# 延时锁

# 自旋锁

## 概述

假设我们有一个具有两个处理器core1和core2的计算机，现在在这台计算机上运行的程序中有两个线程：T1和T2分别在处理器core1和core2上运行，两个线程之间共享着一个资源。

首先，我们说明互斥锁的工作原理，互斥锁是一种sleep-waiting的锁。假设线程T1获取互斥锁并且正在core1上运行时，此时线程T2也想要获取互斥锁（pthread\_mutex\_lock），但是由于T1正在使用互斥锁使得T2被阻塞。当T2处于阻塞状态时，T2倍放入到等待队列中去，处理器core2会去处理其他任务而不必一直等待（忙等）。也就是说处理器不会因为线程阻塞而空闲着，它去处理其他事务去了。

而自旋锁就不同了，自旋锁是一种busy-waiting的锁。也就是说，如果T1正在使用自旋锁，而T2也去申请这个自旋锁，此时T2肯定得不到这个自旋锁。与互斥锁相反的是，此时运行T2的处理器core2会一直不断地循环检查锁是否可用（自旋锁请求），直到获取到这个自旋锁为止。

从自旋锁的名字也可以看出来，如果一个线程想要获取一个被使用的自旋锁，那么它会一直占用CPU请求这个自旋锁使得CPU不能去做其他的事情，直到获取这个锁为止，这就是“自旋锁”的含义。

当发生阻塞时，互斥锁可以让CPU去处理其他的任务，而自旋锁让CPU一直不断循环请求获取这个锁。通过两个含义的对比可以让我们知道“自旋锁”是比较耗费CPU的。

## 函数

### spin\_lock\_init

### spin\_lock

### spin\_trylock

### spin\_unlock

### spin\_is\_locked