Clarity环境搭建

从Docker Hub拉取智能合约运行环境

```
$ docker pull blockstack/blockstack-core:clarity-developer-preview
clarity-developer-preview: Pulling from blockstack/blockstack-core
f28e4bea9e1e: Pull complete
Digest: sha256:8f4a7dab9a2d133722a568a2cf40ebbaceb6cbe82149bc49dcfe760d651f4a67
Status: Downloaded newer image for blockstack/blockstack-core:clarity-developer-preview
docker.io/blockstack/blockstack-core:clarity-developer-preview
```

运行Blockstack-core测试环境

```
$ docker run -it -v $HOME/blockstack-dev-data:/data/ blockstack/blockstack-core:clarity-developer-preview bash
root@d4fa284392a2:/src/blockstack-core#
```

DB Browser for SQLite安装

```
$ sudo apt-get install sqlitebrowser
```

钱包地址生成

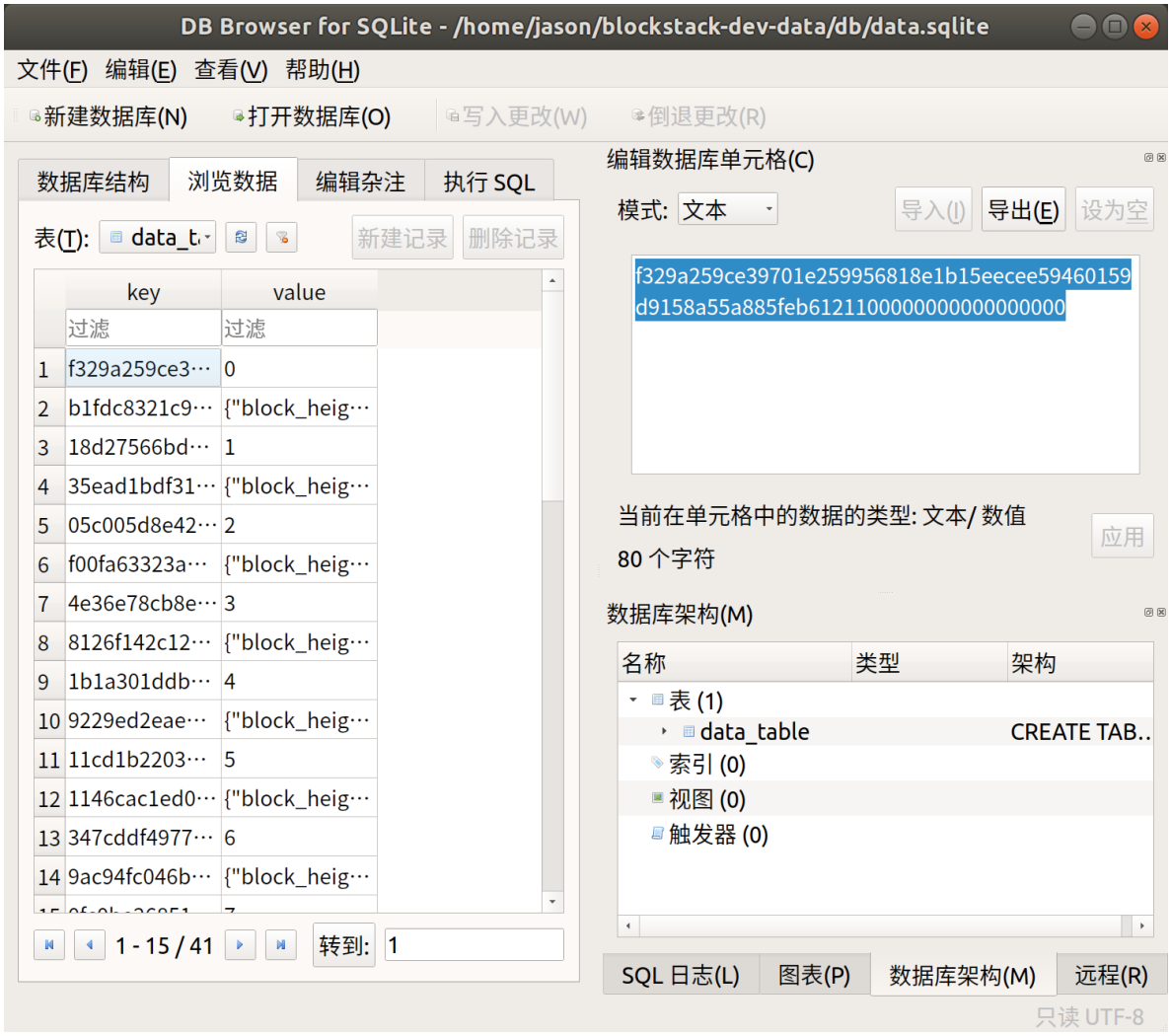
```
root@d4fa284392a2:/src/blockstack-core# clarity-cli generate_address
SP86DRPMQ98TWKWFN8WE6Q17EK7JCTN7HHF35R2E

root@d4fa284392a2:/src/blockstack-core# export DEMO_ADDRESS=SP86DRPMQ98TWKWFN8WE6Q17EK7JCTN7HHF35R2E
```

数据库初始化

```
root@d4fa284392a2:/src/blockstack-core# clarity-cli initialize /data/db
Database created.
```

查看数据库数据



Clarity智能合约开发

demo1：sum

编写合约

```
root@d4fa284392a2:/src/blockstack-core/sample-programs# cat sum.clar
(define-public (sum (a1 uint) (a2 uint))
  (ok (+ a1 a2)))
```

校验合约

```
# Usage: clarity-cli check [program-file.clar] (vm-state.db)
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli check sum.clar /data/db
Checks passed.
```

部署合约

```
# Usage: clarity-cli launch [contract-identifier] [contract-definition.clar] [vm-state.db]
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli launch $DEMO_ADDRESS.sum
sum.clar /data/db
Contract initialized!
```

执行合约

```
# Usage: clarity-cli execute [vm-state.db] [contract-identifier] [public-function-name] [sender-address] [args...]
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli execute /data/db $DEMO_ADDRESS.sum sum $DEMO_ADDRESS u10 u20
Transaction executed and committed. Returned: u30
```

demo2：store

编写合约

```
root@d4fa284392a2:/src/blockstack-core/sample-programs# cat store.clar
(define-data-var value uint u0)
(define-public (myget)
 (ok (var-get value)))
(define-public (myset (nvalue uint))
 (begin (var-set value (+ nvalue u1))
   (ok 'true)))
```

校验合约

```
# Usage: clarity-cli check [program-file.clar] (vm-state.db)
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli check store.clar /data/db
Checks passed.
```

部署合约

```
# Usage: clarity-cli launch [contract-identifier] [contract-definition.clar] [vm-state.db]
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli launch $DEMO_ADDRESS.store store.clar /data/db
Contract initialized!
```

执行合约

```
# 查询初始值为0
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli execute /data/db $DEMO_ADDRESS.store myget $DEMO_ADDRESS
Transaction executed and committed. Returned: u0

#传入参数 2
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli execute /data/db $DEMO_ADDRESS.store myset $DEMO_ADDRESS u2
Transaction executed and committed. Returned: true

# 再次查询得到加 1 后的值 3
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli execute /data/db $DEMO_ADDRESS.store myget $DEMO_ADDRESS
Transaction executed and committed. Returned: u3
```

demo3：多合约调用

编写合约

- 合约1 (contractA)

```
(define-public (sum (a uint) (b uint))
  (ok (+ a b)))
```

- 合约2(contractB)

```
(define-public (callSum)
  (contract-call? .contractA sum u1 u2)
 )
```

校验合约

```
# Usage: clarity-cli check [program-file.clar] (vm-state.db)
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli check store.clar /data/db
Checks passed.
```

部署合约

```
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli launch
$DEMO_ADDRESS.contractA contractA.clar /data/db
Contract initialized!

root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli launch
$DEMO_ADDRESS.contractB contractB.clar /data/db
Contract initialized!
```

执行合约

```
root@d4fa284392a2:/src/blockstack-core/sample-programs# clarity-cli execute /data/db
$DEMO_ADDRESS.contractB callSum $DEMO_ADDRESS
Transaction executed and committed. Returned: u3
```

token.clar代码分析

```
; 定义了账号和余额映射的map
(define-map tokens ((account principal)) ((balance uint)))

; 根据账号获取余额的私有成员方法
(define-private (get-balance (account principal))
  ; 根据账号从map中获取账户余额，如果为空，默认为 0
  (default-to u0 (get balance (map-get? tokens (tuple (account account))))))

; 为账户发行代币
(define-private (token-credit! (account principal) (amount uint))
  ; 如果账户余额≤ 0
  (if (<= amount u0)
    ; 打印错误日志
    (err "must move positive balance")
    ; 获取账户当前余额
    (let ((current-amount (get-balance account)))
      (begin
       ; 将账户当前余额＋发行余额作为账户新的余额，添加到map中
       (map-set tokens (tuple (account account))
                (tuple (balance (+ amount current-amount))))
```

```
        ;返回账户最新的余额
        (ok amount)))))

;代币转移
(define-public (token-transfer (to principal) (amount uint))
 ;获取账户当前余额
 (let ((balance (get-balance tx-sender)))
  ;如果转账金额>账户余额，或转账金额<0
  (if (or (> amount balance) (<= amount u0))
      ;打印错误日志
      (err "must transfer positive balance and possess funds")
      (begin
       ;当前账户减掉转账金额
       (map-set tokens (tuple (account tx-sender))
                (tuple (balance (- balance amount))))
       ;目的账户增加转账金额
       (token-credit! to amount)))))

;挖矿函数
(define-public (mint! (amount uint))
  ;获取当前账户余额
  (let ((balance (get-balance tx-sender)))
   ;为当前账户增加挖矿奖励
   (token-credit! tx-sender amount)))

;为账户 1 发行10000货币
(token-credit! 'SZ2J6ZY48GV1EZ5V2V5RB9MP66SW86PYKKQ9H6DPR u10000)
;为账户2发行300货币
(token-credit! 'SM2J6ZY48GV1EZ5V2V5RB9MP66SW86PYKKQVX8X0G u300)
```

思考题 1：根据今天对于智能合约的讲解，你认为智能合约可以解决哪些现有互联网无法解决的问题？又会带来哪些问题？

可以解决合约条款内容不透明、歧义条款、及履约不及时的问题。

问题：如果条款发生变化，更新不方便。