## CreateBlockChain

```
//单纯的创建区块链db，同时写入创世快
func CreateBlockChain(address string) bool {
    if isFileExist(blockChainFileName) {
        fmt.Println("数据库已经存在，不需要重复创建!")
        return true
    }

    //创建区块链时，向里面写入一个创世快
    db, err := bolt.Open(blockChainFileName, 0600, nil)
    if err != nil {
        fmt.Println("创建区块链失败, err:", err)
        return false
    }

    _ = db.Update(func(tx *bolt.Tx) error {
        b := tx.Bucket([]byte(blockBucket))

        if b == nil {
            //创建bucket
            b, err = tx.CreateBucket([]byte(blockBucket))
            if err != nil {
                fmt.Println("创建bucket失败, err:", err)
                return err
            }

            //写入创世块
            //创建一个挖矿交易，里面写入创世语
            coinbaseTx := NewCoinbaseTx(address, genesisInfo)
            genesisBlock := NewBlock([]*Transaction{coinbaseTx}, nil)

            //第一次：写入区块的数据
            _ = b.Put(genesisBlock.Hash, genesisBlock.Serialize() /*区块转换成字节
流*/)
            //第二次：写入最后一个区块哈希
            _ = b.Put([]byte(lastBlockHashKey), genesisBlock.Hash)
        }

        return nil
    })
    return true
}
```

## NewBlockChain

```
//打开区块链，返回blockchain实例
func NewBlockChain() *BlockChain {
    if !isFileExist(blockChainFileName) {
        fmt.Println("请先创建区块链数据库，执行createDb命令")
```

```
        return nil
    }

    var lastHash []byte

    //打开区块db
    db, err := bolt.Open(blockChainFileName, 0600, nil)
    if err != nil {
        fmt.Println("打开失败区块链失败, err:", err)
        return nil
    }

    err = db.View(func(tx *bolt.Tx) error {
        b := tx.Bucket([]byte(blockBucket))
        if b == nil {
            return errors.New("NewBlockChain 时bucket不应该为nil")
        }

        //bucket已经存在, 直接读取最后一区块的哈希值
        lastHash = b.Get([]byte(lastBlockHashKey))
        fmt.Printf("lastHash : %x\n", lastHash)

        return nil
    })

    if err != nil {
        fmt.Println("NewBlockChain err:", err)
        return nil
    }

    return &BlockChain{db: db, tail: lastHash}
}
```

# 改写CLI

```
//定义一个CLI结构
type CLI struct {
    //bc *BlockChain
}
```

# 改写main

```
package main

func main() {
    cli := CLI{}
    cli.Run()
}
```

# 增加创建命令

```
func (cli *CLI) createDb(address string) {
    if !CreateBlockChain(address) {
        fmt.Println("创建区块链失败!")
    }
    fmt.Println("创建区块链成功!")
}
```

在所有使用blockchain实例的函数前添加如下代码：

```
    bc := NewBlockChain()

    if bc == nil {
        return
    }

    defer bc.db.Close()
```