中，也可以用在交互式系统中。我们将稍后讨论这个问题。

1. 先来先服务

在所有调度算法中，最简单的是非抢占式的先来先服务（first-come first-severd）算法。使用该算法，进程按照它们请求CPU的顺序使用CPU。基本上，有一个就绪进程的单一队列。早上，当第一个作业从外部进入系统，就立即开始并允许运行它所期望的时间。不会中断该作业，因为它需要很长的时间运行。当其他作业进入时，它们就被安排到队列的尾部。当正在运行的进程被阻塞时，队列中的第一个进程就接着运行。在被阻塞的进程变为就绪时，就像一个新来到的作业一样，排到队列的末尾。

这个算法的主要优点是易于理解并且便于在程序中运用。就难以得到的体育或音乐会票的分配问题而言，这对那些愿意在早上两点就去排队的人们也是公平的。在这个算法中，一个单链表记录了所有就绪进程。要选取一个进程运行，只要从该队列的头部移走一个进程即可；要添加一个新的作业或阻塞一个进程，只要把该作业或进程附加在相应队列的末尾即可。还有比这更简单的理解和实现吗？

不过，先来先服务也有明显的缺点。假设有一个一次运行1秒钟的计算密集型进程和很少使用CPU但是每个都要进行1000次磁盘读操作才能完成的大量I/O密集型进程存在。计算密集进程运行1秒钟，接着读一个磁盘块。所有的I/O进程开始运行并读磁盘。当该计算密集进程获得其磁盘块时，它运行下一个1秒钟，紧跟随着的是所有I/O进程。

这样做的结果是，每个I/O进程在每秒钟内读到一个磁盘块，要花费1000秒钟才能完成操作。如果有一个调度算法每10ms抢占计算密集型进程，那么I/O进程将在10秒钟内完成而不是1000秒钟，而且还不会对计算密集型进程产生多少延迟。

2. 最短作业优先

现在来看一种适用于运行时间可以预知的另一个非抢占式的批处理调度算法。例如，一家保险公司，因为每天都做类似的工作，所以人们可以相当精确地预测处理1000个索赔的一批作业需要多少时间。当输入队列中有若干个同等重要的作业被启动时，调度程序应使用最短作业优先（shortest job first）算法，请看图2-40。这里有4个作业A、B、C、D，运行时间分别为8、4、4、4分钟。若按图中的次序运行，则A的周转时间为8分钟，B为12分钟，C为16分钟，D为20分钟，平均为14分钟。

图2-40 最短作业优先调度的例子：a) 按原有次序运行4个作业；b) 按最短作业优先次序运行

现在考虑使用最短作业优先算法运行这4个作业，如图2-40b所示。目前周转时间分别为4、8、12和20分钟，平均为11分钟。可以证明最短作业优先是最优的。考虑有4个作业的情况，其运行时间分别为a、b、c、d。第一个作业在时间a结束，第二个在时间a + b结束，以此类推。平均周转时间为（4a + 3b + 2c + d）/4。显然a对平均值影响最大，所以它应是最短作业，其次是b，再次是c，最后的d只影响它自己的周转时间。对任意数目作业的情况，道理完全一样。

有必要指出，只有在所有的作业都可同时运行的情形下，最短作业优先算法才是最优化的。作为一个反例，考虑5个作业，从A到E，运行时间分别是2、4、1、1和1。它们的到达时间是0、0、3、3和3。开始，只能选择A或B，因为其他三个作业还没有到达。使用最短作业优先，将按照A、B、C、D、E的顺序运行作业，其平均等待时间是4.6。但是，按照B、C、D、E、A的顺序运行作业，其平均等待时间则是4.4。

3. 最短剩余时间优先

最短作业优先的抢占式版本是最短剩余时间优先（shortest remaining time next）算法。使用这个算法，调度程序总是选择剩余运行时间最短的那个进程运行。再次提醒，有关的运行时间必须提前掌握。当一个新的作业到达时，其整个时间同当前进程的剩余时间做比较。如果新的进程比当前运行进程需要更少的时间，当前进程就被挂起，而运行新的进程。这种方式可以使新的短作业获得良好的服务。