**1.进程调度**

**进程分类：**

    传统分类：I/O受限，CPU受限

    另一种分类：

      交互式进程：经常与用户交互，等待键盘鼠标等操作，接受输入后必须很快被唤醒，例如文本编辑器。

      批处理进程：经常在后台运行，不必很快的被响应，例如编译程序，数据库搜索引擎。

      实时进程：有很强的调度需要，绝不会被低优先级进程阻塞，进程响应时间短而且变化很小，例如音乐视频程序。

**进程状态：**

    #define TASK\_RUNNING   0  就绪状态，等待CPU资源

    #define TASK\_INTERRUPTIBLE  1  阻塞状态，可以被信号唤醒

    #define TASK\_UNINTERRUPTIBLE   2  阻塞状态，不可被信号唤醒

    #define TASK\_ZOMBIE   4  僵尸态，实际已运行结束但未注销

    #define TASK\_STOPPED   8  挂起态，用于调试，接收到SIGSTOP后会进入该状态，接收到SIGCONT恢复运行。

**调度策略：**基于分时技术，以时间片为单位对进程进行调度。时间片太短会导致系统额外开销变多，太长会导致进

程看起来并不是并发，因此时间片长度根据具体情况而定。进程的优先级是动态的，调度程序跟踪进程正在做什么周

期性的调整他们的优先级，较长时间未使用cpu的进程的优先级会增加，占用cpu时间长的进程优先级会减小。

**进程抢占：**Linux进程时抢占式的，进程进入TASK\_RUNNING状态，内核会检查它的优先级是否大于当前进程的优先

级。无论进程处于内核态还是用户态都可以被其他进程抢占，被抢占的进程只不过不再使用CPU，但是还处于TASK\_RUNNING

状态

**进程调度算法：**Linux2.6以后每个CPU都有自己的可运行进程队列，较好的处理了进程调度时间和处理器数量的关系。

Linux至少有一个可运行进程，即pid为0的swapper进程，只有当cpu没有其他可运行进程可以执行时才会执行这个进程，

多处理器系统每个CPU都有自己的swapper进程。#pid 0 和 pid 1

    进程有静态优先级和动态优先级，均为100-139，数值越大，优先级越低。根据静态优先级的大小来确定进程所拥有的基本时间片大小，与优先级低的进程相比，优先级高的进程获得更长的CPU时间片

C:\Users\123\Desktop\Snap3.png

    动态优先级是调度程序选择新进程来运行时的数，他与静态优先级关系如下

C:\Users\123\Desktop\Snap2.png

    bonus的值小于5表示降低动态优先级，大于5表示增加动态优先级。bonus的值取决于过去一段时间内进程的平均睡眠

时间。

**实时优先级：**每个实时进程都有一个实时优先级，从1到99，调度程序总是让实时优先级高的进程执行。如果具有相同

的优先级，cpu选择先出现在队列中的进程执行，当出现以下情况是实时进程会被取代。

      被具有更高优先级的进程抢占

      执行阻塞进入睡眠或进程停止，被杀死

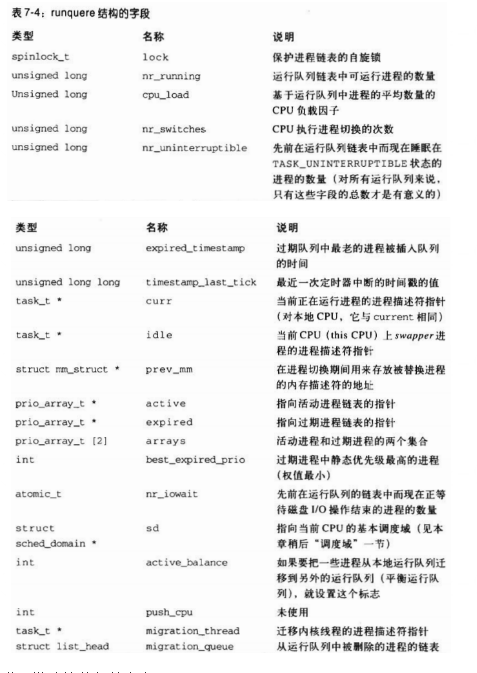
      进程自愿放弃cpu

      基于时间片轮转的实时进程。

**调度中的数据结构 ＃用户态和内核态的抢占形式，内核态可以被抢占吗？多课CPU numa 多核CPU smp 两种内存访问方式，numa时运行队列如何放置。**

**runqueue(运行队列):每个CPU都有自己的运行队列。this\_rq()产生本地cpu运行队列，cpu\_rq(n)产生索引为n的cpu的运行**

**队列。**

****

      进程描述符的相关字段

        thread\_info->flags                          存放TIF\_NEED\_RESCHED标志，如果必须调用调度程序设置该标志

        thread\_info->cpu                            可运行进程所处队列的cpu号

        prio                                                    动态优先级

        static\_prio     静态优先级

        rt\_priority      实时优先级

        (prio\_array\_t \*) array            运行队列集合

        (struct list\_head) run\_list         指向运行队列下一个和前一个元素

        sleep\_avg 平均睡眠时间

        timestamp  最近一次进程切换时间

        time\_slice   进程的时间片中还剩余的时钟节拍数（怎么算出来的）

时钟节拍是怎么确定下来（x86），时钟节拍存在什么变量中。

**调度程序用到的函数：**

       scheduler\_tick() :维持当前最新的time\_slice计数器

       try\_to\_wake\_up() ：唤醒睡眠进程

       recalc\_task\_prio() ：更新进程动态优先级

       schedule() ：选择要被执行的新进程

       load\_balance() ：维持多处理器系统中运行队列的平衡

**系统调用：**

      nice()：允许进程改变他的优先级

      getpriority() setpriority()：返回给定组中所有进程nice最低的进程的值，把给定组中所有进程的nice值进行设置