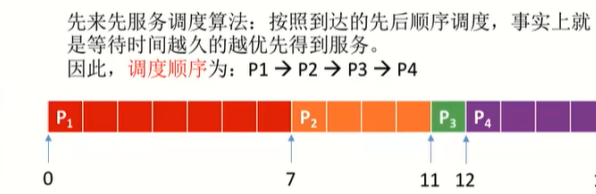
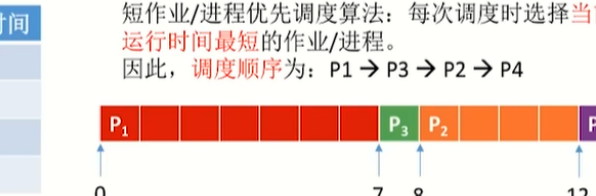
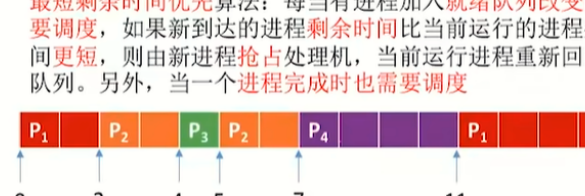
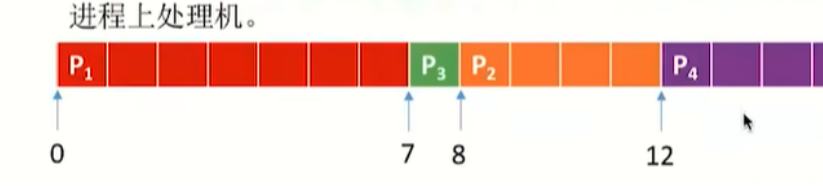
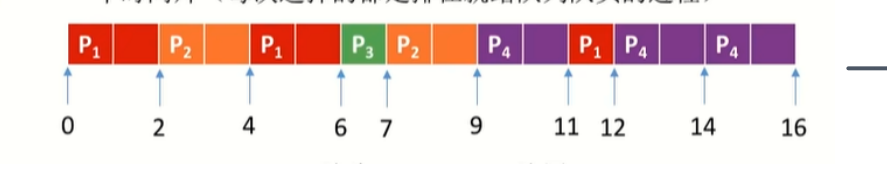
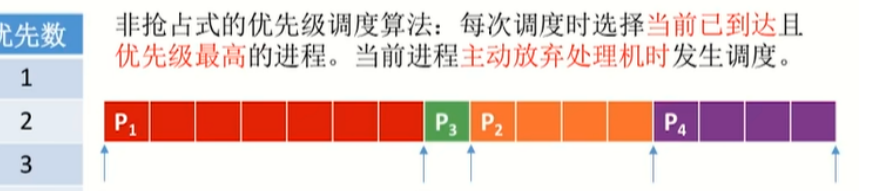
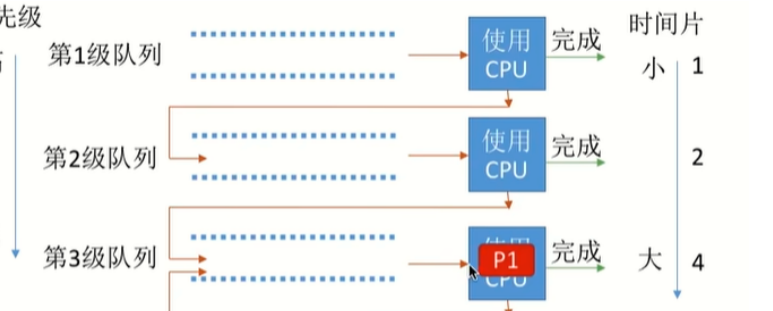


调度算法																																																
	作业/进程调度	算法思想	算法规则	抢占/非抢占 (非抢占是指等到当前的进程执行完毕后再调度新的进程执行, 抢占是指满足调度新进程的规则后, 将正在运行的进程换下cpu, 换新进程上cpu)	优/缺点	是否会出现饥饿现象 (注: 饥饿是指某进程/作业长时间得不到调度)	实例	补充																																								
先来先服务 (FCFS)	都可以	从公平角度考虑	1. 用于进程调度时, 哪个进程先进就绪队列就先被处理。 2. 用于作业调度时, 哪个作业先进后备队列就先被处理。 总结: 先来先服务。	非抢占	1. 优点: 公平, 算法简单 2. 缺点: 排在长进程后面的短进程, 它等待被处理的时间会很长 注: 长短进程指的是指被服务的时间的长短	不会, 只要时间够长, 每一个进程/作业都可以被服务/调度/处理	<div><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th></tr><tr><td>P1</td><td>0</td><td>7</td></tr><tr><td>P2</td><td>2</td><td>4</td></tr><tr><td>P3</td><td>4</td><td>1</td></tr><tr><td>P4</td><td>5</td><td>4</td></tr></table><p>先来先服务调度算法: 按照到达的先后顺序调度, 事实上就是等待时间进入就绪队列的进程优先得到服务。 因此, 调度顺序为: P1 → P2 → P3 → P4</p><p>到达时间: 进入就绪队列的时间 运行时间: 被cpu服务的时间 的图表示进程在cpu上运行的一个时间段</p></div>	进程	到达时间	运行时间	P1	0	7	P2	2	4	P3	4	1	P4	5	4																										
进程	到达时间	运行时间																																														
P1	0	7																																														
P2	2	4																																														
P3	4	1																																														
P4	5	4																																														
短作业优先 (SJF)	都可以, 用于进程调度时被称为短进程优先 (SPF)	保证短作业/进程可以被优先服务	最短的作业/进程被优先服务	1. 短作业优先/短进程优先都是非抢占式 2. 抢占式的版本被称为最短剩余时间优先算法。 算法规则是基于短进程/作业优先的规则, 并增加了进程在被服务的时候被抢占了cpu之后剩余被服务的时间的记录和比较 (较小的先被服务)	优点: 整个进程调度过程效率变高 缺点: 对短进程/作业有利, 对长作业/进程不利。 (拓展: 另外, 作业/进程的运行时间并不一定真实, 不一定能做到真正的短作业优先)	可能会, 因为如果有很多的短进程, 那么长进程会出现长时间无法被服务的情况	<div><div><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th></tr><tr><td>P1</td><td>0</td><td>7</td></tr><tr><td>P2</td><td>2</td><td>4</td></tr><tr><td>P3</td><td>4</td><td>1</td></tr><tr><td>P4</td><td>5</td><td>4</td></tr></table><p>短作业/进程优先调度算法: 每次调度时选择当前已到达且运行时间最短的作业/进程。 因此, 调度顺序为: P1 → P3 → P2 → P4</p></div><div><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th></tr><tr><td>P1</td><td>0</td><td>7</td></tr><tr><td>P2</td><td>2</td><td>4</td></tr><tr><td>P3</td><td>4</td><td>1</td></tr><tr><td>P4</td><td>5</td><td>4</td></tr></table></div><p>最短剩余时间优先算法: 每当有进程加入就绪队列或当前运行进程的时间比当前运行的进程剩余时间更短, 则由新进程抢占处理机, 当前运行进程重新回到就绪队列; 另外, 当一个进程完成时也需要调度。</p></div>	进程	到达时间	运行时间	P1	0	7	P2	2	4	P3	4	1	P4	5	4	进程	到达时间	运行时间	P1	0	7	P2	2	4	P3	4	1	P4	5	4											
进程	到达时间	运行时间																																														
P1	0	7																																														
P2	2	4																																														
P3	4	1																																														
P4	5	4																																														
进程	到达时间	运行时间																																														
P1	0	7																																														
P2	2	4																																														
P3	4	1																																														
P4	5	4																																														
高响应比优先 (HRRN)	都可以	综合考虑作业/进程的等待时间和要求服务的时间	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。 响应比 = $\frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$	非抢占	优点: 综合考虑了等待时间和运行时间 (要求服务的时间) 。根据公式可知等待时间相同, 那么服务时间短的进程响应比大, 被优先服务; 服务时间相同, 那么等待时间长的进程响应比大, 被优先服务	不会, 因为等待时间越长, 响应比越大, 那么就会被服务	<div><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th></tr><tr><td>P1</td><td>0</td><td>7</td></tr><tr><td>P2</td><td>2</td><td>4</td></tr><tr><td>P3</td><td>4</td><td>1</td></tr><tr><td>P4</td><td>5</td><td>4</td></tr></table><p>高响应比优先算法: 非抢占式的调度算法, 只有当前运行的进程主动放弃CPU时 (正常/异常完成, 或主动阻塞), 才需要运行调度, 调度时计算所有就绪进程的响应比, 选响应比最高的进程上处理机。</p></div>	进程	到达时间	运行时间	P1	0	7	P2	2	4	P3	4	1	P4	5	4																										
进程	到达时间	运行时间																																														
P1	0	7																																														
P2	2	4																																														
P3	4	1																																														
P4	5	4																																														
时间片轮转 (RR)	适合进程调度 (因为只有进程才可以被分配时间片)	公平轮流地为各个进程服务, 让每个进程都可以在一段时间内被服务	按照各进程在就绪队列中的顺序, 轮流为各个进程分配一个时间片, 如果进程在时间片内未执行完毕, 则被换下cpu并放回就绪队列尾部重新排队	抢占, 运行到指定时间片被换下, 或者在时间片内运行完则直接换下, 所花费的时间即是进程执行完毕的时间	若进程未能在时间片内运行完, 将被强行剥夺处理机使用权, 因此时间片轮转调度算法属于 抢占式 的算法。由时钟装置发出 时钟中断 来通知CPU时间片已到	不会	<div><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th></tr><tr><td>P1</td><td>0</td><td>5</td></tr><tr><td>P2</td><td>2</td><td>4</td></tr><tr><td>P3</td><td>4</td><td>1</td></tr><tr><td>P4</td><td>5</td><td>6</td></tr></table><p>时间片轮转调度算法: 轮流让就绪队列中的进程依次执行一个时间片 (每次选择的都是排在就绪队列队头的进程)</p><p>队头 → 队尾 就绪队列</p><p>时间片: 2s</p></div>	进程	到达时间	运行时间	P1	0	5	P2	2	4	P3	4	1	P4	5	6																										
进程	到达时间	运行时间																																														
P1	0	5																																														
P2	2	4																																														
P3	4	1																																														
P4	5	6																																														
优先级调度	都可以	根据任务的紧急程度 (优先级) 来决定处理顺序	每个作业/进程都有各自的优先级, 调度时选择优先级最高的	都可以	优点: 用优先级区分紧急程度, 可以灵活调整对各种作业/进程的优先级 缺点: 若有源源不断的低优先级进程到来, 则可能会导致饥饿	会	<div><div><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th><th>优先级</th></tr><tr><td>P1</td><td>0</td><td>7</td><td>1</td></tr><tr><td>P2</td><td>2</td><td>4</td><td>2</td></tr><tr><td>P3</td><td>4</td><td>1</td><td>3</td></tr><tr><td>P4</td><td>5</td><td>4</td><td>2</td></tr></table><p>非抢占式的优先级调度算法: 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。</p><p>抢占式的优先级调度算法: 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。另外, 当就绪队列发生改变时也需要检查是否会发生抢占。</p><p>注: 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。 7时刻 (P2、P3、P4): P1运行完成主动放弃处理机, 其余进程都已到达, P3优先级最高, P3上处理机。 8时刻 (P2、P4): P3完成, P2、P4优先级相同, 由于P2先到达, 因此P2优先上处理机。 12时刻 (P4): P2完成, 就绪队列只剩P4, P4上处理机。 16时刻 (:): P4完成, 所有进程都结束</p></div><div><p>例: 各进程到达就绪队列的时间、需要的运行时间、进程优先级如下表所示。使用抢占式的优先级调度算法, 分析进程运行情况。(注: 优先级越大, 优先级越高)</p><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th><th>优先级</th></tr><tr><td>P1</td><td>0</td><td>7</td><td>1</td></tr><tr><td>P2</td><td>2</td><td>4</td><td>2</td></tr><tr><td>P3</td><td>4</td><td>1</td><td>3</td></tr><tr><td>P4</td><td>5</td><td>4</td><td>2</td></tr></table><p>注: 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 (P2、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 (P4): P2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 (P1、P4): P2完成, 就绪队列只剩P1、P4, P4上处理机。 11时刻 (P3): P4完成, P1上处理机</p></div></div>	进程	到达时间	运行时间	优先级	P1	0	7	1	P2	2	4	2	P3	4	1	3	P4	5	4	2	进程	到达时间	运行时间	优先级	P1	0	7	1	P2	2	4	2	P3	4	1	3	P4	5	4	2	<div><p>补充:</p><p>就绪队列未必只有一个, 可以按照不同优先级来组织, 另外, 也可以把优先级高的进程排在更靠近队头的位置</p><p>根据优先级是否可以动态改变, 可将优先级分为静态优先级和动态优先级两种。</p><p>静态优先级: 创建进程时确定, 之后一直不变。</p><p>动态优先级: 创建进程时有一个初始值, 之后会根据情况动态地调整优先级。</p><p>与I/O型进程相对的是计算型进程 (或称CPU密集型进程)</p><p>如何合理动态设置优先级?</p><p>通常, 系统进程优先级高于用户进程 前台进程优先级高于后台进程 操作系统的编译 (I/O密集型) 或称 (I/O密集型进程) 优先级低</p><p>如果采用动态设置优先级, 那么什么情况下应该调整?</p><p>可以从追求公平、提升资源利用率等角度考虑 如果某进程在就绪队列中等待了很长时间, 则可以适当提升其优先级 如果某进程占用处理机运行了很长时间, 则适当降低其优先级 如果发现一个进程频繁地进行I/O操作, 则可以适当提升其优先级</p><p>1. 前台进程是指当前用户正在操作的进程, 它与用户交互并接收用户输入。 2. 后台进程是在后台运行的进程, 没有与用户的直接交互。</p></div>
进程	到达时间	运行时间	优先级																																													
P1	0	7	1																																													
P2	2	4	2																																													
P3	4	1	3																																													
P4	5	4	2																																													
进程	到达时间	运行时间	优先级																																													
P1	0	7	1																																													
P2	2	4	2																																													
P3	4	1	3																																													
P4	5	4	2																																													
多级反馈队列调度	适合进程调度	对其他调度算法进行折中, 集其优点于一身	1. 设置多个不同等级的就绪队列, 这些队列的优先级从高到低, 时间片从小到大。 2. 新进程先进第一级队列, 按先来先服务的规则 (FCFS) 被分配时间片进行处理, 如果时间片用完进程还没结束则将该进程进入下一级队列的队尾, 如果当前进程处于的队列级数已经是最低级了, 就回到该队列的队尾。 3. 只有上层的就绪队列都空时, 当前队列中的进程才会被依次分配时间片进行处理。	抢占式, 因为只要有新进程进入更高优先级的队列, 那么该新进程就会抢占处理机, 而原来的进程按照规则回到指定队列的队尾	对各类型进程相对公平 (FCFS的优点): 每个新到达的进程都可以很快得到响应 (RR的优点): 短进程只用较少的时间就可完成 (SPF的优点): 不必实现估计进程的运行时间 (避免用户作弊); 可灵活地调整对各类进程的偏好程度, 比如CPU密集型进程、I/O密集型进程 (拓展: 可以将因I/O而阻塞的进程重新放回原队列, 这样I/O型进程就可以保持较高优先级)	会, 更高级的队列中源源不断有进程进入, 那么处于低级队列中的进程会持续等待	<div><p>多级反馈队列调度:</p><table><tr><th>进程</th><th>到达时间</th><th>运行时间</th></tr><tr><td>P1</td><td>0</td><td>8</td></tr><tr><td>P2</td><td>1</td><td>4</td></tr><tr><td>P3</td><td>5</td><td>1</td></tr></table><p>P1[1] → P2[1] → P1[2] → P2[1] → P3[1] → P2[2] → P1[4]</p></div>	进程	到达时间	运行时间	P1	0	8	P2	1	4	P3	5	1																													
进程	到达时间	运行时间																																														
P1	0	8																																														
P2	1	4																																														
P3	5	1																																														
多级队列调度	适合进程调度	系统中按进程类型设置多个队列, 进程创建成功后插入某个队列	<div><p>最高优先级</p><p>系统进程 (如内存管理进程)</p><p>交互式进程 (如游戏、打字软件)</p><p>批处理进程 (如AI模型训练、视频特效渲染)</p><p>最低优先级</p></div> <p>多级队列调度</p>	队列之间可采取固定优先级, 或时间片划分 固定优先级: 高优先级空时低优先级进程才能被调度 时间片划分: 如三个队列分配时间50%、40%、10%	各队列可采用不同的调度策略, 如: 系统进程队列采用优先级调度 交互式队列采用RR 批处理队列采用FCFS	抢占式	视指定的队列访问规则和队列的调度策略而定	视指定的队列访问规则和队列的调度策略而定																																								