# 《操作系统原理》实验报告

## ------实验二 Threads and Locking

年级：2022 级　班级：智算大类十二班 学号：3523403837 姓名：邢俊杰

### 一、实验题目及要求

(1)题目：使用线程和锁来探索并行编程，并且使用哈希表模拟进行

(2)要求：修改代码以确保 "get" 操作能够并行运行并保持正确性，以及使一些 "put" 操作能够并行运行并保持正确性。提示是考虑哪些锁在这个应用程序中是必要的，以确保正确性。

### 二、设计说明

(1)比较使用多个线程和仅一个线程的性能，查看是否使用多个线程可以提高性能。

(2)如果在运行时出现键丢失的情况，请确定可能导致 2 个线程出现键丢失的事件序列，并修改代码以确保键不会丢失。

(3)测试您的代码以确保在单线程和多线程情况下都能正确运行，并尽量提高性能。

### 三、编译、运行、测试说明

1.准备源代码：将您的代码保存在一个以 .c 为扩展名的源代码文件中，比如 ph.c。

2.打开终端：在 Linux 系统上打开终端。

3.使用编译器编译代码：在计算机上编译 "ph.c"，可以使用以下命令：

```
$ gcc -g -O2 ph.c -pthread
```

4.运行可执行文件：运行程序，指定要使用的线程数，例如：

```
$ ./a.out 2
```

这里的 "2" 指定了执行对哈希表执行 "put" 和 "get" 操作的线程数。

### 四、实验过程

（1）声明锁:声明一个数组 locks，用于存储互斥锁。每个桶都有一个对应的互斥锁，所以数组的大小为 NBUCKET，这是哈希表中桶的数量。

（2）“put”操作:首先在函数中通过给定的键（Key）使用取模运算来确定桶的索引，然后获取桶 i 对应的互斥锁，确保当前线程能够独占地访问这个桶。如果其他线程已经获得了这个互斥锁，那么当前线程将被阻塞，直到互斥锁可用。后通过 Insert 方法插入键值到哈希表中的桶 i，最后，释放桶 i 对应的互斥锁，允许其他线程访问这个桶。

（3）“get”操作：同理，首先在函数中通过给定的键（Key）使用取模运算来确定桶的索引，然后获取桶 i 对应的互斥锁，确保当前线程能够独占地访问这个桶。如果其他线程已经获得了这个互斥锁，那么当前线程将被阻塞，直到互斥锁可用。后通过 for 循环用于遍历桶 i 中的链表，以查找与给定键匹配的条目。在循环中，如果找到了与给定键匹配的条目，就

就跳出循环。

这种使用互斥锁的方式确保了每次只有一个线程可以同时访问哈希表中的一个桶，从而避免了并发访问时的竞态条件，确保了数据的一致性和正确性。这对于多线程的并发程序非常重要。

```
//YOUR CODE START
pthread_mutex_t locks[NBUCKET];//声明锁
static
void put(int key, int value)
{
  int i = key % NBUCKET;
  pthread_mutex_lock(&locks[i]);//获取锁
  insert(key, value, &table[i], table[i]);
  pthread_mutex_unlock(&locks[i]);//释放锁
}

static struct entry*
get(int key)
{
  struct entry *e = 0;
  int i = key % NBUCKET;
  pthread_mutex_lock(&locks[i]);//获取锁
  for (e = table[key % NBUCKET]; e != 0; e = e->next) {
    if (e->key == key) break;
  }
  pthread_mutex_unlock(&locks[i]);//释放锁
  return e;
}
//END
```

然后通过$ ./a.out 2 、 $ ./a.out 1 两条线程测试

```
[xingjunjie@localhost 作业2 Threads and Locking]$ ./a.out
./a.out: ./a.out nthread
[xingjunjie@localhost 作业2 Threads and Locking]$ ls
a.out  ph.c  实验2作业.pdf
[xingjunjie@localhost 作业2 Threads and Locking]$ ./a.out 2
0: put time = 0.005526
1: put time = 0.008627
0: get time = 19.550653
0: 0 keys missing
1: get time = 19.848867
1: 0 keys missing
completion time = 19.870484
[xingjunjie@localhost 作业2 Threads and Locking]$ ./a.out 1
0: put time = 0.010759
0: get time = 19.281871
0: 0 keys missing
completion time = 19.293086
[xingjunjie@localhost 作业2 Threads and Locking]$
```

经过测试，代码可以在单线程和多线程情况下都能正确运行，并且在 2 个线程的情况下没有键丢失

**五、实验总结**

通过本次实验，我知道了在多线程环境中，确保每次只有一个线程可以访问共享的数据结构是关键，并且，使用互斥锁时，要考虑锁的粒度，确保在每个互斥锁保护的临界区内执行必要的操作，以避免出现死锁或数据不一致。确保在适当的时候获取和释放锁，避免锁的泄漏。在多线程环境中，调试可能会更加困难。使用工具和技术来帮助查找并发问题，如数据竞争或死锁。同时，进行充分的测试，包括并发测试，以确保代码的正确性。本次实验使我们更紧密深入的了解到了多线程环境中的并发问题，并采取了适当的措施来确保数据的安全性。

完成日期：2023 年 10 月 25 日