		抢占/非抢占	实现		优点	不足		
FIFO	FCFS	非抢占式			利于长作业和CPU限进程	护航效应（例：有一个CPU型进程和多个i/o型进程）		
优先级	SJF（短作业优先） （一种优先级调度算法的特例）	非抢占式	调度下次运行的CPU脉冲长度最短的进程	权限只在CPU脉冲结束后让出	①对一组指定的进程而言，SJF给出最短平均等待时间 ②减少平均周转时间，提高系统吞吐量	①饥饿现象 ②只能通过估计得到进程的剩余时间 -> 预测剩余时间的方法		
		抢占式 （最短剩余时间优先调度） （SRTF）		若有比当前进程剩余时间片更短的进程到达时，让出CPU				
	优先级算法	非抢占式	静态优先级：在进程创建时确定，在整个生命周期中保持不变 ->问题：低优先级进程饥饿 ->解决方法："老化"——根据等待时间提高优先级（动态优先级）					
		抢占式	动态优先级：可随进程推进而改变 ->改变优先级的因素：等待时间、已使用CPU的时间、资源使用情况					
时间片轮转	RR （时间片轮转法） （主要用在分时系统中）	抢占式	若时间片很大：相当于FCFS 若时间片很小：切换上下文过于频繁，系统开销过大		一般来说，响应时间短于SJF	一般来说，平均周转时间长于SJF		
综合	多级队列调度		* 就绪队列被分为多个队列 * 每个队列处理的进程类型不同，可以用不同的调度算法（前台——RR，后台——FCFS） * 调度须在队列间进行 固定优先级调度：只有运行完优先级高的队列中的进程才会运行优先级低的队列中的进程 --> 可能饥饿 基于时间片：每个队列得到一定的CPU时间，进程在给定时间内执行					
	多级反馈队列调度	抢占式	* 存在多个就绪队列，具有不同优先级 * 优先级越高，时间片越小 * 进程可在队列间变换 降低优先级：每个进程执行完一个完整的时间片后被抢占CPU，之后进入下级就绪队列，如此继续，直至将到基本优先级 提高优先级：一个进程从阻塞态变为就绪态时要提高优先级 可实现"老化"，避免饥饿 最后会将i/o型和交互式进程留在较高优先级队列		保证了响应时间（每个进程都首先进入优先级最高的就绪队列）			