

Rust_chain B1/B2 产品说明书

一、产品概述

Rust_chain B1 是一个基于比特币原始白皮书设计的简单区块链实现项目，使用 Rust 语言进行开发。该项目旨在展示区块链的核心概念和基本工作原理，包括区块的创建与验证、交易的处理、工作量证明机制等。

二、产品架构

1. 项目文件结构

项目主要由以下几个部分组成：

Cargo.toml 和 Cargo.lock：Rust 项目的配置文件和依赖锁定文件，用于管理项目的依赖库和版本信息。

src/ 目录：包含项目的源代码文件，各个模块各司其职，共同实现区块链的功能。

2. 核心模块

- **block.rs:**
 - 定义了 Block 结构体，用于表示区块链中的区块。
 - 包含创建新区块、挖矿（PoW）和计算区块哈希值等方法。通过这些方法，实现了区块的基本操作和 PoW 机制，确保了区块链的安全性和一致性。
- **blockchain.rs:**
 - 定义了 Blockchain 结构体，用于管理整个区块链的状态。
 - 包含从文件加载区块链数据、添加新的区块、验证区块链的完整性以及保存区块链到文件等方法。这些方法保证了区块链数据的持久化和一致性。
- **transaction.rs:**
 - 定义了 Transaction 结构体，用于表示区块链中的交易。
 - 实现了交易的创建、签名和验证等功能，确保了交易的合法性和安全性。
- **wallet.rs:**
 - 负责管理用户的钱包信息，包括生成密钥对和签名交易等功能。用户可以使用钱包进行交易操作，确保交易的真实性和不可篡改性。
- **merkle_tree.rs:**

- 实现了默克尔树的构建和哈希计算。默克尔树用于验证交易的完整性，提高了区块链的效率和安全性。
- **node.rs:**
 - 定义了 Node 结构体，用于管理节点的信息，包括节点的地址、区块链和对等节点列表。
 - 可能实现了节点之间的通信功能，如同步区块链数据、广播交易信息等，以支持分布式的区块链网络。
- **cli.rs:**
 - 负责处理命令行界面的输入和输出，通过 clap 库解析用户输入的命令行参数。
 - 提供了创建钱包、发起交易、挖矿新区块和验证区块链等功能，方便用户与区块链进行交互。

三、主要功能

1. 钱包管理

- **创建钱包:** 用户可以通过命令行创建新的钱包，每个钱包都有唯一的地址和对应的密钥对。
- **钱包签名:** 钱包可以对交易进行签名，确保交易的真实性和不可篡改性。

2. 交易处理

- **创建交易:** 用户可以发起一笔交易，指定发送者、接收者和交易金额。
- **交易签名:** 交易在创建后会使用发送者的钱包进行签名，确保交易的合法性。
- **交易验证:** 系统会验证交易的签名，只有签名合法的交易才能被添加到区块链中。

3. 区块管理

- **挖矿新区块:** 矿工可以通过命令行发起挖矿操作，系统会使用工作量证明机制 (PoW) 来挖掘新的区块。
- **区块链验证:** 用户可以验证区块链的完整性，确保区块链中的所有区块和交易都是合法的。

4. 数据持久化

- **加载区块链：**系统可以从文件中加载区块链数据，方便用户恢复之前的区块链状态。
- **保存区块链：**系统会定期将区块链数据保存到文件中，确保数据的持久性。

四、使用方法

用户流程

1. 创建钱包：

用户生成新的钱包，获取钱包地址。

2. 发起交易：

用户输入发送者地址、接收者地址和金额，发起交易。

3. 挖矿确认：

矿工将未确认的交易打包到新区块中。

4. 验证链：

用户验证区块链的完整性。

1. 克隆项目

```
git clone <项目仓库地址>
```

2. 编译项目

```
cargo build
```

3. 运行项目

```
cargo run -- <命令>
```

```
Block mined: 0000045834da28d1979f5d2f64a956d0ec402c430fb963831433cff13dc87302
Syncing blockchain with peer: 127.0.0.1:8081
Blockchain on Node 1:
Block {
  index: 0,
  timestamp: 1741462225,
  transactions: [],
  previous_hash: "0",
  hash: "42c8dccbf6803b5249fb34687828bf413cdb8d8fcefab9ace18cbe1ec48efca2",
  nonce: 0,
  merkle_root: "2e1cfa82b035c26cbbbdade632cea070514eb8b773f616aaef668e2f0be8f10d",
}
Block {
  index: 1,
  timestamp: 1741462225,
  transactions: [
    Transaction {
      sender: "f0dad51c1734df2ca2550f0c468360f006c8b50d3163613787f41e730bf5ff54",
      receiver: "e94c79f05e48038744ee0545baf09ca256ea2d44f324510b04e2f31514d03326",
      amount: 100,
      signature: "20834193f20ab4a1bf1191018834705cd5bab415798cb39b950fa193b533221779031139bafa09767677c4f31b1abbf63f65e619e8d54ace6af28fb7468601",
    },
  ],
  previous_hash: "42c8dccbf6803b5249fb34687828bf413cdb8d8fcefab9ace18cbe1ec48efca2",
  hash: "0000045834da28d1979f5d2f64a956d0ec402c430fb963831433cff13dc87302",
  nonce: 38548,
  merkle_root: "71c0977f9df79cbdc68fe1a53d1a5867ee792e238fcede24c48d9fe53d623268",
}
```

4. 命令行参数

- 创建钱包:

```
cargo run -- create-wallet
```

- 发起交易:

```
cargo run -- add-transaction --sender <SENDER_ADDRESS> --receiver  
<RECEIVER_ADDRESS> --amount 100
```

- 挖矿新区块:


```
cargo run -- mine-block --miner <MINER_ADDRESS>
```

- 验证区块链:

```
cargo run -- validate-chain
```

```
1  mod block;  
2  mod blockchain;  
3  mod transaction;  
4  mod node;  
5  mod merkle_tree;  
6  mod cli;  
7  mod wallet;  
8  mod smart_contract;  
9  mod privacy;  
10  
11 use cli::Cli;  
12 use clap::Parser;  
13  
14 fn main() {  
15     let cli = Cli::parse();  
16     cli.run();  
17 }  
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1

 bash - B1

Running `target/debug/blockchain-rust`

Usage: blockchain-rust <COMMAND>

Commands:

create-wallet	创建一个新钱包
add-transaction	发起一笔交易
mine-block	挖矿新区块
validate-chain	验证区块链

五、注意事项

- 本项目是一个简单的区块链实现，仅用于学习和演示目的，不适合用于生产环境。
- 在使用过程中，请确保项目的依赖库已经正确安装，并且网络连接正常。

Rust_chain B2/B3 产品说明书

注：B3 部分功能为教学模板（如隐私交易验证），开发者需根据业务需求完善核心算法和性能优化

运行示例

1. 创建钱包

```
BASH
cargo run -- create-wallet
```

输出：

```
PLAINTEXT
New wallet created!Wallet address: 8a7c3f...
```

2. 发起交易

```
BASH
cargo run -- add-transaction --sender 8a7c3f... --receiver bob --amount 100
```

输出：

```
PLAINTEXT
Transaction created: Transaction { ... }
```

3. 挖矿确认

```
BASH
cargo run -- mine-block --miner miner
```

输出：

```
PLAINTEXT
New block mined by miner: minerLatest block: Block { ... }
```

4. 验证链

```
BASH
cargo run -- validate-chain
```

输出:

```
PLAINTEXT
Blockchain validity: true
```

5. 部署智能合约

```
BASH
cargo run -- deploy-contract --contract-id my_contract --code "contract_code"
```

输出:

```
PLAINTEXT
Contract deployed: my_contract
```

6. 执行智能合约

```
BASH
cargo run -- execute-contract --contract-id my_contract --method set --args
'["key", "value"]'
```

输出:

```
PLAINTEXT
Contract execution result: Value set successfully
BASH
cargo run -- execute-contract --contract-id my_contract --method get --args
'["key"]'
```

输出:

```
PLAINTEXT
Contract execution result: value
```

7. 创建隐私交易

```
BASH
cargo run -- create-privacy-transaction --amount 100
```

输出:

```
PLAINTEXT
Privacy transaction created and added to blockchain
```