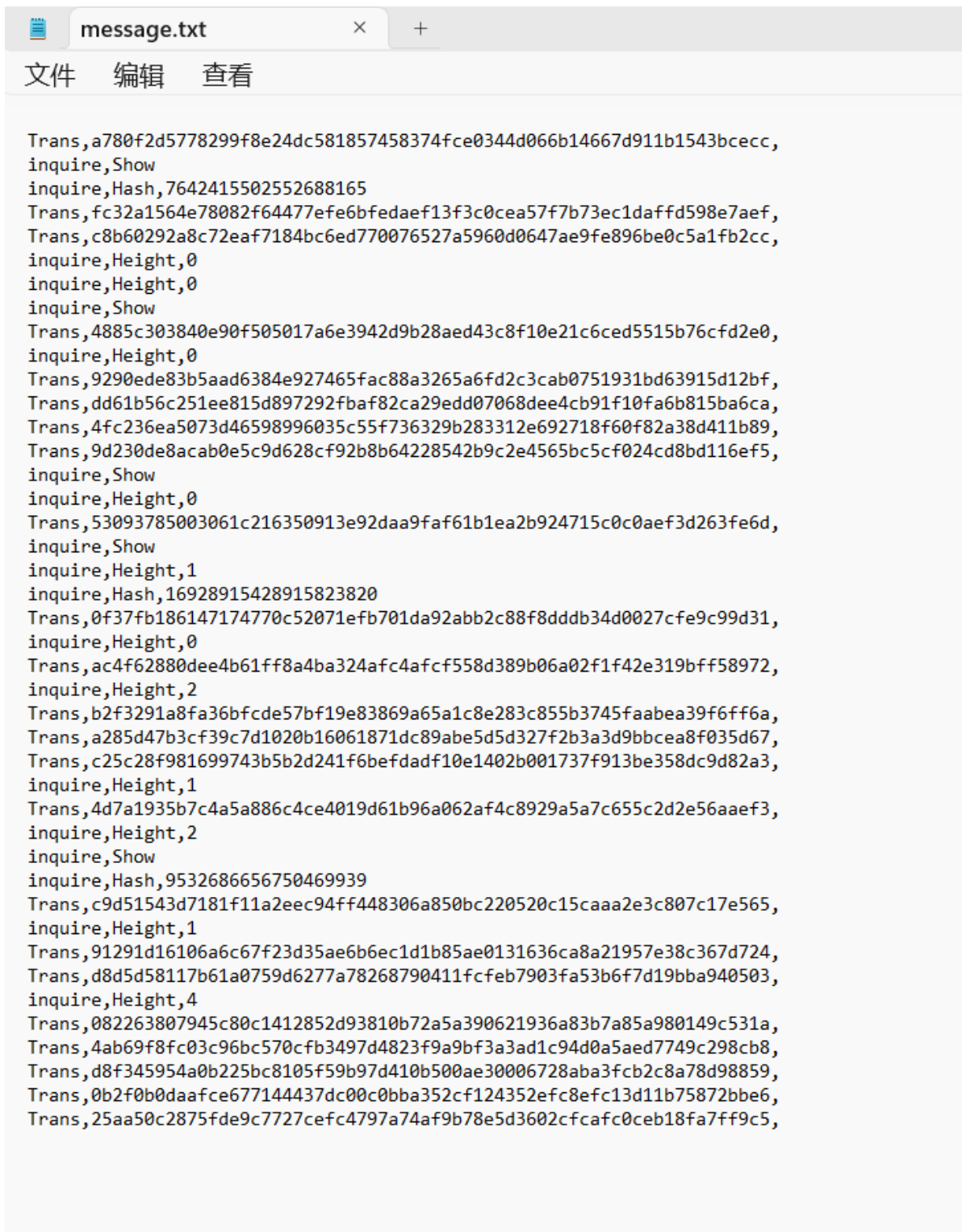
包括了找到和找不到两种情况。

## 文件截图：

### 其中一个节点的文件夹

 1.txt	 2.txt	 3.txt	 4.txt	 5.txt
 6.txt	 7.txt	 8.txt	 9.txt	 10.txt
 11.txt	 12.txt	 13.txt	 14.txt	 15.txt
 16.txt	 17.txt	 18.txt	 19.txt	 20.txt
 21.txt	 22.txt	 23.txt	 24.txt	 25.txt
 26.txt	 27.txt	 28.txt	 29.txt	 30.txt
 31.txt	 32.txt	 33.txt	 34.txt	 35.txt
 36.txt	 block_message.txt	 head.txt	 message.txt	 trans_pool.txt

### p1节点生成的客户端消息：



## 两个结点互发消息

[illegible][illegible][illegible]

(为什么会是这个样子之后会解释)

## 思路分析

按照题目文档要求按部就班实现。

在实现过程中遇到了几个比较困难或是重要的要点：

- 根据实验二的数据生成新的交易
- 删去 txt 第一行或是指定某一行
- 实现"区块消息队列"
- 实现区块冲突的检测
- 传送data

依次解决思路：

**生成交易：**用之前的代码，把交易数据读进来之后随机生成一个高度，找到区块，把对应高度区块的所有交易全部复制下来。这样能有随机性，同时保证了数据较为合理。

**删除操作：**因为对 txt 文件操作，特别是删除非末尾位置较为复杂。所以选择实现上简单的办法：依次读取，把所需要删除的行跳过，删除原文件，重新生成新的文件。利用这种方式来进行。

**区块消息队列：**问题在于，如果使用链表，相当于在实现按队列实现区块之外，还得再为区块消息实现一个队列。感觉这样太复杂，能力有限写不出来，于是换了新的办法：用一个文件来存所有的区块，为了实现区分不同区块，把原本区块的按照换行符分割改为逗号分隔，再在不同区块间用换行符分隔。这样便实现了对区块消息的处理。同时删除操作实现了和 `txt` 删除某行操作的统一。（于是最终呈现为之前的截图那个样子）

**冲突区块的检测:**为了实现检测，遍历区块是必须的，但这里有个问题就是对于最后一个区块如果直接使用 `ifstream` 会因为没有文件而访问失败。所以选择 `ios::app` 追加写模式，对于没有的文件，生成文件之后再访问，对于已有的文件，进入文件之后再 `.seekg()`，`.seekp()` 来实现从头开始读取。

**传送数据：**由于没有学过 json 格式，这个地方实现起来比较困难，没有办法，只能采用最朴素的办法：

严格按照顺序来写data段，每个部分以换行符分割，去掉别的所有的东西。

```
block:
  (next) xxx.txt
prevhash
hash
height
translist
```

把交易项放在最后是由于取出的交易条数由n来决定，放到最后比较灵活。

对于置空的数据项，因为用不上，所以为了减少工作量就没有单独写了（当然，如果要写的话，在前几行随便找个地方插入都可以，所以这不算问题）。

## 具体实现

实现上没什么特别重要的地方，就把思路依次实现就可以了。

由于本实验要求较多，较复杂，为了方便自己理清思路，同时debug，在代码中写了很多注释，代码和注释数量的比值大约10：1。

具体实现可以在代码中体现。

整个lab的文件夹如下：

名称	修改日期	类型	大小
.vscode	2023-11-21 1...	文件夹	
node0	2023-11-21 2...	文件夹	
node1	2023-11-21 1...	文件夹	
nothing	2023-11-21 1...	文件夹	
blocks.csv	2023-10-07 1...	Microsoft E...	6,725 ...
inputs.csv	2022-09-21 2...	Microsoft E...	829 KB
j1.cpp	2023-11-21 2...	C++ 源文件	6 KB
j2.cpp	2023-11-21 1...	C++ 源文件	14 KB
j3.cpp	2023-11-21 1...	C++ 源文件	14 KB
outputs.csv	2022-09-21 2...	Microsoft E...	6,892 ...
shared.h	2023-11-21 1...	C Header ...	1 KB
transactions.csv	2022-09-21 2...	Microsoft E...	2,481 ...
实验04、迷你区块链系统.d...	2023-11-13 8:...	Microsoft ...	86 KB

j1 对应客户端结点，j2，j3 是另外两个结点，数据分别在文件夹 node0 和 node1。

csv 文件是来自实验2中的完整数据。

`shared.h` 头文件原本用于在三个结点间共享高度变量，后来发现三者有不同的高度对应，所以最后没什么用，但是在 `j1` 中仍然使用了其中的变量。

`nothing` 文件夹是用于初始化 `node0` 和 `node1` 的，里面有空白的 `message`, `block_message`, `trans_pool` 和 `head`。