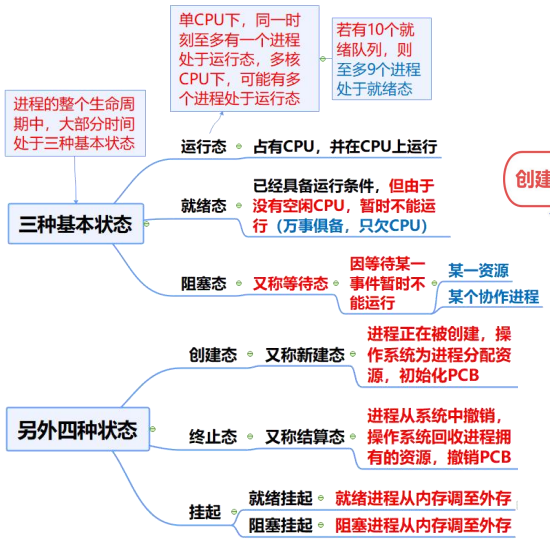
进程切换与调度算法

cctalk：里昂学长

重点

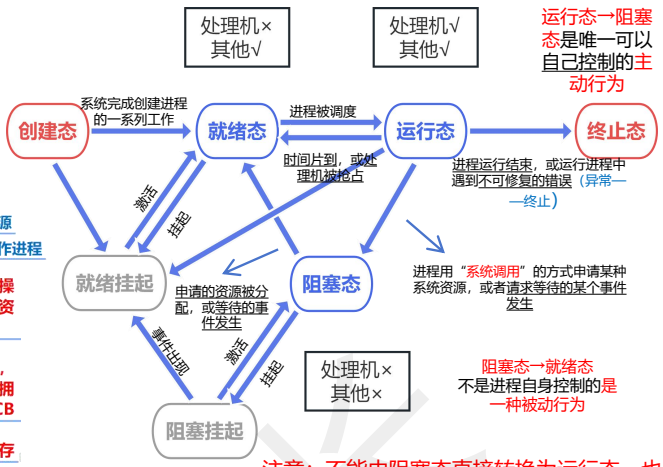
进程的状态



进程PCB中，会有一个变量state来表示当前的状态

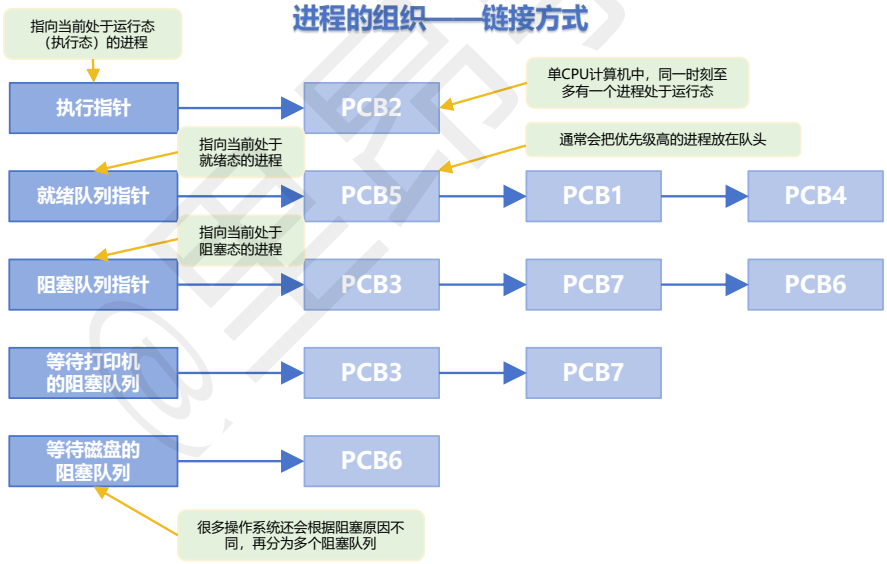
如：表示创建态，表示就绪态，表示运行态

进程状态的转换



注意：不能由阻塞态直接转换为运行态，也不能由就绪态直接转变为阻塞态。（因为进入阻塞态是进程主动请求的，必然需要进程在运行时才能发出这种请求）

进程的组织——链接方式



处理机调度

调度常考的知识点

1.作业是用户提交的，以用户任务为单位
2.进程是系统自动生成的，以操作系统控制为单位
3.进程的调度就是把一个进程从就绪态转换为了运行态，有很多事件会导致进程的调度，注意区分！！！下面有详细介绍

基本概念

1.调度是处理机进行分配，即从就绪队列中按一定算法(公平，高效的原则)选择一个进程并将处理机分配给他运行，以实现进程并发地执行
2.调度是多道程序OS的基础；调度是OS设计的核心问题

调度的层次分类

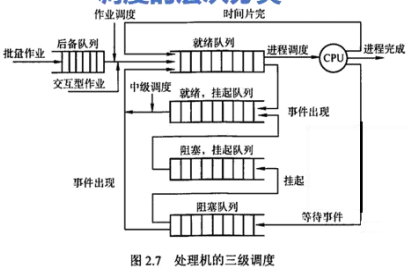


图 2.7 处理机的三级调度

层次分类模型图

1.高级调度/作业调度	• 是内存与辅存的调度，从后备队列中调度作业 • 每个作业只调入调出一次 • 通常存在于多道批处理系统中 • 内存与磁盘之间交换数据的转态转换：就绪态到挂起态（408不考挂起）
2.中级调度/内存调度	• 目的是提高内存利用率和系统吞吐量 • 将暂时不能运行的进程调到外存等待，设为挂起态 • 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 • 是存储器管理中的对换功能
3.低级调度/进程调度	• 从就绪队列中选取一个进程，调用频率很高 • 各种OS都必须配置这种调度
三种调度的联系	1.作业调度为进程活动做准备，进程调度使进程正常活动 2.中级调度将暂时不能运行的进程挂起，中级调度处于另外两个调度之间 3.调用频率：作业调度<内存调度<进程调度 4.进程调度是最基本的，不可或缺

调度的实现

层次分类模型图

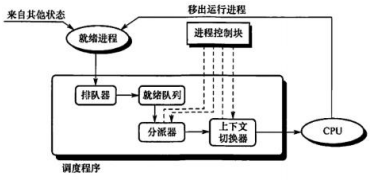


图 2.8 调度程序的结构

调度器的组成	**调度器是什么?**	用于调度和分派CPU的组件
	1.排队器	按策略给就绪进程排出一个或多个队列
	2.分派器	从就绪队列中取出进程，并分配CPU
	3.上下文切换器	在对处理机进行切换时，会发生两对上下文的切换操作
重点 调度的时机	• 先调度再切换 • 如果就绪队列中还有其他进程，则一个进程从阻塞态到就绪态时不一定会发生调度 • 调度时机举例 运行的进程运行完毕 运行的进程时间片用完 运行的进程所需资源未准备好 运行的进程自我阻塞 运行的进程出现错误	
进程的切换	**不能进行调度与切换的情况**	• 在处理中断的过程中 • 进程在OS内核临界区中 其他需要完全屏蔽中断的原子操作过程中
	可以进行调度与切换的情况	• 发生引起调度条件且当前进程无法继续进行下去时（非剥夺调度） • 中断处理结束或自陷处理结束后，被置上请求调度标志（剥夺方式的调度） • 在进程结束时能进行处理机调度 • 创建新进程后能进行处理机调度 • 在系统调用完成并返回用户态时能进行处理机调度 进程处于临界区时，只要不破坏临界资源的使用规则，就不影响处理机的调度
切换的过程	1. 将原进程的信息推入到当前进程的内核堆栈中，并更新堆栈指针 2. 内核从新进程的内核栈中装入新进程的信息 3.内核更新当前运行的进程空间指针，重设PC寄存器后开始运行新的进程	
重点 调度方式		**非抢占调度方式（主动下CPU）** / **抢占调度方式(被迫下CPU)**
	优点	实现简单，系统开销小，适合批处理系统 / 有利于提高系统吞吐率和响应效率
	缺点	不适合分时和大多数的实时系统 / 必须遵循一定的准则(如优先级，短进程优先，时间片原则)

重点 与调度相关的指标

调度最终目标要考虑的元素	**特定用户的要求（长作业还是短作业有利）+ 系统整体效率 + 调度算法的开销**	
常用的指标	**CPU利用率** = $\frac{\text{CPU有效时间}}{\text{CPU有效时间 + CPU空闲等待时间}} = \frac{\text{忙碌的时间}}{\text{总时间}}$	**等待时间** = 等待CPU的时间
	周转时间(t_i) = 作业完成时间 - 作业提交时间	**带权周转时间(w_i)** = $\frac{\text{周转时间}}{\text{作业实际运行时间}}$
	平均周转时间 = $\frac{t_1+t_2+\ldots+t_i}{i}$	**平均带权周转时间** = $\frac{w_1+w_2+\ldots+w_i}{i}$
	响应时间：从用户提交请求到首次响应所用的时间	**系统吞吐量**=单位时间内完成的作业数量

重点

调度算法

举例：假设有4个作业，提交时间分别为8，8.4，8.8，9，运行时间为2，1，0.5，0.2

FCFS先来先服务

FCFS

作业号	提交时间	运行时间	开始时间	等待时间	完成时间	周转时间	带权周转时间
1	8	2	8	0	10	2	1
2	8.4	1	10	1.6	11	2.6	2.6
3	8.8	0.5	11	2.2	11.5	2.7	5.4
4	9	0.2	11.5	2.5	11.7	2.7	13.5

平均等待 t = 1.575，平均周转 T = 2.5，平带权周转 = 5.625

SJF短作业优先

SJF

作业号	提交时间	运行时间	开始时间	等待时间	完成时间	周转时间	带权周转时间
1	8	2	8	0	10	2	1
2	8.4	1	10.7	2.3	11.7	3.3	3.3
3	8.8	0.5	10.2	1.4	10.7	1.9	3.8
4	9	0.2	10	1	10.2	1.2	6

平均等待 t = 1.175，平均周转 T = 2.1，平带权周转 = 3.525

高响应比优先调度算法

响应比 $R_p = \frac{等待时间+要求服务时间}{要求服务时间} = 1 + \frac{等待时间}{要求服务时间}$ （值越大，越先调用）

其余算法

优先级调度算法

- I/O繁忙作业优先于计算繁忙作业【因为IO操作要及时完成，他没有办法长时间保存数据；IO型先处理，后面IO就可以和CPU并行，提高系统利用率】
- 系统进程优先于用户进程，前台进程高于后台进程
- 优先级分为静态优先级和动态优先级

  动态优先级

  如果某进程在就绪队列等待了很长时间，可以适当提升优先级

  如果某进程占用了处理机很长时间，可以适当降低优先级

  频繁IO的进程可以适当提高优先级

有点类似响应比

时间片轮转算法

- 若时间片过大，退化为FCFS
- 若时间片过小，则切换频繁，处理机开销增大

多级队列算法（不用死记硬背，根据题目来）

多级反馈队列算法需要综合考虑

- 优先级数量，优先级之间的转换规则
- 就绪队列数量，就绪队列的调度算法
- 进程在就绪队列间的迁移条件

调度算法对比图 有时候题目也会说短进程优先，做法是一样的

	FCFS	SJF（短作业优先）	高响应比	时间片轮转	多级反馈队列
可抢占?	×	√（又叫最短剩余时间优先算法）	√	√	队列内算法不一定
不可抢占?	√	√	√	×	队列内算法不一定
默认决策模式	非抢占	非抢占	非抢占	抢占	抢占（很少用）
特点&优点	• 从时间角度看公平 • 实现简单 • 有利于长作业 • 不利于短作业 • 有利于CPU繁忙作业 • 不利于IO繁忙作业	• 效率高(可抢占下效率最高) • 平均等待时间最少	• 兼顾长短作业 • 满足短作业优先且不会发生饥饿现象	• 兼顾长短作业 • 绝对公平 • 绝对可抢占的 • 为了多个用户能及时干预系统	• 兼顾长短作业 • 有较好的响应时间 • 可行性强
缺点	• 不利于短作业	• 长作业会饥饿 • 估计时间不易确定	• 计算响应比的开销大	• 平均等待时间最长 • 上下文切换浪费时间	无
适用于	无	• 作业调度 • 批处理系统	无	• 分时系统（更关心响应时间） • 适用于人机交互系统	• 相当通用 • 大家都满意的算法