合约部署情况

16340221 王睿泽

总体情况

在这段时间的部署中，主要加深了对 geth 的操作，对 solidity 语言的了解，和合约部署的情况，实现功能不多。到目前为止实现了交易平台的用户注册，用户修改密码，用户对商品增加和删除，价格修改等。但是已经完成的这部分合约还没有考虑健壮性的问题，将在后面进行补充。

在这段时间进行合约的编写和部署中遇到了一些问题导致进度的缓慢。例如对 gas 的设置过小应用等导致合约并没有成功部署到链上，又例如对 solidity 的 mapping 和数组的特性不了解导致物品的删除函数实现的稍微困难。在这里主要的关联关系都用来 mapping 来实现，而这里的 delete 相当于初始化，所以会出现一些问题（会在下面的测试中提到）

已完成部分的测试

合约的部署

```
contract = eth.contract(abi)
initializer = {from: web3.eth.accounts[0], data: bytecode, gas: 300000}
token = contract.new(initializer)（必要时解锁账户 personal.unlockAccount）
mycontract = contract.at(token.address)
```

最后一步命令输入后，结果有 address 出现地址时就说明部署成功可以调用了

```
  address: "0x01b5fb4767447eb8cdf79f502e83d290b6a8fe9a",
  transactionHash: null,
  addGoods: function(),
  allEvents: function(),
  deleteGoods: function(),
  getGoodsCount: function(),
  getGoodsInformation: function(),
  getGoodsName: function(),
  getGoodsNameByname: function(),
  getGoodsOwner: function(),
  getGoodsPrice: function(),
  getName: function(),
  getNameByAddress: function(),
  getPassword: function(),
  getUserGoodsCount: function(),
  regist: function(),
  setGoodsCount: function(),
  setGoodsInformation: function(),
  setGoodsName: function(),
  setGoodsPrice: function(),
  setPassword: function()
}
```

用户注册和用户查询

每一次修改数据时，都要用 sendTransaction 发送请求

```
> mycontract.regist.sendTransaction('user0','0',{from:eth.accounts[0], gas:300000})
INFO [11-26|21:39:10.492] Submitted transaction                    fullhash=0xb9160e8b9516922747c6c01959586a
105e91beef6b39984714510e91a886c5a0 recipient=0x01b5fb4767447eB8cDF79F502e83d290b6a8fe9A
"0xb9160e8b9516922747c6c01959586a105e91beef6b39984714510e91a886c5a0"
```

若操作失败（例如重复注册）会有下图最下行的情况（即消耗设置的所有 gas）

```
> mycontract.regist.sendTransaction('user0','0',{from:eth.accounts[0], gas:300000})
INFO [11-26|22:12:46.002] Submitted transaction                    fullhash=0x7a1efdee8ea
d385a7b3bedd9e7df7adc31911cb83cb68b0790b8db099e8ef71b recipient=0x01b5fb4767447eB8cDF79F5
02e83d290b6a8fe9A
"0x7a1efdee8ead385a7b3bedd9e7df7adc31911cb83cb68b0790b8db099e8ef71b"
> miner.start(1)
INFO [11-26|22:12:52.638] Updated mining threads                   threads=1
INFO [11-26|22:12:52.638] Transaction pool price threshold updated price=1000000000
null
> INFO [11-26|22:12:52.639] Commit new mining work                   number=149 sealhash=
121a51…f39548 uncles=0 txs=0 gas=0       fees=0           elapsed=111.524µs
INFO [11-26|22:12:52.654] Commit new mining work                   number=149 sealhash=fc
30bd…4533df uncles=0 txs=1 gas=300000  fees=0.0003      elapsed=15.700ms
```

注册成功可以通过以太坊账户地址查询

```
> mycontract.getNameByAddress()
"user0"
```

用户密码的修改

修改前

```
> mycontract.getPassword()
"0"
```

进行修改

```
> mycontract.setPassword.sendTransaction('newpassword', {from:eth.accounts[0], gas:200000})
INFO [11-26|22:17:04.279] Submitted transaction                    fullhash=0x3533bcd7cf372e1c816e
b137c5abd9557c52dae53491eaf6ce727061db3a2e06 recipient=0x01b5fb4767447eB8cDF79F502e83d290b6a8fe9A
"0x3533bcd7cf372e1c816eb137c5abd9557c52dae53491eaf6ce727061db3a2e06"
> miner.start(1)
INFO [11-26|22:17:13.282] Updated mining threads                   threads=1
INFO [11-26|22:17:13.282] Transaction pool price threshold updated price=1000000000
null
> INFO [11-26|22:17:13.283] Commit new mining work                   number=154 sealhash=a45a40…00
a32b uncles=0 txs=0 gas=0       fees=0           elapsed=112.101µs
INFO [11-26|22:17:13.300] Commit new mining work                   number=154 sealhash=d50067…c1d2
a0 uncles=0 txs=1 gas=34728   fees=3.4728e-05  elapsed=16.914ms
```

修改后

```
> mycontract.getPassword()
"newpassword"
```

物品的增添

```
> mycontract.addGoods('goods2','2',2,2,{from:eth.accounts[0], gas:500000})
INFO [11-26|22:19:02.526] Submitted transaction                    fullhash=0xa4eb7b1c4c126afa414c
a3bb87aa7ac30d2a559307101f8e6d7c9a4e1ef2001f recipient=0x01b5fb4767447eB8cDF79F502e83d290b6a8fe9A
"0xa4eb7b1c4c126afa414ca3bb87aa7ac30d2a559307101f8e6d7c9a4e1ef2001f"
> miner.start(1)
INFO [11-26|22:19:07.144] Updated mining threads                   threads=1
INFO [11-26|22:19:07.144] Transaction pool price threshold updated price=1000000000
null
> INFO [11-26|22:19:07.145] Commit new mining work                   number=155 sealhash=c18d61…6e
f300 uncles=0 txs=0 gas=0       fees=0           elapsed=116.321µs
INFO [11-26|22:19:07.171] Commit new mining work                   number=155 sealhash=86d97f…d716
8d uncles=0 txs=1 gas=162103  fees=0.000162103 elapsed=26.318ms
```

在这里为了便于测试，实现的得到物品名的函数是通过输入其在数组中的位置实现。在写报告之前，有过 6 次的增加物品和一次删除物品的测试。所以 goods5 的位置是在 2 上。本次增加的 goods2 应该在 5 上，结果如下

```
> mycontract.getGoodsName(4)
"goods4"
> mycontract.getGoodsName(5)
"goods2"
> mycontract.getGoodsName(1)
"goods1"
> mycontract.getGoodsName(2)
"goods5"
```

物品的删除

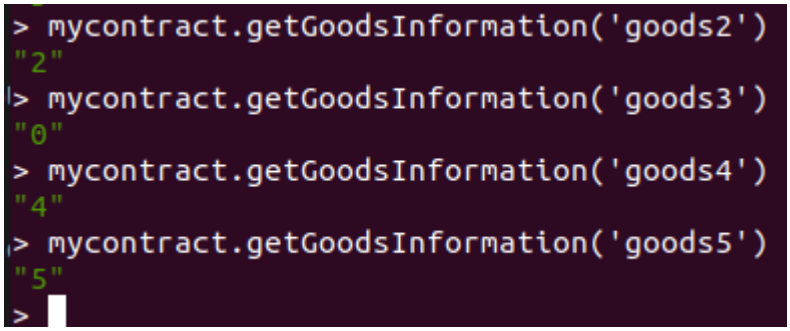
物品的删除是按照名字删除，这次测试尝试删除 goods3，在这里删除的算法是将数组最后一个与目标对换，并将变长数组长度减一。根据下图发现，若参数为 5 会报错，表明数组长度已经比 6 以小了，接着输入参数 4 发现与之前的一样，再输入参数上与上面的输入参数 5 的结果一样，删除成功

```
> mycontract.getGoodsName(5)
Error: new BigNumber() not a base 16 number:
    at L (bignumber.js:3:2876)
    at bignumber.js:3:8435
    at a (bignumber.js:3:389)
    at web3.js:1110:23
    at web3.js:1634:20
    at web3.js:826:16
    at map (<native code>)
    at web3.js:825:12
    at web3.js:4080:18

> mycontract.getGoodsName(4)
"goods4"
> mycontract.getGoodsName(3)
"goods2"
```

物品信息的查询

在这里只对 information 的查询进行测试，发现删除的 goods3 的信息竟然存在（delete mapping 后该映射对应的值为 0，所以输出的 goods0 的信息，在之后会想办法解决这个问题（我恨 solidity））。物品其他信息，例如价格，数量等与 information 结果相同，在这里并不做展示。



代码截图

```
pragma solidity ^0.4.19;

contract TradingSystem {
    struct Goods {
        string owner;
        string name;
        string information;
        uint price;
        uint count;
    }

    struct User {
        string name;
        string password;
        uint goodsCount;
        mapping(string => uint) goodsPos;
    }

    User[] users;
    uint userCount = 0;

    mapping(string => uint) usersFlag;
    mapping(string => uint) userPos;
    mapping(address => string) addressToUser;
    mapping(string => Goods[]) userToGoods;
    //mapping(string => string) userToAddress;

    function regist(string name, string password) public {
        require(usersFlag[name] == 0, "The name has been used!");
        require(usersFlag[addressToUser[msg.sender]] == 0, "The address has registed!");
        users.push(User(name, password, 0));
        userPos[name] = userCount;
        userCount++;
        usersFlag[name]++;
        addressToUser[msg.sender] = name;
    }

    function getName(uint x) public view returns (string){
        return users[x].name;
    }
```

```
41
42      function getNameByAddress() public view returns (string){
43          return addressToUser[msg.sender];
44      }
45
46      function setPassword(string newPassword) public {
47          users[userPos[addressToUser[msg.sender]]].password = newPassword;
48      }
49
50      function getPassword() public view returns (string){
51          return users[userPos[addressToUser[msg.sender]]].password;
52      }
53
54      //Goods operation
55      function addGoods(string name, string information, uint price, uint count) public {
56          users[userPos[addressToUser[msg.sender]]].goodsPos[name] = users[userPos[addressToUser[msg.sender]]].goodsCount;
57          users[userPos[addressToUser[msg.sender]]].goodsCount++;
58          userToGoods[addressToUser[msg.sender]].push(Goods(addressToUser[msg.sender], name, information, price, count));
59      }
60
61      function deleteGoods(string name) public {
62          userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]] = userToGoods[addressToUser
63          users[userPos[addressToUser[msg.sender]]].goodsPos[userToGoods[addressToUser[msg.sender]][(userToGoods[addressToUser[msg.sen
64          delete users[userPos[addressToUser[msg.sender]]].goodsPos[name];
65          userToGoods[addressToUser[msg.sender]].length = userToGoods[addressToUser[msg.sender]].length - 1;
66          users[userPos[addressToUser[msg.sender]]].goodsCount--;
67      }
68
69      function getUserGoodsCount() public view returns (uint) {
70          return users[userPos[addressToUser[msg.sender]]].goodsCount;
71      }
72
73      function setGoodsName(string name, string newName) public {
74          userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].name = newName;
75      }
76
77      function getGoodsName(uint x) public view returns (string) {
78          return userToGoods[addressToUser[msg.sender]][x].name;
79      }
80
```

```
80
81      function getGoodsNameByname(string name) public view returns (string) {
82          return userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].name;
83      }
84
85      function getGoodsOwner(uint x) public view returns (string) {
86          return userToGoods[addressToUser[msg.sender]][x].owner;
87      }
88
89      function setGoodsInformation(string name, string newInformation) public {
90          userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].information = newInformation;
91      }
92
93      function getGoodsInformation(string name) public view returns (string) {
94          return userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].information;
95      }
96
97      function setGoodsPrice(string name, uint newPrice) public {
98          userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].price = newPrice;
99      }
100
101     function getGoodsPrice(string name) public view returns (uint) {
102         return userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].price;
103     }
104
105     function setGoodsCount(string name, uint newCount) public {
106         userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].count = newCount;
107     }
108
109     function getGoodsCount(string name) public view returns (uint) {
110         return userToGoods[addressToUser[msg.sender]][users[userPos[addressToUser[msg.sender]]].goodsPos[name]].count;
111     }
112 }
```

映射

分别有如下映射

1. goodsPos：在结构体 User 中，每个 user 通过物品名（string）到在 UserToGoods 映射中得到数组（Goods）中位置（uint）的映射
2. userFlag：判断用户是否创建，从用户名（string）到一个常数（uint）
3. userPos：通过用户名（string）到在 users 数组中的位置（uint）的映射
4. addressToUser：以太坊用户地址（address）到用户名（string）的映射
5. userToGoods：用户名（string）到物品仓库（Goods[]）的映射

```
struct User {
    string name;
    string password;
    uint goodsCount;
    mapping(string => uint) goodsPos;
}

User[] users;
uint userCount = 0;

mapping(string => uint) usersFlag;
mapping(string => uint) userPos;
mapping(address => string) addressToUser;
mapping(string => Goods[]) userToGoods;
//mapping(string => string) userToAddress;
```