实验报告

一、实验结果

本地测试所有测试点，结果均正确，提交到ics.men.ci后得分71



二、使用的相关api

**1. pmemobj\_root(pop, sizeof(T));**

获得存储在pool中数据结构的入口

**2. pmemobj\_persist(pop, mem\_address, size\_t);**

持久化size\_t大小的内存空间，其中mem\_address为实际内存地址

**3. pmemobj\_memcpy\_persist(pop, mem\_address, void \*, size\_t);**

持久化内存拷贝操作

**4. pmemobj\_direct(PMEMoid data)**

获得data对象的实际内存地址

**5.事务功能：**

TX\_BEGIN(pop) {

/\* TX\_STAGE\_WORK: 事务想要完成的功能区块\*/

} TX\_ONCOMMIT {

/\* TX\_STAGE\_ONCOMMIT: 事务提交时需要做的额外工作\*/

} TX\_ONABORT {

/\* TX\_STAGE\_ONABORT: 事务回滚时需要做的恢复工作\*/

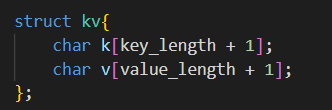
} TX\_FINALLY {

/\* TX\_STAGE\_FINALLY: 事务commit或者abort都会执行的区块\*/

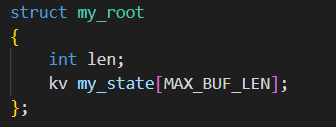
} TX\_END

三、实现思路及代码说明

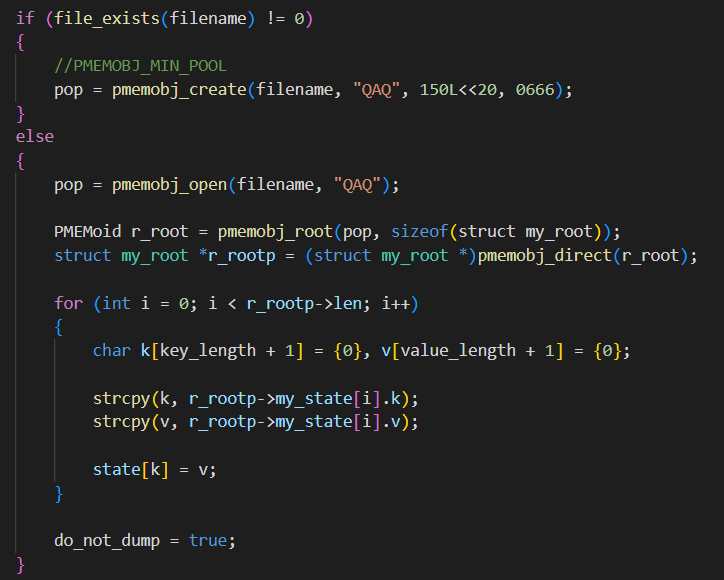
设置结构体类型kv用于存储一对key-value



修改持久化内存中的数据结构my\_root，其包含两个成员变量，my\_state是kv数组，用于存储key-value对。len既表示kv数组中键值对的个数，又可以作为插入新的键值对时需要的偏移量



修改纯读测试时的代码，使其将键值对从持久化内存读入到state中，以进行GET和NEXT操作。使用pmemobj\_root和pmemobj\_direct从持久化内存中得到my\_root数据结构的指针，将my\_state数组中的键值对依次读取后添加到state中即可。



而在进行SET操作时，则需要将q.key和q.value写入持久化内存。先用kv类型的临时变量qdata存下q中的键值对，然后用pmemobj\_persist更新len（加1），并使用pmemobj\_memcpy\_persist来实现my\_state数组中新的键值对的添加

此外还用TX\_BEGIN和TX\_END实现set操作的原子性。

