

## 操作系统 (Operating System)

# 第四章 CPU调度与优化:CPU调度

陈芳林 副教授

哈尔滨工业大学 (深圳)

2024年秋

Email: [chenfanglin@hit.edu.cn](mailto:chenfanglin@hit.edu.cn)

## ■ CPU调度基本概念与调度算法

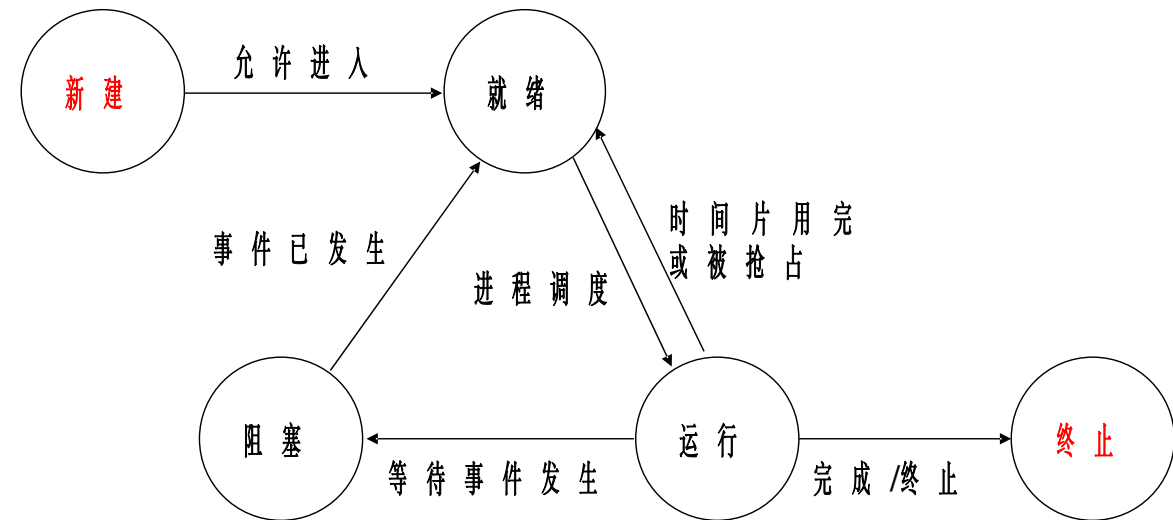
- 基本概念
- 调度准则
- 调度算法

## ■ CPU调度与程序优化

- 为何要优化代码
- 深入理解现代处理器
- 常用程序优化方法

# 什么是处理器 (CPU)调度?

- n 操作系统按照一定的策略从**就绪队列**当中选择一个进程，将**CPU的使用权**交给该进程。



进程状态变化图 (五状态)

- n 调度资源: **CPU的使用权或CPU的时间片**。
- n 调度对象: 进程或线程。其方式与原则是一样的，故经常以进程来说明。

**进程调度  $\Leftrightarrow$  CPU调度**

# 调度切换过程和开销 (进程为例)

## ■ 进程切换过程

- (1) 保存处理器上下文，包括程序计数器和其他寄存器
- (2) 更新PCB信息
- (3) 把进程的PCB移入相应的队列，如就绪、在某事件阻塞等队列
- (4) 选择另一个进程执行，并更新其PCB
- (5) 更新内存管理的数据结构
- (6) 恢复处理器上下文

## ■ 进程切换开销

- 直接开销 (内核完成切换所用时间) :
  - ✓ 保存和恢复寄存器...
  - ✓ 切换虚拟地址空间 (相关指令比较昂贵)
- 间接开销
  - ✓ 高速缓存 (Cache)
  - ✓ 缓冲区 (Buffer Cache)
  - ✓ TLB (Translation Lookup Buffer) 失效

# 非抢占式调度与抢占式调度

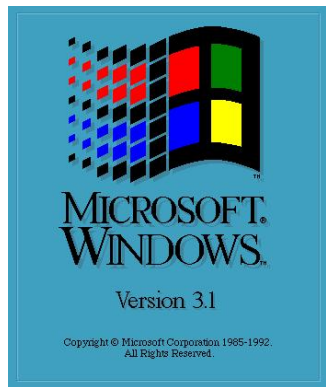
■ **非抢占式调度：**（任务完成或阻塞）**主动让出CPU**，  
调度程序将CPU分配给某就绪进程的调度方式。

- 因等待某些事件而让出CPU
- 进程的任务完成了，自动终止退出

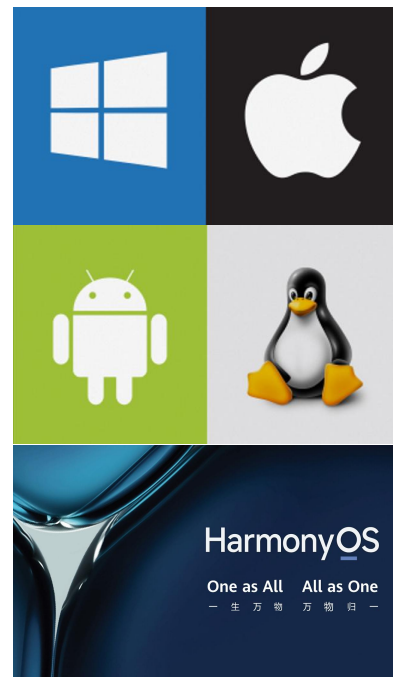
■ **抢占式调度：**操作系统将正在运行的进程**强行暂停**，  
由调度程序将CPU分配给其他**就绪进程**的调度方式。

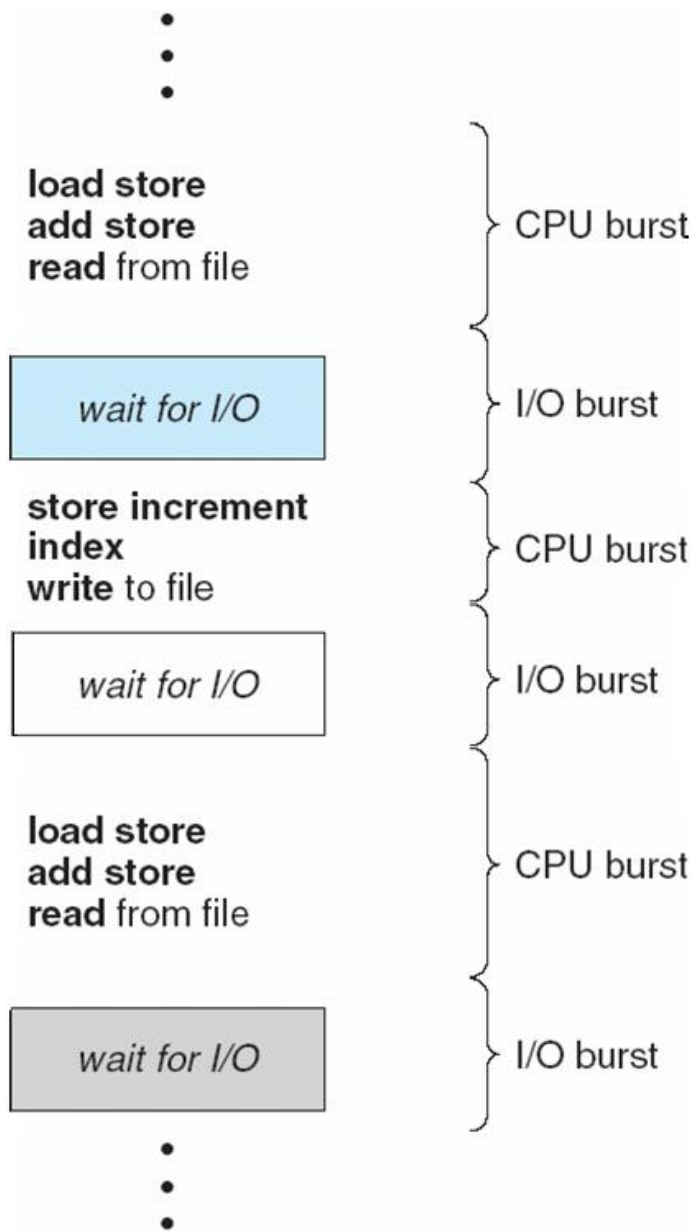
- 规定的时间片到了
- 出现了优先级更高的进程

早期的多  
道批处理  
操作系统



大多数  
现代操  
作系统  
采用。

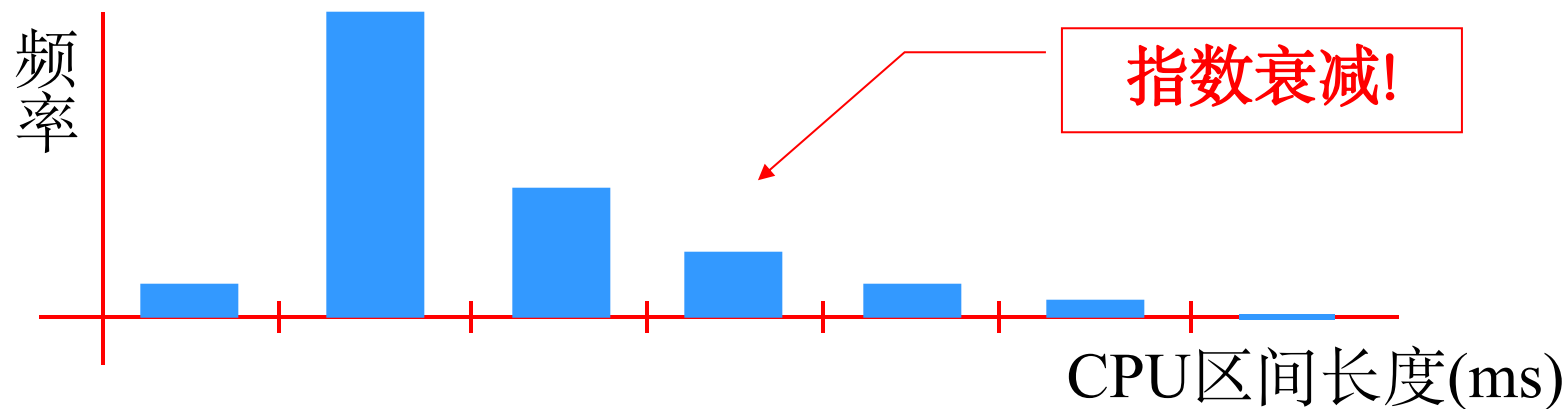




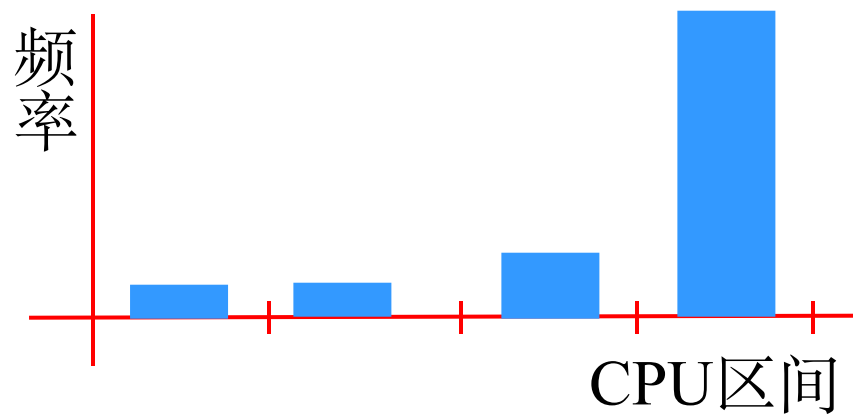
- 程序代码可以分为计算类代码和I/O类代码
- 进程执行过程由CPU执行和I/O等待周期组成
- CPU约束型程序以计算为主，CPU区间会较多，还会有少量长的CPU区间
- I/O约束型程序以I/O为主，但配合I/O处理会有大量短的CPU区间

# 任务的CPU-I/O区间的周期特性

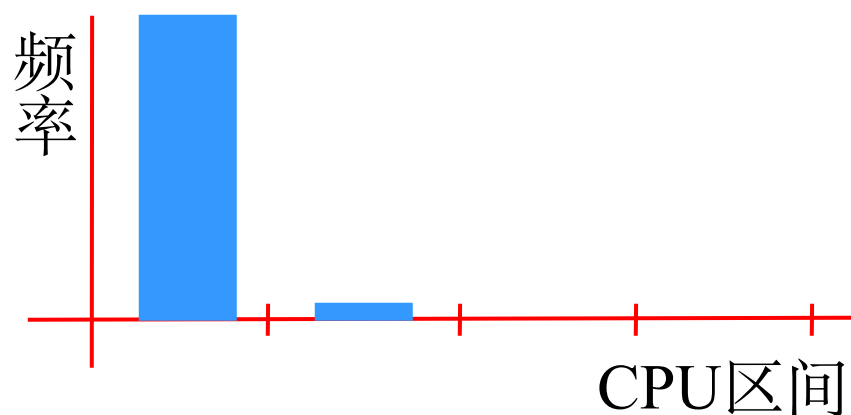
■ 一般任务生存期: I/O(载入), CPU, I/O, ... , CPU ... exit()



■ CPU-约束型



■ I/O-约束型



■ **公平、高效:** 合理地分配CPU, CPU要尽可能的忙, 提高CPU利用率。

- (1) 提高系统运算的吞吐量
- (2) 缩短进程的周转时间
- (3) 缩短进程的等待时间
- (4) 提高用户的响应满意度

## ■ 评价标准

➤ **吞吐量:** 单位时间完成的任务数量

➤ **周转时间:** 从任务提交到任务结束的时间, 即为任务完成时刻减去任务到达就绪队列的时刻。  
 $T_{\text{周转时间}} = T_{\text{完成时刻}} - T_{\text{到达时刻}}$

➤ **等待时间:** 任务在**就绪队列**中等待的时间总和

➤ **响应时间:** 从用户输入到产生反应的时间, 即从任务到达就绪队列到首次运行的时间:  
 $T_{\text{响应时间}} = T_{\text{首次运行时刻}} - T_{\text{到达时刻}}$



# CPU调度准则之间的矛盾关系

## ■ 响应时间与公平性之间的矛盾

- 响应时间短 ⇒ 前台任务的优先级高
- 前台任务优先 ⇒ 后台任务得不到CPU

批处理系统

- 周转时间 (短)
- 吞吐量 (大)

交互式系统

- 响应时间 (短)

## ■ 吞吐量和响应时间之间的矛盾

- 吞吐量大 ⇒ 时间片大
- 时间片大 ⇒ 响应时间长

用户角度		系统角度
性能	周转时间	吞吐量
	响应时间	CPU利用率
其他	可预测性	公平性
		强制优先级
		平衡资源



协调多个目标是操作系统之所以复杂的一个  
重要原因，也是复杂系统的一个基本特点。

- (1) 先到先服务调度 First Come, First Served (FCFS)
- (2) 最短作业优先调度 Shortest Job First (SJF)
- (3) 优先级调度
- (4) 轮询法调度 Round-Robin (RR)
- (5) 多级反馈队列 Multilevel Feedback Queue Scheduling (MLFQ)
- (6) 彩票算法

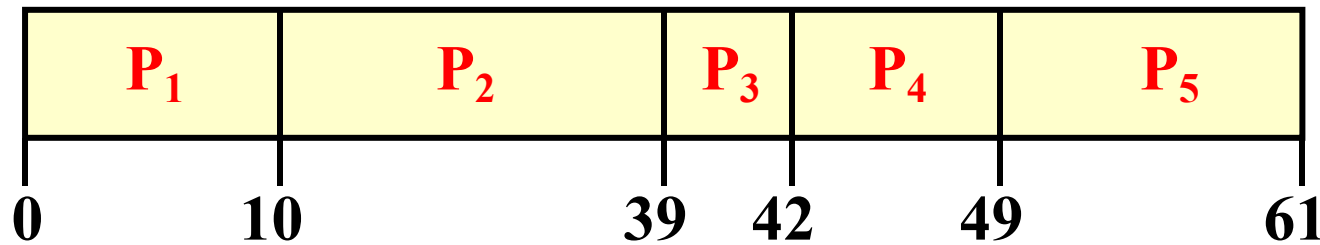
# 先到先服务调度 (FCFS) (1/2)

■ 调度的顺序是任务**到达就绪队列**的顺序。

■ 一个实例

假定任务的到达顺序为:  $P_1$ ,  $P_2$ ,  $P_3$ ,  $P_4$ ,  $P_5$ ;  
到达时刻都为0。

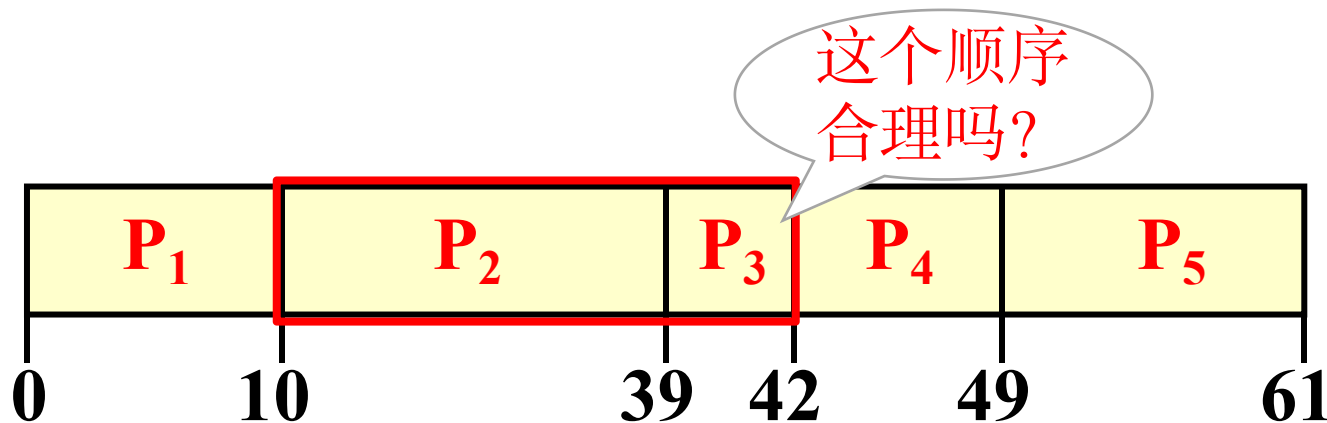
■ 调度结果



暂不考虑上下文切换的消耗

任务	CPU区间(ms)
$P_1$	10
$P_2$	29
$P_3$	3
$P_4$	7
$P_5$	12

# 先到先服务调度 (FCFS) (2/2)



■ 平均等待时间: 任务在就绪队列中等待的时间  
 $(0+10+39+42+49)/5 = 28$

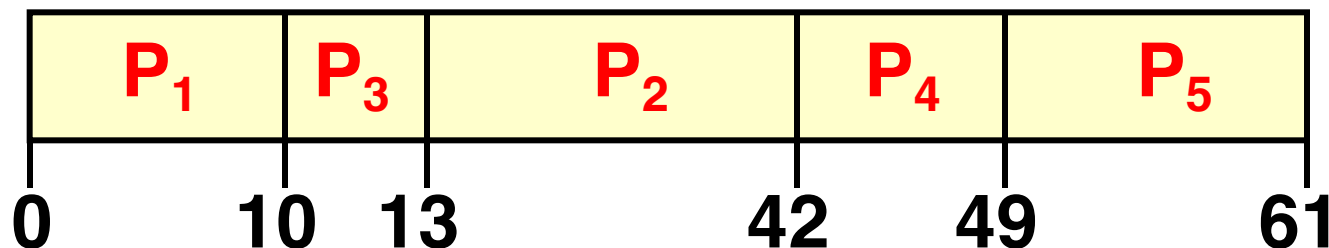
■ 平均周转时间:  $T_{\text{周转时间}} = T_{\text{完成时刻}} - T_{\text{到达时刻}}$   
 $((10-0)+(39-0)+(42-0)+(49-0)+(61-0))/5 = 40.2$

■ 平均响应时间:  $T_{\text{响应时间}} = T_{\text{首次运行时刻}} - T_{\text{到达时刻}}$   
 $(0+(10-0)+(39-0)+(42-0)+(49-0))/5 = 28$



公平、简单;  
非抢占、不适合交互式

## ■ 将 $P_3$ 调到 $P_2$ 前面



■ 平均等待时间:  $(0+10+13+42+49)/5 = 22.8$

■ 平均周转时间:  $((10-0)+(13-0)+(42-0)+(49-0)+(61-0))/5 = 35$

■ 平均响应时间:  $(0+(10-0)+(13-0)+(42-0)+(49-0))/5 = 22.8$

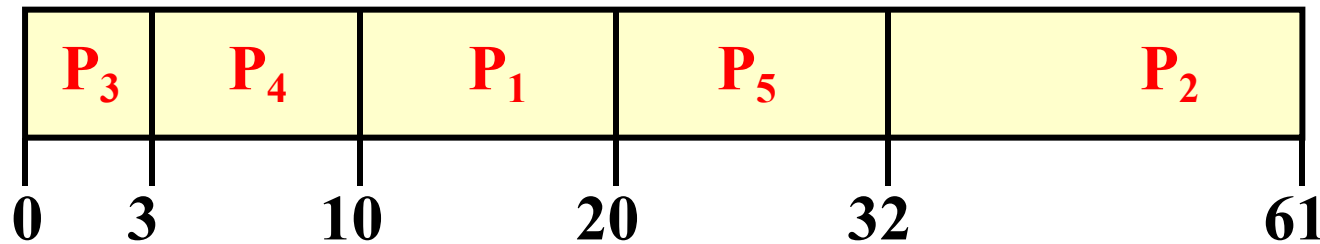
# 最短作业优先调度 (SJF) (1/3)

■ 最短的作业(**CPU区间长度最小**)最先调度。

■ 一个实例

假定任务的到达顺序为:  $P_1$ ,  $P_2$ ,  $P_3$ ,  $P_4$ ,  $P_5$ ; 到达时刻都为0。

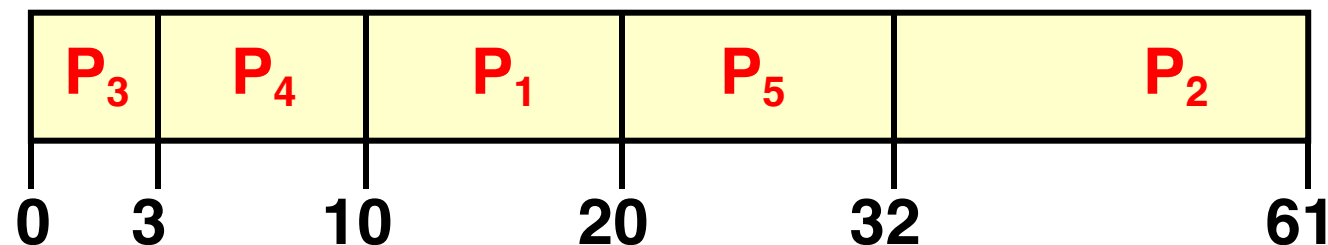
■ 调度结果



暂不考虑上下文切换的消耗

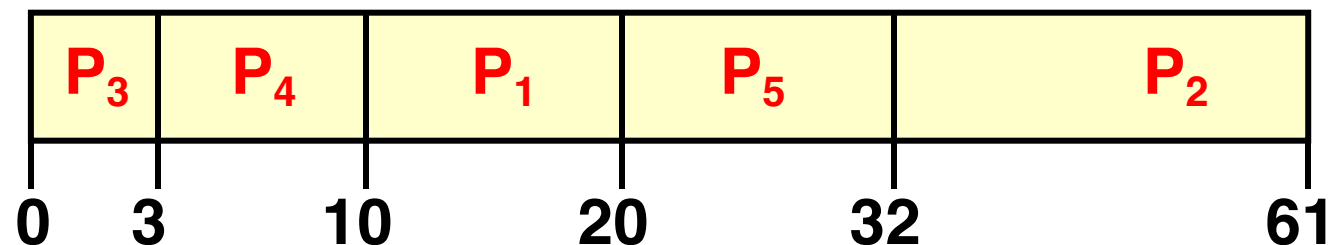
任务	CPU区间(ms)
$P_1$	10
$P_2$	29
$P_3$	3
$P_4$	7
$P_5$	12

# 最短作业优先调度 (SJF) (2/3)



- 平均等待时间:  $(0+3+10+20+32)/5 = 13 < 28$
- 平均周转时间:  $((3-0)+(10-0)+(20-0)+(32-0)+(61-0))/5 = 25.2 < 40.2$
- 平均响应时间:  $(0+(3-0)+(10-0)+(20-0)+(32-0)) / 5 = 13 < 28$

# 最短作业优先调度 (SJF) (3/3)



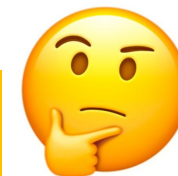
n 证明: 同时到达作业, SJF可以保证最小的平均等待时间

设 $P_1P_2...P_n$ 是调度结果序列,  $p_i$ 是任务CPU区间大小。显然该调度序列的平均等待时间为:

$$0 + p_1 + p_1 + p_2 + p_1 + p_2 + p_3 + \dots = \sum (n-i)p_i$$

如果存在 $i < j$ 而 $p_i > p_j$ , 交换 $p_i, p_j$ 调度顺序会减小平均等待时间。

任务到达的时间有先后怎么办?





# 最短剩余作业优先调度 (SRJF) (1/3)

## ■ 最短剩余作业最先调度 Shortest Remaining Job

First (SRJF) **SJF的可抢占版本**

## ■ 一个实例

假定任务的到达顺序为:  $P_1$ ,  $P_2$ ,  $P_3$ ,  $P_4$ ,  $P_5$ 。

## ■ 调度结果:

任务	到达时间 (ms)	CPU区间 (ms)
$P_1$	0	10
$P_2$	0	29
$P_3$	5	3
$P_4$	5	7
$P_5$	30	12

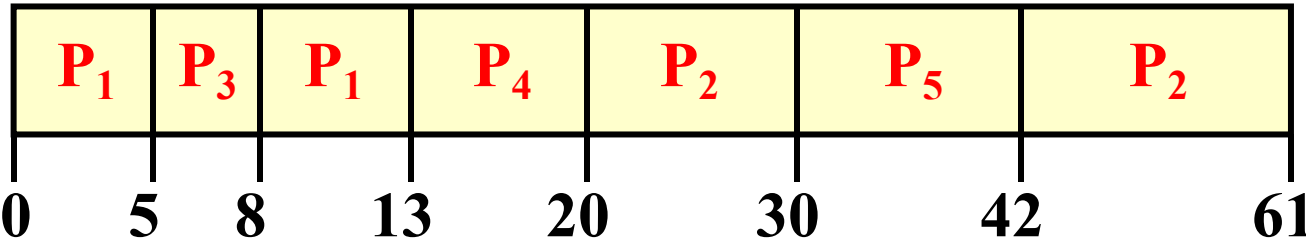
# 最短剩余作业优先调度 (SRJF) (1/3)

■ 最短剩余作业最先调度Shortest Remaining Job First (SRJF) **SJF的可抢占版本**

■ 一个实例

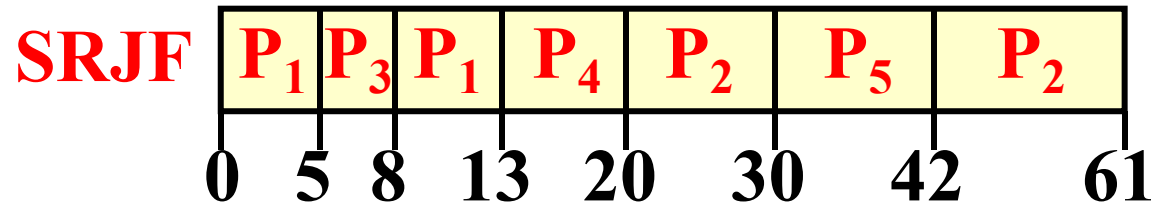
假定任务的到达顺序为:  $P_1$ ,  $P_2$ ,  $P_3$ ,  $P_4$ ,  $P_5$ 。

■ 调度结果:



任务	到达时间 (ms)	CPU区间 (ms)
$P_1$	0	10
$P_2$	0	29
$P_3$	5	3
$P_4$	5	7
$P_5$	30	12

# 最短剩余作业优先调度 (SRJF) (2/3)



■ 平均等待时间(SRJF):

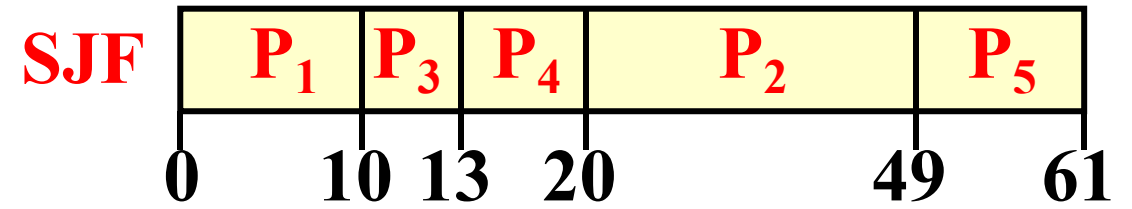
$$(3+32+0+8+0)/5 = 8.6$$

■ 平均周转时间(SRJF):

$$((13-0)+(61-0)+(8-5)+(20-5)+(42-30))/5 = 20.8$$

■ 平均响应时间(SRJF):

$$(0+(20-0)+(5-5)+(13-5)+(30-30))/5 = 5.6$$



■ 平均等待时间(SJF):

$$(0+20+5+8+19)/5 = 10.4$$

■ 平均周转时间(SJF):

$$((10-0)+(49-0)+(13-5)+(20-5)+(61-30))/5 = 22.6$$

■ 平均响应时间(SJF):

$$(0+(20-0)+(10-5)+(13-5)+(49-30))/5 = 10.4$$



抢占显然具有优点，但CPU区间必须是已知!

# 如何知道下一CPU区间大小? (3/3)

n 根据历史进行预测: 指数平均法

$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$ ,  $\tau_{n+1}$ 是预测值,  $t_n$ 是当前CPU区间

$\tau_n = \alpha t_{n-1} + (1 - \alpha)\tau_{n-1}$ ,  $\tau_{n-1} = \alpha t_{n-2} + (1 - \alpha)\tau_{n-2} \dots\dots$

$\tau_{n+1} = \alpha t_n + (1 - \alpha)\alpha t_{n-1} + \dots + (1 - \alpha)^j \alpha t_{n-j} + \dots + (1 - \alpha)^{n+1} \tau_0$

最近信息

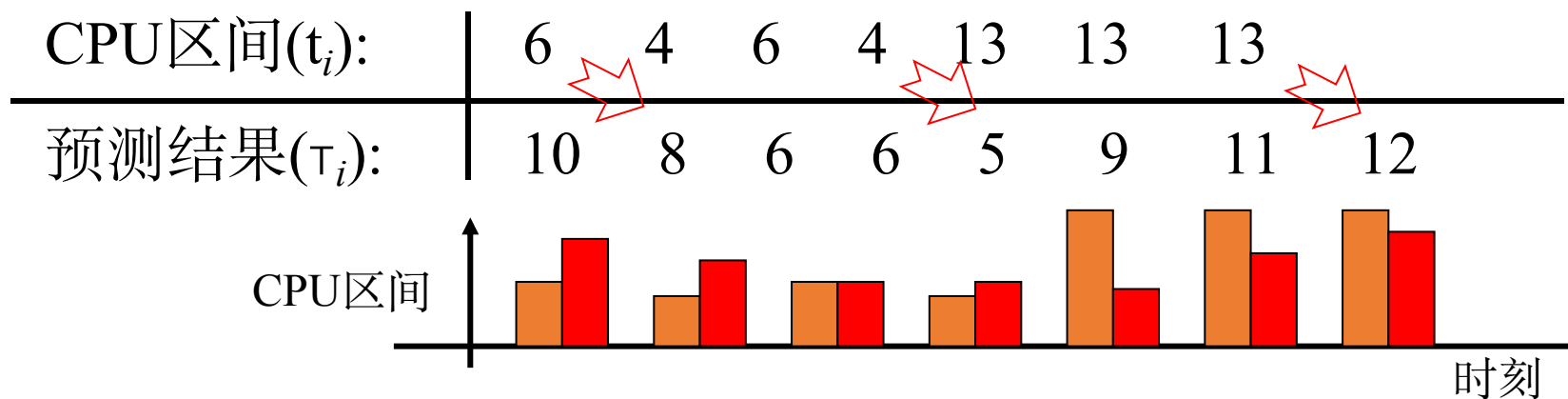
历史信息

$\alpha$ 用来控制最近信息和历史信息对预测的作用

$\alpha \rightarrow 0$ : 忽略近期, 关注历史;

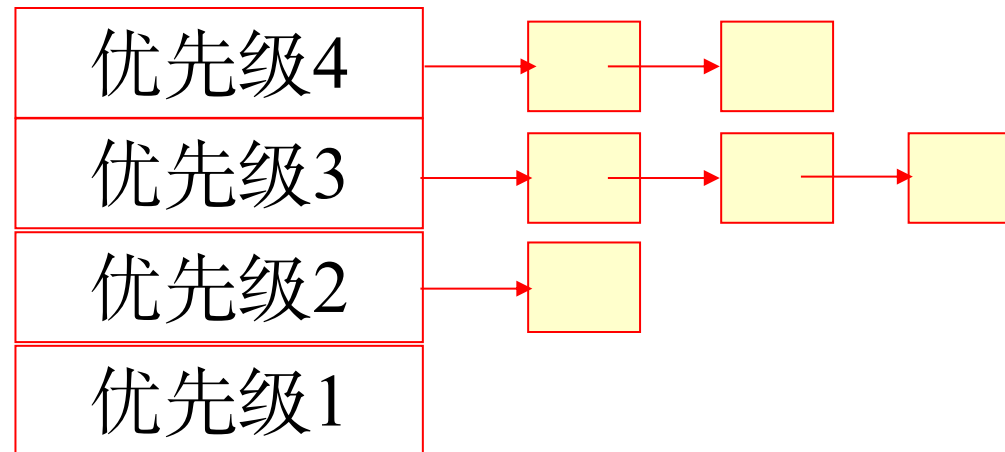
$\alpha \rightarrow 1$ : 关注当前;

通常 $\alpha=0.5$ 。



# SJF的一般化: 优先权调度

- n 每个任务关联一个优先权，调度优先权最高的任务
- n 实例1: I/O bound任务应获得更大的优先权，使得I/O尽量忙，并和CPU并行工作。优先级设为 $1/f$ ， $f$ 为CPU区间所占的比重。
- n 实例2: 定义多个优先队列: 前台任务、后台任务。只有高优先级队列为空时才调度其他任务。



■ 轮询调度算法(Round-Robin)按时间片来轮转调度

■ 一个实例

假定任务的到达顺序为:  $P_1$ ,  $P_2$ ,  $P_3$ ,  $P_4$ ,  $P_5$ ; 到达时刻都为0, 时间片为10。

■ 调度结果:



任务	CPU区间(ms)
$P_1$	10
$P_2$	29
$P_3$	3
$P_4$	7
$P_5$	12

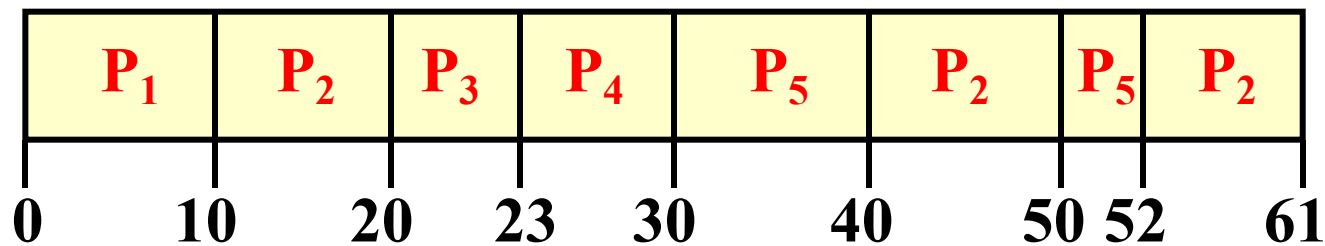
# 轮询调度算法 (RR) (1/3)

■ 轮询调度算法(Round-Robin)按时间片来轮转调度

■ 一个实例

假定任务的到达顺序为:  $P_1$ ,  $P_2$ ,  $P_3$ ,  $P_4$ ,  $P_5$ ; 到达时刻都为0, 时间片为10。

■ 调度结果:



任务	CPU区间(ms)
----	-----------

$P_1$	10
-------	----

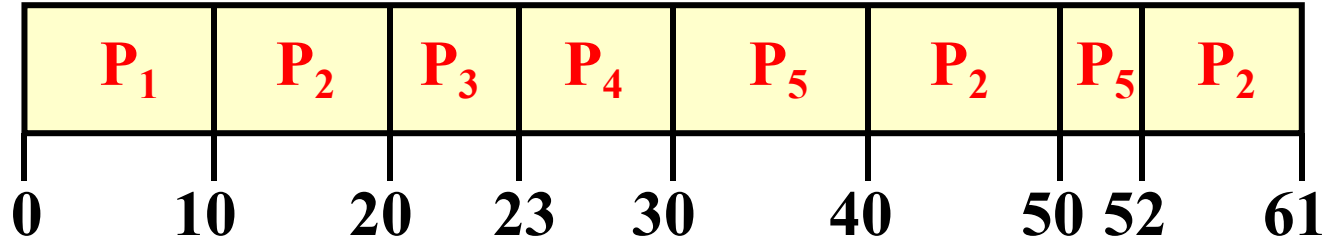
$P_2$	29
-------	----

$P_3$	3
-------	---

$P_4$	7
-------	---

$P_5$	12
-------	----

# 轮询调度算法 (RR) (2/3)



上下文切换次数:

FCFS: 4次

SJF: 4次

RR: 7次

■ 平均周转时间:

$$((10-0)+(61-0)+(23-0)+(30-0)+(52-0))/5 = 35.2$$

FCFS: 40.2

SJF: 25.2

■ 平均响应时间:

$$(0+(10-0)+(20-0)+(23-0)+(30-0))/5 = 16.6$$

FCFS: 28

SJF: 13

■ 平均等待时间:

$$(0+32+20+23+40)/5 = 23$$

FCFS: 28

SJF: 13



RR优点: 定时有响应,  
响应时间较短

RR缺点: 上下文切换  
次数较多





# RR中的时间片该如何设定? (3/3)

## ■时间片太大

- 如时间片500ms，10任务的队列里，每5s才能响应一个任务
- 响应时间太长

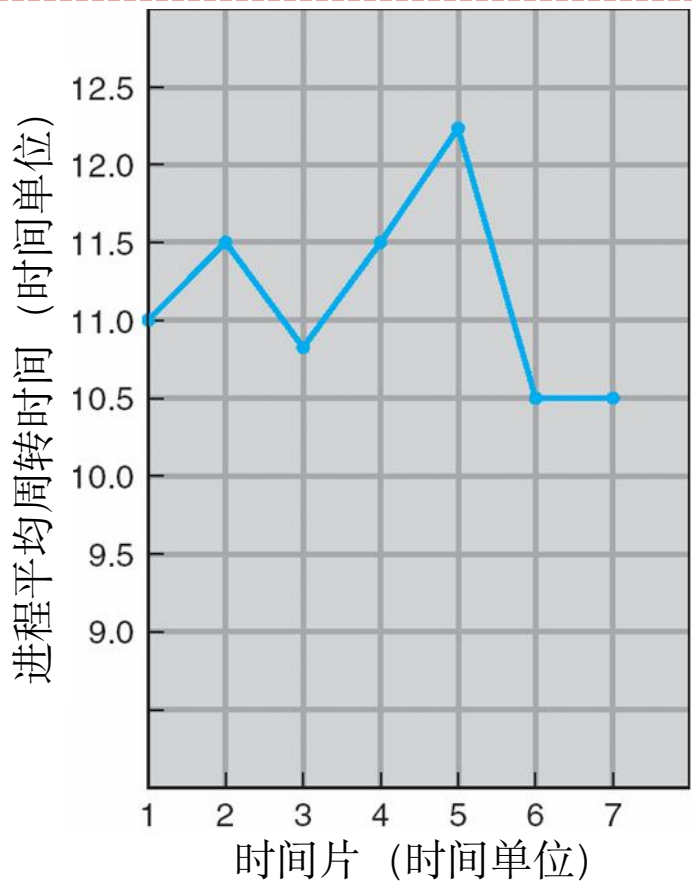
## ■时间片太小

- 如时间片20ms，上下文切换5ms，需要20%的切换代价
- 吞吐量变小，周转时间变长

## ■折中：时间片10-100ms，切换时间0.1-1ms(1%)

不同机器的切换开销差别较大。vmstat, lmbench工具检测。

# RR调度例子：周转时间与时间片



process	time
$P_1$	6
$P_2$	3
$P_3$	1
$P_4$	7

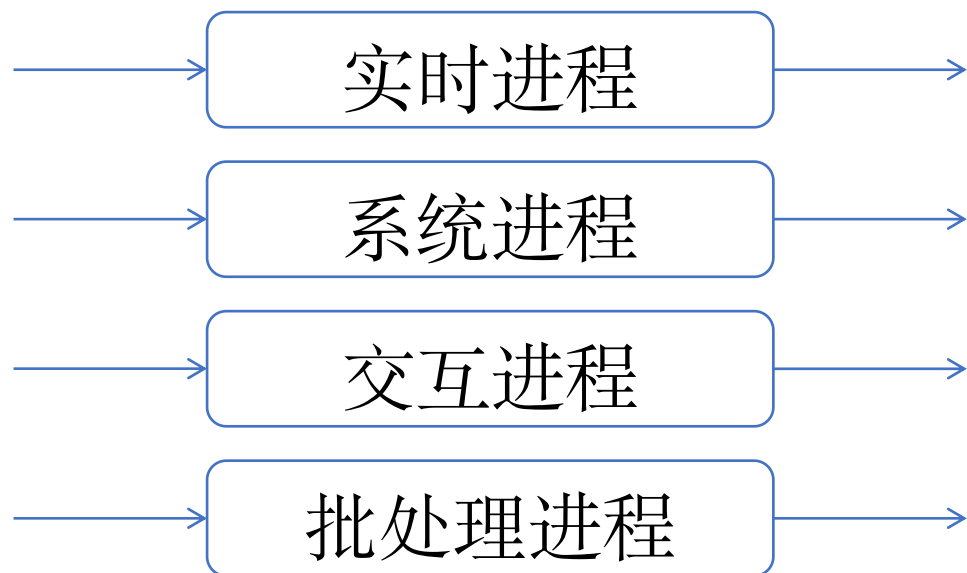
时间片	进程切换过程																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	P1	P2	P3	P4	P1	P2	P4	P1	P2	P4	P1	P4	P1	P4	P1	P4	P4
2	P1		P2		P3	P4		P1		P2	P4		P1		P4		P4
3	P1			P2			P3	P4			P1			P4			P4
4	P1				P2			P3	P4				P1			P4	
5	P1					P2			P3	P4					P1	P4	
6	P1						P2			P3	P4						P4
7	P1						P2			P3	P4						

时间片	P1周转时间	P2周转时间	P3周转时间	P4周转时间	平均周转时间
1	15	9	3	17	$44/4=11.0$
2	14	10	5	17	$46/4=11.5$
3	13	6	7	17	$43/4=10.75$
4	14	7	8	17	$46/4=11.5$
5	15	8	9	17	$49/4=12.25$
6	6	9	10	17	$42/4=10.5$
7	6	9	10	17	$42/4=10.5$

当时间片  $\geq 7$  时  
等同FCFS!

# 多级反馈队列 (MFQS) (1/2)

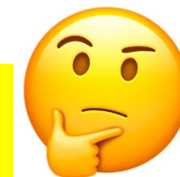
- 前面方法的缺陷: FCFS、SJF、RR这些算法任务都放置在单个队列上, 可能需要进行 $O(n)$ 的搜索。
- 将任务按照优先级分为多个队列, 每次从优先级最高的队列选择任务, 即多级队列(Multilevel Queue)。



任务优先级无法改变, 有可能存在饥饿进程。

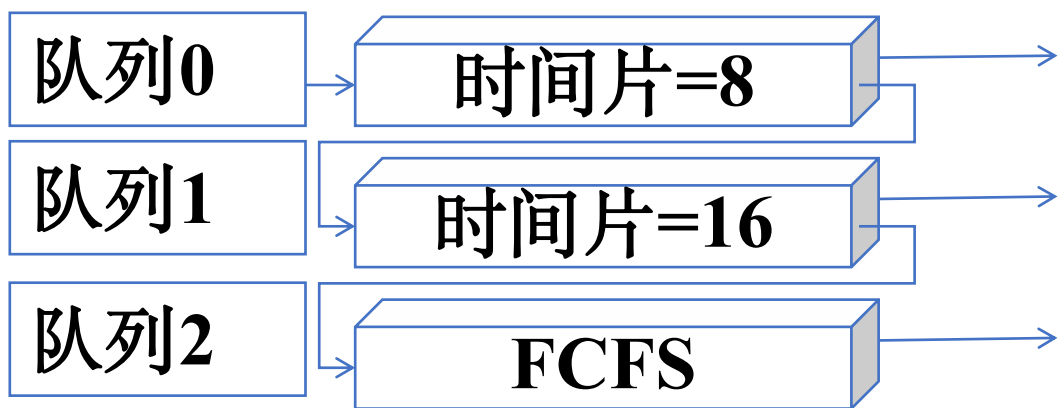
一个有趣故事: 1973年关闭MIT的IBM 7094时, 发现有一个进程在1967年提交但一直未运行。

如何解决进程饥饿问题呢?



■根据任务的CPU周期将任务放置在不同的队列中，允许任务在队列间移动。

- 使用太多CPU时间的任务会被放置在低优先级队列。
- 在低优先级队列过久的任务会被移到高优先级队列 (“老化” Aging)。



多级反馈队列需要考虑如下参数:

- 队列的数目
- 每条队列的调度算法
- 将任务移动到高优先级队列的策略
- 将任务移动到低优先级队列的策略
- 为需要服务的任务分配队列的策略



最为通用，但也是最为复杂的CPU调度算法。

■除了优化响应时间、周转时间等指标，调度过程中也应该保证每个任务都获得一定比例的CPU时间。

## ■彩票算法

- 彩票数表示了任务应该接受到的资源份额
- 彩票数百分比表示了其所占有的系统资源份额

## ■有两个进程A和B

- 进程A有75张彩票 → 占有75%的CPU时间
- 进程B有25张彩票 → 占有25%的CPU时间

## ■ 调度过程

- 调度器随机选择彩票中奖号码
- 加载中奖进程并运行它

## ■ 例如：有100张彩票

- 任务A有75张彩票，假设编码为0~74
- 任务B有25张彩票，假设编码为75~99

调度器选择的中奖彩票： 63 85 70 39 76 17

调度的结果： A B A A B A

随着运行时间的增加，  
两者得到的CPU时间  
比例会越接近期望。

- 并发能提高效率  $\Rightarrow$  并发的核心是进程能让出CPU
- 进程让出CPU  $\Rightarrow$  下一个进程使用CPU  $\Rightarrow$  这个选择就是调度
- 进程、线程(内核级、用户级)都能调度  $\Rightarrow$  任务调度
- 调度任务分类: 交互式, 批处理
- 调度时机分类: 抢占式、非抢占式
- CPU调度算法: FCFS, SJF, RR(交互式), MLFQ, Lottery

■ 1. 现在有三个同时到达的作业J1、J2和J3，它们的执行时间分别是T1、T2和T3，而且 $T1 < T2 < T3$ 。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.  $T1+T2+T3$

B.  $(3T1+2T2+T3)/3$

C.  $(T1+T2+T3)/3$

D.  $(T1+2T2+3T3)/3$

■ 答案： B

■ 解析： 系统采用短作业优先调度算法，作业的执行顺序为J1、J2和J3，它们的周转时间分别为T1， $T1+T2$ 和 $T1+T2+T3$ 。所以平均周转时间为 $(3T1+2T2+T3)/3$ 。



■ 2. 下列有关时间片的进程调度的描述中，错误的是()

- A. 时间片越短，进程切换的次数越多，系统开销也越大
- B. 当前进程的时间片用完后，该进程状态由执行态变为阻塞态
- C. 时钟中断发生后，系统会修改当前的进程在时间片内的剩余时间
- D. 影响时间片大小的主要因素包括响应时间、系统开销和进程数量等

■ 答案： B

■ 3. 假设4个任务到达系统的时刻和运行时间见下表。系统在 $t=2$ 时开始作业调度。若分别采用先来先服务和最短任务优先调度算法，则选中的任务分别是( )

作业	到达时间 $t$	运行时间
J1	0	3
J2	1	3
J3	1	2
J4	3	1

A. J2、J3

B. J1、J4

C. J2、J4

D. J1、J3

■ 答案： D

■ 解析： 先来先服务调度算法时作业来得越早，优先级越高，因此会选择J1。  
短作业优先调度算法是作业运行时间越短，优先级越高，因此会选择J3。

*Hope you enjoyed the OS course!*