**Threads and Locking**

一、实验目的

探索线程和线程的并行编程，使用哈希表锁定

二、实验内容

[Homework: Locking (mit.edu)](https://pdos.csail.mit.edu/6.828/2018/homework/lock.html)

三、实验步骤与实验结论

（1）下载ph.c并编译运行，ph.c程序主要工作是使用随机数生成10000个keys，然后利用线性hash将其插入5个哈希槽中，最后再取出每个key。

（2） $ gcc -g -O2 ph.c -pthread

$ ./a.out 2

pc.c程序主要工作是使用随机数生成keys，然后利用线性hash将其插入5个哈希槽中，最后再取出每个key。

　　 当使用2个线程来运行时，将会把keys分成2组，让2个线程分别put key，最后2个线程都做get key

可以指定对哈希表执行放置和获取操作的线程数。每个线程分两个阶段运行。第一阶段，每个线程把NKEYS/nthread个键放到哈希表中。在第二阶段，每个线程从哈希表中获取NKEYS。打印语句告诉我们每个线程的每个阶段需要多长时间。底部的完成时间是应用程序的总运行时间。

（3）定义并初始化全局变量lock

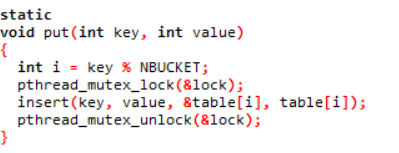




（4）put函数加互斥锁

只用1个互斥锁时，加锁解锁操作可以在put函数前后。若根据槽的个数使用相应个数的自旋锁，加锁解锁操作可以在insert函数前后，能有效地提高运行效率。

　　 由于get操作并未涉及到修改hash表，只有读取操作，所以可以不用进行锁保护。



（5）可知

1）单线程与两个线程的完成时间大致相同，但运行得到的结果两个线程的情况是单线程的两倍。

2）在单线程运行时key的丢失为0，而双线程时对于key的丢失时很严重的.

（6）

Q:首先用一个线程测试你的代码，然后用两个线程测试它。它是正确的吗(即，你消除了丢失的钥匙吗？)？双线程版本比单线程版本快吗？

A：此时，密钥不再丢失了。但是双线程和单线程执行的平均时间是差不多的。

Q:那么为什么在两个线程的情况下，密钥会丢失呢？

A：在用户没有设定线程间的调度策略时，系统默认采取基于时间片轮转的调度策略。所以当NKEYS=100000时，会出现两个线程的多次轮转，有可能导致key的丢失。

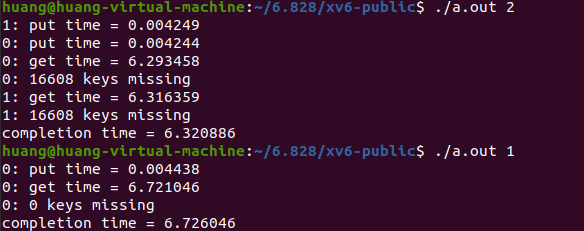
put函数调用的insert函数。在执行insert函数时可能发生了线程切换，此时的n还是为空，再进行连接时，就会丢失掉线程2插入的表项。

Q：修改代码，以便在保持正确性的同时并行运行get操作。(提示:这个应用程序中的锁对于正确性是必要的吗？）

A：get只是遍历哈希表，并不会改变哈希表，所以加不加锁都不会导致key丢失。所以单纯只讨论key会不会丢失时get里不用加locks

Q：修改您的代码，以便一些put操作并行运行，同时保持正确性。

A：取消get操作中的lock操作



四、反思总结

知道了线程执行顺序，不一定每次都相同，需要等所有线程完成put操作，所以先完成put操作的子线程会print出put的时间然后阻塞等另一个线程完成put操作。然后再通过get来遍历哈希表