- xiabee

以太坊模拟挖矿实验

一、实验目的

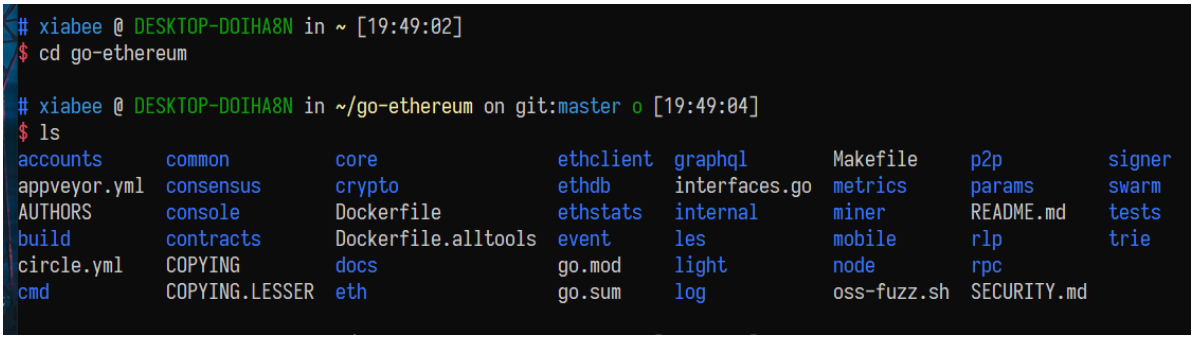
- 了解比特币、以太坊运行机制
- 模拟以太坊挖矿

二、实验过程

0x00 安装

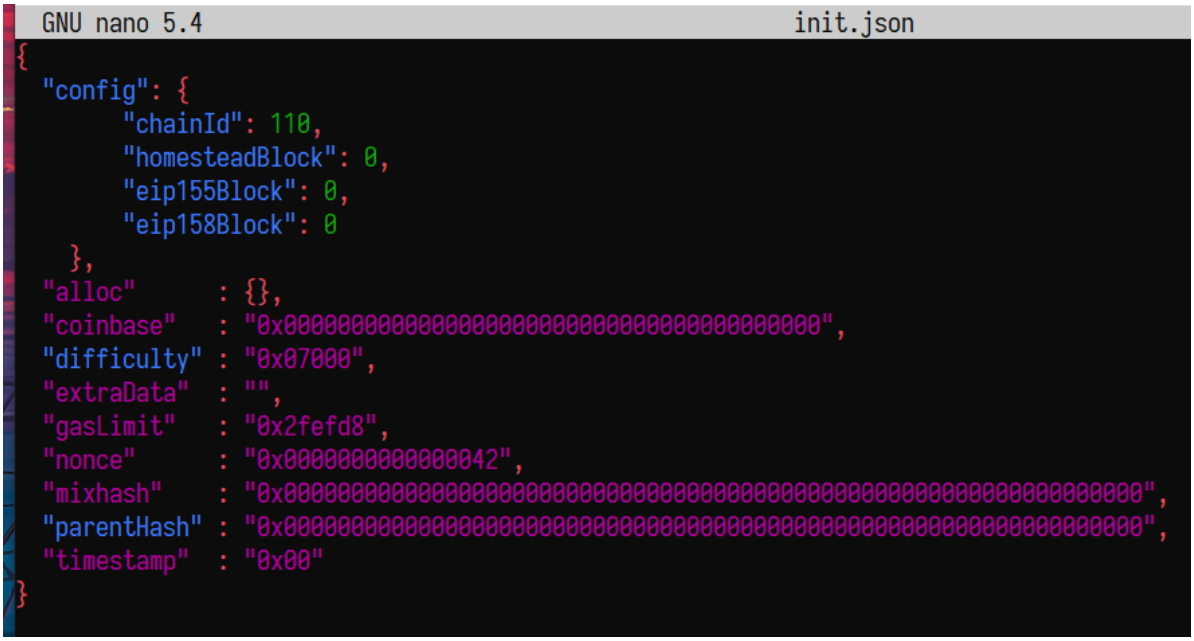
本次实验选择在`Kali-Linux`平台下安装`geth`

```
sudo apt-get install golang
# 安装go语言编译环境

go env -w GOPROXY=https://goproxy.cn
# 更换go语言代理

git clone https://github.com/ethereum/go-ethereum
cd go-ethereum
make all
# 下载安装以太坊

export PATH=$PATH:/home/$USER/go-ethereum/build/bin
# 设置环境变量
```



0x01 初始化

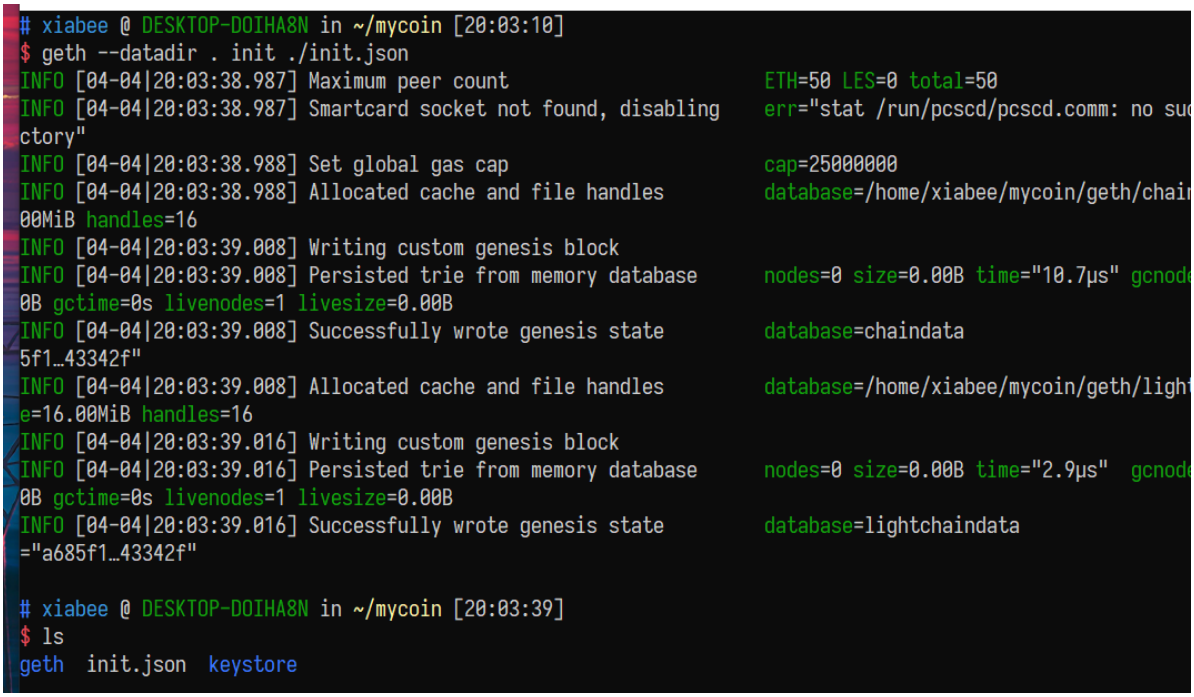
```
mkdir mycoin
touch init.json
nano init.json
# 创建初始化json
```



`init.json`:

```
{
  "config": {
        "chainId":666,
        "homesteadBlock": 0,
        "eip150Block": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
  "alloc"      : {},
  "difficulty" : "0x400",
  "extraData"  : "",
  "gasLimit"   : "0x7A1200",
  "parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00"
}
```

```
geth --datadir . init ./init.json
```



0x02 进入后台

查看指定区块与账户：

```
cd
geth --networkid "30" --nodiscover --datadir="mycoin" console 2>>
"mycoin"/"err.log"
eth.getBlock(0)
eth.accounts
```

```
}
>  eth.getBlock(0)
{
  difficulty: 1024,
  extraData: "0x",
  gasLimit: 8000000,
  gasUsed: 0,
  hash: "0xa685f1d0e28a81ae2ac45201342481dd4be1d13407fc63d571af86fe3843342f",
  logsBloom: "0x000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000",
  miner: "0x0000000000000000000000000000000000000000",
  mixHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  nonce: "0x0000000000000000",
  number: 0,
  parentHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 506,
  stateRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  timestamp: 0,
  totalDifficulty: 1024,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
> eth.accounts
[]
```

此时账户为空

创建账户：

```
personal.newAccount("pay")
personal.newAccount("collect")
# 创建两个账户
```

```
> personal.newAccount("pay")
"0x37cb08a71a5bcae399183b571fe51e1e050c4630"
> personal.newAccount("collect")
"0xf7385d0f4572f46c41367a68b3634eb53b47aa68"
>
```

此时查看账户：可以看到两个账户的公钥地址

```
> eth.accounts
["0xdeada991bda83c68e2e4428d75bc86d59e3a4efb", "0x9623aaf34567f3edb7ae0530bef9147f3de7b106"]
>
```

0x03 模拟挖矿

```
eth.coinbase
# 先设置挖矿地址，默认是第一个

eth.getBalance(eth.accounts[0])
# 查看账户余额
```

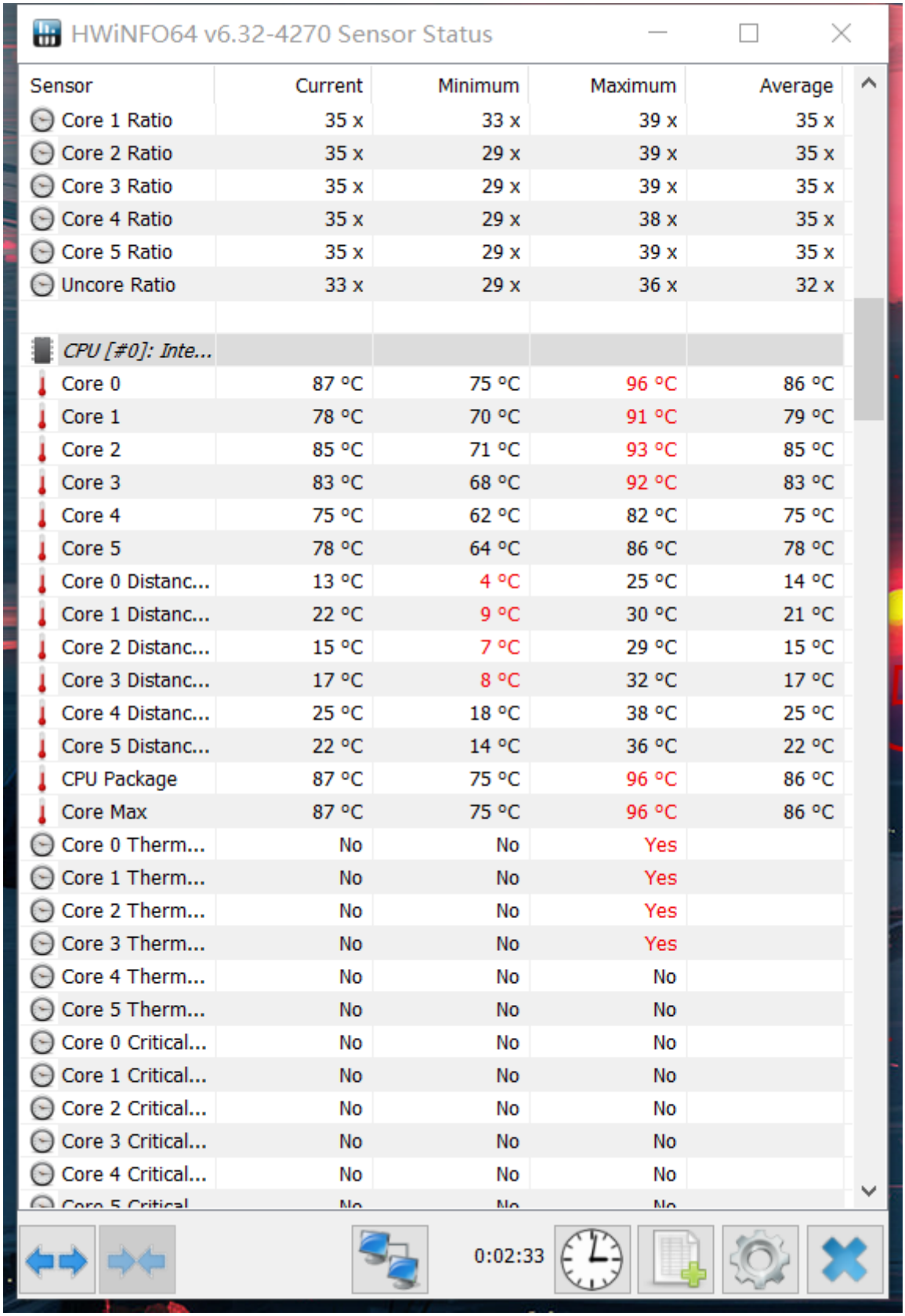
```
> eth.coinbase
"0xdeada991bda83c68e2e4428d75bc86d59e3a4efb"
> eth.getBalance(eth.accounts[0])
0
>
```

```
miner.start()
# 开始挖矿

miner.stop()
# 结束挖矿

miner.getHashrate()
# 查看挖矿状态
```

emmm可能是系统版本的原因，我的`miner.getHashrate()`一直失败：

```
> miner.getHashrate()
Error: the method miner_getHashrate does not exist/is not available
        at web3.js:6347:37(47)
        at web3.js:5081:62(37)
        at <eval>:1:18(3)

> miner.getHashrate()
Error: the method miner_getHashrate does not exist/is not available
        at web3.js:6347:37(47)
        at web3.js:5081:62(37)
        at <eval>:1:18(3)

> miner.getHashrate()
Error: the method miner_getHashrate does not exist/is not available
        at web3.js:6347:37(47)
        at web3.js:5081:62(37)
        at <eval>:1:18(3)
```

挖矿时默认使用`CPU`，没有限制功率，笔记本的`CPU`有四个核直接过热降频：


HWiNFO64 v6.32-4270 Sensor Status
Sensor Current Minimum Maximum Average
Core 1 Ratio 35 x 33 x 39 x 35 x
Core 2 Ratio 35 x 29 x 39 x 35 x
Core 3 Ratio 35 x 29 x 39 x 35 x
Core 4 Ratio 35 x 29 x 38 x 35 x
Core 5 Ratio 35 x 29 x 39 x 35 x
Uncore Ratio 33 x 29 x 36 x 32 x
CPU [#0]: Inte...
Core 0 87 °C 75 °C 96 °C 86 °C
Core 1 78 °C 70 °C 91 °C 79 °C
Core 2 85 °C 71 °C 93 °C 85 °C
Core 3 83 °C 68 °C 92 °C 83 °C
Core 4 75 °C 62 °C 82 °C 75 °C
Core 5 78 °C 64 °C 86 °C 78 °C
Core 0 Distanc... 13 °C 4 °C 25 °C 14 °C
Core 1 Distanc... 22 °C 9 °C 30 °C 21 °C
Core 2 Distanc... 15 °C 7 °C 29 °C 15 °C
Core 3 Distanc... 17 °C 8 °C 32 °C 17 °C
Core 4 Distanc... 25 °C 18 °C 38 °C 25 °C
Core 5 Distanc... 22 °C 14 °C 36 °C 22 °C
CPU Package 87 °C 75 °C 96 °C 86 °C
Core Max 87 °C 75 °C 96 °C 86 °C
Core 0 Therm... No No Yes
Core 1 Therm... No No Yes
Core 2 Therm... No No Yes
Core 3 Therm... No No Yes
Core 4 Therm... No No No
Core 5 Therm... No No No
Core 0 Critical... No No No
Core 1 Critical... No No No
Core 2 Critical... No No No
Core 3 Critical... No No No
Core 4 Critical... No No No
0:02:33

此时的账户余额：

```
> miner.stop()
null
> eth.getBalance(eth.accounts[0])
4.725e+21
> eth.getBalance(eth.accounts[1])
0
```

0x04 转账

先查看一下账户：公钥地址分别为 `0xdeada991bda83c68e2e4428d75bc86d59e3a4efb` 和 `0x9623aaf34567f3edb7ae0530bef9147f3de7b106`，第一个账户余额非零，第二个账户余额为零

```
> eth.accounts
["0xdeada991bda83c68e2e4428d75bc86d59e3a4efb", "0x9623aaf34567f3edb7ae0530bef9147f3de7b106"]
>
```

转账：

```
user1 = "0xdeada991bda83c68e2e4428d75bc86d59e3a4efb"
user2 = "0x9623aaf34567f3edb7ae0530bef9147f3de7b106"
# 设置汇款地址和收款地址

amount = web3.toWei(1, "ether")
# 设置汇款金额，1个以太币

eth.sendTransaction({from: user1, to: user2, value:amount})
```

```
> eth.accounts
["0xdeada991bda83c68e2e4428d75bc86d59e3a4efb", "0x9623aaf34567f3edb7ae0530bef9147f3de7b106"]
> user1 = "0xdeada991bda83c68e2e4428d75bc86d59e3a4efb"
ser2 = ""0xdeada991bda83c68e2e4428d75bc86d59e3a4efb"
> user2 = "0x9623aaf34567f3edb7ae0530bef9147f3de7b106"
"0x9623aaf34567f3edb7ae0530bef9147f3de7b106"
> amount = web3.toWei(1, "ether")
"1000000000000000000"
> eth.sendTransaction({from: user1, to: user2, value:amount})
Error: authentication needed: password or unlock
        at web3.js:6347:37(47)
        at web3.js:5081:62(37)
        at <eval>:1:20(10)
```

此时提示账户未解锁，需要解锁才能转账

解锁账户：

```
personal.unlockAccount(user1)
# 会提示输入phrase，即之前设置的"pay"

eth.sendTransaction({from: user1, to: user2, value:amount})
# 再次转账，打印交易哈希
```

```
> personal.unlockAccount(user1)
Unlock account 0xdeada991bda83c68e2e4428d75bc86d59e3a4efb
Passphrase:
true
> eth.sendTransaction({from: user1, to: user2, value:amount})
"0xcf8af6fc9b32dd4f8be8c6c88078be868978911f261c788d068d5619b2efda1d"
>
```

多次转账后，收款账户余额并没有增加——这是因为没有矿工来挖矿处理。每次交易的确认都需要挖矿，也就是被其他矿工共识确认，然后才能加入区块链的账本中。

```
> eth.sendTransaction({from: user1, to: user2, value:amount})
"0x2431e938e0c44b64106878af63931ecc58f8ed9d3daee93bb88f73c0beb04ad1"
> eth.getBalance(user1)
4.725e+21
> eth.getBalance(user2)
0
>
```

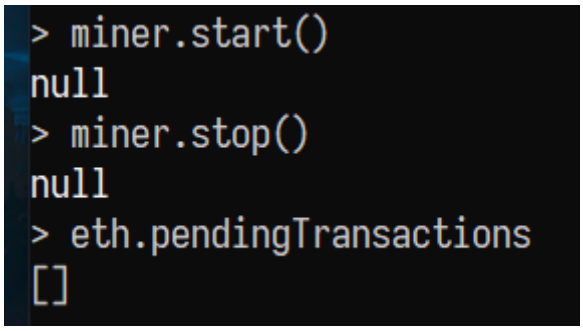
查看交易：`eth.pendingTransactions`

```
> eth.pendingTransactions
[{
    blockHash: null,
    blockNumber: null,
    from: "0xdeada991bda83c68e2e4428d75bc86d59e3a4efb",
    gas: 21000,
    gasPrice: 1000000000,
    hash: "0xcf8af6fc9b32dd4f8be8c6c88078be868978911f261c788d068d5619b2efda1d",
    input: "0x",
    nonce: 0,
    r: "0x82089555a76b0988c7b3445ba3526533d3a570a0352d94a905a494ef57777337",
    s: "0x70cddc97d2f5f0a3f54eff50bf86d384b770cde25572ee3c372f3b56e14d823a",
    to: "0x9623aaf34567f3edb7ae0530bef9147f3de7b106",
    transactionIndex: null,
    type: "0x0",
    v: "0x557",
    value: 1000000000000000000
}, {
    blockHash: null,
    blockNumber: null,
    from: "0xdeada991bda83c68e2e4428d75bc86d59e3a4efb",
    gas: 21000,
    gasPrice: 1000000000,
    hash: "0x2822d5fd0d0ccaf307117df3aac0e46ef47eed95ff8e0027379bc43157a6c7cd",
    input: "0x",
    nonce: 1,
    r: "0xfb1454370380badea1c99c45f84f20236828d869c16c8ed85282180ef54c7e0f",
    s: "0x427a6956f856ee1ec29d44dfb367e03a1e3264800fe54d4a82956e12ac995f9a",
>
```

`user1`继续挖矿，完成交易：

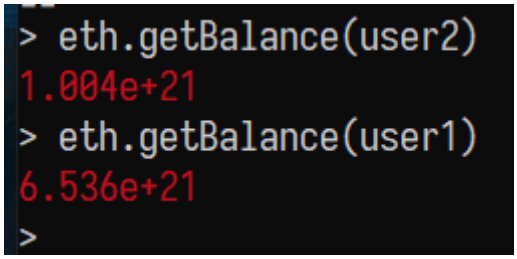
```
miner.start()
miner.stop()
# 挖矿开始与停止

eth.pendingTransactions
# 查看交易状态
```

此时交易以全部完成：



再次查看账户余额：收款账户`user2`收到汇款，转账成功

```
eth.getBalance(user2)
eth.getBalance(user1)
```



三、实验心得

- 本次实验模拟了以太坊创建账户、挖矿、转账的全过程，让我们对区块链有了新的认识
- 交易需要矿工确认，符合区块链去中心化的特性，此次模拟加深了交易确认的理解
- CPU挖矿真的很耗电……笔记本挖矿直接热到降频……
- 不要在宿舍里挖矿，比较费室友x