**浅析 solidity 接口**

solidity 的接口（interface），和其他语言的接口类似，只定义方法，不定义实现。形式如下所示：

interface interfaceContract {

function interfaceTestFunc() external returns(string);

}

**注意：**

1、在两个.sol文件中都声明接口，或者两个合约写到一个.sol文件里，那就只要声明一次；

2、在一个合约中实现METHOD\_A，该合同必须继承自接口interfaceContract；

3、在另一个合约中创建一个interfaceContract实例，该实例接受实现接口的合约的地址；

4、通过这个实例调用目标合约的方法，获取目标合约的数据

合约之间的调用大致可分为两种:

1、一种是编译时同时编译的两个合约.

2、一种是分开两次编译的合约【**需要用到接口**】。

**第一种：编译时，两个合约在一个.sol文件中，同时编译两个合约.**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract AccessGetter{

uint public data = 10;

function f() public returns (uint) {

data++;

return data;

}

}

contract SetVar{

function AddData(AccessGetter accessget) public returns (uint) {

return accessget.f();

}

}

**第二种：两个合约分别在两个.sol文件中，需要*分别*编译两个合约，此时就需要通过接口.**

pragma solidity ^0.8.0;

contract AccessGetter{

uint public data = 10;

function f() public returns (uint){

data++;

return data;

}

}

观察到第一个合约的函数，接下来在第二个合约前面定义好接口

pragma solidity ^0.8.0;

// 声明接口

contract AccessGetter{

function f() public external returns(uint);

}

contract SetVar{

function AddData(AccessGetter accessget) public return(uint) {

return accessget.f();

}

合约在A.sol中：

pragma solidity ^0.4.0;

// 声明一个接口 **interfaceContract**

interface interfaceContract {

function interfaceTestFunc() returns(string);

}

// 合约**interContract** 实现 **interfaceContract**

contract interContract is interfaceContract {

function interfaceTestFunc() returns(string) {

return "This is interfaceTestFunc";

}

}

这里interContract合约实现了接口interfaceContract的interfaceTestFunc方法

合约在B.sol文件中

pragma solidity ^0.4.0;

interface interfaceContract {

function interfaceTestFunc() returns(string);

}

contract interTest {

function func(address \_addr) public returns(string) {

//\_addr 是实现interfaceContract接口的合约interContract的地址，

//\_inter 是实现interfaceContract接口的具有\_addr地址的合约实例

interfaceContract \_inter = interfaceContract(\_addr);

return \_inter.interfaceTestFunc();

}

}

上述A、B为不同的文件