的必要信息。

第四，在一个I/O中断发生时，必须做出调度决策。如果中断来自I/O设备，而该设备现在完成了工作，某些被阻塞的等待该I/O的进程就成为可运行的就绪进程了。是否让新就绪的进程运行，这取决于调度程序的决定，或者让中断发生时运行的进程继续运行，或者应该让某个其他进程运行。

如果硬件时钟提供50Hz、60Hz或其他频率的周期性中断，可以在每个时钟中断或者在每k个时钟中断时做出调度决策。根据如何处理时钟中断，可以把调度算法分为两类。非抢占式调度算法挑选一个进程，然后让该进程运行直至被阻塞（阻塞在I/O上或等待另一个进程），或者直到该进程自动释放CPU。即使该进程运行了若干个小时，它也不会被强迫挂起。这样做的结果是，在时钟中断发生时不会进行调度。在处理完时钟中断后，如果没有更高优先级的进程等待到时，则被中断的进程会继续执行。

相反，抢占式调度算法挑选一个进程，并且让该进程运行某个固定时段的最大值。如果在该时段结束时，该进程仍在运行，它就被挂起，而调度程序挑选另一个进程运行（如果存在一个就绪进程）。进行抢占式调度处理，需要在时间间隔的末端发生时钟中断，以便把CPU控制返回给调度程序。如果没有可用的时钟，那么非抢占式调度就是惟一的选择了。

3. 调度算法分类

毫无疑问，不同的环境需要不同的调度算法。之所以出现这种情形，是因为不同的应用领域（以及不同的操作系统）有不同的目标。换句话说，在不同的系统中，调度程序的优化是不同的。这里有必要划分出三种环境：

1) 批处理。

2) 交互式。

3) 实时。

批处理系统在商业领域仍在广泛应用，用来处理薪水册、存货清单、账目收入、账目支出、利息计算（在银行）、索赔处理（在保险公司）和其他的周期性的作业。在批处理系统中，不会有用户不耐烦地在终端旁等待一个短请求的快捷响应。因此，非抢占式算法，或对每个进程都有长时间周期的抢占式算法，通常都是可接受的。这种处理方式减少了进程的切换从而改善了性能。这些批处理算法实际上相当普及，并经常可以应用在其他场合，这使得人们值得去学习它们，甚至是对于那些没有接触过大型机计算的人们。

在交互式用户环境中，为了避免一个进程霸占CPU拒绝为其他进程服务，抢占是必需的。即便没有进程想永远运行，但是，某个进程由于一个程序错误也可能无限期地排斥所有其他进程。为了避免这种现象发生，抢占也是必要的。服务器也归于此类，因为通常它们要服务多个突发的（远程）用户。

然而在有实时限制的系统中，抢占有时是不需要的，因为进程了解它们可能会长时间得不到运行，所以通常很快地完成各自的工作并阻塞。实时系统与交互式系统的差别是，实时系统只运行那些用来推进现有应用的程序，而交互式系统是通用的，它可以运行任意的非协作甚至是有恶意的程序。

4. 调度算法的目标

为了设计调度算法，有必要考虑什么是一个好的调度算法。某些目标取决于环境（批处理、交互式或实时），但是还有一些目标是适用于所有情形的。在图2-39中列出了一些目标，我们将在下面逐一讨论。

在所有的情形中，公平是很重要的。相似的进程应该得到相似的服务。对一个进程给予较其他等价的进程更多的CPU时间是不公平的。当然，不同类型的进程可以采用不同方式处理。可以考虑一下在核反应堆计算机中心安全控制与发放薪水处理之间的差别。

与公平有关的是系统策略的强制执行。如果局部策略是，只要需要就必须运行安全控制进程（即便这意味着推迟30秒钟发薪），那么调度程序就必须保证

所有系统
公平——给每个进程公平的CPU份额
策略强制执行——看到所宣布的策略执行
平衡——保持系统的所有部分都忙碌
批处理系统
吞吐量——每小时最大作业数
周转时间——从提交到终止间的最小时间
CPU利用率——保持CPU始终忙碌
交互式系统
响应时间——快速响应请求
均衡性——满足用户的期望
实时系统
满足截止时间——避免丢失数据
可预测性——在多媒体系统中避免品质降低

图2-39 在不同环境中调度算法的一些目标