

# 《区块链》期末 project 热身报告

陈铭涛  
16340024

## 1.以太坊的安装、私有链创世区块搭建、私有链节点的加入

### 安装

本次使用 macOS 宿主机系统与在 Virtual Box 下运行的 Ubuntu 虚拟机系统作为两个节点进行。使用的以太坊客户端为 geth，使用从源码编译方式安装。

安装前的开发环境配置安装有 go 1.10.1，在目标目录下使用 git clone 获取 GitHub 上的 go-ethereum 仓库：

```
git clone https://github.com/ethereum/go-ethereum
```

在仓库目录下使用以下命令完成 geth 编译：

```
make geth
```

在 Ubuntu 下执行编译若出现错误，需从源码找到 vendor/github.com/karalabe/hid/hid\_enabled.go 文件并在其代码中的 cgo 编译参数找到并修改如下语句然后重新进行编译：

```
#cgo linux,!android LDFLAGS: -lrt -liconv
```

完成编译后将仓库目录下的 build/bin/geth 文件夹加入环境 PATH 中即可使用。

### 私有链创世区块搭建

使用类似如下的 genesis.json 文件作为创世区块的配置（注释为各值的作用解释，实际使用 JSON 时需删去）：

```
{
  "config": {
    // 用于防止 Replay Attack
    "chainId": 666,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  }
}
```

```

    },
    //用于在创世区块直接向特定的地址加入指定数量的 Wei
    "alloc"      : {},
    //成功挖出区块时奖励转入的地址
    "coinbase"   : "0x0000000000000000000000000000000000000000",
    //挖矿过程中查找 nonce 值的难度
    "difficulty" : "0x00000001",
    //创世区块的 extraData, 每个挖出的区块都可以有最长 32 字节的 extraData
    "extraData"  : "",
    //最高 gas 花费
    "gasLimit"   : "0xffffffff",
    //nonce 为 PoW 挖矿时的数, 与 mixhash 结合以验证一个区块是否正确被挖出
    "nonce"      : "0x0000000000000042",
    "mixhash"    :
    "0x0000000000000000000000000000000000000000000000000000000000000000",
    //上一区块头的哈希
    "parentHash" :
    "0x0000000000000000000000000000000000000000000000000000000000000000",
    //区块时间戳
    "timestamp"  : "0x00"
}

```

使用如下命令完成创世区块创建：

```
geth --datadir data init genesis.json
```

其中 --datadir 后接区块链数据存放的目录，此处为当前目录下的 data 文件夹，也可改为其他文件夹。

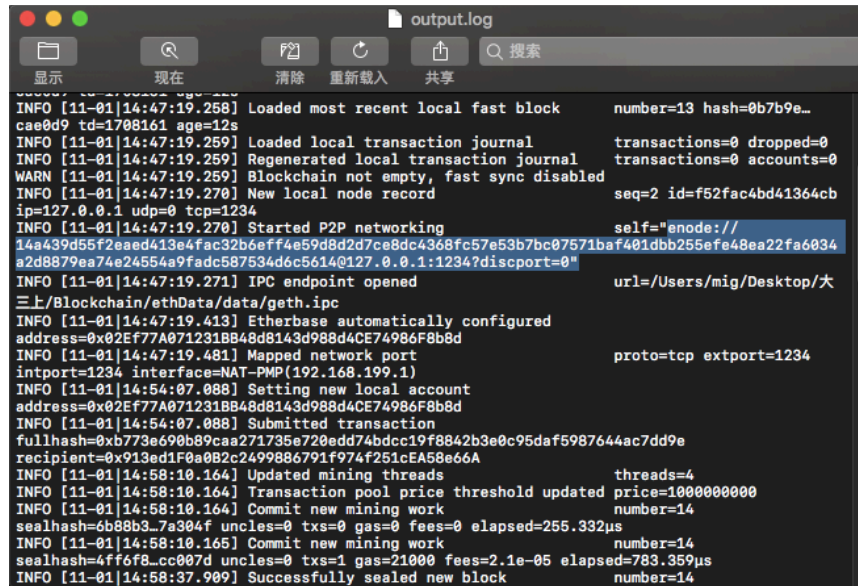
## 私有链节点的加入

在 macOS 和 Ubuntu 中都完成创世区块创建后使用如下命令启动 geth console：

```
geth --datadir data --nodiscover --port 1234 console 2>output.log
```

其中 --nodiscover 参数用于防止客户端自动连入其他 P2P 节点，--port 后接端口号，将程序的日志输出输出至 output.log 文件。

在 macOS 下的日志文件中找到节点 enode：



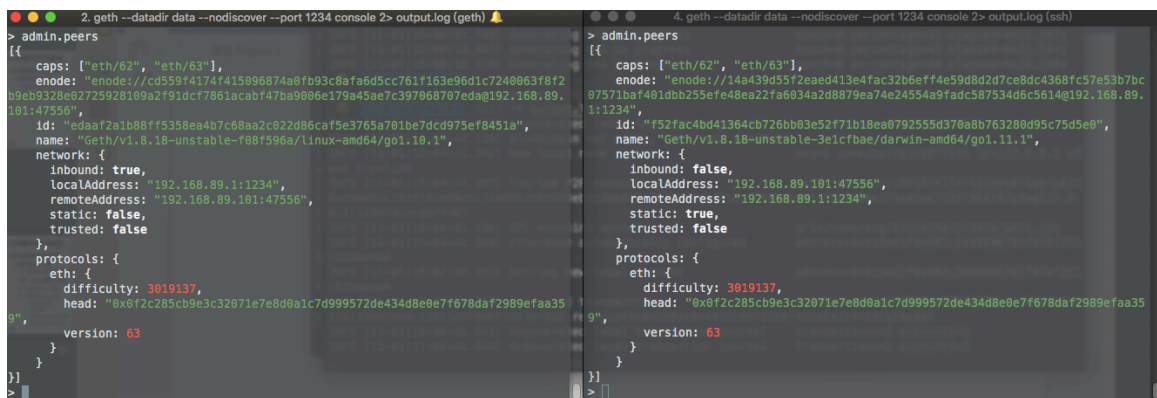
```
INFO [11-01|14:47:19.258] Loaded most recent local fast block   number=13 hash=0b7b9e...
cae0d9 td=1708161 age=12s
INFO [11-01|14:47:19.259] Loaded local transaction journal      transactions=0 dropped=0
INFO [11-01|14:47:19.259] Regenerated local transaction journal  transactions=0 accounts=0
WARN [11-01|14:47:19.259] Blockchain not empty, fast sync disabled
INFO [11-01|14:47:19.270] New local node record                 seq=2 id=f52fac4bd41364cb
ip=127.0.0.1 udp=0 tcp=1234
INFO [11-01|14:47:19.270] Started P2P networking                self="enode://
14a439d55f2eae413e4fac32b6eff4e59d8d2d7ce8dc4368fc57e53b7bc07571baf401dbb255efe48ea22fa6034a2d8879ea74e24554a9fadc587534d6c5614@127.0.0.1:1234?discport=0"
INFO [11-01|14:47:19.271] IPC endpoint opened                   url=/Users/mig/Desktop/大
三上/Blockchain/ethData/data/geth.ipc
INFO [11-01|14:47:19.413] Etherbase automatically configured
address=0x02Ef77A071231BB48d8143d988d4CE74986F8b8d
INFO [11-01|14:47:19.481] Mapped network port                  proto=tcp extport=1234
intport=1234 interface=NAT-PMP(192.168.199.1)
INFO [11-01|14:54:07.088] Setting new local account
address=0x02Ef77A071231BB48d8143d988d4CE74986F8b8d
INFO [11-01|14:54:07.088] Submitted transaction
fullhash=0xb773e690b89caa271735e720edd74bdcc19f8842b3e0c95daf5987644ac7dd9e
recipient=0x913ed1F0a0B2c2499886791f974f251cEA58e66A
INFO [11-01|14:58:10.164] Updated mining threads               threads=4
INFO [11-01|14:58:10.164] Transaction pool price threshold updated price=100000000
number=14
INFO [11-01|14:58:10.164] Commit new mining work               sealhash=6b88b3_7a304f uncles=0 txs=0 gas=0 fees=0 elapsed=255.332µs
number=14
INFO [11-01|14:58:10.165] Commit new mining work               sealhash=4ff6f8_cc007d uncles=0 txs=1 gas=21000 fees=2.1e-05 elapsed=783.359µs
number=14
INFO [11-01|14:58:37.909] Successfully sealed new block        number=14
```

在虚拟机下的geth console 中使用如下代码添加节点：

```
admin.addPeer("enode://14a439d55f2eae413e4fac32b6eff4e59d8d2d7ce8dc4368fc57e53b7bc07571baf401dbb255efe48ea22fa6034a2d8879ea74e24554a9fadc587534d6c5614@192.168.89.1:1234")
```

函数的参数为添加的节点的 enode，后面需加上添加节点的 ip 与端口。

若成功查看 admin.peers 便可以看到添加的节点：



```
> admin.peers
[{"caps": ["eth/62", "eth/63"],
  enode: "enode://cd559f4174f415096874a0fb93c8fa6d5cc761f163e96d1c7240063f8f2b9eb9328e02725920109a2f91dcf7861acabf47ba90806e179a45ae7c397068707eda@192.168.89.101:147556",
  id: "edanf2a1b88f15358ea4b7c68aa2c022d06caf5e3765a701be7dcd975ef8451a",
  name: "Geth/v1.8.18-unstable-f80f996a/linux-amd64/go1.10.1",
  network: {
    inbound: true,
    localAddress: "192.168.89.1:1234",
    remoteAddress: "192.168.89.101:147556",
    static: false,
    trusted: false
  },
  protocols: {
    eth: {
      difficulty: 3019137,
      head: "0x0f2c285cb9e3c32071e7e8d0a1c7d999572de434d8e0e7f678daf2989efaa359",
      version: 63
    }
  }
}]

> admin.peers
[{"caps": ["eth/62", "eth/63"],
  enode: "enode://14a439d55f2eae413e4fac32b6eff4e59d8d2d7ce8dc4368fc57e53b7bc07571baf401dbb255efe48ea22fa6034a2d8879ea74e24554a9fadc587534d6c5614@192.168.89.1:1234",
  id: "f52fac4bd41364cb726bb03e2f71b18ea0792355d370a8b763280d95c75d5e0",
  name: "Geth/v1.8.18-unstable-3e1cfbae/darwin-amd64/go1.11.1",
  network: {
    inbound: false,
    localAddress: "192.168.89.101:47556",
    remoteAddress: "192.168.89.1:1234",
    static: true,
    trusted: false
  },
  protocols: {
    eth: {
      difficulty: 3019137,
      head: "0x0f2c285cb9e3c32071e7e8d0a1c7d999572de434d8e0e7f678daf2989efaa359",
      version: 63
    }
  }
}]
```

(图中左为 macOS 下的结果，右为使用 ssh 连接至虚拟机完成操作后的结果，虚拟机的 ip 地址为192.168.89.101)

## 2.对getBlock中所得区块的各个字段进行解释

getBlock 的结果如图：

[illegible]

各字段的作用分别如下：

- difficulty: 当前区块的难度
- extraData: 当前区块的32字节额外信息
- gasLimit: 当前区块允许的最大 gas 花费
- gasUsed: 当前区块所有 transaction 所使用的总 gas
- hash: 区块的哈希值
- logsBloom: 区块日志的布隆过滤器
- miner: 挖掘出该区块的矿工地址

- mixHash: 用于 PoW 验证区块有效性。
- nonce: 与 mixHash 结合用于验证区块的有效性
- number: 区块的编号
- parentHash: 上一区块的哈希
- receiptsRoot: transaction 结果的 trie 根
- sha3Uncles: 区块 uncles 的 sha3结果
- size: 区块大小, 单位为字节
- stateRoot: 区块最终状态的trie 根
- timestamp: 区块时间戳
- totalDifficulty: 区块链到此区块为止的总难度
- transactions: 当前区块记录的 transaction 的 hash 列表
- transactionsRoot: 当前区块的 transaction trie 的根
- uncles: uncle blocks 的列表, uncle blocks 是上一区块为当前区块祖先 (前6个区块内) 的其他区块。

### 3.对日志输出进行解释

日志主要的几种输出如下：

- a. 客户端程序启动时日志如下：

```
output.log
显示 现在 清除 重新载入 共享
INFO [11-01|14:47:19.222] Maximum peer count ETH=25 LES=0 total=25
INFO [11-01|14:47:19.236] Starting peer-to-peer node instance=Geth/v1.8.18-unstable-3e1cfbae/darwin-amd64/go1.11.1
INFO [11-01|14:47:19.236] Allocated cache and file handles database=/Users/mig/Desktop/大三上/Blockchain/ethData/data/geth/chaindata cache=768 handles=1024
INFO [11-01|14:47:19.257] Initialised chain configuration config="{ChainID: 666 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <nil> Constantinople: <nil> Engine: unknown}"
INFO [11-01|14:47:19.257] Disk storage enabled for ethash caches dir=/Users/mig/Desktop/大三上/Blockchain/ethData/data/geth/ethash count=3
INFO [11-01|14:47:19.257] Disk storage enabled for ethash DAGs dir=/Users/mig/.ethash count=2
INFO [11-01|14:47:19.258] Initialising Ethereum protocol versions="[63 62]" network=1
INFO [11-01|14:47:19.258] Loaded most recent local header number=13 hash=0b7b9e...cae0d9 td=1708161 age=12s
INFO [11-01|14:47:19.258] Loaded most recent local full block number=13 hash=0b7b9e...cae0d9 td=1708161 age=12s
INFO [11-01|14:47:19.258] Loaded most recent local fast block number=13 hash=0b7b9e...cae0d9 td=1708161 age=12s
INFO [11-01|14:47:19.259] Loaded local transaction journal transactions=0 dropped=0
INFO [11-01|14:47:19.259] Regenerated local transaction journal transactions=0 accounts=0
WARN [11-01|14:47:19.259] Blockchain not empty, fast sync disabled
INFO [11-01|14:47:19.270] New local node record seq=2 id=f52fac4bd41364cb ip=127.0.0.1 udp=0 tcp=1234
INFO [11-01|14:47:19.270] Started P2P networking self="enode://14a439d55f2eae413e4fac32b6eff4e59d8d2d7ce8dc4368fc57e53b7bc07571baf401dbb255efe48ea22fa6034a2d8879ea74e24554a9fad587534d6c5614@127.0.0.1:1234?discport=0"
INFO [11-01|14:47:19.271] IPC endpoint opened url=/Users/mig/Desktop/大三上/Blockchain/ethData/data/geth.ipc
INFO [11-01|14:47:19.413] Etherbase automatically configured address=0x02Ef77A071231BB48d8143d988d4CE74986F8b8d
INFO [11-01|14:47:19.481] Mapped network port proto=tcp extport=1234 intport=1234 interface=NAT-PMP(192.168.199.1)
INFO [11-01|14:54:07.088] Setting new local account address=0x02Ef77A071231BB48d8143d988d4CE74986F8b8d
```

主要内容为对读取区块链配置，初始化以太坊协议，分配缓存与数据文件空间，读取最近的区块数据，启动 P2P 网络。

b. 创建一笔 transaction 时的日志如下：

```
INFO [11-01|14:54:07.088] Submitted transaction
fullhash=0xb773e690b89caa271735e720edd74bdcc19f8842b3e0c95daf5987644ac7dd9e
recipient=0x913ed1F0a0B2c2499886791f974f251cEA58e66A
```

包含了 transaction 的哈希和接收方地址，此时 transaction 将为 pending 状态，等待写入新区块。

c. 挖矿过程中日志如下：

```
INFO [11-01|17:23:58.144] Updated mining threads threads=4
INFO [11-01|17:23:58.147] Transaction pool price threshold updated price=1000000000
INFO [11-01|17:23:58.158] Commit new mining work number=24 sealhash=95f91d_4f0ce2 uncles=0 txs=0 gas=0 fees=0 elapsed=8.124ms
INFO [11-01|17:23:58.174] Commit new mining work number=24 sealhash=d928ff_619bed uncles=0 txs=1 gas=21000 fees=2.1e-05 elapsed=23.691ms
INFO [11-01|17:24:13.512] Successfully sealed new block number=24 sealhash=d928ff_619bed hash=30ddb9_53f205 elapsed=15.337s
INFO [11-01|17:24:13.514] block reached canonical chain number=17 hash=80b27e_c50989
INFO [11-01|17:24:13.514] mined potential block number=24 hash=30ddb9_53f205
```

其中updated mining threads 将挖矿线程设为4 表示使用4个进程开始挖矿；commit new mining word 表示 miner 已启动但是尚未完成区块；Successfully sealed new block 表示已完成一个区块的 PoW 求解。mined potential block 表示该区块已挖出；block reached canonical chain表示区块已加入该私有链的主链。

#### 4. 编写简单的智能合约，在 remix 下进行调试，并部署在链上进行调用

使用了如下的简单的智能合约，支持的功能为获取合约 balance，将合约 balance 取回 sender，以及将合约的一定量的 balance 发送至指定地址：

```
pragma solidity ^0.4.18;

contract buy {

    event fallbackTriggered(bytes data);
    function() public payable{emit fallbackTriggered(msg.data);}

    function buyFrom(address seller, uint amount) public {
        seller.transfer(amount);
    }



    function withdraw() public returns (uint) {
        uint left = address(this).balance;
        msg.sender.transfer(left);
        return left;
    }

    function getCurrBalance() public view returns (uint) {
        return address(this).balance;
    }
}
```

部署前的 account balance:

```
> eth.getBalance(eth.accounts[0])
500499979000000000000000
```

从 Remix 编译获取以下内容复制到 geth 中进行链上部署：

```
WEB3DEPLOY  

var buyContract = web3.eth.contract([{"constant":false,"inputs":[],"name":"withdraw","outp
var buy = buyContract.new(
    {
        from: web3.eth.accounts[0],
        data: '0x608060405234801561001057600080fd5b506101c9806100206000396000f30060806040526
        gas: '4700000'
    }, function (e, contract){
        console.log(e, contract);
        if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address + ' transactionHash:
        }
    })
})
```



部署完后可查到 account balance 已被扣除对应的 gas :

```
> eth.getBalance(eth.accounts[0])  
500499804919000000000
```

向合约转入 ether 前后调用 checkCurrBalance 查看当前合约的 balance:

```
> buy.getCurrBalance()  
0  
> eth.sendTransaction({from: eth.accounts[0], to: buy.address, value: web3.toWei(20, "ether")})  
"0xf19b4d9851706f548c389af6915a986efe04fb74fa976b86d4055feb8d35345c"  
> buy.getCurrBalance()  
20000000000000000000  
>
```

调用buyFrom 向另一新创建的 account转入 balance:

```
> buy.getCurrBalance()  
20000000000000000000  
> personal.newAccount()  
Passphrase:  
Repeat passphrase:  
"0x0881883f7bee9a99e008e2de74de54f6a0d93baf"  
> buy.buyFrom(eth.accounts[1], web3.toWei(10, "ether"))  
"0x40a1a69d3b6ab94df6d1628c0fa64baf9d3f826d643f45388ce22569d6d8eae4"  
> eth.getBalance(eth.accounts[1])  
10000000000000000000  
> buy.getCurrBalance()  
10000000000000000000  
>
```

调用 withdraw 向原账户转回 balance:

```
> buy.getCurrBalance()  
10000000000000000000  
> eth.getBalance(eth.accounts[0])  
48049972660000000000  
> buy.withdraw()  
"0x66c3a62add047de2148a945eb14017e8824795eea6beabe9f24fc151e9769c32"  
> buy.getCurrBalance()  
0  
> eth.getBalance(eth.accounts[0])  
49049969724000000000  
>
```



## 5.对交易的字段进行解释

交易的各字段如下：

```
> eth.sendTransaction({from: eth.accounts[0], to: eth.accounts[1], value: web3.toWei(12, "ether")})
"0x65856f73afb074ba4619860393fbcf3d4b502010d345022177e3bba12795bed6"
> eth.getTransaction("0x65856f73afb074ba4619860393fbcf3d4b502010d345022177e3bba12795bed6")
{
  blockHash: "0x72b04799090e8d7397557c777a9018730fa0084a4fcb735a8183fce434f8f890",
  blockNumber: 2387,
  from: "0x12e81f3eb4585352a79a47e7b3806e31f97e1376",
  gas: 90000,
  gasPrice: 1000000000,
  hash: "0x65856f73afb074ba4619860393fbcf3d4b502010d345022177e3bba12795bed6",
  input: "0x",
  nonce: 26,
  r: "0xb4b7744705b073692b0d5ce6914e80d68a21e94add42ec3d3df3c3ed65eeefce",
  s: "0x25640c991e43fe69af6390ec226838422389f9b31709536ce5f1554bac30ea81",
  to: "0x0881883f7bee9a99e008e2de74de54f6a0d93baf",
  transactionIndex: 0,
  v: "0x558",
  value: 12000000000000000000
}
```

各字段解释如下：

- blockHash: 记录该条交易的区块哈希
- blockNumber: 记录该条交易的区块编号
- gas: 交易所需的 gas 数量
- gasPrice: gas 价格,  $\text{gas} * \text{gasPrice}$  为所支付的矿工费
- hash: 交易的哈希值
- input: 发往接收地址的消息的字节码
- nonce: 发送账号的发送交易序号
- r, s, v: 三者用于 ECDSA 签名
- from, to: 交易发送和接收方地址

- transactionIndex: 区块中该交易的索引值
- value: 交易数额, 单位为 wei。