

本节内容

互斥锁

王道考研/CSKAOYAN.COM

1

进程互斥：锁

1. 互斥锁

解决临界区最简单的工具就是互斥锁（mutex lock）。一个进程在进入临界区时应获得锁；在退出临界区时释放锁。函数 `acquire()` 获得锁，而函数 `release()` 释放锁。

每个互斥锁有一个布尔变量 `available`，表示锁是否可用。如果锁是可用的，调用 `acquire()` 会成功，且锁不再可用。当一个进程试图获取不可用的锁时，会被阻塞，直到锁被释放。

```
acquire()
{
    while(!available)
        ; //忙等待
    available = false; //获得锁
}

release() {
    available = true; //释放锁
}
```

`acquire()` 或 `release()` 的执行必须是原子操作，因此互斥锁通常采用硬件机制来实现。

互斥锁的主要缺点是忙等待，当有一个进程在临界区中，任何其他进程在进入临界区时必须连续循环调用 `acquire()`。当多个进程共享同一 CPU 时，就浪费了 CPU 周期。因此，互斥锁通常用于多处理器系统，一个线程可以在一个处理器上等待，不影响其他线程的执行。

需要连续循环忙等的互斥锁，都可称为自旋锁（spin lock），如 TSL 指令、swap 指令、单标志法

王道考研/CSKAOYAN.COM

2

进程互斥：锁

特性：

- 需忙等，进程时间片用完才下处理机，违反“让权等待”
- 优点：等待期间不用切换进程上下文，多处理器系统中，若上锁的时间短，则等待代价很低
- 常用于多处理器系统，一个核忙等，其他核照常工作，并快速释放临界区
- 不太适用于单处理机系统，忙等的过程中不可能解锁

```
do {
    entry section;
    critical section;
    exit section;
    remainder section;
} while(true)
```

//进入区
//临界区
//退出区
//剩余区

```
acquire()
while(!available)
    ;
available = false;
```

//忙等待
//获得锁

```
release() {
    available = true;
```

//释放锁

王道考研/CSKAOYAN.COM