【大作业（一）】前期热身报告

姓名：陈灿辉　　学号：17343008　　班级：软工一班

实验环境

操作系统：macOS Mojave 10.14.6

由于没有使用课程提供的虚拟机，姑这里简单地介绍在macOS上配置实验环境的过程

安装依赖：

build_chain.sh `脚本依赖于` openssl, curl。macOS执行`brew install openssl curl`即可

- 创建操作目录

```
cd ~ && mkdir -p fisco && cd fisco
```

- 下载`build_chain.sh`脚本

```
bash <(curl -s https://raw.githubusercontent.com/FISCO-BCOS/FISCO-BCOS/master/tools/get_buildchain.sh)
```

私有链的搭建

由于在之前我就根据文档搭建了环境，姑部分环境搭建的过程在下面的报告中会有所省略

重新搭建链的时候，由于之前就有nodes文件夹，输出如下

```
[appledeMacBook-Air:fisco apple$ bash build_chain.sh -l "127.0.0.1:4" -p 30300,20
200,8545
[WARN] /Users/apple/fisco/nodes DIR exists, please clean old DIR!
```

为了重新搭链，我们把nodes文件夹删除。

在fisco目录下执行下面的指令，生成一条单群组4节点的FISCO链。请确保机器的`30300~30303，20200~20203，8545~8548`端口没有被占用

```
bash build_chain.sh -l "127.0.0.1:4" -p 30300,20200,8545
```

执行后的结果如下

```
[appledeMacBook-Air:fisco apple$ bash build_chain.sh -l "127.0.0.1:4" -p 30300,20
200,8545
[INFO] Downloading fisco-bcos binary from https://github.com/FISCO-BCOS/FISCO-BC
OS/releases/download/v2.1.0/fisco-bcos-macOS.tar.gz ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   614    0   614    0     0    564      0 --:--:--  0:00:01 --:--:--   564
  9 6284k    9  577k    0     0  26012      0  0:04:07  0:00:22  0:03:45  8222
curl: (28) Operation too slow. Less than 102400 bytes/sec transferred the last 2
0 seconds
[INFO] Download speed is too low, try https://www.fisco.com.cn/cdn/fisco-bcos/re
leases/download/v2.1.0/fisco-bcos-macOS.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 6284k  100 6284k    0     0  1169k      0  0:00:05  0:00:05 --:--:-- 1289k
```

命令执行成功会输出`All completed`

```
============================================================
Generating CA key...
============================================================
Generating keys ...
Processing IP:127.0.0.1 Total:4 Agency:agency Groups:1
============================================================
Generating configurations...
Processing IP:127.0.0.1 Total:4 Agency:agency Groups:1
============================================================
[INFO] Start Port        : 30300 20200 8545
[INFO] Server IP         : 127.0.0.1:4
[INFO] State Type        : storage
[INFO] RPC listen IP     : 127.0.0.1
[INFO] Output Dir        : /Users/apple/fisco/nodes
[INFO] CA Key Path       : /Users/apple/fisco/nodes/cert/ca.key
============================================================
[WARN] RPC listens 127.0.0.1 will cause nodes' JSON-RPC and Channel service to b
e inaccessible form other machines
[INFO] Execute the following command to get FISCO-BCOS console
 bash <(curl -s https://raw.githubusercontent.com/FISCO-BCOS/console/master/tool
s/download_console.sh)
============================================================
[INFO] All completed. Files in /Users/apple/fisco/nodes
appledeMacBook-Air:fisco apple$
```

这时候就会出现nodes文件夹，其文件结构如下所示，可以看到链上的节点的信息都储存在nodes文件夹下，一共有4个节点node0~3

```
[appledeMacBook-Air:fisco apple$ tree nodes/
nodes/
├── 127.0.0.1
│   ├── fisco-bcos
│   ├── node0
│   │   ├── conf
│   │   │   ├── ca.crt
│   │   │   ├── group.1.genesis
│   │   │   ├── group.1.ini
│   │   │   ├── node.crt
│   │   │   ├── node.key
│   │   │   └── node.nodeid
│   │   ├── config.ini
│   │   ├── data
│   │   │   └── group1
│   │   │       ├── block
│   │   │       │   └── RocksDB
│   │   │       │       ├── 000003.log
│   │   │       │       ├── CURRENT
│   │   │       │       ├── IDENTITY
│   │   │       │       └── LOCK
```

其文件结构如下所示

```
nodes/
├── 127.0.0.1
```

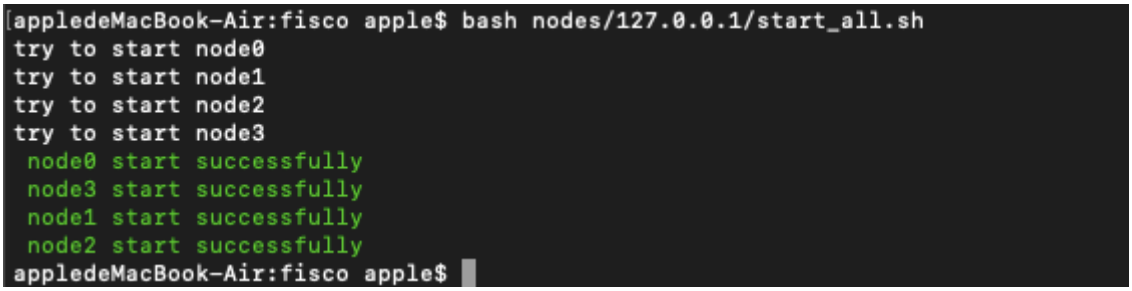
```
|   ├── fisco-bcos # 二进制程序
|   ├── node0 # 节点0文件夹
|   |   ├── conf # 配置文件夹
|   |   |   ├── ca.crt # 链根证书
|   |   |   ├── group.1.genesis # 群组1初始化配置，该文件不可更改
|   |   |   ├── group.1.ini # 群组1配置文件
|   |   |   ├── node.crt # 节点证书
|   |   |   ├── node.key # 节点私钥
|   |   |   └── node.nodeid # 节点id，公钥的16进制表示
|   |   ├── config.ini # 节点主配置文件，配置监听IP、端口等
|   |   ├── start.sh # 启动脚本，用于启动节点
|   |   └── stop.sh # 停止脚本，用于停止节点
|   ├── node1 # 节点1文件夹
|   |.....
|   ├── node2 # 节点2文件夹
|   |.....
|   ├── node3 # 节点3文件夹
|   |.....
|   ├── sdk # SDK需要用到的
|   |   ├── ca.crt # 链根证书
|   |   ├── sdk.crt # SKD所需的证书文件，建立连接时使用
|   |   └── sdk.key # SDK所需的私钥文件，建立连接时使用
├── cert # 证书文件夹
|   ├── agency # 机构证书文件夹
|   |   ├── agency.crt # 机构证书
|   |   ├── agency.key # 机构私钥
|   |   ├── agency.srl
|   |   ├── ca-agency.crt
|   |   ├── ca.crt
|   |   └── cert.cnf
|   ├── ca.crt # 链证书
|   ├── ca.key # 链私钥
|   ├── ca.srl
|   └── cert.cnf
```

启动FISCO BCOS链

启动所有节点

```
bash nodes/127.0.0.1/start_all.sh
```

启动成功会输出类似下面内容的响应。



检查进程是否启动

```
ps -ef | grep -v grep | grep fisco-bcos
```

结果如下，可以看到有4个进程运行起来

```
appledeMacBook-Air:fisco apple$ ps -ef | grep -v grep | grep fisco-bcos
  501  4550     1   0 11:21上午 ttys000    0:01.72 /Users/apple/fisco/nodes/127.0.0.1/node1/../fisco-bcos -c config.ini
  501  4551     1   0 11:21上午 ttys000    0:01.71 /Users/apple/fisco/nodes/127.0.0.1/node0/../fisco-bcos -c config.ini
  501  4554     1   0 11:21上午 ttys000    0:01.75 /Users/apple/fisco/nodes/127.0.0.1/node3/../fisco-bcos -c config.ini
  501  4555     1   0 11:21上午 ttys000    0:01.72 /Users/apple/fisco/nodes/127.0.0.1/node2/../fisco-bcos -c config.ini
appledeMacBook-Air:fisco apple$
```

检查日志输出

如下，查看节点node0链接的节点数

```
tail -f nodes/127.0.0.1/node0/log/log*  | grep connected
```

正常情况会不停地输出链接信息，从输出可以看出node0与另外3个节点有链接。

```
appledeMacBook-Air:fisco apple$ tail -f nodes/127.0.0.1/node0/log/log*  | grep connected
info|2019-10-19 11:26:20.300807|[P2P][Service] heartBeat,connected count=3
info|2019-10-19 11:26:30.302861|[P2P][Service] heartBeat,connected count=3
^Z
[1]+  Stopped                 tail -f nodes/127.0.0.1/node0/log/log* | grep connected
```

执行下面指令，检查是否在共识

```
tail -f nodes/127.0.0.1/node0/log/log*  | grep +++
```

正常情况会不停输出`++++Generating seal`，表示共识正常。

```
appledeMacBook-Air:fisco apple$ tail -f nodes/127.0.0.1/node0/log/log*  | grep +++
info|2019-10-19 11:27:53.413825|[g:1][CONSENSUS][SEALER]++++++++++++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=338cecbc...
info|2019-10-19 11:27:57.443469|[g:1][CONSENSUS][SEALER]++++++++++++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=129b15bd...
^Z
[2]+  Stopped                 tail -f nodes/127.0.0.1/node0/log/log* | grep +++
appledeMacBook-Air:fisco apple$
```

配置及使用控制台

由于我的电脑已经有相关环境及其配置了，这里就不再赘述环境配置的问题。

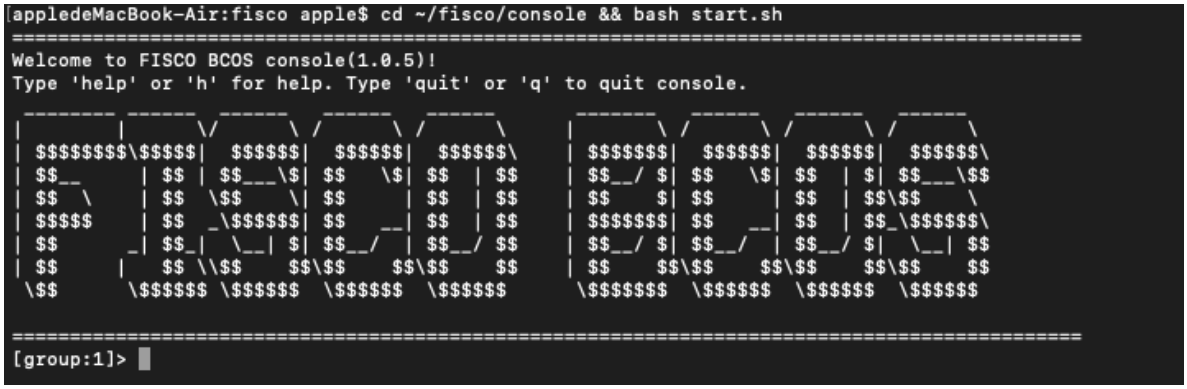
配置控制台证书

```
cp nodes/127.0.0.1/sdk/* console/conf/
```

直接启动控制台

```
cd ~/fisco/console && bash start.sh
```

启动后可以看到如下画面，这说明控制台已经成功启动了



使用控制台获取信息

获取客户端的版本

获取节点连接信息，如下，可以看到我们当前节点一共有3个peer，也就是当前网络上有4个节点（包含当前节点）

合约的部署

这里我使用的是样例的HelloWorld合约

HelloWorld合约提供两个接口，分别是`get()`和`set()`，用于获取/设置合约变量`name`。合约内容如下:

```
pragma solidity ^0.4.24;

contract HelloWorld {
    string name;

    function HelloWorld() {
        name = "Hello, World!";
    }

    function get()constant returns(string) {
```

```
        return name;
    }

    function set(string n) {
        name = n;
    }
}
```

HelloWorld合约已经内置于控制台中，位于控制台目录下`solidity/contracts/HelloWorld.sol`

```
[[group:1]> deploy HelloWorld                                                   ]
contract address: 0x6b7b56d7e35b74c74b81679deca93210e130c01a
```

调用合约的过程如下

```
# 查看当前块高
[group:1]> getBlockNumber
1

# 调用get接口获取name变量 此处的合约地址是deploy指令返回的地址
[group:1]> call HelloWorld 0xb3c223fc0bf6646959f254ac4e4a7e355b50a344 get
Hello, World!

# 查看当前块高，块高不变，因为get接口不更改账本状态
[group:1]> getBlockNumber
1

# 调用set设置name
[group:1]> call HelloWorld 0xb3c223fc0bf6646959f254ac4e4a7e355b50a344 set
"Hello, FISCO BCOS"
0x21dca087cb3e44f44f9b882071ec6ecfcb500361cad36a52d39900ea359d0895

# 再次查看当前块高，块高增加表示已出块，账本状态已更改
[group:1]> getBlockNumber
2

# 调用get接口获取name变量，检查设置是否生效
[group:1]> call HelloWorld 0xb3c223fc0bf6646959f254ac4e4a7e355b50a344 get
Hello, FISCO BCOS

# 退出控制台
[group:1]> quit
```

```
[[group:1]> getBlockNumber
1

[[group:1]>
[[group:1]> call HelloWorld 0x6b7b56d7e35b74c74b81679deca93210e130c01a get
Hello, World!

[[group:1]> call HelloWorld 0x6b7b56d7e35b74c74b81679deca93210e130c01a set "陈灿
辉"
transaction hash: 0xb25afbb79aee287c6a1084cbb399d91baee54653f162bc87ef61f906c797
ec6b

[[group:1]> getBlockNumber
2

[[group:1]> call HelloWorld 0x6b7b56d7e35b74c74b81679deca93210e130c01a ge t
The method gett with 0 parameter is undefined of the contract.

[[group:1]> call HelloWorld 0x6b7b56d7e35b74c74b81679deca93210e130c01a get
陈灿辉
```

加入新节点

原来的网络中有4个节点，分别是node0~3,这里我们添加多一个新节点node4

1 . 进入nodes同级目录，在该目录下拉取并执行`gen_node_cert.sh`生成节点目录，目录名以node4为例，node4内有`conf/`目录；

```
# 获取脚本
$ curl -LO https://raw.githubusercontent.com/FISCO-BCOS/FISCO-
BCOS/master/tools/gen_node_cert.sh && chmod u+x gen_node_cert.sh
# 执行，-c为生成节点所提供的ca路径，agency为机构名，-o为将生成的节点目录名
$ ./gen_node_cert.sh -c nodes/cert/agency -o node4
```

【注意：我本来是在mac下运行的，但是运行到这一步的时候发现~mac下面会有bug，坑了我好久。因此不得不转战到Linux下运行】

```
[INFO] All completed. Files in /home/fisco-bcos/fisco/nodes
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ ./gen_node_cert.sh -c nodes/cert/agenc
y -o node4
bash: ./gen_node_cert.sh: Permission denied
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ sudo ./gen_node_cert.sh -c nodes/cert/
agency -o node4
sudo: ./gen_node_cert.sh: command not found
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ sudo bash ./gen_node_cert.sh -c nodes/
cert/agency -o node4
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ ls
build_chain.sh  build.log  gen_node_cert.sh  node4  nodes
```

2 . 拷贝node4到`nodes/127.0.0.1/`下，与其他节点目录（`node0`、`node1`）同级；

```
$ cp -r ./node4 nodes/127.0.0.1/
```

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ sudo cp -r ./node4 nodes/127.0.0.1/
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ ls
build_chain.sh  build.log  gen_node_cert.sh  node4  nodes
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ cd nodes
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes$ ls
127.0.0.1  cert
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes$ cd 127.0.0.1/
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ ls
fisco-bcos  node0  node1  node2  node3  node4  sdk  start_all.sh  stop_all.sh
```

3 . 进入`nodes/127.0.0.1/`，拷贝`node0/config.ini`、`node0/start.sh`和`node0/stop.sh`到node2目录;

```
$ cd nodes/127.0.0.1/
$ cp node0/config.ini node0/start.sh node0/stop.sh node4/
```

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ cp node0/config.ini no
de0/start.sh node0/stop.sh node4/
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ ls
fisco-bcos  node0  node1  node2  node3  node4  sdk  start_all.sh  stop_all.sh
```

4 . 修改`node4/config.ini`。对于`[rpc]`模块，修改`listen_ip`、`channel_listen_port`和`jsonrpc_listen_port`；对于`[p2p]`模块，修改`listen_port`并在`node.`中增加自身节点信息;

相关修改如下

```
[rpc]
    listen_ip=127.0.0.1
    channel_listen_port=20204
    jsonrpc_listen_port=8549
[p2p]
    listen_ip=0.0.0.0
    listen_port=30304
    ;enable_compress=true
    ; nodes to connect
    node.0=127.0.0.1:30300
    node.1=127.0.0.1:30301
    node.2=127.0.0.1:30302
    node.3=127.0.0.1:30303
    node.4=127.0.0.1:30304
```

5 . 节点3拷贝节点1的`node1/conf/group.3.genesis`（内含**群组节点初始列表**）和`node1/conf/group.3.ini`到`node4/conf`目录下，不需改动;

```
$ cp node1/conf/group.3.genesis node4/conf/
$ cp node1/conf/group.3.ini node4/conf/
```

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ cp node1/conf/group.1.
genesis node4/conf/
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ cp node1/conf/group.1.
ini node4/conf
conf/       config.ini
```

移动后文件夹内容如下

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1/node4/conf$ ls
agency.crt  group.1.genesis  node.crt  node.nodeid
ca.crt      group.1.ini      node.key  node.private
```

此时运行,

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ sudo bash node4/start.
sh
 node4 start successfully
```

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ sudo bash node4/start.sh
 node4 start successfully
fisco-bcos@fiscobcos-VirtualBox:~/fisco/nodes/127.0.0.1$ tail -f node4/log/log* | grep P2P
info|2019-10-20 00:04:19.712467|[P2P][Service] Connection established,nodeID=80cc057a...,endpoint=127.0.0.1:30301
info|2019-10-20 00:04:19.712813|[P2P][Service] Connection established,nodeID=39a98659...,endpoint=127.0.0.1:30302
info|2019-10-20 00:04:29.464401|[P2P][Service] heartBeat,connected count=4
info|2019-10-20 00:04:39.464931|[P2P][Service] heartBeat,connected count=4
info|2019-10-20 00:04:49.465192|[P2P][Service] heartBeat,connected count=4
info|2019-10-20 00:04:59.465426|[P2P][Service] heartBeat,connected count=4
info|2019-10-20 00:05:09.465582|[P2P][Service] heartBeat,connected count=4
```

可以看到新加入的节点4已经与原有的节点建立连接，并加入到网络中

区块字段解释

第一个区块的字段

```
[group:1]> getBlockByNumber 1
{
    "dbHash":"0xd9eda29a1c2c631636e990d9b3642ca37e0395456a3723bfddce4c7558fcd010",
    "extraData":[

    ],
    "gasLimit":"0x0",
    "gasUsed":"0x0",
    "hash":"0x01ec041d65de3c6025365d53a2aa866f75c01253b6f436f2509a634880004cd6",
    "logsBloom":"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
    "number":"0x1",
    "parentHash":"0xee4fda29379b5f55522929eff2d3a9eafdb3ef4a19e6bec8daef9dbdae251462",
    "receiptsRoot":"0xc2ecb86f8089b30e927795e3e9c7192ac9c7c335b882fcb6cc9ab56b0266e8e1",
    "sealer":"0x0",
    "sealerList":[
        "5d51003ac452803c6e0a022de4942703da04bd66de8861bc83af30356eb7ef0fa7dc86ac2c29aacfea5f4db79b335ca0422e02e3998c5aa979be0a517bf2ed5b",
        "a0000db8f5733dad713f6fcda75b2b19285d68008ae88d44a88b22b0050fb7a89550ef6aaaf884faa2a45351df3619f0e155ac15031109eb6e34bf5ccc43c6df",
        "cbe7a4d2f03786ea34ed20fbef3b6fc690c4d43e49fead7054b92fc1ba4cf7f3ed7207e266154373de2ac6dcdd00091893e322e1037d9ae1135f2c8c76d94bcf",
        "fb8f6d9d860e0cb12a16df1e03752a471009b5e6b989e05490777c41adc59f8beec680b429b12904b18477e6c06ad32dabce4f8bc59fb756fc6894c90a908b2b"
    ],
    "stateRoot":"0xd9eda29a1c2c631636e990d9b3642ca37e0395456a3723bfddce4c7558fcd010",
    "timestamp":"0x16de2ed8358",
    "transactions":[
        "0x4880ba6d065bf87784d6ba8c84e8dce938c0a665181cdfbf8351066736fcac3e"
    ],
    "transactionsRoot":"0x95e862e90e5c97051426f798c3bc2b0c4160210d58939b40af68626b7ccb49ae"
}
```

第二个区块的字段

```
[group:1]> getBlockByNumber 2
{
    "dbHash":"0x8402361b530ca0630186ea8357be47ac9e89e49adae55d460f8b08d55405f8c2",
    "extraData":[

    ],
    "gasLimit":"0x0",
    "gasUsed":"0x0",
    "hash":"0x59229111616d29b8fde2178812518740d7697f5d268a12ce129857697bb3c44b",
    "logsBloom":"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
    "number":"0x2",
    "parentHash":"0x01ec041d65de3c6025365d53a2aa866f75c01253b6f436f2509a634880004cd6",
    "receiptsRoot":"0xd1f57b3299437a607bec6ceb0fed2a3b98196cf85db0b8f645f72bd86cd10a0c",
    "sealer":"0x2",
    "sealerList":[
        "5d51003ac452803c6e0a022de4942703da04bd66de8861bc83af30356eb7ef0fa7dc86ac2c29aacfea5f4db79b335ca0422e02e3998c5aa979be0a517bf2ed5b",
        "a0000db8f5733dad713f6fcda75b2b19285d68008ae88d44a88b22b0050fb7a89550ef6aaaf884faa2a45351df3619f0e155ac15031109eb6e34bf5ccc43c6df",
        "cbe7a4d2f03786ea34ed20fbef3b6fc690c4d43e49fead7054b92fc1ba4cf7f3ed7207e266154373de2ac6dcdd00091893e322e1037d9ae1135f2c8c76d94bcf",
        "fb8f6d9d860e0cb12a16df1e03752a471009b5e6b989e05490777c41adc59f8beec680b429b12904b18477e6c06ad32dabce4f8bc59fb756fc6894c90a908b2b"
    ],
    "stateRoot":"0x8402361b530ca0630186ea8357be47ac9e89e49adae55d460f8b08d55405f8c2",
    "timestamp":"0x16de2ee9648",
    "transactions":[
        "0xb25afbb79aee287c6a1084cbb399d91baee54653f162bc87ef61f906c797ec6b"
    ],
    "transactionsRoot":"0x9b9db6a523495d49e06643450b30cecda23e1f9e443f06fc3057ed526cfd0ac5"
}
```

- dbHash
- extraData: 字符串，额外数据，可以包含个性信息，或者是附加说明。
- gasLimit:Number，设置对gas的消耗总量限制，用来限制区块能包含的交易信息总和
- gasUsed:Number，当前区块累计使用的总的gas，
- hash:字符串，区块的哈希值，可以看到两个hash是不同的
- logsBloom:字符串，由日志信息组成的一个Bloom过滤器(数据结构)bloom过滤器是用来快速 的查找log的
- number:当前区块的计数(从创世区块1开始递增1)
- parentHash:字符串，32字节，父区块的Hash值。Block2的parentHash的值等于Block1的Hash

- receiptsRoot：此区块所有交易收据的树的根节点Hash值
- sealer,sealer对应的是Group member Consensus nodes (**Sealer**)也就是共识节点。也就是对应着发布这个区块的共识节点编号。
- sealerList：网络上共识节点列表。本测试系统中对应的就是一开始生成的4个节点。
- stateRoot:字符串，32字节，此区块最终状态树根节点的Hash值，Merkel tree的结构，添加一 个区块后根节点也会随之发生变换
- timestamp:Number，此区块初始化时的unix的时间戳
- transactions:数组，交易对象，或32字节的交易哈希
- transactionsRoot:字符串，32字节，此区块的所有交易组成的树的根节点Hash值

可以发现这里面没有nonce，这是因为FISCO BCOS是联盟链，在联盟链上采用的共识机制并不是PoW

构建第一个区块链应用--练习

这部分我是在完成前面的学习后，根据官方文档进一步练习学习的。

我主要通过这部分了解到一下内容：

1. 如何将一个业务场景的逻辑用合约的形式表达
2. 如何将Solidity合约转化成Java类
3. 如何配置Web3SDK
4. 如何构建一个应用，并集成Web3SDK到应用工程
5. 如何通过Web3SDK调用合约接口，了解Web3SDK调用合约接口的原理

区块链资产管理的合约如下

```
pragma solidity ^0.4.24;

import "./Table.sol";

contract Asset {
    // event
    event RegisterEvent(int256 ret, string account, uint256 asset_value);
    event TransferEvent(int256 ret, string from_account, string to_account,
uint256 amount);

    constructor() public {
        // 构造函数中创建t_asset表
        createTable();
    }

    function createTable() private {
        TableFactory tf = TableFactory(0x1001);
        // 资产管理表, key : account, field : asset_value
        // |  资产账户(主键)      |     资产金额       |
        // |-------------------- |-------------------|
        // |        account      |    asset_value    |
        // |---------------------|-------------------|
        //
        // 创建表
        tf.createTable("t_asset", "account", "asset_value");
    }

    function openTable() private returns(Table) {
        TableFactory tf = TableFactory(0x1001);
        Table table = tf.openTable("t_asset");
        return table;
```

```
    }

    /*
    描述 : 根据资产账户查询资产金额
    参数 :
            account : 资产账户

    返回值:
            参数一: 成功返回0, 账户不存在返回-1
            参数二: 第一个参数为0时有效, 资产金额
    */
    function select(string account) public constant returns(int256, uint256) {
        // 打开表
        Table table = openTable();
        // 查询
        Entries entries = table.select(account, table.newCondition());
        uint256 asset_value = 0;
        if (0 == uint256(entries.size())) {
            return (-1, asset_value);
        } else {
            Entry entry = entries.get(0);
            return (0, uint256(entry.getInt("asset_value")));
        }
    }

    /*
    描述 : 资产注册
    参数 :
            account : 资产账户
            amount  : 资产金额
    返回值:
            0  资产注册成功
            -1 资产账户已存在
            -2 其他错误
    */
    function register(string account, uint256 asset_value) public
returns(int256){
        int256 ret_code = 0;
        int256 ret= 0;
        uint256 temp_asset_value = 0;
        // 查询账户是否存在
        (ret, temp_asset_value) = select(account);
        if(ret != 0) {
            Table table = openTable();

            Entry entry = table.newEntry();
            entry.set("account", account);
            entry.set("asset_value", int256(asset_value));
            // 插入
            int count = table.insert(account, entry);
            if (count == 1) {
                // 成功
                ret_code = 0;
            } else {
                // 失败? 无权限或者其他错误
                ret_code = -2;
            }
        } else {
```

```
            // 账户已存在
            ret_code = -1;
        }

        emit RegisterEvent(ret_code, account, asset_value);

        return ret_code;
    }

    /*
    描述 : 资产转移
    参数 :
            from_account : 转移资产账户
            to_account :  接收资产账户
            amount :  转移金额
    返回值:
            0  资产转移成功
            -1 转移资产账户不存在
            -2 接收资产账户不存在
            -3 金额不足
            -4 金额溢出
            -5 其他错误
    */
    function transfer(string from_account, string to_account, uint256 amount)
public returns(int256) {
        // 查询转移资产账户信息
        int ret_code = 0;
        int256 ret = 0;
        uint256 from_asset_value = 0;
        uint256 to_asset_value = 0;

        // 转移账户是否存在?
        (ret, from_asset_value) = select(from_account);
        if(ret != 0) {
            ret_code = -1;
            // 转移账户不存在
            emit TransferEvent(ret_code, from_account, to_account, amount);
            return ret_code;

        }

        // 接受账户是否存在?
        (ret, to_asset_value) = select(to_account);
        if(ret != 0) {
            ret_code = -2;
            // 接收资产的账户不存在
            emit TransferEvent(ret_code, from_account, to_account, amount);
            return ret_code;
        }

        if(from_asset_value < amount) {
            ret_code = -3;
            // 转移资产的账户金额不足
            emit TransferEvent(ret_code, from_account, to_account, amount);
            return ret_code;
        }

        if (to_asset_value + amount < to_asset_value) {
```

```
            ret_code = -4;
            // 接收账户金额溢出
            emit TransferEvent(ret_code, from_account, to_account, amount);
            return ret_code;
        }

        Table table = openTable();

        Entry entry0 = table.newEntry();
        entry0.set("account", from_account);
        entry0.set("asset_value", int256(from_asset_value - amount));
        // 更新转账账户
        int count = table.update(from_account, entry0, table.newCondition());
        if(count != 1) {
            ret_code = -5;
            // 失败? 无权限或者其他错误?
            emit TransferEvent(ret_code, from_account, to_account, amount);
            return ret_code;
        }

        Entry entry1 = table.newEntry();
        entry1.set("account", to_account);
        entry1.set("asset_value", int256(to_asset_value + amount));
        // 更新接收账户
        table.update(to_account, entry1, table.newCondition());

        emit TransferEvent(ret_code, from_account, to_account, amount);

        return ret_code;
    }
}
```

合约编译

Java程序无法直接调用Solidity合约，需要先将Solidity合约文件编译为Java文件。

控制台提供了编译工具，可以将`Asset.sol`合约文件存放在`console/contracts/solidity`目录。利用console目录下提供的`sol2java.sh`脚本进行编译，操作如下：

```
# 切换到fisco/console/目录
$ cd ~/fisco/console/
# 编译合约，后面指定一个Java的包名参数，可以根据实际项目路径指定包名
$ ./sol2java.sh org.fisco.bcos.asset.contract
```

执行结果如下所示

```
appledeMacBook-Air:console apple$ ./sol2java.sh org.fisco.bcos.asset.contract

Compile solidity contract files to java contract files successfully!
```

运行成功之后，将会在`console/contracts/sdk`目录生成java、abi和bin目录，如下所示。

```
|-- abi # 生成的abi目录，存放solidity合约编译生成的abi文件
|   |-- Asset.abi
|   |-- Table.abi
|-- bin # 生成的bin目录，存放solidity合约编译生成的bin文件
|   |-- Asset.bin
|   |-- Table.bin
```

```
|-- contracts # 存放solidity合约源码文件，将需要编译的合约拷贝到该目录下
|   |-- Asset.sol # 拷贝进来的Asset.sol合约，依赖Table.sol
|   |-- Table.sol # 默认提供的系统CRUD合约接口文件
|-- java  # 存放编译的包路径及Java合约文件
|   |-- org
|        |--fisco
|             |--bcos
|                  |--asset
|                       |--contract
|                            |--Asset.java  # Asset.sol合约生成的Java文件
|                            |--Table.java  # Table.sol合约生成的Java文件
|-- sol2java.sh
```

java目录下生成了`org/fisco/bcos/asset/contract/`包路径目录，该目录下包含`Asset.java`和`Table.java`两个文件，其中`Asset.java`是Java应用调用`Asset.sol`合约需要的文件。

`Asset.java`的主要接口：

```
package org.fisco.bcos.asset.contract;

public class Asset extends Contract {
    // Asset.sol合约 transfer接口生成
    public RemoteCall<TransactionReceipt> transfer(String from_account, String
to_account, BigInteger amount);
    // Asset.sol合约 register接口生成
    public RemoteCall<TransactionReceipt> register(String account, BigInteger
asset_value);
    // Asset.sol合约 select接口生成
    public RemoteCall<Tuple2<BigInteger, BigInteger>> select(String account);

    // 加载Asset合约地址，生成Asset对象
    public static Asset load(String contractAddress, Web3j web3j, Credentials
credentials, ContractGasProvider contractGasProvider);

    // 部署Assert.sol合约，生成Asset对象
    public static RemoteCall<Asset> deploy(Web3j web3j, Credentials credentials,
ContractGasProvider contractGasProvider);
}
```

查看生成的Java文件后发现，前面有一长串的二进制String

```
@SuppressWarnings("unchecked")
public class Asset extends Contract {
    public static String BINARY = "608060405234801561001057600080fd5b50610028610
02d6401000000000264010000000009004565b610185565b60006110019050807affffffffffffff
ffffffffffffffffffffffffff166356004b6a6040518163ffffffff167c010000000000000000000
0000000000000000000000000000000000000000281526004018080602001806020018060200184810
3845260078152602001807f745f617373657400000000000000000000000000000000000000000
```

这个应该对应的就是solidity合约编译后生成的二进制内容

SDK配置

获取Java工程项目

```
# 获取Java工程项目压缩包
$ cd ~
$ curl -LO https://github.com/FISCO-BCOS/LargeFiles/raw/master/tools/asset-app.tar.gz
# 解压得到Java工程项目asset-app目录
$ tar -zxf asset-app.tar.gz
```

asset-app项目的目录结构如下：

```
|-- build.gradle // gradle配置文件
|-- gradle
|   |-- wrapper
|       |-- gradle-wrapper.jar // 用于下载Gradle的相关代码实现
|       |-- gradle-wrapper.properties // wrapper所使用的配置信息，比如gradle的版本等信息
|-- gradlew // Linux或者Unix下用于执行wrapper命令的Shell脚本
|-- gradlew.bat // Windows下用于执行wrapper命令的批处理脚本
|-- src
|   |-- main
|   |   |-- java
|   |         |-- org
|   |             |-- fisco
|   |                   |-- bcos
|   |                         |-- asset
|   |                               |-- client // 放置客户端调用类
|   |                                       |-- AssetClient.java
|   |                               |-- contract // 放置Java合约类
|   |                                       |-- Asset.java
|   |-- test
|       |-- resources // 存放代码资源文件
|           |-- applicationContext.xml // 项目配置文件
|           |-- contract.properties // 存储部署合约地址的文件
|           |-- log4j.properties // 日志配置文件
|           |-- contract //存放solidity约文件
|                   |-- Asset.sol
|                   |-- Table.sol
|
|-- tool
    |-- asset_run.sh // 项目运行脚本
```

拷贝区块链节点对应的SDK证书

```
# 进入~目录
# 拷贝节点证书到项目的资源目录
$ cd ~
$ cp fisco/nodes/127.0.0.1/sdk/* asset-app/src/test/resources/
```

然后进行编译

```
# 切换到项目目录
$ cd ~/asset-app
# 编译项目
$ ./gradlew build
```

编译结果如下所示

```
[appledeMacBook-Air:asset-app apple$ ./gradlew build
Downloading https://services.gradle.org/distributions/gradle-5.6.2-bin.zip
..........................................................................
.........

Welcome to Gradle 5.6.2!

Here are the highlights of this release:
 - Incremental Groovy compilation
 - Groovy compile avoidance
 - Test fixtures for Java projects
 - Manage plugin versions via settings script

For more details see https://docs.gradle.org/5.6.2/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 5m 22s
4 actionable tasks: 4 executed
```

在部署之前，将节点运行起来

```
[appledeMacBook-Air:fisco apple$ bash nodes/127.0.0.1/start_all.sh
try to start node0
try to start node1
try to start node2
try to start node3
 node0 start successfully
 node1 start successfully
 node3 start successfully
 node2 start successfully
```

部署节点，并且调用其相关函数的过程如下所示

```
[appledeMacBook-Air:dist apple$ bash asset_run.sh deploy
 deploy Asset success, contract address is 0xb299750c6bf06202ac70be4f02715c8ef5b
e0411
[appledeMacBook-Air:dist apple$ ls
apps            conf            lib
asset_run.sh    contract        log
[appledeMacBook-Air:dist apple$ code asset_run.sh
[appledeMacBook-Air:dist apple$ bash asset_run.sh register Alice 100000
 register asset account success => asset: Alice, value: 100000
[appledeMacBook-Air:dist apple$ bash asset_run.sh register Bob 100000
 register asset account success => asset: Bob, value: 100000
[appledeMacBook-Air:dist apple$ bash asset_run.sh query Alice
 asset account Alice, value 100000
[appledeMacBook-Air:dist apple$ bash asset_run.sh query Bob
 asset account Bob, value 100000
[appledeMacBook-Air:dist apple$ bash asset_run.sh transfer Alice Bob 50000
 transfer success => from_asset: Alice, to_asset: Bob, amount: 50000
[appledeMacBook-Air:dist apple$ bash asset_run.sh query Alice
 asset account Alice, value 50000
```

在另外一个终端中开启console，也调用合约相关函数

```
[[group:1]> call Asset 0xb299750c6bf06202ac70be4f02715c8ef5be0411 select "Alice"
[0, 50000]
```

同时还可以查看到区块的信息，如下为转账交易对应的区块

```
[[group:1]> getBlockByNumber 4
{
    "dbHash":"0x3efba4b68c87027be24d19642bb6c872fe1b1ce1761a8e1ee201874cc4af6d0e
",
    "extraData":[

    ],
    "gasLimit":"0x0",
    "gasUsed":"0x0",
    "hash":"0xa3afcf9c5384ab16405bdc8411d8fe19bfbb1e195ea3dc98aa2ae1a82b4d9bd0",
    "logsBloom":"0x00000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000",
    "number":"0x4",
    "parentHash":"0x5a7a4db3537c0046e012af3cf890a7b84e1cf1253b7a91cb4fdd6bc1c845
3342",
    "receiptsRoot":"0x473f6724637604be6c244a1e2e1af0ea86201bf5affdefe37fe3fec370
607547",
    "sealer":"0x1",
    "sealerList":[
        "87dd5a9b530df90a0b336b4f00b4d827c9ebeca613a518a48a833bb6147c9beef6d30e7
9d533f82c1da58679fac7508b4a4391bec59977e8256be374cbe9335f",
        "9ac0b0ef921dab0657dd6cc0fb8481b6e7e223ccb333bcc86417003924ec0a6d65c66ec
cff4388a3924329d1e000c81da6af371e8d66701c1cd53fea6e671bcc",
        "9c8d813f24784925630350f2561f74f780abfd504d385403f6c740ffe73ceb458b93770
713460c57b8a99740abd1511b8dfc58a490c7a00d9f5bc1c6623b391f",
        "cae4cbbe33e69cad63b4efc75d786afe59a157062ffda33fd0cd8ac17d5edc61aca188c
70b8246a8d7a3c4a4e9fa61dca31e7841069db07940ecf56538f569f9"
    ],
    "stateRoot":"0x3efba4b68c87027be24d19642bb6c872fe1b1ce1761a8e1ee201874cc4af6
d0e",
    "timestamp":"0x16de4174349",
    "transactions":[
```

实验小结

通过这次实验，我在自己的Mac电脑上配置了FISCO BCOS的环境。并且在阅读文档的过程中，我培养锻炼了阅读文档和实操的能力。在实验过程中，也发现了FISCO BCOS与以太坊的区别。我们之前学习的比特币和以太坊都是公有链，基于POW的思想来维护整个区块链系统的共识。但是FISCO BCOS是联盟链，同时使用了不同的共识机制来维护区块链系统的共识，这一点与以太坊有着明显的区别。但是区块链系统的本质等还是差不多的，因此在实验过程中还是发现FISCO BCOS与以太坊等有着较为相似的地方。初步热身，让我对FISCO BCOS系统有了初步的了解。相信随着对该系统了解的不断深入，最后一定能做出一个不错的大作业的。