调度算法 - brizer的博客 - 博客频道

分类：
操作系统（4）

目录(?)[+]

在多道程序环境中，主存中有着多个进程，其数目往往多于处理机数量。这就要求系统能按照某种算法，动态地把处理机分配给就绪队列中的一个进程，使之执行，分配处理机的任务是由处理机调度程序完成的。

处理机调度

在多道程序系统中，一个作业被提交后必须经过处理机调度后，方能获得处理机执行。对于批量型作业而言，通常需要经历作业调度（也称为高级调度）和进程调度（也称为低级调度）两个过程才能获得处理机；而对于终端型作业而言，通常只需要经过进程调度就可以获得处理机。除了上述两种调度，操作系统中往往也设置了中级调度，用来提高内存的利用率。

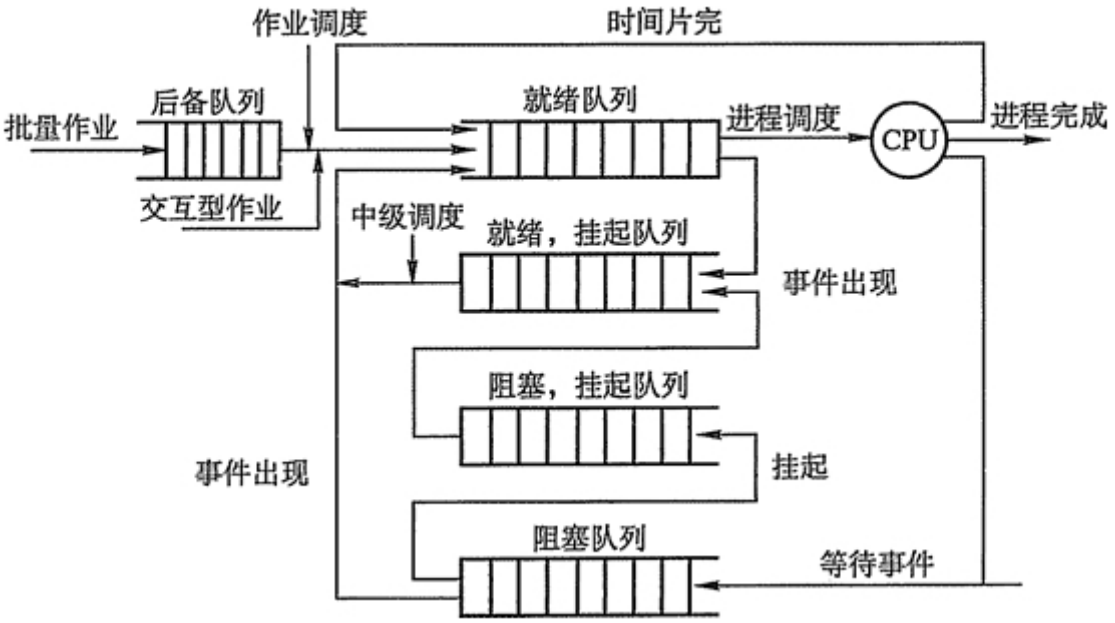
下面我们分别来谈谈这几种调度。首先是高级调度，其主要功能就是根据某种算法，把外存上处于后备队列中的那些作业调入内存，也就是说，调度的对象是作业。那么什么叫做作业呢？

作业是一个比程序更为广泛的概念，它不仅包含了通常的程序和数据，而且配有一份作业说明书，系统就是根据该说明书来对程序的运行进行控制。前面所说的某种算法，就是我们后面会提到的几种常用调度算法。

低级调度用于决定就绪队列中的哪个进程应该获得处理机，然后再由分派程序执行把处理机分派给该进程的具体操作。

中级调度的主要目的是为了提高内存利用率和系统吞吐量。它的工作原理就是将暂时不能运行的进程调至外存上去，此时的状态称为挂起。相反当内存空闲时，再将他们调回内存，此时的状态称为就绪，挂在就绪队列上等待进程的调度。

三种调度的关系如下图：



那么我们如何评价不同的调度算法呢？为了比较各算法的性能，人们提出了很多评价准则：

1. cpu利用率：cpu是计算机系统中最重要的昂贵的资源，所以应该尽可能使cpu保持工作状态；

2. 系统吞吐量：单位时间内cpu完成作业的数量，长作业需要消耗较长的处理机时间，所以会降低系统的吞吐量；

3. 周转时间：从作业提交到作业完成所经历的时间，包括作业等待、在就绪队列中排队、在处理机上运行以及进行输入输出操作所花费时间的总和。

周转时间=作业完成时间-作业提交时间

平均周转时间=（作业1的周转时间+作业2的周转时间+…+作业n的周转时间）/n

带权周转时间=周转时间/作业实际运行时间

平均带权周转时间=（作业1的带权周转时间+…+作业n的带权周转时间）/n

4. 等待时间: 是指进程处于等处理机状态时间之和，等待时间越长，用户满意度越低。处理机调度算法实际上并不影响作业执行或输入/输出操作的时间，只影响作业在就绪队列中等待所花的时间。因此，衡量一个调度算法优劣常常只需简单地考察等待时间。

5. 响应时间: 是指从用户提交请求到系统首次产生响应所用的时间。在交互式系统中，周转时间不可能是最好的评价准则，一般采用响应时间作为衡量调度算法的重要准则之一。从用户角度看，调度策略应尽量降低响应时间，使响应时间处在用户能接受的范围之内。

下面我们来正式介绍一下操作系统中的几种常用调度算法吧：

先来先服务调度算法

FCFS调度算法是一种最简单的调度算法，该调度算法既可以用于作业调度也可以用于进程调度。

在作业调度中，算法每次从后备作业队列中选择最先进入该队列的一个或几个作业，将它们调入内存，分配必要的资源，创建进程并放入就绪队列。

在进程调度中，FCFS调度算法每次从就绪队列中选择最先进入该队列的进程，将处理机分配给它，使之投入运行，直到完成或因某种原因而阻塞时才释放处理机。
下面通过一个实例来说明FCFS调度算法的性能。假设系统中有4个作业，它们的提交时间分别是8、8.4、8.8、9，运行时间依次是2、1、0.5、0.2，系统采用FCFS调度算法

作业号	提交时间	运行时间	开始时间	等待时间	完成时间	周转时间	带权周转时间
1	8	2	8	0	10	2	1
2	8.4	1	10	1.6	11	2.6	2.6
3	8.8	0.5	11	2.2	11.5	2.7	5.4
4	9	0.2	11.5	2.5	11.7	2.7	13.5

通过分析可得：

平均等待时间 t = (0+1.6+2.2+2.5)/4=1.575
平均周转时间 T = (2+2.6+2.7+2.7)/4=2.5
平均带权周转时间 W = (1+2.6+5.4+13.5)/4=5.625

从表面上看，它对所有作业都是公平的，但若一个长作业先到达系统，就会使后面许多短作业等待很长时间，因此它不能作为分时系统和实时系统的主要调度策略。但它常被结合在其他调度策略中使用。例如，在使用优先级作为调度策略的系统中，往往对多个具有相同优先级的进程按FCFS原则处理。
FCFS调度算法的特点是算法简单，但效率低；对长作业比较有利，但对短作业不利（相对SJF和高响应比）；有利于CPU繁忙型作业，而不利于I/O繁忙型作业。

短作业（进程）优先调度算法

短作业（进程）优先调度算法是指对短作业（进程）优先调度的算法。短作业优先(SJF)调度算法是从后备队列中选择一个或若干个估计运行时间最短的作业，将它们调入内存运行。而短进程优先(SPF)调度算法，则是从就绪队列中选择一个估计运行时间最短的进程，将处理机分配给它，使之立即执行，直到完成或发生某事件而阻塞时，才释放处理机。

还是上面的例子，我们换用短作业法就会得到不一样的结果：

作业号	提交时间	运行时间	开始时间	等待时间	完成时间	周转时间	带权周转时间
1	8	2	8	0	10	2	1
2	8,4	1	10.7	2.3	11.7	3.3	3.3
3	8.8	0.5	10.2	1.4	10.7	1.9	3.8
4	9	0.2	10	1	10.2	1.2	6

平均等待时间 t = (0+2.3+1.4+1)/4=1.175
平均周转时间 T = (2+3.3+1.9+1.2)/4=2.1
平均带权周转时间 W = (1+3.3+3.8+6)/4=3.525

看起来短作业优先调度算法似乎很不错，但是也存在不容忽视的缺点：

该算法对长作业不利，SJF调度算法中长作业的周转时间会增加。更严重的是，如果有一长作业进入系统的后备队列，由于调度程序总是优先调度那些 （即使是后进来的）短作业，将导致长作业长期不被调度（“饥饿”现象，注意区分“死锁”。后者是系统环形等待，前者是调度策略问题）。

该算法完全未考虑作业的紧迫程度，因而不能保证紧迫性作业会被及时处理。

由于作业的长短只是根据用户所提供的估计执行时间而定的，而用户又可能会有意或无意地缩短其作业的估计运行时间，致使该算法不一定能真正做到短作业优先调度。
注意，SJF调度算法的平均等待时间、平均周转时间最少。

优先级调度算法

在作业调度中，优先级调度算法每次从后备作业队列中选择优先级最高的一个或几个作业，将它们调入内存，分配必要的资源，创建进程并放入就绪队列。在进程调度中，优先级调度算法每次从就绪队列中选择优先级最高的进程，将处理机分配给它，使之投入运行。

根据新的更高优先级进程能否抢占正在执行的进程，可将该调度算法分为：

非剥夺式优先级调度算法。当某一个进程正在处理机上运行时，即使有某个更为重要或紧迫的进程进入就绪队列，仍然让正在运行的进程继续运行，直到由于其自身的原因而主动让出处理机时（任务完成或等待事件），才把处理机分配给更为重要或紧迫的进程。

剥夺式优先级调度算法。当一个进程正在处理机上运行时，若有某个更为重要或紧迫的进程进入就绪队列，则立即暂停正在运行的进程，将处理机分配给更重要或紧迫的进程。

而根据进程创建后其优先级是否可以改变，可以将进程优先级分为以下两种：

静态优先级。优先级是在创建进程时确定的，且在进程的整个运行期间保持不变。确定静态优先级的主要依据有进程类型、进程对资源的要求、用户要求。

动态优先级。在进程运行过程中，根据进程情况的变化动态调整优先级。动态调整优先级的主要依据为进程占有CPU时间的长短、就绪进程等待CPU时间的长短。

基于时间片的轮转调度算法

时间片轮转调度算法主要适用于分时系统。在这种算法中，系统将所有就绪进程按到达时间的先后次序排成一个队列，进程调度程序总是选择就绪队列中第一个进程执行，即先来先服务的原则，但仅能运行一个时间片，如100ms。在使用完一个时间片后，即使进程并未完成其运行，它也必须释放出（被剥夺）处理机给下一个就绪的进程，而被剥夺的进程返回到就绪队列的末尾重新排队，等候再次运行。

在时间片轮转调度算法中，时间片的大小对系统性能的影响很大。如果时间片足够大，以至于所有进程都能在一个时间片内执行完毕，则时间片轮转调度算法就退化为先来先服务调度算法。如果时间片很小，那么处理机将在进程间过于频繁切换，使处理机的开销增大，而真正用于运行用户进程的时间将减少。因此时间片的大小应选择适当。

顶
1