3.2　作业与作业调度

在多道批处理系统中，作业是用户提交给系统的一项相对独立的工作。

操作员把用户提交的作业通过相应的输入设备输入到磁盘存储器，并保存在一个后备作业队列中。再由作业调度程序将其从外存调入内存。

3.2.1　批处理系统中的作业

1. 作业和作业步

(1) 作业(Job)

(2) 作业步(Job Step)

作业：用户一次请求计算机系统为其完成任务所做工作的总和。它通常包括程序、数据和作业说明书，系统根据说明书对程序进行控制。

- 批处理系统中，是以作业为基本单位从外存调入内存的

作业步：指每个作业在运行期间，都必须经过若干个相对独立，又相互关联的顺序加工步骤，其中的每一个加工步骤称为作业步。

一个典型的作业可分成三个作业步：①“编译”作业步，②“链接装配”作业步，③“运行”作业步

2. 作业控制块(Job Control Block，JCB)

- 为了管理和调度作业，在多道批处理系统中，为每个作业设置了一个作业控制块JCB；
- JCB是作业在系统中存在的标志，保存了系统对作业进行管理和调度所需的全部信息；
- 通常在JCB中包含的内容有：作业标识、用户名称、用户账号、作业类型(CPU 繁忙型、I/O 繁忙型、批量型、终端型)、作业状态、调度信息(优先级、作业运行时间)、资源需求(预计运行时间、要求内存大小等)、资源使用情况等。

作业的工作步骤：

- 作业进入系统时，“作业注册”程序为该作业创建一个作业控制块JCB，然后根据作业类型，将它放到相应的作业后备队列中等待调度；
- 调度程序根据一定的算法调度作业，被调度到的作业将被装入内存；
- 作业运行期间，系统按照JCB中的信息和作业说明书对作业进行控制；
- 作业执行结束进入完成状态时，系统负责回收分配给它的资源，撤销该作业控制块。

3. 作业运行的三个阶段和三种状态

作业从进入系统到运行结束，通常需要经历收容（后备）、运行和完成三个阶段。相应的作业也就有“后备状态”、“运行状态”和“完成状态”。

(1) 后备阶段

(2) 运行阶段

(3) 完成阶段

- 操作员把用户提交的作业通过某种输入方式输入到硬盘上，再为该作业建立JCB，并把它放入作业后备队列中。
- 此时的作业状态称为“后备状态”。

3. 作业运行的三个阶段和三种状态

作业从进入系统到运行结束，通常需要经历收容（后备）、运行和完成三个阶段。相应的作业也就有“后备状态”、“运行状态”和“完成状态”。

(1) 后备阶段

(2) 运行阶段

(3) 完成阶段

- 作业被作业调度选中后，便为它分配必要的资源和建立进程，并将它放入就绪队列。
- 一个作业从第一次进入就绪状态开始，直到运行结束前，都处于“运行阶段”。

3. 作业运行的三个阶段和三种状态

作业从进入系统到运行结束，通常需要经历收容（后备）、运行和完成三个阶段。相应的作业也就有“后备状态”、“运行状态”和“完成状态”。

(1) 后备阶段

(2) 运行阶段

(3) 完成阶段

- 当作业完成或发生异常情况而提前结束时，作业就进入“完成阶段”，此时作业的状态为“完成状态”。
- 系统中的“终止作业”程序将会回收已分配给该作业的JCB和所有资源，并将作业运行结果信息形成输出文件后输出。

3.2.2　作业调度的主要任务

作业调度的主要任务是，根据JCB中的信息，检查系统中的资源能否满足作业对资源的需求，以及按照一定的调度算法，从外存的后备队列中选取某些作业调入内存，并为它们创建进程、分配必要的资源。然后再将新创建的进程排在就绪队列上等待调度。

因此，作业调度也称为接纳调度(Admission Scheduling)。

在每次执行作业调度时，都需做出以下两个决定。

1. 接纳多少个作业
2. 接纳哪些作业

3.2.2　作业调度的主要任务

作业调度的主要任务是，根据JCB中的信息，检查系统中的资源能否满足作业对资源的需求，以及按照一定的调度算法，从外存的后备队列中选取某些作业调入内存，并为它们创建进程、分配必要的资源。然后再将新创建的进程排在就绪队列上等待调度。

因此，作业调度也称为接纳调度(Admission Scheduling)。

在每次执行作业调度时，都需做出以下两个决定。

1. 接纳多少个作业
2. 接纳哪些作业

接纳多少个作业：

- 取决于多道程序度（**Degree of Multiprogramming**），即允许多少个作业同时在内存中运行。
- 内存中同时运行的作业数目太多，会影响系统的服务质量。如，周转时间长。
- 内存中同时运行的作业数目太少，会导致系统资源利用率和系统吞吐量低。

——多道程序度的确定需要根据计算机的系统规模、运行速度、作业大小，以及能否获得较好的系统性能等情况作出适当的抉择 。

1. 接纳多少个作业
2. 接纳哪些作业

接纳哪些作业：

- 选择后备队列中的哪些作业调入内存，取决于调度算法。
  - 先来先服务调度算法——最简单的
  - 短作业优先调度算法——较常用
  - 基于作业优先级的调度算法——较常用
  - “响应比高者优先”调度算法——较好

◆批处理系统中，作业进入系统后总是先驻留在外存的作业后备队列上，因此需要有作业调度，以便将它们分批装入内存；

◆分时系统中，为了及时响应，用户通过键盘输入的命令或数据都被直接从送入内存，因此无需配置作业调度机制，但需要某种接纳控制措施来限制进入系统的用户数目；

◆实时系统中，同样不需要作业调度，而必须具有接纳控制措施。

3.2.3　先来先服务(FCFS)和短作业优先(SJF)调度算法

1. 先来先服务(first-come first-served，FCFS)调度算法

- FCFS是最简单的调度算法
- 该算法**既可用于作业调度，也可用于进程调度**
- 当在作业调度中采用该算法时，系统将按照作业到达的先后次序来进行调度，或者说它是优先考虑在系统中等待时间最长的作业，而不管该作业所需执行时间的长短，从后备作业队列中选择几个最先进入该队列的作业，将它们调入内存，为它们分配资源和创建进程。然后把它放入就绪队列。
- 采用FCFS算法时，进程一直运行到完成或发生某事件而阻塞后，进程调度程序才将处理机分配给其他进程。

3.2.3　先来先服务(FCFS)和短作业优先(SJF)调度算法

1. 先来先服务(first-come first-served，FCFS)调度算法

- FCFS是最简单的调度算法
- 该算法既可用于作业调度，也可用于进程调度
- 当在作业调度中采用该算法时，系统将按照作业到达的先

FCFS算法在单处理机系统中很少作为主调度算法，经常把它与其他调度算法相结合使用，形成一种更为有效的调度算法。

如，可在系统中按照进程的优先级设置多个队列，每个优先级一个队列，每一个队列的调度都基于FCFS算法。

- 采用FCFS算法时，进程一直运行到完成或发生某事件而阻塞后，进程调度程序才将处理机分配给其他进程。

2. 短作业优先(short job first, SJF)的调度算法

在实际情况中，短作业（进程）占有很大比例，为了能使它们能比长作业优先执行，而产生了短作业优先调度算法。

1) 短作业优先算法

- SJF算法是以作业的长短来计算优先级，作业越短，其优先级越高；
- 作业的长短是以作业所要求的运行时间来衡量的；
- SJF算法可以分别用于作业调度和进程调度。

2) 短作业优先算法的缺点

SJF调度算法较之FCFS算法有了明显的改进，但仍然存在不容忽视的缺点：

(1) 必须预知作业的运行时间

即使是程序员也很难准确估计作业的运行时间，如果估计过低，系统就可能按估计的时间终止作业的运行，但此时作业并未完成，故一般都会偏长估计。

(2) 对长作业非常不利

长作业的周转时间会明显地增长。更严重的是，该算法完全忽视作业的等待时间，可能使作业等待时间过长，出现饥饿现象。

2) 短作业优先算法的缺点

SJF调度算法较之FCFS算法有了明显的改进，但仍然存在不容忽视的缺点：

(3) 人-机无法实现交互。

(4) 完全未考虑作业的紧迫程度，故不能保证紧迫性作业能得到及时处理。

3.2.4　优先级调度算法和高响应比优先调度算法

1. 优先级调度算法(priority-scheduling algorithm，PSA)

我们可以这样来看作业的优先级：

- 对于先来先服务调度算法，作业的等待时间就是作业的优先级，等待时间越长，其优先级越高；
- 对于短作业优先调度算法，作业的长短就是作业的优先级，作业所需运行的时间越短，其优先级越高。

这两种优先级都不能反映作业的紧迫程度。

3.2.4　优先级调度算法和高响应比优先调度算法

1. 优先级调度算法(priority-scheduling algorithm，PSA)

我们可以这样来看作业的优先级：

- 对于先来先服务调度算法，作业的等待时间就是作业的优先级，等待时间越长，其优先级越高；
- 对于短作业优先调度算法，作业的长短就是作业的优先级，作业所需运行的时间越短，其优先级越高。
- 优先级调度算法：基于作业的紧迫程度，由外部赋予作业相应的优先级，调度算法根据该优先级进行调度。

2. 高响应比优先调度算法(Highest Response Ratio Next，HRRN)

- 在批处理系统中，FCFS算法所考虑的只是作业的等待时间，而忽视了作业的运行时间；
- SJF算法正好与之相反，只考虑作业的运行时间，而忽视了作业的等待时间；
- 高响应比优先调度算法则是既考虑了作业的等待时间，又考虑作业运行时间的调度算法，因此既照顾了短作业，又不致使长作业的等待时间过长，从而改善了处理机调度的性能。

高响应比优先算法实现：

如果我们能为每个作业引入一个动态优先级，即优先级是可以改变的，令它随等待时间延长而增加，这将使长作业的优先级在等待期间不断地增加，等到足够的时间后，必然有机会获得处理机。该优先级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**，故该优先级又相当于响应比R_P。据此，优先级又可表示为：

$$R_P = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{等待时间}}{\text{要求服务时间}} + 1 = \frac{\text{响应时间}}{\text{要求服务时间}}$$

① 如果作业的等待时间相同，则要求服务的时间越短，其优先级越高，因此类似于**SJF**（短作业优先）算法；

② 如果要求服务的时间相同，作业的优先权取决于其等待时间，该算法又类似于**FCFS**（先来先服务）算法；

③ 对于长作业的优先级，可以随着等待时间的增加而提高，当其等待时间足够长时，也可以获得处理机。

- 该算法实现了较好的折中
- 但该算法每次进行调度前，都需要先做响应比的计算，会增加系统开销。

$$R_P = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{等待时间}}{\text{要求服务时间}} + 1 = \frac{\text{响应时间}}{\text{要求服务时间}}$$

3.3　进 程 调 度

进程调度是OS中必不可少的一种调度。因此在三种类型的OS中，都无一例外地配置了进程调度。

此外它也是对系统性能影响最大的一种处理机调度，相应的，有关进程调度的算法也较多。

3.3.1　进程调度的任务、机制和方式

1. 进程调度的任务

进程调度的任务主要有三个：

(1) 保存处理机的现场信息

(2) 按某种算法选取进程

(3) 把处理器分配给进程

- 如程序计数器、多个通用寄存器中的内容等

3.3.1　进程调度的任务、机制和方式

1. 进程调度的任务

进程调度的任务主要有三个：

(1) 保存处理机的现场信息

(2) 按某种算法选取进程

(3) 把处理器分配给进程

- 按照某算法从就绪队列中选取一个进程，将其状态改为运行状态，并准备把处理机分配给它

3.3.1　进程调度的任务、机制和方式

1. 进程调度的任务

进程调度的任务主要有三个：

(1) 保存处理机的现场信息

(2) 按某种算法选取进程

(3) 把处理器分配给进程

- 由分派程序把处理器分配给进程，此时需要把选中进程的PCB内有关处理机现场的信息装入处理器相应的各个寄存器中，把处理器的控制权交予该进程，让它从上次的断点处恢复运行

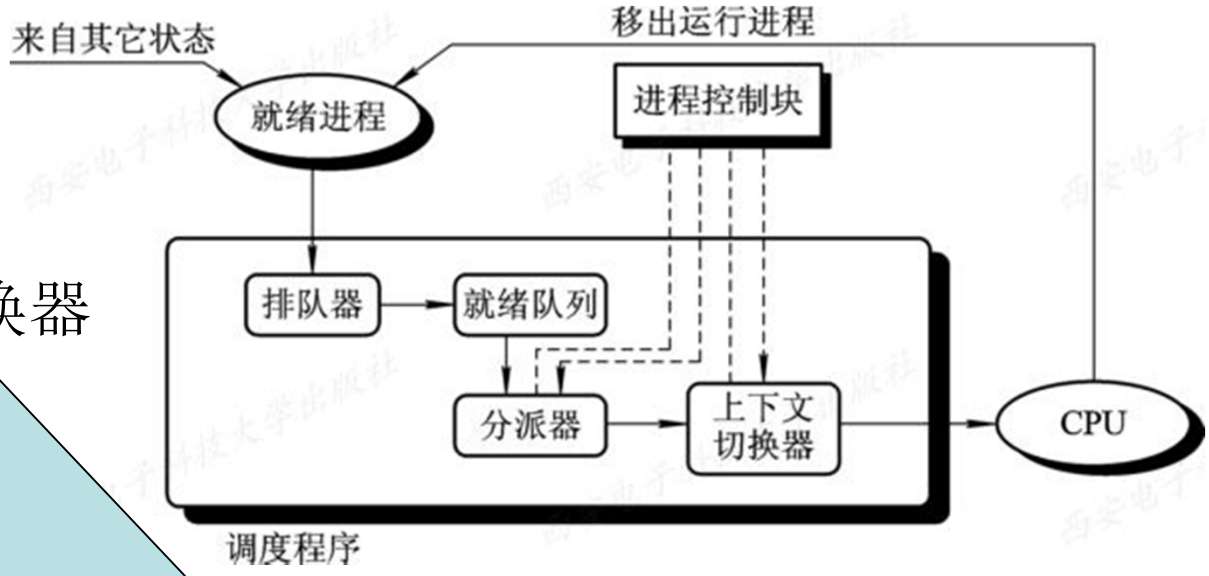
2. 进程调度机制

为了实现进程调度，在进程调度机制中，应具有如下三个基本部分：

(1) 排队器

(2) 分派器

(3) 上下文切换器



- 为了提高进程调度的效率，应事先将系统中的所有就绪进程按照一定的策略排成一个或多个队列，以便调度程序能最快地找到它。
- 以后每当有一个进程转变为就绪状态时，排队器便将它插入到相应的就绪队列中

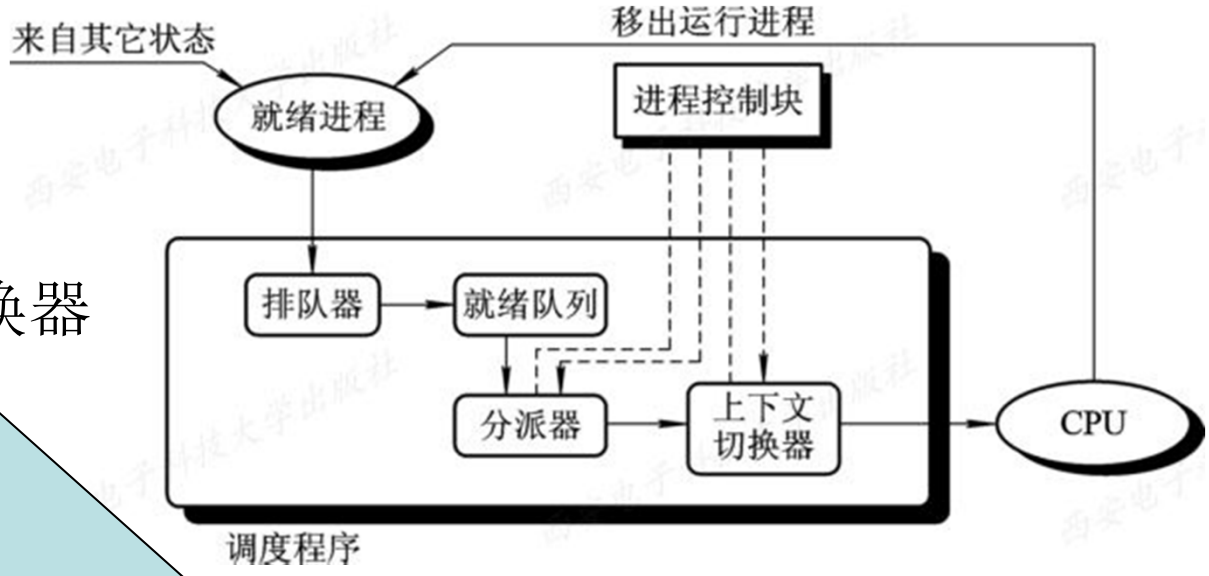
2. 进程调度机制

为了实现进程调度，在进程调度机制中，应具有如下三个基本部分：

(1) 排队器

(2) 分派器

(3) 上下文切换器



- 分派器依据进程调度程序所选定的进程，将其从就绪队列中取出，然后进行从分派器到新选出进程间的上下文切换，将处理机分配给新选出的进程。

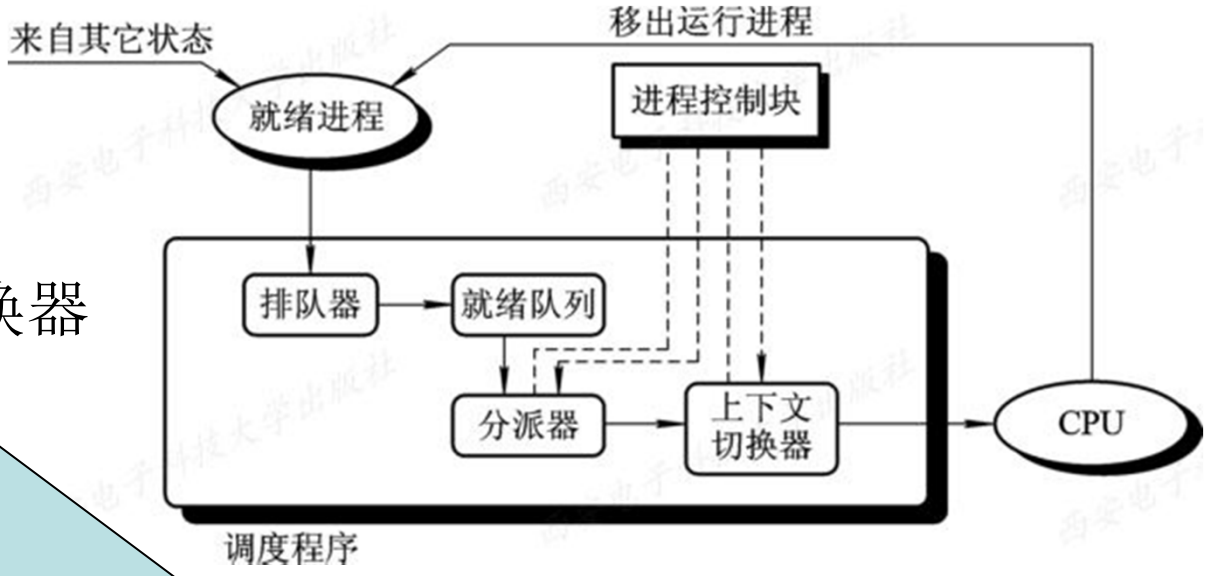
2. 进程调度机制

为了实现进程调度，在进程调度机制中，应具有如下三个基本部分：

(1) 排队器

(2) 分派器

(3) 上下文切换器



- 对处理机切换时，会发生两对上下文切换：

① OS保存当前进程的上下文，即将当前进程的处理机寄存器内容保存到该进程PCB内的相应单元，然后再装入分派进程的上下文，以便分派程序运行；

② 移除分派程序的上下文，把新选进程的CPU现场信息装入处理机各个相应的寄存器，以便新选进程运行。

3. 进程调度方式

1) 非抢占方式(Nonpreemptive Mode)

在采用这种调度方式时，一旦把处理机分配给某进程后，就一直让它运行下去，决不会因为时钟中断或任何其它原因去抢占当前正在运行进程的处理机，直至该进程完成，或发生某事件而被阻塞时，才把处理机分配给其它进程。

非抢占式调度方式可能引起进程调度的原因：

① 正在执行的进程运行完毕，或发生某事件使其无法再继续运行；

② 正在执行的进程因提出I/O请求而暂停执行；

③ 在进程通信或同步中，执行了某种原语操作，如block原语

3. 进程调度方式

1) 非抢占方式(Nonpreemptive Mode)

在采用这种调度方式时，一旦把处理机分配给某进程后，就一直让它运行下去，决不会因为时钟中断或任何其它原因

非抢占式调度方式优点：实现简单，系统开销小，适合大多数批处理系统，但不适合分时系统和实时系统

2) 抢占方式(Preemptive Mode)

这种调度方式允许调度程序根据某种原则，去暂停某个正在执行的进程，将已分配给该进程的处理机重新分配给另一进程。

2) 抢占方式(Preemptive Mode)

这种调度方式允许调度程序根据某种原则，去暂停某个正在执行的进程，将已分配给该进程的处理机重新分配给另一进程。

在现代OS中广泛采用抢占方式，这是因为：对于批处理系统，可以防止一个长进程长时间地占用处理机，以确保处理机能为所有进程提供更为公平的服务。在分时系统中，只有采用抢占方式才有可能实现人—机交互。在实时系统中，抢占方式能满足实时任务的需求。

但抢占方式比较复杂，所需付出的系统开销也较大。

2) 抢占方式(Preemptive Mode)

这种调度方式允许调度程序根据某种原则，去暂停某个

抢占方式必须遵循的主要原则：

① 优先权原则

允许优先级高的新到进程抢占当前进程的处理机。

② 短进程优先原则

允许新到的短进程可以抢占当前长进程的处理机，即新到的进程比正在执行的进程明显短时，将处理机分配给新到的短进程。

③ 时间片原则

各进程按照时间片轮转运行时，当正在执行的进程的一个时间片用完后，便停止该进程的执行而重新进行调度。

但抢占方式比较复杂，所需付出的系统开销也较大。

3.3.2　轮转调度算法

在分时系统中，最简单也较常用的是基于时间片的轮转（round robin，RR）调度算法。

该方法采取了非常公平的处理机分配方式，让就绪队列上的每个进程每次仅运行一个时间片。

1. 轮转法的基本原理

在轮转（RR）法中，系统将所有的就绪进程按FCFS策略排成一个就绪队列。系统可设置每隔一定时间（如30 ms）便产生一次中断，去激活进程调度程序进行调度，把CPU分配给队首进程，并令其执行一个时间片。当它运行完毕后，又把处理机分配给就绪队列中新的队首进程，也让它执行一个时间片。这样，就可以保证就绪队列中的所有进程在确定的时间段内，都能获得一个时间片的处理机时间。

2. 进程切换时机

在RR调度算法中，应在何时进行进程的切换，可分为两种情况：

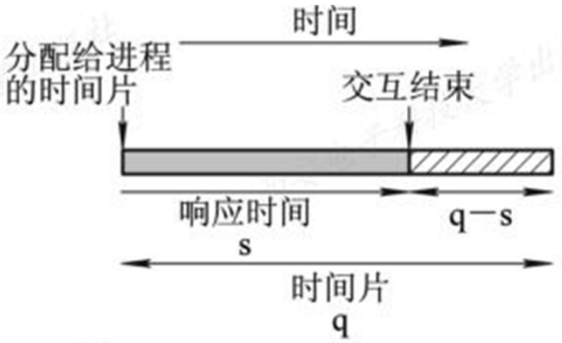
① 若一个时间片尚未用完，正在运行的进程便已经完成，就立即激活调度程序，将它从就绪队列中删除，再调度就绪队列中队首的进程运行，并启动一个新的时间片。

② 在一个时间片用完时，计时器中断处理程序被激活。如果进程尚未运行完毕，调度程序将把它送往就绪队列的末尾。

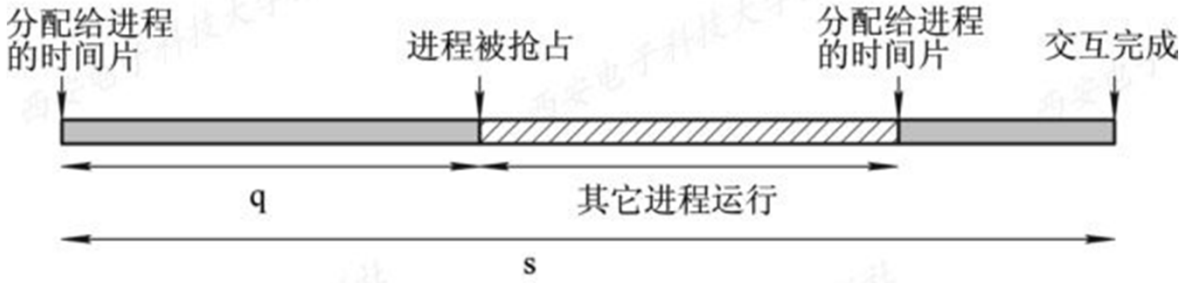
3. 时间片大小的确定

在轮转算法中，时间片的大小对系统性能有很大的影响。

下图示出了时间片大小对响应时间的影响，其中图（a）是时间片略大于典型交互的时间，而图（b）是时间片小于典型交互的时间。



(a) 时间片大于交互时间



(b) 时间片小于交互时间

下图示出了时间片分别为q = 1和q = 4时对平均周转时间的影响。

作业 / 情况 / 时间片	进程名	A	B	C	D	E	平均
	到达时间	0	1	2	3	4	
	服务时间	4	3	4	2	4	
RR q=1	完成时间	15	12	16	9	17	
	周转时间	15	11	14	6	13	11.8
	带权周转时间	3.75	3.67	3.5	3	3.33	3.46
RR q=4	完成时间	4	7	11	13	17	
	周转时间	4	6	9	10	13	8.4
	带权周转时间	1	2	2.25	5	3.33	2.5

图3-3　q = 1和q = 4时进程的周转时间

3.3.3　优先级调度算法

时间片轮转调度算法中做了一个隐含的假设，即系统中所有进程的紧迫性是相同的，但实际情况并非如此。因此，在进程调度算法中引入优先级，形成优先级调度算法。

1. 优先级调度算法的类型

优先级进程调度算法，是把处理机分配给就绪队列中优先级最高的进程。这时，又可进一步把该算法分成如下两种。

(1) 非抢占式优先级调度算法。

(2) 抢占式优先级调度算法。

3.3.3　优先级调度算法

时间片轮转调度算法中做了一个隐含的假设，即系统中

（1）非抢占式优先级调度算法

一旦把处理机分配给就绪队列中优先级最高的进程后，该进程便一直执行下去直至完成，或者因该进程发生某事件而放弃处理机时，系统方可将处理机重新分配给另一优先级最高的进程。

（2）抢占式优先级调度算法

把处理机分配给优先级最高的进程，使之执行。进程执行期间，只要出现了另一个优先级更高的进程，调度程序就将处理机分配给新到的优先级最高的进程。

◆ 抢占式优先级调度算法常用于对实时性要求较高的系统中。

2. 优先级的类型

1) 静态优先级

静态优先级是在创建进程时确定的，在进程的整个运行期间保持不变。优先级是利用某一范围内的一个整数来表示的，例如0～255中的某一整数，又把该整数称为优先数。确定进程优先级大小的依据有如下三个：

(1) 进程类型

(2) 进程对资源的需求

(3) 用户要求

- 通常系统进程的优先级高于一般用户进程的优先级

2. 优先级的类型

1) 静态优先级

静态优先级是在创建进程时确定的，在进程的整个运行期间保持不变。优先级是利用某一范围内的一个整数来表示的，例如0～255中的某一整数，又把该整数称为优先数。确定进程优先级大小的依据有如下三个：

(1) 进程类型

(2) 进程对资源的需求

(3) 用户要求

- 对资源要求少的进程应赋予较高的优先级

2. 优先级的类型

1) 静态优先级

静态优先级是在创建进程时确定的，在进程的整个运行期间保持不变。优先级是利用某一范围内的一个整数来表示的，例如0～255中的某一整数，又把该整数称为优先数。确定进程优先级大小的依据有如下三个：

(1) 进程类型

(2) 进程对资源的需求

(3) 用户要求

- 根据进程的紧迫程度及用户所付费用的多少确定优先级

2) 动态优先级

动态优先级是指在创建进程之初，先赋予其一个优先级，然后其值随进程的推进或等待时间的增加而改变，以便获得更好的调度性能。

- 若所有进程都具有相同的优先级初值，则最先进入就绪队列的进程会引起优先级变得最高，而优先获得处理机，这相当于FCFS算法；
- 若所有的就绪进程具有各不相同的优先级初值，对于优先级初值较低的进程，在等待了足够的时间后，也可以获得处理机；
- 当采用抢占式调度方式时，若再规定当前进程的优先级随运行时间的推移而下降，则可以防止一个长作业长期地垄断处理机。

3.3.4　多队列调度算法

前面的各种调度算法，尤其在应用于进程调度时，由于系统中仅设置一个进程的就绪队列，无法满足系统中不同用户对进程调度策略的不同要求，在多处理机系统中，这种单一调度策略实现机制的缺点更显突出，由此，多级队列调度算法能够在一定程度上弥补这一缺点。

该算法将系统中的进程就绪队列从一个拆分为若干个，将不同类型或性质的进程固定分配在不同的就绪队列，不同的就绪队列采用不同的调度算法，一个就绪队列中的进程可以设置不同的优先级，不同的就绪队列本身也可以设置不同的优先级。

3.3.4　多队列调度算法

前面的各种调度算法，尤其在应用于进程调度时，由于

- 多处理机系统中，由于多队列调度算法安排了多个就绪队列，因此很方便为每个处理机设置一个单独的就绪队列。
- 不仅每个处理机的调度可以实施各自不同的调度策略，而且对于一个含有多个线程的进程而言，可以根据其要求将所有线程分配在一个就绪队列，全部在一个处理机上运行。
- 对于一组需要相互合作的进程或线程而言，也可以将它们分配到一组处理机所对应的多个就绪队列，使得它们能同时获得处理机并行执行。

的优先级。

3.3.5 多级反馈队列(multileved feedback queue)调度算法

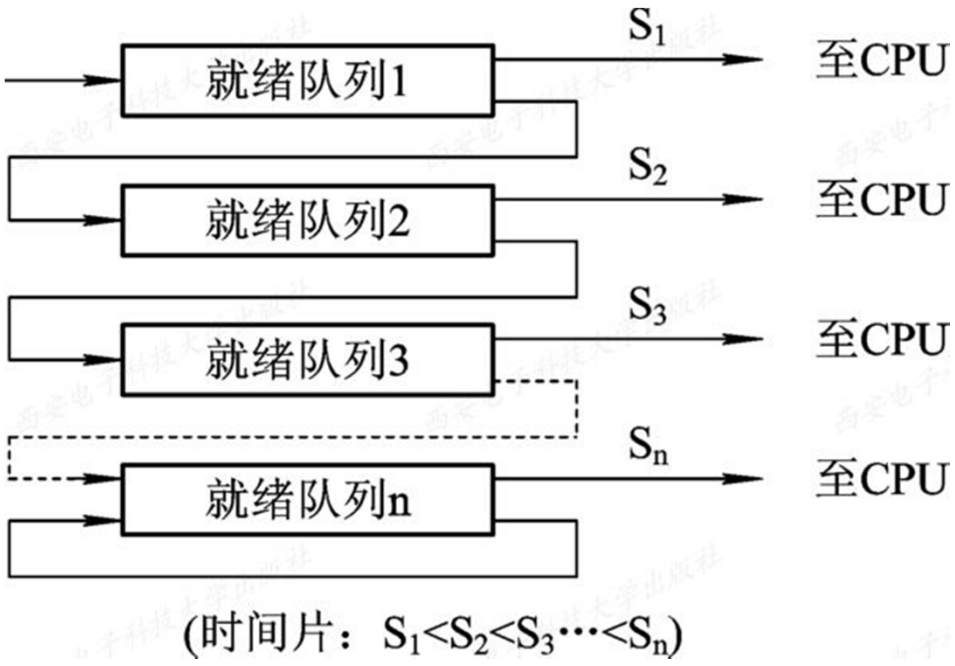
前面介绍的进程调度算法都有一定的局限性，如果未指明进程长度，则短进程优先和基于进程长度的抢占式调度算法都将无法使用。

多级反馈队列调度算法不必事先知道各种进程所需的执行时间，还可以较好地满足各种类型进程的需要，是公认的一种较好的进程调度算法。

1. 调度机制

多级反馈队列调度算法的调度机制可描述如下：

(1) 设置多个就绪队列，并为每个队列赋予不同的优先级。



- 第一个队列的优先级最高，第二个次之，其余队列优先级逐个降低
- 优先级越高的队列中，时间片越小，第i+1个队列的时间片要比第i个的时间片长一倍

(2) 每个队列都采用FCFS算法。

当新进程进入内存后，首先将它放入第一队列的末尾，按FCFS原则等待调度。

当轮到该进程执行时，如它能在该时间片内完成，便可撤离系统。否则，即它在一个时间片结束时尚未完成，调度程序将其转入第二队列的末尾等待调度；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，……，依此类推。当进程最后被降到第n队列后，在第n队列中便采取按RR方式运行。

(3) 按队列优先级调度。

调度程序首先调度最高优先级队列中的诸进程运行，仅当第一队列空闲时才调度第二队列中的进程运行；换言之，仅当第1～(i-1)所有队列均空时，才会调度第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任一优先级较高的队列，此时须立即把正在运行的进程放回到第i队列的末尾，而把处理机分配给新到的高优先级进程。

2. 调度算法的性能

在多级反馈队列调度算法中，如果规定第一个队列的时间片略大于多数人机交互所需之处理时间时，便能较好地满足各种类型用户的需要。

(1) 终端型用户

(2) 短批处理作业用户

(3) 长批处理作业用户

- 终端型用户提交的作业多属于交互型作业，通常较小，系统只要能使这些作业在第一队列规定的时间片内完成，便可使终端型用户感到满意。

2. 调度算法的性能

在多级反馈队列调度算法中，如果规定第一个队列的时间片略大于多数人机交互所需之处理时间时，便能较好地满足各种类型用户的需要。

(1) 终端型用户

(2) 短批处理作业用户

(3) 长批处理作业用户

- 这类作业，如果能在第一队列中执行完成，便获得与终端型作业一样的响应时间。
- 对于稍长的短作业，也只需在第二和第三队列中各执行一时间片完成，其周转时间仍然较短

2. 调度算法的性能

在多级反馈队列调度算法中，如果规定第一个队列的时间片略大于多数人机交互所需之处理时间时，便能较好地满足各种类型用户的需要。

(1) 终端型用户

(2) 短批处理作业用户

(3) 长批处理作业用户

- 长作业，将依次在第1，2，…，n个队列中运行，然后再按轮转方式运行。用户不必担心其作业长期得不到处理。

3.3.6　基于公平原则的调度算法

以上几种调度算法仅保证优先运行，并不保证作业占用了多少处理机时间，也未考虑到调度的公平性。

1. 保证调度算法

保证调度算法是另外一种类型的调度算法，它向用户所做出的保证并不是优先运行，而是明确的性能保证，该算法可以做到调度的公平性。

一种比较容易实现的性能保证是处理机分配的公平性。如果在系统中有n个相同类型的进程同时运行，为公平起见，须保证每个进程都获得相同的处理机时间1/n。

在实施公平调度算法时系统中必须具有这样一些功能：

(1) 跟踪计算每个进程自创建以来已经执行的处理时间。

(2) 计算每个进程应获得的处理机时间，即自创建以来的时间除以n。

(3) 计算进程获得处理机时间的比率，即进程实际执行的处理时间和应获得的处理机时间之比。

(4) 比较各进程获得处理机时间的比率。如进程A的比率最低，为0.5，而进程B的比率为0.8，进程C的比率为1.2等。

(5) 调度程序应选择比率最小的进程将处理机分配给它，并让该进程一直运行，直到超过最接近它的进程比率为止。

2. 公平分享调度算法

分配给每个进程相同的处理机时间，显然，这对诸进程而言，是体现了一定程度的公平，但如果各个用户所拥有的进程数不同，就会发生对用户的不公平问题。

例如，系统中仅有2个用户，用户1启动了4个进程，用户2只启动1个进程。若采用轮转法让每个进程轮流运行一个时间片，对进程而言很公平，但用户1和用户2得到的处理机时间分别为80%和20%，对于用户2而言显然有失公平。

2. 公平分享调度算法

分配给每个进程相同的处理机时间，显然，这对诸进程

公平分享调度算法：

- 调度的公平性主要是针对用户而言，使所有用户能获得相同的处理机时间，或所要求的的时间比例。
- 由于调度是以进程为基本单位，因此必须考虑每个用户所拥有的进程数目。

片，对进程而言很公平，但用户1和用户2得到的处理机时间分别为80%和20%，对于用户2而言显然有失公平。

例如，系统中有两个用户，用户1有4个进程A、B、C、D，用户2只有1个进程E。

为保证两个用户能获得相同的处理机时间，则必须执行如下所示的强制调度序列：

A E B E C E D E A E B E C E D E ……

如果希望用户1所获得的的处理机时间是用户2的两倍，则必须执行如下所示的强制调度序列：

A B E C D E A B E C D E ……

练习

1．假设下述四个作业同时到达，当使用最高优先数优先调度算法时，作业的平均周转时间为____D____小时。

作业	所需运行时间	优先数
1	2	4
2	5	9
3	8	1
4	3	8

A．4.5　　B．10.5　　C．4.75　　D．10.25

思考

2. 已知4个作业的提交时间和运行时间如下：

作业	提交时间	运行时间
1	8.00	2.50
2	8.20	1.20
3	8.30	0.30
4	9.00	0.50

用先来先服务和短作业优先调度算法进行调度，计算在每一种调度算法下的平均周转时间和平均带权周转时间。并说明哪一种算法的调度性能更好些。