智能合约高级测试

七哥

https://x.com/0xqige

测试命令说明

```
forge test -h
```

Test filtering:

```
    --match-test <REGEX>         仅运行与指定的正则表达式模式匹配的测试函数 [别名: mt]
    --no-match-test <REGEX>      仅运行不符合指定正则表达式模式的测试函数 [别名: nmt]
    --match-contract <REGEX>     仅运行与指定正则表达式模式匹配的合约中的测试 [别名: mc]
    --no-match-contract <REGEX>  仅运行不符合指定正则表达式模式的合约中的测试 [别名: nmc]
```

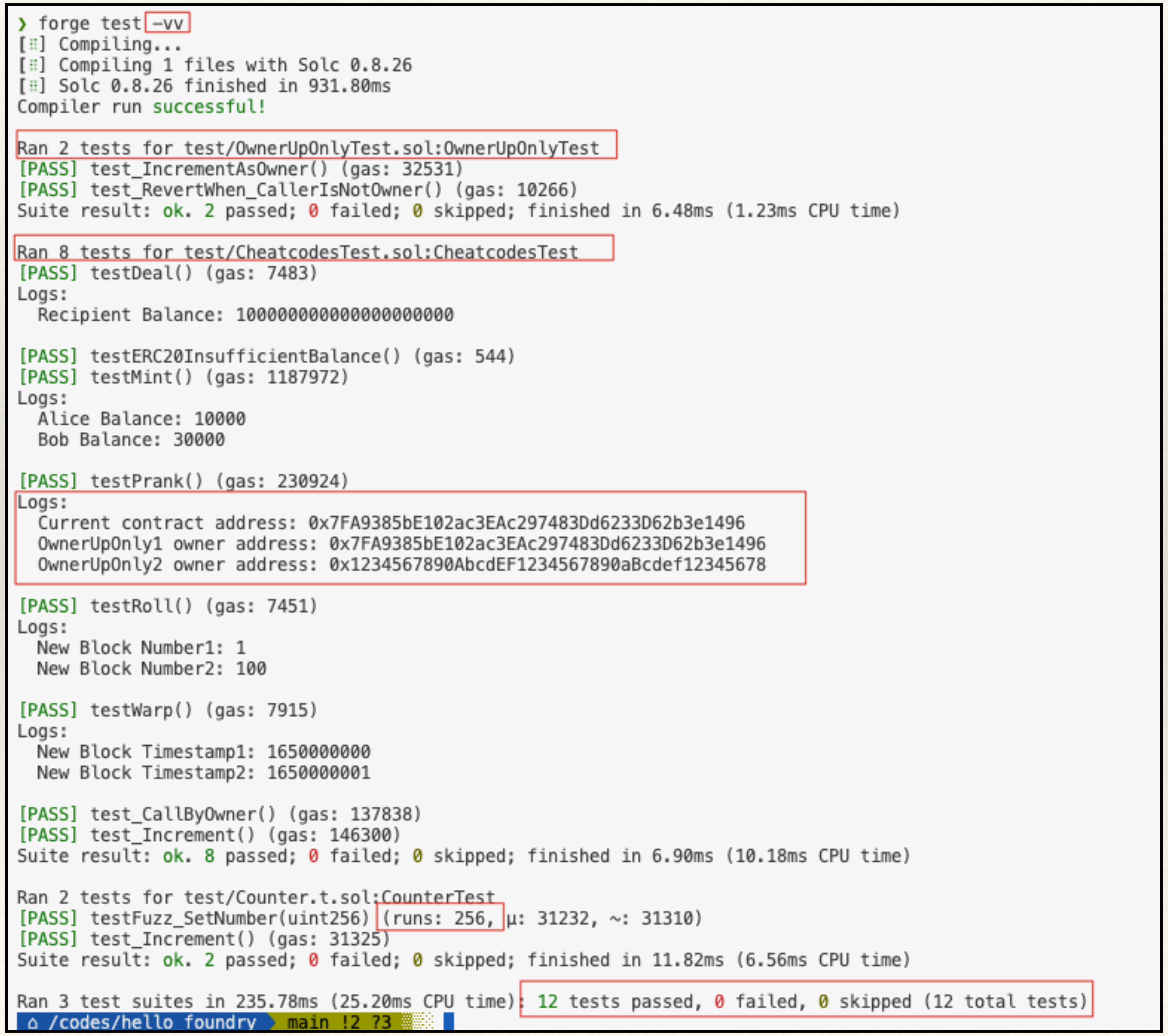
Watch options:

```
 -w, --watch [<PATH>...]    监视指定的文件或目录的更改
    --no-restart            在命令仍在运行时不要重新启动它
    --watch-delay <DELAY>   文件更新防抖延迟
    --show-progress         显示测试执行进度
```

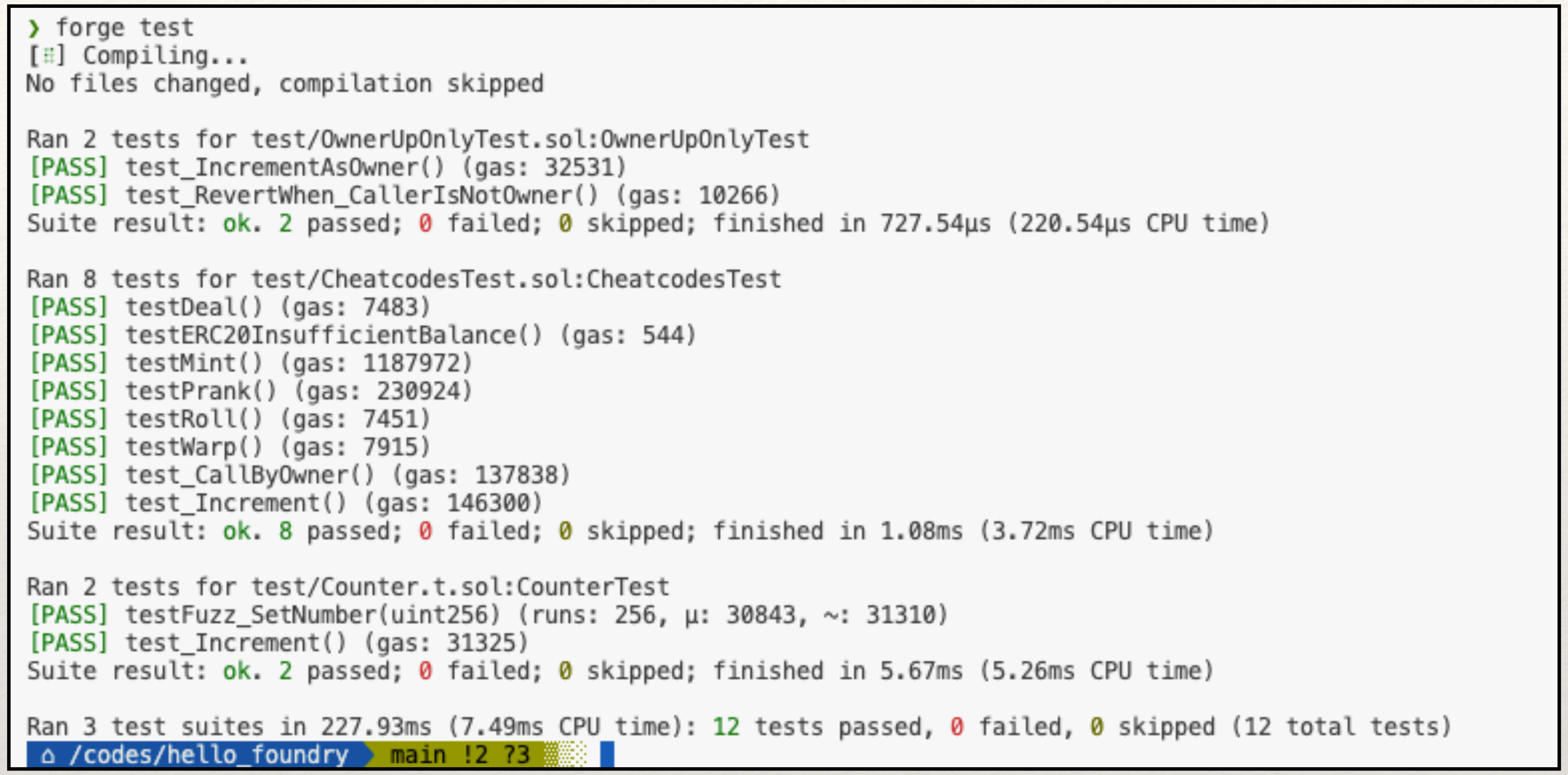
示例

```
forge test  --mt testMint -w --show-progress -vvv
```

测试报告



```
❯ forge test -vv
[⠊] Compiling...
[⠊] Compiling 1 files with Solc 0.8.26
[⠊] Solc 0.8.26 finished in 931.80ms
Compiler run successful!

Ran 2 tests for test/OwnerUpOnlyTest.sol:OwnerUpOnlyTest
[PASS] test_IncrementAsOwner() (gas: 32531)
[PASS] test_RevertWhen_CallerIsNotOwner() (gas: 10266)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 6.48ms (1.23ms CPU time)

Ran 8 tests for test/CheatcodesTest.sol:CheatcodesTest
[PASS] testDeal() (gas: 7483)
Logs:
  Recipient Balance: 100000000000000000000

[PASS] testERC20InsufficientBalance() (gas: 544)
[PASS] testMint() (gas: 1187972)
Logs:
  Alice Balance: 10000
  Bob Balance: 30000

[PASS] testPrank() (gas: 230924)
Logs:
  Current contract address: 0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496
  OwnerUpOnly1 owner address: 0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496
  OwnerUpOnly2 owner address: 0x1234567890AbcdEF1234567890aBcdef12345678

[PASS] testRoll() (gas: 7451)
Logs:
  New Block Number1: 1
  New Block Number2: 100

[PASS] testWarp() (gas: 7915)
Logs:
  New Block Timestamp1: 1650000000
  New Block Timestamp2: 1650000001

[PASS] test_CallByOwner() (gas: 137838)
[PASS] test_Increment() (gas: 146300)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 6.90ms (10.18ms CPU time)

Ran 2 tests for test/Counter.t.sol:CounterTest
[PASS] testFuzz_SetNumber(uint256) (runs: 256, μ: 31232, ~: 31310)
[PASS] test_Increment() (gas: 31325)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 11.82ms (6.56ms CPU time)

Ran 3 test suites in 235.78ms (25.20ms CPU time): 12 tests passed, 0 failed, 0 skipped (12 total tests)
⌂ /codes/hello_foundry  main !2 ?3
```



```
❯ forge test
[⠊] Compiling...
No files changed, compilation skipped

Ran 2 tests for test/OwnerUpOnlyTest.sol:OwnerUpOnlyTest
[PASS] test_IncrementAsOwner() (gas: 32531)
[PASS] test_RevertWhen_CallerIsNotOwner() (gas: 10266)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 727.54µs (220.54µs CPU time)

Ran 8 tests for test/CheatcodesTest.sol:CheatcodesTest
[PASS] testDeal() (gas: 7483)
[PASS] testERC20InsufficientBalance() (gas: 544)
[PASS] testMint() (gas: 1187972)
[PASS] testPrank() (gas: 230924)
[PASS] testRoll() (gas: 7451)
[PASS] testWarp() (gas: 7915)
[PASS] test_CallByOwner() (gas: 137838)
[PASS] test_Increment() (gas: 146300)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 1.08ms (3.72ms CPU time)

Ran 2 tests for test/Counter.t.sol:CounterTest
[PASS] testFuzz_SetNumber(uint256) (runs: 256, µ: 30843, ~: 31310)
[PASS] test_Increment() (gas: 31325)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 5.67ms (5.26ms CPU time)

Ran 3 test suites in 227.93ms (7.49ms CPU time): 12 tests passed, 0 failed, 0 skipped (12 total tests)
⌂ /codes/hello_foundry  main !2 ?3
```

v 越多，显示的测试报告越详细

- -vv: 增加显示测试过程中的日志
- -vvv：增加显示失败测试的堆栈跟踪
- -vvvv：显示所有测试的堆栈跟踪，并显示失败测试的setup跟踪
- -vvvvv: 始终显示堆栈跟踪和设置跟踪。

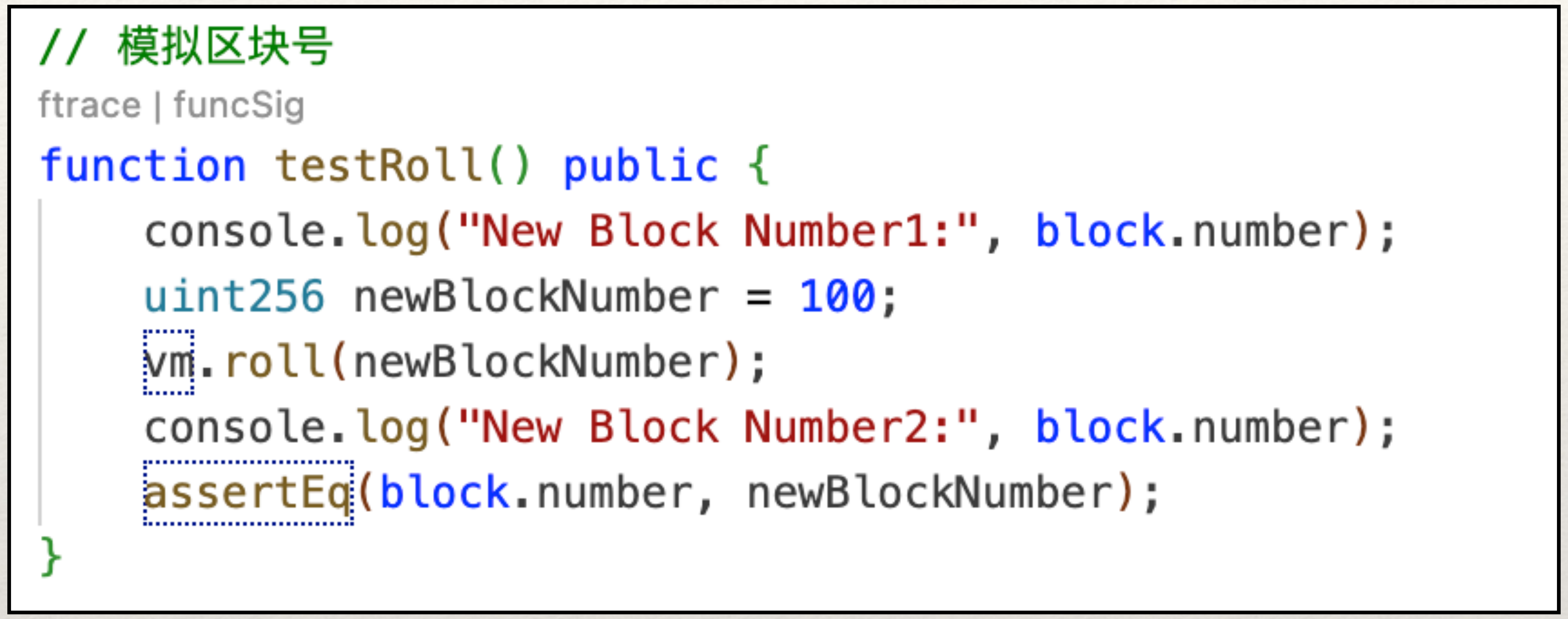
Cheatcodes介绍

Cheatcodes是一组特殊的命令，用于在测试中模拟各种场景和条件。

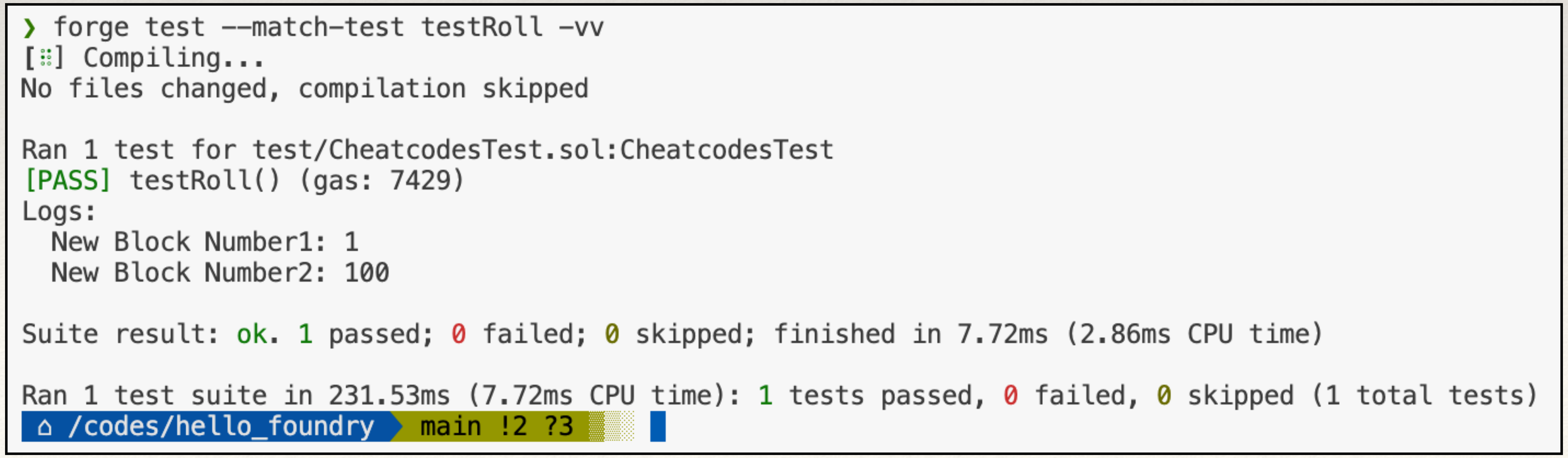
- vm.roll(uint256 blockNumber)：模拟区块号的变更。
- vm.prank(address sender)：更改消息发送者。
- vm.warp(uint256 timestamp)：改变区块时间戳。
- vm.deal(address to, uint256 amount)：重置ETH余额到指定地址。
- deal(address token, address to, uint256 amount)：重置ERC20代币余额。

修改区块高度

vm.roll(uint256 blockNumber)



```
// 模拟区块号
ftrace | funcSig
function testRoll() public {
    console.log("New Block Number1:", block.number);
    uint256 newBlockNumber = 100;
    vm.roll(newBlockNumber);
    console.log("New Block Number2:", block.number);
    assertEq(block.number, newBlockNumber);
}
```

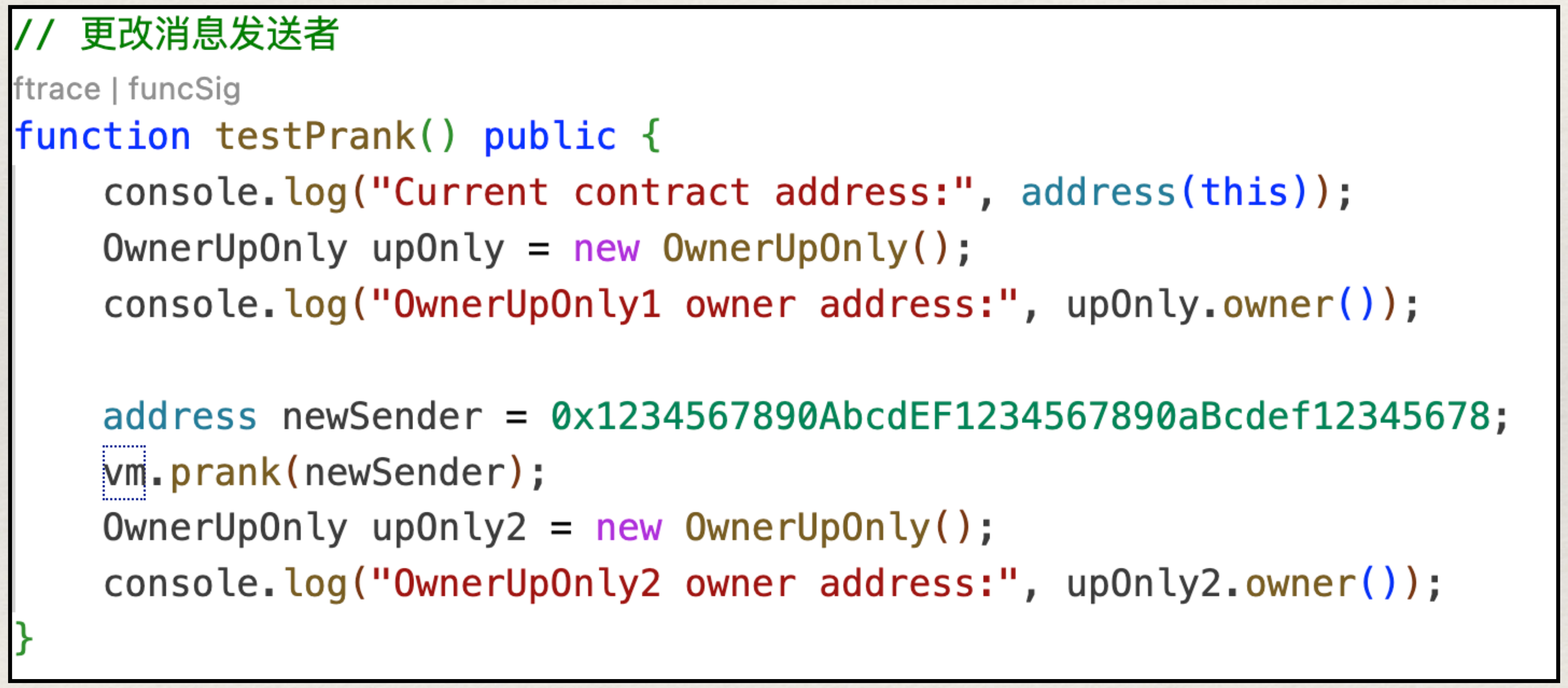


```
❯ forge test --match-test testRoll -vv
[⠊] Compiling...
No files changed, compilation skipped

Ran 1 test for test/CheatcodesTest.sol:CheatcodesTest
[PASS] testRoll() (gas: 7429)
Logs:
  New Block Number1: 1
  New Block Number2: 100

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 7.72ms (2.86ms CPU time)

Ran 1 test suite in 231.53ms (7.72ms CPU time): 1 tests passed, 0 failed, 0 skipped (1 total tests)
⌂ /codes/hello_foundry  main !2 ?3
```

更改消息发送者

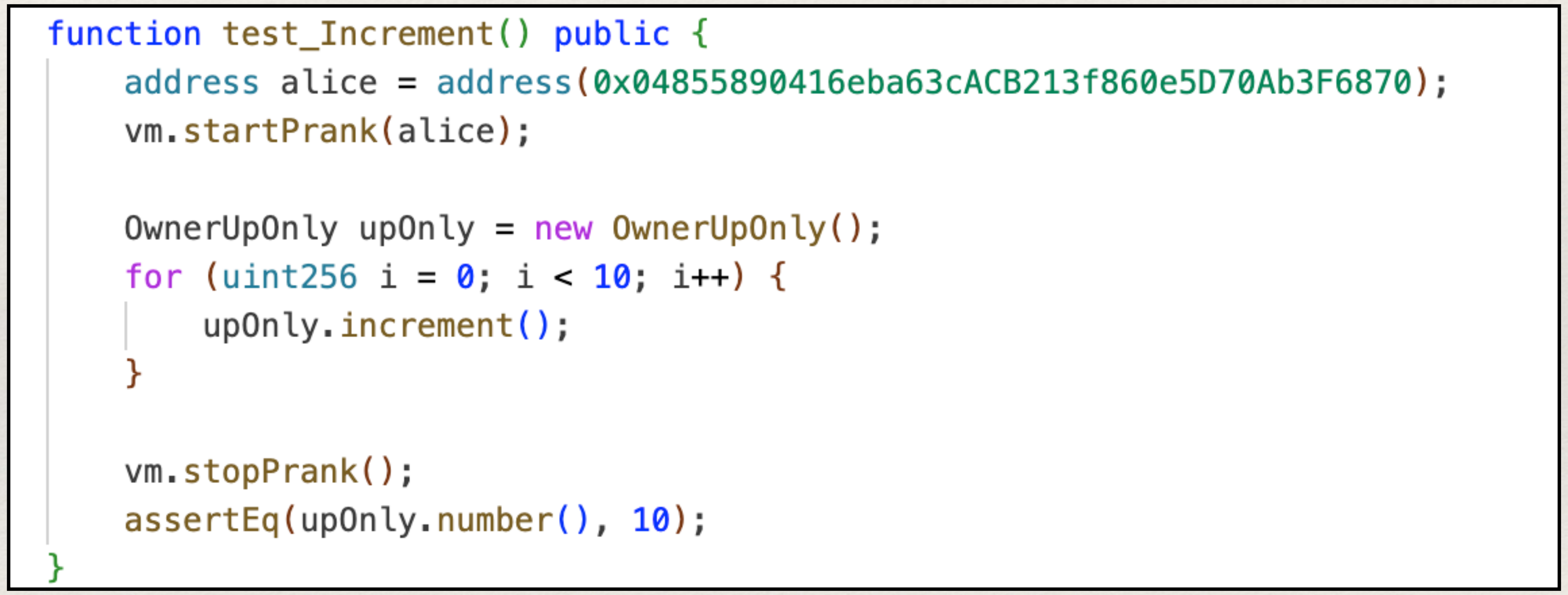
vm.prank(address newSender)

vm.prank



```
// 更改消息发送者
ftrace | funcSig
function testPrank() public {
    console.log("Current contract address:", address(this));
    OwnerUpOnly upOnly = new OwnerUpOnly();
    console.log("OwnerUpOnly1 owner address:", upOnly.owner());

    address newSender = 0x1234567890AbcdEF1234567890aBcdef12345678;
    vm.prank(newSender);
    OwnerUpOnly upOnly2 = new OwnerUpOnly();
    console.log("OwnerUpOnly2 owner address:", upOnly2.owner());
}
```

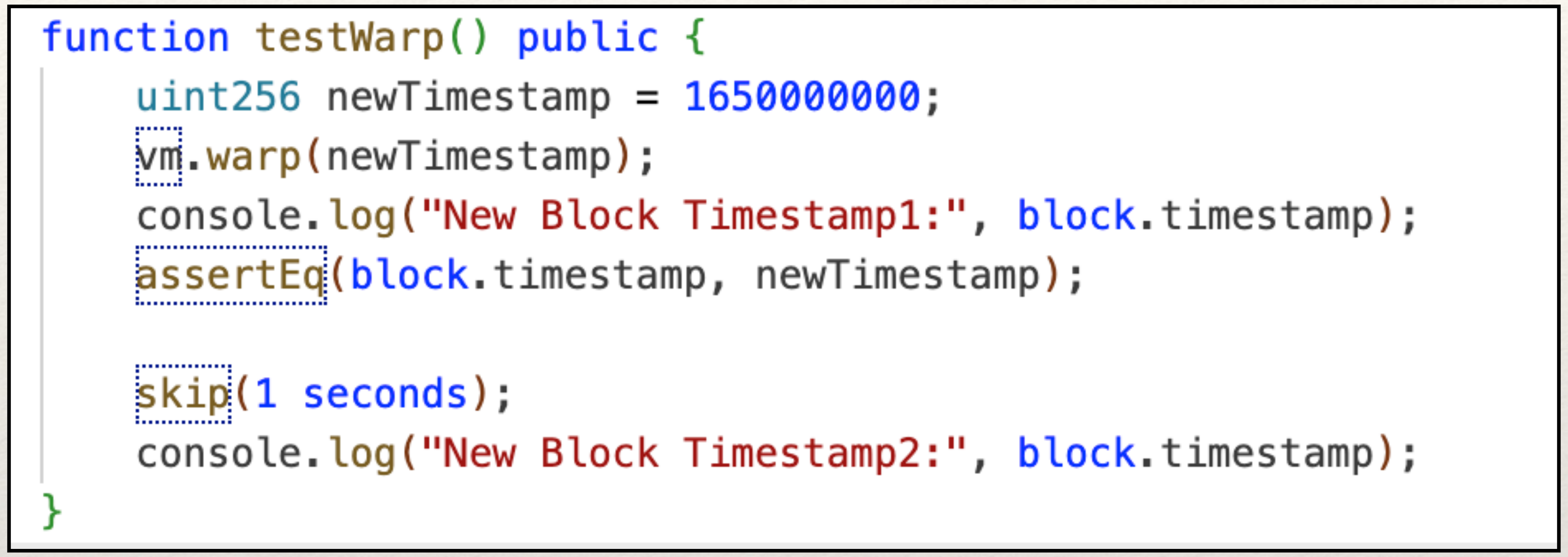
vm.startPrank



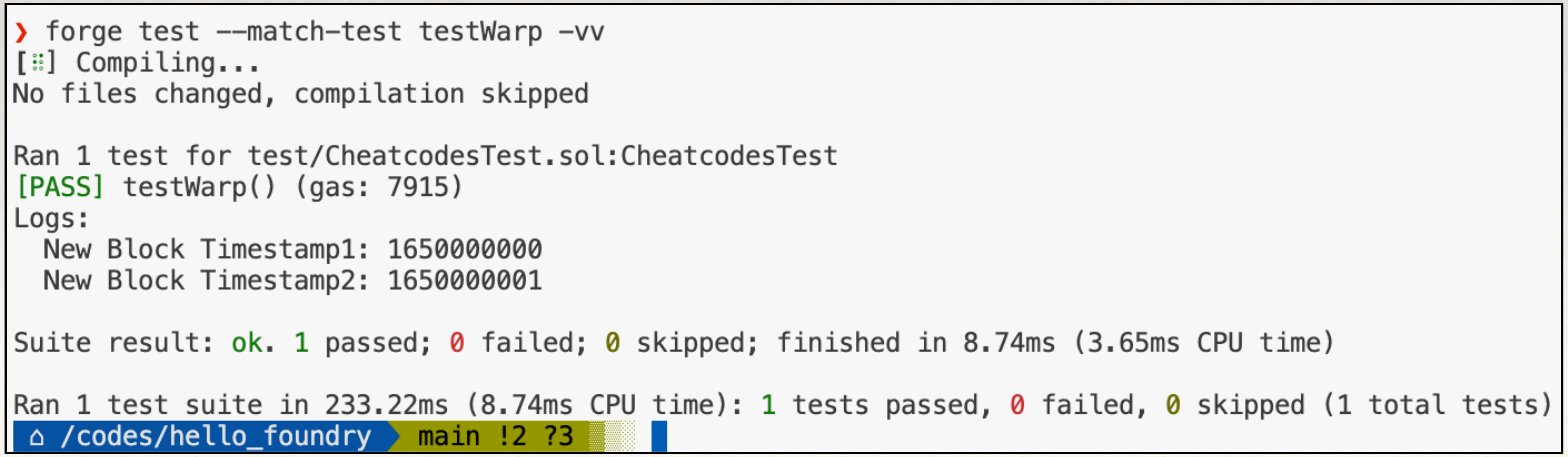
```
function test_Increment() public {
    address alice = address(0x04855890416eba63cACB213f860e5D70Ab3F6870);
    vm.startPrank(alice);

    OwnerUpOnly upOnly = new OwnerUpOnly();
    for (uint256 i = 0; i < 10; i++) {
        upOnly.increment();
    }

    vm.stopPrank();
    assertEq(upOnly.number(), 10);
}
```

改变区块时间戳

vm.warp(uint256 timestamp)



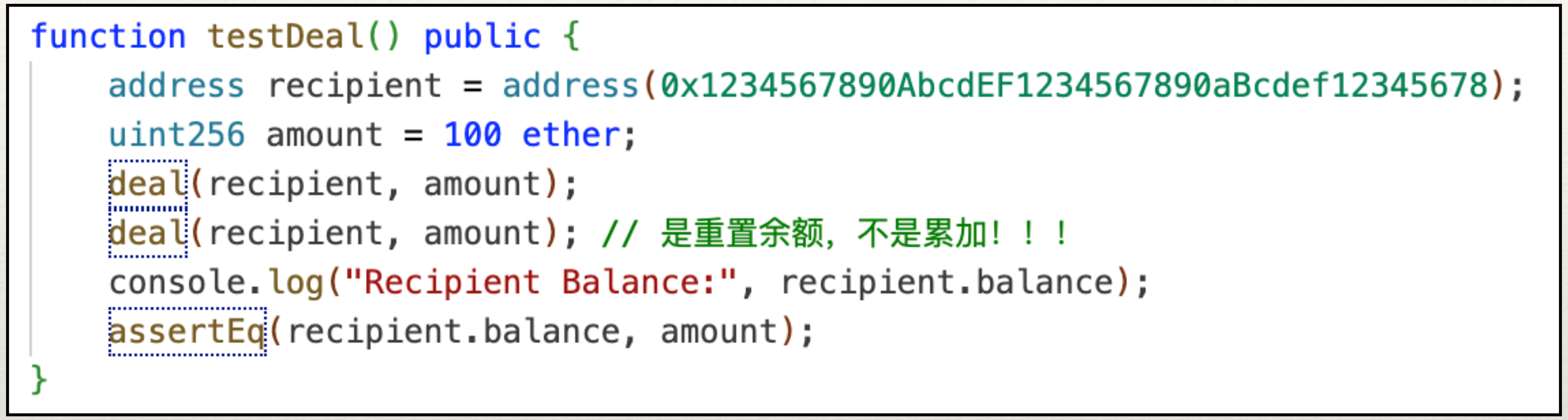
```
function testWarp() public {
    uint256 newTimestamp = 1650000000;
    vm.warp(newTimestamp);
    console.log("New Block Timestamp1:", block.timestamp);
    assertEq(block.timestamp, newTimestamp);

    skip(1 seconds);
    console.log("New Block Timestamp2:", block.timestamp);
}
```



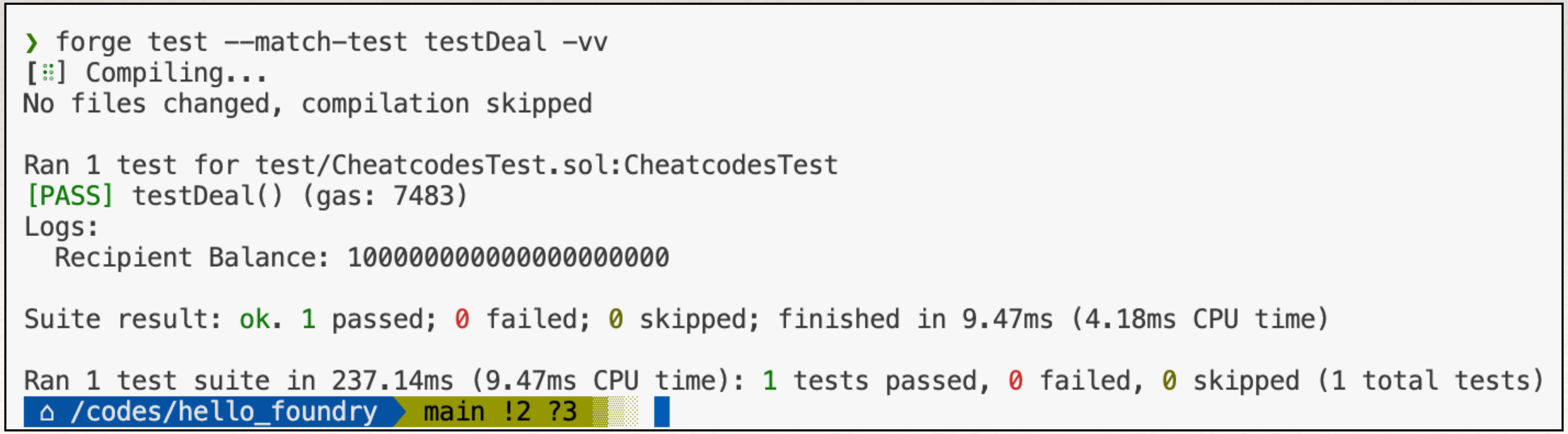
```
❯ forge test --match-test testWarp -vv
[⠊] Compiling...
No files changed, compilation skipped

Ran 1 test for test/CheatcodesTest.sol:CheatcodesTest
[PASS] testWarp() (gas: 7915)
Logs:
  New Block Timestamp1: 1650000000
  New Block Timestamp2: 1650000001

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 8.74ms (3.65ms CPU time)

Ran 1 test suite in 233.22ms (8.74ms CPU time): 1 tests passed, 0 failed, 0 skipped (1 total tests)
⌂ /codes/hello_foundry  main !2 ?3
```

重置ETH余额

vm.deal(address to,uint256 give)

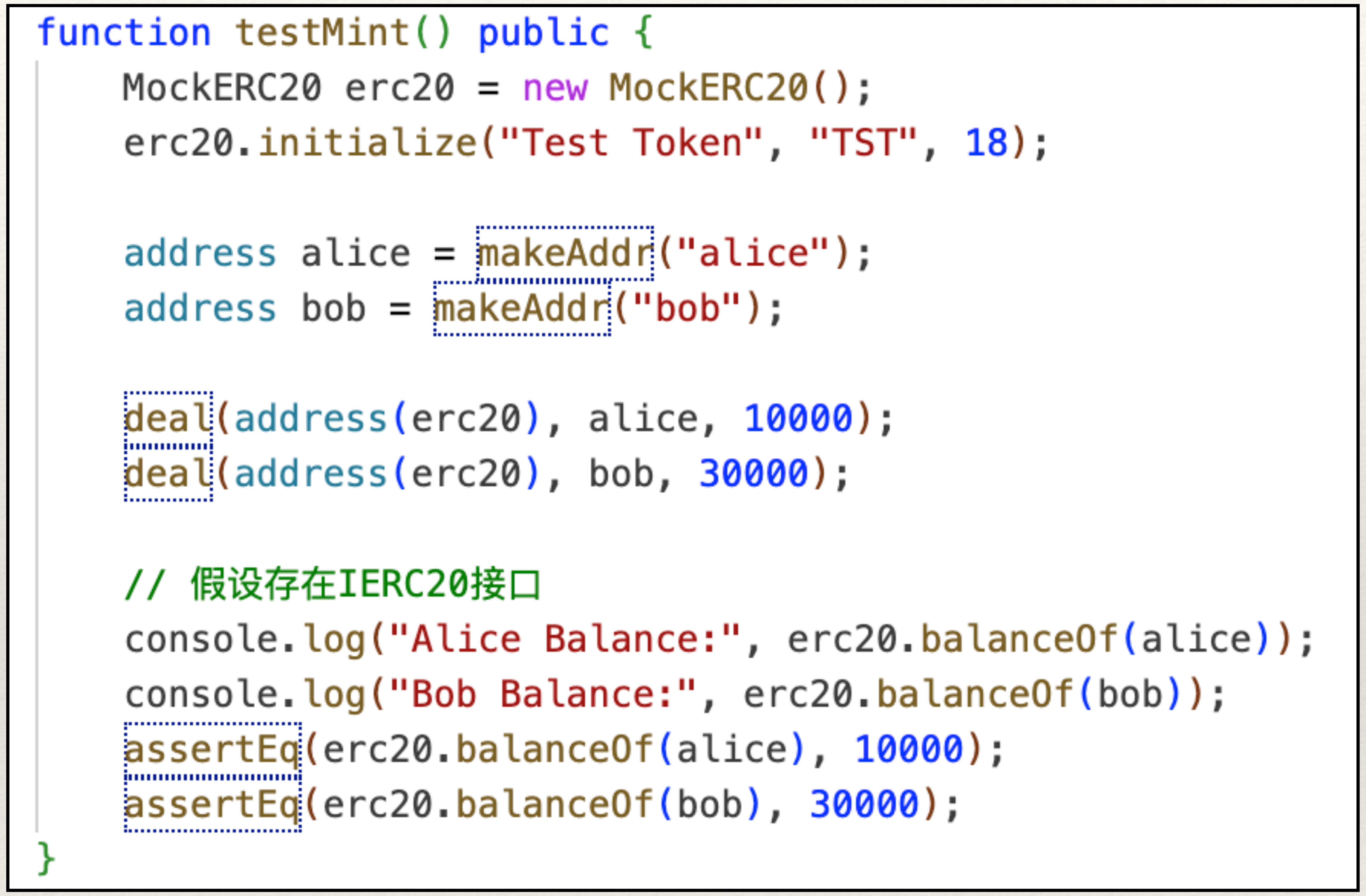


```
function testDeal() public {
    address recipient = address(0x1234567890AbcdEF1234567890aBcdef12345678);
    uint256 amount = 100 ether;
    deal(recipient, amount);
    deal(recipient, amount); // 是重置余额，不是累加！！！
    console.log("Recipient Balance:", recipient.balance);
    assertEq(recipient.balance, amount);
}
```

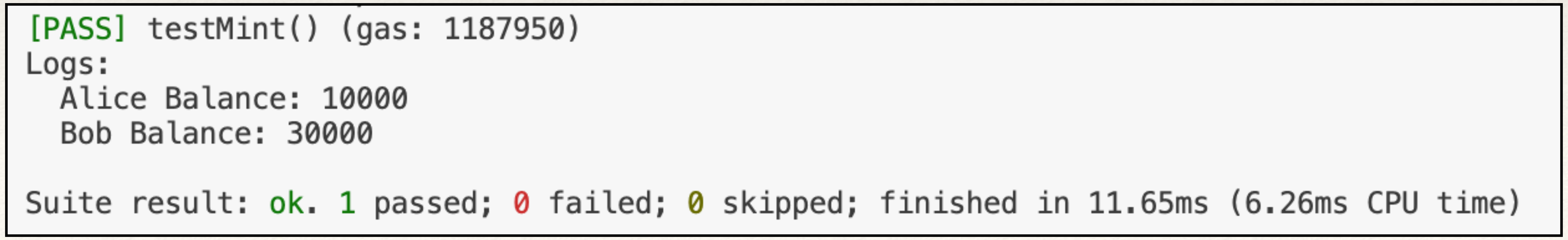


```
❯ forge test --match-test testDeal -vv
[⠊] Compiling...
No files changed, compilation skipped

Ran 1 test for test/CheatcodesTest.sol:CheatcodesTest
[PASS] testDeal() (gas: 7483)
Logs:
  Recipient Balance: 100000000000000000000

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 9.47ms (4.18ms CPU time)

Ran 1 test suite in 237.14ms (9.47ms CPU time): 1 tests passed, 0 failed, 0 skipped (1 total tests)
⌂ /codes/hello_foundry  main !2 ?3
```

重置 ERC20 余额



```
function testMint() public {
    MockERC20 erc20 = new MockERC20();
    erc20.initialize("Test Token", "TST", 18);

    address alice = makeAddr("alice");
    address bob = makeAddr("bob");

    deal(address(erc20), alice, 10000);
    deal(address(erc20), bob, 30000);

    // 假设存在IERC20接口
    console.log("Alice Balance:", erc20.balanceOf(alice));
    console.log("Bob Balance:", erc20.balanceOf(bob));
    assertEq(erc20.balanceOf(alice), 10000);
    assertEq(erc20.balanceOf(bob), 30000);
}
```



```
[PASS] testMint() (gas: 1187950)
Logs:
  Alice Balance: 10000
  Bob Balance: 30000

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 11.65ms (6.26ms CPU time)
```

断言合约执行错误

测试合约执行是否符合预期的revert

```solidity
    function test_CallByOwner() public {
        OwnerUpOnly upOnly = new OwnerUpOnly();
        upOnly.increment();

        address alice = address(0x04855890416eba63cACB213f860e5D70Ab3F6870);
        vm.expectRevert("Unauthorized");
        vm.prank(alice);
        upOnly.increment();
    }
```

三种方式

```solidity
     /// Expects an error on next call with any revert data.
    function expectRevert() external;

    /// Expects an error on next call that starts with the revert data.
    function expectRevert(bytes4 revertData) external;

    /// Expects an error on next call that exactly matches the revert data.
    function expectRevert(bytes calldata revertData) external;
```

断言合约执行错误

在 Solidity 0.8.4 版本中支持自定义 error 类型

```
  error ERC20InsufficientBalance(address sender, uint256 balance, uint256 needed);
```

ERC20转账时检查余额

```
if (fromBalance < value) {
   revert ERC20InsufficientBalance(from, fromBalance, value);
}
```

等同

```
require(fromBalance < value,
    string(abi.encodeWithSelector(ERC20InsufficientBalance.selector, from, fromBalance, value))
);
```

Test断言error 方式

```
vm.expectRevert(
 abi.encodeWithSignature("ERC20InsufficientBalance(address,uint256,uint256)",from,fromBalance,value)
);
```

```
vm.expectRevert(
 abi.encodeWithSelector(ERC20InsufficientBalance.selector,,from,fromBalance,value)
);
```

断言合约事件

测试合约执行是否有出现符合预期的合约Event记录

```
    function expectEmit(bool checkTopic1, bool checkTopic2, bool checkTopic3, bool checkData);
    function expectEmit(bool checkTopic1, bool checkTopic2, bool checkTopic3, bool checkData, address emitter)
    function expectEmit();
    function expectEmit(address emitter);
```

测试ERC20转账事件

```
    function testERC20EmitsBatchTransfer() public {

        for (uint256 i = 0; i < users.length; i++) {
            // topic0 (always checked), topic1 (true), topic2 (true), NOT topic3 (false), and data (true).
            vm.expectEmit(true, true, false, true);
            emit Transfer(address(this), users[i], 10);
        }

        // 期望出现 `BatchTransfer(uint256 numberOfTransfers)` 事件.
        vm.expectEmit(false, false, false, true);
        emit BatchTransfer(users.length);
        myToken.batchTransfer(users, 10);
    }
```

模糊测试 (Fuzz Testing)

通过随机输入数据来测试合约的健壮性

示例： 随机输入 x 测试 .setNumber(x) 的健壮性

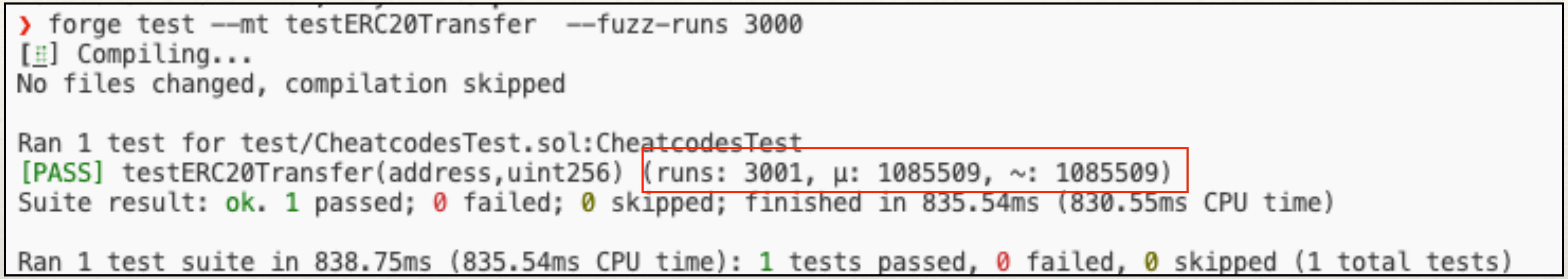
```
    function testFuzz_SetNumber(uint256 x) public {
        counter.setNumber(x);
        assertEq(counter.number(), x);
    }
```

- 支持多个随机参数
- 通过 vm.assume 来设定随机取值范围

示例： 随机转账测试

```
function testERC20Transfer(address to, uint256 amount) public {
        vm.assume(to != address(0));
        vm.assume(amount > 0 && amount < 1e9 ether);
        address alice = makeAddr("alice");
        // 模拟ERC20代币
        MockERC20 erc20 = new MockERC20();
        erc20.initialize("Test Token", "TST", 18);
        deal(address(erc20), alice, 1e9 ether);

        uint256 balanceTo = erc20.balanceOf(to);
        uint256 balanceAlice = erc20.balanceOf(alice);
        vm.prank(alice);
        erc20.transfer(to, amount);

        assertEq(erc20.balanceOf(to), balanceTo + amount);
        assertEq(erc20.balanceOf(alice), balanceAlice - amount);
}
```

模糊测试报告



```
❯ forge test --mt testERC20Transfer  --fuzz-runs 3000
[⠊] Compiling...
No files changed, compilation skipped

Ran 1 test for test/CheatcodesTest.sol:CheatcodesTest
[PASS] testERC20Transfer(address,uint256) (runs: 3001, μ: 1085509, ~: 1085509)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 835.54ms (830.55ms CPU time)

Ran 1 test suite in 838.75ms (835.54ms CPU time): 1 tests passed, 0 failed, 0 skipped (1 total tests)
```

方法1：foundry.toml 中配置

```
[fuzz]
runs=256
max_test_rejects=65536
```

方法2：单个TestCase中配置

```
    /**
     * forge-config: default.fuzz.runs = 1024
     * forge-config: default.fuzz.max-test-rejects = 500
     */
    function testERC20Transfer(address to, uint256 amount) public
```

不变量测试 (Invariant Testing)

验证合约在多次操作后测试不变量（变量值始终不变）

不变量例子:

- 对于 Uniswap 来说，“*xy=k* 公式始终成立”
- 对于 ERC-20 代币而言，“所有用户余额的总和等于总供应量”。
- Gas优化后，“新版本合约Gas开销总是低于旧版本”

不变测试也在函数名称前添加 invariant 前缀来表示

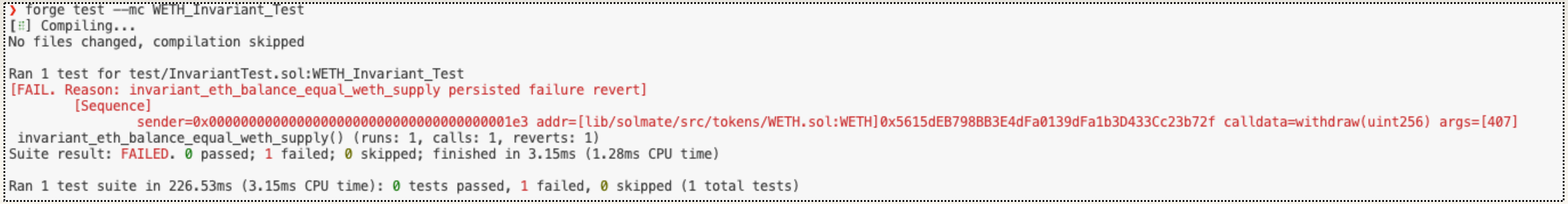
```
    function invariant_example() external {
        if (protocolCondition) return;

        assertEq(val1, val2);
    }
```

不变量测试 (Invariant Testing)

测试失败

```solidity
contract WETH_Invariant_Test is Test {
    WETH public weth;

    function setUp() public {
        weth = new WETH();
    }
    /// forge-config: default.invariant.fail-on-revert = true
    function invariant_eth_balance_equal_weth_supply() public {
        assertEq(address(weth).balance, weth.totalSupply());
    }
}
```

如何在测试中设置测试环境?



```
❯ forge test --mc WETH_Invariant_Test
[⠊] Compiling...
No files changed, compilation skipped

Ran 1 test for test/InvariantTest.sol:WETH_Invariant_Test
[FAIL. Reason: invariant_eth_balance_equal_weth_supply persisted failure revert]
        [Sequence]
                sender=0x00000000000000000000000000000000000001e3 addr=[lib/solmate/src/tokens/WETH.sol:WETH]0x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f calldata=withdraw(uint256) args=[407]
 invariant_eth_balance_equal_weth_supply() (runs: 1, calls: 1, reverts: 1)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 3.15ms (1.28ms CPU time)

Ran 1 test suite in 226.53ms (3.15ms CPU time): 0 tests passed, 1 failed, 0 skipped (1 total tests)
```

不变量测试 (Invariant Testing)

需编写Handle合约来处理细节

实现一个 Hanlder，可以使用 Test功能

```
contract WETHHandler is Test {
    WETH public weth;

    constructor(WETH weth_) {
        weth = weth_;
        deal(address(this), 100000 ether);
    }

    function deposit(uint256 amount) public {
        vm.assume(amount > 0 && amount < 100000 ether);
        weth.deposit{value: amount}();
    }

    function withdraw(uint256 amount) public {
        vm.assume(amount > 0 && amount < 100000 ether);
        weth.withdraw(amount);
    }
}
```

实现一个不变量测试，targetContract 指定目标合约

```
contract WETH_Invariant_Test02 is Test {
    WETH public weth;
    WETHHandler public handler;

    function setUp() public {
        weth = new WETH();


        handler = new WETHHandler(weth);
        targetContract(address(handler));
    }

    function invariant_eth_balance_equal_weth_supply()
    public {
        assertEq(address(weth).balance, weth.totalSupply());
    }
}
```

作业说明

- 代码在自己的 github 提交
- 在 decert.me 提交领取证书
- **不可抄袭作业**，一经发现将不再检查抄袭者作业！

作业

- 测试 NFTMarket 合约：测试Case
  - 上架NFT：测试上架成功和失败情况，要求断言错误信息和上架事件。
  - 购买NFT：测试购买成功、自己购买自己的NFT、NFT被重复购买、支付Token过多或者过少情况，要求断言错误信息和购买事件。
  - 模糊测试：测试随机使用 0.01-10000 Token价格上架NFT，并随机使用任意Address购买NFT
  - 「可选」不可变测试：测试无论如何买卖，NFTMarket合约中都不可能有 Token 持仓

谢谢