链滴

列出目前 OS 涉及到的几种进程调度算法，并简述对他们的理解

作者：xhaoxiong

原文链接：https://ld246.com/article/1525264100684

来源网站：链滴

[优点:有利于长作业]
1、先来先服务服务调度
先来先服务（FCFS）调度算法是一种最简单的调度算法，该算法既可用于作业调度，也可用于进程调
。

FCFS算法比较有利于长作业（进
程），而不利于短作业（进程）。由此可知，本算法适合于CPU繁忙型作业，而不利于I/O繁忙型的
业（进程）。

[优点:有利于短作业]
2、短作业优先调度
短作业（进程）优先调度算法（SJ/PF）是指对短作业或短进程优先调度的算法，该算法既可用于作
调度，也可用于进程调度。

但其对长作业不利；不能保证紧迫性作业（进程）被及时处理；作业的长短只是被估算出来的。

[优点:满足紧迫作业]
3、优先级调度算法
为了照顾紧迫型作业，使之在进入系统后便获得优先处理，引入了最高优先权优先(FPF)调度算法。此
法常被用于批处理系统中，作为作业调度算法，也作为多种操作系

统中的进程调度算法，还可用于实时系统中。当把该算法用于作业调度时，系统将从后备队列中选择
干个优先权最高的作业装入内存。当用于进程调度时，该算法是把处理机

分配给就绪队列中优先权最高的进程，这时，又可进一步把该算法分成如下两种。

1）非抢占式优先权算法
在这种方式下，系统一旦把处理机分配给就绪队列中优先权最高的进程后，该进程便一直执行下去，
至完成；或因发生某事件使该进程放弃处理机时，系统方可再将处理

机重新分配给另一优先权最高的进程。这种调度算法主要用于批处理系统中；也可用于某些对实时性
求不严的实时系统中。

2) 抢占式优先权调度算法
在这种方式下，系统同样是把处理机分配给优先权最高的进程，使之执行。但在其执行期间，只要又
现了另一个其优先权更高的进程，进程调度程序就立即停止当前进

程(原优先权最高的进程)的执行，重新将处理机分配给新到的优先权最高的进程。因此，在采用这种
度算法时，是每当系统中出现一个新的就绪进程i 时，就将其优先权Pi与正

在执行的进程j 的优先权Pj进行比较。如果Pi≤Pj，原进程Pj便继续执行；但如果是Pi>Pj，则立即停止P
的执行，做进程切换，使i 进程投入执行。显然，这种抢占式的优先权调

度算法能更好地满足紧迫作业的要求，故而常用于要求比较严格的实时系统中，以及对性能要求较高
批处理和分时系统中。

[优点:折衷的一种作业方法，缺点：增加系统开销]
4、高响应比优先调度
在批处理系统中，短作业优先算法是一种比较好的算法，其主要的不足之处是长作业的运行得不到保
。如果我们能为每个作业引入前面所述的动态优先权，并使作业的优

先级随着等待时间的增加而以速率a 提高，则长作业在等待一定的时间后，必然有机会分配到处理机该优先权的变化规律可描述为：

$$\text{优先权} = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$$

由于等待时间与服务时间之和就是系统对该作业的响应时间，故该优先权又相当于响应比RP。据此，可表示为：

$$R_P = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{响应时间}}{\text{要求服务时间}}$$

由上式可以看出：

(1) 如果作业的等待时间相同，则要求服务的时间愈短，其优先权愈高，因而该算法有利于短作业。

(2) 当要求服务的时间相同时，作业的优先权决定于其等待时间，等待时间愈长，其优先权愈高，因它实现的是先来先服务。

(3) 对于长作业，作业的优先级可以随等待时间的增加而提高，当其等待时间足够长时，其优先级便升到很高，从而也可获得处理机。简言之，该算法既照顾了短作业，
又考虑了作业到达的先后次序，不会使长作业长期得不到服务。因此，该算法实现了一种较好的折衷当然，在利用该算法时，每要进行调度之前，都须先做响应比的计算，这

会增加系统开销。

[调度分配一些可以抢占的资源]
5、轮转调度
时间片轮转(Round Robin，RR)法的基本思路是让每个进程在就绪队列中的等待时间与享受服务的时成比例。在时间片轮转法中，需要将CPU的处理时间分成固定大小

的时间片，例如，几十毫秒至几百毫秒。如果一个进程在被调度选中之后用完了系统规定的时间片，又未完成要求的任务，则它自行释放自己所占有的CPU而排到就绪队列

的末尾，等待下一次调度。同时，进程调度程序又去调度当前就绪队列中的第一个进程。

显然，轮转法只能用来调度分配一些可以抢占的资源。这些可以抢占的资源可以随时被剥夺，而且可将它们再分配给别的进程。CPU是可抢占资源的一种。但打印机等

资源是不可抢占的。由于作业调度是对除了CPU之外的所有系统硬件资源的分配，其中包含有不可抢资源，所以作业调度不使用轮转法。在轮转法中，时间片长度的选取非

常重要。首先，时间片长度的选择会直接影响到系统的开销和响应时间。如果时间片长度过短，则调程序抢占处理机的次数增多。这将使进程上下文切换次数也大大增加，从

而加重系统开销。反过来，如果时间片长度选择过长，例如，一个时间片能保证就绪队列中所需执行间最长的进程能执行完毕，则轮转法变成了先来先服务法。时间片长度的

选择是根据系统对响应时间的要求和就绪队列中所允许最大的进程数来确定的。

在轮转法中，加入到就绪队列的进程有3种情况，一种是分给它的时间片用完，但进程还未完成，回就绪队列的末尾等待下次调度去继续执行。另一种情况是分给该进程
的时间片并未用完，只是因为请求I/O或由于进程的互斥与同步关系而被阻塞。当阻塞解除之后再回就绪队列。第三种情况就是新创建进程进入就绪队列。如果对这些进程区

别对待，给予不同的优先级和时间片，从直观上看，可以进一步改善系统服务质量和效率。例如，我可把就绪队列按照进程到达就绪队列的类型和进程被阻塞时的阻塞原因分

成不同的就绪队列，每个队列按FCFS原则排列，各队列之间的进程享有不同的优先级，但同一队列内先级相同。这样，当一个进程在执行完它的时间片之后，或从睡眠中被

唤醒以及被创建之后，将进入不同的就绪队列。

[综合性机制]
6、多级反馈队列
前面介绍的各种用作进程调度的算法都有一定的局限性。如短进程优先的调度算法，仅照顾了短进程忽略了长进程，而且如果并未指明进程的长度，则短进程优先和基于

进程长度的抢占式调度算法都将无法使用。而多级反馈队列调度算法则不必事先知道各种进程所需的行时间，而且还可以满足各种类型进程的需要，因而它是目前被公认的一

种较好的进程调度算法。在采用多级反馈队列调度算法的系统中，调度算法的实施过程如下所述。

(1) 应设置多个就绪队列，并为各个队列赋予不同的优先级。第一个队列的优先级最高，第二个队列之，其余各队列的优先权逐个降低。该算法赋予各个队列中进程执行

时间片的大小也各不相同，在优先权愈高的队列中，为每个进程所规定的执行时间片就愈小。例如，二个队列的时间片要比第一个队列的时间片长一倍，……，第i+1个队列

的时间片要比第i个队列的时间片长一倍。

(2) 当一个新进程进入内存后，首先将它放入第一队列的末尾，按FCFS原则排队等待调度。当轮到该程执行时，如它能在该时间片内完成，便可准备撤离系统；如果它在

一个时间片结束时尚未完成，调度程序便将该进程转入第二队列的末尾，再同样地按FCFS原则等待调执行；如果它在第二队列中运行一个时间片后仍未完成，再依次将它放

入第三队列，……，如此下去，当一个长作业(进程)从第一队列依次降到第n队列后，在第n 队列便采按时间片轮转的方式运行。

(3) 仅当第一队列空闲时，调度程序才调度第二队列中的进程运行；仅当第1～(i-1)队列均空时，才会度第i队列中的进程运行。如果处理机正在第i队列中为某进程服务

时，又有新进程进入优先权较高的队列(第1～(i-1)中的任何一个队列)，则此时新进程将抢占正在运行程的处理机，即由调度程序把正在运行的进程放回到第i队列的末尾，把

处理机分配给新到的高优先权进程。

原文来自csdn