- 调度的指标
  - 吞吐（Throughput），周转时间（Turnaround time），CPU利用率（CPU utilization），响应时间（Response Time）
- 批处理系统的调度（Scheduling in Batch Systems）
  - 先来先服务（First-Come First-Served, or First In First Out）
  - 最短作业优先（Shortest Job First）
  - 最短剩余时间优先（Shortest Remaining Time Next）

- 交互式系统中的调度（Scheduling in Interactive Systems）
  - 轮转调度（Round-Robin）
  - 优先级调度（Priority Scheduling）
  - 多级反馈队列（Multilevel Feedback Queue）
    - Game the system
- 实时系统的调度（Scheduling in Real-Time Systems）
  - 准入控制（Admission-control）
  - 单调速率调度（Rate Monotonic Scheduling）
    - N个进程的最坏情况下的CPU利用率
  - 最早截止期限优先（Earliest Deadline First Scheduling）

一、调度指标

- CPU利用率
- 吞吐量：单位时间完成的作业量
- 周转时间：作业从提交到完成的总时间（包括等待，就绪排队，运行，IO）
  带权周转时间=周转/实际运行时间
- 等待时间：
- 响应时间：提交请求到首次产生响应的时间

二、批处理系统调度

FCFS

- 不可剥夺
- 不利于短作业
- 有利于CPU繁忙，不利于IO繁忙

SJF

> 估计运行时间最短的，实际不一定估得准

- 默认不抢占，实际可抢占可不抢占
- 优：平均等待、平均周转最少
- 缺：不利于长作业（饥饿，非死锁），未考虑优先级

11、最短作业优先算法在什么情况下是最优的？怎么证明？

在所有作业都可同时运行的情况下，且优化目标是平均周转时间时，最短作业优先算法是最优的

考虑有4个作业的情况，其运行时间分别为a,b,c,d。第一个作业在时间a结束，第二个在时间b结束，以此类推。

平均周转时间为 $\frac{4a+3b+2c+d}{4}$

显然a对平均值影响最大，所以它应是最短作业。如果a这个位置不安排最短的作业，那么一定可以把这个位置的作业与最短作业互换位置，使得时间更少

最短剩余时间优先

可抢占版本的SJF

三、交互式系统调度

时间片轮转调度

> 带时间片的FCFS

- 默认抢占，时间到了一定抢占
- 缺：平均等待时间长，上下文切换频繁浪费时间
- 适用于分时系统

优先级调度

- 分类：是否抢占，优先级是否改变
- 其他方法：时钟中断时降低当前进程优先级（老化防止饥饿）、赋予最大时间片

多级反馈队列

- 多个就绪队列不同优先级，一队完才二队(修正：给一个最大时间片)
- 高优先级队列时间片小
- 队内FCFS，到时间完不成去下一队列末尾
- 优点：能够将**IO密集型,交互进程(使用CPU时间少的进程)与CPU密集型进程**分开,前者向更高优先队列移动,后者向更低优先队列移动

13、怎么解决多级反馈队列调度算法中的饿死问题?

饥饿：指系统不能保证某个进程的等待时间上界，使该进程长时间等待，当等待时间给进程推进和响应带来明显影响时，称进程饥饿。

饿死：当饥饿到一定程度的进程所赋予的任务即使完成也不再具有实际意义时，称该进程被饿死

解决方法：定时将所有任务优先级提升到最高。因为在一个长任务被分配到最低优先级之后，如没有一种机制提高任务优先级，且此时一直有短任务要调度，那么这个长任务就不会有被执行的机会，进而造成饿死

Game the system?

阻碍低优先级队列运行，始终在高优先级跑

解决：不管要检查时间片是否用完，还要记录运行总时间，每个优先级队列都要给限制

Game the system

if a programmer knows how the scheduler works

- then he/she can write code that will force the system to block on some low-latency I/O operation (e.g., sleep for a few milliseconds) just before the quantum expires.
- That way, the process will be **rewarded** for not using up its quantum even if it repeatedly uses up **a large chunk** of it.
- And the process remains at a high priority level.

Solution — Keep tracking:

- Instead of simply checking whether a process uses up its time slice, the scheduler keeps **track** of the **total time** that the process spent running over a larger time interval (several time slices).
- Each priority queue will have a maximum CPU time allotment associated with it.

四、实时系统调度

- 有截止期限

准入控制

就是说如对于周期性有期限任务，要能分配的开

- 每个分式都代表了一个任务的利用率

Admission-control:

There are *m* periodic processes, with p_i period and c_i processing time, then it is schedulable if it satisfies:

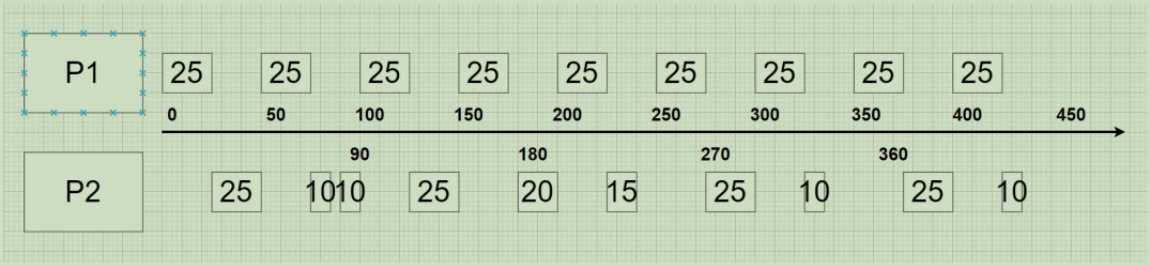
$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

单调速率调度

6、

考虑两个实时任务P1 和P2，其中P1的周期50ms，而P2是90ms，P1的处理时间是25ms，而P2是35ms，P1和P2的截止时间都是在下个迭代来临之前。如果使用单调速率算法，给出其最终成功或失败的原因

单调速率算法：任务的周期越短，优先级越高，且支持抢占调度



结果：成功，分析如下：

P1周期更短因此优先级高

- 每个P1周期到来时，若空闲则直接运行P1，若P2正在运行则中断并抢占运行P1。
- 每个P2周期到来时，若空闲则运行P2直到P1周期到来，若P1正在运行则等P1结束后运行P2
- 在 P1 和 P2 的周期最小公倍数 450 ms 内，单调速率算法能够成功执行正确，因此永远成功

- CPU利用率有限：调度N个进程的最坏情况下的CPU利用率

$$N(2^{1/N}-1)$$

> CPU利用率不能超过这个值,否则不保证满足进程截止期限
>
> 当进程数量接近无穷时，它大约接近 69%。有两个进程的系统，CPU 利用率是 83%

最早截止期限优先调度

截止期限越早,优先级越高

、

T：调度算法

二道作业批处理系统：内存最多可以放两个作业

到来顺序都是指0时刻，除了FCFS无意义

作业调度：决定谁先调入内存（作业优先数调度不踢人）

进程调度：决定内存里谁先运行

优先数越小，优先级越高

抢占式缺点：进程调度浪费时间

各类用户：终端型用户作业短小放1级；短批处理12级；长批处理以此1-n运行不饿死

注意带权平均是大于1的