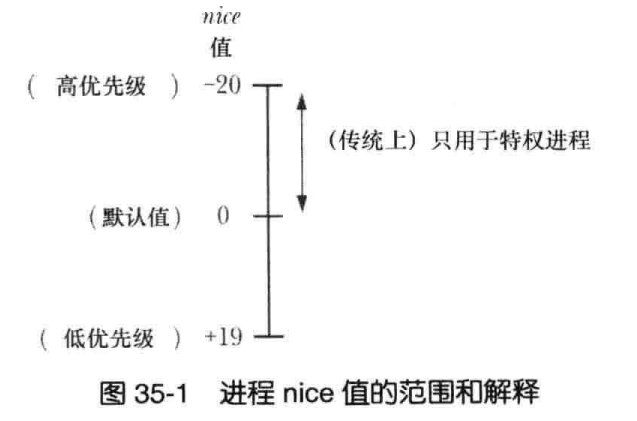
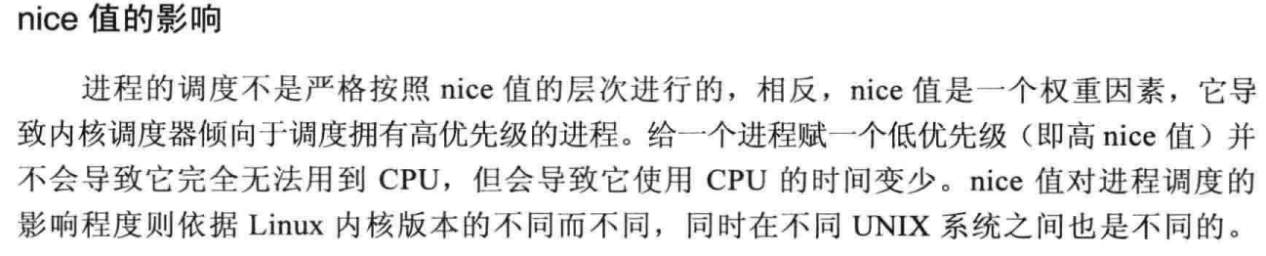
第三十五章

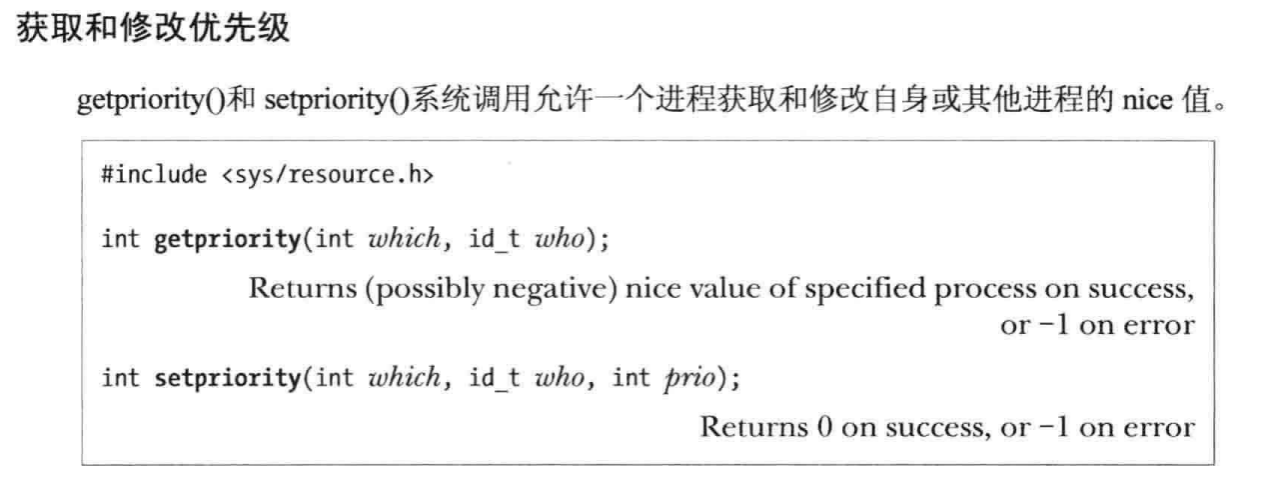
1. 进程优先级和调度：
2. 进程优先级（nice值）：

进程特性nice值间接地影响内核调度算法，取值范围是-20到19 。在传统的Unix实现中，只有特权进程才能给自己赋予高优先级，而非特权进程只能给自己赋予低优先级。



Nice值在fork和exec中得到保持。





至于非特权进程可以改变nice的值的权限，Linux实现和SUSv3规定不一样。具体查看书本。从内核2.6.12开始，非特权进程可以提高优先级，不过与RLIMIT\_NICE有关。

1. 实时进程调度概述：

实时应用对调度器有更加严格的要求：

1. 实时应用必须为外部输入提供担保最大响应时间。
2. 高优先级进程必须保证互斥的访问CPU直到它完成或自动释放。
3. 实时应用能够精确的控制其组件进程的调度时间。

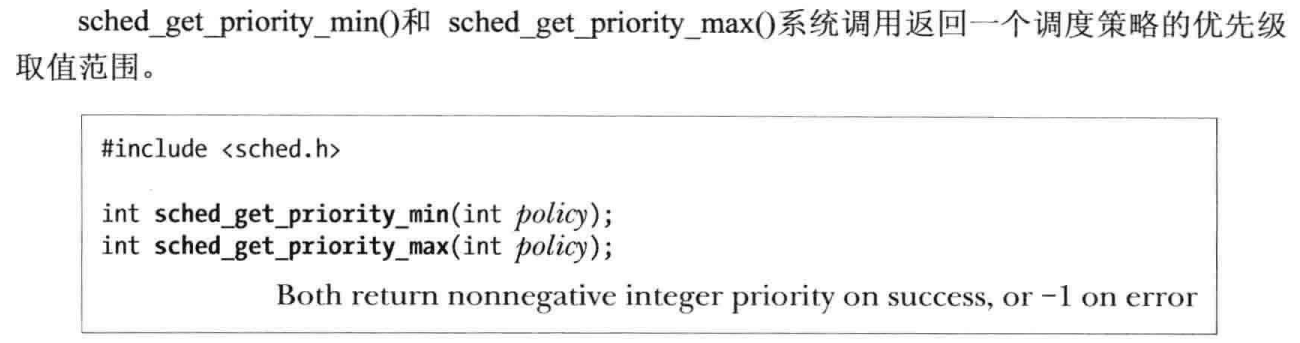
SUSv3提供了两个实时调度策略，这两种策略优先级高于标准的循环时间分享策略。

对于多处理系统而言，高优先级先运行的规则不一定适用，因为每个CPU维护各自的队列。

软实时和硬实时的对比。硬实时更严格一点，要求更高一点。

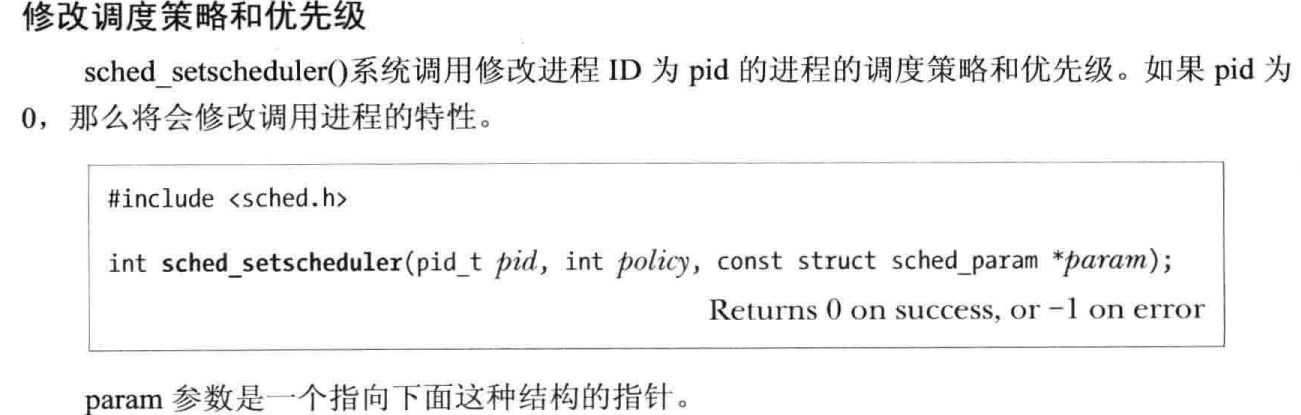
介绍了实时调度策略：SCHED\_FIFO,SCHED\_RR和SCHED\_BATCH,SCHED\_IDLE策略。

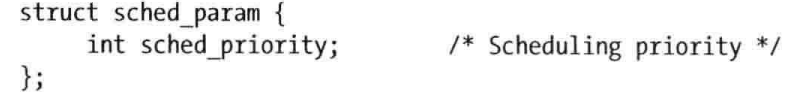
1. 实时调度API：

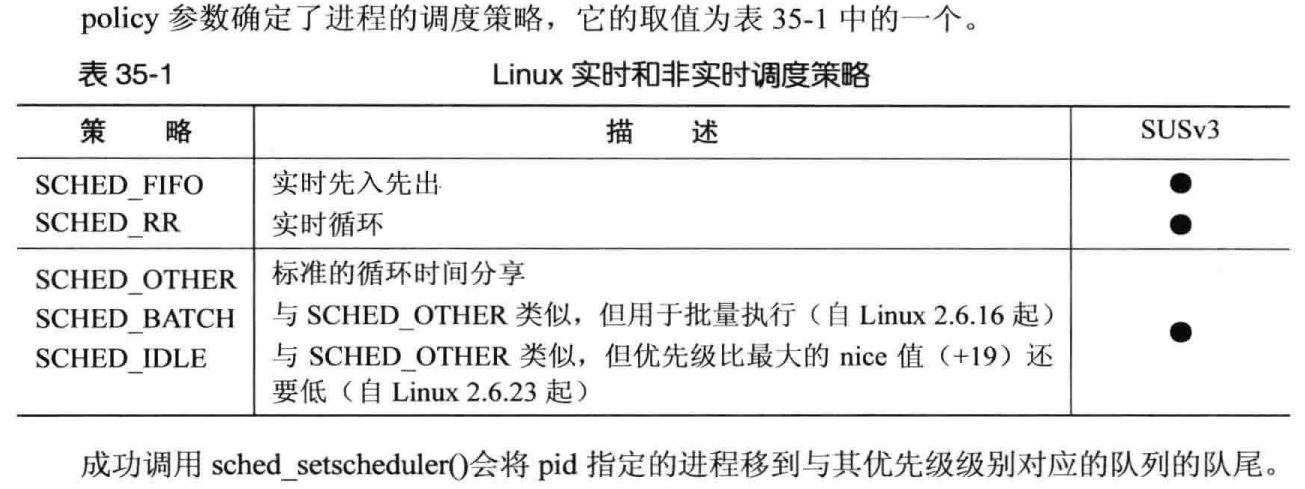


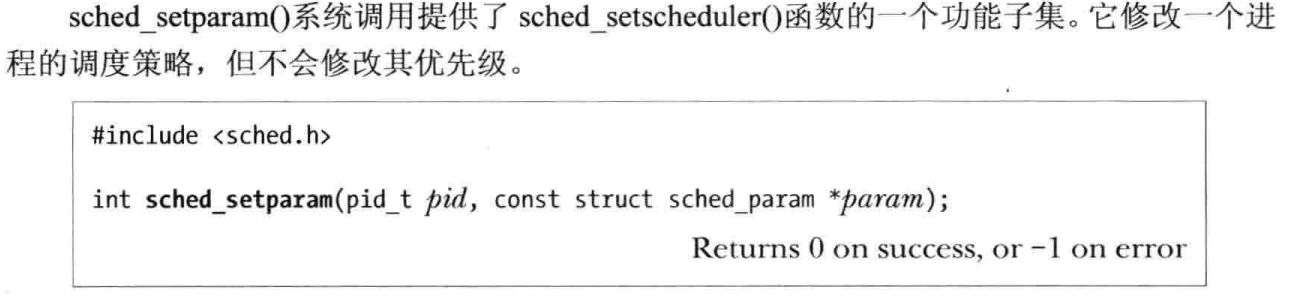
在Linux上，实时调度范围是1-99.

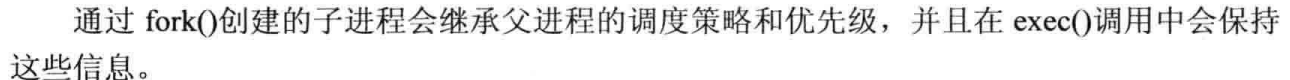
1. 修改和获取策略和优先级：

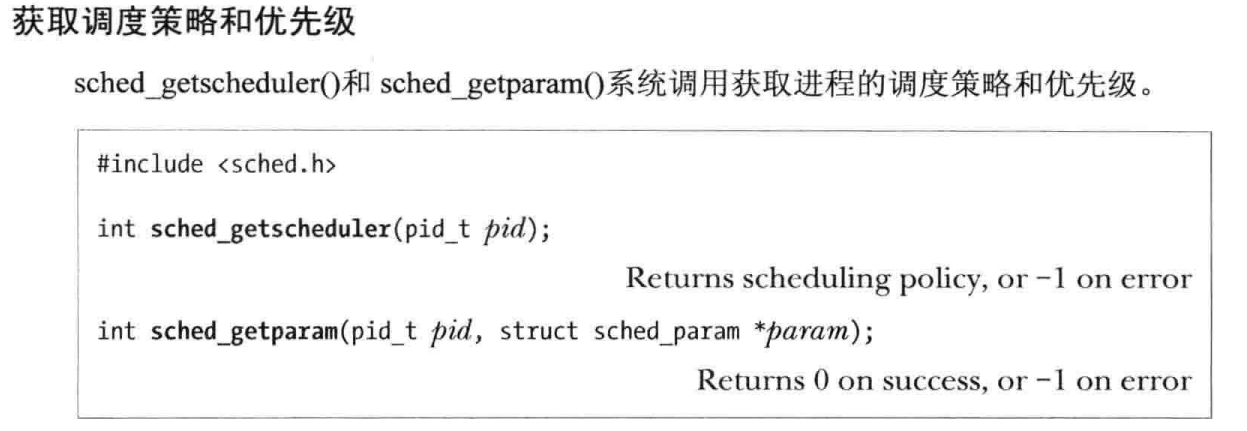












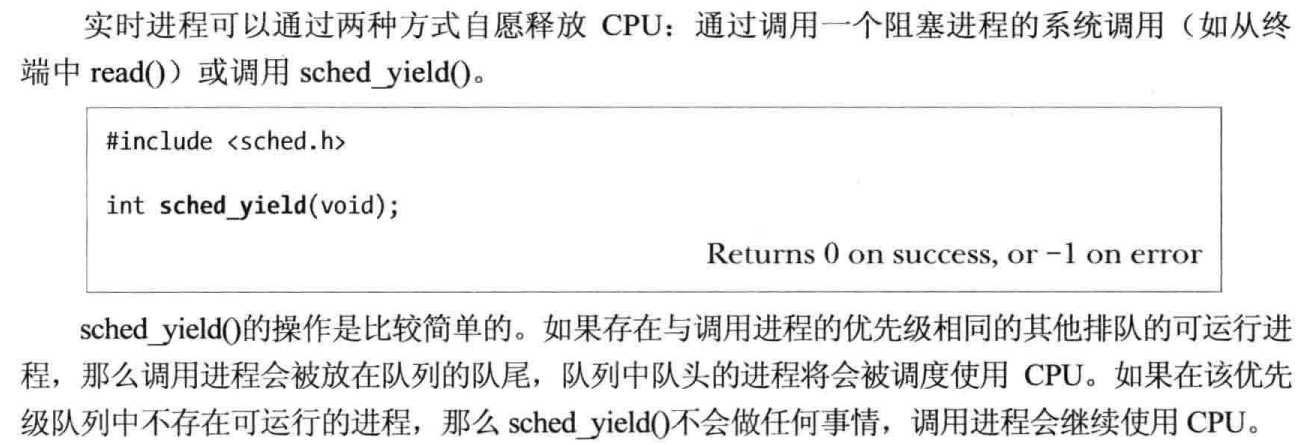
1. 权限和资源限制会影响对调度参数的变更：

主要讲述了非特权进程能对哪些进程做哪些改变，非标准的资源限制RLIMIT\_RTPRIO能够带来什么改变，具体查看书本。

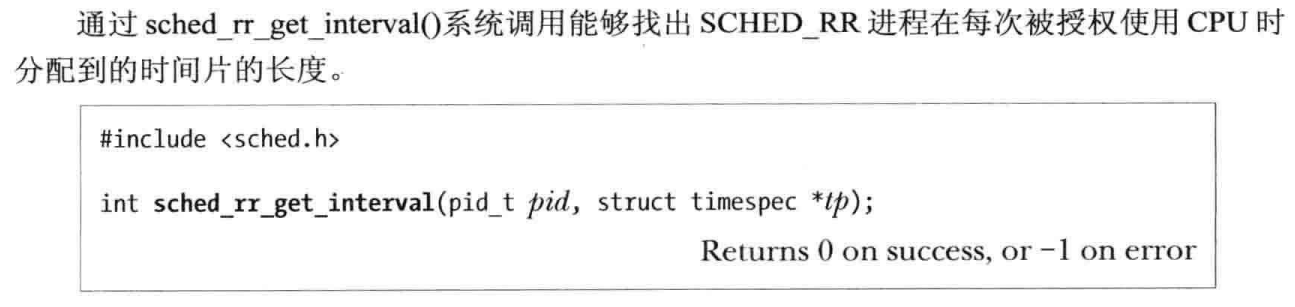
1. 防止实时进程锁住系统：
2. 设置一个合理的CPU软时间。有两种方法。
3. 使用alarm设置一个定时器。
4. 创建一个优先级更高的监控进程。
5. 避免子进程进程特权调度策略：

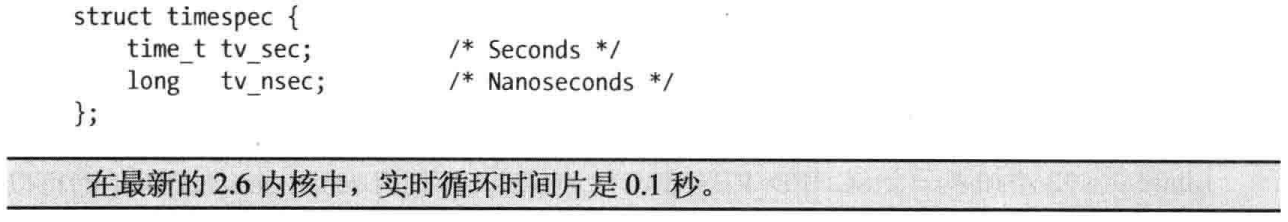
Linux增加了SCHED\_RESET\_ON\_FORK标志。

1. 释放CPU：



1. SCHED\_RR时间片：

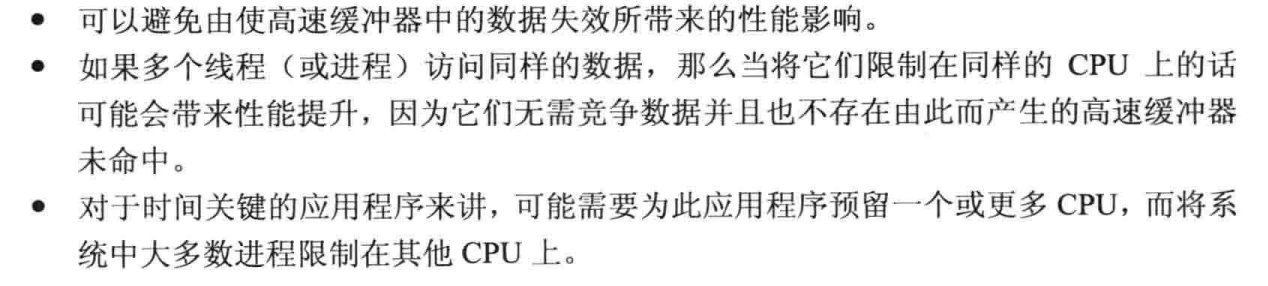


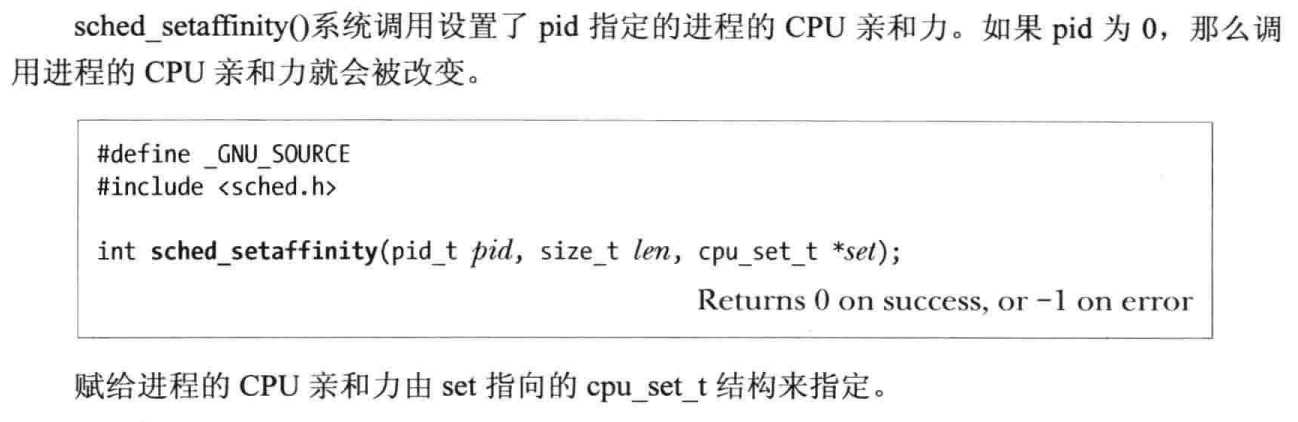


1. CPU亲和力：

在多处理系统上，一个进程在同一个CPU上运行能保证较高的效率，因为同一个CPU上有缓存，内核为此会尝试给进程软亲和力。

也可以手动设置硬亲和力，原因如下：





CPU亲和力是线程级特性。

