## 2.4 CPU 调度

1. 调度的基本概念

处理机调度负责动态地把处理器分配给进程或内核级线程。处理机调度也称CPU调度或进程调度，在有线程的操作系统中也称线程调度。为了最大限度的提高CPU 利用率，多道程序设计的目标是保持总是有进程可供执行。在单处理机系统中，一次只能运行一个进程；其它的任何进程都必须等到CPU 空闲时才能够被重新调度。

多道程序设计的思想十分简单。一个进程持续运行直到它必须等待某些操作（I/O请求是个典型）的完成。在简单的计算机系统中，进程等待时CPU 将处于空闲状态；这浪费了所有的等待时间。利用多道程序设计，我们可以有效地利用这段时间。在内存中同时保留多个进程。当一个进程必须等待时，操作系统将CPU 撤离该进程并把CPU 分配给另一个进程。然后以这种方式继续运行。

从处理器调度的对象、时间、功能等不同角度，我们可把处理器调度分成不同类型。处理器调度不仅涉及选择哪一个就绪进程进入运行状态，还涉及何时启动一个进程的执行。按照调度所涉及的层次的不同，我们可把处理器调度分成高级调度、中级调度和低级调度三个层次。

高级调度也称为作业调度或宏观调度。从用户工作流程的角度，一次作业提交若干个流程，其中每个程序按照流程进行调度执行。中级调度涉及进程在内外存间的交换。从存储器资源管理的角度来看，把进程的部分或全部换出到外存上，可为当前运行进程的执行提供所需的内存空间，将当前进程所需部分换入到内存。指令和数据必须在内存里才能被处理器直接访问。低级调度也称为微观调度。从处理器资源分配的角度来看，处理器需要经常选择就绪进程或线程进入运行状态。

2.调度时机、切换与过程

有四种情况都会发生 CPU 调度：当一个进程从运行状态转换到等待状态时；当一个进程从运行状态转换到就绪状态时；当一个进程从等待状态转换到就绪状态时；当一个进程终止运行时。

从操作系统角度观察，进程或者其 PCB 总是在各种等待队列中，或者在队列之间迁移。如图 2-8。

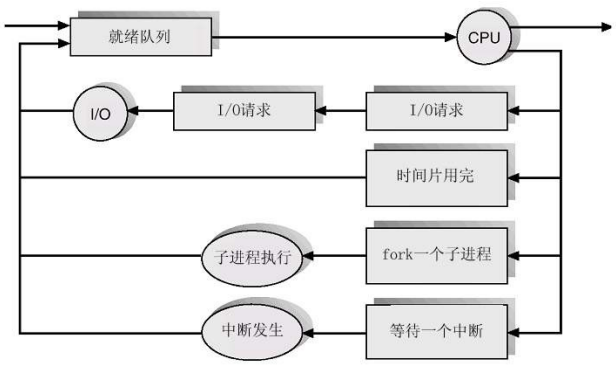


图 2-8　进程在各队列中的迁移

其中，就绪队列是个特殊的队列，它所包含的进程都已经得到了几乎所有资源，只缺少 CPU 了。CPU 调度的任务，就是从就绪队列中选择一个等待进程，并为其分配 CPU；选

择的目标，则是如何使 CPU 得到最好的利用，用户进程得到最好的服务。从图中看，CPU 调度使选中的进程从“就绪队列”节点，迁移到“CPU”节点。操作系统配备有专门的调度程序，列入 schedule()函数，只有该调度程序被调用执行了，才会发生一次 CPU 调度。所谓调度时机，就是什么时候去调用这个程序。

3.调度的基本准则

在操作系统中，如何选择调度方式和算法，在很大程度上取决于操作系统的类型和目标，选择调度方式和算法的准则有的是面向用户的，有的是面向系统的。常见的指标有 CPU 利用率、吞吐率、响应时间、周转时间、等待时间等。

（1）面向用户的准则和评价

1) 周转时间短

周转时间短是评价批处理系统的重要性能指标。作业周转时间是指从作业提交给系统开始，到作业完成为止的这段时间间隔。

计算机系统要使大多数用户对周转时间感到满意，引入平均周转时间 T 的评价指标，对有 n 个作业的系统 T 定义为

$$T = 1/n \times [\sum_{i=1}^{n} Ti]$$

Ti 为第 i 个作业的周转时间。

2) 响应时间快

响应时间是评价分时系统的性能指标。响应时间是从用户通过键盘提交一个请求开始，直至系统首次产生响应为止的时间。

3) 截止时间的保证

它是用来评价实时系统的重要指标，截止时间是某任务必须执行的最迟时间，或完成的最迟时间。

4) 优先权准则

在选择批处理、分时和实时系统的调度算法时，都可引用优先权准则，以便让那些紧急的作业（或事件），得到及时的处理。在要求较严格的场合，往往还需选择抢占调度方式，才能保证紧急作业得到及时的处理。

（2）面向系统的准则

1)达到系统设计目标

系统的设计目标是选择算法的主要依据。例如批处理系统所追求的是充分发挥和提高计算机的效率，分时系统则侧重于保护用户的请求及时给予响应，实时系统所关心的是不要丢失实时信息并给予处理，计算中心要求系统吞吐量要大等。

2)系统吞吐量大

用来评价批处理系统的重要指标。系统吞吐量是单位时间内完成的作业数，它与批处理作业的平均长度具有密切关系。

3)处理机利用率高

对于大中型多用户系统，由于 CPU 价格十分昂贵，所以处理机利用率成为衡量大、中型系统性能的十分重要指标，但对单用户微机或某些实时系统，该准则就不那么重要。

4)各类资源的平衡利用

在大中型系统中，有效地利用各类资源（包括 CPU、外存、I/O 设备等）也是一个重要指标，对于微型机和某些实时系统，该准则也不重要。

4.调度方式

进程调度可采用下述两种方式：

（1）非抢占方式

采用这种调度方式时，一旦把处理机分配给某进程后，便让进程一直执行，直到该进程完成或发生某事件而被阻塞时，才把处理机分配给其它进程，不允许某进程抢占已经分配出去的处理机。

（2）抢占方式

这种调度方式，允许进程调度程序根据某个原则，去停止某个正在执行的进程，将已分配给进程的处理机，重新分配给另一个进程。抢占的原则有：

- 时间片原则。各进程按时间片运行，当一个时间片用完后，便停止该进程的执行而重新进行调度。这个原则适用于分时系统。
- 优先权原则。通常是对一些重要的和紧急的进程赋予较高的优先权。当这种进程进入就绪队列时，例如由阻塞态转换为就绪态，或从静止就绪态转为活动就绪态时，或新创建进入就绪态的进程进入就绪队列时，如果其优先权比正在执行的进程优先权高，便停止正在执行的进程，将处理机分配给优先权高的进程，使之执行。

5.典型调度算法

典型的调度算法有：先来先服务调度算法，短作业（短任务、短进程、短线程）优先调度算法，时间片轮转调度算法，优先级调度算法，高响应比优先调度算法，多级反馈队列调度算法，等等。

先来先服务（First Come First Served，FCFS）调度算法是按照进程进入就绪队列的先后次序来挑选进程，先进入就绪队列的进程优先被挑中。先来先服务调度算法是最简单的调度算法，但是它会让短进程等待非常长的进程。FCFS 会产生所谓的 Belady 异常现象。

最短作业/进程优先（Shortest Job/Process First，SJF/SPF）调度算法是以进程所要求的 CPU 时间为标准，总是选取估计计算时间最短的进程投入运行。最短进程优先调度算法是局部最佳的方法，它满足最短平均等待时间的要求。但实现 SJF 调度比较困难，因为预测进程的下一个 CPU 需求区间的长度有难度。SJF 算法是优先权调度算法的特例，优先权调度和 SJF 调度会产生饥饿，老化（Aging）技术可解决饥饿问题。

时间片轮转（Round Robin,RR）调度算法是调度程序每次把 CPU 分配给就绪队列首进程使用一个时间片，例如 100ms，就绪队列中的每个进程轮流地运行一个这样的时间片。时间片轮转调度算法对于分时（交互）系统更为合适。RR 算法的主要问题是选择时间片，如果时间片太大，那么 RR 调度就成了 FCFS 调度；如果时间片太小，那么因上下文切换而引起的调度开销就过大。

优先级（Priority）调度算法是根据确定的优先数来选取进程，每次总是选择优先级高的进程。规定用户进程优先数的方法是多种多样的，进程的优先级的设置可以是静态的，也可以是动态的。

高响应比优先（Highest Response Ratio Next，HRRN）调度算法计算就绪队列进程的响应，调度时总是选择响应比最高的就绪进程得到 CPU。响应比=（进程已等待时间 + 进程要求运行时间）／进程要求运行时间。此调度算法首先有利于短进程，但也兼顾到等待时间长的进程，该算法是 FCFS 算法和 SJF 算法的折衷。

多队列调度算法（Multilevel Queue，MQ）是根据进程的性质和类型的不同，将就绪队列再分为若干个子队列，所有进程根据不同情况排入相应的队列中，而不同的就绪队列采用不同的调度算法。最为常用的是前台交互队列（使用 RR 调度）和后台批处理队列（使用 FCFS 调度）的组合。

多级反馈队列（Multilevel Feedback Queue Scheduling，MFQ）调度算法在多级队列算法的基础上，允许就绪进程在队列之间迁移。

FCFS 算法是非抢占的，RR 算法是抢占的，SJF 算法和优先级算法既可以是抢占的，也可以是非抢占的。

如果操作系统在内核级支持线程，那么必须调度线程而不是调度进程。