『0005』- 以太坊智能合约生命周期（Ethereum smart contracts lifecycle）

> 孔壹学院：国内区块链职业教育领先品牌
>
> 作者：黎跃春，区块链、高可用架构工程师
> 微信：liyc1215 QQ群：348924182 博客：http://liyuechun.org

上一篇中，我们讲解了『0004』- 基于Ethereum Wallet的Solidity HelloWorld智能合约(Smart Contract)。本篇文章我们将一步步带大家掌握以太坊智能合约的生命周期。

合约对象初始化

上一节中我们提到`Solidity`编写合约和面向对象编程语言非常相似，我们可以通过构造函数（constructor）来初始化合约对象。构造函数就是方法名和合约名字相同的函数，创建合约时会调用构造函数对状态变量进行数据初始化操作。

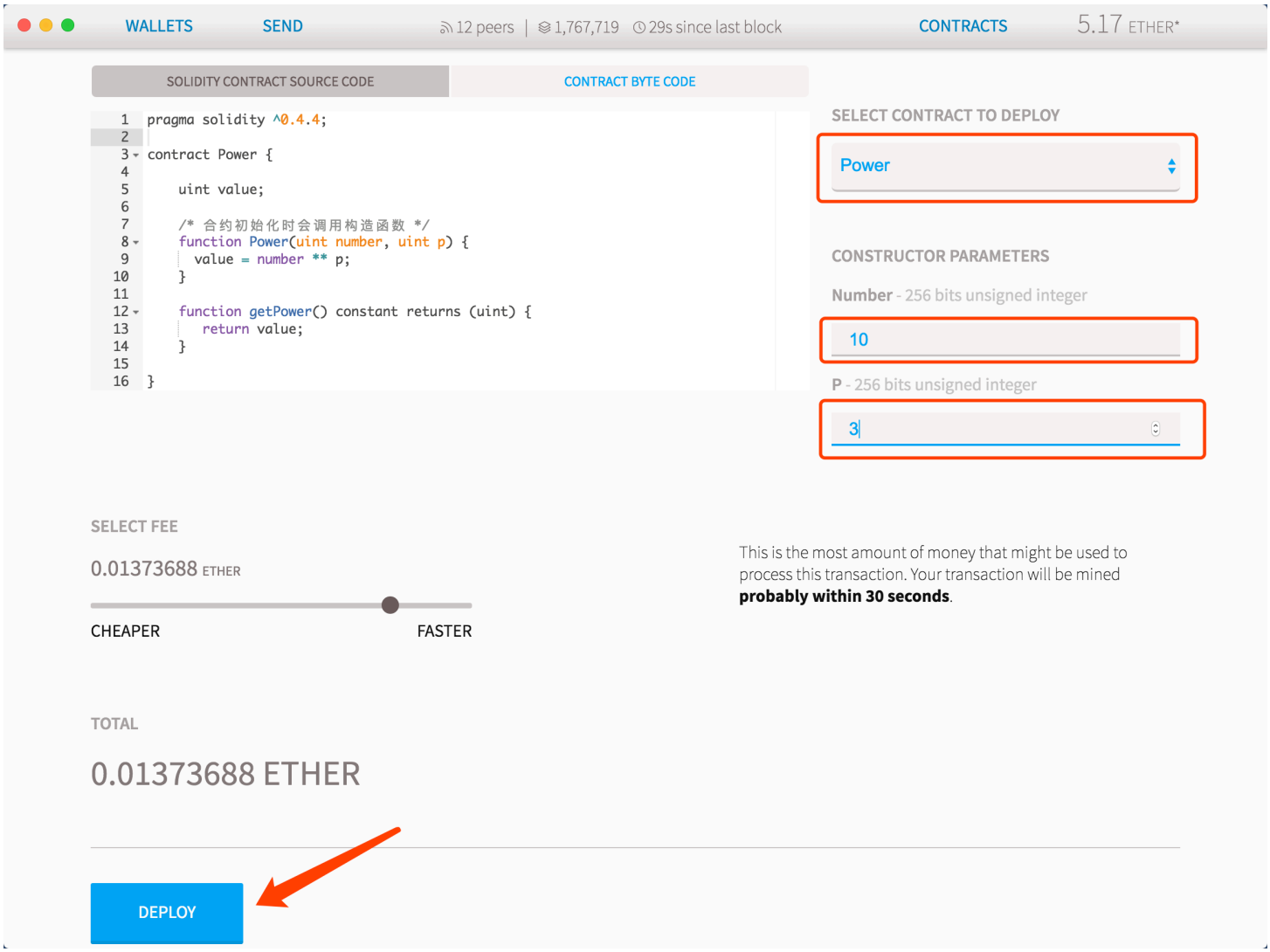
```
pragma solidity ^0.4.4;

contract Power {

    uint value;

    /* 合约初始化时会调用构造函数 */
    function Power(uint number, uint p) {
      value = number ** p;
    }

    function getPower() constant returns (uint) {
       return value;
    }

}
```

同一个合约是否可同时拥有两个构造函数?

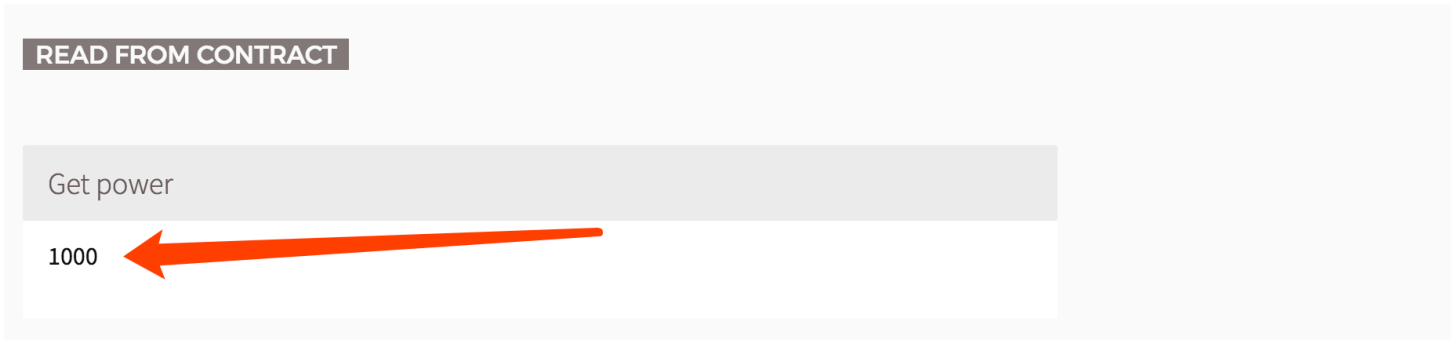
由上图不难看出，当我们同一个合约如果有多个构造函数时，会出现下面的错误提示：

```
More than one constructor defined.
     function Power(uint number) {
     ^
Spanning multiple lines.
Another declaration is here
```

如果你尝试部署`Power`合约，你将看到需要提供两个参数进行合约交易。



合约部署成功后，我们将会看到`getPower`函数返回的值。



构造函数（constructor）会在合约创建时对数据进行初始化，传统的类会在对象销毁时调用析构函数(destructor)，在以太坊智能合约中，同样在合约销毁时，自动调用析构函数销毁相关数据。

合约owner

让我们来对我们上一篇中的`Counter`合约进行改进，合约对象创建时在构造函数中保存我们当前合约的合约地址，在`increment`函数中增加一个判断，只有当在我们合约内部调用时，状态变量`count`才加`1`，否则不做任何其他操作。

```
pragma solidity ^0.4.4;

contract Counter {

    uint count = 0;
    address owner; //存储Counter合约owner

    function Counter() {
       owner = msg.sender; // 存储Counter合约owner
    }

    function increment() public {
       if (owner == msg.sender) { // 判断是谁在调用`increment`方法
          count = count + 1;
       }
    }

    /* 读取count的值 */
    function getCount() constant returns (uint) {
       return count;
    }

}
```

因此我们给合约增加了一层保护层，只有创建合约的creator才能够自己调用 `increment` 方法让 `count` 加 `1` 。

析构函数 - 一个合约的销毁

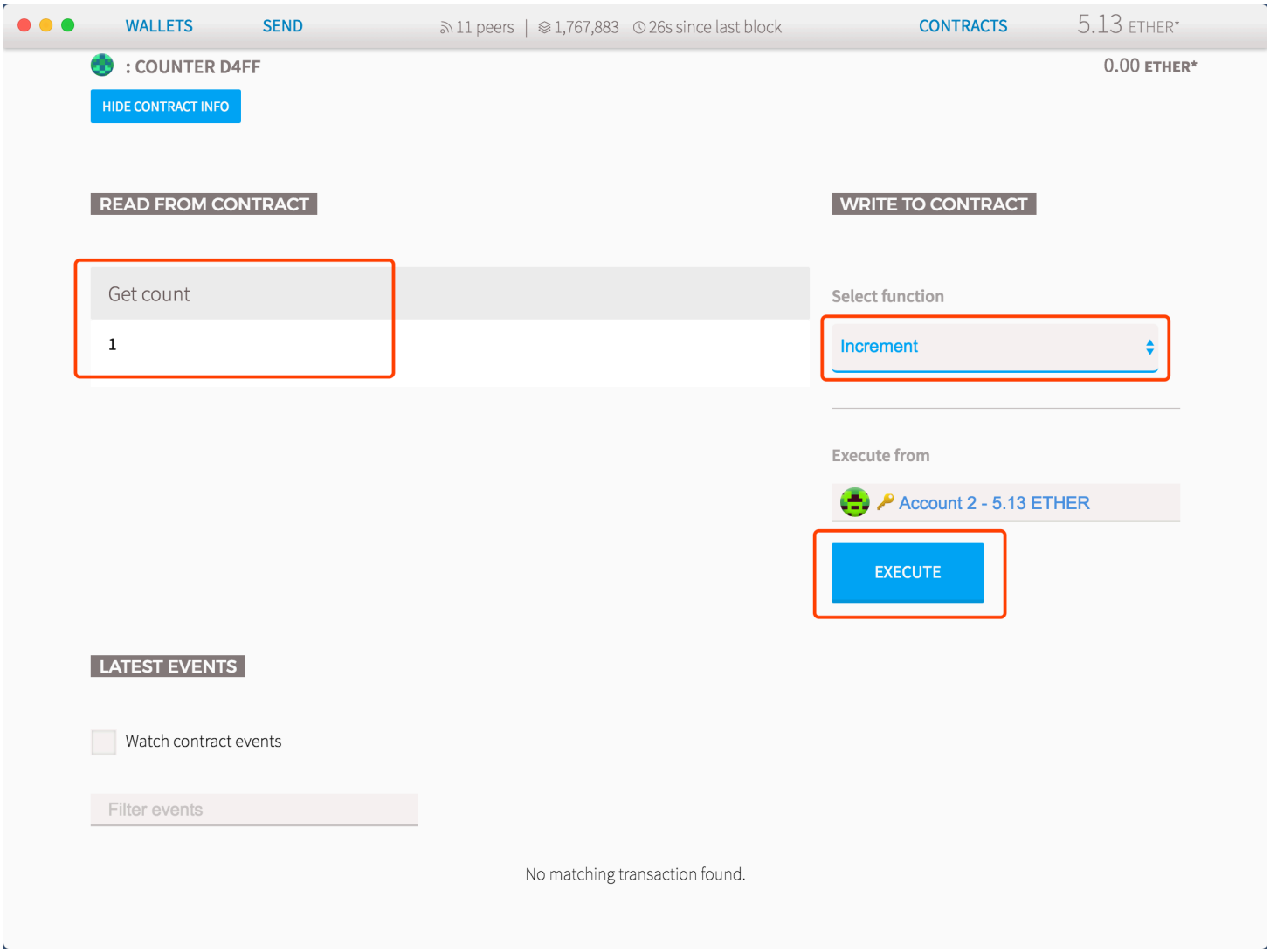
最完美的事情是有开始有结尾，合约也一样，它也可以结束。当一个合约通过 `kill` 方法将其杀死，那么我们将不能再和这个合约进行交互，如果一个合约被销毁，那么当前地址指向的是一个 `僵尸对象` ，这个僵尸对象调用任何方法都会抛出异常。你想销毁合约，需要调用 `selfdestruct(address)` 才能将其进行销毁。

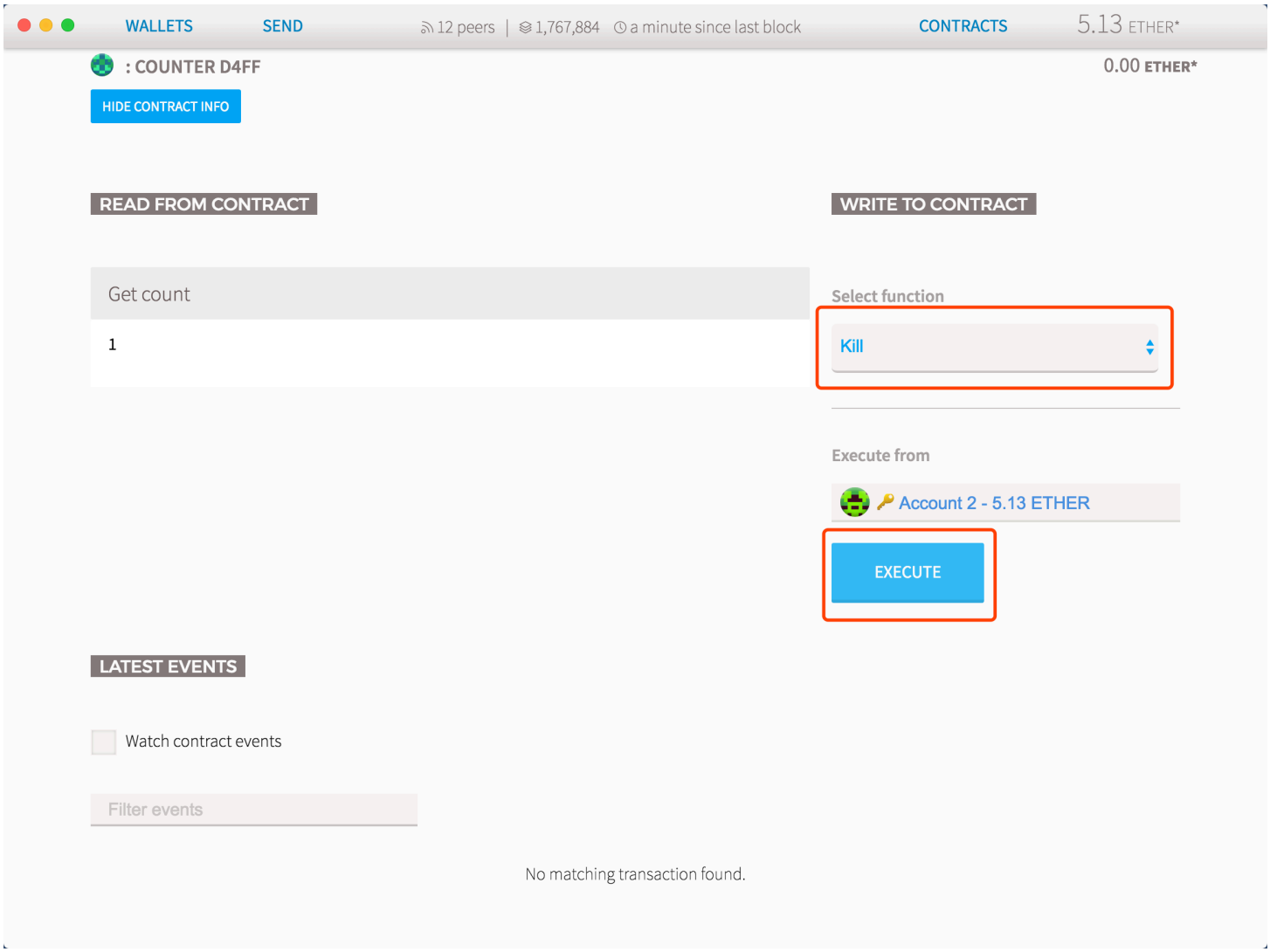
```
pragma solidity ^0.4.4;

contract Counter {

    uint count = 0;
    address owner;

    function Counter() {
       owner = msg.sender;
    }
```

```
    function increment() public {
       if (owner == msg.sender) {
          count = count + 1;
       }
    }

    function getCount() constant returns (uint) {
       return count;
    }

    function kill() {
       if (owner == msg.sender) { // 检查谁在调用
          selfdestruct(owner); // 销毁合约
       }
    }

}
```

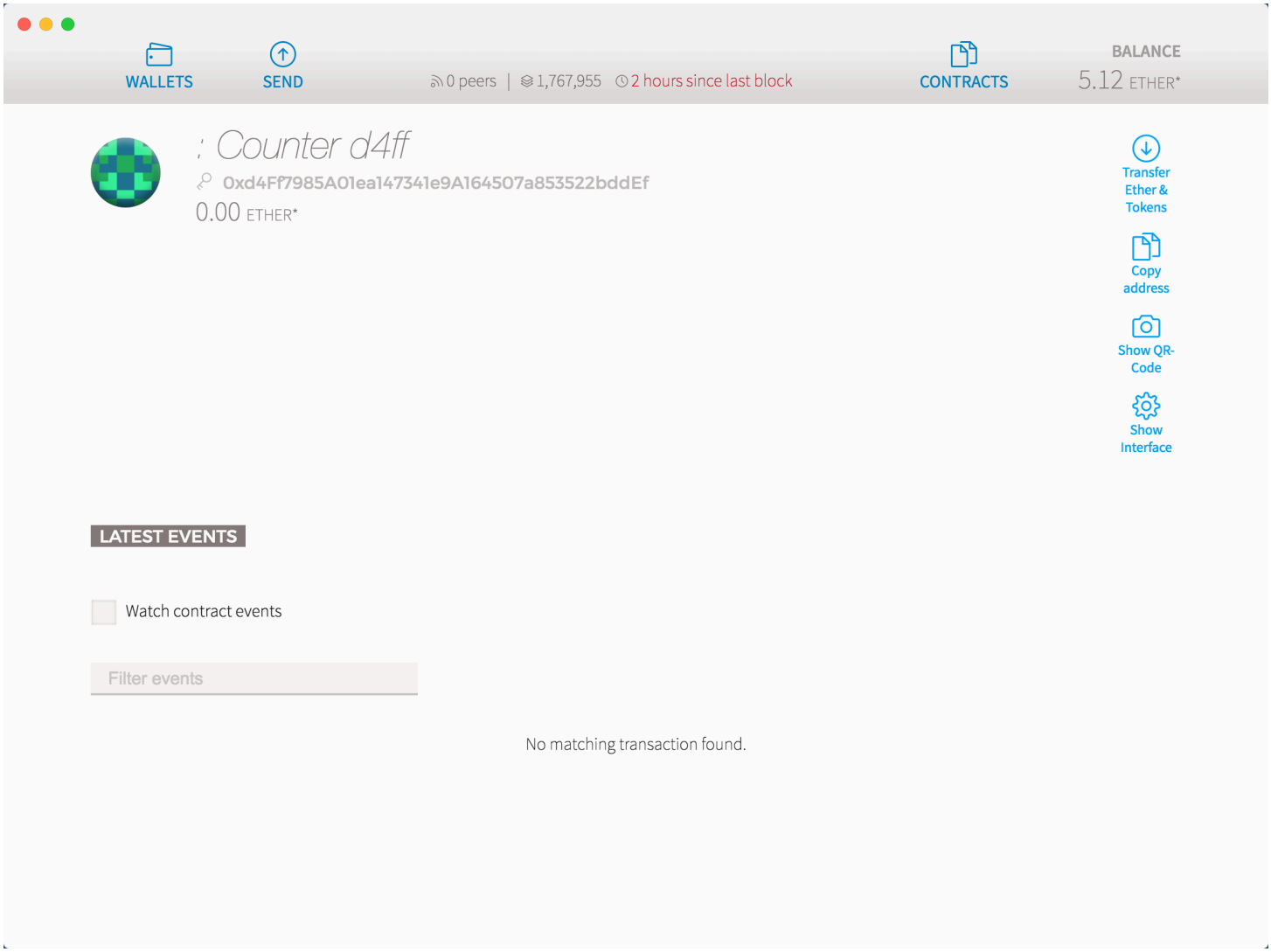
在其他一些比较老的教程里面，你可能会看到 `suicide()` 方法，但是为了语言更好的可读性，这个方法目前已经重新命名，以后如有需要，大家直接调用 `selfdestruct()` 方法就好。

接下来让我们调用我们的kill方法查看效果：


WALLETS
SEND
11 peers | 1,767,883 26s since last block
CONTRACTS
5.13 ETHER*
: COUNTER D4FF
0.00 ETHER*
HIDE CONTRACT INFO
READ FROM CONTRACT
WRITE TO CONTRACT
Get count
1
Select function
Increment
Execute from
Account 2 - 5.13 ETHER
EXECUTE
LATEST EVENTS
Watch contract events
Filter events
No matching transaction found.



执行kill方法，过一会儿刷新，你会发现刚才的合约已经没了。



小结

本节中，我们学习了一个简单但是完整的合约，详细讲解了合约的构造函数、析构函数，以及如何部署到区块链，当我们调用`kill`方法时，会调用析构函数将合约销毁的整个过程。

技术交流

- 区块链技术交流QQ群：348924182
- 「区块链部落」官方公众号

长按，识别二维码，加关注