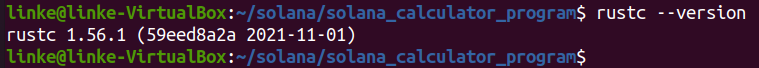
使用solana program构建一个计算器

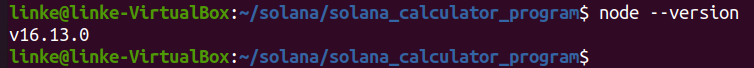
Solana是2021年兴起的一条明星公链，目前是排名前列的主流公链之一，其智能合约被称为program，是用rust语言开发，今天演示一下如何在solana上开发一个计算器program，并编写测试代码对其进行测试。

* 1.安装环境

首先安装rust，我已经安装过了，就不演示了，查看一下rust的版本



还需要安装nodejs，已经安装过了，查一下版本



安装solana（注意是全局安装，并不是安装在某个目录中）

文本

描述已自动生成

安装solana之后，注销然后重新登录用户，或者执行

export PATH="/home/linke/.local/share/solana/install/active\_release/bin:$PATH"

全局安装mocha

sudo npm install -g mocha

文本

描述已自动生成

全局安装Anchor，这是solana目前最好的开发工具，类似于hardhat对于以太坊的地位。它提供了一个Rust DSL(基本上是一个更简单的Rust)，可以与IDL、CLI和workspace管理一起使用。Anchor从传统的Solana program中抽象出了许多潜在的安全漏洞，负责序列化和反序列化，将大量繁琐的代码变成宏和许多其他好的特性。

sudo npm i -g @project-serum/anchor-cli

文本

描述已自动生成

查看anchor的版本

anchor --version

文本

描述已自动生成

* 2.通过solana命令行运行命令

首先执行solana config get，获取solana当前的配置

文本

描述已自动生成

在开发program时环境最好设置为本地环境，上图中显式RPC URL是mainnet，需要运行以下命令设置为本地环境

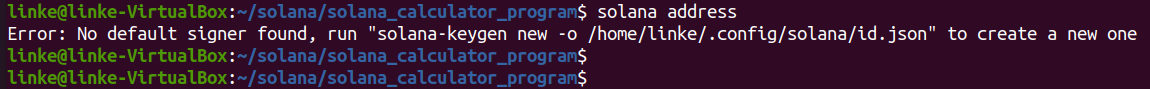
solana config set --url localhost

文本

描述已自动生成

接下来需要知道我们的帐户/钱包地址，并空投一些SOL tokens到这个地址，以处理与Solana program交互所带来的所有部署、交易等成本。

执行solana address，获取地址



如果之前没有设置过密钥对，那么将不会在结果中有密钥对路径。需要新建一个密钥对。

目前/home/linke/.config/solana/id.json是不存在的。

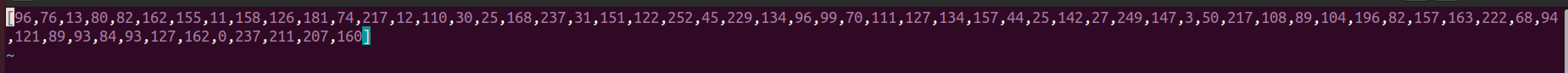
执行以下命令新建一个密钥对

solana-keygen new -o /home/linke/.config/solana/id.json

文本

描述已自动生成

BIP39 Passphrase我没有设置，直接按enter键跳过。这个新建的密钥被放在/home/linke/.config/solana/id.json文件中，查看一下这个文件



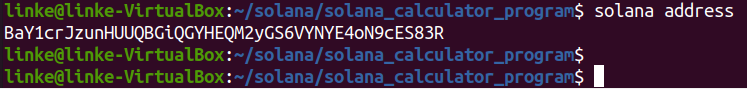
公钥为

BaY1crJzunHUUQBGiQGYHEQM2yGS6VYNYE4oN9cES83R

助记词如下

nephew reason mutual idle slush flat define orient dream antenna next panther

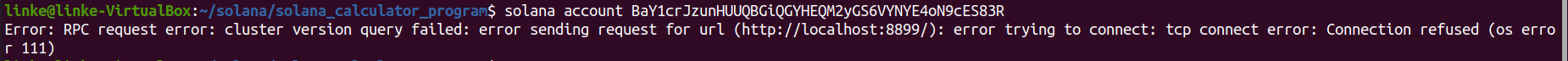
现在有了密钥对，再次执行solana address



BaY1crJzunHUUQBGiQGYHEQM2yGS6VYNYE4oN9cES83R

执行以下命令获取账户详情

solana account BaY1crJzunHUUQBGiQGYHEQM2yGS6VYNYE4oN9cES83R



报错，原因应该是还没启动本地网络。接下来先启动本地网络。可以将这个本地网络看作运行在操作系统上的模拟Solana区块链。开发和测试我们的program都需要这个网络。

要在一个单独的terminal中启动本地网络，使用以下命令

solana-test-validator

文本

描述已自动生成

在不做任何操作的情况下，Processed Slot, Confirmed Slot, Finalized Slot一直在递增。

现在再次执行

solana account BaY1crJzunHUUQBGiQGYHEQM2yGS6VYNYE4oN9cES83R

就成功了

文本

描述已自动生成

可以看到，余额是500000000 SOL，owner是11111111111111111111111111111111

接下来可以请求给当前账号空投100 SOL，执行

solana airdrop 100

文本

描述已自动生成

可以看到，余额增加了100

* 3.搭建Anchor项目

初始化一个Anchor项目，然后继续做我们自己的修改。进入自己新建的一个目录/home/linke/solana/solana\_calculator\_program，然后执行

anchor init mycalculatordapp

cd mycalculatordapp

图片包含 图表

描述已自动生成

进入目录，目录结构如下图

文本

描述已自动生成

首先，我们检查是否可以看到programs, app, migrations目录等。

如果有这些目录，我们可以去programs/mycalculatordapp/src/lib.rs查看Anchor提供给我们的默认program。这是在Anchor上最基本的例子，只是一个用户定义的函数Initialize无论何时被调用都会成功退出程序，没什么特别的。现在，让我们使用以下命令编译这个program

anchor build

文本

描述已自动生成

发生这个错是因为没有使用nightly版本，把rust从稳定版本切换到nightly版本，执行命令

rustup default nightly

文本

描述已自动生成

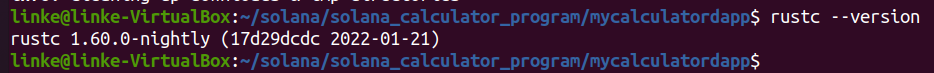
切换到nightly版本之后，发现还是报同样的错，然后根据stackoverflow的一个链接

<https://stackoverflow.com/questions/70639179/rust-unable-to-run-edition2021>

需要更新rust，执行如下命令

rustup update

成功更新版本之后再次查看版本



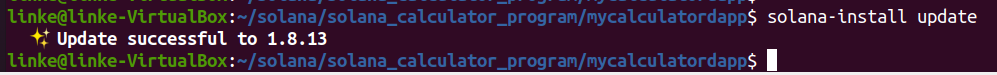
现在rust的版本已经最新了，再次执行anchor build，还是报同样的错。

根据另一个stackoverflow链接

<https://stackoverflow.com/questions/70700551/cargo-version-2021-required-on-solana-anchor-build>

是由于solana的版本太低，需要更新solana，执行

solana-install update



现在再次执行anchor build，就成功了，log太长，就只截头尾了

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

从log的最后一句话可以看出，部署后program的keypair文件为

/home/linke/solana/solana\_calculator\_program/mycalculatordapp/target/deploy/mycalculatordapp-keypair.json

anchor build将在项目目录中创建一个名为target的新文件夹。这个目标文件夹包含idl目录，该目录也包含我们program的idl。IDL/Interface Description Language（接口描述语言）描述了智能合约中公开的指令，和solidity中的ABI很相似，用于测试和前端集成。接下来，我们可以继续测试这个program，这样我们就可以熟悉如何在Anchor中完成测试。进入tests/mycalculatordapp.ts，在这里，将看到一个用javascript编写的测试，用于交互和测试默认program。

build后target目录结构

文本

描述已自动生成

IDL文件内容

文本

描述已自动生成

测试文件的内容

文本

描述已自动生成

此处暂时不解释测试文件的内容。

接下来，进入之前运行本地solana区块链（solana-test-validator）的terminal，使用Ctrl+C退出本地区块链

文本

描述已自动生成

接下来，执行测试文件中的测试，执行命令

anchor test



可以看到，测试通过。

接下来进入programs目录，开始导入一些由Anchor提供的很好的Rust crates，它将帮助我们构建计算器app。

* 4.定义我们的programs

清除programs/mycalculatordapp/src/lib.rs中除了模板代码之外的其他代码，清除前：

文本

描述已自动生成

清除后：

文本

描述已自动生成

现在，让我们简单地定义用于编写计算器dapp的函数签名（function signatures），而暂时不需要编写逻辑。程序的大部分放在#[program]宏下的模块中。我们只需在pub mod mycalculatordapp下定义它们，稍后再编写逻辑。这些函数的定义是这样的

文本

描述已自动生成

这里，pub是public的意思，fn是function的意思，这意味着它们是可以从program中调用的公共函数，也就是说，它是一个客户端可调用的program函数。这些函数的第一个参数总是Context<>，它由solana账户数组和program ID组成，这在本质上是调用solana上任何program所需的数据。第一个函数的下一个参数是一个名为init\_message的字符串，我们将使用它作为存储在计算器上的消息(想想每次你启动你的手机，pc，计算器等时，你是如何看到一些文本的)。在其他函数中，num1和num2形参是整数类型，我们将对它们执行数学运算。ProgramResult是函数的返回类型，它实际上只是提供函数结果和/或错误的一种更简单的方法。

* 5.为我们的第一个solana program编写逻辑

现在让我们先写出create函数的逻辑，让我们首先明确这个函数和程序的总体意图。这里我们需要关注三件事。第一个是我们将存储在计算器中的greeting消息，第二个将是所有数学运算的结果，第三个是余数，这将用于除法，因为Anchor当前不支持浮点值。因此，我们希望我们的计算器账户(将处理program的所有计算内容的主账户)有三个字段，即:greeting、result、remainder。此外，由于相同的账户将用于计算并且具有不同的参数，我们希望计算器账户是可变的，即能够把更改持久化。现在在create函数中编写以下代码逻辑。

文本

描述已自动生成

Ok(())语法用于ProgramResult类型的错误处理。你可以把Ok(())想象成一个门，它让program在没有错误时继续，但在遇到错误时将program发送到另一种状态。

关于solana账户的小知识：

一个账户（account）实际上并不是一个钱包。相反，它是合约在调用之间持久化数据的一种方式。这包括base\_account中的计数（count）等信息，以及有关账户权限的信息。Accounts以lamports的形式支付租金，如果它用完，则该账户将从区块链中清除。有两年租金价值的账户是免租金的，可以永远留在链上。

* 6.定义计算器账户（account）的结构

现在我们来定义计算器账户是什么。正如前面提到的，Solana上的所有东西/内容都是一个账户，所以我们将使用Anchor的强大的宏将一个struct转换为我们的计算器账户。

接下来在pub mod mycalculatordapp之外定义account，如下

文本

描述已自动生成

现在就可以开始实现计算器的实际功能了。

* 7.第一个计算器函数的逻辑

让我们先写出加法的逻辑。正如您可能会想的那样，这里我们只需要将两个参数相加的结果保存在计算器账户的result字段中。在add函数中编写以下代码

文本

描述已自动生成

注意到Context<Addition>，如前所述，它是必须传递给这个特定函数的账户列表。我们只需要向add函数发送计算器账户，不需要其他账户。现在，有了这些知识，让我们定义Addition，编写计算器账户的声明，如下

文本

描述已自动生成

* 8.编写减法、乘法、除法

和开发加法类似，举一反三，开发减法、乘法、除法，唯一特殊的是除法，因为有余数。开发完成后program代码如下

文本

描述已自动生成

形状

描述已自动生成

程序代码还缺少一点东西，在create函数中有个Create结构体还没有定义。对于这个结构体，我们希望传递三个账户，第一个账户显然是计算器帐户，因为它在这个函数本身中使用。第二个账户是用户账户，第三个账户是system\_program。使这个结构体更特别的是，在这个结构体中，我们必须给出命令来实际创建计算器账户(我们已经在后面的加减乘除函数中使用了这个账户)。这个计算器账户的创建将花费我们一些钱(SOL)，这将由我们刚刚提到的用户账户支付，以及我们有空间（space）参数，我们指定计算器账户需要多少空间，最后system\_program只是Solana区块链的系统规格，表现为账户的形式。

Create struct的代码如下

文本

描述已自动生成

正如前面所说的，我们使用了派生的Accounts宏，因为我们必须在这里合并3个账户，并且对于这三个账户，我们分别使用了account宏。现在来看看这些宏使用的参数。init参数用于创建一个由当前program(即我们的mycalculatordapp program)拥有的新账户。无论何时使用init参数，我们都必须指定payer或将在Solana区块链上为创建账户付费的账户，以及用于创建新账户的space参数。

mut参数将一个账户标记为可变的，这本质上意味着我们的账户将被更改，Solana将需要更新账户中的数据。因此，始终使用mut参数来持久化更改。

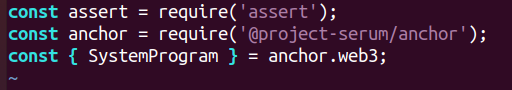
这里使用的另一个新概念是Signer类型。这用于强制authority账户签署交易的约束。

到此为止，计算器program开发完毕。

* 9.测试计算器program

进入tests/mycalculatordapp.ts文件，把内容全部清空，从头开始写测试用例。

第一步是导入必要的库和常量



现在，因为我们将使用Mocha来测试我们的program，所以我们将创建测试用例的框架。基本上，Mocha是这样工作的，它把describe块作为测试块，在这些describe块中，有很多使用it块编写的测试用例。创建下面的代码框架：

文本

描述已自动生成

provider是一个到Solana网络连接的抽象。在测试中，Anchor框架将基于环境(anchor.Provider.local())为我们创建provider。

现在，program是一个抽象，它结合了Provider、idl和programID(在构建program时生成)，并允许我们针对我们的program调用RPC方法。calculator变量是使用anchor.web3生成的keypair，我们将用它来测试我们的program。

* 10.编写第一个测试用例测试create函数

调用program函数的方法非常简单。我们将使用program RPCs (远程过程调用)来访问该函数，然后我们将使用web3.js库来创建账户，这些账户必须作为参数传递给这些函数。第一个测试用例的代码如下

文本

描述已自动生成

现在，我们在上面的代码中所做的只是通过使用web3库生成一个新账户来创建一个计算器账户。然后使用program RPC，我们调用了create函数，并为该函数提供了所需的参数，它们是calculator, user, systemProgram和init\_message字符串。

在这个函数运行后，我们只需获取计算器账户，检查它的greeting字段，并验证它是否已更改为Welcome to Solana。之后，我们将计算器账户保存在一个名为\_calculator的变量中，以便以后可以引用它。

* 11.编写第二个测试用例测试add函数

在第二个测试中，我们开始测试我们编写的Solana program的计算功能。首先，我们将编写加法功能的测试。为了测试，我们不能直接使用数字，因此我们将不得不将它们转换为Anchor big numbers。现在，编写下面的代码来测试mycalculatordapp program的add函数。

文本

描述已自动生成

在使用RPC运行add函数之后，我们获取计算器账户并检查计算器账户的字段。greeting字段应该不受影响，仍然是“Welcome to Solana”，result字段现在应该等于2和3的和，即5。

* 12.编写减法、乘法、除法的测试

依葫芦画瓢，编写剩余的测试，完整的测试代码如下

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

测试用例写完了，接下来就是运行测试

* 13.运行测试

在运行测试之前，确保本地网络即solana-test-validator没有运行。且确保solana config get命令获取到的network是localhost。

运行anchor test，遇到如下错误

文本

描述已自动生成

解决方案：

首先，把tests/mycalculatordapp.ts改为tests/mycalculatordapp.js，即修改文件格式，从ts变为js

然后，在Anchor.toml中，把test = "yarn run ts-mocha -p ./tsconfig.json -t 1000000 tests/\*\*/\*.ts"改为test = "mocha -t 1000000 tests/"

电脑屏幕截图

描述已自动生成

然后再运行anchor test，所有测试全部通过，见下图。

文本

描述已自动生成

以上就演示了如何在Solana区块链上创建计算器program，并测试其功能。Solana是一条高性能的明星公链，发展势头强劲，值得关注。