2.4.3 交互式系统中的调度

现在考察用于交互式系统中的一些调度算法，它们在个人计算机、服务器和其他类系统中都是常用的。

1. 轮转调度

一种最古老、最简单、最公平且使用最广的算法是轮转调度（round robin）。每个进程被分配一个时间段，称为时间片（quantum），即允许该进程在该时间段中运行。如果在时间片结束时该进程还在运行，则将剥夺CPU并分配给另一个进程。如果该进程在时间片结束前阻塞或结束，则CPU立即进行切换。时间片轮转调度很容易实现，调度程序所要做的就是维护一张可运行进程列表，如图2-41a所示。当一个进程用完它的时间片后，就被移到队列的末尾，如图2-41b所示。

时间片轮转调度中惟一有趣的一点是时间片的长度。从一个进程切换到另一个进程是需要一定时间进行管理事务处理的——保存和装入寄存器值及内存映像、更新各种表格和列表、清除和重新调入内存高速缓存等。假如进程切换（process switch），有时称为上下文切换（context switch），需要1ms，包括切换内存映像、清除和重新调入高速缓存等。再假设时间片设为4ms。有了这些参数，则CPU在做完4ms有用的工作之后，CPU将花费（即浪费）1ms来进行进程切换。因此，CPU时间的20%浪费在管理开销上。很清楚，这一管理时间太多了。

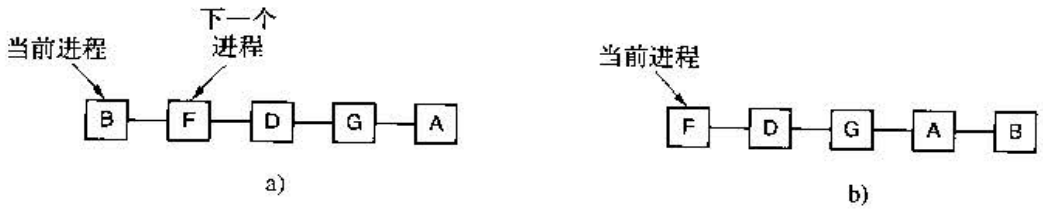

图2-41 轮转调度：a) 可运行进程列表，b) 进程B用完时间片后的可运行进程列表

为了提高CPU的效率，我们可以将时间片设置成，比方说，100ms，这样浪费的时间只有1%。但是，如果在一段非常短的时间间隔内到达50个请求，并且对CPU有不同的需求，那么，考虑一下，在一个服务器系统中会发生什么呢？50个进程会放在可运行进程的列表中。如果CPU是空闲的，第一个进程会立即开始执行，第二个直到100ms以后才会启动，以此类推。假设所有其他进程都用足了它们的时间片的话，最不幸的是最后一个进程在获得运行机会之前将不得不等待5秒钟。大部分用户会认为5秒的响应对于一个短命令来说是缓慢的。如果一些在队列后端附近的请求仅要求几毫秒的CPU时间，上面的情况会变得尤其糟糕。如果使用较短的时间片的话，它们将会获得更好的服务。

另一个因素是，如果时间片设置长于平均的CPU突发时间，那么不会经常发生抢占。相反，在时间片耗费完之前多数进程会完成一个阻塞操作，引起进程的切换。抢占的消失改善了性能，因为进程切换只会发生在确实逻辑上有需要的时候，即进程被阻塞不能够继续运行。

可以归结如下结论：时间片设得太短会导致过多的进程切换，降低了CPU效率；而设得太长又可能引起对短的交互请求的响应时间变长。将时间片设为20ms~50 ms通常是一个比较合理的折中。

2. 优先级调度

轮转调度做了一个隐含的假设，即所有的进程同等重要，而拥有和操作多用户计算机系统的人对此常有不同的看法。例如，在一所大学里，等级顺序可能是教务长首先，然后是教授、秘书、后勤人员，最后是学生。这种将外部因素考虑在内的需要就导致了优先级调度。其基本思想很清楚：每个进程被赋予一个优先级，允许优先级最高的可运行进程先运行。

即使在只有一个用户的PC机上，也会有多个进程，其中一些比另一些更重要。例如，与在屏幕上实时显示视频电影的进程相比，在后台发送电子邮件的守护进程应该被赋予较低的优先级。

为了防止高优先级进程无休止地运行下去，调度程序可以在每个时钟滴答（即每个时钟中断）降低当前进程的优先级。如果这个动作导致该进程的优先级低于次高优先级的进程，则进行进程切换。一个可采用的方法是，每个进程可以被赋予一个允许运行的最大时间片，当这个时间片用完时，下一个次高优先级的进程获得机会运行。

优先级可以是静态赋予或动态赋予。在一台军用计算机上，可以把将军所启动的进程设为优先级