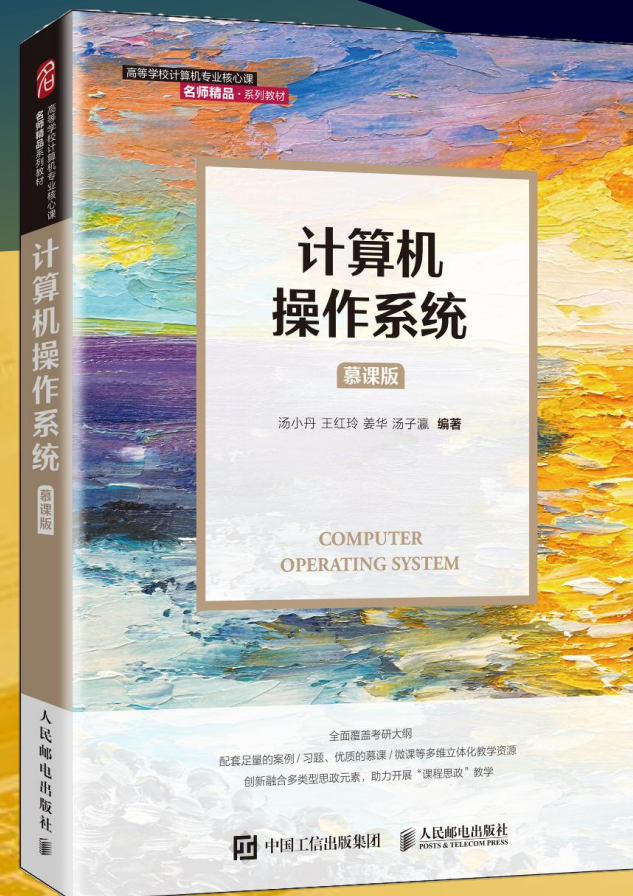




经典教材《计算机操作系统》**最新版**

第3章 处理机调度与死锁

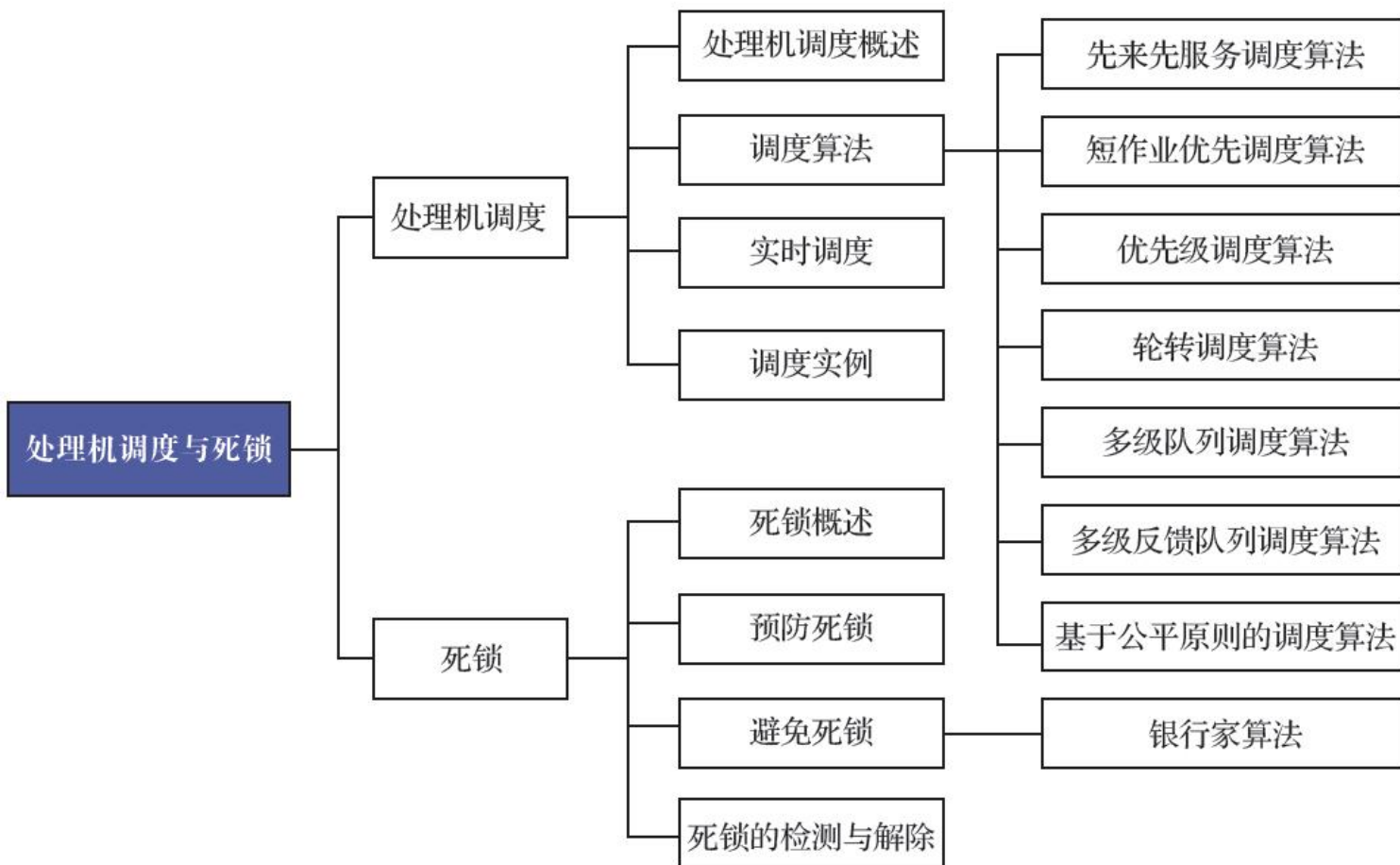
主讲教师：李灵慧













第3章知识导图

第1章	操作系统引论
第2章	进程的描述与控制
第3章	处理机调度与死锁
第4章	进程同步
第5章	存储器管理
第6章	虚拟存储器
第7章	输入/输出系统
第8章	文件管理
第9章	磁盘存储器管理
第10章	多处理机操作系统
第11章	虚拟化和云计算
第12章	保护和安全





内容导航:

-  3.1 处理机调度概述
-  3.2 调度算法
-  3.3 实时调度
-  3.4 Linux进程调度
-  3.5 死锁概述
-  3.6 预防死锁
-  3.7 避免死锁
-  3.8 死锁的检测与解除

第3章 处理机调度与死锁

1. 在多道程序系统中，一个作业从提交到执行，通常都要经历**多级调度**

➢ 如高级调度、低级调度、中级调度以及I / O调度等

2. 系统的**运行性能**在很大程度上取决于调度

➢ 如吞吐量的大小、周转时间的长短、响应的及时性等

3. 调度是多道程序系统的关键

CPU资源管理——多道程序设计面临的挑战

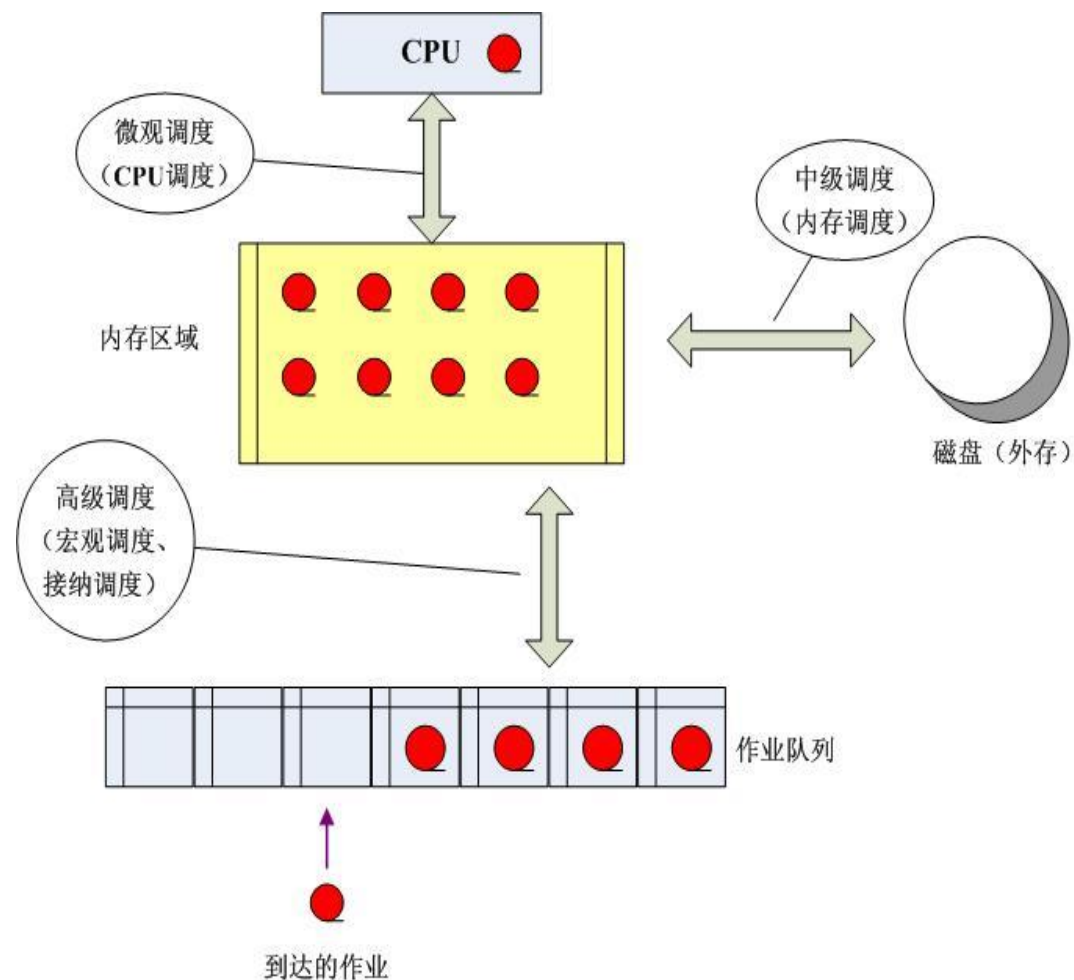
- ❑ 批处理系统：如何安排内存中多个作业的运行顺序？
- ❑ 交互式系统：如何更好应对不同的交互式请求？
- ❑ 实时系统：如何保证实时服务的高质量？

进程调度——有效的管理CPU资源

- ❑ When：何时进行进程调度？
- ❑ How：遵循何种规则完成调度？
- ❑ What：调度过程中需要完成哪些工作？

进程调度的级别

- ❑ 高级调度：也称宏观调度，决定哪些程序可以进入系统
- ❑ 中级调度：也称内存调度，决定内存中程序的位置和状态
- ❑ 低级调度：也称微观调度，决定CPU资源在就绪进程间的分配





高级调度（长程调度/作业调度）



低级调度（短程调度/进程调度）



中级调度（中程调度/内存调度）



调度对象：作业（配有作业说明书）

任务：根据某种算法，决定将外存上处于后备队列中的作业调入内存，并为它们创建进程和分配必要的资源。然后，将新创建的进程排在就绪队列上等待调度。

主要用于多道批处理系统中，在分时系统、实时系统、PC机上不存在这种调度。



引入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；使那些**暂时不能运行**的进程不再占用宝贵的内存资源，将它们调至外存上去等待，把此时的进程状态称为就绪驻外存状态或**挂起**状态；



当这些**进程**具备运行条件、且内存又稍有空闲时，由中级调度来决定把就绪进程，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；



即“**对换**”功能；短期**调整系统负荷**，平顺系统操作

将在第5章 存储器管理中介绍



调度对象：进程

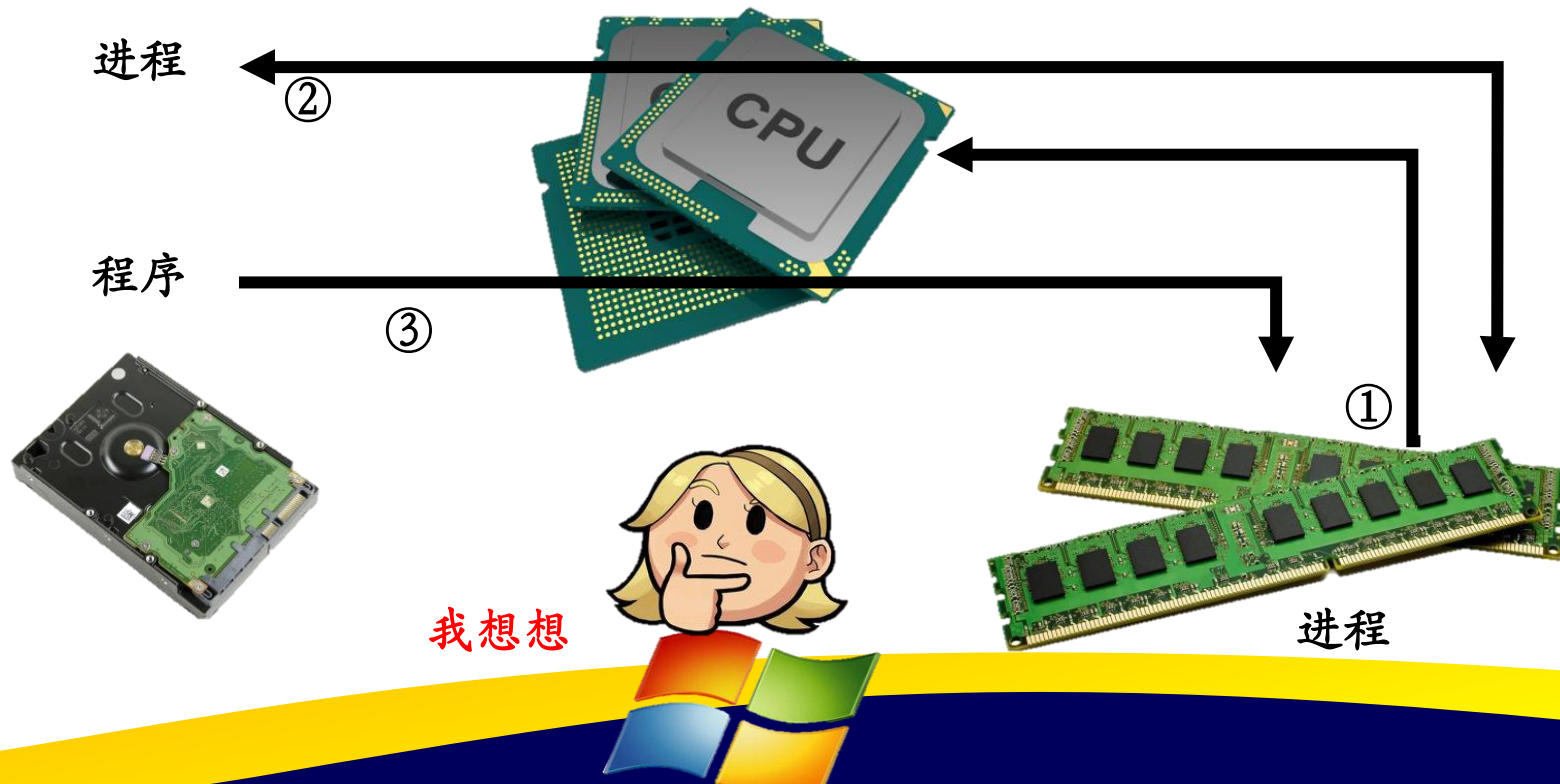


根据某种调度算法，决定就绪队列中的哪个进程应**获得处理机**



应用在于多道批处理、分时和实时OS，最基本的进程管理功能，调度频率很高

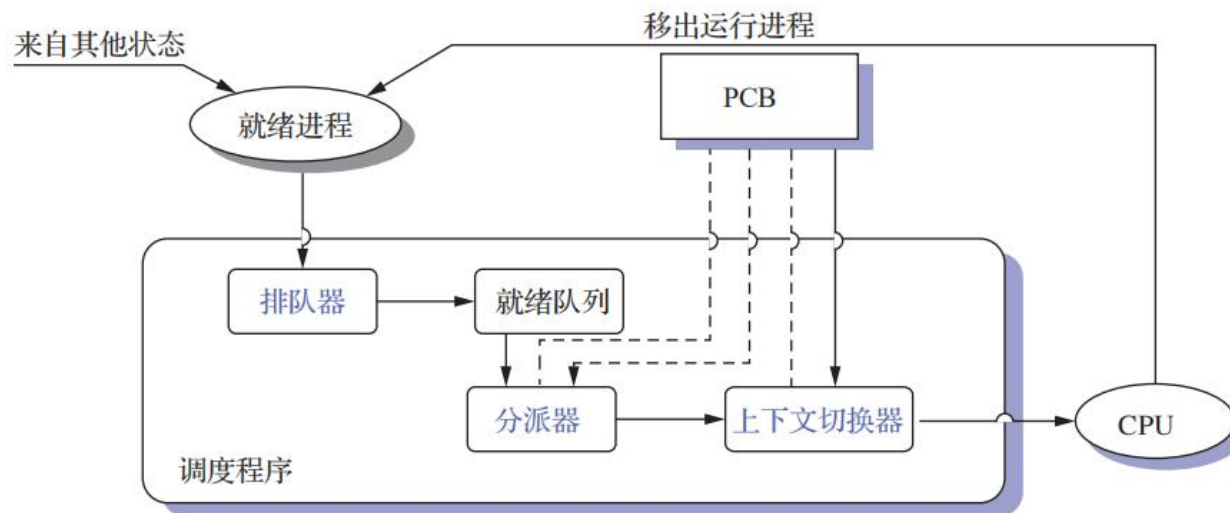
调度类型	运行频率	运行时间	算法复杂性
①低级调度	高	短	低
②中级调度	中等	较短	中等
③高级调度	低	长	高





进程调度的任务

- 保存处理机的现场信息
- 按某种算法选取进程
- 把处理器分配给进程



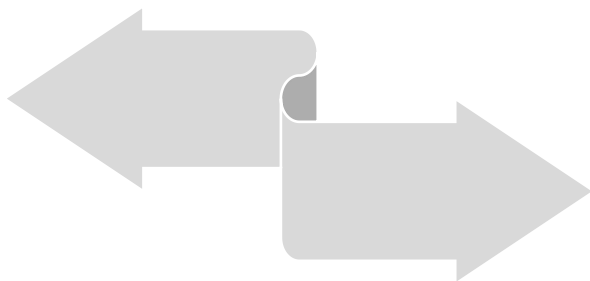
进程调度机制（调度程序分为三部分）

- 排队器：用于将就绪进程插入相应的就绪队列
- 分派器：用于将选定的进程移出就绪队列
- 上下文切换器：进行新旧进程之间的上下文切换



非抢占方式：

一旦把处理机分配给某进程后，便让该进程一直执行，直至该进程完成或发生某事件而被阻塞时，才再把处理机分配给其他进程，决不允许某进程抢占已经分配出去的处理机。



抢占方式：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。(比较复杂，效率更高，现代OS广泛采用)

- 优先权原则：允许优先权高的新到进程抢占当前进程的处理机
- 短作业优先原则：短作业可以抢占当前较长作业的处理机
- 时间片原则：各进程按时间片运行，当一个时间片用完后，便停止该进程的执行而重新进行调度另外进程



共同目标:

- 资源利用率
- 公平性
- 平衡性
- 策略强制执行

多种利用率, CPU、
设备等

$$\text{CPU利用率} = \frac{\text{CPU有效工作时间}}{\text{CPU有效工作时间} + \text{CPU空闲等待时间}}$$



批处理系统的目标:

- 平均周转时间短、系统吞吐量高、处理机利用率高



分时系统的目标:

- 响应时间快、均衡性



实时系统的目标:

- 截止时间的保证、可预测性

批处理系统的目标



周转时间：

$$T = \frac{1}{n} \left(\sum_{i=1}^n T_i \right)$$

- 从作业提交给系统开始，到作业完成为止的这段时间间隔，针对单个进程来讲。

- 平均周转时间

- 带权周转时间：权值为作业周转时间 T 与系统为之服务时间 T_s 之比。

- 平均带权周转时间

$$W = \frac{1}{n} \left(\sum_{i=1}^n \frac{T_i}{T_{s_i}} \right)$$



吞吐量：

- 单位时间内所完成的作业数



等待时间（进程调度）：

- 进程在就绪队列中等待调度的所有时间之和。就绪队列而不是等待队列

分时系统的目标



响应时间：

- 从用户通过键盘提交请求开始，直到系统首次显示出处理结果为止的一段时间。



响应时间包括：

- ①从键盘输入的请求信息传送到处理机的时间
- ②处理机对请求信息进行处理的时间
- ③将所形成的响应回送到终端显示器的时间



均衡性

- 响应时间快慢与用户请求复杂度相适应

实时系统的目标








截止时间：

➤ 是指某任务必须开始执行的最迟时间，或必须完成的最迟时间

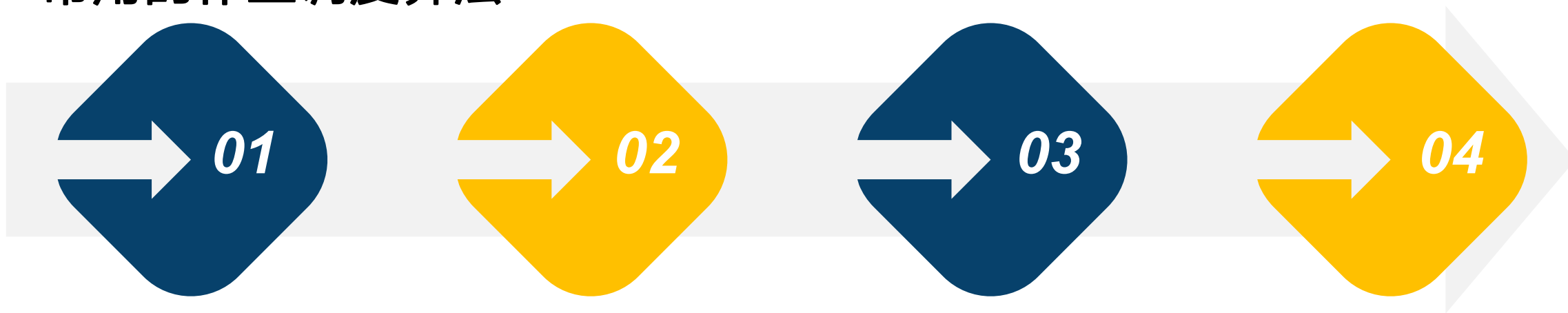


内容导航:

-  3.1 处理机调度概述
-  3.2 调度算法
-  3.3 实时调度
-  3.4 Linux进程调度
-  3.5 死锁概述
-  3.6 预防死锁
-  3.7 避免死锁
-  3.8 死锁的检测与解除

第3章 处理机调度与死锁

常用的作业调度算法



➤ 先来先服务调度算法(FCFS)

➤ 短作业优先调度算法(SJF)

➤ 优先级调度算法(PR)

➤ 高响应比优先调度算法(HRRN)

FCFS、SJF、PR既可用于作业调度，也可用于进程调度
HRRN算法常用于做作业调度



先来先服务调度算法(FCFS)



短作业优先调度算法(SJF)



优先权调度算法(PR)



时间片轮转调度算法(RR)

比较单一的调度算法



多级队列调度算法



多级反馈队列调度算法



基于公平原则的调度算法

综合性的调度算法，多种调度算法可以同时使用



先来先服务 (FCFS) 调度算法

最简单的调度算法，符合用户的直观需求

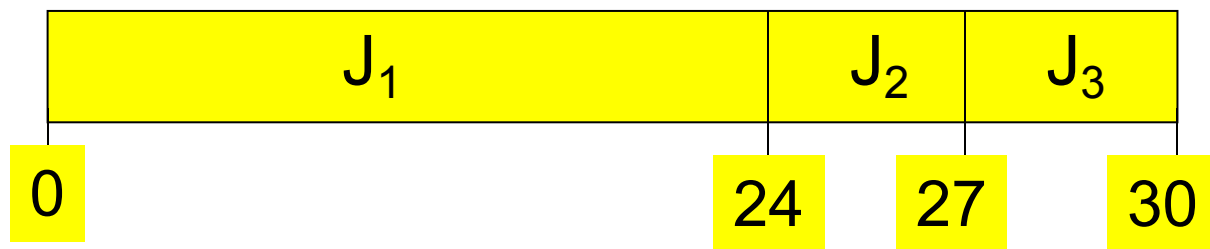
00

按照**作业**到达的先后次序来进行调度

作业	运行时间
J1	24
J2	3
J3	3

00

假定作业到达顺序如下: J1 , J2 , J3
该调度的甘特图(Gantt)为:



➤ 平均等待时间 = $(0 + 24 + 27)/3 = 17$

➤ 平均周转时间 = $(24 + 27 + 30)/3 = 27$

01

假定**进程**到达顺序如下 P2 , P3 , P1 .

02

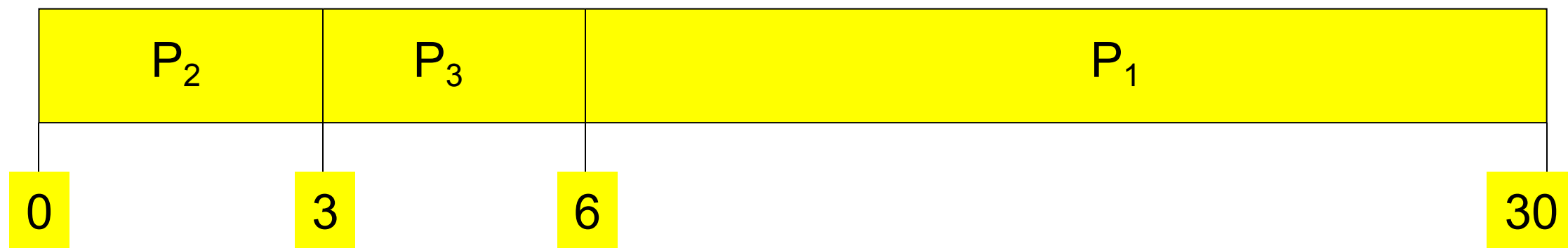
该调度的Gantt图为：

03

仅到达时间不一样，比前例好得多，两个指标都好很多

04

此结果产生是由于短进程先于长进程到达



➤ 平均等待时间 = $(6 + 0 + 3)/3 = 3$

➤ 平均周转时间 = $(30 + 3 + 6)/3 = 13$

先来先服务调度算法(FCFS)的特点

1. 作业调度和进程调度均可，最简单，本质上属非抢占方式
2. 有利于长作业/进程，不利于短作业
3. 有利于CPU繁忙型的作业(如通常的科学计算)，而不利于I/O繁忙的作业/进程（如大多数的事务处理）

例

进程名	到达时间	服务时间	开始时间	完成时间	周转时间	带权周转时间
A	0	4	0	4	4	1
B	1	5	4	9	8	1.6
C	2	2	9	11	9	4.5
D	3	4	11	15	12	3
平均					8.25	2.525



0 4 9 11 15

A B C D



SJF算法：既可用于作业，也可用于进程

- 对作业：从后备队列中选择若干个估计运行时间最短的作业。
- 对进程：关联到每个进程下次运行的CPU区间长度，调度最短的进程。



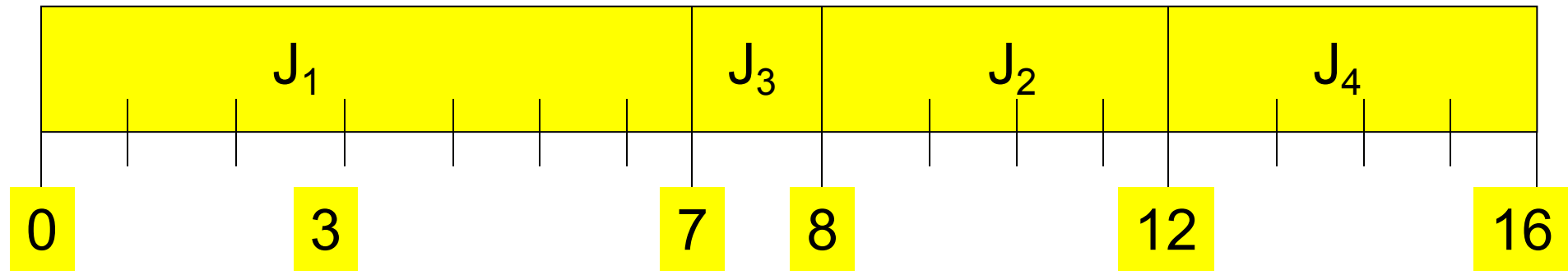
对进程调度，SJF有两种模式：

- 非抢占式SJF
- 抢占式SJF—抢占发生在有比当前进程剩余时间片更短的进程到达时，也称为最短剩余时间优先调度



SJF是最优的（对一组指定的进程而言），它给出了最短的平均等待时间。

<u>作业</u>	<u>到达时间</u>	<u>运行时间</u>
J_1	0.0	7
J_2	2.0	4
J_3	4.0	1
J_4	5.0	4

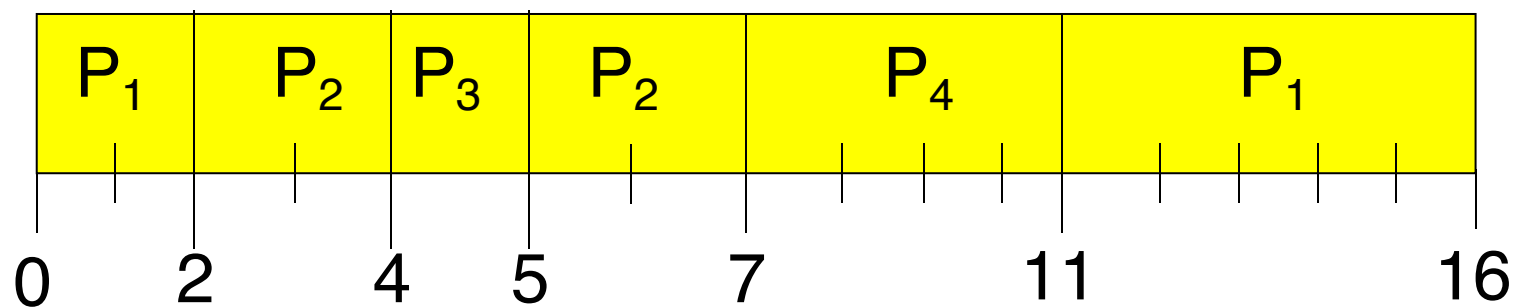


➤ 平均等待时间 = $(0 + 6 + 3 + 7)/4 = 4$

➤ 平均周转时间 = $(7 + 10 + 4 + 11)/4 = 8$

例：

进程	到达时间	区间时间
P1	0	7
P2	2	4
P3	4	1
P4	5	4







➤ 平均等待时间 = $(9 + 1 + 0 + 2)/4 = 3$

➤ 平均周转时间 = $(16 + 5 + 1 + 6)/4 = 7$

SJF比FCFS算法有明显改进

缺点:

-  只能估算进程的运行时间（进程没运行前不知道进程执行时间，因此估值不准确），所以通常用于作业调度(作业有作业说明书，显示作业执行时间)
-  对长作业不利
-  采用SJF算法时，人-机无法实现交互
-  完全未考虑作业的紧迫程度

调度算法 \ 作业情况	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	5	2	4	
FCFS (a)	完成时间	4	7	12	14	18	
	周转时间	4	6	10	11	14	9
	带权周转时间	1	2	2	5.5	3.5	2.8
SJF (b)	完成时间	4	9	18	6	13	
	周转时间	4	8	16	3	9	8
	带权周转时间	1	2.67	3.1	1.5	2.25	2.1



既可用于作业调度，也可用于进程调度。



基于作业/进程的紧迫程度，由外部赋予作业相应的优先级，调度算法根据优先级进行调度。

- 每个进程都有一个优先数，优先数为整数。
- 默认：小的优先数具有高优先级。
- 目前主流的操作系统调度算法。



高响应比优先调度算法是一种优先级调度算法，用于作业调度。



优先级调度算法的类型

- 非抢占式
- 抢占式

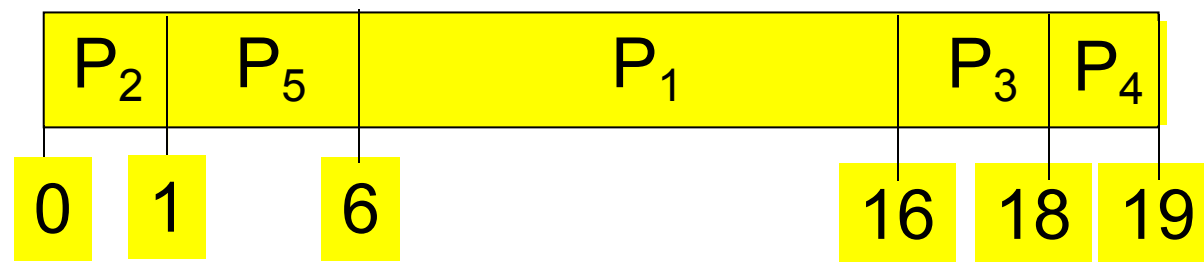


优先级类型

- 静态优先级
 - ❑ 创建进程时确定优先数(整数), 在进程的整个运行期间保持不变
 - ❑ 优点: 简单易行, 系统开销小
 - ❑ 缺点: 不够精确, 可能会出现优先级低的进程长期没有被调度的情况
- 动态优先级
 - ❑ 创建进程时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

进程	优先级	运行时间
P1	3	10
P2	1	1
P3	3	2
P4	4	1
P5	2	5

进程已经处在内存中了，就绪状态



- 平均等待时间 = $(0 + 6 + 16 + 18 + 1) / 5 = 8.2$
- 平均周转时间 = $(16 + 1 + 18 + 19 + 6) / 5 = 12$



优先级调度算法的优缺点



优点

- 实现简单，考虑了进程的紧迫程度
- 灵活，可模拟其它算法

存在问题

- 饥饿 —— 低优先级的进程可能永远得不到运行

解决方法

- 老化 —— 视进程等待时间的延长提高其优先数

- 既考虑作业的等到时间，又考虑作业的运行时间
- 优先级： $\text{优先级} = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$
- 响应比： $R_p = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{响应时间}}{\text{要求服务时间}}$
- 如等待时间相同，运行时间越短，类似于SJF
- 如运行时间相同，取决于等待时间，类似于FCFS
- 长作业可随其等待时间的增加而提高，也可得到服务
- 缺点：每次调度之前，都需要计算响应比，增加系统开销

三个作业在一台处理机上单道运行，
9:40 进行作业调度，问三个作业的执行次序？

9:40 调度时：

$$J1: 1 + 100/120 = 1 + 5/6$$

$$J2: 1 + 70/60 = 1 + 7/6$$

$$J3: 1 + 10/15 = 1 + 4/6$$

∴ 选择J2作业调度

执行次序：J2、J3、J1

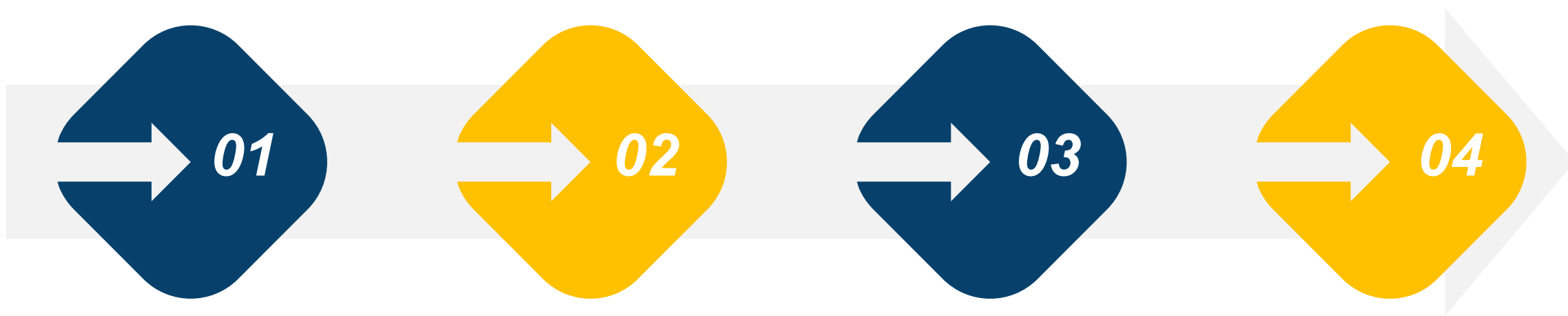
作业名	到达时间	服务时间
J1	8:00	2小时
J2	8:30	1小时
J3	9:30	0.25小时

10:40（J2完成）调度时：

$$J1: 1 + 160/120 = 1 + 4/3$$

$$J3: 1 + 70/15 = 1 + 14/3$$

∴ 选择J3作业调度



专为分时系统设计，类似于FCFS，但增加了抢占

时间片

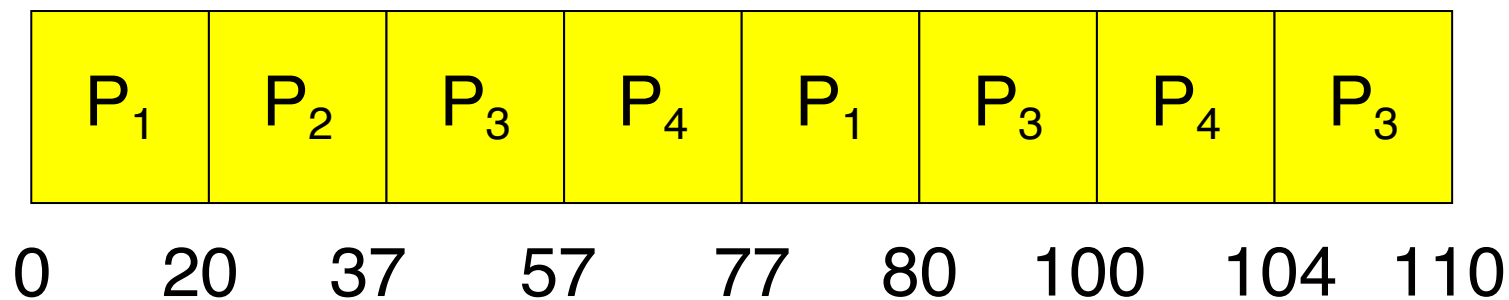
➤ 小单位的CPU时间，通常为10~100毫秒

为每个进程分配不超过一个时间片的CPU。时间片用完后，该进程将被抢占并插入就绪队列末尾，循环执行

假定就绪队列中有 n 个进程、时间片为 q ，则每个进程每次得到 $1/n$ 的、不超过 q 单位的成块CPU时间，没有任何一个进程的等待时间会超过 $(n-1)q$ 单位

进程	运行时间
P1	23
P2	17
P3	46
P4	24

➤ Gantt图如下:



- 平均等待时间: $(57+20+64+80)/4 = 55.25$
- 平均响应时间: $(0+20+37+57)/4 = 28.5$
- 通常, RR的平均周转时间比SJF长, 但响应时间要短一些.



特性

- q 大 FCFS
- q 小 增加上下文切换的时间

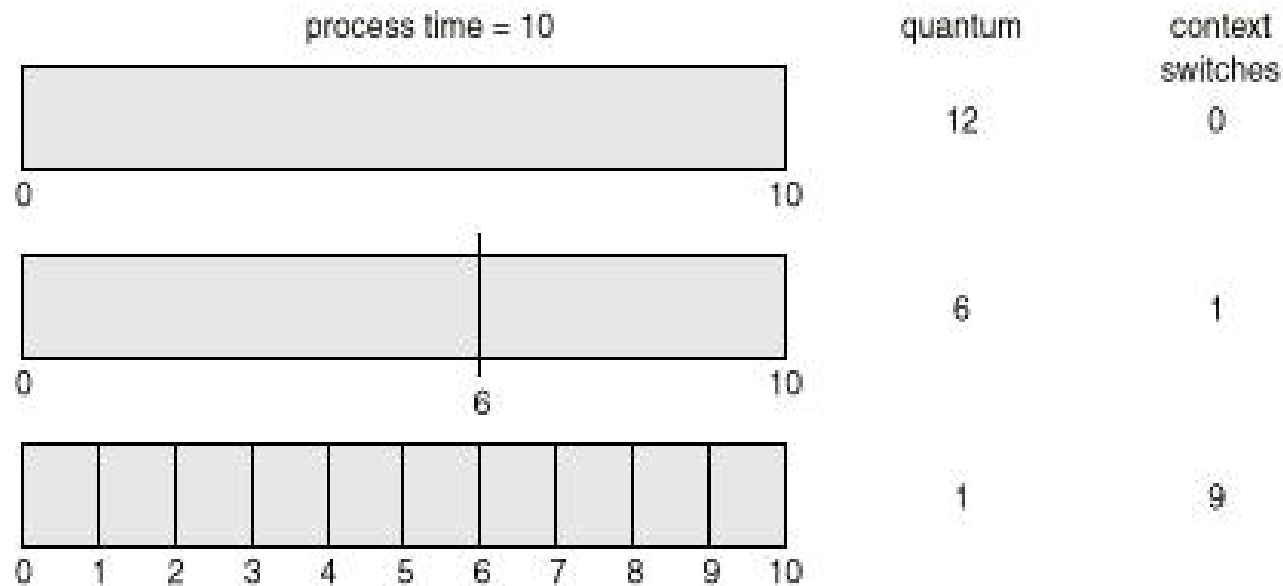


时间片设置应考虑

- 系统对响应时间的要求
- 就绪队列中进程的数目
- 系统的处理能力



一般准则： 时间片/10 > 进程上下文切换时间



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	12	10	16	11	17	
	周转时间	12	9	14	8	13	11.2
	带权周转时间	3	3	3.5	4	3.25	3.35
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	3.25	2.7



就绪队列从一个分为多个，如：

- 前台[交互式]
- 后台[批处理]



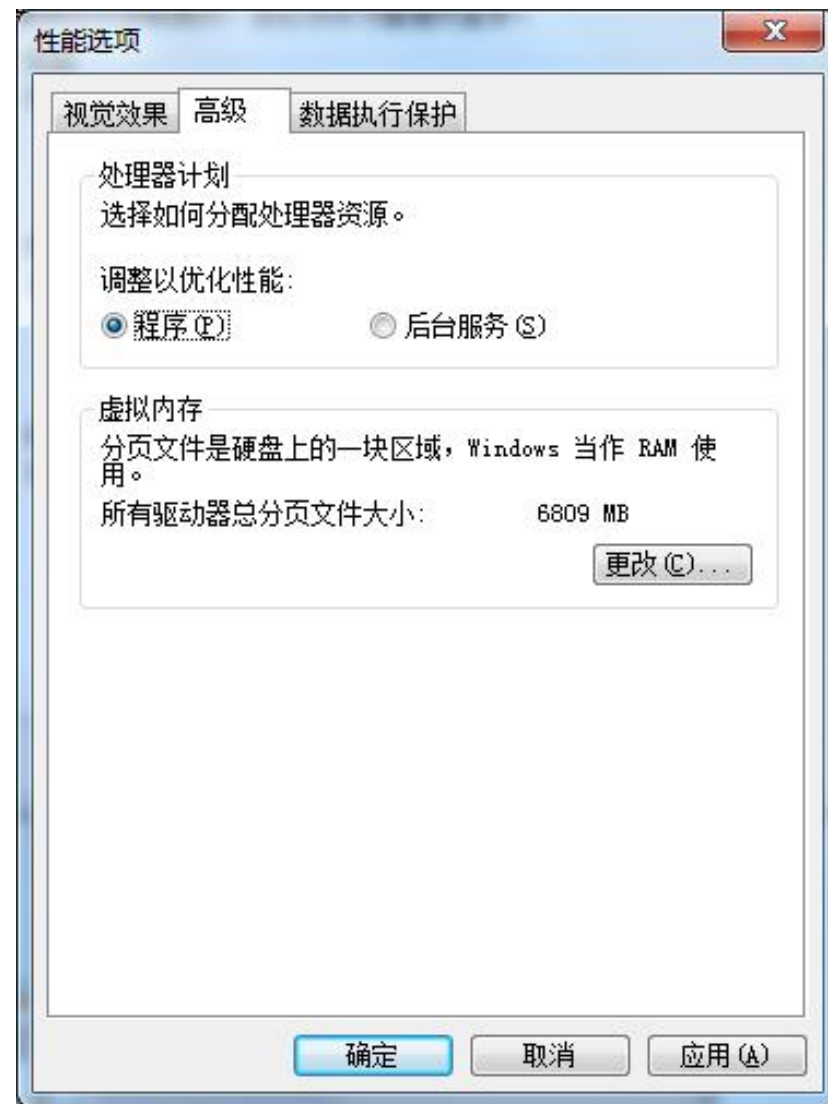
每个队列有自己的调度算法

- 前台 – RR
- 后台 – FCFS



调度须在队列间进行

- 固定优先级调度，即前台运行完后再运行后台，有可能产生饥饿。
- 给定时间片调度，即每个队列得到一定的CPU时间，进程在给定时间内执行；如80%的时间执行前台的RR调度，20%的时间执行后台的FCFS调度





进程能在不同的队列间移动



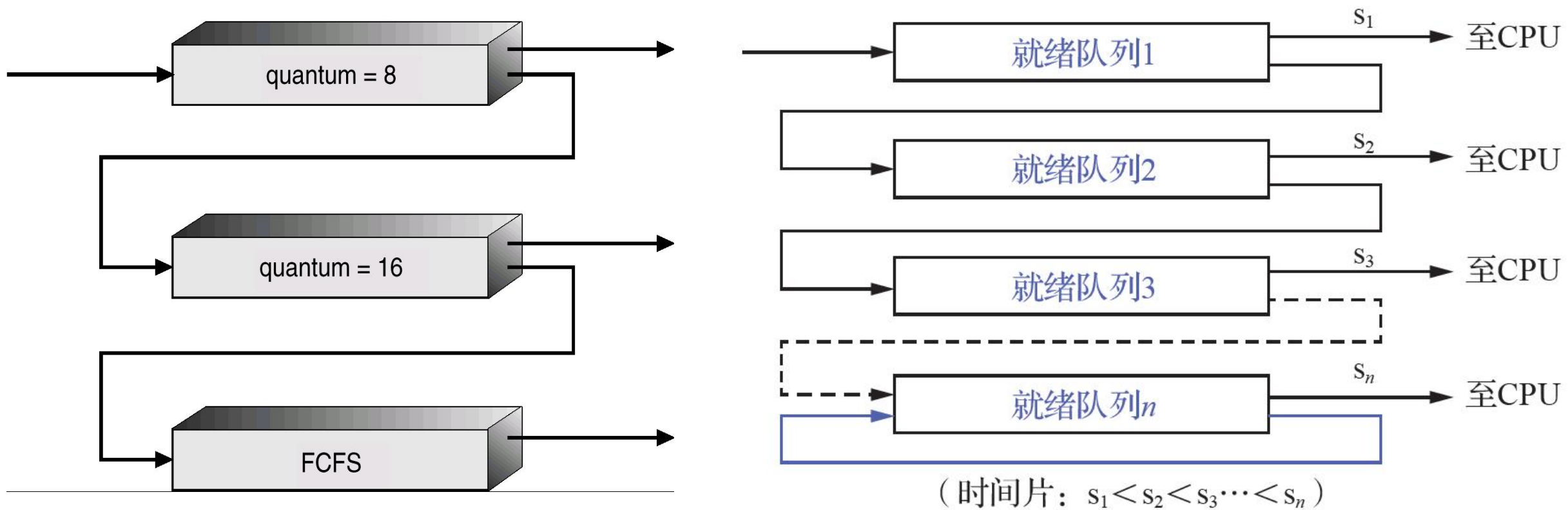
其他调度算法的局限性

- 短进程优先的调度算法，仅照顾了短进程而忽略了长进程
- 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。

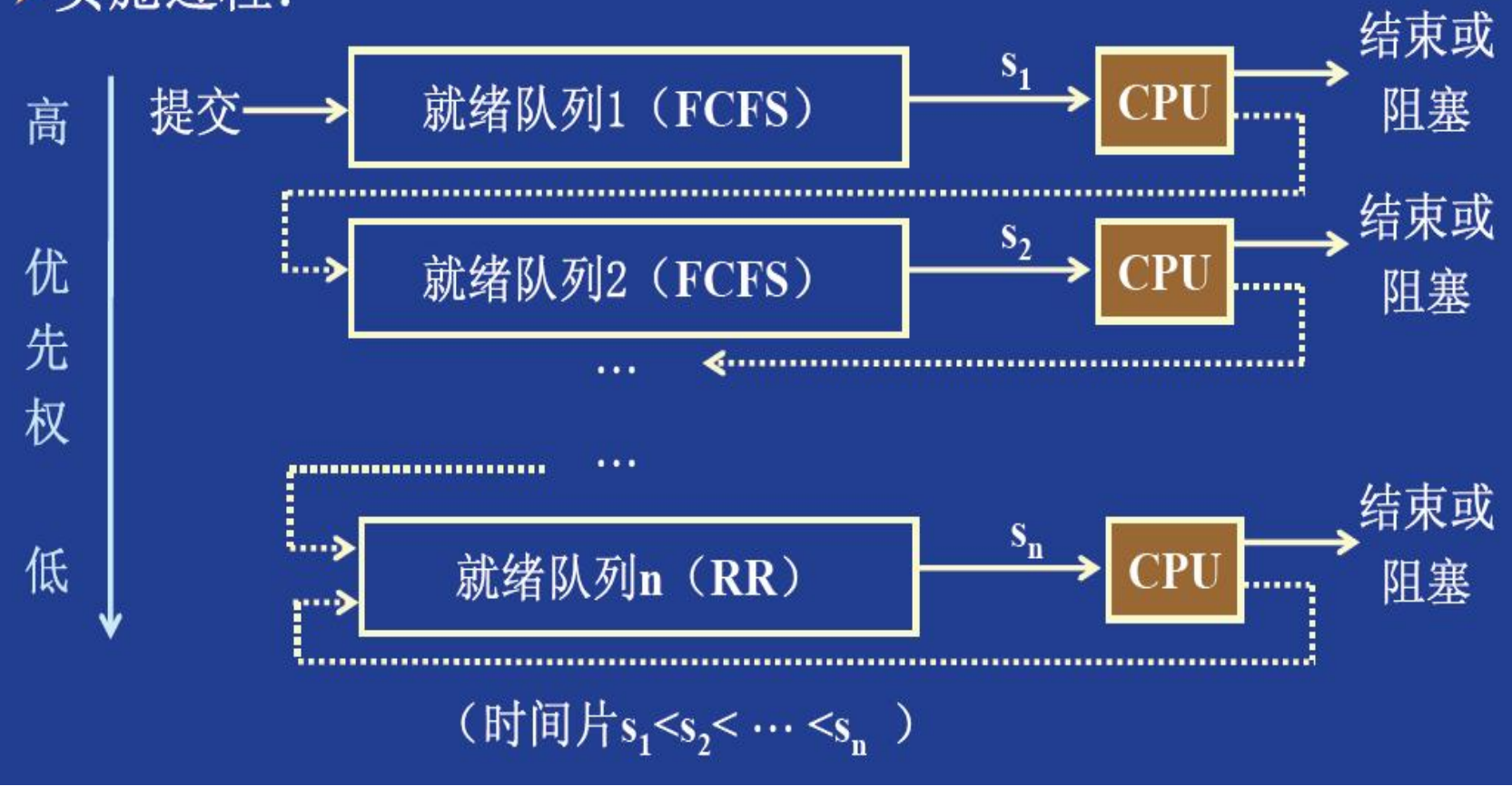


优点：

- 不必事先知道各种进程所需的执行时间；
- 可以满足各种类型进程的需要。

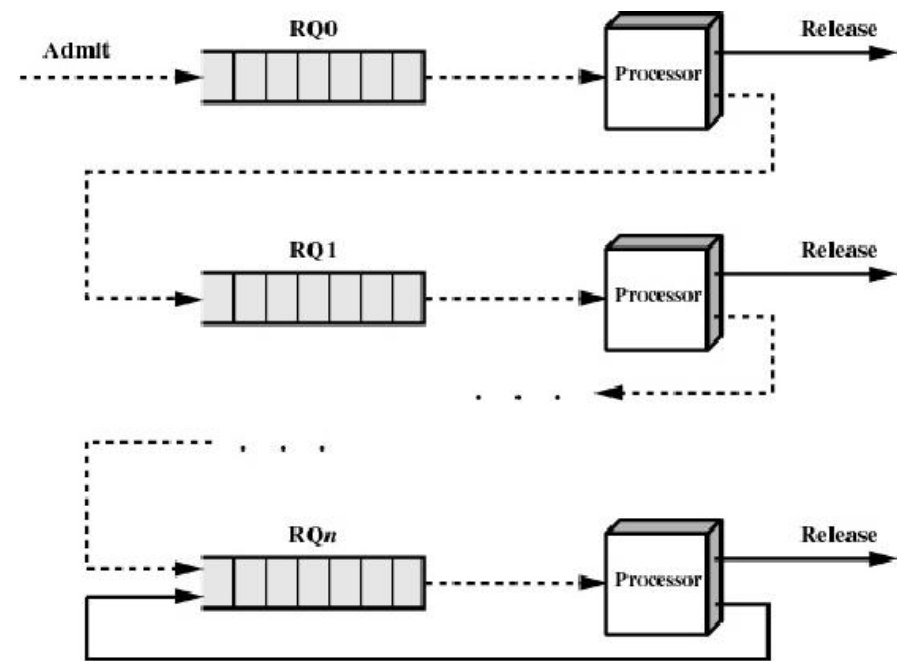


➤ 实施过程:



例如，若一个进程总共需运行100个时间片

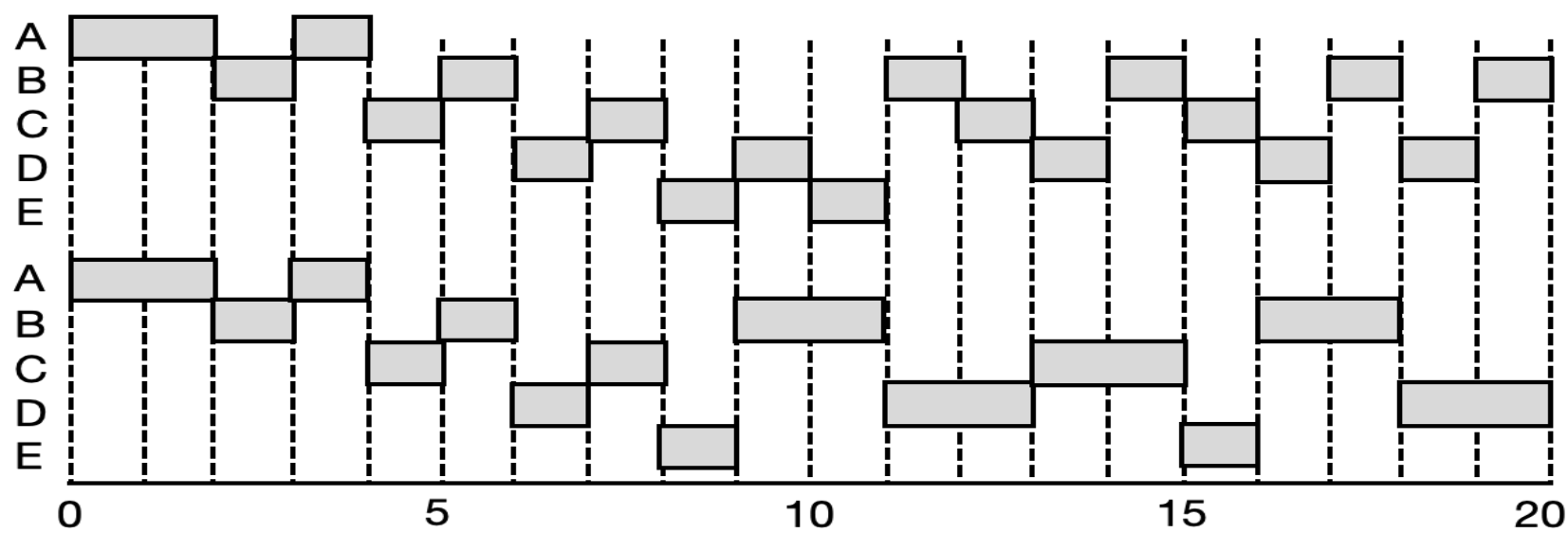
1. 初始时指定它在优先级最高的进程组中，很快就会在CPU上运行一个时间片，之后优先级也降低一个级别
2. 当它第二次有机会在CPU上运行时，它将运行 $2t$
3. 以后它将在CPU上运行的时间长度依次是4, 8, 16, 32和64个 t ，最后一次运行时，只须64个 t 中37个 t 就可完成
4. 总共需调度7次。比较单纯的轮转法，节省了93次切换时间



练习题

有下述五个进程，按照多级反馈调度算法进行调度。每个进程的到达时间和服务时间如下表所示。分别描述当时间片长度 $P=1$ 和 $P=2^i$ 时，各个时间片的执行情况，并计算每个进程的结束时间、周转时间和带权周转时间。

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



		A	B	C	D	E	平均
q=1	完成时间	4	20	16	19	11	
	周转时间	4	18	12	13	3	10
	带权周转时间	1.33	3.00	3.00	2.60	1.50	2.29
q=1 q=2	完成时间	4	18	15	20	16	
	周转时间	4	16	11	14	8	10.6
	带权周转时间	1.33	2.67	2.75	2.8	4	2.71

多级反馈队列调度算法的性能

1. 终端型作业用户

- ◆ 在第一队列中完成，作业短，交互型；

2. 短批处理作业用户

- ◆ 周期时间较短，通常三个队列即可完成；

3. 长批处理作业用户

- ◆ 依次在前 n 个队列中执行，然后再按轮转方式运行。



主要考虑调度的公平性。



保证调度算法：

- 性能保证，而非优先运行；
- 如保证处理机分配的公平性（处理机时间为 $1/n$ ）。



公平分享调度算法：

- 调度的公平性主要针对用户而言；
- 使所有用户能获得相同的处理机时间或时间比例。

例如：用户1： A,B,C,D； 用户2： E

为了使用户获得相同的处理机时间，强制调度算法为： A E B E C E D E A E B E C E D E



第四次作业

简答题

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18						

计算题
综合应用题

19	20	21	22
----	----	----	----

23	24	25
----	----	----

标黄色为本次作业