区块链技术及其应用

期末设计

姓名：曹啸

学号：2017080106018

所属学院和班级：机电 机械六班

2020年 6月 6日

使用go语言创建区块链

1 前言

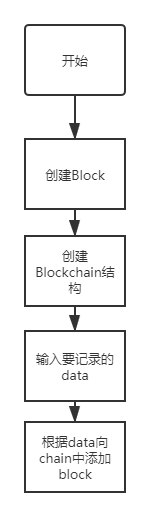
区块链技术是21世纪一项具有革命性的信息技术，本质上是一个去中心化的数据库，是指通过去中心化和去信任的方式集体维护一个可靠数据库的技术方案。其独一无二之处在于它是一个公开的而非私人数据库。每个都能拥有它一个完整或部分完整的副本。正是由于区块链，加密货币和智能合约才得以成为现实。

区块链不仅仅是计算机科学，还同时涉及政治经济体制、社会分工机制等众多方面，本文内容与社会科学部分无关，仅仅作为非CS专业学生站在技术角度对区块链进行粗浅的研究。

本文内容涉及区块链的基本原型构建、基础工作量证明的构建、使用数据库保存区块链数据、使用命令行接口进行用户交互等。

2 区块链的搭建

2.1 搭建基本工作原型

 工作流程：

区块的数据结构如下

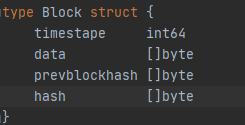


图2.1

Block中需要计算hash，之后将hash更新到Block中

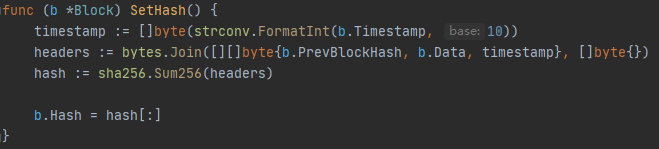


图2.2 计算hash

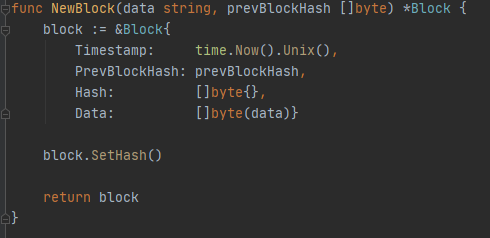


图2.3 更新hash到block中

之后，将block加入到链中。首先我们使用数组结构建立一条链，然后不断将最新的区块加入到链里。



图2.4 建立区块链

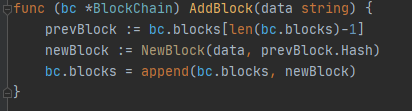


图2.5 将新的block加入到链里

至此，我们已经建立起区块链的基本架构，之后只需要在main函数中输入多个新的区块的data，使用循环结构将新的block更新到链中即可。程序运行结果如下。

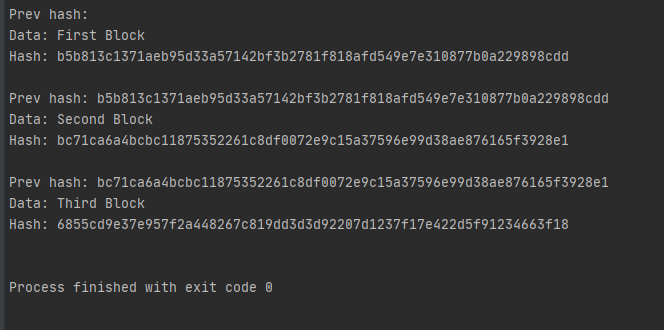


图2.6 运行结果

2.2 搭建工作量证明

工作量（work）即将一个区块加入到区块链中所需要消耗的算力。一般来说，我们通过建立起对hash的要求，譬如前六位均为0，来对区块加入提出要求。在hash满足要求后，其余各个节点还需进行提出添加区块请求的节点的工作量证明（proof of work）。为提供可变的hash，我们改进block的数据结构，加入nonce。

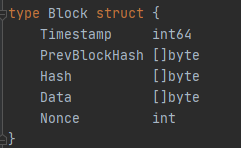


图2.7 新的block

而hash计算也改进为：

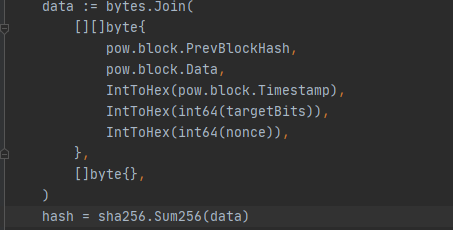


图2.8 hash的计算方法

同时，我们建立起对hash的目标要求（target），将其与block合并作为一个新结构体。

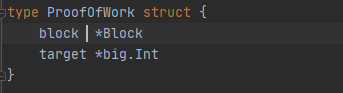


图2.9 设定target+block结构体

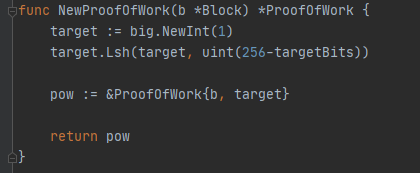


图2.10 计算target

而图中的targetBits是预先设定的难度值，target=1左移256-targetBits位。



图2.11 targetBits设定

之后只需要寻找符合要求的有效hash。

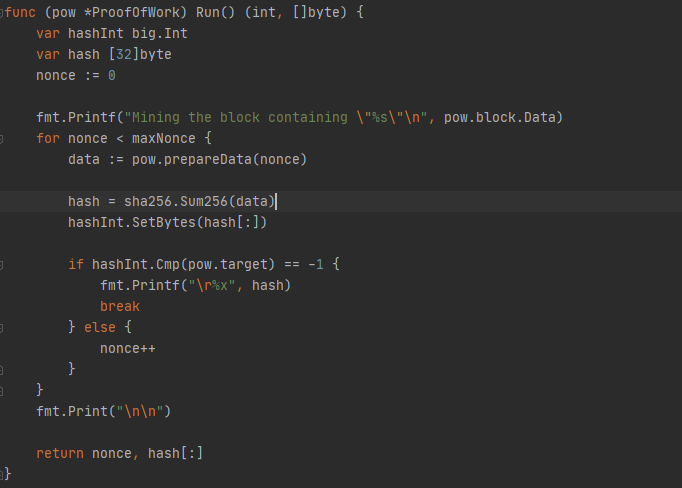


图2.12 计算hash

为了防止nonce产生溢出，我们预先设定一个maxNonce对nonce的上限进行约束。

在得到合适的hash后，我们将hash和nonce更新到最新的block中

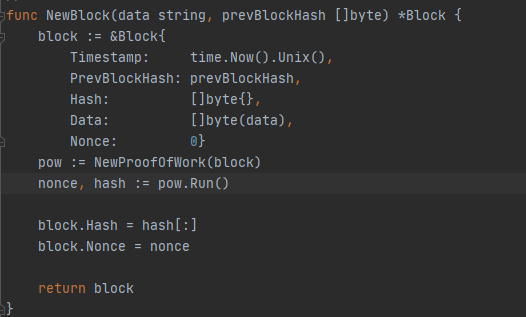


图2.13 更新block中的nonce和hash

之后，我们要验证该工作量的有效性，验证的方法是根据提交的block中的data，按照之前计算hash的方式计算一边，看hash是否符合要求。

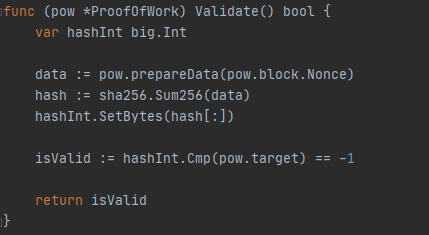


图2.14 验证工作量的有效性

之后的步骤和基本构架基本相同，判断工作量有效后，将block添加到链中（但在本程序中，其实出现工作量证明false的可能性很低，所以可以不必判断）

在运行程序时，可以发现，程序运行的时间明显增加，而所有的hash也满足我们所设定的条件——前六位均为0。

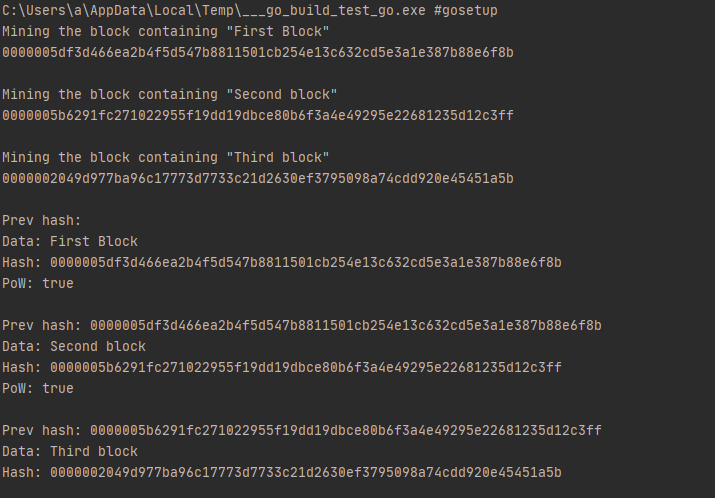


图2.15 程序运行结果

2.3 使用数据库记录并保存数据

目前的区块链已经具备了一些基本功能，但是作为一种智能账本，还不具备数据存储能力，即每次打开都是一条全新的链，所以我们需要使用数据库对区块链的信息进行保存。

本例所用的数据库为BoltDB，该数据库是go语言自带的数据库，不需要额外架构服务器。BoltDB使用键值存储的方式，键值对被存储在bucket中。在获取数据时，我们需要获得bucket以及其中相应的键。此外，由于BoltDB没有数据类型，键与值均为byte array形式，所以在存储前和提取时，我们需要进行数据结构的转化。最终，我们建立的数据库中需要的键值对为hash>block结构&&1>最后一个block的hash‘。

下面是我们的主要工作：

Ⅰ. 打开一个bolt文件

Ⅱ. 判断是否存在一个block chain

Ⅲ, 如果存在，创建block chain实例并将实例中最后block的hash与block创建键值对，1与最后block的hash创建键值对。

Ⅳ. 如果不存在，创建创世区块并存储到数据库，其余内容同上。

实际工作如下：

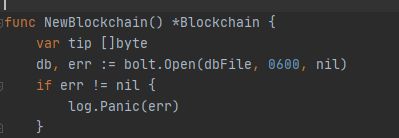


图2.16 打开一个标准文件

之后先判断是否以存储一条block chain：

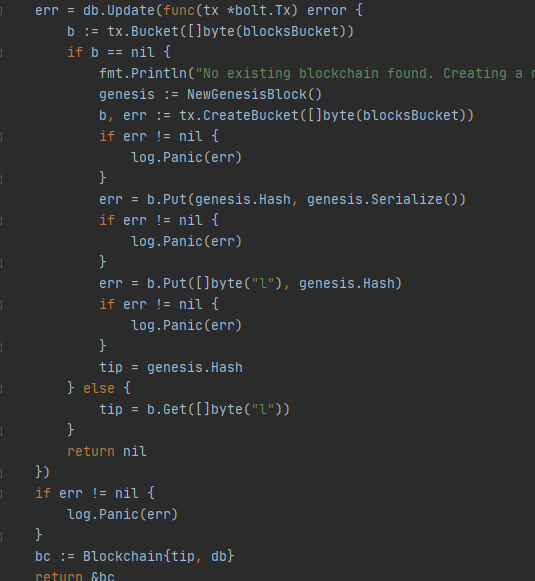


图2.17 建立block chain实例并创建键值对

此时，block chain 的结构与之前又发生了变化。我们使用byte array的形式来存储每个block的hash，使用blot数据库来存储block的信息。

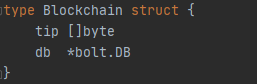


图2.18 Block chain的结构

之后我们使用与之前相同的方法创建新的block，但是prevhash的获得要需要从数据库中提取。

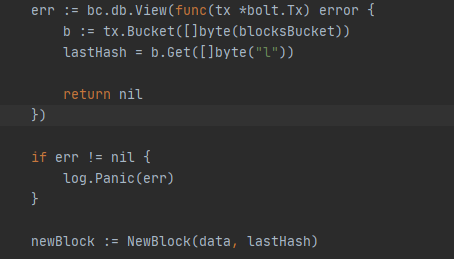


图2.19 提取最后一个block的hash来创建新的block

之后，我们将newblock保存到数据库中，创建新的键值对，并使得1仍然指向最后一个block的hash。

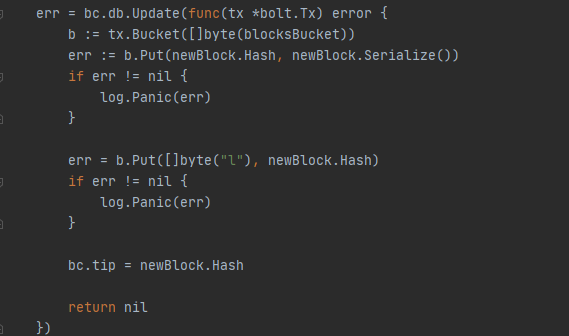


图2.20 更新数据库，将最新的block的信息存储在bolt中

至此，数据库就已具备了存储区块以及区块链信息的能力。但由于区块的存储形式由byte array变成了数据库形式，无法直接顺序打印。所以为了方便我们读取和打印。所以我们创建一个迭代器，并将迭代器与区块链连接。

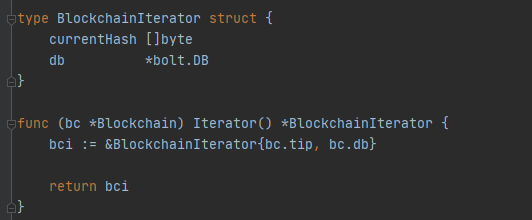


图2.21 迭代器与区块链连接

之后，我们只需要借助数据库，使用对应的hash来提取对应的键值block，然后进行数据类型转化，即可得到可打印的block。

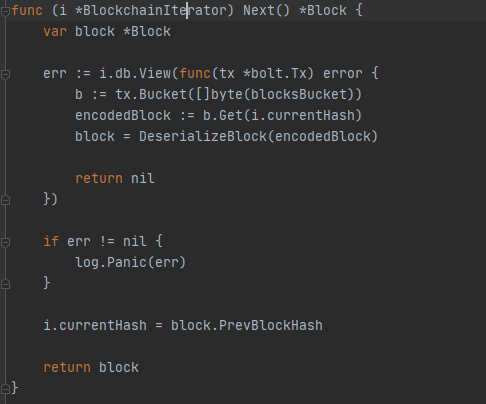


图2.22 获取下一个区块

2.4 使用命令行接口进行用户交互

目前我们已经完成了数据的存储，但是区块链还不具备与用户进行信息交互的功能，信息只能在编写程序时预先输入，所以我们需要设计交互程序，从而实现用户自由存储和查看信息。

我们采用命令行操作的形式进行交互，所有的操作都通过CLI结构体来处理。

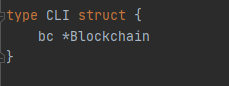


图2.23 CLI结构体

同时，我们在程序运行时给用户命令提示。

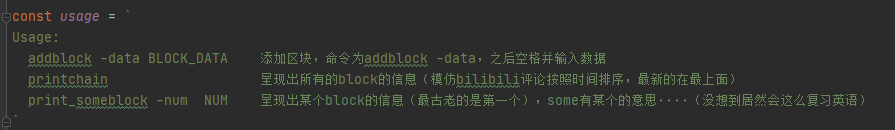


图2.24 命令

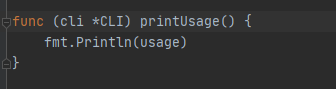


图2.25 打印命令

接下来我们设计CLI的入口函数，使用flag包解析命令行参数，创建子命令addblock、printchain、print\_someblock，将data标志添加到addblock，num添加到print\_someblock后，然后检查用户提供的命令，提取对应的子命令，然后调用相关函数即可。



图2.26 flag解析命令

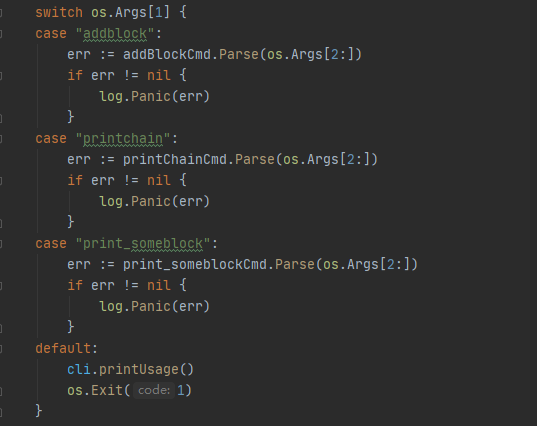


图2.27 检查命令并解析子命令

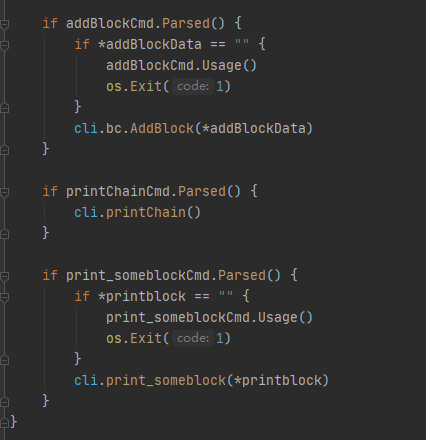


图2.28 调用相关的函数

三个函数的补充如下：

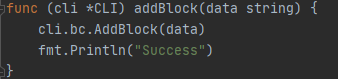


图2.29 addblock函数

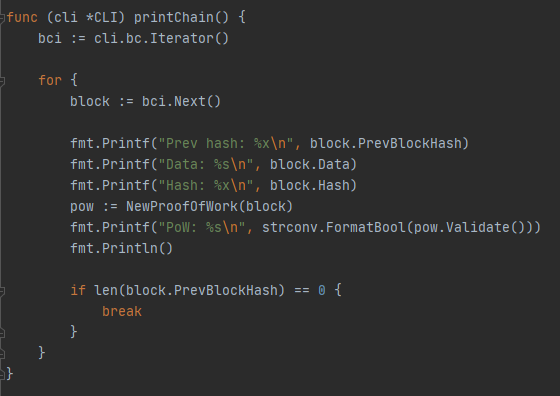
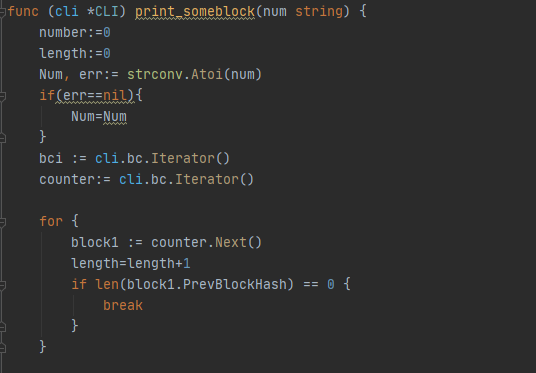


图2.30 printblock函数



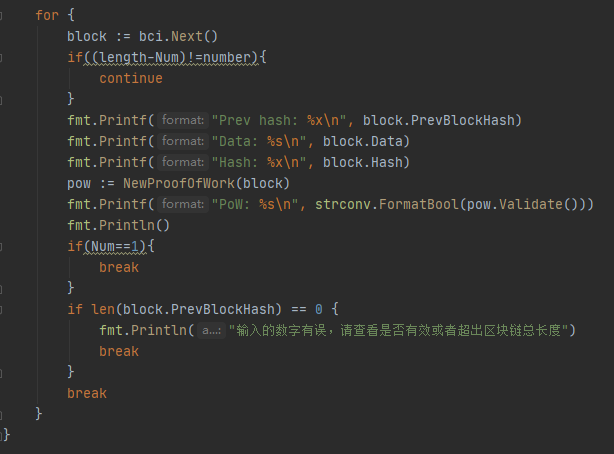


图2.31&2.32 print\_someblock函数

至此，我们的带交互功能的区块链便已经建设完成。只需要保存为.go文件，然后使用命令行运行即可，示例如下。

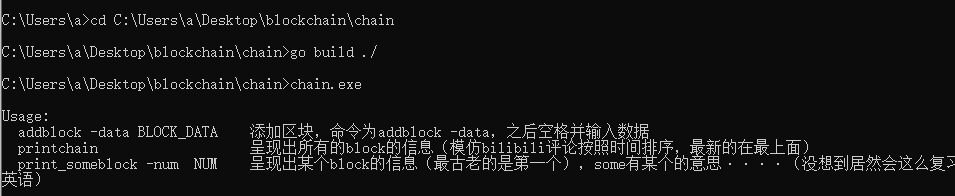


图2.33 文件生成&命令展示

使用命令行进入相应的文件夹，使用go build命令生成.exe文件，之后根据提示界面，使用相应的命令即可。命令演示如下：

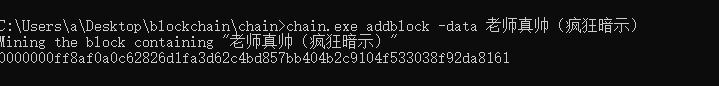


图2.34 addblock命令展示

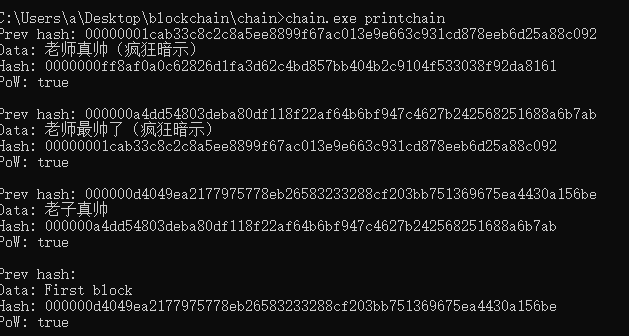


图2.35 printchain命令展示

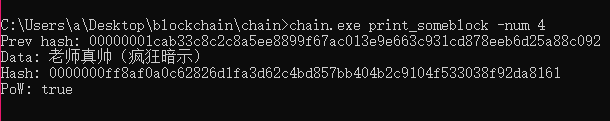


图2.36 print\_someblock 命令展示

至此，本文所涉及的区块链构建已全部完成，文字部分参考了区块链白皮书。同时，感谢老师一学期的辛苦授课。