第三讲　处理机调度

本讲内容

考点一: 处理机调度 ★

考点二: 调度算法 ★★★★★

考点三: 上下文切换机制 ★★★

考点四: 死锁 ★★★★★

考点一：

处理机调度

考点框架

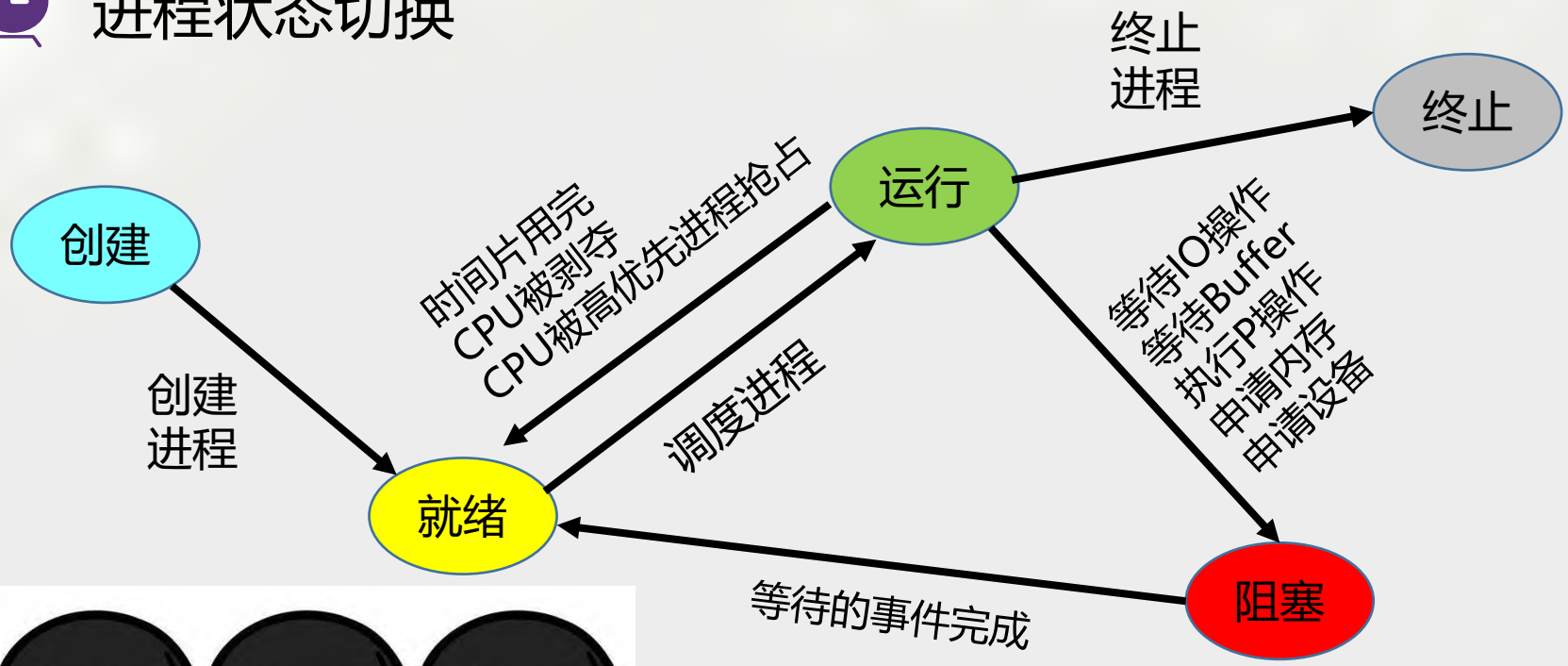
- 处理机调度的层次
- 处理机调度的方式
- 选择调度算法的若干准则

考点一: 进程的基本概念

处理机调度的层次

考点一: 处理机调度

进程状态切换



依赖于调度

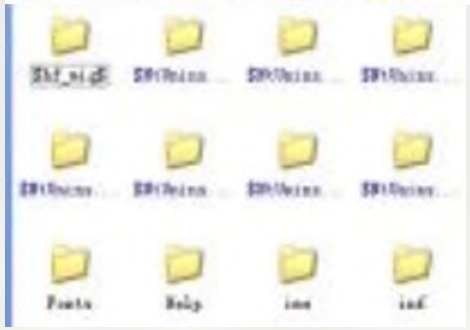
考点一: 处理机调度

进程状态切换

目的：合理地处理计算机的软/硬件资源

- 调度概念
- 调度层次
  - 作业调度（高级调度）
  - 内存调度（中级调度）
  - 进程调度（低级调度）

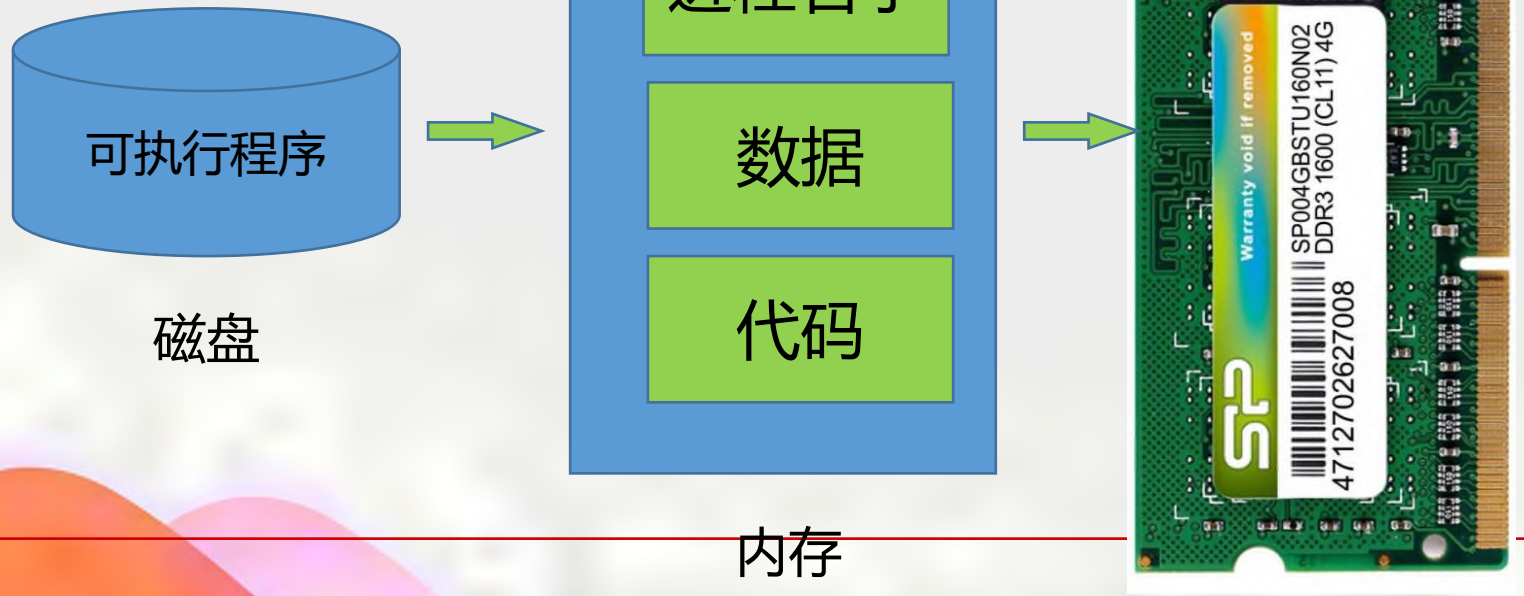
- 当有诸多任务要处理，但由于资源有限，这些事情没法同时处理。
- 这就需要确定某种规则来决定处理这些任务的顺序，这就是“调度”研究的问题。



考点一: 处理机调度

处理机调度的层次

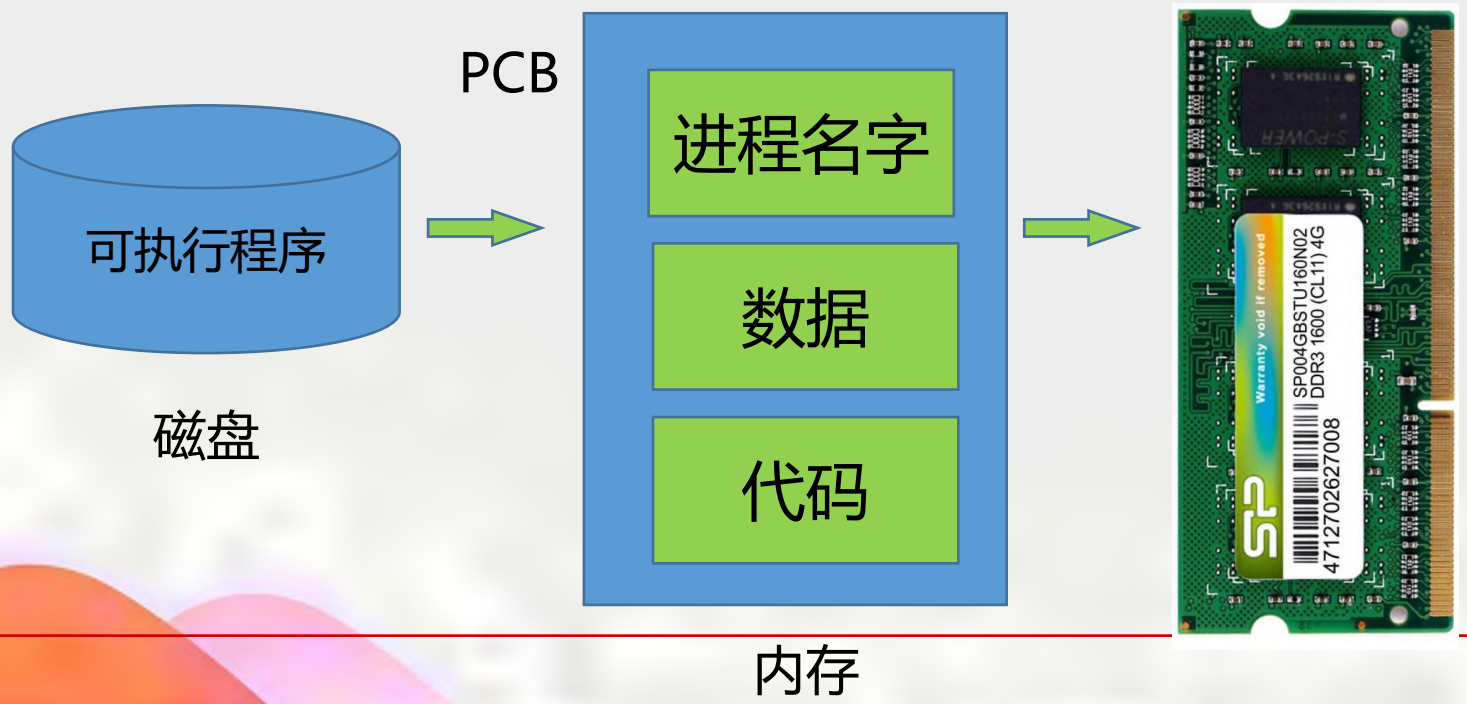
- 高级调度（作业调度）
- 批处理系统中，将外存上后备队列中的某些作业调入内存，为其创建进程，分配资源、插入就绪队列 。



考点一: 处理机调度

处理机调度的层次

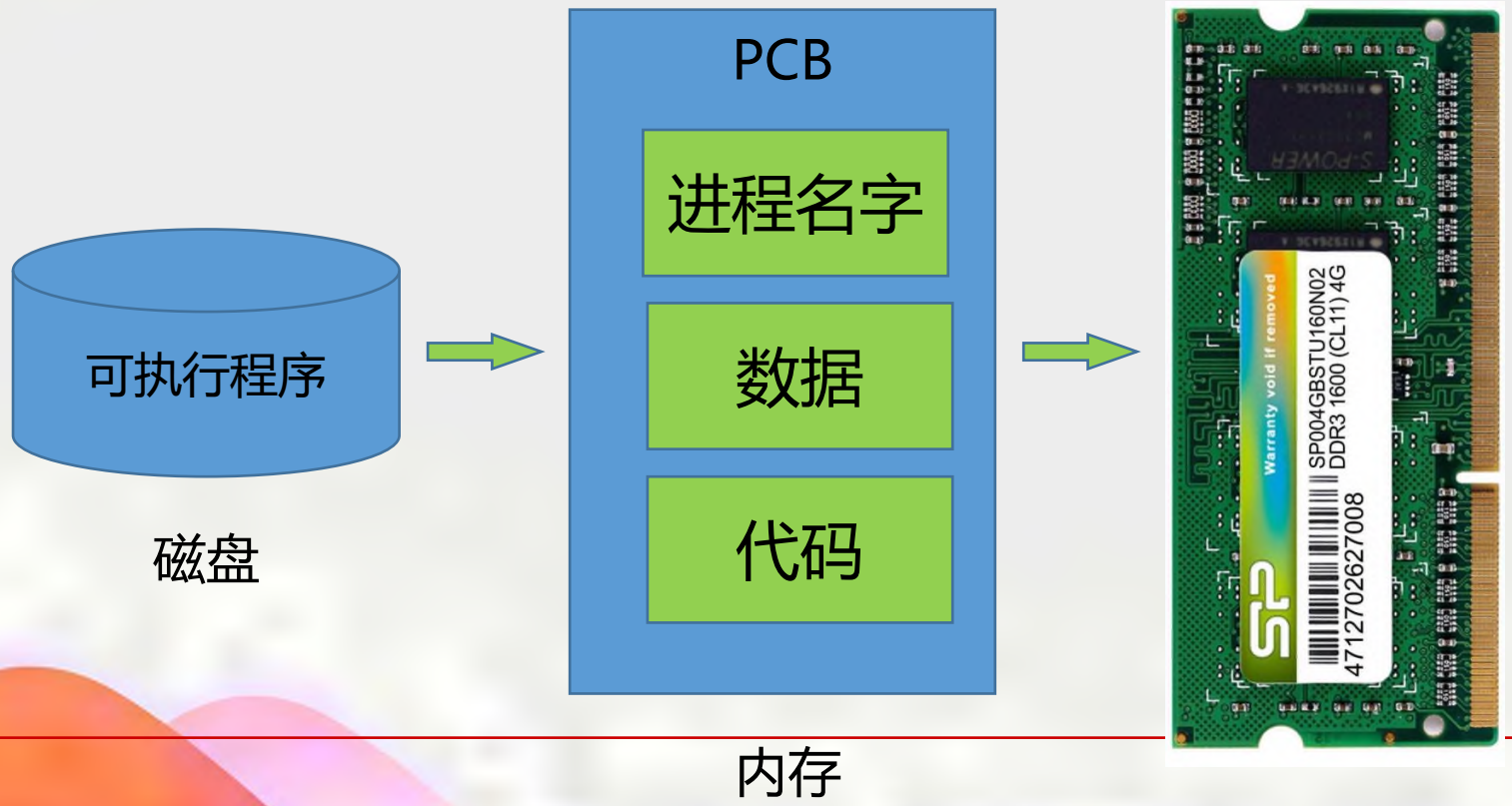
- 高级调度（作业调度）
- 高级调度是辅存（外存、磁盘）与内存之间的调度。每个作业只调入一次，调出一次。
- 作业调入时会建立相应的PCB，作业调出时才撤销PCB。



考点一: 处理机调度

处理机调度的层次

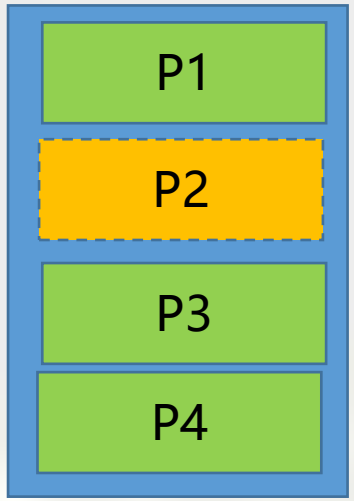
- 高级调度（作业调度）
- 高级调度主要是指调入的问题，因为只有调入的实际需要操作系统来确定，但调出的实机必然是作业运行结束才调出。



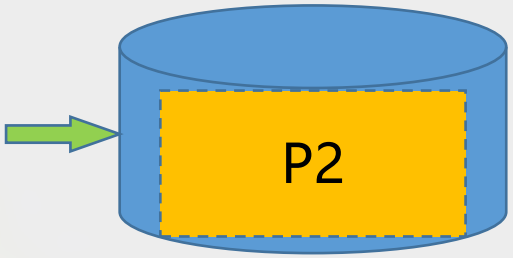
考点一: 处理机调度

处理机调度的层次

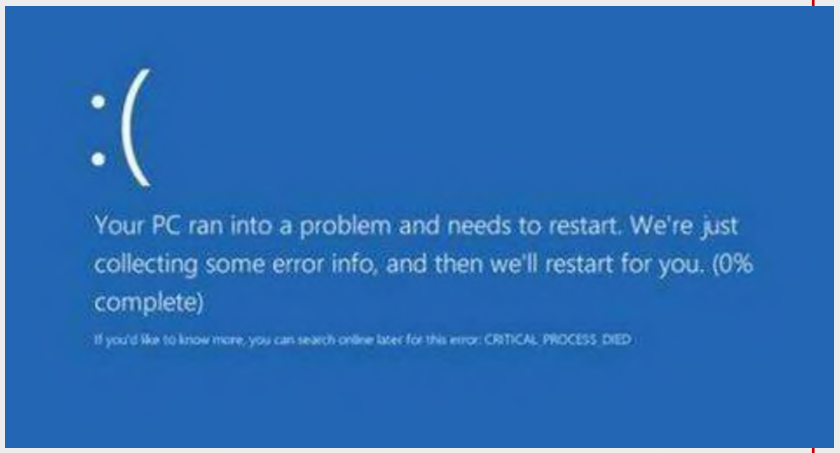
- 中级调度（内存调度 / 交换调度）
- 内存可用空间紧张时，将暂不具备运行条件的进程挂起，释放其所占内存资源，调至外存



内存



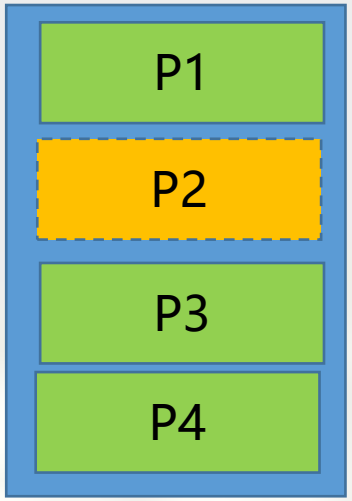
磁盘



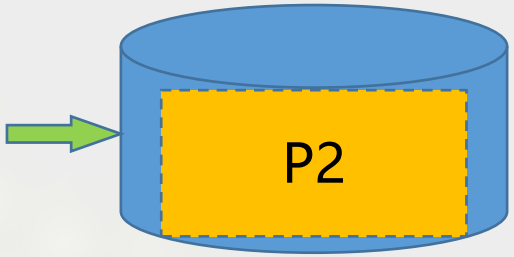
考点一: 处理机调度

处理机调度的层次

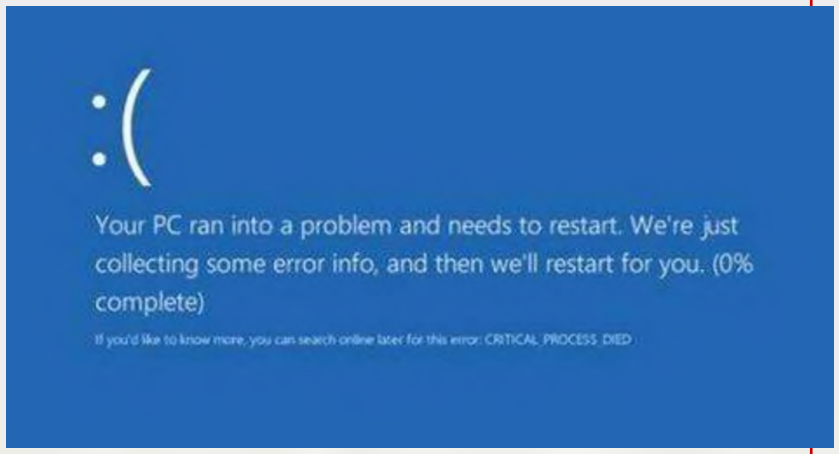
- 中级调度（内存调度 / 交换调度）
  - 当这些进程重又具备运行条件、且内存又稍有空闲时，由中级调度来决定把外存上的哪些又具备运行条件的就绪进程，重新调入内存，并修改其状态为就绪状态，挂在就绪队列上等待进程调度。



内存



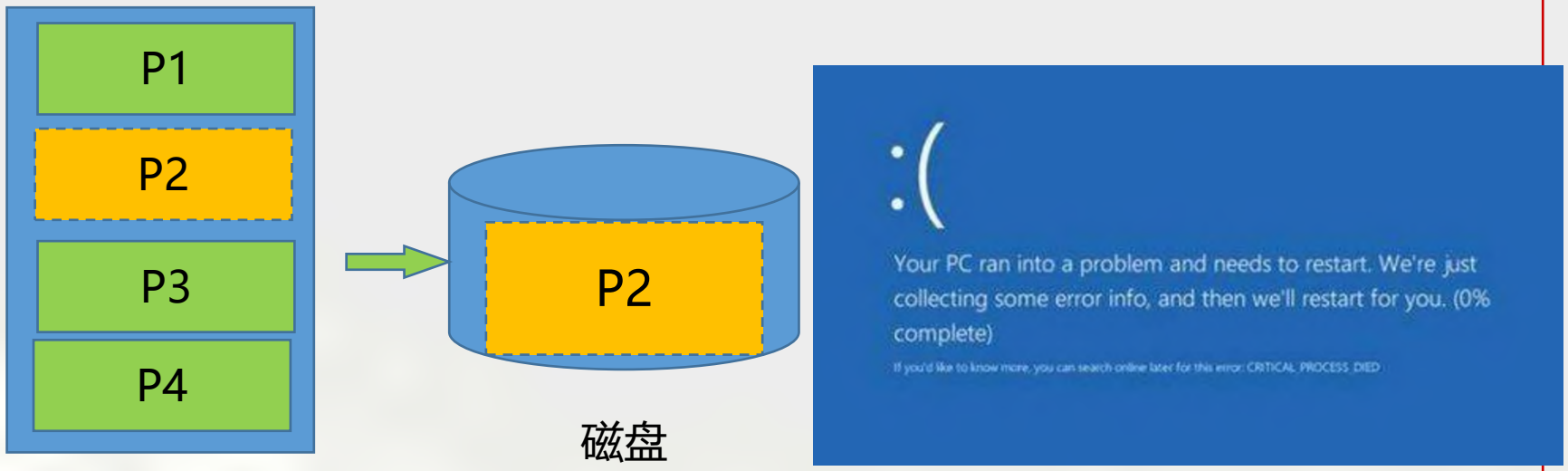
磁盘



考点一: 处理机调度

处理机调度的层次

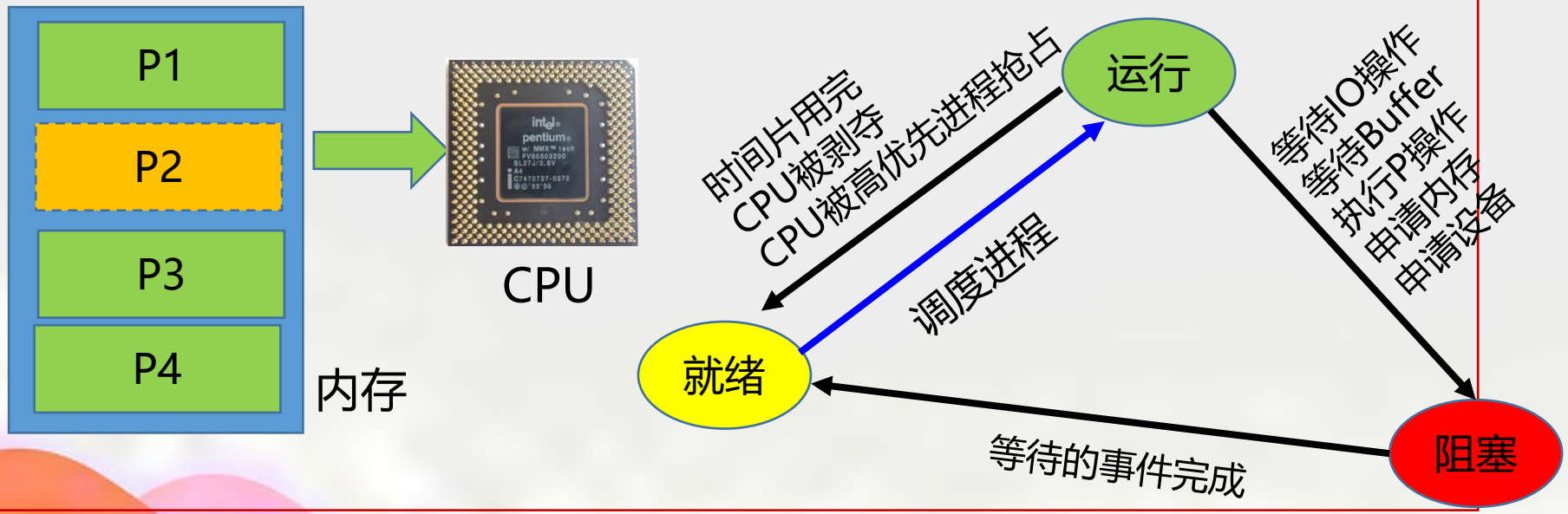
- 中级调度（内存调度 / 交换调度）
  - 中级调度根据主存资源决定主存中所能容纳的进程数目，并根据进程的当前状态来决定外存和主存中进程的对换。中级调度起到平滑和调整系统负荷的作用，提高主存利用率和系统吐吞率。



考点一: 处理机调度

处理机调度的层次

- 低级调度（进程调度 / 线程调度）
- 将处理机分配就绪队列中的某进程
- 最基本的调度，操作系统的核心部分；批处理操作系统、分时操作系统、实时操作系统等必备功能



考点一: 处理机调度

处理机调度的层次

	要做什么	调度发生在哪	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

三种调度算法的关系和比较

考点一: 处理机调度

处理机调度的方式

考点一: 处理机调度

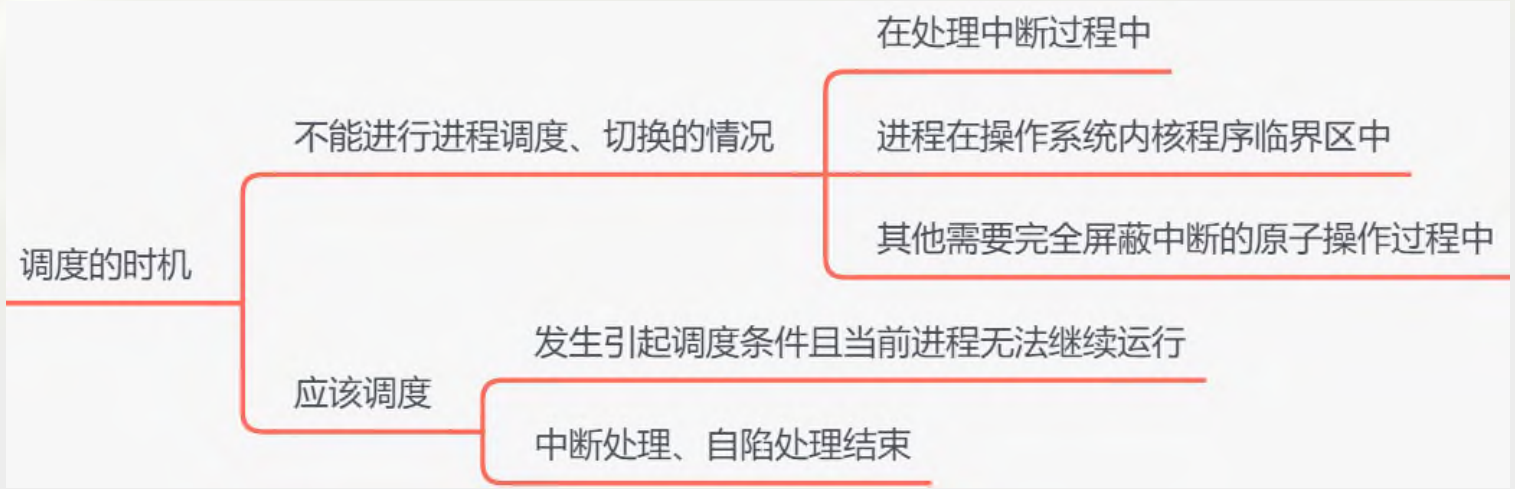
- 非抢占式优先权调度算法【主动放弃处理机】
- 在这种方式下，系统一旦把处理机分配给就绪队列中优先权最高的进程后，该进程便一直执行下去，直至完成； 或因发生某事件使该进程放弃处理机时。

考点一: 处理机调度

- 抢占式优先权调度算法【被迫放弃处理机】
- 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，只要又出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。

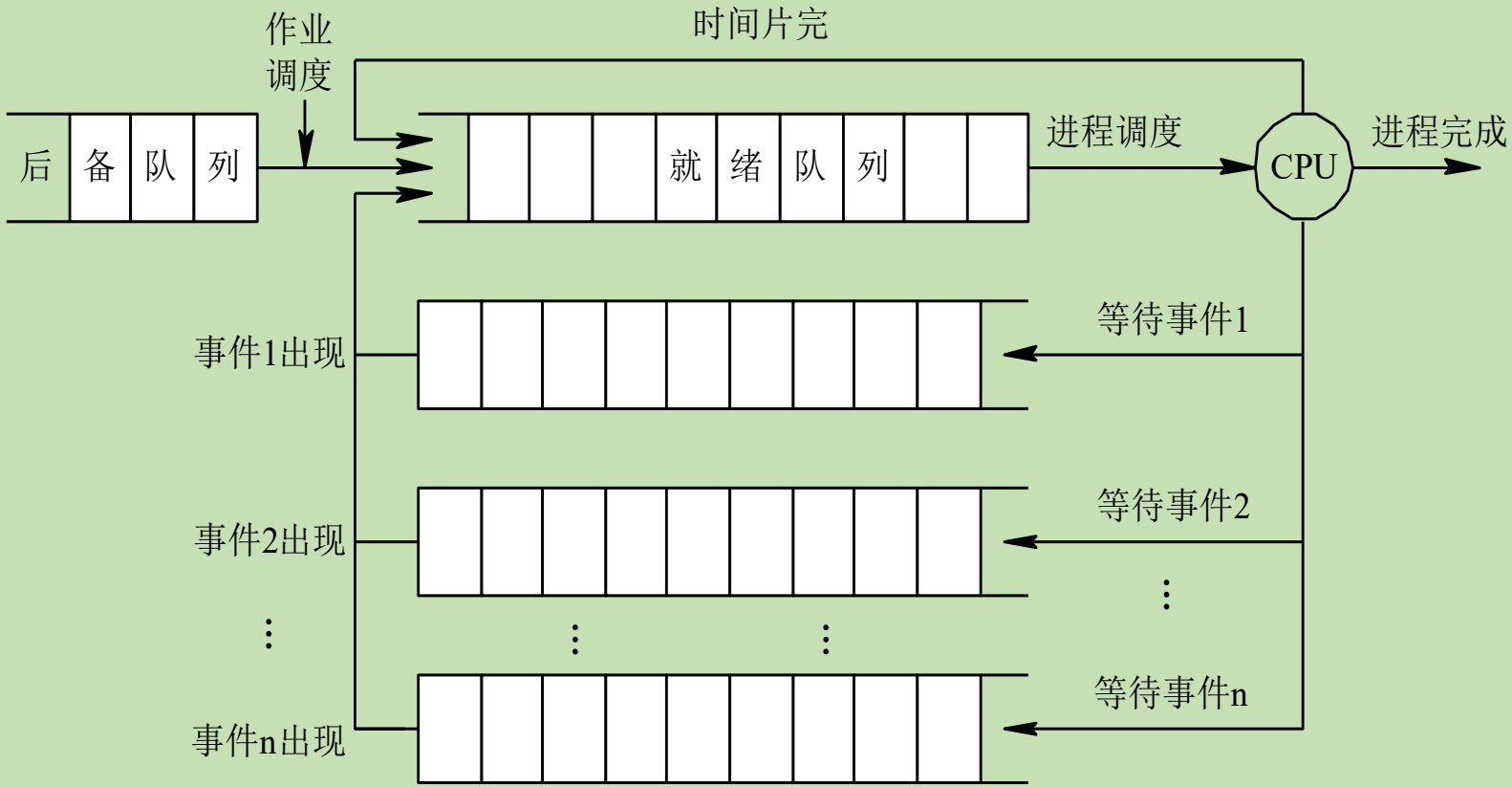
考点一: 处理机调度

- 调度的时机



考点一: 处理机调度

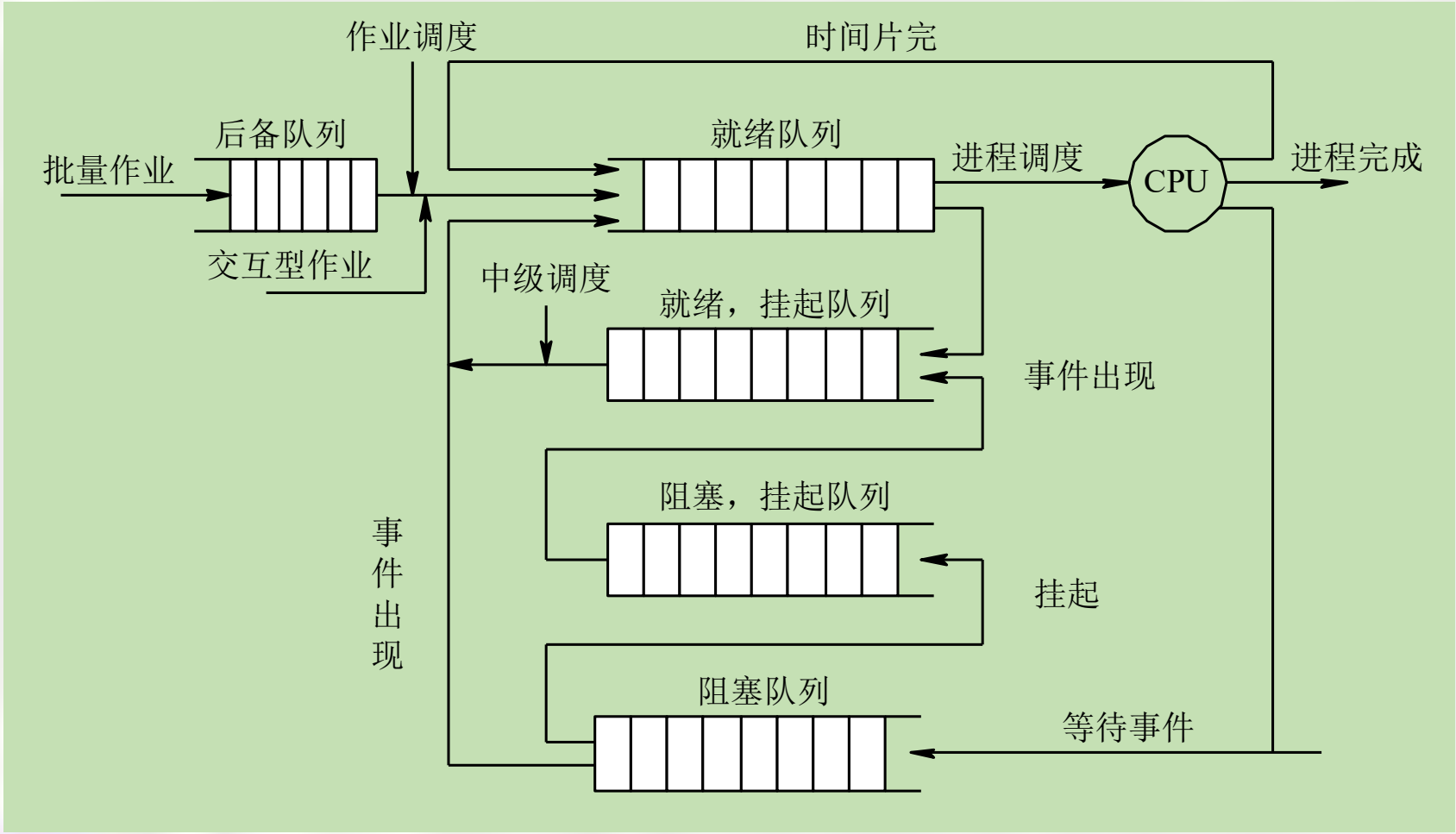
- 调度的实现方式



具有高、低两级调度的调度队列模型

考点一: 处理机调度

- 调度的实现方式



具有三级调度时的调度队列模型

考点一: 处理机调度

选择调度算法的若干准则

考点一: 处理机调度

- 基本准则
  - CPU利用率
  - 系统吞吐量
  - 周转时间
    - =作业完成时间-作业提交时间
    - 平均周转时间
      - 短作业优先，平均周转时间最小
      - 多个进程不同调度算法——甘特图
    - 带权周转时间
      - 作业周转时间/作业实际运行时间
  - 等待时间
  - 响应时间

考点一: 处理机调度

- CPU利用率
- CPU造价极其昂贵，应该让CPU尽可能多地工。
- CPU利用率:指CPU“忙碌”的时间占总时间的比例。利用率=忙碌的时间 / 总时间
- 例如: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?

  CPU利用率 = 10/15 = 66.66%

  打印机利用率 = 5/15 = 33.33%

考点一: 处理机调度

- 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。
- 系统吞吐量 = 总共完成了多少道作业 / 总共花了多少时间
- 例如: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 。

考点一: 处理机调度

- 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。
- 它包括四个部分: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。
- ①作业在外存后备队列上等待作业调度(高级调度)的时间
- ②进程在就绪队列上等待进程调度(低级调度)的时间
- ③进程在CPU上执行的时间
- ④进程等待I/O操作完成的时间。

考点一: 处理机调度

- 具体的计算公式是:
- ①周转时间=作业完成时间-作业提交时间（用户更关心单个作业周转时间）
- ②平均周转时间=各作业周转时间之和/作业数（操作系统更关心整体表现）
- ③带权周转时间=作业周转时间/作业实际运行的时间（必然大于等于1，越小越好）
- ④平均带权周转时间=各作业带权周转时间之和/作业数

考点一: 处理机调度

- 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。

考点一: 处理机调度

- 对于用户来说，更关心自己的单个作业的周转时间；
- 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。
- 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。

【牛刀小试】

1．【重庆理工大学 2014】下面哪种情况不会引发进程调度？（ ）

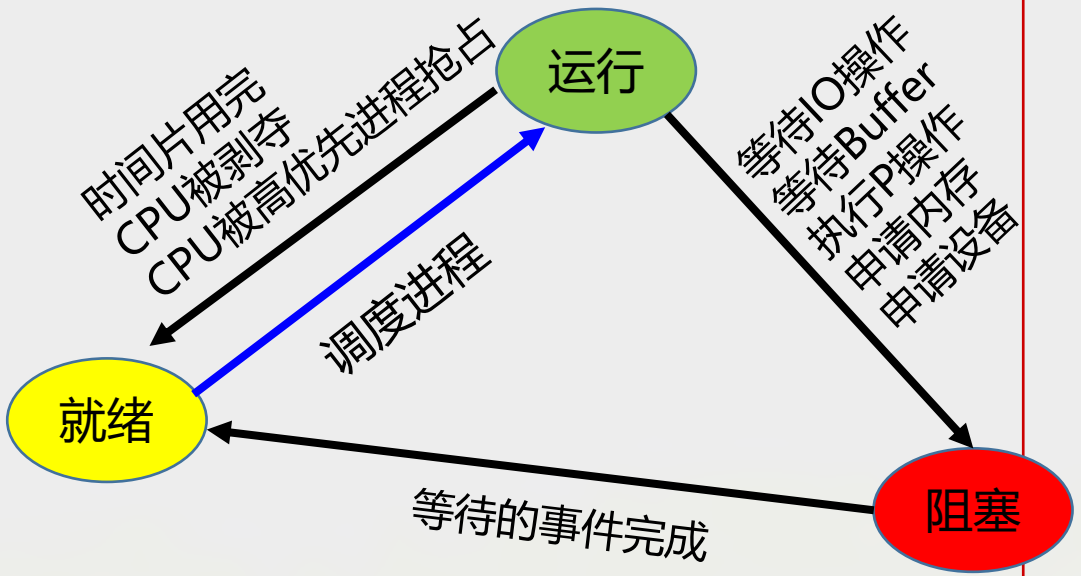
A．进程正常结束或异常中止

B．正在执行的进程因I/O请求而被阻塞

C．某等待打印机的进程发现其他使用打印机的进程已经打印完毕

D．在引入时间片的系统中，时间片用完

C【**解析**】所谓进程调度，是指在处于就绪态的一堆进程/线程里，按照一定的调度算法，选出一个进程/线程并给它分配CPU时间让它运行，从而实现多进程/多线程的并发执行。



1．【重庆理工大学 2014】下面哪种情况不会引发进程调度？（ ）

A．进程正常结束或异常中止

B．正在执行的进程因I/O请求而被阻塞

C．某等待打印机的进程发现其他使用打印机的进程已经打印完毕

D．在引入时间片的系统中，时间片用完

C【**解析**】引起进程调度的因素：

(1)正在执行的进程执行完毕，A正确；

(2)执行中的进程提出I/O请求等，进入了阻塞状态，B正确；

(3)时间片用完（时间片轮转调度方式下），D正确；

(4)在进程通信或同步过程中执行了某种原语操作；

(5)高优先级的进程进入。

1．【重庆理工大学 2014】下面哪种情况不会引发进程调度？（ ）

A．进程正常结束或异常中止

B．正在执行的进程因I/O请求而被阻塞

C．某等待打印机的进程发现其他使用打印机的进程已经打印完毕

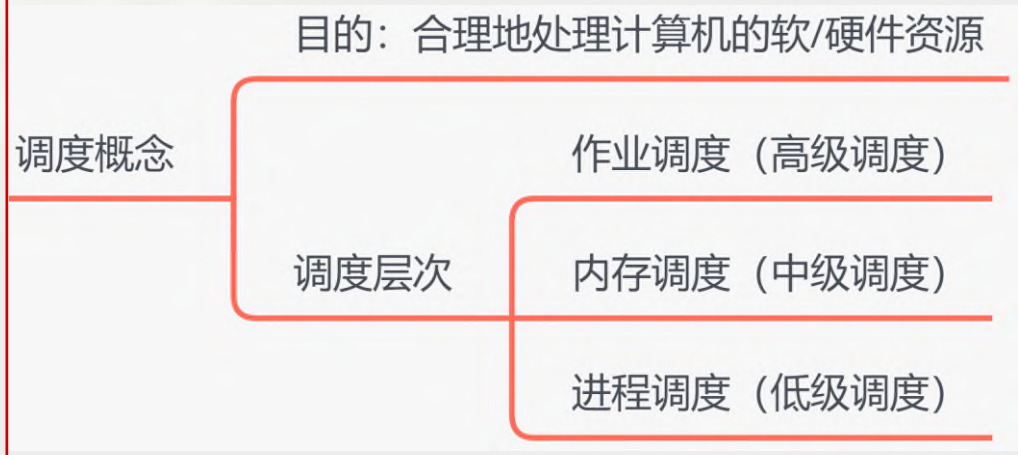
D．在引入时间片的系统中，时间片用完

C【**解析**】正等待打印机的进程处于阻塞状态，而其发现其他打印进程使用打印机结束后，该进程可以获得打印机使用权，那么它的状态从阻塞态变为就绪态，不一定会被立即调度，不会引起进程调度，因此选择C。

2．【南京理工大学 2015】处理机调度分三层，其中进程调度是(　)。

A．高级调度　B．中级调度

C．低级调度　D．作业调度



2．【南京理工大学 2015】处理机调度分三层，其中进程调度是(　)。

A．高级调度　B．中级调度

C．低级调度　D．作业调度

C【**解析**】处理机调度分三层，分别为高级调度、中级调度和低级调度，其中进程调度是低级调度，因此选择C。

3．【重庆理工大学 2016】进程调度的关键问题是(　)。

A．时间片大小　　B．进程调度算法

C．CPU速度　　D．内存空间的大小

B【解析】无论是在批处理系统还是分时系统中，用户进程数一般都多于处理机数，这将导致它们互相争夺处理机。系统进程也同样需要使用处理机。这就要求进程调度程序按一定的策略，动态地把处理机分配给处于就绪队列中的某一个进程，以使之执行。

所以进程调度的关键问题就是调度算法的选择。调度算法的选择是进程调度（处理机管理→调度（作业调度、进程调度））中最核心的内容，因此选择B。

4．【南京工业大学 2012】下列各项中，不一定是进程调度时机的是(　　)。

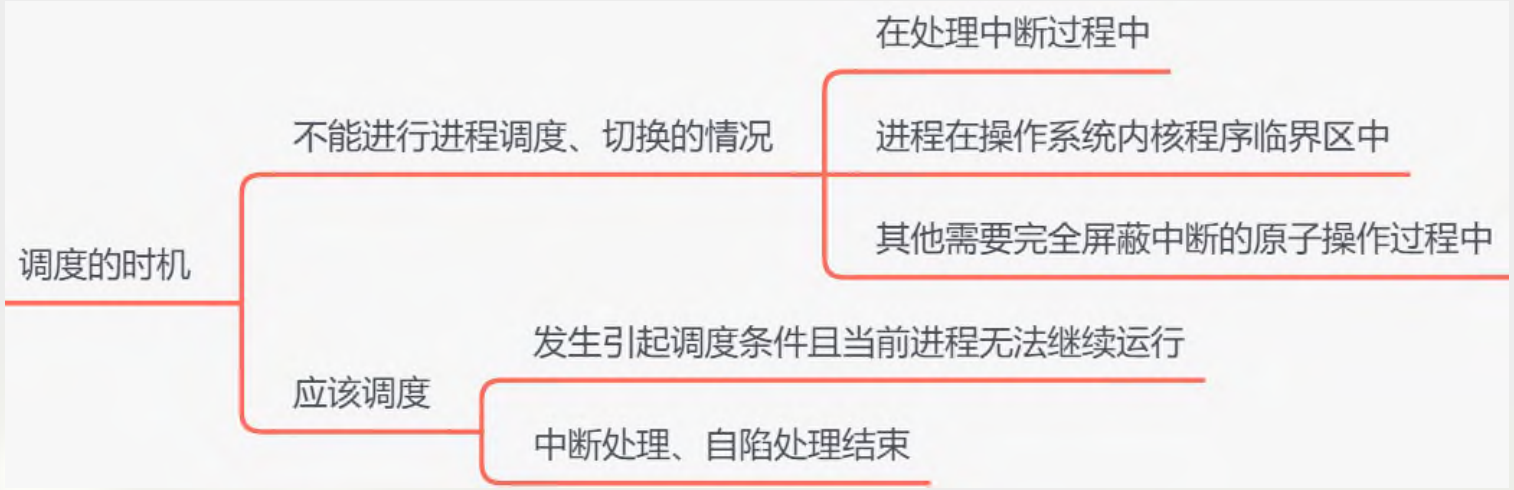
A．现运行的进程正常结束或异常结束

B．现运行的进程从运行态进入就绪态

C．现运行的进程从运行态进入等待态

D．有一进程从等待态进入就绪态

- 调度的时机



4．【南京工业大学 2012】下列各项中，不一定是进程调度时机的是(　)。

A．现运行的进程正常结束或异常结束

B．现运行的进程从运行态进入就绪态

C．现运行的进程从运行态进入等待态

D．有一进程从等待态进入就绪态

引起进程调度的因素：

(1)正在执行的进程执行完毕，A正确；

(2)执行中的进程提出I/O请求等，进入了阻塞状态，B正确；

(3)时间片用完（时间片轮转调度方式下）

(4)在进程通信或同步过程中执行了某种原语操作；

(5)高优先级的进程进入。

4．【南京工业大学 2012】下列各项中，不一定是进程调度时机的是(　)。

A．现运行的进程正常结束或异常结束

B．现运行的进程从运行态进入就绪态

C．现运行的进程从运行态进入等待态

D．有一进程从等待态进入就绪态

D【**解析**】所谓进程调度，是指在处于就绪态的一堆进程/线程里，按照一定的调度算法，选出一个进程/线程并给它分配CPU时间让它运行，从而实现多进程/多线程的并发执行。若有CPU资源让出，不一定能发生进程调度；若没有CPU资源让出，则一定不会发生进程调度。有一进程从等待态进入就绪态，没有分配处理机，因此选择D。

5．【华东师范大学 2015】在批处理作业中，周转时间是(　)。

A．作业运行时间

B．作业等待时间

C．作业的相对等待时间

D．作业被调度进入内存到运行完毕的时间

D【**解析**】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，因此周转时间是作业等待时间和运行时间之和，因此选择D。

6．【北京交通大学 2016】(　)是指从作业提交给系统到作业完成的时间间隔。

A．周转时间　　B．响应时间

C．等待时间　　D．运行时间

A【**解析**】作业周转时间：作业完成时间 作业提交时间；作业响应时间：从提交第一个请求到产生第一个响应所用的时间；作业等待时间：作业开始执行时间—作业提交时间；作业运行时间：作业开始执行时间 作业完成时间，因此选择A。

7．【北京交通大学　2017】进程调度又称为低级调度，其主要功能是(　　)。

A．选择一个作业调入内存

B．选择一个主存中的进程调出到外存

C．选择一个外存中的进程调入到主存

D．将一个就绪的进程投入运行

- 低级调度（进程调度 / 线程调度）
- 将处理机分配就绪队列中的某进程
- 最基本的调度，操作系统的核心部分；批处理操作系统、分时操作系统、实时操作系统等必备功能

7．【北京交通大学　2017】进程调度又称为低级调度，其主要功能是(　　)。

A．选择一个作业调入内存

B．选择一个主存中的进程调出到外存

C．选择一个外存中的进程调入到主存

D．将一个就绪的进程投入运行

D【解析】选择一个作业调入内存属于高级调度；选择一个主存中的进程调出到外存，再从外存调入到主存属于中级调度；将一个就绪的进程投入运行属于低级调度，因此选择D。

8．作业是用户提交的，进程是系统自动生成的，除此之外，两者的区别是(　)。

A．两者执行不同的程序段

B．前者以用户任务为单位，后者以操作系统控制为单位

C．前者是批处理的，后者是分时的

D．后者可并发执行，前者则不同

B【**解析**】作业是从用户角度出发，它是由用户提交以用户任务为单位；进程是从操作系统出发，它由系统生成，是操作系统的资源分配和独立运行的基本单位，因此选择B。

9．下列关于调度的叙述中，正确的有(　)。

Ⅰ．CPU调度算法决定了进程执行的顺序，若有n个进程需要调度，就有n^2种可能的调度算法

Ⅱ．时间片用完，进程正常终止或异常终止，进程提出I/O请求后被阻塞都会引起进程调度

Ⅲ．面向系统选择调度方式和调度算法时，应遵循的准则包括系统吞吐量高、响应时间快等

Ⅳ．操作系统中的三级调度是指作业调度、进程调度和资源调度

A．仅Ⅰ、Ⅲ　　B．仅Ⅰ、Ⅳ　　C．仅Ⅱ、Ⅲ、Ⅳ　　D．仅Ⅱ

D【**解析**】CPU调度算法决定了进程执行的顺序，若有n个进程需要调度，就有n!种可能的调度顺序；

- 时间片用完，进程正常终止或异常终止，进程提出I/O请求后被阻塞都会引起进程调度；
- 面向系统选择调度方式和调度算法时，应遵循的准则包括系统吞吐量高、处理机利用率好、各类资源的平衡利用，面向用户选择调度算法时，应遵循周转时间短、响应时间快、截止时间的保证、优先权准则；
- 操作系统中的三级调度是指作业调度、进程调度和内存调度，因此选择D。

10．进程（线程）调度的时机有(　)。

Ⅰ．运行的进程（线程）运行完毕

Ⅱ．运行的进程（线程）所需资源未准备好

Ⅲ．运行的进程（线程）的时间片用完

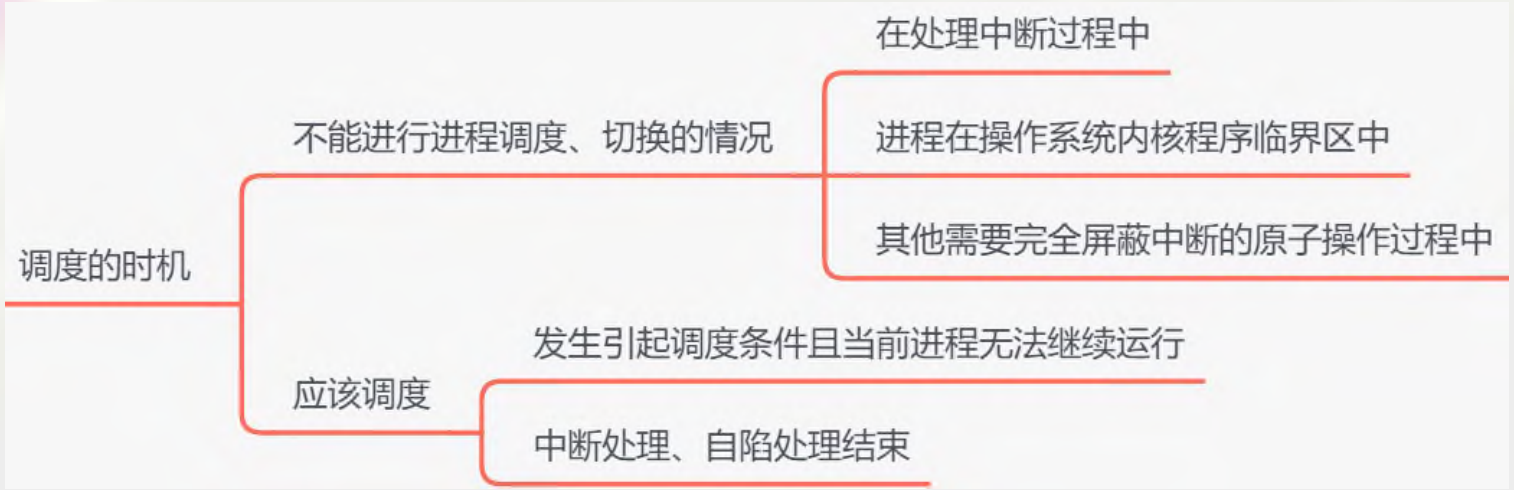
Ⅳ．运行的进程（线程）自我阻塞

Ⅴ.运行的进程（线程）出现错误

A．仅Ⅱ、Ⅲ、Ⅳ和Ⅴ　　B．仅Ⅰ和Ⅲ

C．仅Ⅱ、Ⅳ和Ⅴ　　D．全部都是

调度的时机


调度的时机
不能进行进程调度、切换的情况
在处理中断过程中
进程在操作系统内核程序临界区中
其他需要完全屏蔽中断的原子操作过程中
应该调度
发生引起调度条件且当前进程无法继续运行
中断处理、自陷处理结束

引起进程调度的因素：

(1)正在执行的进程执行完毕，

(2)执行中的进程提出I/O请求等，进入了阻塞状态，

(3)时间片用完（时间片轮转调度方式下）

(4)在进程通信或同步过程中执行了某种原语操作；

(5)高优先级的进程进入。

D【**解析**】进程（线程）调度的时机：

运行的进程（线程）运行完毕、

运行的进程（线程）自我阻塞、

运行的进程（线程）的时间片用完、

运行的进程（线程）所需资源没有准备好、

运行的进程（线程）出现错误。

当CPU方式是抢占方式时，就绪队列中的某个进程（线程）的优先级高于当前运行进程（线程）的优先级时，也会发生进程（线程）调度。

谢谢大家

考点二:

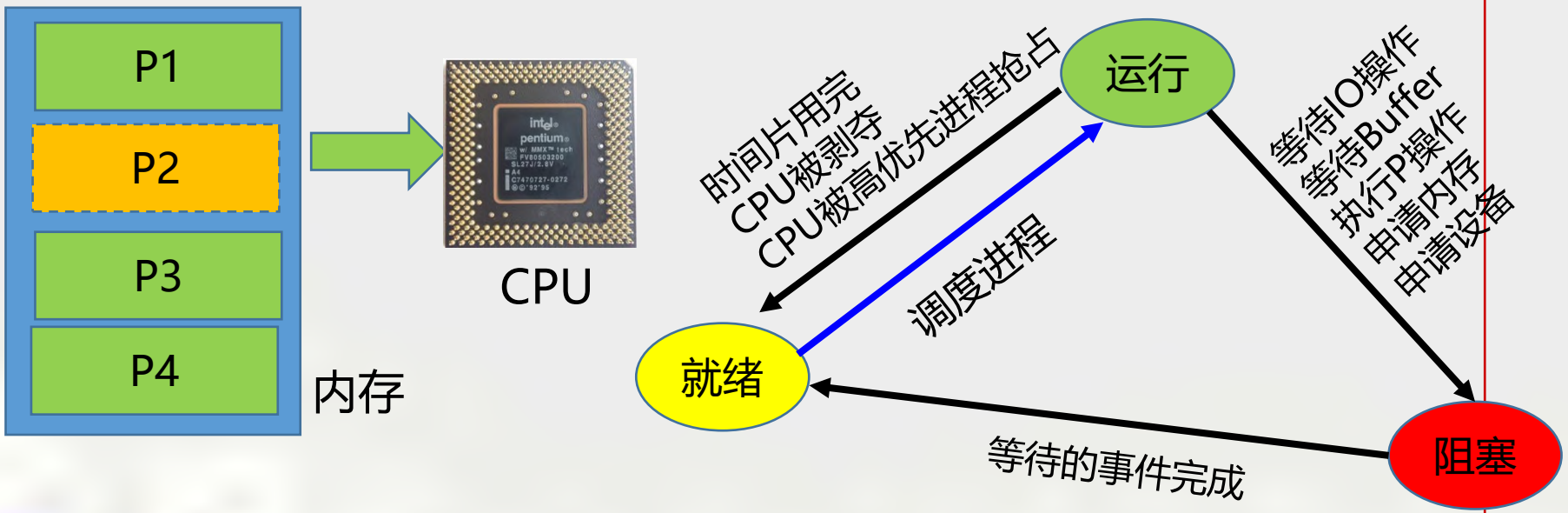
调度算法

考点框架

- 先来先服务算法
- 短作业优先调度算法
- 高优先权优先调度算法
- 高响应比优先调度算法
- 时间片轮转调度算法
- 多级反馈队列调度算法

考点二: 调度算法

- 低级调度（进程调度 / 线程调度）
- 将处理机分配就绪队列中的某进程
- 最基本的调度，操作系统的核心部分；批处理操作系统、分时操作系统、实时操作系统等必备功能



考点二: 调度算法

调度算法	算法思想	作业调度	进程调度
先来先服务	从后备作业队列中选择若干最先进入	√	√
短作业优先	以运行/计算时间为优先级	√	√
优先级调度	以优先数为优先级	√	√
高响应比优先	响应比=等待时间+运行时间/ 运行时间	√	X
时间片轮转	所有进程按照轮询模式运行	X	√
多级反馈队列	设置优先级不同的队里 队列内部按照FCFS规则	X	√

调度算法总述

考点二: 调度算法

先来先服务算法

考点二: 调度算法

- 以等待时间为优先级的调度方法。
- 从后备作业队列中选择若干最先进入该队列的作业调入内存，为其分配资源、创建进程，然后放入就绪队列

考点二: 调度算法

作业	到达时间	执行顺序	服务时间	开始执行时间	执行结束时间	周转时间	带权周转时间
J1	1	1	**6**	1	6+1	7-1	6/6=**1**
J2	2	2	**160**	7	160+7	167-2	165/160=**1.03**
J3	3	3	**8**	167	8+167	175-3	172/8=**21.5**
J4	4	4	**120**	175	120+175	295-4	291/120=**2.43**
J5	5	5	**3**	295	3+295	298-5	293/3=**97.67**
平均						927/5 =185.4	123.63/5 =24.73

考点二: 调度算法

- 该算法公平，算法简单，实现简单；
- 但是随机平均等待时间波动很大，平均的周转时间也会比较大，CPU密集型进程会导致IO设备闲置，IO密集也会出现CPU等待，花费时间少的任务可能排在花费时间长的任务后面，没有考虑抢占；也可能导致I/O和CPU之间的重叠处理（cpu密集型进程会导致I/O设备闲置时，I/O密集型进程也在等待）；
- 本算法适合于CPU密集型计算任务，不适合于IO密集型任务；
- 该算法有利于长作业，不利于短作业；

考点二: 调度算法

短作业优先调度算法

考点二: 调度算法

- 短作业(进程)优先调度算法SJ(P)F，是指对短作业或短进程优先调度的算法。它们可以分别用于作业调度和进程调度。
- 短作业优先(SJF)的调度算法，以运行/计算时间为优先级，优先将运行计算时间短的若干作业调入内存，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。
- 短进程优先(SPF)调度算法，则是从就绪队列中选出一估计运行时间最短的进程，将处理机分配给它，使它立即执行并一直执行到完成，或发生某事件而被阻塞放弃处理机时，再重新调度。

考点二: 调度算法

- 当一个更短时间进程来了之后，有两种策略：
- 不抢占，将这个时间更短的程序排在前面，但是继续执行本来在执行的进程，这种是非抢占性的。（SPN）
- 抢占，将这个时间更短的程序与正在执行的程序剩余所需要执行的时间进行比较，如果跟多，这打断正在执行的程序，这种是可抢占的。

考点二: 调度算法

作业	到达时间	执行顺序	服务时间	开始执行时间	执行结束时间	周转时间	带权周转时间
J1	1	1	6	1	6+1	7-1	6/6=1
J2	2	5	160	138	160+138	298-2	296/160=1.85
J3	3	3	8	10	8+10	18-3	15/8=1.875
J4	4	4	120	18	120+18	138-4	134/120=1.12
J5	5	2	3	7	3+7	10-5	5/3=1.67
平均						456/5 =91.2	7.51/5 =1.5

考点二: 调度算法

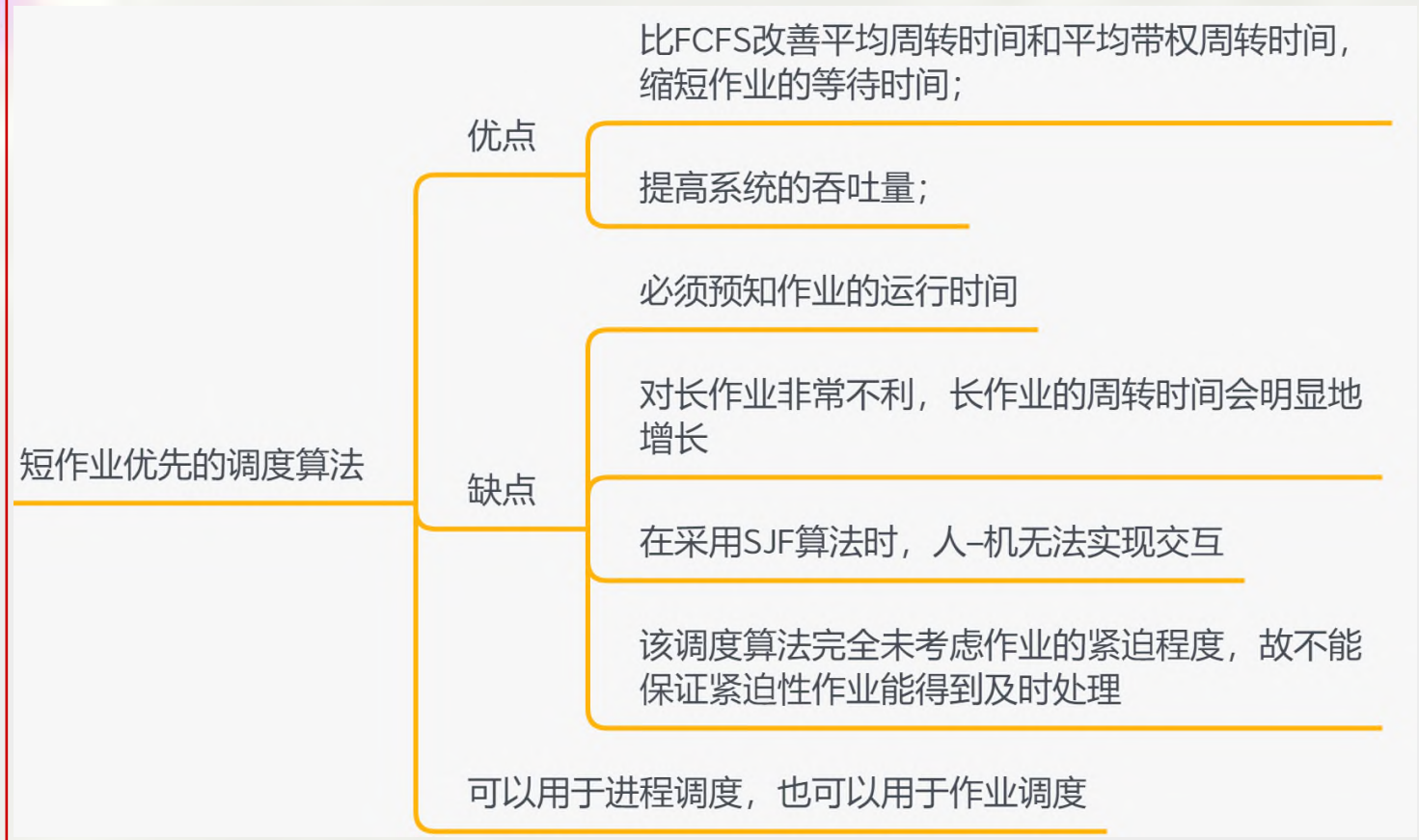
给出一个可抢占方式的例子来讲解算法的执行原理，设4个就绪作业到达系统和所需CPU时间如表3-3所示。

作业名	到达系统时间	所需CPU时间
作业1	0	8
作业2	2	4
作业3	3	9
作业4	5	5

考点二: 调度算法

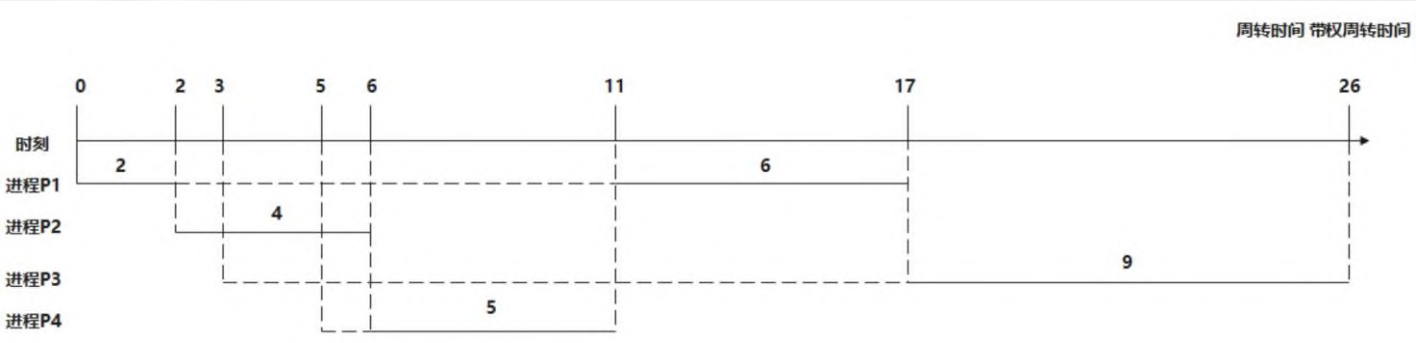
- 最短剩余时间优先(SRTF)算法是剥夺式的最短时间优先调度算法。每当有进程加入就绪队列改变时就需要调度，如果新作业需要的CPU时间比当前正在执行的作业剩余所需CPU时间短，则新作业将抢占当前作业的处理器。另外，当一个进程完成时也需要调度。此算法适用于作业调度和进程调度

考点二: 调度算法


短作业优先的调度算法
优点
比FCFS改善平均周转时间和平均带权周转时间，缩短作业的等待时间；
提高系统的吞吐量；
缺点
必须预知作业的运行时间
对长作业非常不利，长作业的周转时间会明显地增长
在采用SJF算法时，人–机无法实现交互
该调度算法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
可以用于进程调度，也可以用于作业调度

考点二: 调度算法

- 按照最短剩余时间优先算法调度4个作业，请分别计算各作业的周转时间、带权周转时间、平均周转时间和平均带权周转时间，



- 进程P1：周转时间：17-0=17　　带权周转时间：17/8=2.13
- 进程P2：周转时间：6-2=4　　带权周转时间：4/4=1
- 进程P3：周转时间：26-3=23　　带权周转时间：23/9=2.55
- 进程P4：周转时间：11-5=6　　带权周转时间：6/5=1.2

考点二: 调度算法

高优先权优先调度算法

考点二: 调度算法

- 高优先权(静态优先权)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。一般地，优先权是利用某一范围内的一个整数来表示的，例如，0~7或0~255中的某一整数， 又把该整数称为优先数。
- 该算法调度时，依赖外部赋予的优先级（假设优先数越大，优先级越小），
- 高优先权调度算法也有抢占和不可抢占两种方式，

考点二: 调度算法

作业/优先数	到达时间	执行顺序	服务时间	开始执行时间	执行结束时间	周转时间	带权周转时间
J1/ 6	1	1	**6**	1	6+1	7-1	6/6=**1**
J2 /5	2	5	**160**	138	160+138	298-2	296/160=**1.85**
J3/ 3	3	4	**8**	130	8+130	138-3	135/8=**16.88**
J4/ 2	4	3	**120**	10	120+10	130-4	126/120=**1.05**
J5/ 1	5	2	**3**	7	3+7	10-5	5/3=**1.67**
平均						568/5 =113.6	22.45/5 =4.49

考点二: 调度算法

- 非抢占式优先权调度算法的特点：
- 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，直至完成，
- 因发生某事件使该进程放弃处理机时，系统才将处理机重新分配给另一优先权最高的进程
- 主要用于批处理系统中，也可用于某些对实时性要求不严的实时系统中

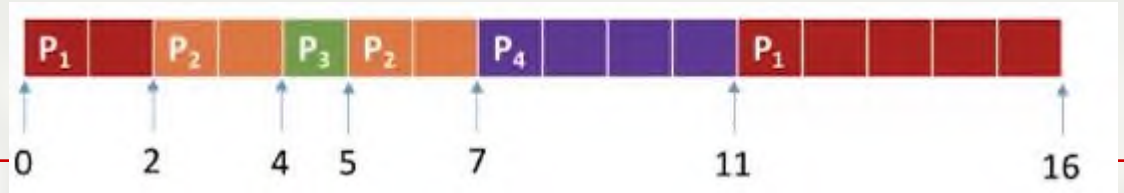
考点二: 调度算法

- 可抢占的优先权调度算法，分析进程运行情况。(优先数越大，优先级越高
- 调度过程：每次调度时选择当前已到达且优先级最高的进程。当前进程主动放弃处理机时发生调度。就是在运行进程的过程中，放弃当前进行，去进行优先级高的进程。

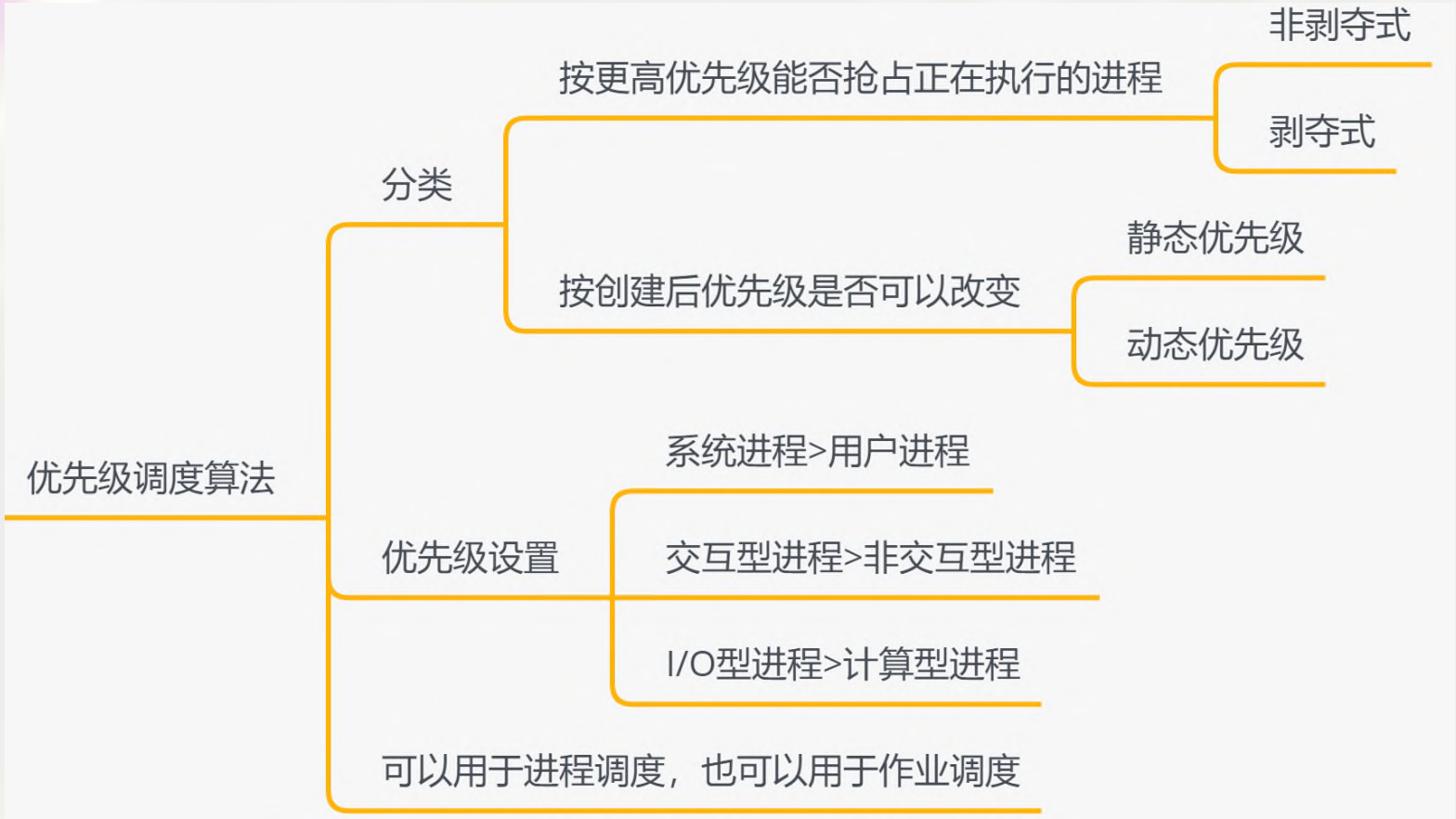
考点二: 调度算法

进程	到达时间	运行时间	优先数
P1	0	7	1
P2	2	4	2
P3	4	1	3
P4	5	4	2

- 以下括号内表示当前处于就绪队列的进程
- 0时刻(P1):只有P1到达，P1上处理机。
- 2时刻(P2):P2到达就绪队列，优先级比P1更高，发生抢占。P1回到就绪队列，P2上处理机。
- 4时刻(P1、P3):P3到达，优先级比P2更高，P2回到就绪队列，P3抢占处理机。
- 5时刻(P1、P2、P4):P3完成，主动释放处理机，同时，P4也到达，由于P2比P4更先进入就绪队列，因此选择P2上处理机
- 7时刻(P1P4):P2完成，就绪队列只剩P1、P4，P4上处理机。11时刻(P1):P4完成，P1上处理机
- 16时刻():P1完成，所有进程均完成



考点二: 调度算法


优先级调度算法
分类
按更高优先级能否抢占正在执行的进程
非剥夺式
剥夺式
按创建后优先级是否可以改变
静态优先级
动态优先级
优先级设置
系统进程>用户进程
交互型进程>非交互型进程
I/O型进程>计算型进程
可以用于进程调度，也可以用于作业调度

考点二: 调度算法

高响应比优先调度算法

考点二: 调度算法

- 高响应比优先调度算法既考虑作业的执行时间也考虑作业的等待时间，综合了先来先服务和最短作业优先两种算法的特点。
- 该算法中的响应比是指作业等待时间与运行比值，响应比公式定义如下：
- 响应比 =（等待时间+要求服务时间）/ 要求服务时间,即RR=（w+s）/s=1+w/s，
- 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，因此响应比一定是大于1的。

考点二: 调度算法

- 优先权Rp =1+等待时间/服务时间=响应(周转)时间/服务时间

作业	到达时间	服务时间	响应比 J1-J5	执行顺序	开始执行时间	执行结束时间	周转时间	带权周转时间
J1	1	**6**	**1+0/6 (1时刻)**	**1**	1	6+1	7-1	6/6=**1**
J2	2	**160**	**1+5/160 (7时刻)**					
J3	3	**8**	**1+4/8 (7时刻)**					
J4	4	**120**	**1+3/120 (7时刻)**					
J5	5	**3**	**1+2/3 (7时刻)**	**2**	7	3+7	10-5	5/3=**1.67**
平均								

考点二: 调度算法

- 优先权Rp =1+等待时间/服务时间=响应(周转)时间/服务时间

作业	到达时间	服务时间	响应比 J1-J5	执行顺序	开始执行时间	执行结束时间	周转时间	带权周转时间
J1	1	**6**	**1(1时刻)**	**1**	1	6+1	7-1	6/6=**1**
J2	2	**160**	**1+8/160 (10时刻)**					
J3	3	**8**	**1+7/8 (10时刻)**	**3**	10	8+10	18-3	15/8=**1.875**
J4	4	**120**	**1+6/120 (10时刻)**					
J5	5	**3**	**1+2/3 (7时刻)**	**2**	7	3+7	10-5	5/3=**1.67**
平均								

考点二: 调度算法

- 优先权Rp =1+等待时间/服务时间=响应(周转)时间/服务时间

作业	到达时间	服务时间	响应比 J1-J5	执行顺序	开始执行时间	执行结束时间	周转时间	带权周转时间
J1	1	**6**	**1(1时刻)**	**1**	1	6+1	7-1	6/6=**1**
J2	2	**160**	**1+16/160 (18时刻)**					
J3	3	**8**	**1+7/8 (10时刻)**	**3**	10	8+10	18-3	15/8=**1.875**
J4	4	**120**	**1+14/120 (18时刻)**	**4**	18	120+18	138-4	134/120=**1.12**
J5	5	**3**	**1+2/3 (7时刻)**	**2**	7	3+7	10-5	5/3=**1.67**
平均								

考点二: 调度算法

- 优先权Rp =1+等待时间/服务时间=响应(周转)时间/服务时间

作业	到达时间	服务时间	响应比 J1-J5	执行顺序	开始执行时间	执行结束时间	周转时间	带权周转时间
J1	1	**6**	**1(1时刻)**	**1**	1	6+1	7-1	6/6=**1**
J2	2	**160**	**1+16/160 (138时刻)**	**5**	138	160 +138	298-2	296/160=**1.85**
J3	3	**8**	**1+7/8 (10时刻)**	**3**	10	8+10	18-3	15/8=**1.875**
J4	4	**120**	**1+14/120 (18时刻)**	**4**	18	120+18	138-4	134/120=**1.12**
J5	5	**3**	**1+2/3 (7时刻)**	**2**	7	3+7	10-5	5/3=**1.67**
平均							456/5 =91.2	7.51/5 =1.5

考点二: 调度算法

- 高响应比优先调度算法
  - 原理
    - 在每次选择作业投入运行时，先计算此时后备作业队列中每个作业的响应比RP然后选择其值最大的作业投入运行
    - 优先权=(等待时间+要求服务时间)/要求服务时间=响应时间/要求服务时间=1+等待时间/要求服务时间
  - 特点
    - 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，因而类似于SJF算法，有利于短作业
    - 当要求服务的时间相同时，作业的优先权又决定于其等待时间，因而该算法又类似于FCFS算法
    - 对于长时间的优先级，可以为随等待时间的增加而提高，当等待时间足够长时，也可获得处理机
  - 只用于作业调度

考点二: 调度算法

时间片轮转调度算法

考点二: 调度算法

- 按照各进程到达就绪队列的顺序，轮流让各个进程执行一个时间片。
- 当执行的时间片用完时，由一个计时器发出时钟中断请求，调度程序便据此信号来停止该进程的执行，并将它送往就绪队列的末尾；
- 再把处理机分配给就绪队列中新的队首进程，同时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，在一给定的时间内，均能获得一时间片的处理机执行时间

考点二: 调度算法

进程	P1	P2	P3	P4	P5
到达时间	1	1	1	1	1
服务时间	6	160	8	120	3

时间片调度算法：时间片为2

考点二: 调度算法

- 32时刻前： P2 共占用4个时间片，服务时长8，剩余服务时间152；P4 共占用3个时间片，服务时长6，剩余服务时间114
- 32时刻后： P4，P2交替各使用1个时间片，直到P4运行结束
  - P4从32时刻到运行结束，需要57个时间片，则P4，P2共需114个时间片，服务时长228，截至时刻 32+228=260。其中P4最后一个开始时刻为256，P2最后一个开始时刻为258
  - 260时刻开始：P2还需152-114=38个服务时间，19个时间片。运行结束时刻为260+38=298

P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
1(2)	3(2)	5(2)	7(2)	9(2)	11(2)	13(2)	15(2)	17(2)	**19(1)**

P1	P2	P3	P4	P2	P3	P4	P2	P4	P2	...	P4	P2
20(2)	22(2)	24(2)	26(2)	28(2)	**30(2)**	32(2)	34(2)	36(2)	38(2)	...	**256(2)**	**258(40)**

考点二: 调度算法

- 时间片调度算法：时间片为2

进程/优先数	到达时间	完成顺序	服务时间	开始执行时间	执行结束时间	周转时间	带权周转时间
P1	1	2	**6**	1	20+2	22-1	21/6=**3.5**
P2	1	5	**160**	3	258+40	298-1	297/160=**1.856**
P3	1	3	**8**	5	30+2	32-1	31/8=**3.875**
P4	1	4	**120**	7	256+2	258-1	257/120=**2.142**
P5	1	1	**3**	9	19+1	20-1	19/3=**6.333**
平均						625/5 =125	17.706/5 =3.541

考点二: 调度算法

- 时间片调度算法：时间片为4

进程	P1	P2	P3	P4	P5
到达时间	1	1	1	1	1
服务时间	6	160	8	120	3

考点二: 调度算法

- 30时刻前： P2 共占用2个时间片，服务时长8，剩余服务时间152；P4 共占用1个时间片，服务时长4，剩余服务时间116
- 30时刻后： P4，P2交替各使用1个时间片，直到P4运行结束
  - P4从30时刻到运行结束，需要116/4=29个时间片，则P4，P2共需58个时间片，服务时长232，截至时刻 30+232-1=261。其中P4最后一个开始时刻为254，P2最后一个开始时刻为258
  - 262时刻开始：P2 还需152-116=36个服务时间，9个时间片（258开始需要40个服务时间，10个时间片）。运行结束时刻为262+36-1=298-1=297

P1	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P2**
1(4)	5(4)	9(4)	13(4)	**17(3)**	**20(2)**	22(4)	**26(4)**	30(4)	34(4)

P4	**P2**	**P4**	**P2**	**…**	**P4**	**P2**
38(4)	42(4)	46(4)	50(4)	…	254(4)	258(40)

考点二: 调度算法

分时操作系统为保证响应用户请求的及时性，每个进程运行一个时间片后，释放处理器，调度就绪队列下一进程

时间片轮转

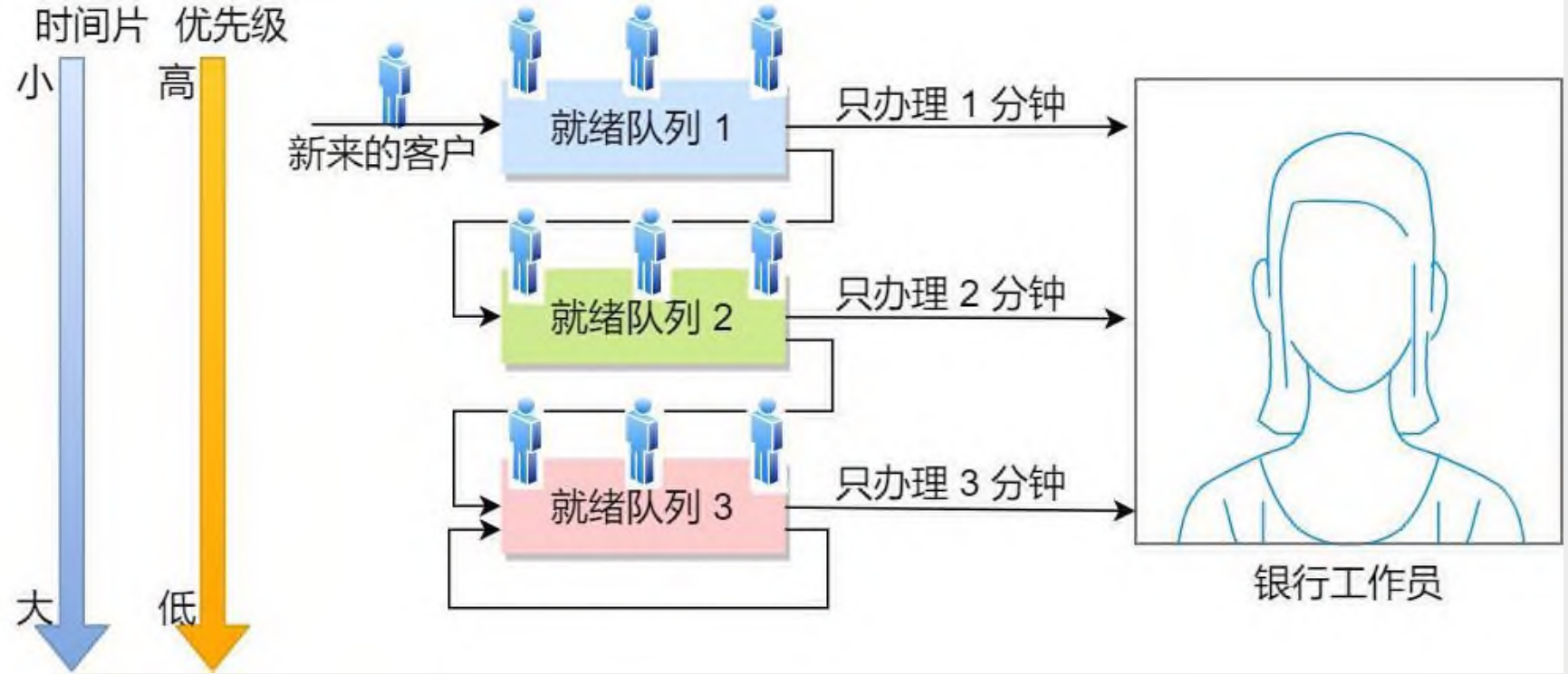
- 可以保证就绪队列中的所有进程，在一给定的时间内，均能获得一时间片的处理机执行时间
- 只用于进程调度

考点二: 调度算法

多级反馈队列调度算法

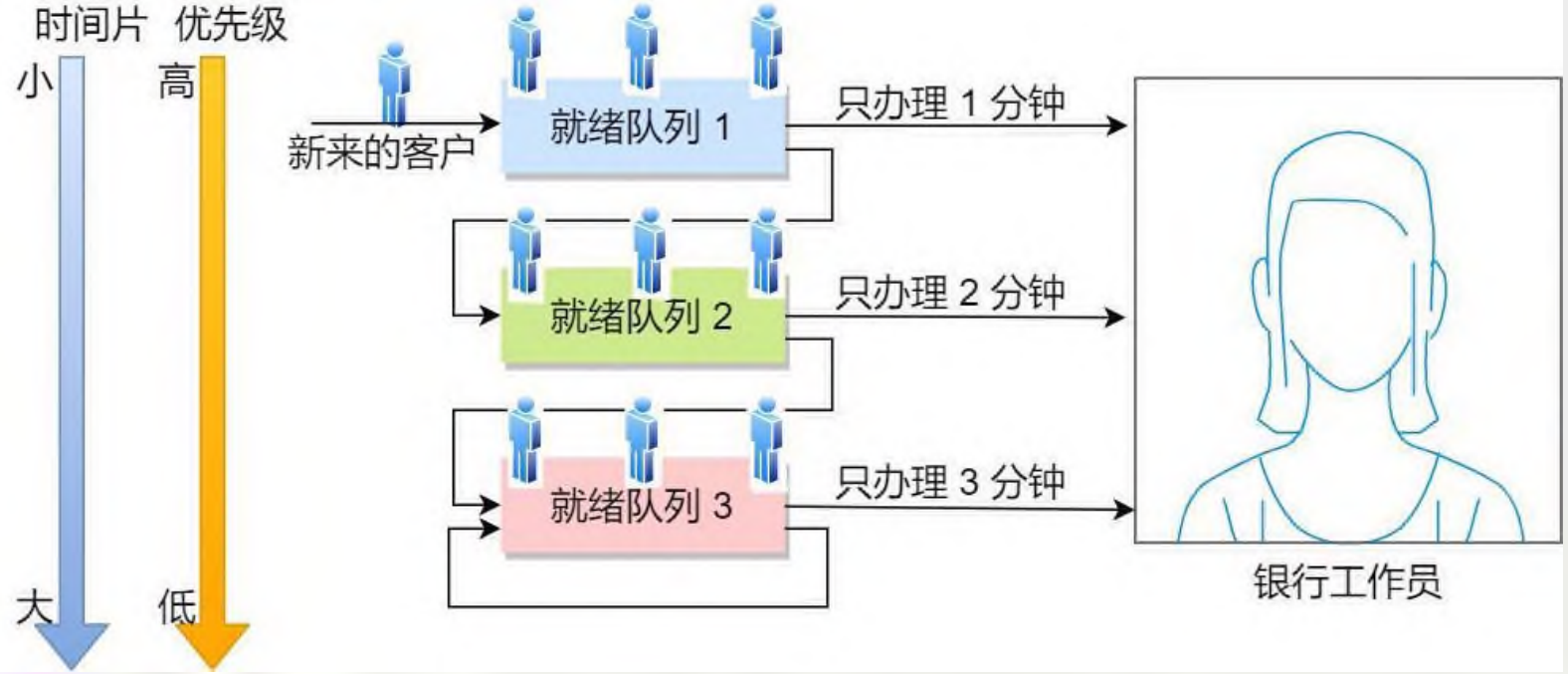
考点二: 调度算法

- 多级反馈队列调度算法是基于时间片轮轮转的调度算法，其基本思想是:
- 设置多级就绪队列，各级队列优先级从高到低，时间片从小到大



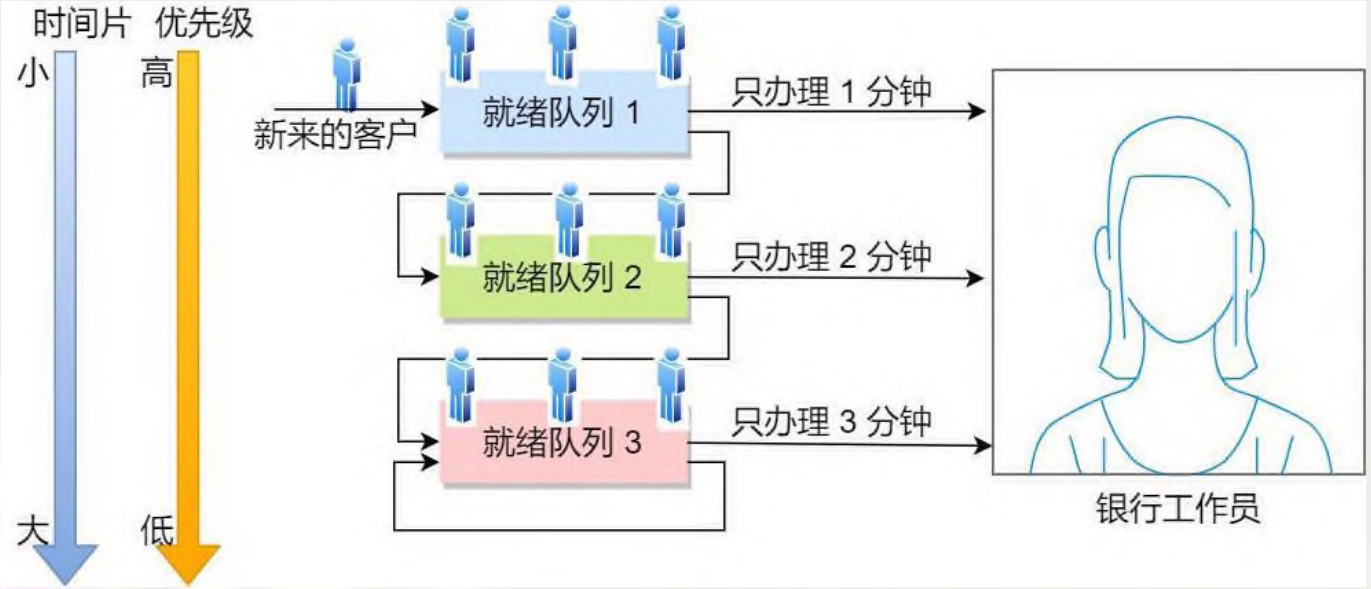
考点二: 调度算法

- 多级反馈队列调度算法是基于时间片轮轮转的调度算法，其基本思想是:
- 新进程到达时先进入第1级队列，按FCFS原则排队等待被分配时间片。若用完时间片进程还未结束，则进程进入下一级队列队尾。如果此时已经在最下级的队列，则重新放回最下级队列队尾



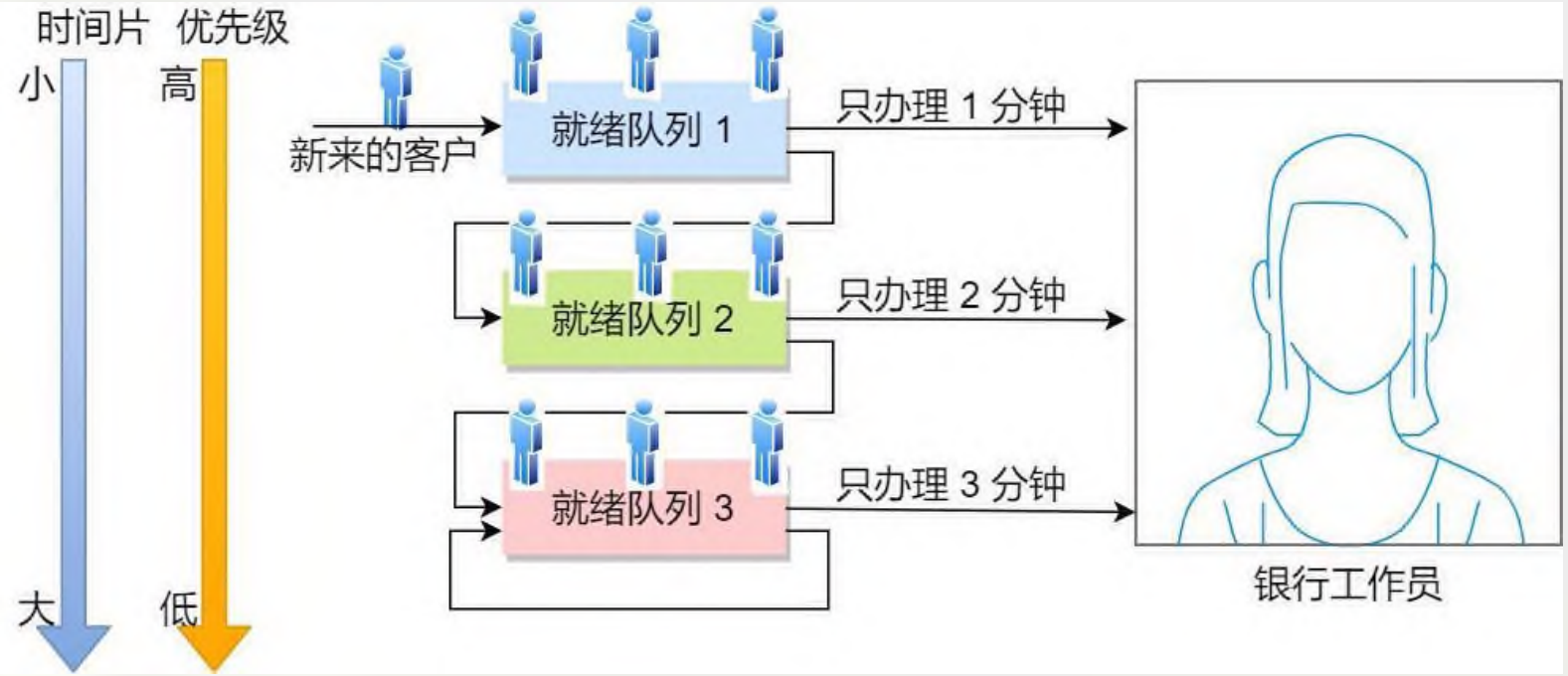
考点二: 调度算法

- 多级反馈队列调度算法是基于时间片轮轮转的调度算法，其基本思想是:
- 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进程的处理机，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。



考点二: 调度算法

- 多级反馈队列调度算法是基于时间片轮轮转的调度算法，其基本思想是:
- 只有第 k 级队列为空时，才会为 k+1 级队头的进程分配时间片被抢占处理机的进程重新放回原队列队尾



考点二: 调度算法

- 多级反馈队列调度算法
  - 动态调整进程优先级和时间片大小
  - 实现
    - 设置多个就绪队列，各队列优先级不同
      - 队列中按先来先服务
    - 各队列中进程执行时间片的大小不同
      - 优先级越高，时间片越短
    - 新进程进入内存，放入第1级队列末尾
      - 每次时间片结束，未完成都放入下一级队列末尾
    - 仅当第1级队列为空时，才调度第2级队列中的进程
  - 只用于进程调度

考点二: 调度算法

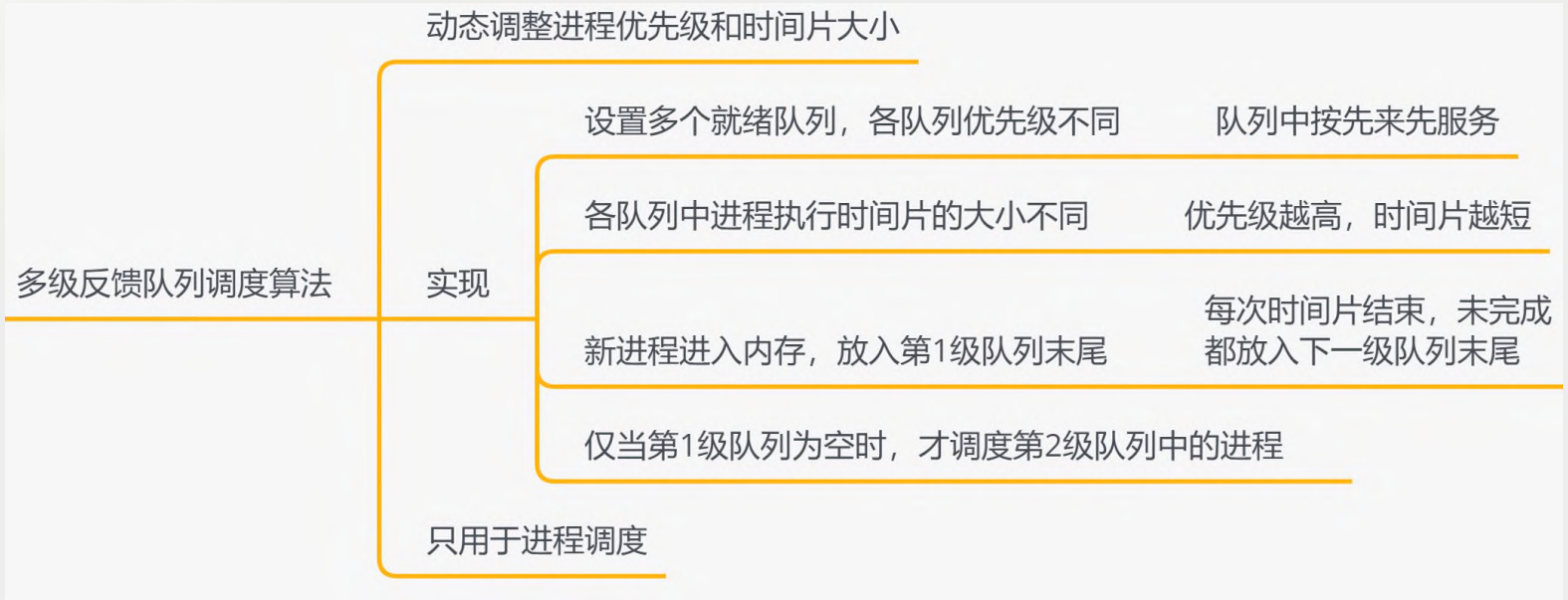
调度算法	作业调度	进程调度	优先级	抢占式	饥饿
先来先服务	√	√	等待时间	X	X
短作业优先	√	√	运行时间	√ / X	√
优先级调度	√	√	优先数	√ / X	√
高响应比优先	√	X	响应比	√ / X	X
时间片轮转	X	√	轮询模式	√	X
多级反馈队列	X	√	队列位置	√	X

【牛刀小试】

1．【吉林大学 2013】多级反馈进程调度算法不具备的特性是(　)。

A．资源利用率高　　B．响应速度快

C．系统开销小　　D．并行度高



1．【吉林大学 2013】多级反馈进程调度算法不具备的特性是(　)。

A．资源利用率高　　B．响应速度快

C．系统开销小　　D．并行度高

A【**解析**】多级反馈队列调度算法的特性:

(1)对各类型进程相对公平(FCFS的优点);

(2)每个新到达的进程都可以很快得到响应（时间片轮转调度算法的优点）；

(3)短进程只需要较少的时间就可以完成(SPF的优点);

(4)不必实现估计进程时间（避免用户作假）；

(5)可以灵活地设置对各类进程的偏好程度（优先级算法的优点）；

(6)系统开销小，因此选择A。

2．以下算法与作业运行时间有关的是(　)。

A．优先级调度　　B．时间片轮转

C．短作业优先　　D．先来先服务

调度算法	作业调度	进程调度	优先级	抢占式	饥饿
先来先服务	√	√	等待时间	X	X
短作业优先	√	√	运行时间	√ / X	√
优先级调度	√	√	优先数	√ / X	√
高响应比优先	√	X	响应比	√ / X	X
时间片轮转	X	√	轮询模式	√	X
多级反馈队列	X	√	队列位置	√	X

2．以下算法与作业运行时间有关的是(　)。

A．优先级调度　　B．时间片轮转

C．短作业优先　　D．先来先服务

C【**解析**】优先级调度和作业的运行时间无关；

时间片轮转每个时间片都是一样的，和作业的预期执行时间并无关联；

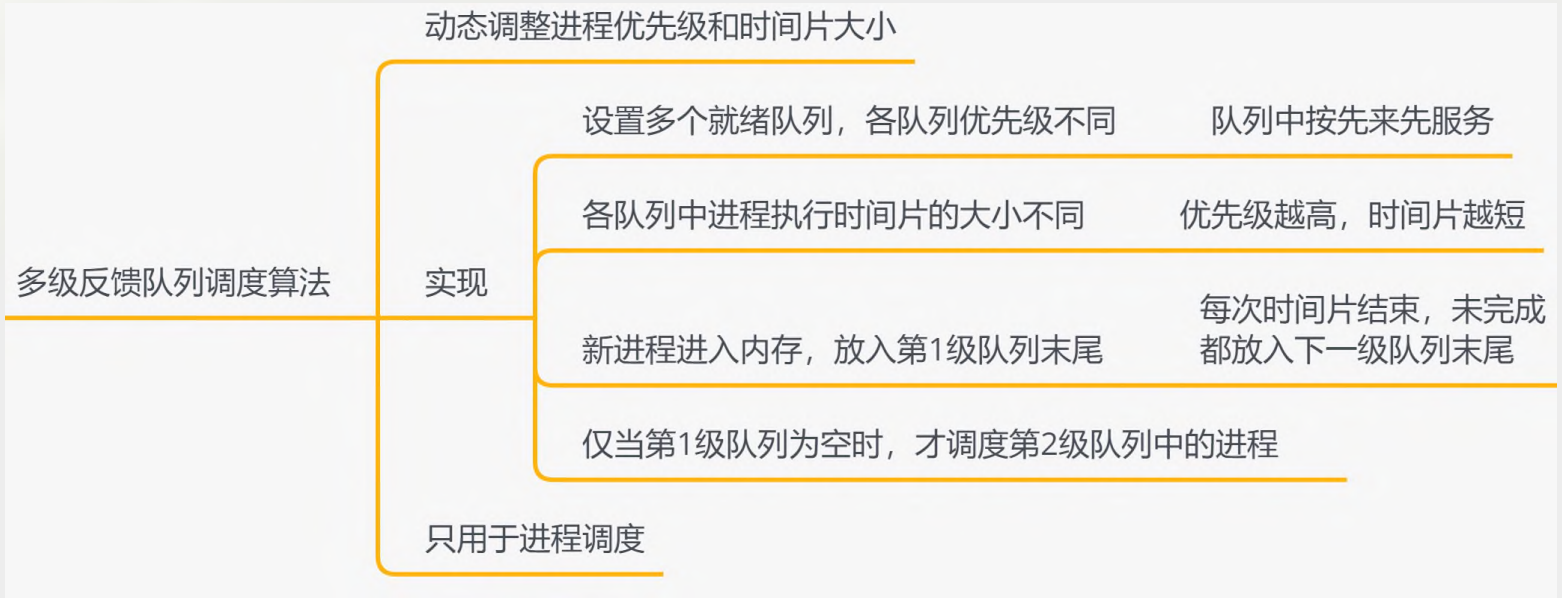
短作业和预期执行时间有关，有比较时间大小的过程；

先来先服务是就绪队列顺序问题，和时间无关，因此选择C。

3．(　)进程调度算法综合考虑了CPU密集型进程和I/O密集型进程。

A．时间片轮转　　B．优先级

C．多重队列　　D．FIFO



3．(　)进程调度算法综合考虑了CPU密集型进程和I/O密集型进程。

A．时间片轮转　　B．优先级

C．多重队列　　D．FIFO

C【**解析**】多重队列通过将进程分类，在系统中设置若干队列进行进程调度，综合考虑了CPU密集型进程和I/O密集型进程。

4．下列关于剥夺式系统的说法中，正确的是(　)。

A．若系统采用时间片轮转调度进程，则系统采用的是不可剥夺式调度

B．若由于某种事件引起调度，则该系统是剥夺式调度

C．实时系统通常采用剥夺式调度

D．在剥夺式系统中，进程的周转时间较非剥夺式系统可预见

C【**解析**】

- 非剥夺方式：分派程序一旦把处理机分配给某进程便让它一直运行下去，直到进程完成或发生某事件而阻塞时，才把处理机分配给另一个进程。即使在就绪队列有优先级高于当前执行进程时，当前进程仍将占用处理机，直到该进程自己因调用原语操作或等待I/O而进入阻塞、睡眠状态，或时间片用完时才重新发生调度让出处理机。

4．下列关于剥夺式系统的说法中，正确的是(　)。

A．若系统采用时间片轮转调度进程，则系统采用的是不可剥夺式调度

B．若由于某种事件引起调度，则该系统是剥夺式调度

C．实时系统通常采用剥夺式调度

D．在剥夺式系统中，进程的周转时间较非剥夺式系统可预见

C【**解析**】

- 剥夺方式：当一个进程正在运行时，系统可以基于某种原则，剥夺已分配给它的处理机，将之分配给其他进程。剥夺原则：优先权原则、短进程优先原则、时间片原则。实时系统通常采用剥夺式调度，时间片轮转法是一种运行时间一到就剥夺进程处理器使用权的剥夺式调度。

5．有5个批处理作业几乎同时到达，其预计运行时间分别为10,6,2,4,8，其优先级（由外部设定）分别为3,5,2,1,4，这里5为最高优先级。以下各种调度算法中，平均周转时间为14的是(　)调度算法（同一时刻只有一个作业运行）。

A．时间片轮转（时间片大小为2）　　B．优先级

C．先来先服务（按照顺序10,6,2,4,8）　　D．短作业优先

D【解析】

- 时间片轮转（时间片大小为2）平均周转时间：(30+22+6+16+28)/5=20.4；
- 优先级平均周转时间：(6+14+24+26+30)/5=20；
- 先来先服务（按照顺序10,6,2,4,8）平均周转时间：(10+16+18+22+30)/5=19.2；
- 短作业优先平均周转时间：(2+6+12+20+30)/5=14，因此选择D。

6．针对以下的每种情况，选择调度算法。为了照顾紧急作业用户，应采用(　　)；为了能实现多终端人机交互，应采用(　)；为了能使短作业、长作业和交互作业用户都满意，应采用(　)。

Ⅰ．剥夺式优先级调度算法　Ⅱ．时间片轮转调度算法

Ⅲ．多级反馈队列调度算法　Ⅳ．先到先服务调度算法

A．Ⅰ、Ⅱ、Ⅲ　　B．Ⅳ、Ⅱ、Ⅲ　　C．Ⅰ、Ⅱ、Ⅳ　　D．Ⅰ、Ⅳ、Ⅲ

调度算法	作业调度	进程调度	优先级	抢占式	饥饿
先来先服务	√	√	等待时间	X	X
短作业优先	√	√	运行时间	√/ X	√
优先级调度	√	√	优先数	√/ X	√
高响应比优先	√	X	响应比	√/ X	X
时间片轮转	X	√	轮询模式	√	X
多级反馈队列	X	√	队列位置	√	X

6．针对以下的每种情况，选择调度算法。为了照顾紧急作业用户，应采用(　　)；为了能实现多终端人机交互，应采用(　)；为了能使短作业、长作业和交互作业用户都满意，应采用(　)。

Ⅰ．剥夺式优先级调度算法　　Ⅱ．时间片轮转调度算法

Ⅲ．多级反馈队列调度算法　　Ⅳ．先到先服务调度算法

A．Ⅰ、Ⅱ、Ⅲ　　B．Ⅳ、Ⅱ、Ⅲ　　C．Ⅰ、Ⅱ、Ⅳ　　D．Ⅰ、Ⅳ、Ⅲ

A【**解析**】照顾紧急作业用户，即选择优先级高的作业优先调度，采用基于优先级的剥夺式调度算法；实现人机交互，要保证每个作业都能在一定时间内轮到，采用时间片轮转调度算法；为了使各种作业用户满意，要处理多级反馈，所以选择多级反馈队列调度算法。

7．进程调度算法采用固定时间片轮转调度算法，当时间片过大时，会使时间片轮转算法变为(　)调度算法。

A．高响应比优先　　B．先来先服务

C．短进程优先　　D．以上都不对

B【**解析**】时间片轮转调度算法在实际运行中是按先后顺序使用时间片，当时间片过大时，我们可以认为其大于进程需要的运行时间，即转变为先来先服务调度算法。

8．设4个作业从8:00开始，每小时到达1个（即8:00，9:00，10:00，11:00依次到达1个作业），每个作业的执行时间均为2小时，系统采用先来先服务和不抢占的调度策略，它们在一台处理器上按照单道运行，则10:00到达的那个作业的响应比为(　)，整个系统的平均周转时间为(　)小时。

A．2，1　　B．2，3.5　　C．3，3.5　　D．3，5

B【**解析**】采用先来先服务和不抢占的调度策略，响应比=（等待时间+运行时间）/运行时间，10:00到达的那个作业开始时间为12:00，等待时间为2，响应比=(2+2)/2=2。周转时间=完成时间-到达时间，

8:00的作业的周转时间=10-8=2，

9:00的作业的周转时间=12-9=3，

10:00的作业的周转时间=14-10=4，

11:00的作业的周转时间=16-11=5，平均周转时间=(2+3+4+5)/4=3.5，因此选择B。

9．假设系统中所有进程同时到达，则使进程平均周转时间最短的是(　　)调度算法。

A．先来先服务　　B．短进程优先

C．时间片轮转　　D．优先级

B【**解析**】如果多个进程同时到达系统，则平均周转时间最短的进程调度算法是短进程优先调度算法，因此选择B。

10．在进程调度算法中，对短进程不利的是(　)。

A．短进程优先调度算法　　B．先来先服务调度算法

C．高响应比优先调度算法　D．多级反馈队列调度算法

B【**解析**】在先来先服务调度算法中，若一个长进程（作业）先到达系统，则会使后面许多短进程（作业）等待很长时间，因此对短进程（作业）不利。

11．下列调度算法中，(　)调度算法是绝对可抢占的。

A．先来先服务　B．时间片轮转

C．优先级　　D．短进程优先

调度算法	作业调度	进程调度	优先级	抢占式	饥饿
先来先服务	√	√	等待时间	X	X
短作业优先	√	√	运行时间	√ / X	√
优先级调度	√	√	优先数	√ / X	√
高响应比优先	√	X	响应比	√ / X	X
时间片轮转	X	√	轮询模式	√	X
多级反馈队列	X	√	队列位置	√	X

11．下列调度算法中，(　)调度算法是绝对可抢占的。

A．先来先服务　B．时间片轮转

C．优先级　　D．短进程优先

B【**解析**】时间片轮转调度算法是按固定的时间配额来运行，时间一到不管是否完成，当前的进程必须撤下，调度新的进程，因此它是由时间配额决定的、是绝对可抢占的。

12．现在有三个同时到达的作业J_1，J_2和J_3，它们的执行时间分别是T_1，T_2，T_3，且$T_1<T_2<T_3$。系统按单道方式运行且采用短作业优先调度算法，则平均周转时间是(　)。

A．$T_1+T_2+T_3$　　B．$(3T_1+2T_2+T_3)/3$

C．$(T_1+T_2+T_3)/3$　　D．$(T_1+2T_2+3T_3)/3$

B【**解析**】短作业优先调度算法：$J_1\rightarrow J_2\rightarrow J_3$，作业$J_1$的周转时间为$T_1$，作业$J_2$的周转时间为$T_1+T_2$，作业J3的周转时间为$T_1+T_2+T_3$，平均周转时间=$(T_1+T_1+T_2+T_1+T_2+T_3)/3=(3T_1+2T_2+T_3)/3$，因此选择B。

13．设有三个作业J_1，J_2和J_3，其运行时间分别是2h，5h，3h，假定它们同时到达，并在同一台处理器上以单道方式运行，则平均周转时间最小的执行顺序是(　)。

A．J_1，J_2，J_3　　B．J_3，J_2，J_1　　C．J_2，J_1，J_3　　D．J_1，J_3，J_2

D【**解析**】选项A的平均周转时间=(2+7+10)/3≈6.33，选项B的平均周转时间=(3+8+10)/3=7，选项C的平均周转时间=(5+7+10)/3≈7.33，选项D的平均周转时间=(2+5+10)/3≈5. 67，因此选择D。

14．（　)有利于CPU繁忙型的作业，而不利于I/O繁忙型的作业。

A．时间片轮转调度算法　　B．先来先服务调度算法

C．短作业（进程）优先算法　D．优先权调度算法

B【**解析**】先来先服务(FCFS)调度算法是一种最简单的调度算法，在作业调度中采用该算法时，每次调度是从后备作业队列中选择一个或多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、创建进程，然后放入就绪队列。

- FCFS调度算法比较有利于长作业，而不利于短作业。所谓CPU繁忙型的作业，是指该类作业需要大量的CPU时间进行计算，而很少请求I/O操作。I/O繁忙型的作业是指CPU处理时，需频繁地请求I/O操作。所以CPU繁忙型作业更接近于长作业，因此选择B。

15．【北京交通大学 2018】（多选）适用于分时系统的调度算法有(　)。

A．高响应比优先　　B．短进程优先

C．时间片轮转　　D．多级反馈队列

CD【**解析**】高响应比优先调度算法和短进程优先调度算法适用于批处理系统；时间片轮转调度算法和多级反馈队列调度算法适用于分时系统。因此选择CD。

16．【桂林电子科技大学　2018】为了使系统具有最高的吞吐量，作业调度算法应(　)。

A．满足所有用户　　B．设计简单一些

C．优先处理运行时间短的作业　　D．借助于进程调度

C【**解析**】短作业优先调度算法可以改善平均周转时间和平均带权周转时间，缩短作业的等待时间，提高系统的吞吐量。因此选择C。

17．【南京工业大学 2018】采用时间片轮转法进行进程调度是为了(　)。

A．多个终端都能得到系统的及时响应　B．先来先服务

C．优先级较高的进程得到及时响应　D．需要CPU最短的进程先做

A【**解析**】采用时间片轮转法进行进程调度是为了多个终端都能得到系统的及时响应，因此选择A。

18．【中国计量大学　2020】在采用高响应比优先调度算法中，如果所有进程都同时到达（即等待时间一样），则此时的优先权调度算法实际上和(　　)相同。

A．先来先服务调度算法　　B．短作业优先调度算法

C．时间片轮转调度算法　　D．以上都不是

B【**解析**】响应比 =（等待时间+要求服务时间）/ 要求服务时间,即RR=（w+s）/s=1+w/s，

在响应比最高者优先的作业调度算法中，当各个作业等待时间相同时，计算时间短的作业将得到优先调度，因此选择B。

【大显身手】

1．【广东工业大学　2014】假设在单道批处理环境下有5个作业，它们的到达时间和服务时间如下表所示，忽略I/O以及其他开销时间，若按先来先服务(FCFS)、短作业优先(SJF)两种调度算法进行调度，请给出各个作业的完成时间、周转时间、带权周转时间、平均周转时间和平均带权周转时间。

作业	进入系统时间	估计运行时间/分钟
1	8:00	40
2	8:20	30
3	8:30	12
4	9:00	18
5	9:10	5

【解析】先来先服务(FCFS)调度顺序：作业1→作业2→作业3→作业4→作业5。

作业1完成时间：8:40，周转时间=完成时间-到达时间=8:40-8:00=40(分钟)，带权周转时间=周转时间/运行时间=40/40=1。

作业2完成时间：8:40+0:30=9:10，周转时间=完成时间-到达时间=9:10-8:20=50(分钟)，带权周转时间=周转时间/运行时间=50/30=5/3。

作业3完成时间：9:10+0:12=9:22，周转时间=完成时间-到达时间=9:22-8:30=52(分钟)，带权周转时间=周转时间/运行时间=52/12=13/3。

作业4完成时间：9:22+0:18=9:40，周转时间=完成时间-到达时间=9:40-9:00=40(分钟)，带权周转时间=周转时间/运行时间=40/18=20/9。

作业5完成时间：9:40+0:05=9:45，周转时间=完成时间-到达时间=9:45-9:10=35(分钟)，带权周转时间=周转时间/运行时间=35/5=7。

平均周转时间=(40+50+52+40+35)/5=43.4（分钟），平均带权周转时间=(1+5/3+13/3+20/9+7)/5≈3.24。

作业	进入系统时间	估计运行时间/分钟
1	8:00	40
2	8:20	30
3	8:30	12
4	9:00	18
5	9:10	5

短作业优先(SJF)调度顺序：作业1→作业3→作业2→作业5→作业4。

作业1完成时间：8:40，周转时间=完成时间-到达时间=8:40-8:00=40(分钟)，带权周转时间=周转时间/运行时间=40/40=1。

作业3完成时间：8:40+0:12=8:52，周转时间=完成时间-到达时间=8:52-8:30=22(分钟)，带权周转时间=周转时间/运行时间=22/12=11/6。

作业2完成时间：8:52+0:30=9:22，周转时间=完成时间-到达时间=9:22-8:20=62(分钟)，带权周转时间=周转时间/运行时间=62/30=31/15。

作业5完成时间：9:22+0:05=9:27，周转时间=完成时间-到达时间=9:27-9:10=17(分钟)，带权周转时间=周转时间/运行时间=17/5。

作业4完成时间：9:27+0:18=9:45，周转时间=完成时间-到达时间=9:45-9:00=45(分钟)，带权周转时间=周转时间/运行时间=45/18=5/2。

平均周转时间=(40+22+62+17+45)/5=37.2（分钟），平均带权周转时间=(1+11/6+31/15+17/5+5/2)/5 ≈ 2.16。

作业	进入系统时间	估计运行时间/分钟
1	8:00	40
2	8:20	30
3	8:30	12
4	9:00	18
5	9:10	5

2．【山东大学　2017】设有四个进程P1，P2，P3，P4，它们到达就绪队列的时间及执行时间如下表所示，若采用剥夺式短作业优先、非剥夺式短作业优先及时间片轮转法（时间片=2），分别给出各进程关于各算法的调度次序及平均等待时间。

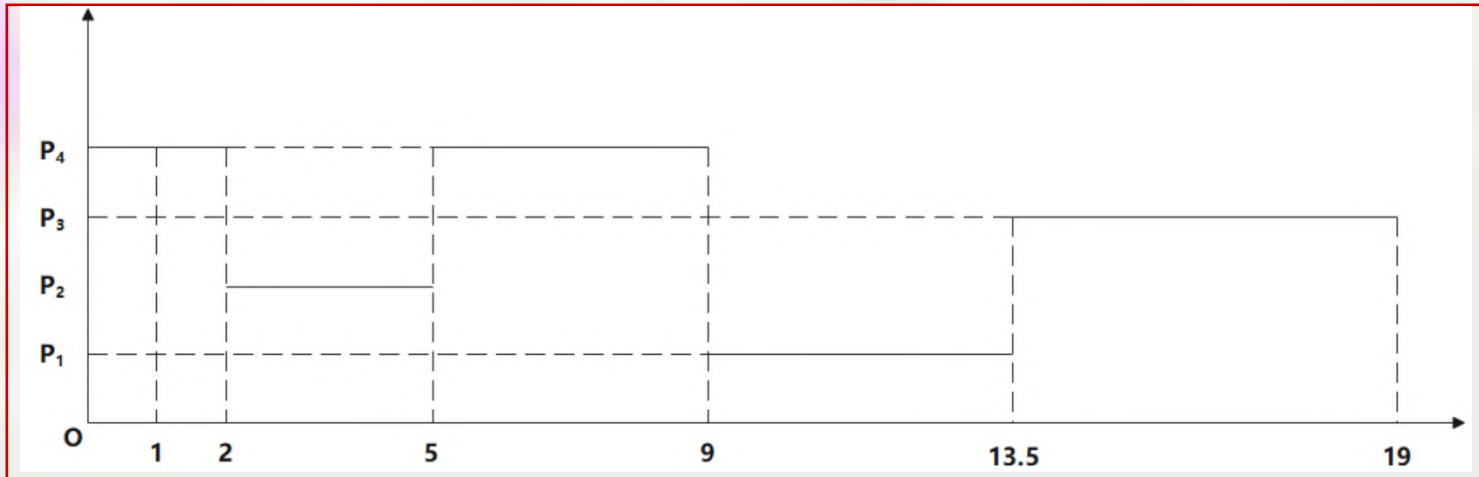
进程	到达时间	执行时间
P_1	0	6
P_2	1	5.5
P_3	2	3
P_4	3	4.5

非剥夺式短作业优先的调度顺序P1—>P3—>P4—>P2，

P1	P3	P4	P2	进程
6	3	4.5	5.5	运行时间
0	6	9	13.5	19

- 进程P1的等待时间=完成时间-到达时间-运行时间=6-0-6=0
- 进程P2的等待时间=完成时间-到达时间-运行时间=19-1-5.5=12.5，
- 进程P3的等待时间=完成时间-到达时间-运行时间=9-2-3=4
- 进程P4的等待时间=完成时间-到达时间-运行时间=13.5-3-4.5=6，
- 则平均等待时间为(0+12.5+4+6)/4=5.625；

进程	到达时间	执行时间
P_1	0	6
P_2	1	5.5
P_3	2	3
P_4	3	4.5



进程	到达时间	执行时间
P_1	0	6
P_2	1	5.5
P_3	2	3
P_4	3	4.5

- P1的周转时间是13.5
- P2的周转时间是5
- P3的周转时间是19
- P4的周转时间是9
- 则平均等待时间为(13.5+5+19+6)/4;

时间片轮转法(时间片=2)的调度顺序：

$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_1 \rightarrow P_2 \rightarrow P_4$。

P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P4
2	2	2	2	2	2	1	2	2	1.5	0.5

0 2 4 6 8 10 12 13 15 17 18.5 19

进程P1的等待时间=完成时间-到达时间-运行时间=17-0-6=11

进程P2的等待时间=完成时间-到达时间-运行时间=18.5-1-5.5=12

进程P3的等待时间=完成时间-到达时间-运行时间=13-2-3=8

进程P4的等待时间=完成时间-到达时间-运行时间=19-3-4.5=11.5,

则平均等待时间为(11+12+8+11.5)/4=10.625。

进程	到达时间	执行时间
P_1	0	6
P_2	1	5.5
P_3	2	3
P_4	3	4.5

3．【重庆理工大学　2014】有四个进程P1，P2，P3，P4，它们进入就绪队列的先后顺序为P1，P2，P3，P4，它们的优先级和需要的处理时间如下表所示。假定这四个进程在执行过程中不会发生等待事件，忽略进程调度所花费的时间，从某个时刻开始进程调度，请回答下面的问题。

进程	需要的处理时间	优先级
P_1	8	3
P_2	6	1
P_3	22	5
P_4	4	4

(1)采用“先来先服务”调度算法时，写出进程的执行顺序，计算各进程在就绪队列中等待的时间以及平均等待时间。

(1) 它们进入就绪队列的先后顺序为P1，P2，P3，P4，

采用“先来先服务”调度算法，进程的调度顺序：$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$。

进程	P_1	P_2	P_3	P_4
运行时间	8	6	22	4

0　　8　　14　　36　　40

- 进程P_1的等待时间=完成时间-运行时间=8-8=0，
- 进程P_2的等待时间=完成时间-运行时间=14-6=8，
- 进程P_3的等待时间=完成时间-运行时间=36-22=14，
- 进程P_4的等待时间=完成时间 运行时间=40-4=36，
- 则平均等待时间=(0+8+14+36)/4=14.5。

进程	需要的处理时间	优先级
P_1	8	3
P_2	6	1
P_3	22	5
P_4	4	4

3．【重庆理工大学　2014】有四个进程P1，P2，P3，P4，它们进入就绪队列的先后顺序为P1，P2，P3，P4，它们的优先级和需要的处理时间如下表所示。假定这四个进程在执行过程中不会发生等待事件，忽略进程调度所花费的时间，从某个时刻开始进程调度，请回答下面的问题。

进程	需要的处理时间	优先级
P_1	8	3
P_2	6	1
P_3	22	5
P_4	4	4

(2)采用“非抢占式的优先级”调度算法时，写出进程的执行顺序，计算各进程在就绪队列中等待的时间以及平均等待时间。

采用“非抢占式的优先级”调度算法（优先级数小的优先级高），进程的调度顺序：$P_2 \rightarrow P_1 \rightarrow P_4 \rightarrow P_3$。

进程	P_2	P_1	P_4	P_3
运行时间	6	8	4	22

0　　6　　14　　18　　40

- 进程P_1的等待时间=完成时间-运行时间=8-8=0，
- 进程P_2的等待时间=完成时间-运行时间=14-6=8，
- 进程P_3的等待时间=完成时间-运行时间=36-22=14，
- 进程P_4的等待时间=完成时间 运行时间=40-4=36，
- 则平均等待时间=(0+8+14+36)/4=14.5。

进程	需要的处理时间	优先级
P_1	8	3
P_2	6	1
P_3	22	5
P_4	4	4

3．【重庆理工大学　2014】有四个进程P1，P2，P3，P4，它们进入就绪队列的先后顺序为P1，P2，P3，P4，它们的优先级和需要的处理时间如下表所示。假定这四个进程在执行过程中不会发生等待事件，忽略进程调度所花费的时间，从某个时刻开始进程调度，请回答下面的问题。

进程	需要的处理时间	优先级
P_1	8	3
P_2	6	1
P_3	22	5
P_4	4	4

(3)采用“时间片轮转法”调度算法时（时间片大小为4），写出进程的执行顺序，计算各进程在系统中停留的时间以及平均停留的时间。

(3)采用“时间片轮转法”调度算法（时间片大小为4），进程的调度顺序：$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3$。

进程	P_1	P_2	P_3	P_4	P_1	P_2	P_3
运行时间	4	4	4	4	4	2	18

0 4 8 12 16 20 22 40

进程	需要的处理时间	优先级
P_1	8	3
P_2	6	1
P_3	22	5
P_4	4	4

- 进程P_1的等待时间=完成时间-运行时间=20-8=12
- 进程P_2的等待时间=完成时间-运行时间=22-6=16
- 进程P_3的等待时间=完成时间 运行时间=40-22=18
- 进程P_4的等待时间=完成时间运行时间=16-4=12
- 则平均等待时间=(12+16+18+12)/4=14.5。

谢谢大家

考点三:

上下文切换机制

考点框架

- CPU上下文
- 系统调用上下文切换
- 进程上下文切换
- 线程上下文切换
- 中断上下文切换

考点三: 上下文切换机制

CPU上下文

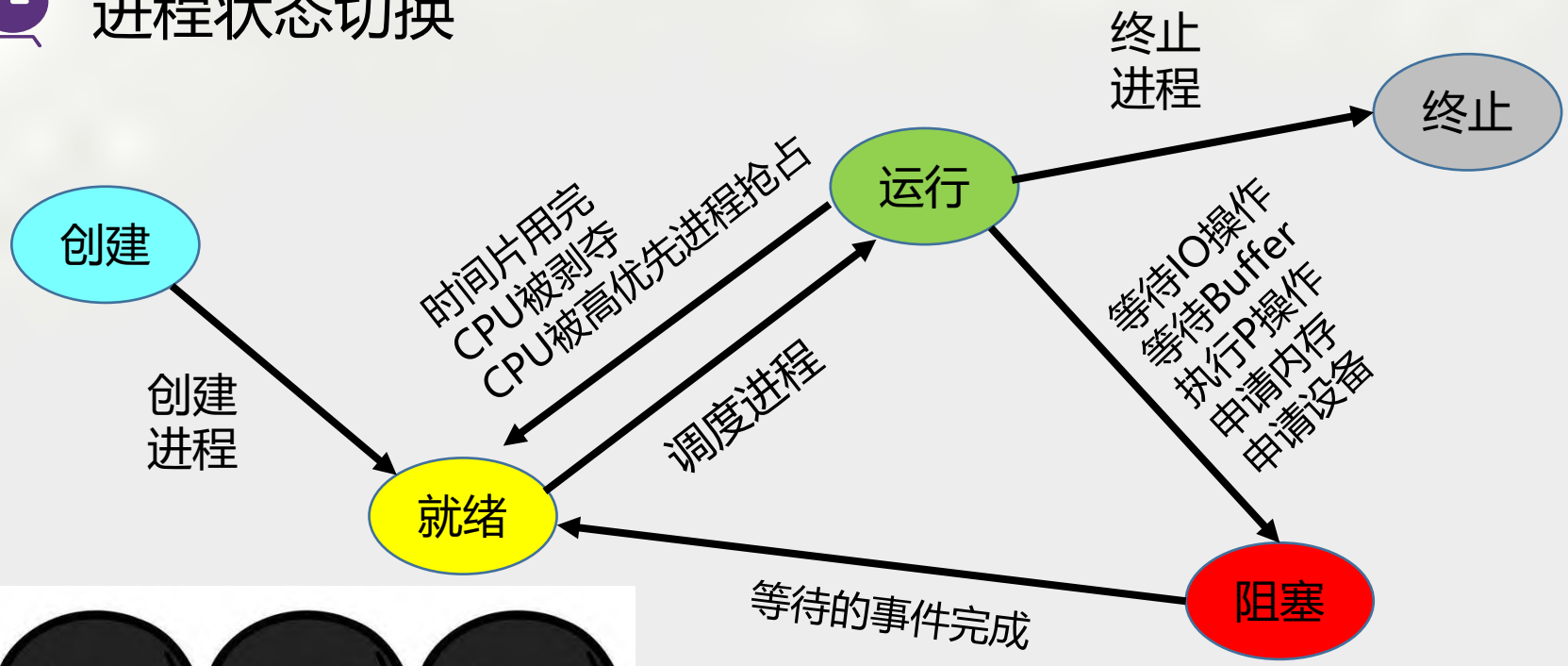
考点三: 上下文切换机制

共享性

并发性

考点三: 上下文切换机制

进程状态切换



依赖于调度

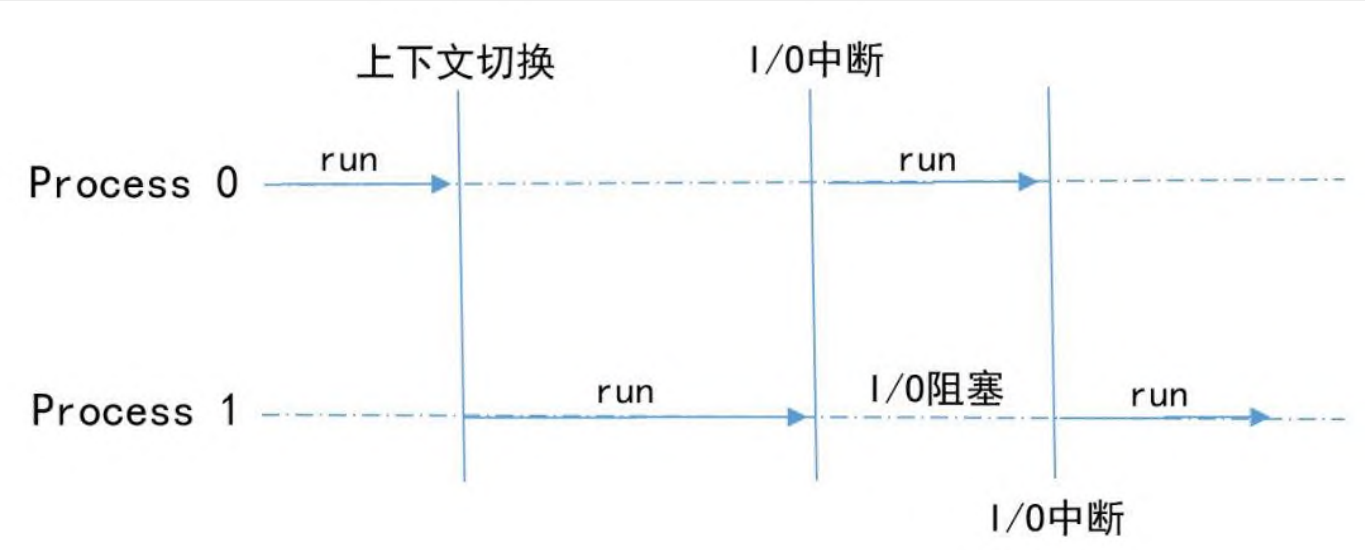
考点三: 上下文切换机制

进程控制块

考点三: 上下文切换机制

CPU上下文

- CPU上下文：CPU是被所有进程共享的，CPU的状态指的是CPU在运行某个进程时某些寄存器和程序计数器。
- CPU 寄存器和程序计数器就是 CPU 上下文，因为它们都是 CPU 在运行任何任务前，必须的依赖环境。



考点三: 上下文切换机制

CPU上下文

- CPU上下文：CPU是被所有进程共享的，CPU的状态指的是CPU在运行某个进程时某些寄存器和程序计数器。
- CPU 寄存器和程序计数器就是 CPU 上下文，因为它们都是 CPU 在运行任何任务前，必须的依赖环境。
- CPU 上下文切换，就是先把前一个任务的 CPU 上下文（也就是 CPU 寄存器和程序计数器）保存起来，然后加载新任务的上下文到这些寄存器和程序计数器，最后再跳转到程序计数器所指的新位置，运行新任务。
- 被保存下来的上下文会存储在系统内核中，等任务重新调度执行时再次加载进来。

考点三: 上下文切换机制

CPU上下文

CPU的上下文切换分为几种场景：

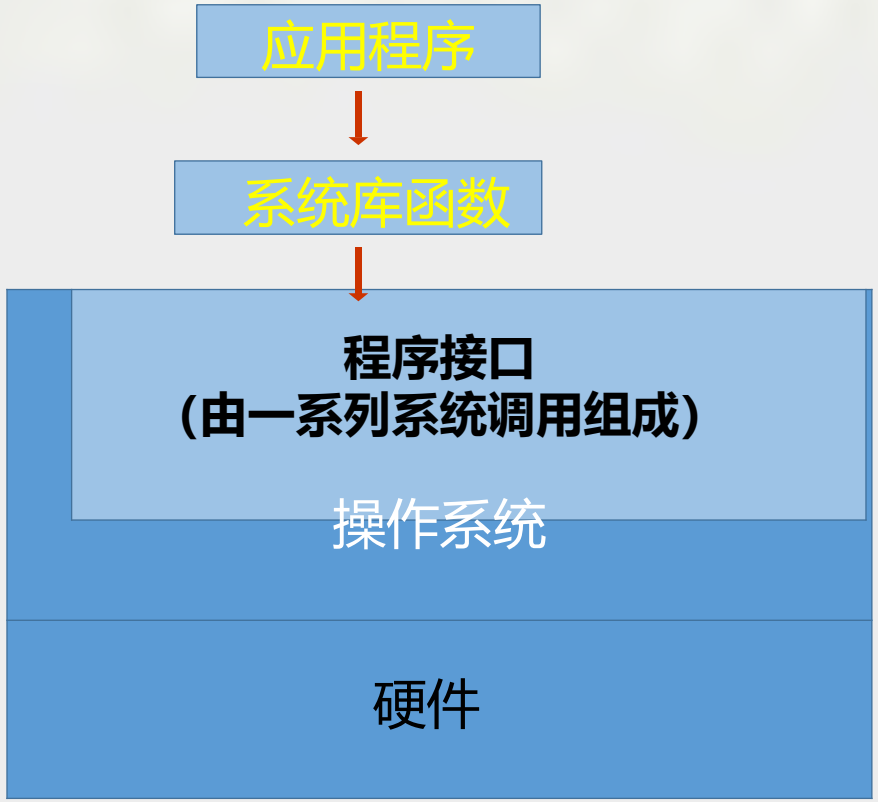
- 进程上下文切换
- 系统调用上下文切换
- 线程上下文切换
- 中断上下文切换

考点三: 上下文切换机制

系统调用上下文切换

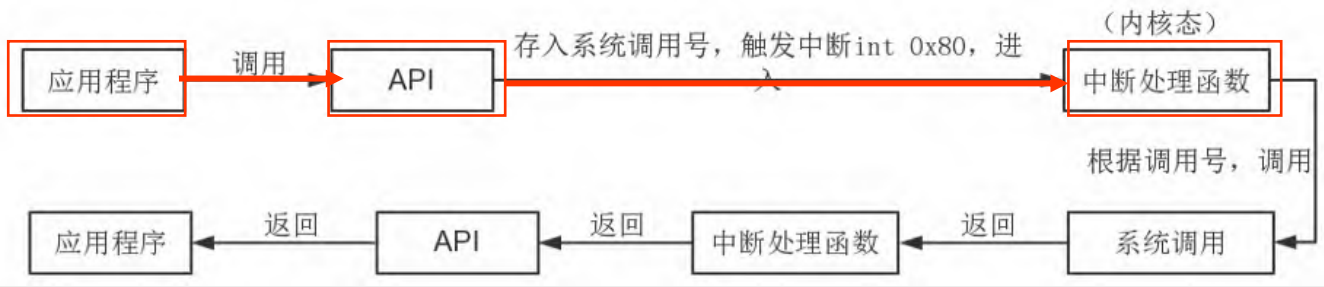
考点三: 上下文切换机制

- 系统调用方式：OS提供了一组系统调用，用户可在自己的应用程序中通过相应的系统调用，来操纵计算机。



考点三: 上下文切换机制

系统调用过程

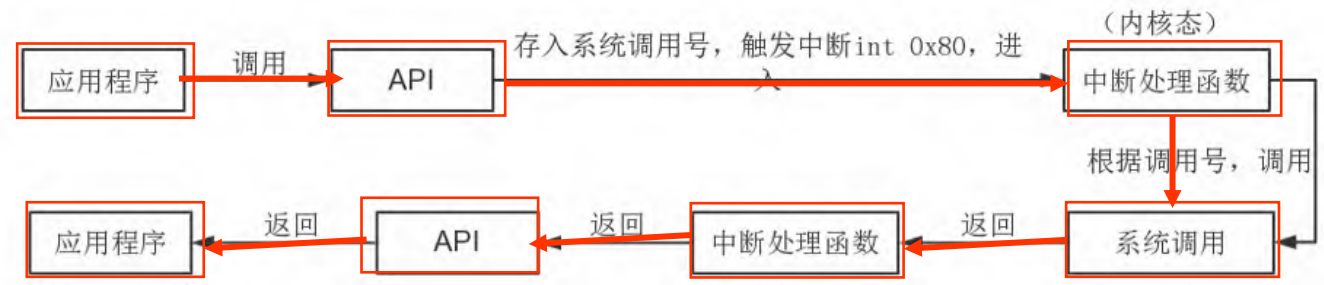


① 应用程序通过调用API，发起系统调用（请注意，此时程序处于用户态）。

② 当系统调用发生时，处理器通过一种特殊的机制（此处的中断编号是int 0x80），通常是中断或者异常处理，把控制流程转移到内核态的一些特定的位置（（请注意，通过中断，已经从用户态切换到了内核态）。处理器模式转变成特权模式。

考点三: 上下文切换机制

系统调用过程



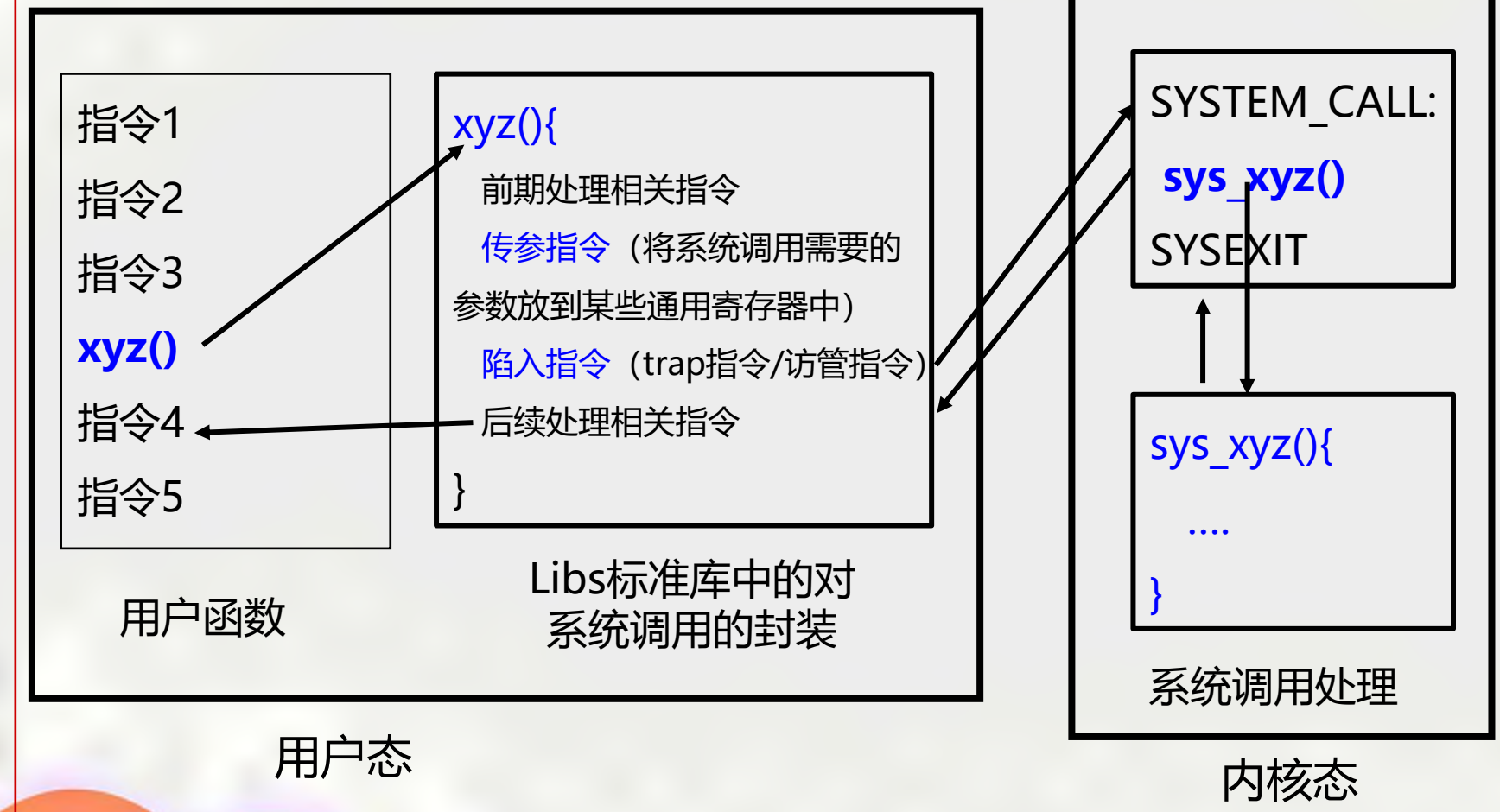
④ 由内核程序执行被请求的功能代码。这个功能代码代表着对一段标准程序段的执行，用以完成所请求的功能。

⑤ 处理结束之后，程序状态恢复系统调用之前的现场；把运行模式从特权模式恢复成为用户模式。

⑥ 最后通过API将控制权转移回原来的用户程序。

考点三: 上下文切换机制

系统调用过程



考点三: 上下文切换机制

系统调用过程

- 系统调用会将CPU从用户态切换到核心态，以便 CPU 访问受到保护的内核内存。
- 系统调用的过程会发生 CPU 上下文的切换，CPU 寄存器里原来用户态的指令位置，需要先保存起来。
- 为了执行内核态代码，CPU 寄存器需要更新为内核态指令的新位置。
- 跳转到内核态运行内核任务。
- 系统调用结束后，CPU 寄存器需要恢复原来保存的用户态，然后再切换到用户空间，继续运行进程。
- 一次系统调用的过程，其实是发生了两次 CPU 上下文切换

考点三: 上下文切换机制

系统调用过程

- 一次系统调用的过程，其实是发生了两次 CPU 上下文切换。（用户态-内核态-用户态）
- 注意: 系统调用过程中，并不会涉及到虚拟内存等进程用户态的资源，也不会切换进程。这跟我们通常所说的进程上下文切换是不一样的。进程上下文切换，是指从一个进程切换到另一个进程运行；而系统调用过程中一直是同一个进程在运行。

考点三: 上下文切换机制

进程上下文切换

考点三: 上下文切换机制

- 进程的上下文：进程的物理实体（代码和数据等）和支持运行的环境合称为进程的上下文。进程的上下文包括用户级上下文和系统级上下文。
- 用户级上下文：由用户的程序块、数据块、运行时的堆和用户栈（统称为用户堆栈）等组成的用户空间信息被称为用户级上下文。
- 系统级上下文：由进程标识信息、进程现场信息、进程控制信息（包含进程表、页表、打开文件表等）和系统内核栈等组成的内核空间信息被称为系统级上下文。

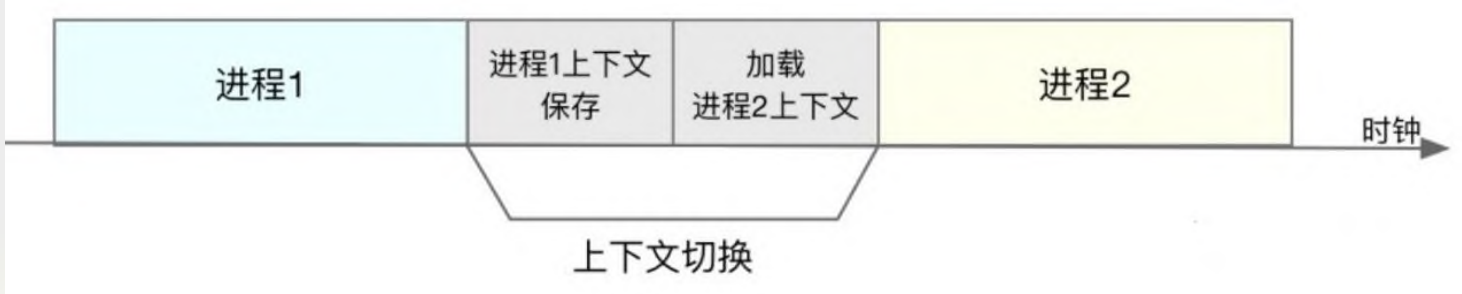
考点三: 上下文切换机制

上下文信息

系统级上下文	进程标识信息
	进程现场信息
	进程控制信息
	系统内核级
用户级上下文	用户堆栈
	用户数据块
	用户程序块
	共享地址空间

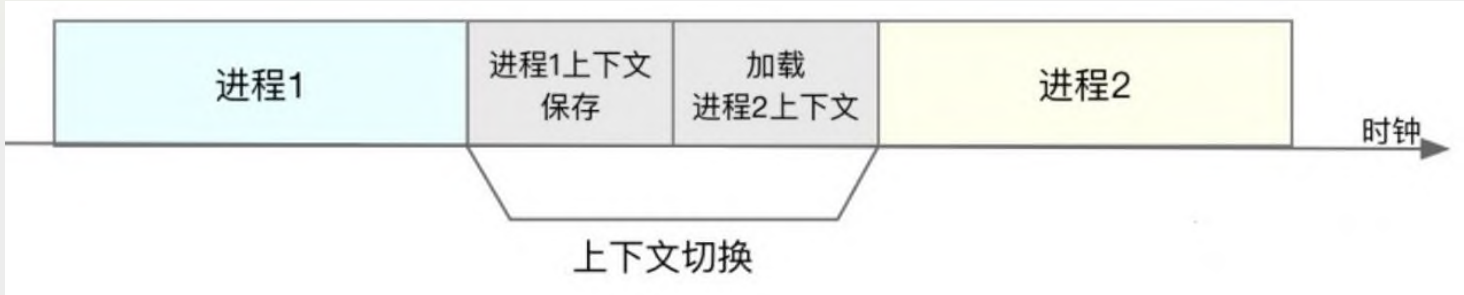
考点三: 上下文切换机制

- 进程是由内核来管理和调度的，进程的切换只能发生在内核态。所以，进程的上下文不仅包括了虚拟内存、栈、全局变量等用户空间的资源，还包括了内核堆栈、寄存器等内核空间的状态。
- 进程的上下文切换就比系统调用时多了一步：在保存当前进程的内核状态和 CPU 寄存器之前，需要先把该进程的虚拟内存、用户栈等保存下来；而加载下一个进程的内核态后，还需要加载这个进程的虚拟内存和用户栈



考点三: 上下文切换机制

- 只有在进程调度的时候，才需要切换上下文。
- Linux 为每个 CPU 都维护了一个就绪队列，将活跃进程（即正在运行和正在等待 CPU 的进程）按照优先级和等待 CPU 的时间排序，然后选择最需要 CPU 的进程，也就是优先级最高和等待 CPU 时间最长的进程来运行。



考点三: 上下文切换机制

需要注意的是：进程调度（低级调度），就是按照某种算法从就绪队列中选择一个进程为其分配处理机。

需要进行进程调度与切换的情况：

（I）当前运行的进程主动放弃处理机

①进程正常终止

②运行过程中发生异常而终止

③进程主动请求阻塞（如 等待I/O）

（II）当前运行的进程被动放弃处理机

①分给进程的时间片用完

②有更紧急的事需要处理（如 I/O中断）

③有更高优先级的进程进入就绪队列

（III）不能进行进程调度与切换的情况

①在处理中断的过程中。中断处理过程复杂，与硬件密切相关，很难做到在中断处理过程中进行进程切换

②进程在操作系统内核程序临界区中

③在原子操作过程中（原语）。原子操作不可中断，要一气呵成（如修改PCB中进程状态标志，并把PCB放到相应队列）

考点三: 上下文切换机制

线程上下文切换

考点三: 上下文切换机制

- 线程是调度的基本单位，而进程则是资源拥有的基本单位。
- 所谓内核中的任务调度，实际上的调度对象是线程；而进程只是给线程提供了虚拟内存、全局变量等资源。
- 当进程只有一个线程时，可以认为进程就等于线程
- 当进程拥有多个线程时，这些线程会共享进程的虚拟内存和全局变量等资源。这些资源在上下文切换时是不需要修改的。

考点三: 上下文切换机制

- 线程也有自己的私有数据，比如栈和寄存器等，这些在上下文切换时也是需要保存的。线程的上下文切换其实就可以分为两种情况：
- 两个线程属于不同进程，因为资源不共享，切换过程和进程上线文切换一样
- 两个线程属于同一个进程，只需要切换线程的私有数据、寄存器等不共享的数据

考点三: 上下文切换机制

中断上下文切换

考点三: 上下文切换机制

- 中断会打断进程的政策调度和执行，转而调用中断处理程序，响应设备事件，而在打断其他进程时，就需要将进程当前的状态保存下来，这样再中断结束后，进程仍然可以从原来的状态恢复运行。
- 中断上下文就可以理解为硬件传递过来的这些参数和内核需要保存的一些环境，主要是被中断的进程的环境。

考点三: 上下文切换机制

- 中断上下文切换并不涉及到进程的用户态，即便中断过程打断了一个正在处理用户态的进程，也不需要保存和恢复这个进程的虚拟内存，全局变量等用户态资源。
- 在发生中断时，内核就在被中断进程的上下文中（就相当于在同一个进程执行了CPU上下文切换，从该进程的用户态切换到内核态），在内核态下执行中断服务例程。但同时会保留所有需要用到的资源，以便中继服务结束时能恢复被中断进程 的执行。

考点三: 上下文切换机制

- 对同一个CPU来说，中断处理比进程拥有更高的优先级，所以中断上下文切换并不会与进程上下文切换同时发生，同样由于中断会打断正常进程的调度和执行，所以大部分中断程序都短小精悍，以便快速执行结束
- 运行在进程上下文的内核代码是可以被抢占的。但是一个中断上下文，通常都会始终占有CPU，不可以被打断。

考点三: 上下文切换机制

运行在中断上下文的代码就要受一些限制，不能做下面的事情：

- 睡眠或者放弃CPU：这样做的后果是灾难性的，因为内核在进入中断之前会关闭进程调度，一旦睡眠或者放弃CPU，这时内核无法调度别的进程来执行，系统就会死掉；
- 尝试获得信号量：如果获得不到信号量，代码就会睡眠，会产生和上面相同的情况；
- 执行耗时的任务：中断处理应该尽可能快，因为内核要响应大量服务和请求，中断上下文占用CPU时间太长会严重影响系统功能；
- 访问用户空间的虚拟地址：因为中断上下文是和特定进程无关的，它是内核代表硬件运行在内核空间，所以在中断上下文无法访问用户空间的虚拟地址。

考点三: 上下文切换机制

- 注意: 中断上下文切换也会消耗CPU，切换次数过多会耗费大量CPU，降低系统整体性能

谢谢大家