分布式系统课程大作业

Client-Server架构的集中式

分布式键值存储系统

**21307376 曹永皓 系统结构班**

一、开发环境

操作系统：Windows10；

编程语言：Pyhton 3.8.10

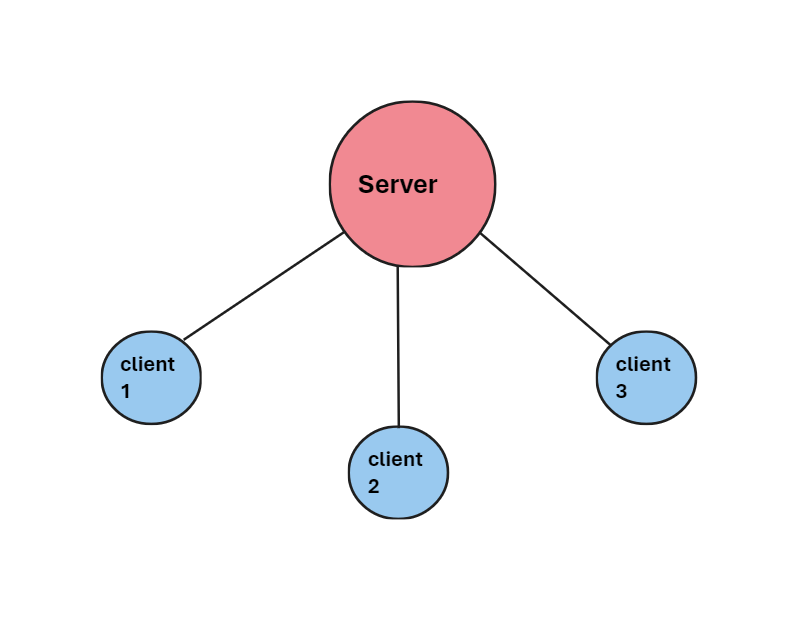
IDE：VSCode

二、项目描述

本项目实现了一个Client-Server架构的集中式的分布式键值存储系统。

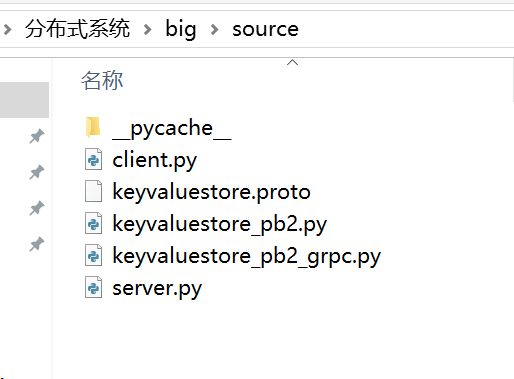
该系统的运作方式是：一个服务器保持运行状态，监听来自多个客户端的关于键值存储的操作，包含“PUT”“GET”“DEL”，即“添加/修改键值对”，“读取键值对”，“删除键值对”三种基本操作。客户端与服务器端的通信方式为RPC。并且该系统通过简单的锁机制，实现了当其中一个客户端节点对某一特定键进行访问操作时，其余客户端节点对该键值对的访问操作将被阻塞（中断），同时一并保证了面向客户的单调写一致性。

**测试时的系统架构图：**



**测试方式：**

项目的文件结构如下：

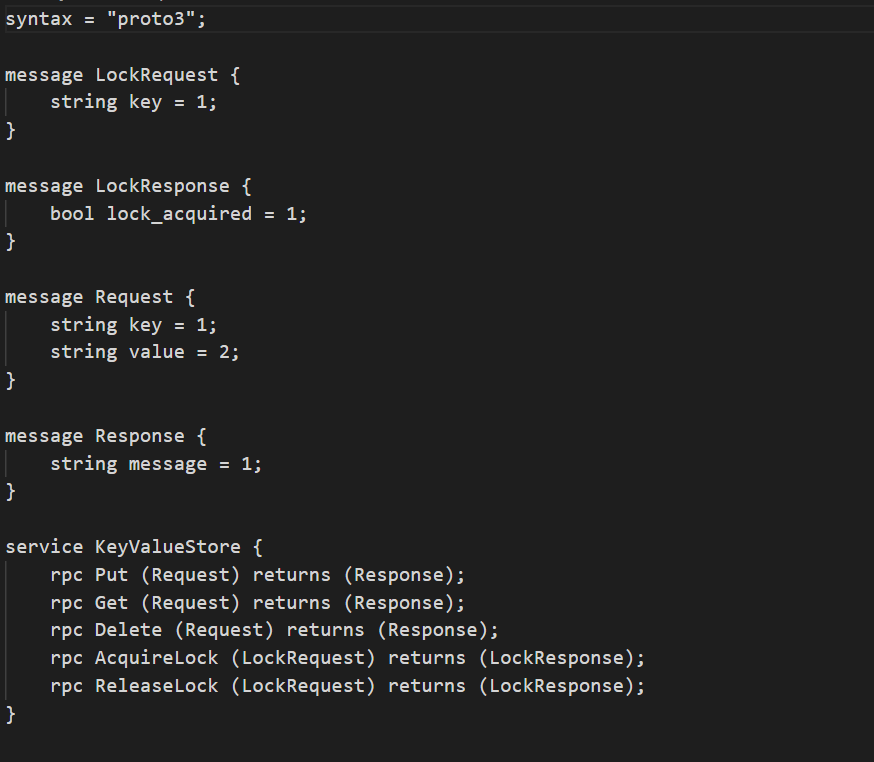


进行测试时由于只有一台设备，故采用开启多个终端窗口的方式模拟多个节点。

首先开启一个终端，在项目目录下运行server.py文件，以运行服务器节点；随后额外开启3个终端，分别运行client.py文件，以运行客户端节点。在3个客户端上进行关于键值存储的各项操作。

三、项目实现

客户端和服务器端节点之间的通信按要求需使用RPC机制，这里使用由google提供的gRPC。首先编写一份.proto文件，定义gRPC的消息和服务，如图所示：



再在终端执行指令：

python -m grpc\_tools.protoc --python\_out=. --grpc\_python\_out=. -I. FileServer.proto

以编译生成keyvaluestore\_pb2.py和keyvaluestore\_pb2\_grpc.py两个文件，用于后续的RPC通信实现。（这里已经事先安装好了实现gRPC的所需包以及编译proto文件的工具）

分别编写服务器端和客户端的代码。

**服务器端：**

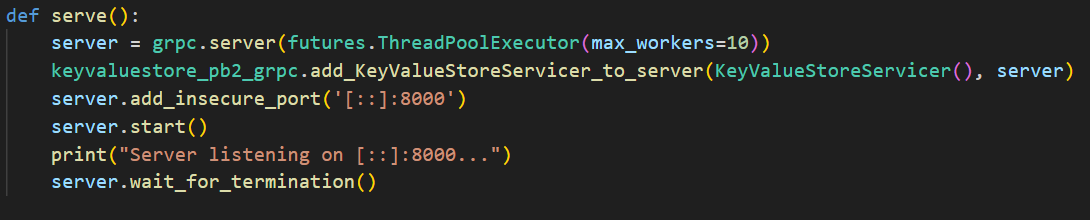
定义一个键值存储服务器类，拥有一个字典成员datastore，用于存储键值对；一个字典成员locks，作为缓冲区用于记录客户端访问指定键时所提供的锁。

根据所编写的keyvaluestore,proto文件中所定义的服务名称，定义成员函数Put(),Get(),Del()。这里以函数Put()为例，解释在服务器端执行键值添加或修改的过程。



客户端发送来的请求消息包含了指定的键，以及希望写入的对应的值。服务器检查锁缓冲区中是否含有该键对应的锁，若含有锁，则允许访问，将datastore中的指定键对应的值进行添加或修改，完成写入操作，并向客户端返回写入成功的消息；若不含有锁，则不允许访问，向客户端返回写入失败的消息。

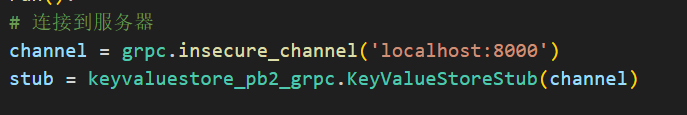
Get(),Del()函数也以类似的方式进行实现，具体可查看附带的源码。



函数serve()定义了启动服务器的方法，架设通过gRPC进行通信的服务器，将端口设为8000

**客户端：**

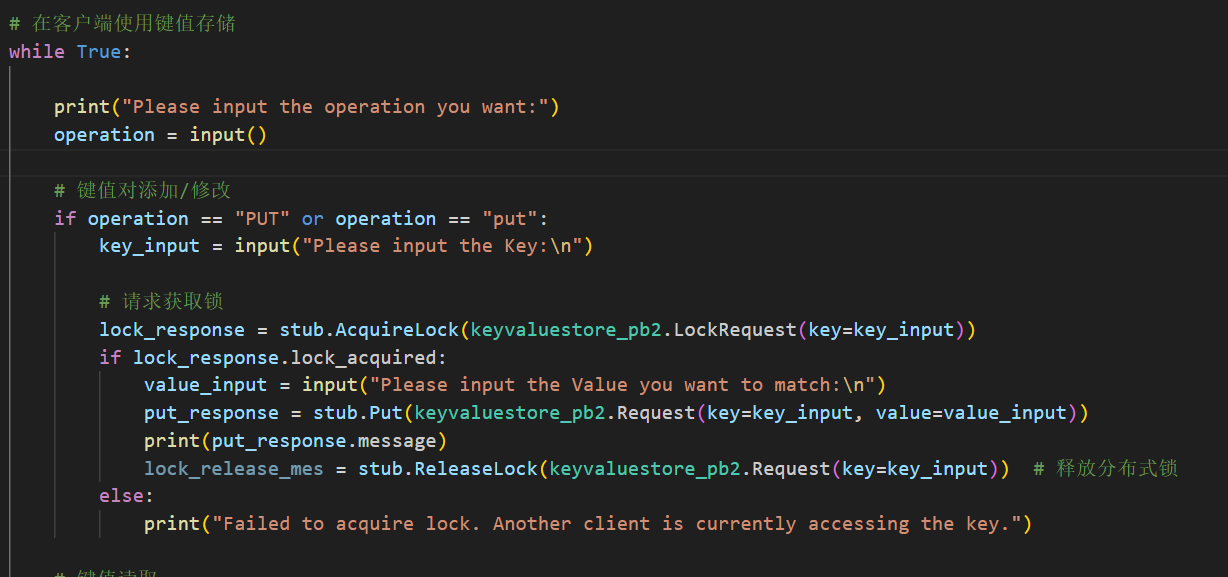
首先连接到服务器



客户端启动后，会进入一个while True的死循环中，直到用户输入退出指令以结束客户端的运行。

客户端启动后，可以输入的指令有put、get、del、quit。

这里同样以PUT操作为例，展示在客户端的键值存储操作实现

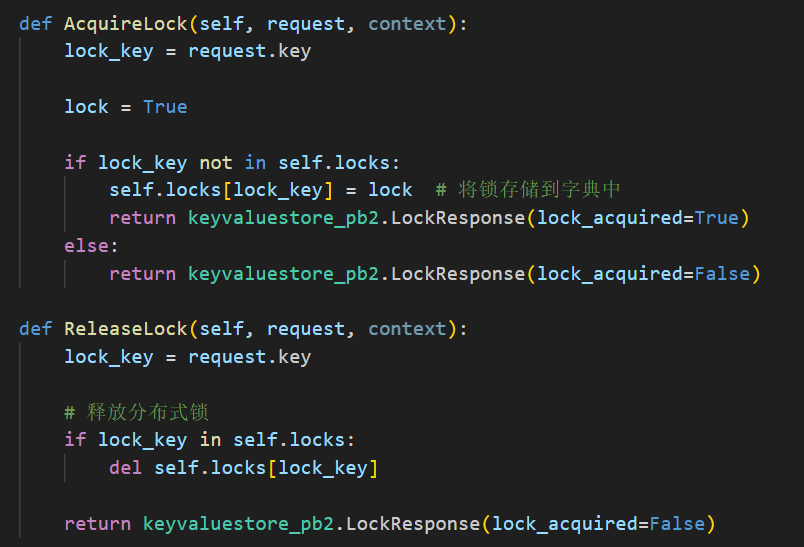


首先向服务器端发送申请锁的请求，根据从stub获得的发送回来的消息判断锁是否已成功申请。若已经成功申请锁，则输入希望写入的值，提交写入并获得从服务器端返回的写入是否成功的消息；若未能成功申请锁，说明有其他节点（进程）正在访问当前的键，输出占用消息并中断操作。

GET，PUT操作也以类似的方式实现，重点在输入希望访问的键后向服务器端申请对应的锁。

**锁机制的实现：**

在服务器端定义有两个函数AcquireLock()和ReleaseLock()，对应.proto文件中定义的服务名。两函数的实现如下：



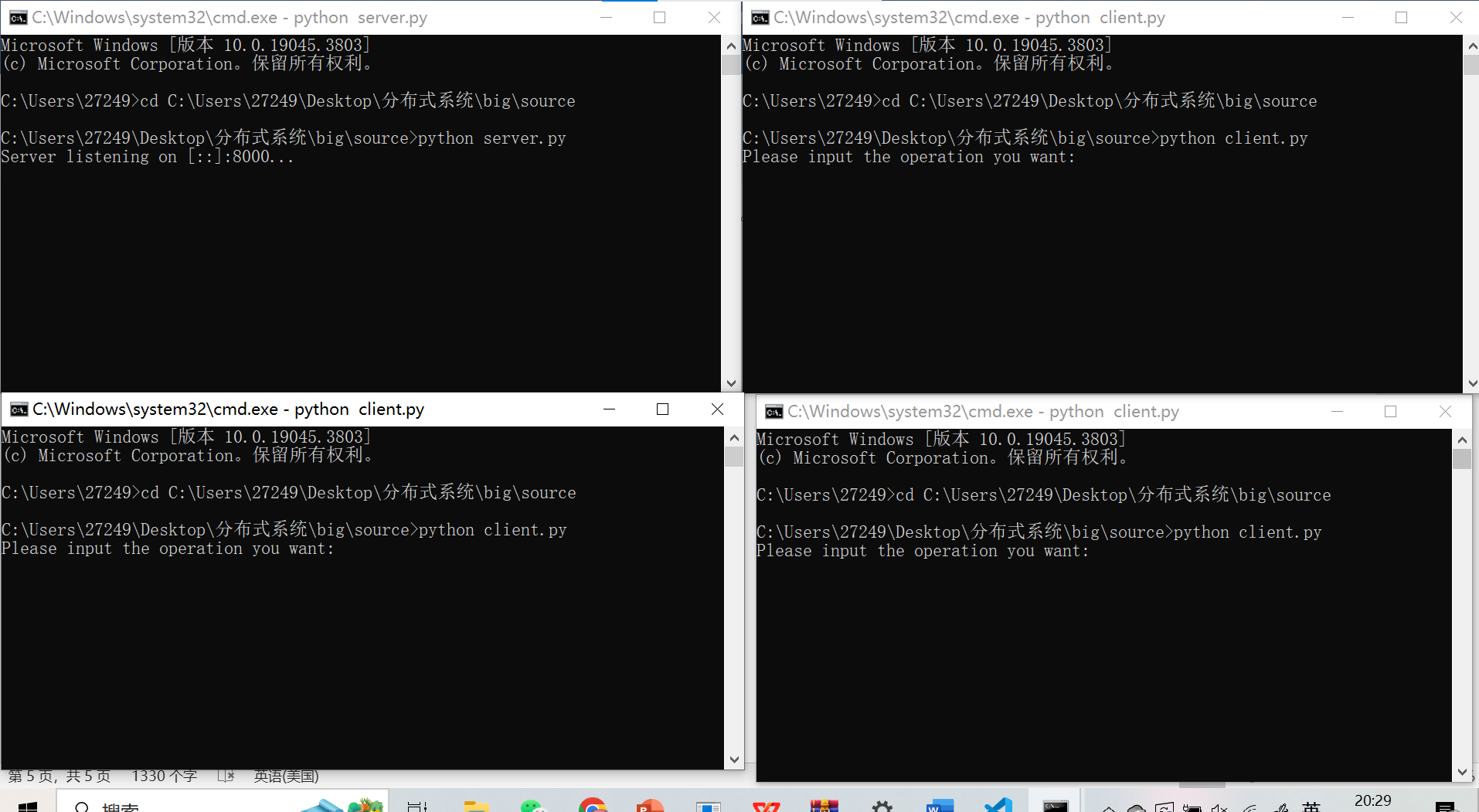
客户端申请锁时，根据占用情况决定是否将锁缓冲区中的对应位置加锁，并返回申请结果消息；客户端申请释放锁时，将对应键位置的锁去除。

**一致性实现：**

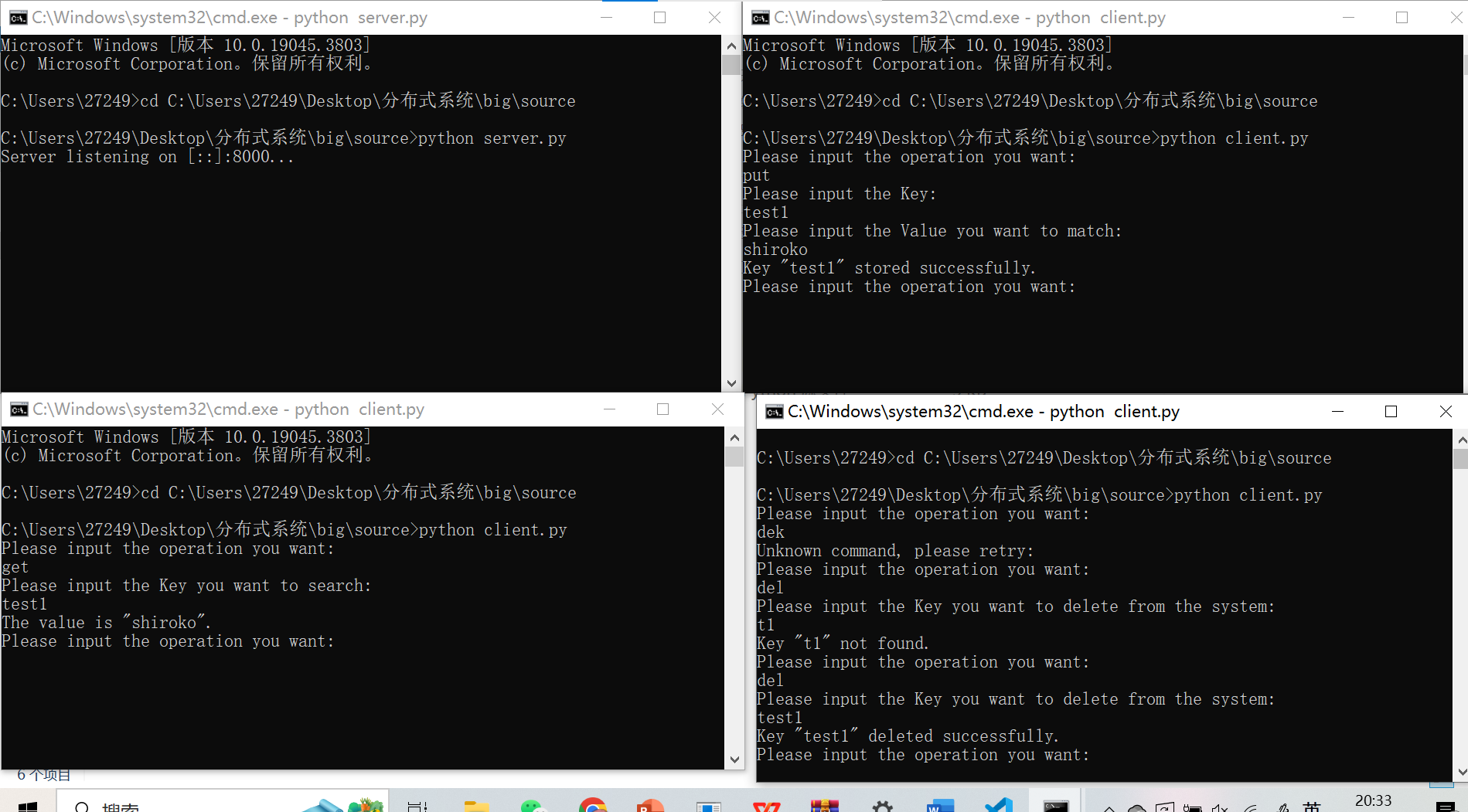
在锁机制下，多个客户不能同时访问同一个键。面向客户的单调写一致性被自然地保证了。

四、操作测试

启动终端，运行各自对应的python文件，其中左上为服务器端，右上、左下、右下分别为客户端1、2、3

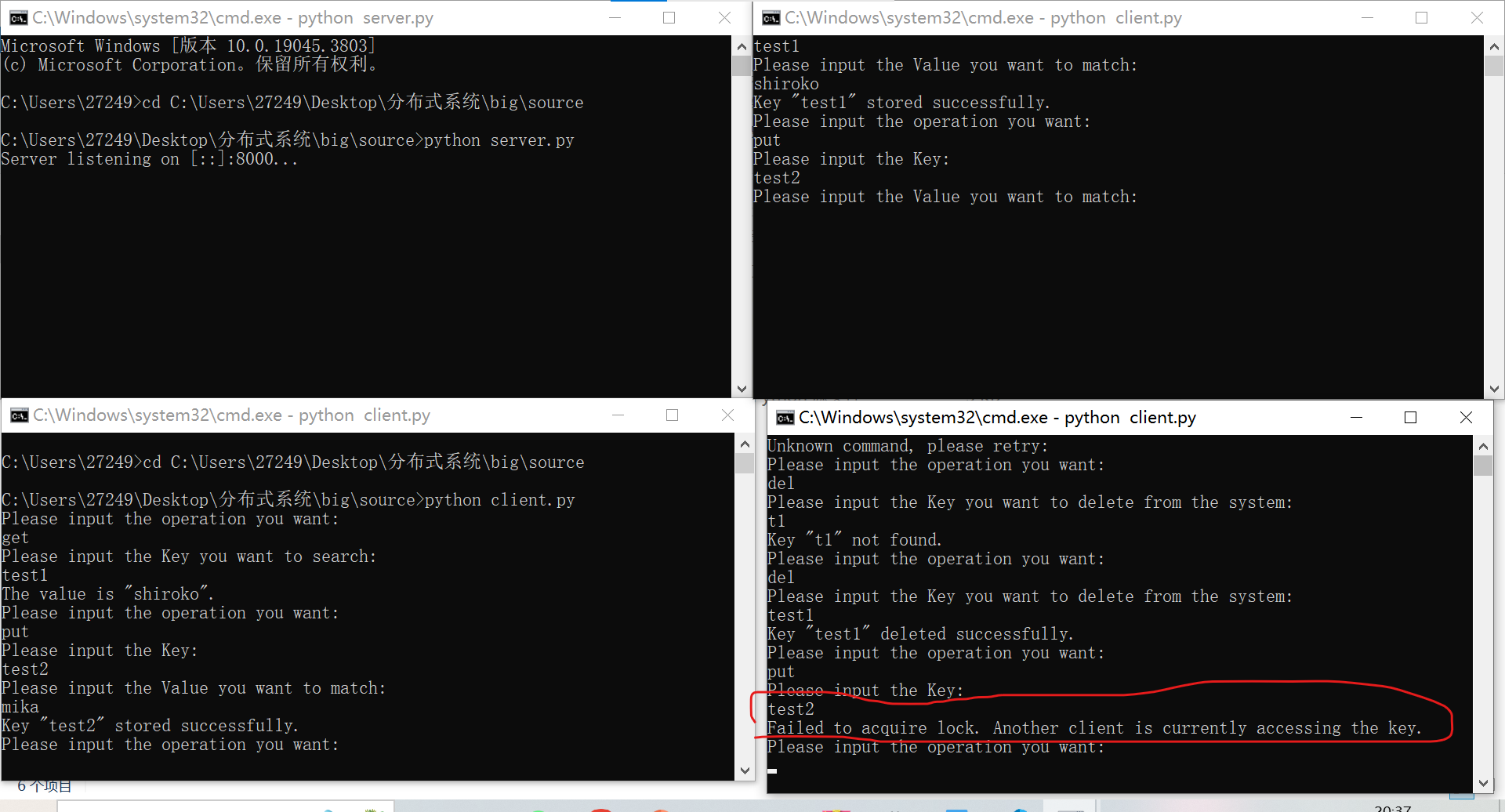


在不同的客户端，执行PUT，GET，DEL操作



所有操作均成功执行。

下面演示锁机制：两个客户端节点（进程）同时访问相同的键



可以看到在客户端1进行PUT操作时，客户端3想要访问相同的键，但过程被中断。