自旋锁的性能问题

忙等待式的解法的通用模型是这样的，当一个进程想要进入临界区时，先检查是否允许进入，若不允许，则该进程将原地等待，直到允许为止。

```
locked = UNLOCK

while True:
  if locked != UNLOCK: #Case 1: 无法获取锁，进入原地等待。
      retry
  locked = LOCK # Case 2: 成功获得锁
```

同时，自旋锁会有如下的性能问题。

1. 自旋（共享变量）会触发处理器间的缓存同步，延迟增加。
2. 除了进入临界区的线程，其他处理器上的线程都在空转。
3. 争抢锁的处理器越多，利用率越低。
4. 获得自旋锁的线程可能被操作系统切换出去。

> 实现 100% 的资源浪费

自旋锁的使用场景

- 临界区几乎不“拥堵”
- 持有自旋锁时禁止执行流切换

使用场景：操作系统内核的并发数据结构（短临界区）

- 操作系统可以关闭中断和抢占
- 保证锁的持有者在很短的时间内可以释放锁

即，几乎不存在进程会等待锁的情况。

互斥锁--让权等待

作业那么多，与其干等 Online Judge 发布，不如把自己（CPU）让给其他作业（线程）执行？

“让”不是 C 语言代码可以做到的（C 代码只能计算）。把锁的实现放到操作系统里就好啦！

```
syscall(SYSCALL_lock, &lk);
// 试图获得 lk，但如果失败，就切换到其他线程
syscall(SYSCALL_unlock, &lk);
// 释放 lk，如果有等待锁的线程就唤醒
```