NVM Lab

由于这部分实验上课时摸鱼成分较高，所以完成时在很大程度上是依赖他人指导完成的

主要用到的函数

```
void pmemobj_persist(PMEMobjpool *pop, const void *addr,
    size_t len);
void *pmemobj_memcpy_persist(PMEMobjpool *pop, void *dest,
    const void *src, size_t len);
```

pmemobj_persist（） 强制范围 [*addr*， *addr+len*） 中的任何更改持久存储在持久内存中

pmemobj_memcpy_persist() is an alias for **pmemobj_memcpy**() with flags equal to 0

以上两个函数是SET里将.in文件写入到pool池中

思路

基于助教下发文件主要有两处改动

第一处改动

```
case Query::SET:
        {
            node tmp;
            strcpy(tmp.key,q.key.c_str());
            strcpy(tmp.value,q.value.c_str());
            pmemobj_memcpy_persist(pop, rootp->buf + rootp->len,
&tmp, sizeof(tmp));
            rootp->len++;
            pmemobj_persist(pop, &rootp->len, sizeof(rootp-
>len));
            break;
        }
```

原文件SET时并未将内容刷回至nvm中，基于PPT上代码就行

```
void set_name(const char *my_name) {
root->length = strlen(my_name);
pmemobj_persist(&root->length, sizeof (root->length));
pmemobj_memcpy_persist(root->name, my_name, root->length);
}
```

要注意的是，刷回的内容既有tmp也有$rootp->len$，$rootp->len$要先加后刷回，不然最终得到的n会少1；还有由于使用了.c_str()，所以需要注意node里定义的key和value要在最大长度上+1（因为补充了\0）

第二处改动

```
PMEMoid root = pmemobj_root(pop, sizeof(struct my_root));
struct my_root *rootp = (struct my_root *)pmemobj_direct(root);
static int I = 0;
for (; I < rootp->len; I++)
    {
        char* k = rootp->buf[I].key;
        char* v = rootp->buf[I].value;
        state[k] = v;
    }
```

原文件是直接open池子，然后利用fread读入了池子里的key和value，然后此处直接利用PEMoid直接把池子内的内容存到map（内存）里，同时由于防止中断导致频繁进行这个工作，将I设置为static

同时经同学指导，原来利用fread是因为我使用这种读法会导致数组越界，其实rootp->buf[i]的内容以已经远远多于MAX_BUF_LEN，但是由于lib库优秀的自动对齐和指针字节运算size的优势，这不会导致RE

然后与低分代码对比发现，这种方法更适用于本次实验。因为本次实验在.out文件里不存在追加写，只会在.in里完成SET工作，因此这就允许我们在一旦开始调用.out文件时，就将.in里的内容搬到map里，后续GET和NEXT时已经非常优秀了，用下发的代码就行

吐槽

我第一次测试拿到了75，后来加了些可有可无的优化到63，然后重复测试第一次代码在63-69之间波动，就拿过一次75，不知道最终评测会不会取个平均值啥的