**补充实验 Fabric SmallBank 链码**

**【实验介绍】**

现有一个业务场景，一家银行向外提供储蓄金和存折服务，使用Fabric进行简单的存证。每个账户可以在银行里开户，每个账户所拥有对应的储蓄金和存折。用户可以对储蓄金和存折中的金额进行存钱、取钱、转账等从操作。

这个联盟找到你，让你根据上述场景完成一个链码，以满足他们的应用需求。

关于存折和储蓄金，你可以简单的使用“账户\_savings”和“账户\_checkings”来简单表示它们在fabric中的key值。

我们将继续在之前的fabric1.3上进行实验，我们在fabric/examples/chaincode/go下新建了链码smallbank，并给定了chaincode.go的框架，你只需要将欠缺的代码补全，然后运行e2e案例即可。

**【实验要求】**

需要你实现的功能点主要包括：

（1）创建一个新的账户；

（2）向储蓄账户增加一定余额；

（3）向支票账户增加一定余额

（4）在两个支票账户间转账

（5）减少一个支票账户；

（6）将储蓄账户的资金全部转到支票账户；

（7）上述功能点根据实现与否按点给分。自己的想法进行额外的功能点添加，按点加分，根据其复杂度、难度、创新等酌情加分。

在代码中已经添加了针对性的注释，完成todo即可。

**【实验准备】**

在原有的fabric环境下进行修改即可。

**【实验过程】**

本实现是在e2e案例的基础上修改完成的，代码已经上传到了

<https://github.com/ZhmYe/Fabric_1.3_Lab_10.git的lab_11>分支下。

由于实验十中大家对上述代码进行了修改，因此git pull会出现需要merge的要求，这里建议大家在完成实验十以后，将整个代码git stash了或者将原来的fabric文件夹重命名为其它名字，如果git stash的话，可以参考实验十中的相关命令给相关的.sh文件赋予权限。

git stash命令参考如下：

# 将原来的改动去掉

git stash

# 更新代码

git pull

# 切换分支

git checkout lab\_11

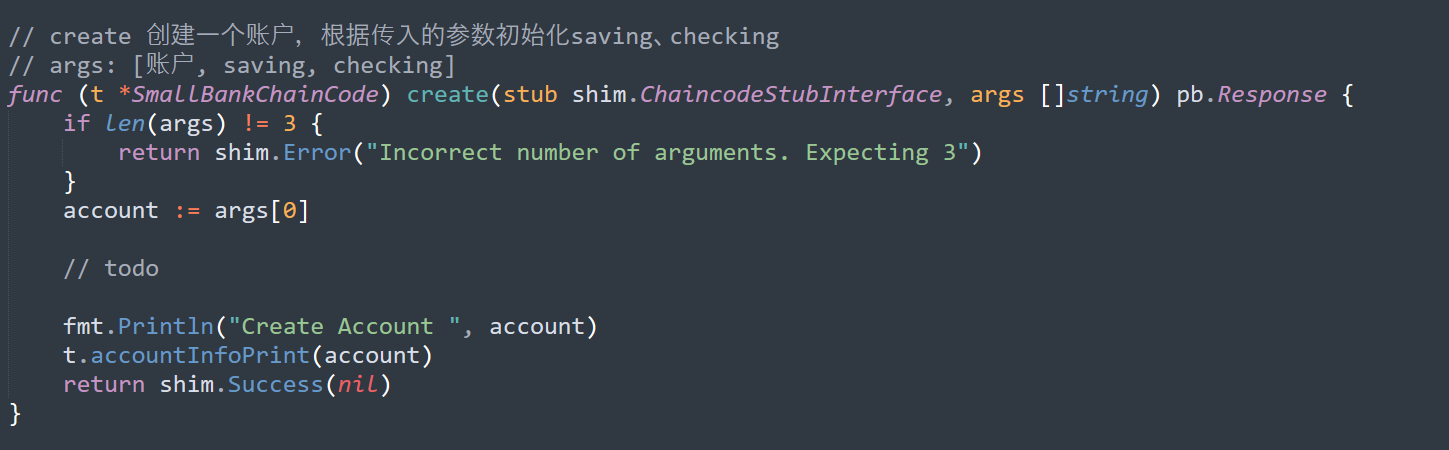
# 然后参考实验十进行chmod +x

本次实验需要完成的代码在fabric目录下examples/chaincode/go/smallbank/chaincode.go

需要完成的函数有6个，都已经在代码里加上了相关的解释，下面做一些简单的介绍

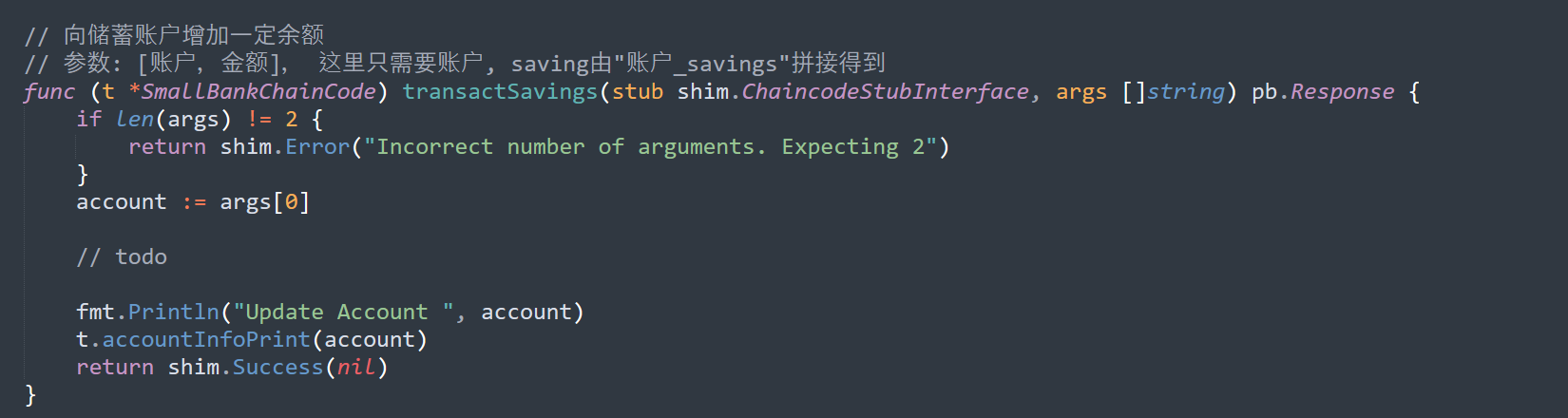
**1 待完成的函数**

1. **create 创建账户**



这个函数接受3个参数，添加一个新的账户account，并初始化它的储蓄金余额、存折余额为saving和checking。在本次实验中，一个账户的储蓄金地址用“账户\_savings”表示，存折地址用“账户\_checkings”表示。

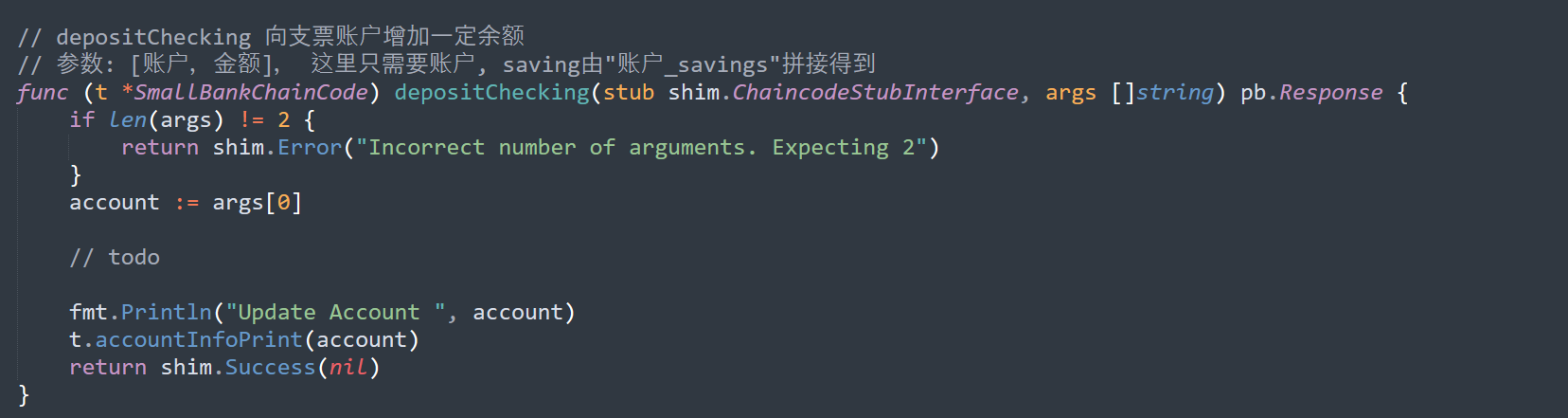
1. **transactSavings 存储蓄金**



这个函数接受2个参数，向一个账户account的储蓄金中增加一定金额amount，也就是模拟用户存储蓄金的过程。

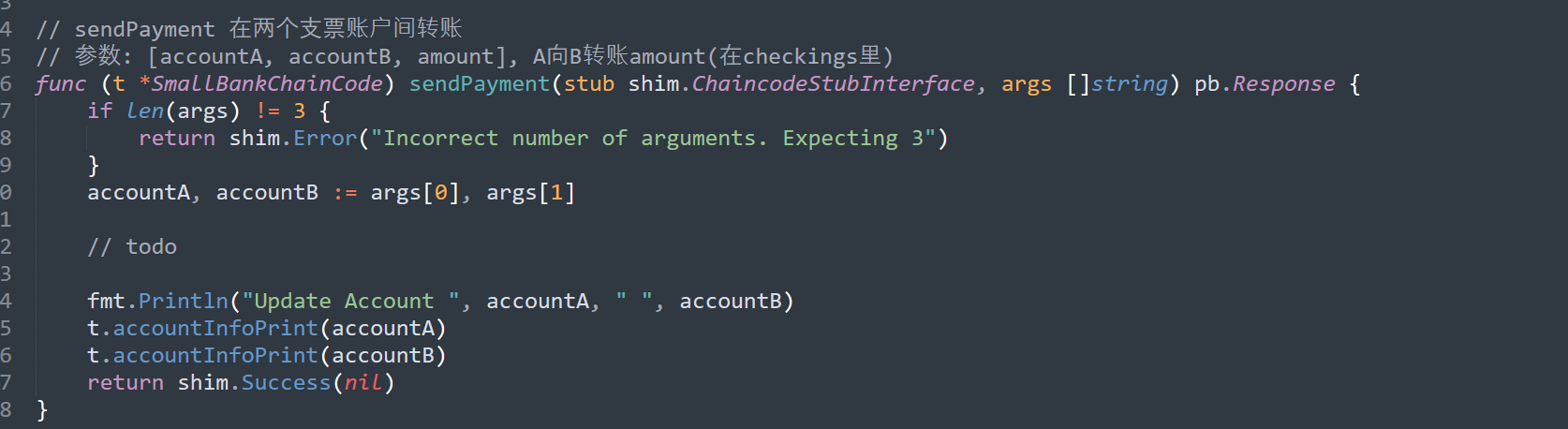
1. **depositChecking 存存折**

这个函数和(2)类似，就是把储蓄金换成了存折。



1. **sendPayment 存折转账**

这个函数接受3个参数，简单来说就是使得账户A的存折余额少一部分转给账户B，其中注意要判断余额是否不足

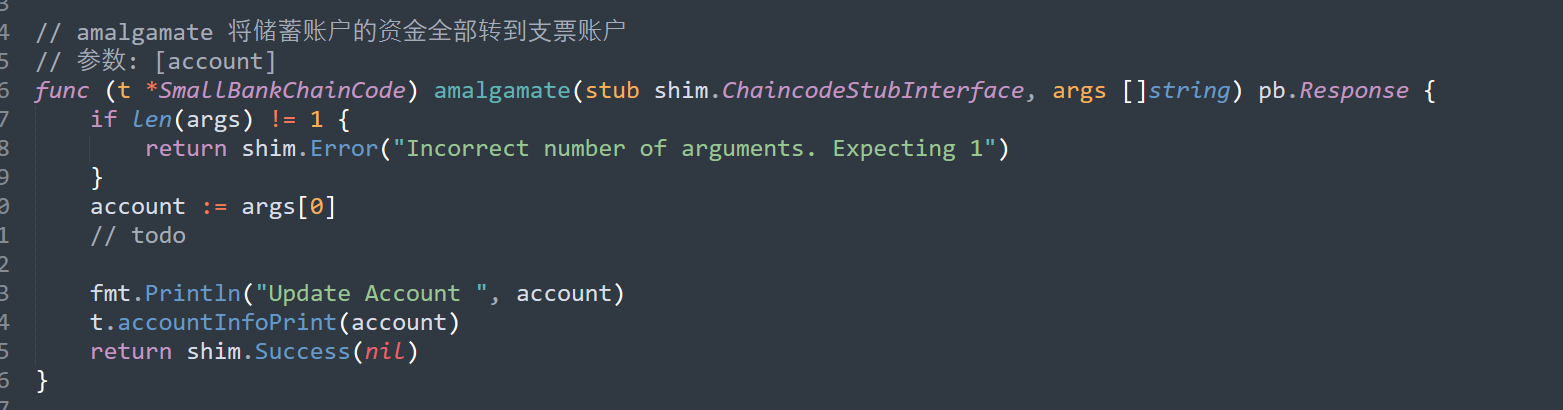


1. **writeCheck 取存折**

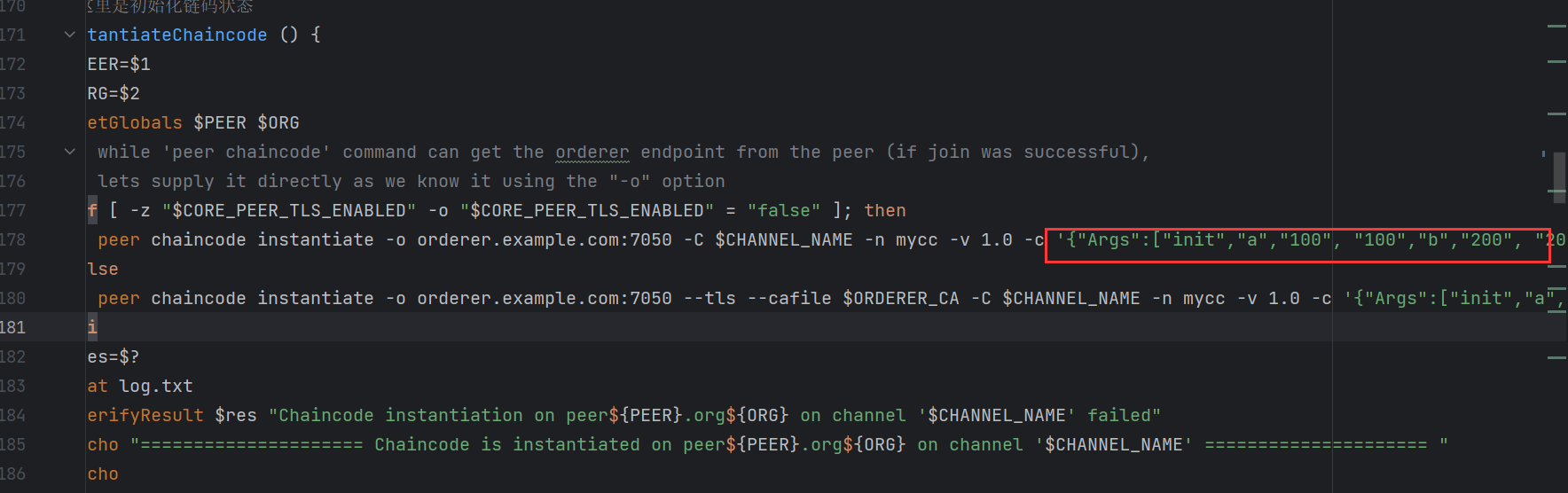
这个函数接受2个参数，也就是从存折中取出一部分钱，同样也要注意判断是否余额不足

1. **amalgamate 将储蓄金全部转出到存折**

这个函数接受一个参数，即将account的saving里的余额全部转到checking中



另外，我们在e2e\_cli/script/script.sh下初始化了这个链码，初始化了a,b两个账户，其中a的储蓄金、存折余额分别为100,100，而b则为200,200



**2 运行e2e**

在完成相关代码编写后

启动:

|  |
| --- |
| ./network\_setup.sh up |

首先确认是否能看到END-E2E的输出，保证链码已经正确安装。

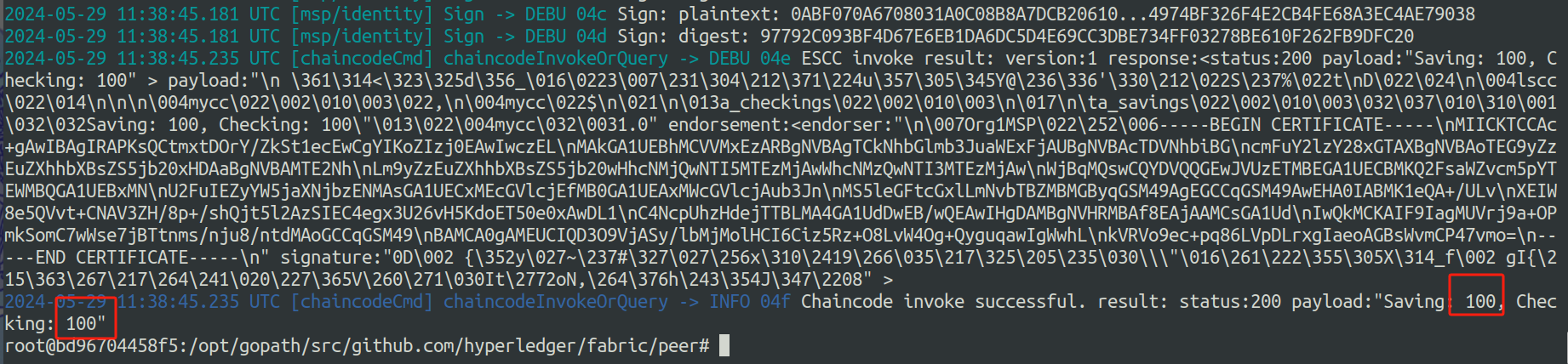
**3 测试**

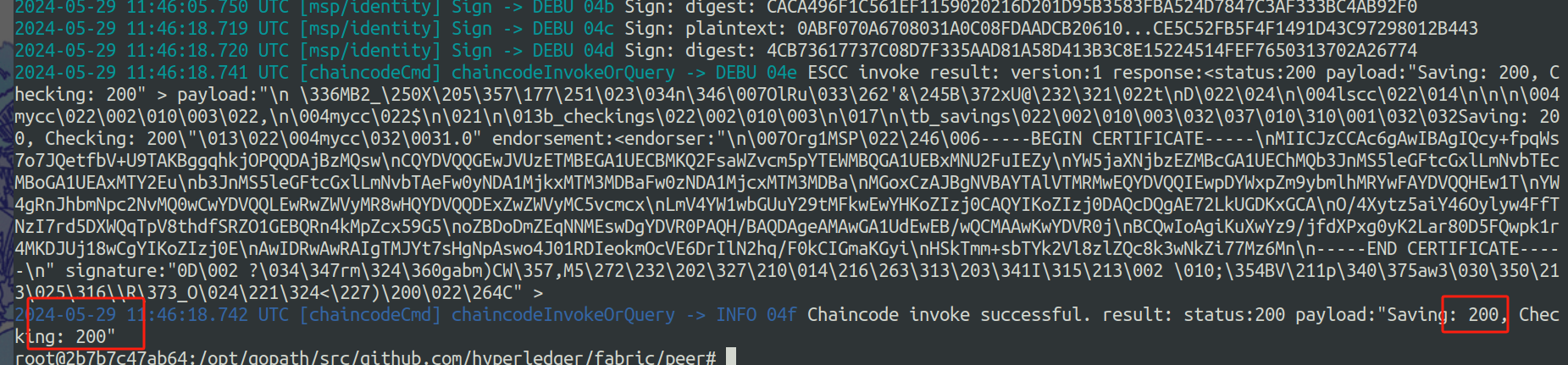
进入cli客户端：

|  |
| --- |
| docker exec -it cli bash |

在e2e\_cli/script/script.sh下，我们初始化了两个账户a,b,我们先来查看他们的余额是否正确。

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["query","a"]}'  peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["query","b"]}' |





和我们初始化的bill0账单的金额相同

然后我们新建一个账户

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["create","c","50","50"]}' |

新建账户后我们查询其余额，看是否与我们新建时传入的数据一致

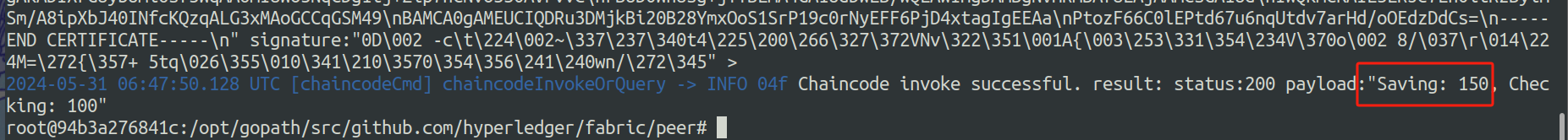
|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["query","c"]}' |

查询的命令下面不再重复给出。



然后我们向a账户的储蓄金中转入50

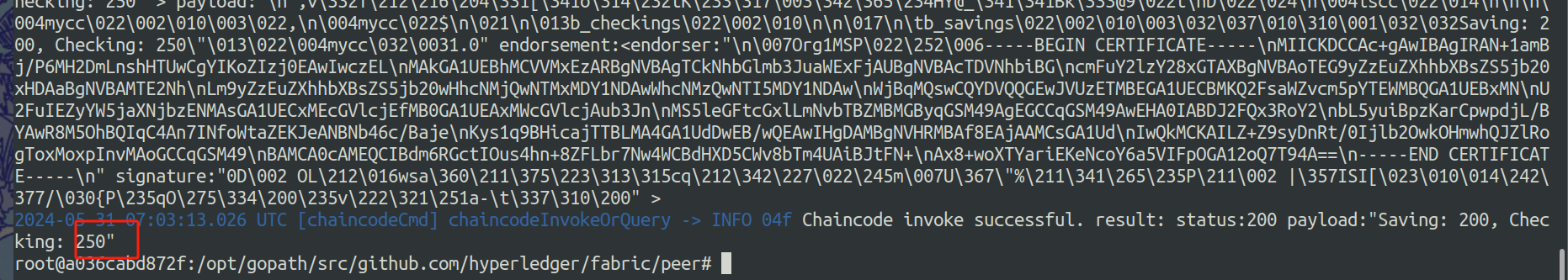
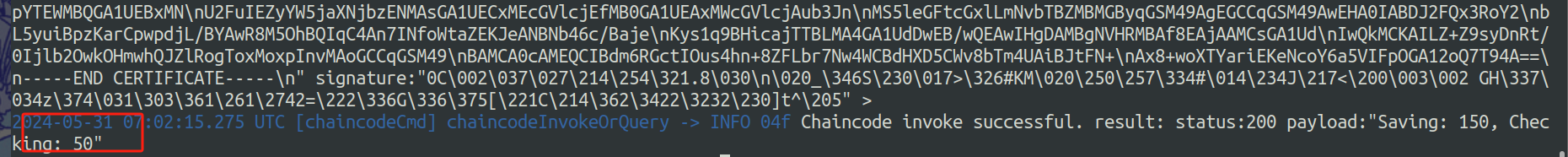
|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["transactSavings","a","50"]}' |

查询a的余额，可以看到其savings增加了50。

添加存折的部分和上面类似，不再列出。

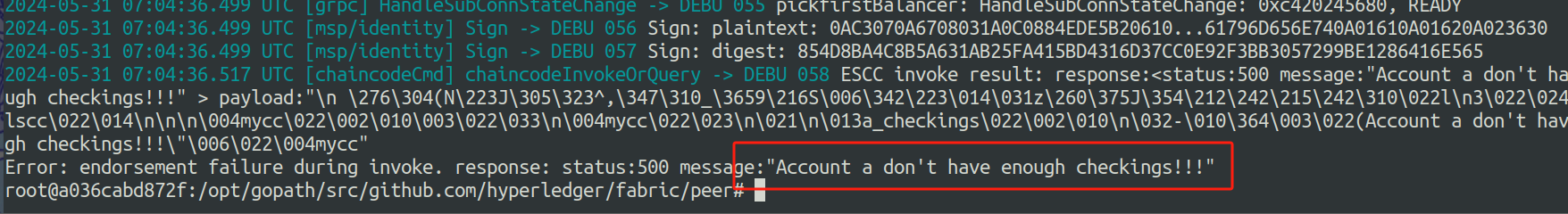
接下来我们模拟a和b之间的存折转账，设置转账金额为50，此时a原本的存折金额为100，b为200

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["sendPayment","a","b","50"]}' |

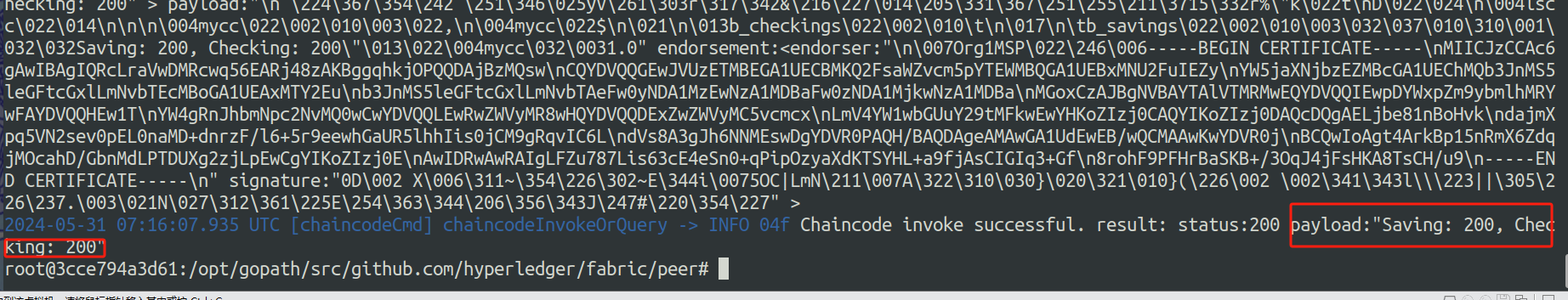
随后我们查询a和b的余额，可以看到余额已经分别变化为50,250

接下来，我们再模拟余额不足的转账，我们再让a转账给b 60，此时会出现报错。

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["sendPayment","a","b","60"]}' |

然后我们再模拟b取出其一部分存折，取出50,

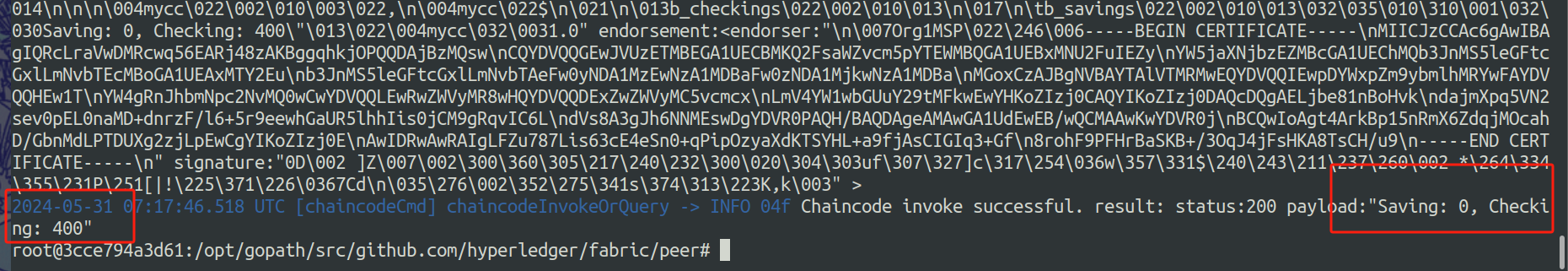
|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["writeCheck","b","50"]}' |

随后查询余额，可以看到b的存折余额为200

最后我们再模拟b将所有储蓄金全部转到存折中

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["amalgamate","b"]}' |

查询余额可以看到，b的储蓄金余额为0，而存折余额为400。

**【实验小结】**

本实验设计了一个实际的业务场景smallbank，让大家尝试对fabric的e2e业务案例的实现代码进行修改，使其符合新的需求。通过对业务逻辑链码的修改和其联盟网络配置情况的查看和部署以及前几个相关实验中对于fabric框架的介绍，帮助大家可以简单地实现链码的编写和部署，让大家对于fabric如何实现基于许可的区块链网络和相应的业务逻辑的实现过程有一个大致的了解。

**【习题】**

1. 简单地说明一下客户端申请调用链码函数进行后fabric的运行流程。

2. 你可以根据自己的想法进行额外的功能点添加，按点加分，根据其复杂度、难度、创新等酌情加分。

**【参考文献】**

1. 邹均，张海宁，唐屹，李磊等. 区块链技术指南. 机械工业出版社
2. 冯翔. 区块链开发实战: Hyperledger Fabric 关键技术与案例分析 Key technology and case analysis for Hyperledger Fabric[M]. 机械工业出版社, 2018.
3. Hyperledger Community. Hyperledger Fabric(OL). (2020-06-01) [2020-07-12]. <https://github.com/hyperledger/fabric>.
4. Hyperledger Community. Smart Contracts and Chaincode(OL). (2020-08-20) [2020-12-11]. <https://github.com/hyperledger/fabric-docs-i18n/blob/release-2.2/docs/locale/es/source/smartcontract/smartcontract.md>.