处理器调度

Linux
Android
Linux
OpenStack
Mac OS
Windows

处理器调度

本讲内容

1. 处理器调度的层次
2. 处理器调度的算法
3. 单道环境下的调度
4. 多道环境下的调度
5. 低级调度方式算法

调度的基本概念

调度的基本概念


BANK
VIP
普通客户先到先服务
VIP客户可优先被服务

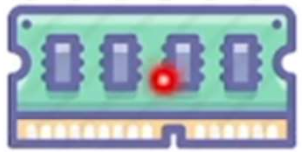

3
1
10
3
③
①
④
②
时间短的优先
时间相同的先排队先使用

调度的基本概念

当有一堆任务要处理，但由于资源有限，这些事情没法同时处理。这就需要确定某种规则来决定处理这些任务的顺序，这就是“调度”研究的问题。

在多道程序系统中，进程的数量往往是多于处理机的个数的，这样不可能同时并行地处理各个进程。处理机调度，就是从就绪队列中按照一定的算法选择一个进程并将处理机分配给它运行，以实现进程的并发执行。

2 作业与进程



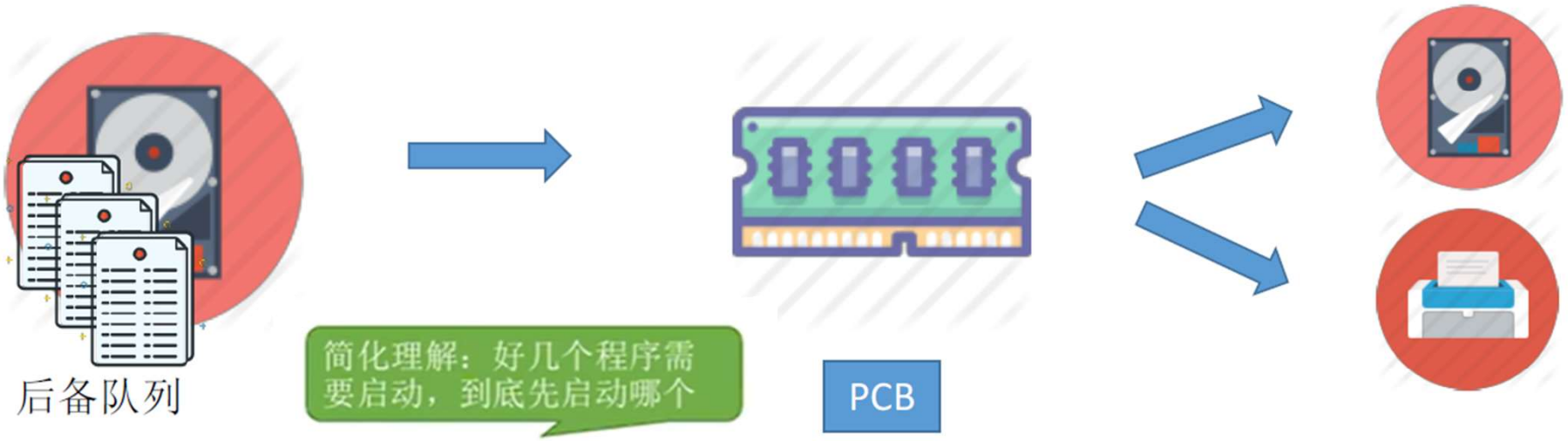
内存空间有限，有时无法将用户提交的作业全部放入内存

作业: 一个具体的任务，系统从外存挑选一个程序，把它放入内存中运行

用户向系统提交一个作业 ≈ 用户让操作系统启动一个程序(来处理一个具体的任务)

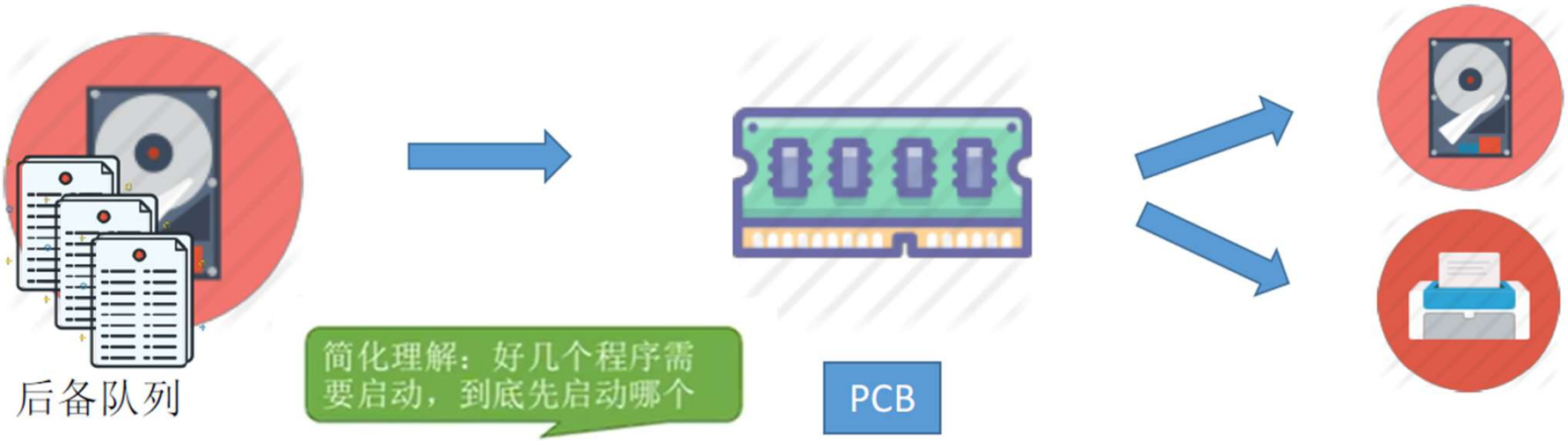
- 作业是用户向计算机提交的任务实体，而进程则是完成任务的执行实体，是系统分配资源的基本单位
- 一个作业可由多个进程组成，且必须至少由一个进程组成，反过来不成立
- Linux等分时系统中，并不强调作业调度和进程调度的区别

调度的三个层次——高级调度



由于内存空间有限，有时无法将用户提交的作业全部放入内存，因此就需要确定某种规则来决定将作业调入内存的顺序。

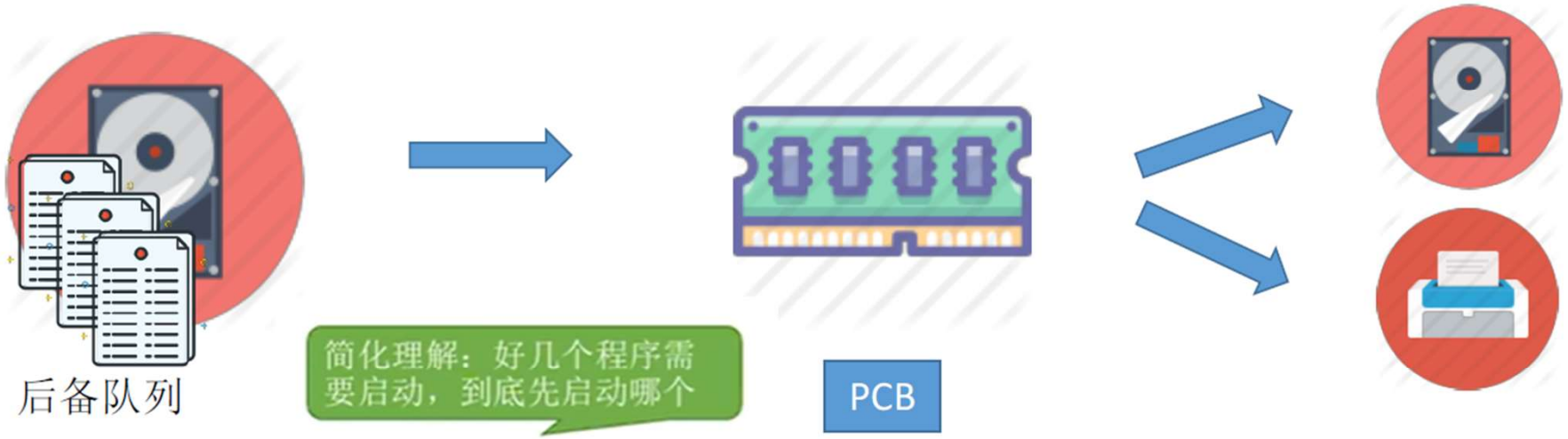
调度的三个层次——高级调度



由于内存空间有限，有时无法将用户提交的作业全部放入内存，因此就需要确定某种规则来决定将作业调入内存的顺序。

高级调度（作业调度）。按一定的原则从外存上处于后备队列的作业中挑选一个（或多个）作业，给他们分配内存等必要资源，并建立相应的进程（建立PCB），以使它（们）获得竞争处理机的权利。

调度的三个层次——高级调度



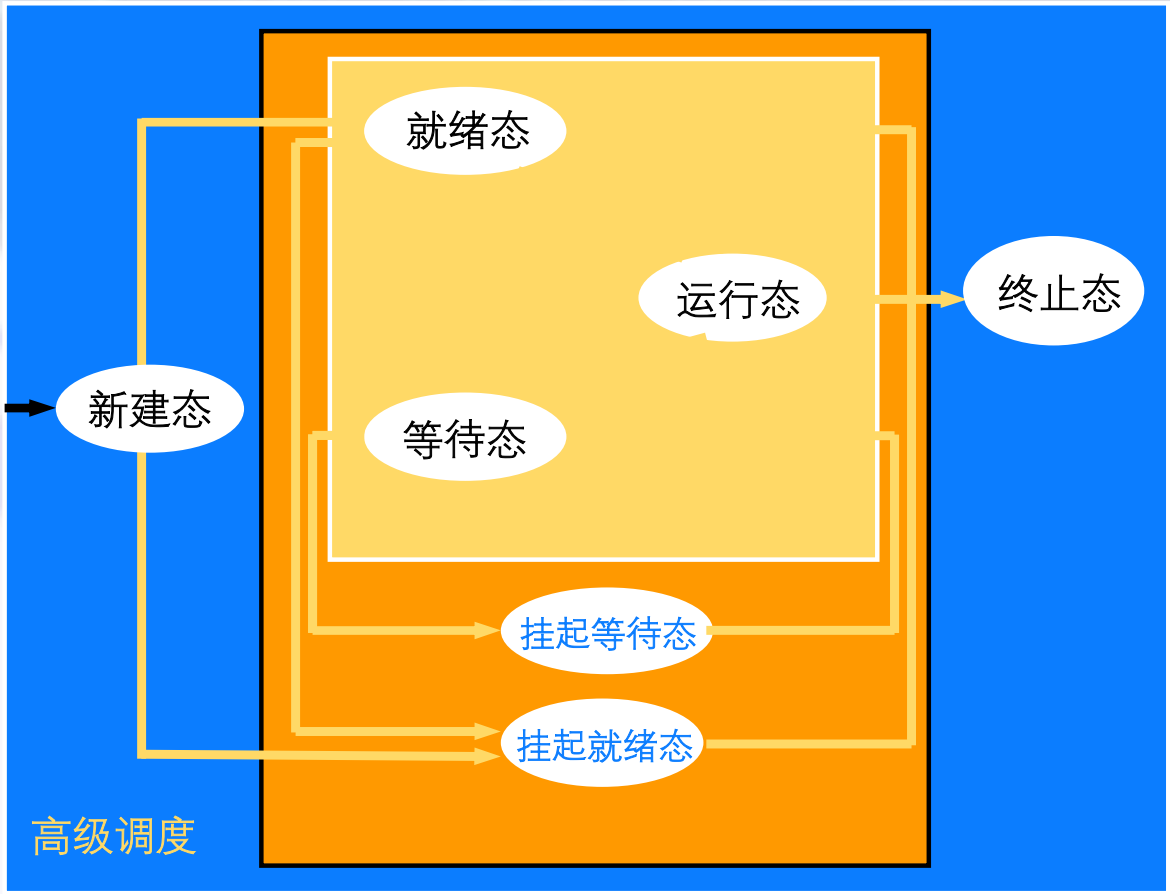
由于内存空间有限，有时无法将用户提交的作业全部放入内存，因此就需要确定某种规则来决定将作业调入内存的顺序。

高级调度（作业调度）。按一定的原则从外存上处于后备队列的作业中挑选一个（或多个）作业，给他们分配内存等必要资源，并建立相应的进程（建立PCB），以使它（们）获得竞争处理机的权利。

高级调度是辅存（外存）与内存之间的调度。每个作业只调入一次，调出一次。作业调入时会建立相应的PCB，作业调出时才撤销PCB。高级调度主要是指调入的问题，因为只有调入的时机需要操作系统来确定，但调出的时机必然是作业运行结束才调出。

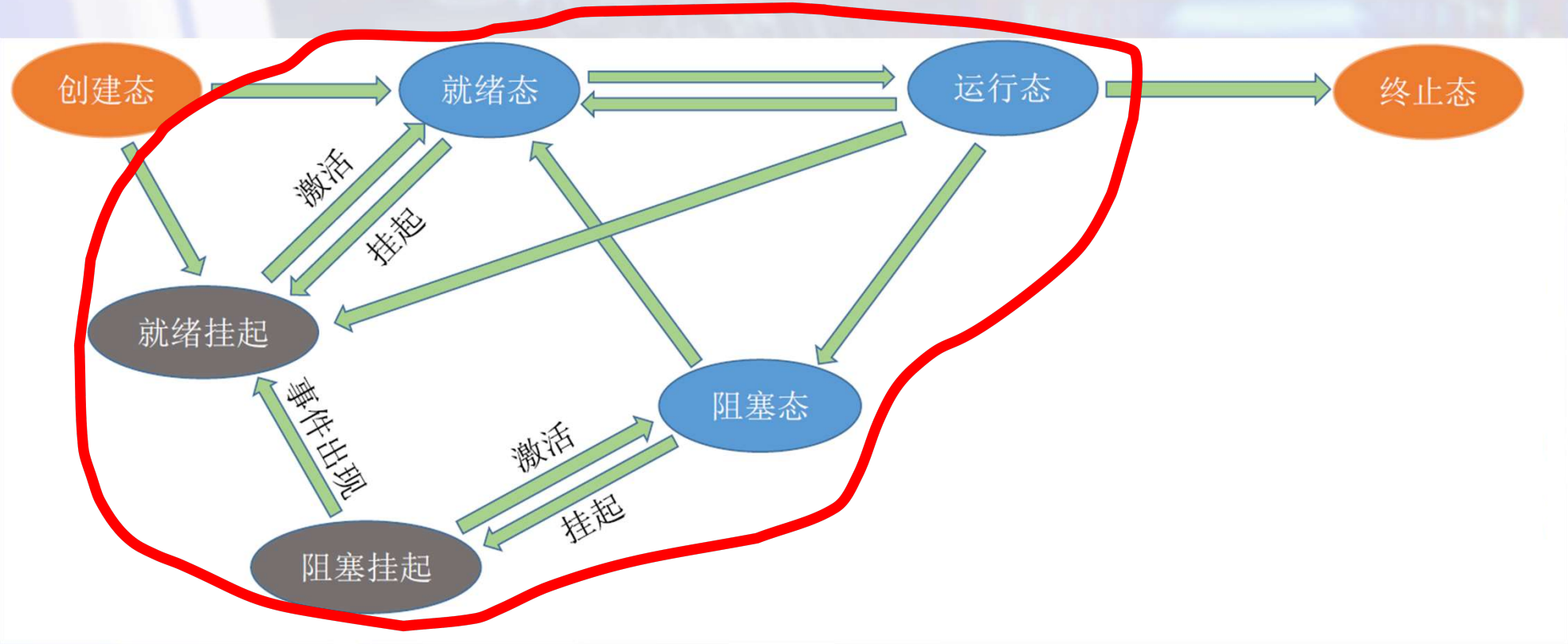
处理器调度的层次

1 调度层次 高级调度: 外存-内存

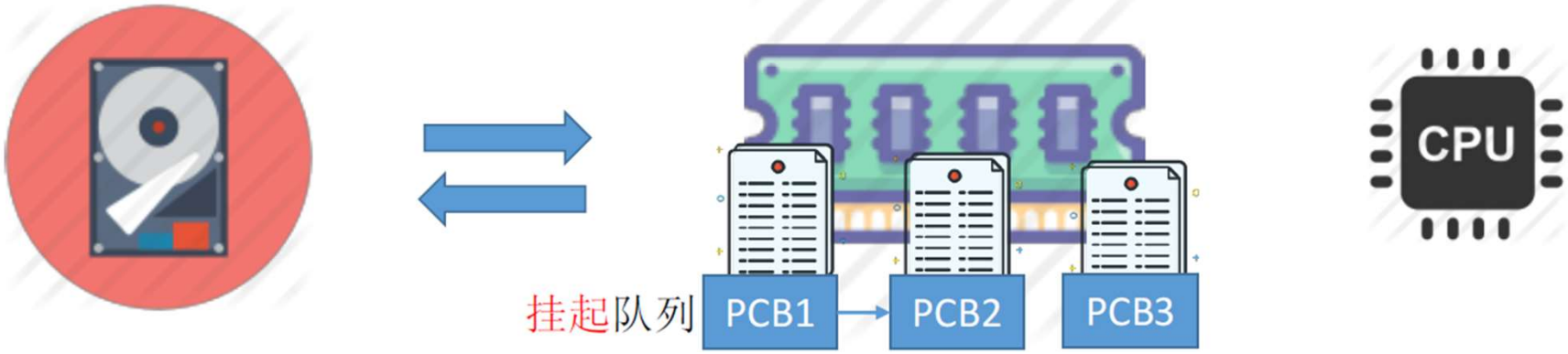


处理器调度的层次

1 调度层次 高级调度: 外存-内存



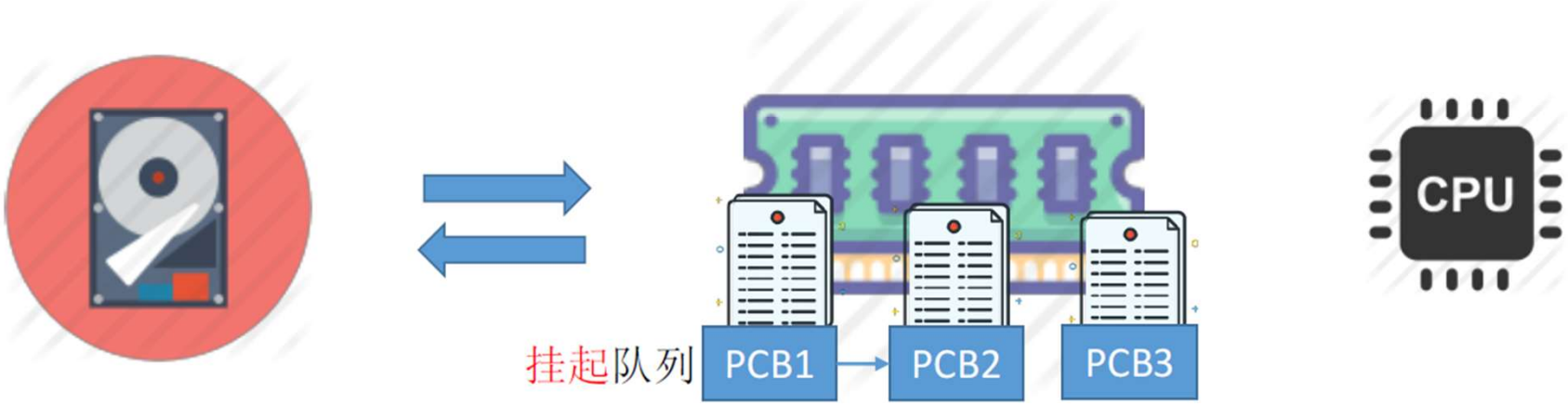
调度的三个层次——中级调度



引入了虚拟存储技术之后，可将暂时不能运行的进程调至外存等待。等它重新具备了运行条件且内存又稍有空闲时，再重新调入内存。

这么做的目的是为了提高内存利用率和系统吞吐量。

调度的三个层次——中级调度



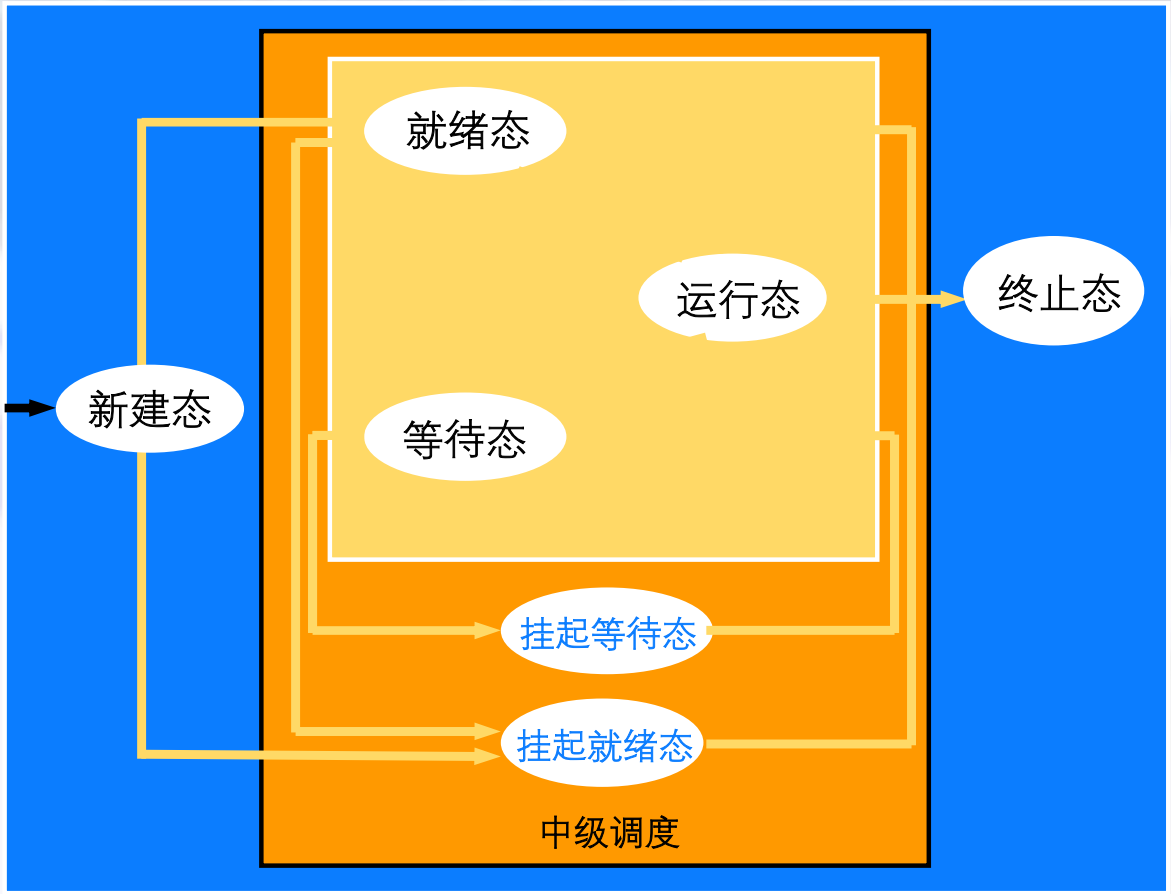
引入了虚拟存储技术之后，可将暂时不能运行的进程调至外存等待。等它重新具备了运行条件且内存又稍有空闲时，再重新调入内存。
这么做的目的是为了提高内存利用率和系统吞吐量。
暂时调到外存等待的进程状态为挂起状态。值得注意的是，PCB并不会一起调到外存，而是会常驻内存。PCB中会记录进程数据在外存中的存放位置，进程状态等信息，操作系统通过内存中的PCB来保持对各个进程的监控、管理。被挂起的进程PCB会被放到的挂起队列中。

中级调度（内存调度），就是要决定将哪个处于挂起状态的进程重新调入内存。
一个进程可能会被多次调出、调入内存，因此中级调度发生的频率要比高级调度更高。

例如：手机切换程序时候的卡顿

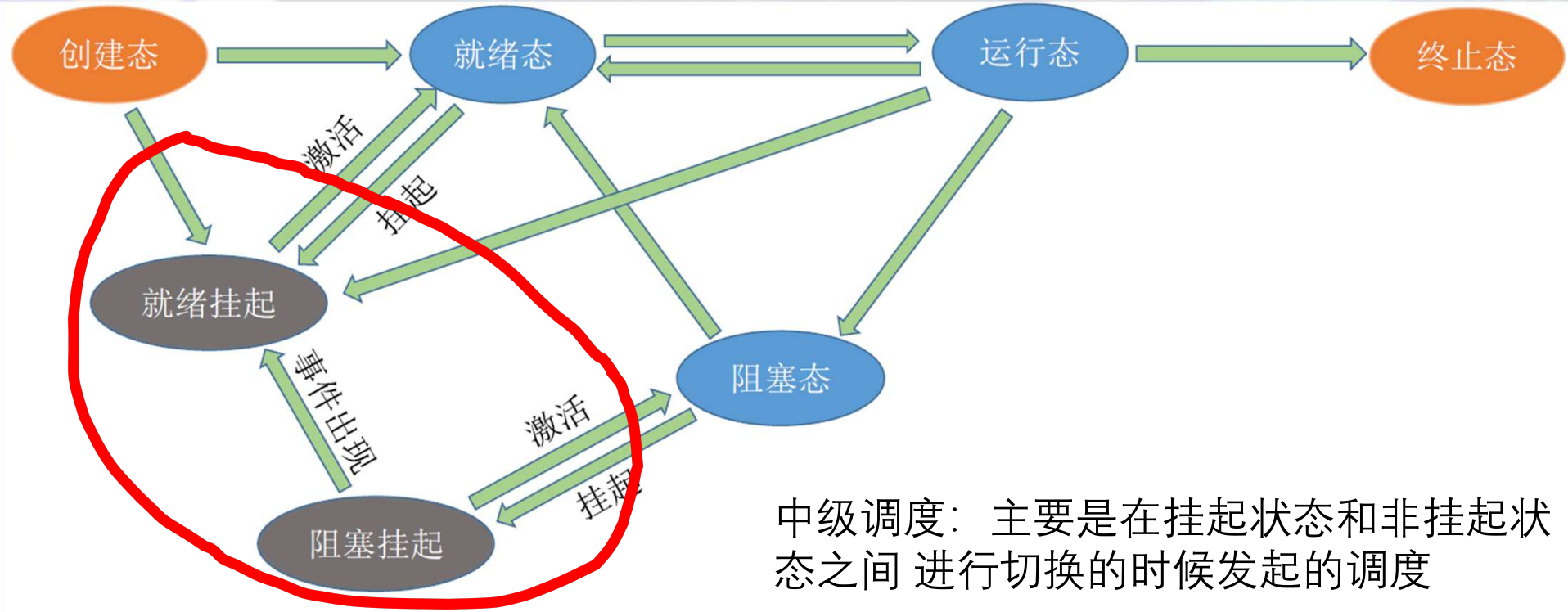
处理器调度的层次

1 调度层次 中级调度：内存-外存



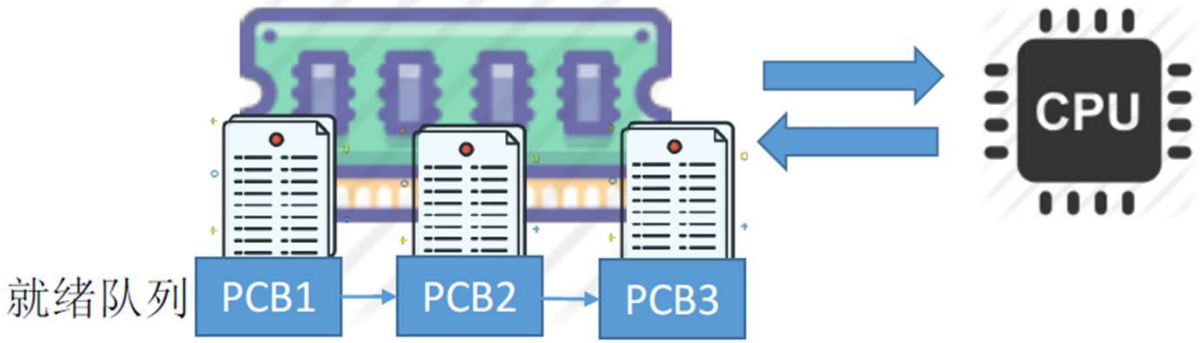
处理器调度的层次

1 调度层次



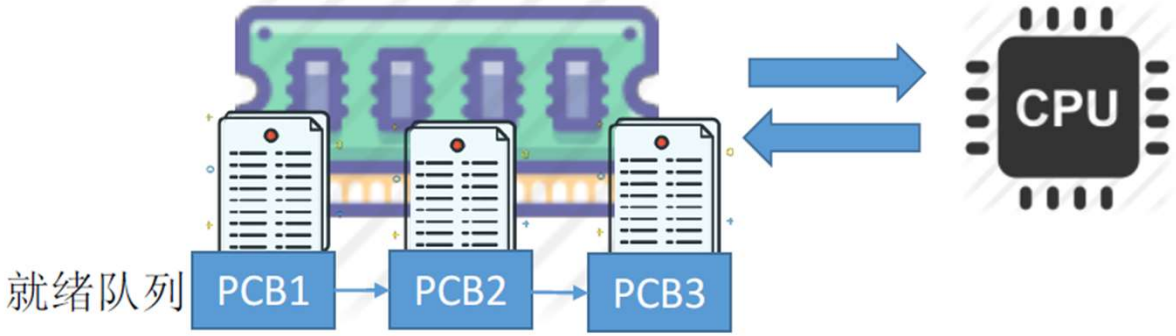
中级调度：主要是在挂起状态和非挂起状态之间 进行切换的时候发起的调度

调度的三个层次——低级调度



低级调度（进程调度），其主要任务是按照某种方法和策略从就绪队列中选取一个进程，将处理机分配给它。

调度的三个层次——低级调度

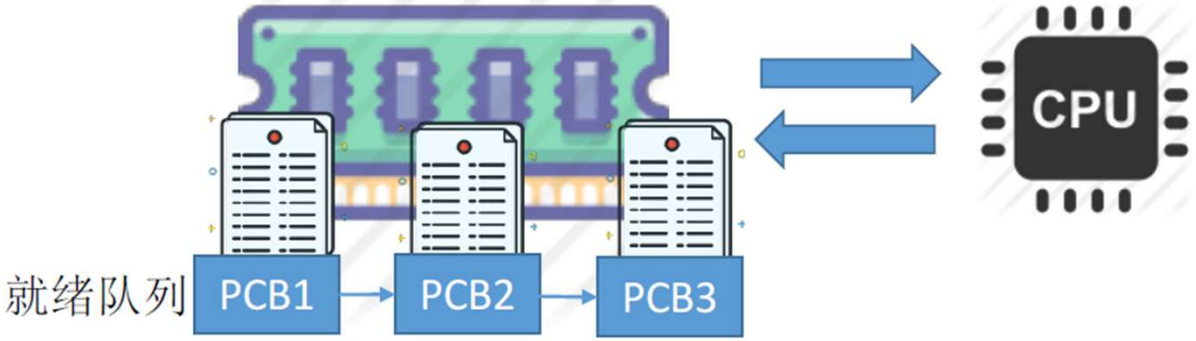


低级调度（进程调度），其主要任务是按照某种方法和策略从就绪队列中选取一个进程，将处理机分配给它。

进程调度是操作系统中最基本的一种调度，在一般的操作系统中都必须配置进程调度。

进程调度的频率很高，一般几十毫秒一次。

调度的三个层次——低级调度



低级调度（进程调度），其主要任务是按照某种方法和策略从就绪队列中选取一个进程，将处理机分配给它。

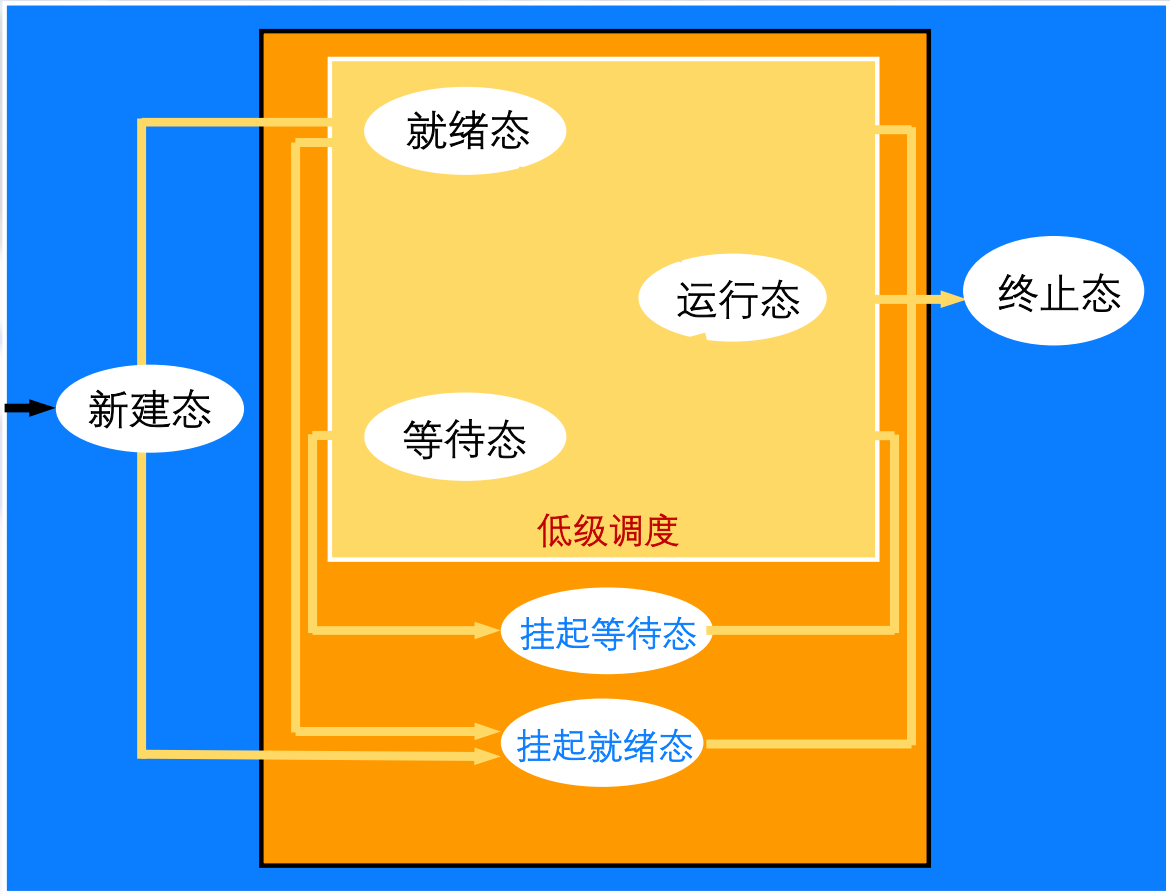
进程调度是操作系统中最基本的一种调度，在一般的操作系统中都必须配置进程调度。

进程调度的频率很高，一般几十毫秒一次。

低级调度(处理机调度)，使得各个进程很快速的轮流上CPU上执行，从宏观上看各个进程是同时在执行。

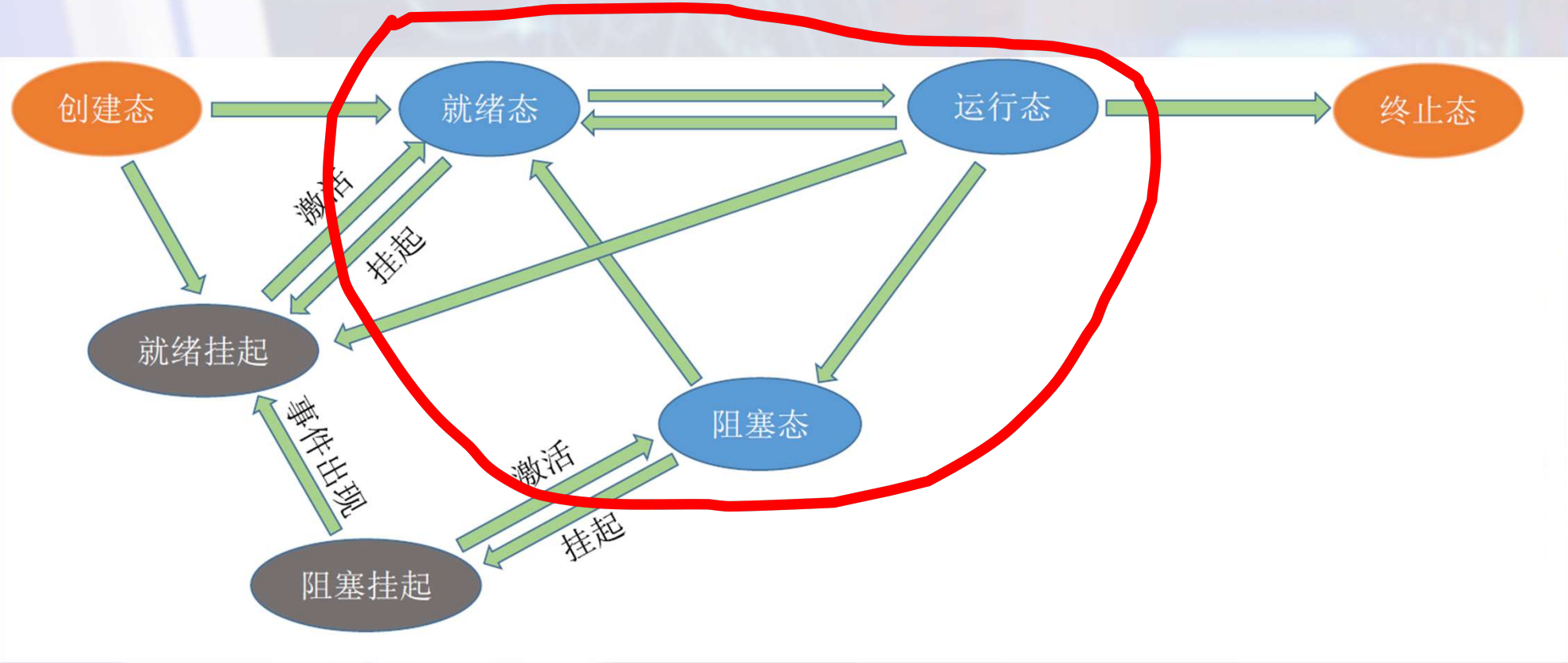
处理器调度的层次

1 调度层次 低级调度：内存-CPU



处理器调度的层次

1 调度层次 低级调度：内存-CPU



三层调度的联系、对比

	要做什么	调度发生在..	发生频率	对进程状态的影响
高级调度（作业调度）	按照某种规则，从后备队列中选择合适的作业将其调入内存，并为其创建进程	外存→内存（面向作业）	最低	无→创建态→就绪态
中级调度（内存调度）	按照某种规则，从挂起队列中选择合适的进程将其数据调回内存	外存→内存（面向进程）	中等	挂起态→就绪态（阻塞挂起→阻塞态）
低级调度（进程调度）	按照某种规则，从就绪队列中选择一个进程为其分配处理机	内存→CPU	最高	就绪态→运行态

处理器调度

本讲内容

1. 处理器调度的层次
2. 处理器调度的算法
3. 单道环境下的调度
4. 多道环境下的调度
5. 低级调度方式算法

处理器调度的算法

2 调度原则

- 合理性：既要保证系统实现特殊功能要求，同时要对各个任务合理地分配到处理器份额

- 有效性：处理器、内存和I/O设备得到合理有效的分配，使系统资源得到充分的利用

处理器调度的算法

3 理想目标

- 单位时间内运行尽可能多的作业
- 使处理器尽可能保持“忙碌”
- 响应时间和周转时间能够尽可能短
- 使各种I/O设备得以充分利用
- 对所有的作业都是公平合理的

处理器调度的算法

4 设计理念

- 调度算法应与系统设计目标保持一致
- 注意系统资源均衡使用
- 保证提交的作业在截止时间内完成
- 设法缩短作业平均周转时间

大多数操作系统都采用比较简单的调度算法

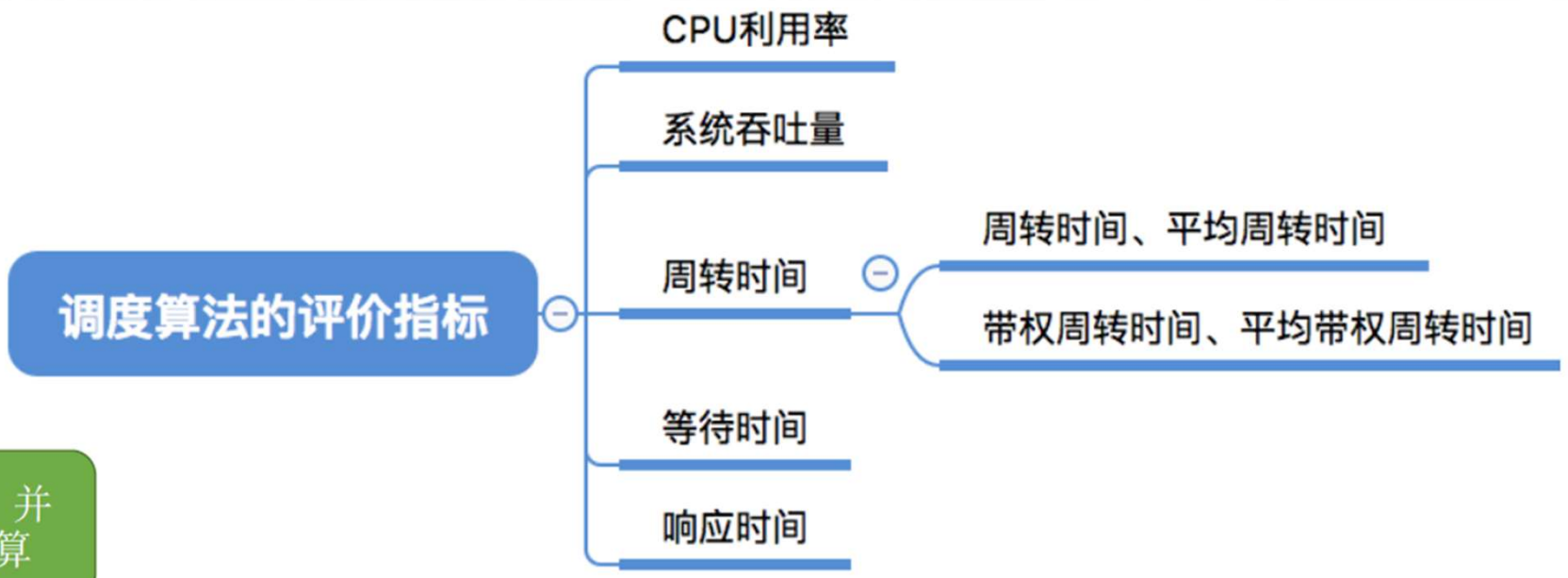
处理器调度的算法

5 调度决策因素

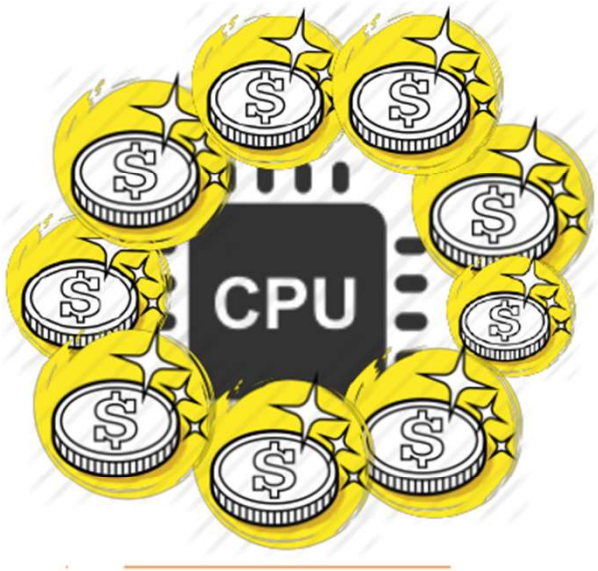
- 作业到达时间
- 预先为作业确定的优先级
- 作业所需的CPU时间
- 存储要求
- 其他的资源要求

处理器调度的算法

6 性能衡量主要指标



CPU利用率

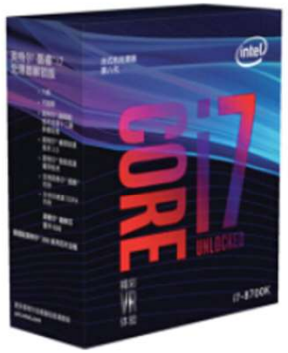


由于早期的CPU造价极其昂贵，因此人们会希望让CPU尽可能多地工作

CPU利用率：指CPU “忙碌”的时间占总时间的比例。

$$\text{利用率} = \frac{\text{忙碌的时间}}{\text{总时间}}$$

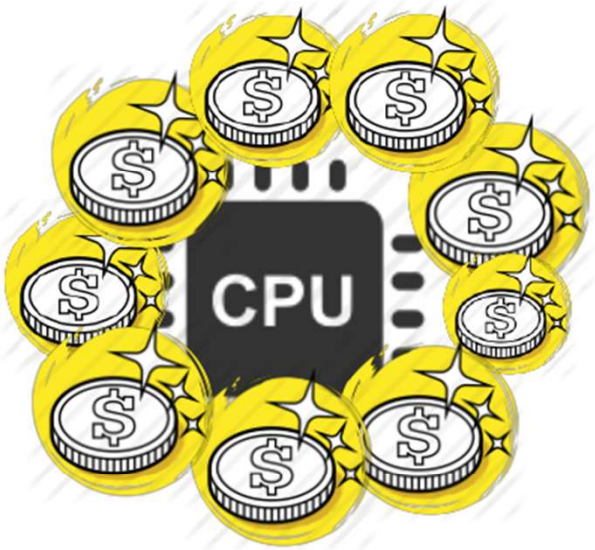
有的题目还会要求计算某种设备的利用率



¥2699.00

英特尔（Intel） i7 8700K 酷睿六核 盒装
CPU处理器 【8器降临,芯有灵7】

CPU利用率



由于早期的CPU造价极其昂贵，因此人们会希望让CPU尽可能多地工作

CPU利用率：指CPU “忙碌”的时间占总时间的比例。

$$\text{利用率} = \frac{\text{忙碌的时间}}{\text{总时间}}$$

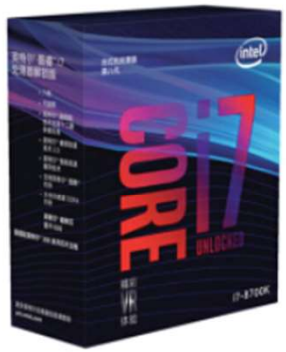
有的题目还会要求计算某种设备的利用率

Eg：某计算机只支持单道程序，某个作业刚开始需要在CPU上运行5秒，再用打印机打印输出5秒，之后再执行5秒，才能结束。在此过程中，CPU利用率、打印机利用率分别是多少？

$$\text{CPU利用率} = \frac{5+5}{5+5+5} = 66.66\%$$

$$\text{打印机利用率} = \frac{5}{15} = 33.33\%$$

通常会考察多道程序并发执行的情况，可以用“甘特图”来辅助计算



¥2699.00

英特尔（Intel） i7 8700K 酷睿六核 盒装

CPU处理器 【8器降临,芯有灵7】

系统吞吐量

对于计算机来说，希望能用尽可能少的时间处理完尽可能多的作业

系统吞吐量：单位时间内完成作业的数量


无比希望的眼神

$$\text{系统吞吐量} = \frac{\text{总共完成了多少道作业}}{\text{总共花了多少时间}}$$

Eg：某计算机系统处理完10道作业，共花费100秒，则系统吞吐量为？

10/100 = 0.1 道/秒

周转时间

对于计算机的用户来说，他很关心自己的作业从提交到完成花了多少时间。
周转时间，是指从作业被提交给系统开始，到作业完成为止的这段时间间隔。
它包括四个部分：作业在外存后备队列上等待作业调度（高级调度）的时间、进程在就绪队列上等待进程调度（低级调度）的时间、进程在CPU上执行的时间、进程等待I/O操作完成的时间。后三项在一个作业的整个处理过程中，可能发生多次。

周转时间

对于计算机的用户来说，他很关心自己的作业从提交到完成花了多少时间。
周转时间，是指从作业被提交给系统开始，到作业完成为止的这段时间间隔。
它包括四个部分：作业在外存后备队列上等待作业调度（高级调度）的时间、进程在就绪队列上等待进程调度（低级调度）的时间、进程在CPU上执行的时间、进程等待I/O操作完成的时间。后三项在一个作业的整个处理过程中，可能发生多次。

（作业）周转时间 = 作业完成时间 – 作业提交时间

对于用户来说，更关心自己的单个作业的周转时间

$$\text{平均周转时间} = \frac{\text{各作业周转时间之和}}{\text{作业数}}$$

对于操作系统来说，更关心系统的整体表现，因此更关心所有作业周转时间的平均值

处理器调度的算法

6 性能衡量主要指标

作业平均周转时间

假定有*n*个作业，其中作业*i*进入系统时间为*Si*，它被选中执行，得到结果的时间为*Ei*，其周转时间为

$$Ti = Ei - Si$$

则这批作业平均周转时间为：

$$T = (\sum_{i=1}^{n} Ti) \times \frac{1}{n}$$

各个作业的周转时间之和 / 总的作业数量

周转时间

对于计算机的用户来说，他很关心自己的作业从提交到完成花了多少时间。
周转时间，是指从作业被提交给系统开始，到作业完成为止的这段时间间隔。
它包括四个部分：作业在外存后备队列上等待作业调度（高级调度）的时间、进程在就绪队列上等待进程调度（低级调度）的时间、进程在CPU上执行的时间、进程等待I/O操作完成的时间。后三项在一个作业的整个处理过程中，可能发生多次。

（作业）周转时间 = 作业完成时间 – 作业提交时间

对于用户来说，更关心自己的单个作业的周转时间

$$\text{平均周转时间} = \frac{\text{各作业周转时间之和}}{\text{作业数}}$$

对于操作系统来说，更关心系统的整体表现，因此更关心所有作业周转时间的平均值

思考：有的作业运行时间短，有的作业运行时间长，因此在周转时间相同的情况下，运行时间不同的作业，给用户的感觉肯定是不一样的

一个有味道的例子：排队等厕所

周转时间

$$\text{带权周转时间} = \frac{\text{作业周转时间}}{\text{作业实际运行的时间}} = \frac{\text{作业完成时间} - \text{作业提交时间}}{\text{作业实际运行的时间}}$$

带权周转时间必然 ≥ 1

带权周转时间与周转时间都是越小越好

$$\text{平均带权周转时间} = \frac{\text{各作业带权周转时间之和}}{\text{作业数}}$$

对于周转时间相同的两个作业，实际运行时间长的作业在相同时间内被服务的时间更多，带权周转时间更小，用户满意度更高。

对于实际运行时间相同的两个作业，周转时间短的带权周转时间更小，用户满意度更高。

处理器调度的算法

6 性能衡量主要指标

平均带权周转时间

ri 为作业i的实际在CPU上的执行时间

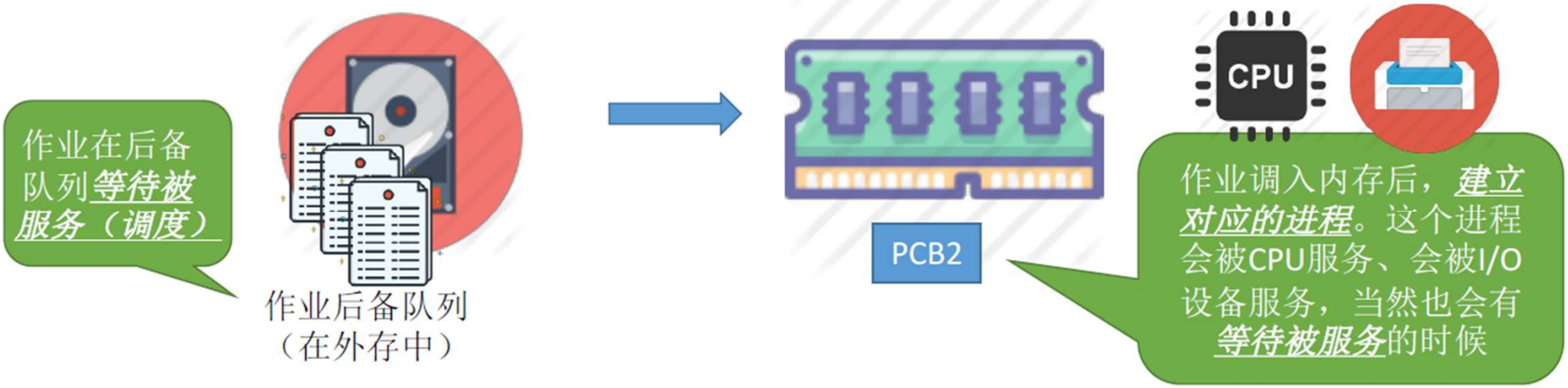
$$W=(\sum_{i=1}^{n}\frac{Ti}{ri})\times\frac{1}{n}$$

平均周转时间：衡量不同调度算法对同一个作业流的性能

平均带权周转时间：同一调度算法对不同作业流的性能衡量

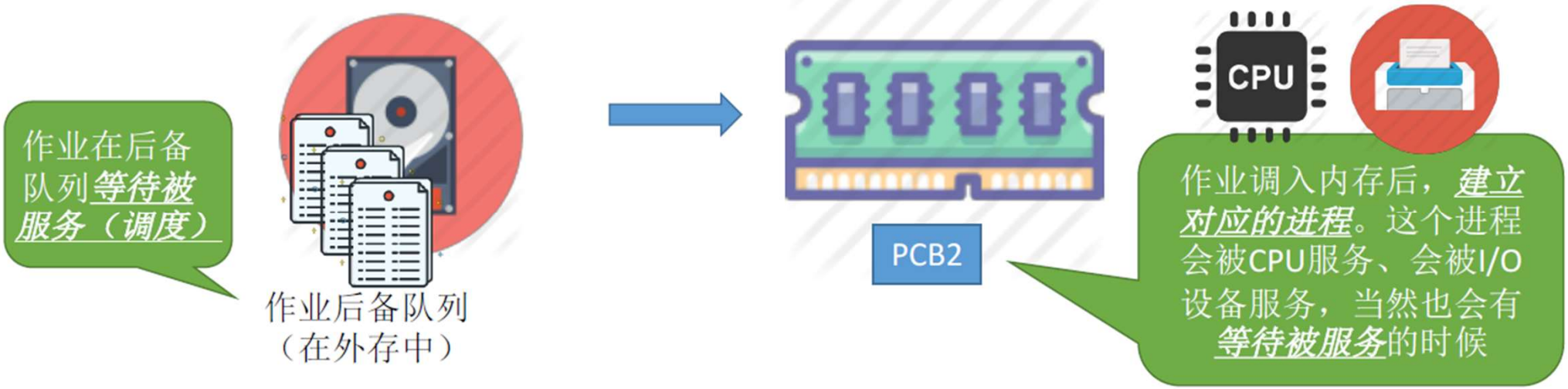
等待时间

计算机的用户希望自己的作业尽可能少的等待处理机
等待时间，指进程/作业处于等待处理机状态时间之和，等待时间越长，用户满意度越低。



等待时间

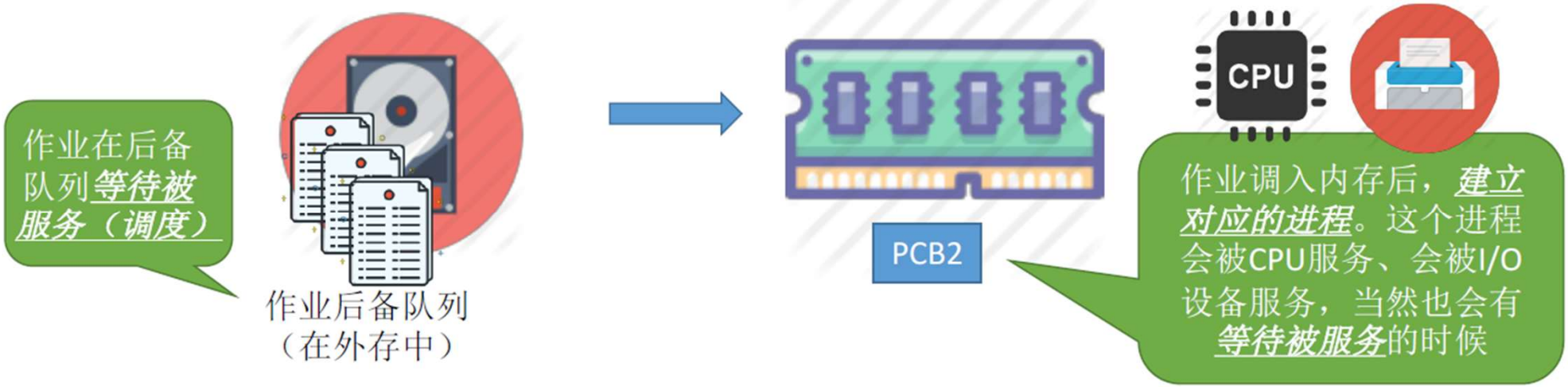
计算机的用户希望自己的作业尽可能少的等待处理机
等待时间，指进程/作业处于等待处理机状态时间之和，等待时间越长，用户满意度越低。



对于进程来说，等待时间就是指进程建立后等待被服务的时间之和，在等待I/O完成的期间其实进程也是在被服务的，所以不计入等待时间。
对于作业来说，不仅要考虑建立进程后的等待时间，还要加上作业在外存后备队列中等待的时间。

等待时间

计算机的用户希望自己的作业尽可能少的等待处理机
等待时间，指进程/作业处于等待处理机状态时间之和，等待时间越长，用户满意度越低。



对于进程来说，等待时间就是指进程建立后等待被服务的时间之和，在等待I/O完成的期间其实进程也是在被服务的，所以不计入等待时间。
对于作业来说，不仅要考虑建立进程后的等待时间，还要加上作业在外存后备队列中等待的时间。

一个作业总共需要被CPU服务多久，被I/O设备服务多久一般是确定不变的，因此调度算法其实只会影响作业/进程的等待时间。当然，与前面指标类似，也有“平均等待时间”来评价整体性能。

- 作业周转时间考虑了整个作业经历的所有时间，而作业等待时间仅关注作业未被执行的时长

响应时间

对于计算机用户来说，会希望自己的提交的请求（比如通过键盘输入了一个调试命令）尽早地开始被系统服务、回应。

响应时间，指从用户提交请求到首次产生响应所用的时间。

$$\text{响应比} = \frac{\text{作业周转时间}}{\text{作业处理时间}}$$

$$= \frac{\text{作业处理时间} + \text{作业等待时间}}{\text{作业处理时间}}$$

$$= 1 + \frac{\text{作业等待时间}}{\text{作业处理时间}}$$

处理器调度的算法

7 典型算法

先来先服务算法FCFS

以作业到达系统的时间为唯一的参数决定谁先谁后

最短作业优先算法SJF

只考虑进程的CPU执行时间，计算量越小的作业就会优先得到服务，减少等待时间，就减少了整体的周转时间

最短剩余时间优先算法SRTF

SJF的变形：剩余的计算时间最短的作业被调度，非抢占式SJF变成抢占式调度算法，来了一个剩余时间更短的进程就会抢占

处理器调度的算法

7 典型算法

- 先来先服务算法FCFS
- 最短作业优先算法SJF
- 最短剩余时间优先算法SRTF
- 最高响应比优先算法HRN

以上三个算法只考虑某一个角度，不够公平

对FCFS和SJF两种算法的思考...

FCFS 算法是在每次调度的时候选择一个等待时间最长的作业（进程）为其服务。但是没有考虑到作业的运行时间，因此导致了对短作业不友好的问题

SJF 算法是选择一个执行时间最短的作业为其服务。但是又完全不考虑各个作业的等待时间，因此导致了对长作业不友好的问题，甚至还会造成饥饿问题

能不能设计一个算法，即考虑到各个作业的等待时间，也能兼顾运行时间呢？

厉害了，我的哥

高响应比优先算法

处理器调度的算法

7 典型算法

- 先来先服务算法FCFS
- 最短作业优先算法SJF
- 最短剩余时间优先算法SRTF

以上三个算法只考虑某一个角度，不够公平

- 最高响应比优先算法HRN

响应比 = 作业周转时间 / 作业处理时间 = （作业处理时间+作业等待时间）/ 作业处理时间

= 1 + （作业等待时间 / 作业处理时间）

响应比受制于： 作业的等待时间 和 作业处理时间， 两方面的制约

处理器调度的算法

7 典型算法

最高响应比优先算法HRN

HRN既考虑了等待时间，又考虑了作业在CPU上的运行时间，公平

响应比 = 作业周转时间 / 作业处理时间 =（作业处理时间+作业等待时间）/ 作业处理时间

= 1 +（作业等待时间 / 作业处理时间）

1. 在作业等待时间一定的情况下，作业的处理时间越小，响应比越大

   符合前面最短作业优先的思想

2. 在作业处理时间一定的情况下，作业的等待时间越大，响应比越大

   计算量比较大的进程往后排，但当它的等待比较长的时间，也会被优先调度

处理器调度

本讲内容

1. 处理器调度的层次
2. 处理器调度的算法
3. 单道环境下的调度
4. 多道环境下的调度
5. 低级调度方式算法

单道环境下的调度

作业	进入时间	运行时间（分钟）
JOB1	8：00	120
JOB2	8：50	50
JOB3	9：00	10
JOB4	9：50	20

单道环境下的调度

作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
JOB1	8：00	120				
JOB2	8：50	50				
JOB3	9：00	10				
JOB4	9：50	20				
作业平均周转时间 T = 作业带权平均周转时间 W =						

先来先服务调度算法

单道环境下的调度

	作业	进入时间	估计运行时间(分钟)	开始时间	结束时间	周转时间(分钟)	带权周转时间
①	JOB1	8：00	120	8：00	10：00	120	1
②	JOB2	8：50	50	10：00	10：50	120	2.4
③	JOB3	9：00	10	10：50	11：00	120	12
④	JOB4	9：50	20	11：00	11：20	90	4.5
作业平均周转时间 T = 112.5 = 450 / 4 作业带权平均周转时间 W = 4.975 = 19.9 / 4						450	19.9

先来先服务调度算法

周转时间 = 结束时间 – 进入时间
平均周转时间 = 所有的周转时间相加 / 4
代权周转时间 = 每个周转时间 / 运行时间

先来先服务（FCFS, First Come First Serve）

FCFS

算法思想	主要从“公平”的角度考虑（类似于我们生活中排队买东西的例子）
算法规则	按照作业/进程到达的先后顺序进行服务
用于作业/进程调度	用于作业调度时，考虑的是哪个作业先到达后备队列；用于进程调度时，考虑的是哪个进程先到达就绪队列
是否可抢占？	非抢占式的算法
优缺点	优点：公平、算法实现简单 缺点：排在长作业（进程）后面的短作业需要等待很长时间，带权周转时间很大，对短作业来说用户体验不好。即，FCFS算法对长作业有利，对短作业不利（Eg：排队买奶茶...）
是否会导致饥饿	不会

某进程/作业长期得不到服务

单道环境下的调度

作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
JOB1	8：00	120				
JOB2	8：50	50				
JOB3	9：00	10				
JOB4	9：50	20				
作业平均周转时间 T = 作业带权平均周转时间 W =						

最短作业优先调度算法

单道环境下的调度

	作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
①	JOB1	8：00	120	8：00	10：00	120	1
④	JOB2	8：50	50	10：30	11：20	150	3
②	JOB3	9：00	10	10：00	10：10	70	7
③	JOB4	9：50	20	10：10	10：30	40	2
	作业平均周转时间 T = 95 作业带权平均周转时间 W = 3.25					380	13

最短作业优先调度算法

T、W的值，比先来先服务算法，要更好一点

短作业优先（SJF, Shortest Job First）

SJF

算法思想	追求最少的平均等待时间，最少的平均周转时间、最少的平均平均带权周转时间
算法规则	最短的作业/进程优先得到服务（所谓“最短”，是指要求服务时间最短）
用于作业/进程调度	即可用于作业调度，也可用于进程调度。用于进程调度时称为“短进程优先（SPF, Shortest Process First）算法”
是否可抢占？	SJF和SPF是非抢占式的算法。但是也有抢占式的版本——最短剩余时间优先算法（SRTN, Shortest Remaining Time Next）
优缺点	优点：“最短的”平均等待时间、平均周转时间 缺点：不公平。对短作业有利，对长作业不利。可能产生饥饿现象。另外，作业/进程的运行时间是由用户提供的，并不一定真实，不一定能做到真正的短作业优先
是否会导致饥饿	会。如果源源不断地有短作业/进程到来，可能使长作业/进程长时间得不到服务，产生“饥饿”现象。如果一直得不到服务，则称为“饿死”

单道环境下的调度

作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
JOB1	8：00	120				
JOB2	8：50	50				
JOB3	9：00	10				
JOB4	9：50	20				
作业平均周转时间 T = 作业带权平均周转时间 W =						

最高响应比优先调度算法

单道环境下的调度

JOB2: (10:00-8:50)/50 = 70/50 = 1.4

JOB3: (10:00-9:00)/10 =60/10 = 6

JOB4: (10:00-9:50)/20 =10/20 = 0.5

	作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
①	JOB1	8：00	120	8：00	10：00	120	1
	JOB2	8：50	50	10：10	11：00	130	2.6
②	JOB3	9：00	10	10：00	10：10	70	7
	JOB4	9：50	20	11：00	11：20	90	4.5
	作业平均周转时间 T = 102.5 作业带权平均周转时间 W = 3.775					410	15.1

最高响应比优先调度算法

响应比 = 1 +（作业等待时间 / 作业处理时间）

单道环境下的调度

JOB2: (10:10-8:50)/50
= 80/50 = 1.6

JOB2 虽然执行时间长，但是它的等待时间更长，选择JOB2

JOB4: (10:10-9:50)/20
=20/20 = 1

	作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
①	JOB1	8：00	120	8：00	10：00	120	1
③	JOB2	8：50	50	10：10	11：00	130	2.6
②	JOB3	9：00	10	10：00	10：10	70	7
	JOB4	9：50	20	11：00	11：20	90	4.5
	作业平均周转时间 T = 102.5 作业带权平均周转时间 W = 3.775					410	15.1

最高响应比优先调度算法（更加公平、合理）

响应比 = 1 +（作业等待时间 / 作业处理时间）

高响应比优先（HRRN, Highest Response Ratio Next）

HRRN

- **算法思想**：要综合考虑作业/进程的等待时间和要求服务的时间
- **算法规则**：在每次调度时先计算各个作业/进程的响应比，选择响应比最高的作业/进程为其服务

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

响应比 ≥ 1

- **用于作业/进程调度**：即可用于作业调度，也可用于进程调度
- **是否可抢占？**：非抢占式的算法。因此只有当前运行的作业/进程主动放弃处理机时，才需要调度，才需要计算响应比
- **优缺点**：
  - 综合考虑了等待时间和运行时间（要求服务时间）
  - 等待时间相同时，要求服务时间短的优先（SJF 的优点）
  - 要求服务时间相同时，等待时间长的优先（FCFS 的优点）
  - 对于长作业来说，随着等待时间越来越久，其响应比也会越来越大，从而避免了长作业饥饿的问题
- **是否会导致饥饿**：不会

处理器调度

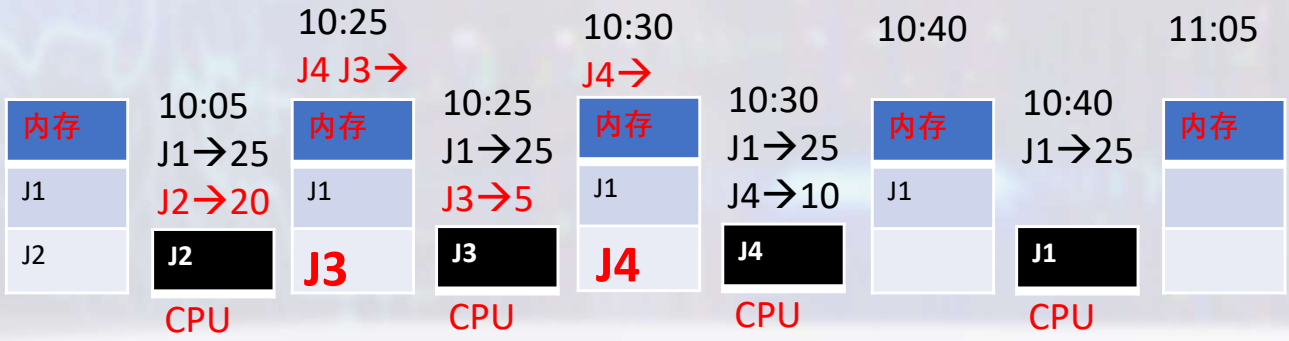
本讲内容

1. 处理器调度的层次
2. 处理器调度的算法
3. 单道环境下的调度
4. 多道环境下的调度
5. 低级调度方式算法

多道环境下的调度

两道环境下有四个作业

- 系统采用短作业优先作业调度算法，作业被调度运行后不再退出 （内存不可被剥夺）
- 当作业投入CPU运行后，可按照作业运行时间长短调整作业执行的次序 (CPU可剥夺)



作业	进入时间	估计运行时间（分钟）
JOB1	10：00	30
JOB2	10：05	20
JOB3	10：10	5
JOB4	10：20	10

两道运行环境允许同时存在两个进程在内存中，CPU可以在这两个进程之间进行切换。

多道环境下的调度

两道环境下有四个作业

10：00，JOB1进入，JOB1被调入执行
10：05，JOB2到达，JOB2也被调入
10：10，JOB3到达输入井，JOB3不能进入内存
10：20，JOB4到达输入井，JOB4不能进入内存
10：25，JOB2运行结束，退出，JOB3进入内存
10：30，JOB3运行结束，退出，JOB4进入内存
10：40，JOB4运行结束，退出，JOB1继续运行
11：05，JOB1运行结束，退出

作业	进入时间	估计运行时间（分钟）
JOB1	10：00	30
JOB2	10：05	20
JOB3	10：10	5
JOB4	10：20	10

多道环境涉及到内存、CPU两次调度

多道环境下的调度

	作业	进入时间	估计运行时间(分钟)	开始时间	结束时间	周转时间(分钟)	带权周转时间
④	JOB1	10：00	30	10：00	11：05	65	4.167
①	JOB2	10：05	20	10：05	10：25	20	1
②	JOB3	10：10	5	10：25	10：30	20	4
③	JOB4	10：20	10	10：30	10：40	20	2
	作业平均周转时间 T = 31.25 作业带权平均周转时间 W = 2.79					125	11.167

知识回顾与重要考点

算法	可抢占？	优点	缺点	考虑到等待时间&运行时间？	会导致饥饿？
先来先服务算法FCFS	非抢占式	公平；实现简单	对短作业不利	等待时间√ 运行时间×	不会
最短作业优先算法SJF	默认为非抢占式，也有SJF的抢占式版本最短剩余时间优先算法（SRTN）	“最短的”平均等待/周转时间；	对长作业不利，可能导致饥饿；难以做到真正的短作业优先	等待时间× 运行时间√	会
最高响应比优先算法HRN	非抢占式	上述两种算法的权衡折中，综合考虑的等待时间和运行时间		等待时间√ 运行时间√	不会

注：这几种算法主要关心对用户的公平性、平均周转时间、平均等待时间等评价系统整体性能的指标，但是不关心“响应时间”，也并不区分任务的紧急程度，因此对于用户来说，交互性很糟糕。因此这三种算法一般适合用于早期的批处理系统，当然，FCFS算法也常结合其他的算法使用，在现在也扮演着很重要的角色。而适合用于交互式系统的调度算法将在下个小节介绍...

处理器调度

本讲内容

1. 处理器调度的层次
2. 处理器调度的算法
3. 单道环境下的调度
4. 多道环境下的调度
5. 低级调度方式算法

低级调度方式算法

1 低级调度流程

记住进程的状态

决定进程什么时候获得处理器

决定进程占用处理器多长时间

把处理器分配给进程

收回处理器

低级调度方式算法

2 低级调度方式

可剥夺式（可抢占式Preemptive）：

- 比正在运行的进程优先级更高的进程就绪时，可强行剥夺正在运行进程的CPU，提供给具有更高优先级的进程使用，或是当运行进程时间片用完后被剥夺

不可剥夺式（不可抢占式 Non-preemptive ）：

- 某一进程被调度运行后，除非由于它自身的原因不能运行，否则一直运行下去

低级调度方式算法

- 先来先服务算法
- 时间片轮转调度算法
- 优先权调度
- 多级反馈队列调度
- 保证调度算法
- 彩票调度算法

低级调度方式算法

1 先来先服务算法

- 按照进程进入就绪队列的先后次序分配处理器
- 进程一旦占有处理器将一直运行，直到结束或阻塞

不可抢占的方法

- 算法容易实现

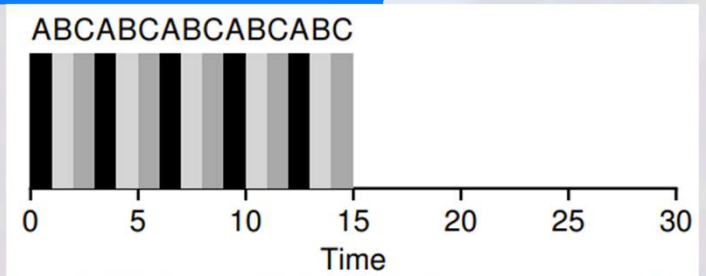
用一个队列和指针

- 算法效率不高，不利于I/O频繁的进程

由于不可剥夺，I/O频繁但CPU简单使用的进程，就一直等待没法被调度

低级调度方式算法

2 时间片轮转调度算法



- 就绪队列中的每个进程轮流地运行一个时间片
- 时间片结束时，强迫当前进程让出处理器，等候下一轮调度
- 一些I/O频繁进程，只需要在CPU上简单计算一下，就去使用I/O设备可以让渡给其他进程，使CPU和I/O设备处于并行使用
- 算法可防止很少使用外围设备的进程过长占用处理器，使要使用外围设备的进程有机会去启动外围设备

低级调度方式算法

2 时间片轮转调度算法

时间片取选

- 轮转法调度是一种剥夺式调度，进程切换开销较大，开销与时间片的大小有关
- 时间片取值太小，多数进程不能在一个时间片内运行完毕，切换频繁，开销增大
- 时间片取值太大，随就绪队列里进程数目增加，轮转一次的总时间增大，对每个进程的响应速度放慢了

降低为先来先服务

- 时间片大小的确定要从进程个数、切换开销、系统效率和响应时间等方面综合考虑

时间片大小，可以设置为一个动态的值，根据进程的情况调整

低级调度方式算法

3 保证调度算法

- 对每个任务做出明确的性能保证，然后去实现它
- 在一个有n个进程运行的系统中，保证每个进程获得CPU处理能力的1/n （保证公平）
- 跟踪各个进程自创建以来已经使用了多少CPU时间，根据各个进程应获得的CPU时间，计算实际获得的CPU时间和应获得的CPU时间之比，优先调度比率最低的进程

低级调度方式算法

4 彩票调度算法

- 为每个进程发放一定数量的彩票，调度程序随机选择一张彩票，持有该彩票的进程获得系统资源
- 如果某些进程需要更多的机会，可被给予更多彩票，增加其中奖机会
- 算法反应迅速，合作进程还可交换彩票

处理器调度

本讲内容

1. 处理器调度的层次
2. 处理器调度的算法
3. 单道环境下的调度
4. 多道环境下的调度
5. 低级调度方式算法

1. 进程控制块（答PCB也对）是操作系统为了管理进程设置的一个专门的数据结构，用它来记录进程的外部特征，描述进程的运动变化过程。

2. 进程的基本状态有执行态、就绪态和阻塞(等待)态。

1. 下列作业调度算法中，（D）与作业的运行时间和等待时间有关。

A、先来先服务算法 B、短作业优先算法

C、均衡调度算法 D、最高响应比调度算法

4、在操作系统中，PSW的中文全称是（A）

A、程序状态字 B、进程标识符 C、作业控制块 D、进程控制块

6、以下关于进程的说法，错误的是（B）

A、进程是程序在处理机上的一次执行过程

B、一个进程是由若干作业组成的

C、在线程出现后，进程仍然是操作系统中资源分配的最小单位

D、进程具有创建其他进程的功能

1. 简单叙述进程的基本特征。

（1）结构性。进程由程序、数据、进程控制块和栈四个部分构成。
（2）共享性。并发的进程共享系统的资源。
（3）动态性。系统中的进程有诞生、有消亡，是动态变化的。
（4）独立性。进程是系统进行资源分配、调度和保护的独立单位。
（5）制约性。系统中并发执行的进程，可能彼此协作，也可能是彼此竞争，体现了制约性。
（6）并发性。多个进程在系统中并发运行，分时、分空间的使用计算机系统的资源。

2. 阐述进程调度的常用算法：先来先服务、轮转法。

1）先来先服务算法（2分）

按照进程进入就绪队列的先后次序分配处理器。先进入就绪队列的进程优先被挑选，运行进程一旦占有处理器将一直运行直到结束或阻塞。

2）时间片轮转调度算法（3分）

调度程序每次把CPU分配给就绪队列首进程使用一个时间片，就绪队列中的每个进程轮流地运行一个时间片。当这个时间片结束时，强迫一个进程让出处理器，让它排列到就绪队列的尾部，等候下一轮调度。

单道批处理环境下有5个作业，各作业进入系统的时间和估计运行时间如题下表所示。如果应用短作业优先的作业调度算法，试将表格填写完整。

作　业	进入系统时间	估计运行时间/分钟	结 束 时 间	带权周转时间
1	8:00	40		
2	8:20	30		
3	8:30	12		
4	9:00	18		
5	9:10	5		

作业	进入系统时间	估计运行时间/分钟	结束时间	带权周转时间
1	8:00	40	8:40	1
2	8:20	30	9:22	2.07
3	8:30	12	8:52	1.83
4	9:00	18	9:45	2.5
5	9:10	5	9:27	3.4

一个具有两道作业的批处理系统中，作业调度采用以短作业优先的非抢占式算法，进程调度采用以优先数小者优先的抢占式算法，请根据下表计算各作业的结束时间和周转时间。

作 业 名	到 达 时 间	估计运行时间	优 先 数
A	9：00	30分钟	4
B	9：10	10分钟	3
C	9：15	5分钟	5
D	9：20	40分钟	2

作业	结束时间	周转时间/分钟
A	9:40	40
B	9:20	10
C	10:25	70
D	10:20	60