中国科学院大学
University of Chinese Academy of Sciences

操作系统课程 B0911010H-01

CPU调度

中国科学院大学计算机学院

2025-10-15

内容提要

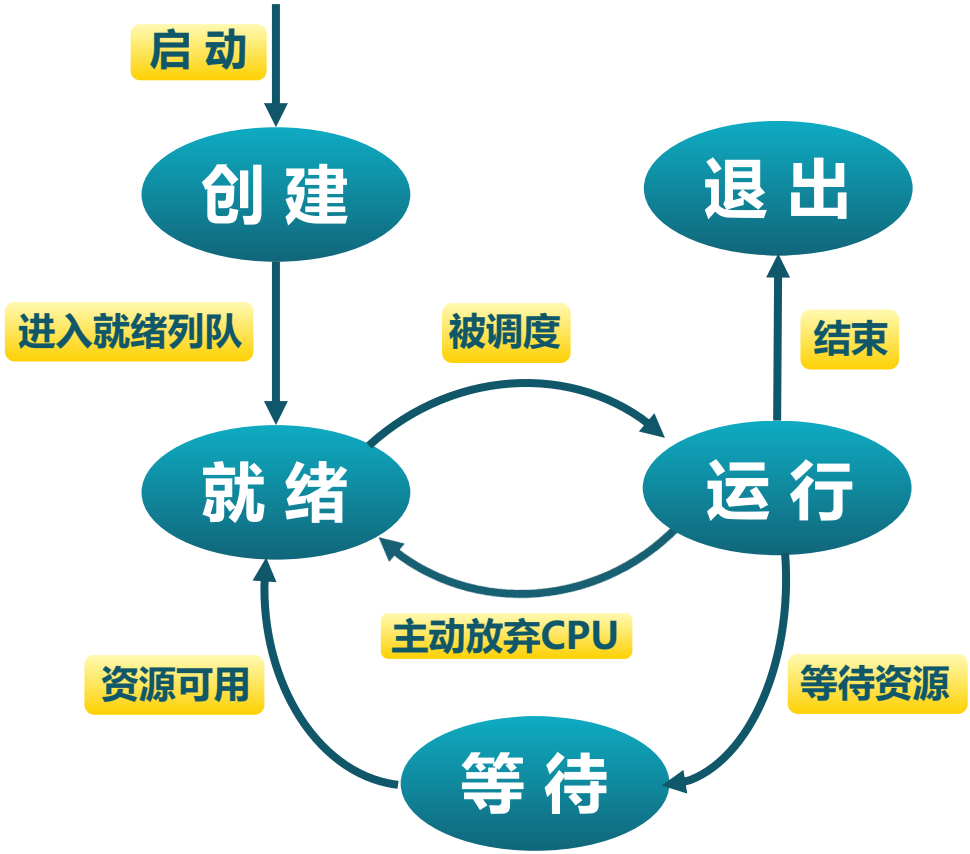
- CPU调度基础
- CPU调度算法

进程调度

- 进程切换与调度
  - CPU是共享资源，进程按照一定策略使用CPU时长
  - 当发生进程切换时
    - 进程A被切换，不再使用CPU
    - 决定接下来哪个进程使用CPU？
  - 进程切换场景
    - 进程运行结束，或者主动放弃CPU使用权
    - 分配给进程的时间片用完
    - 进程等待IO被阻塞、等待资源被阻塞（例如拿不到锁）
    - 内核不让进程使用CPU，例如有更高优先级进程要运行
  - 进程调度类型
    - 非抢占式调度
    - 抢占式调度

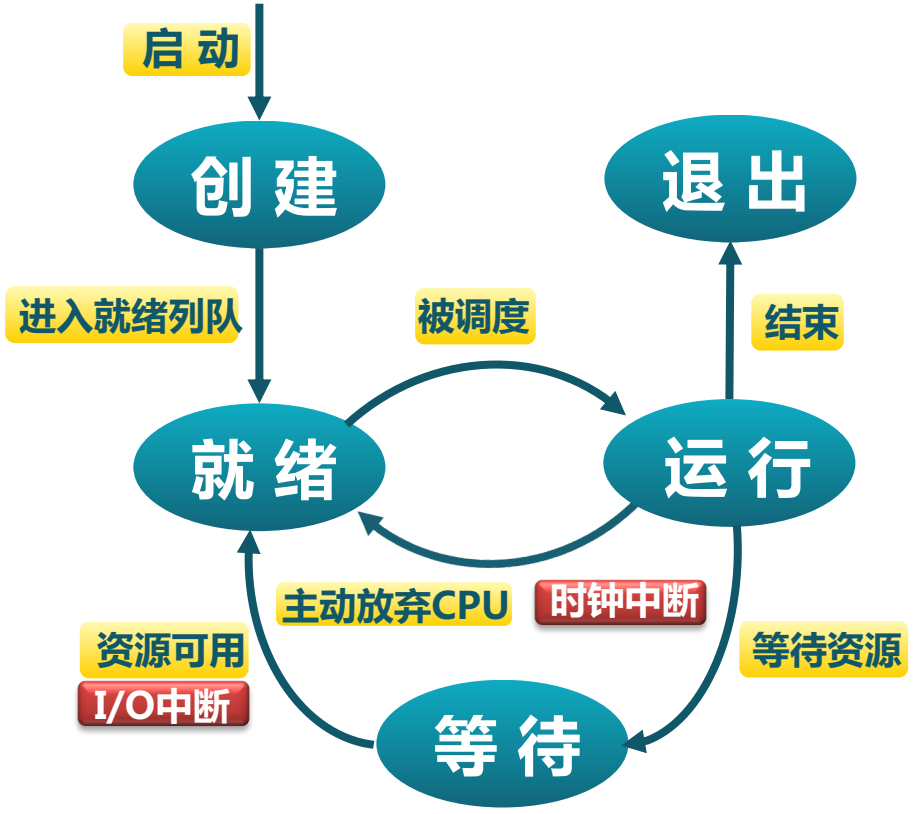
非抢占式调度

- 非抢占式调度
  - 进程运行结束
  - 进程主动放弃CPU使用权：sched_yield()系统调用
  - 进程进入等待状态，例如等待IO、等待资源
- 内核调度器执行调度 do_scheduler()



抢占式调度

- 抢占：不等一个进程运行结束，内核调度另一个进程执行
- 为什么要抢占
  - 让多个进程更加公平使用CPU资源，避免饥饿
  - 交叠使用I/O和计算资源，提升资源利用率
  - 有紧急任务（高优先级进程）要执行
- 何时进行抢占
  - 进程时间片用完
  - 高优先级进程就绪

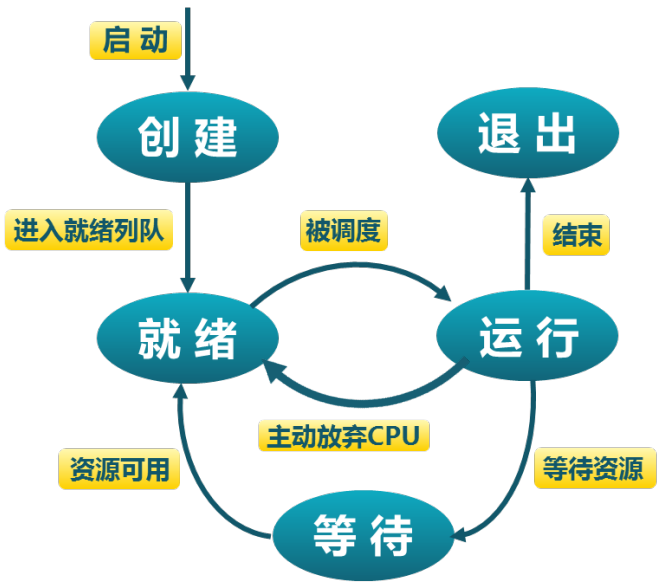


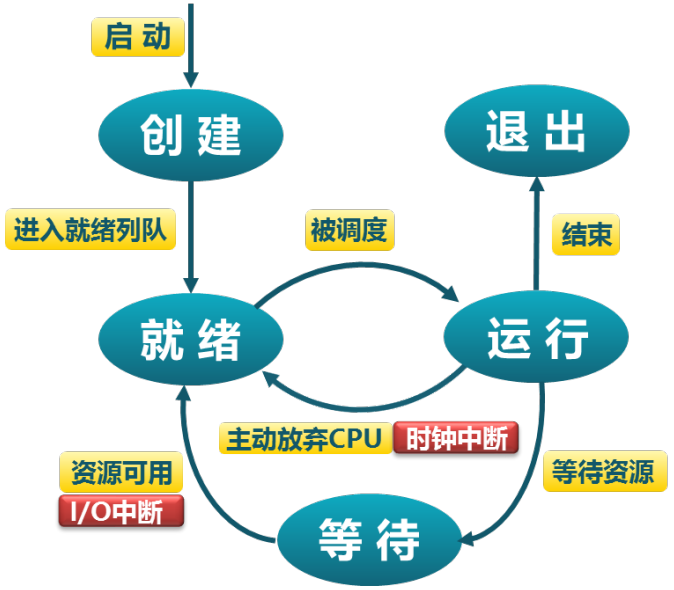
抢占式调度和中断处理

- 中断处理基本流程
  - CPU检查中断条件是否满足
    - 有中断请求
    - CPU允许中断
  - 如果CPU允许中断，关中断，不再响应新中断
  - 保存被中断的现场
  - 判断中断类型，调用执行中断处理程序
    - 时间中断：递减进程时间片，判断时间片是否用完
    - I/O中断：进行I/O，例如将数据从外设拷贝到内核内存中
  - 调用调度器，恢复现场
  - 开中断

调度器

- 进程切换与调度器
  - 保存当前进程的上下文
  - 选择下一个待运行的进程
  - 加载被调度进程的上下文，并跳转执行
- 非抢占式调度与抢占式调度的区别





内容提要

- CPU调度基础
- CPU调度算法

调度准则

- 假设
  - 一个用户运行一个程序，一个程序创建一个进程（单线程）
  - 程序之间是独立的
- 常用指标
  - 吞吐率：每秒处理请求数
  - 响应时间（等待时间）：从提交一个请求到产生响应所用时间（适用于交互式系统）
  - 周转时间：从作业提交到作业完成的时间间隔（适用于批处理系统）
  - 公平性：每个程序是否都有执行机会，防止饥饿

调度准则

- 批处理和交互系统设计目标
  - 保证公平性
    - 每个作业都有机会运行； 没有人会“饥饿”
  - 最大化CPU资源利用率
    - 不包括idle进程
  - 最大化吞吐率
    - 操作数/秒（最小化开销，最大化资源利用率）
  - 最小化周转时间
    - 批处理作业：执行时间（从提交到完成）
  - 缩短响应时间
    - 交互式作业：响应时间（例如，键盘打字）
  - 均衡性
    - 满足用户需求
    - 提升计算机系统各部件利用率

调度准则

- 不同类型计算机系统的需求不一样
  - 服务器
    - 吞吐率
    - 响应时间
    - 公平性
  - 个人计算机
    - 响应时间
  - 工业控制计算机
    - 实时性

先到先服务（FCFS）算法

- 什么是先到先服务？
  - 一直运行到结束
  - 一直运行到阻塞或者主动放弃CPU
  - 用于非抢占式调度，适用于批处理系统
- 例子1
  - P1 = 24s, P2 = 3s, 且P3 = 3s，同时提交，顺序运行
  - 平均周转时间 =（24 + 27 + 30）/ 3 = 27

- 例子2
  - 同样的进程，但是以不同的顺序运行：P2，P3，P1
  - 平均周转时间 =（3 + 6 + 30）/ 3 = 13

先到先服务算法分析

- 优点
  - 实现简单
- 缺点
  - 平均周转时间波动较大
    - 短进程（作业）排的顺序不同，周转时间不同
  - 可能导致I/O资源利用率低
    - CPU密集型进程先执行，导致I/O密集进程没有机会使用I/O设备，进而导致I/O设备空闲，I/O资源利用率低
    - 如果I/O密集进程先执行，是否会导致CPU利用率低？

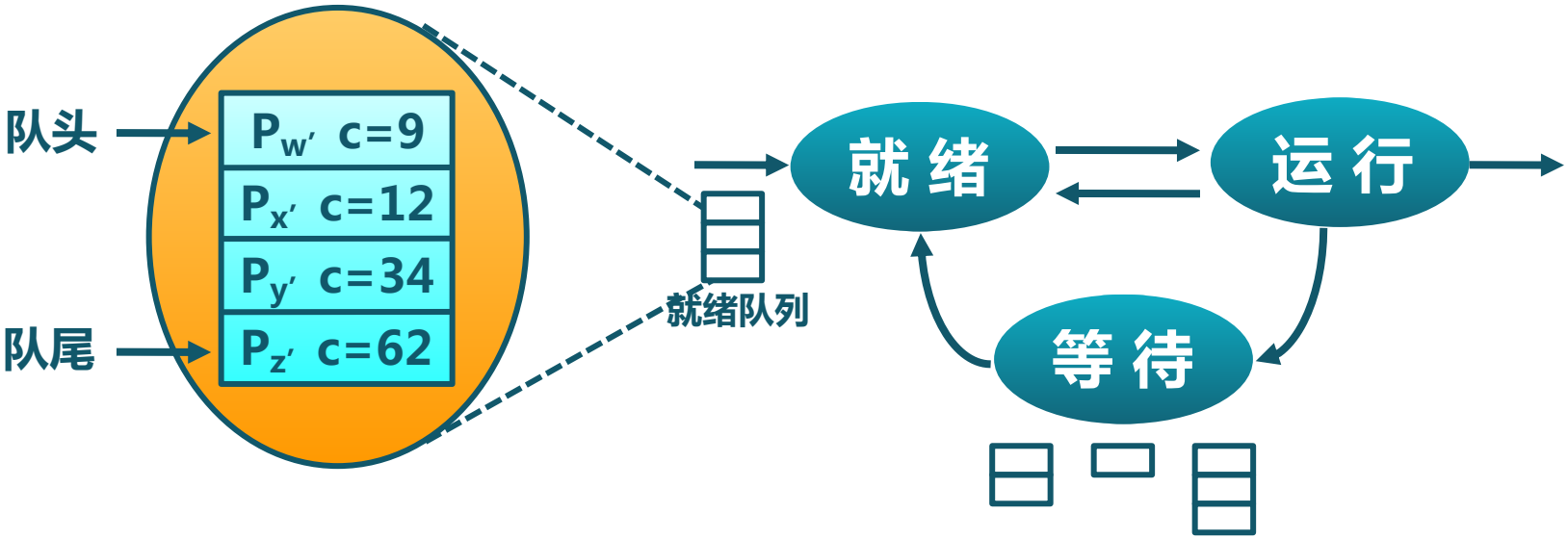
最短时间优先

- 最短时间优先（Short Time to Complete First, STCF）
  - 非抢占调度
- 例子
  - P1 = 6s, P2 = 8s, P3 = 7s, P4 = 3s
  - 所有作业同时到达
  - 平均响应时间 =（0 + 3 + 9 + 16）/ 4 = 7

FCFS算法平均响应时间 =（0 + 6 + 14 + 21）/ 4 = 10.25

最短剩余时间优先

- 最短剩余时间优先（Short Remaining Time to Complete First, SRTCF）
  - 抢占式调度
  - 选择就绪队列中剩余时间最短进程占用CPU进入运行状态
  - 就绪队列按剩余时间来排序

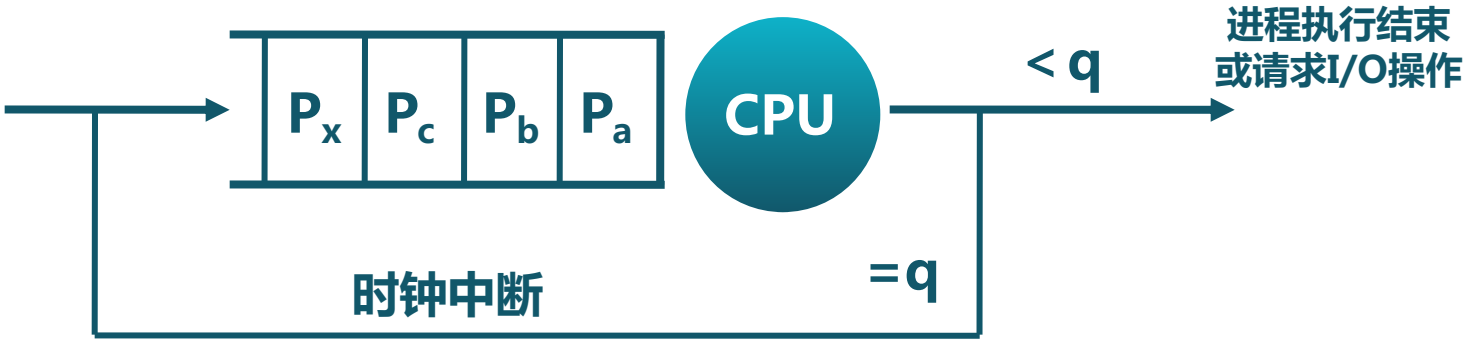


STCF和SRTCF分析

- 优点
  - 平均响应时间短
- 可能会造成饥饿
  - 连续的短进程会使长进程无法获得CPU资源
- 挑战
  - 如何预估作业的执行时间？
    - 批处理系统中，用户提交作业时会设定时限，将其作为执行时间

时间片轮转算法（RR，Round Robin）

- 和FCFS算法类似，但是增加了时间片
- 时间片结束时，调度器按FCFS算法切换到下一个就绪进程
- 轮转调度是抢占式调度
- 多用于交互式系统

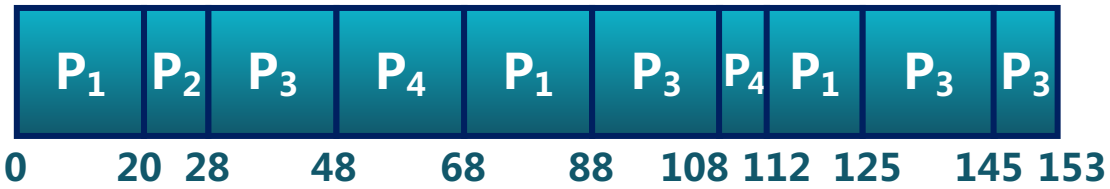


时间片为20的轮转算法示例

- 示例: 4个进程的执行时间如下

P1	53
P2	8
P3	68
P4	24

- 甘特图



等待时间 $P_1=(68-20)+(112-88)=72$

$P_2=(20-0)=20$

$P_3=(28-0)+(88-48)+(125-108)=85$

$P_4=(48-0)+(108-68)=88$

平均等待时间 = (72+20+85+88)/4=66.25

时间片长度选择

- 大时间片
  - 等待时间过长
  - 极端情况下退化为FCFS
- 小时间片
  - 响应时间快
  - 产生大量上下文切换，影响系统吞吐
- 经验规则
  - 选择一个合适的时间片，使上下文切换开销处于1%以内
  - /proc/sys/kernel/sched_latency_ns

不同时间片的调度效果

- 示例: 4个进程的执行时间如下

P1	53
P2	8
P3	68
P4	24

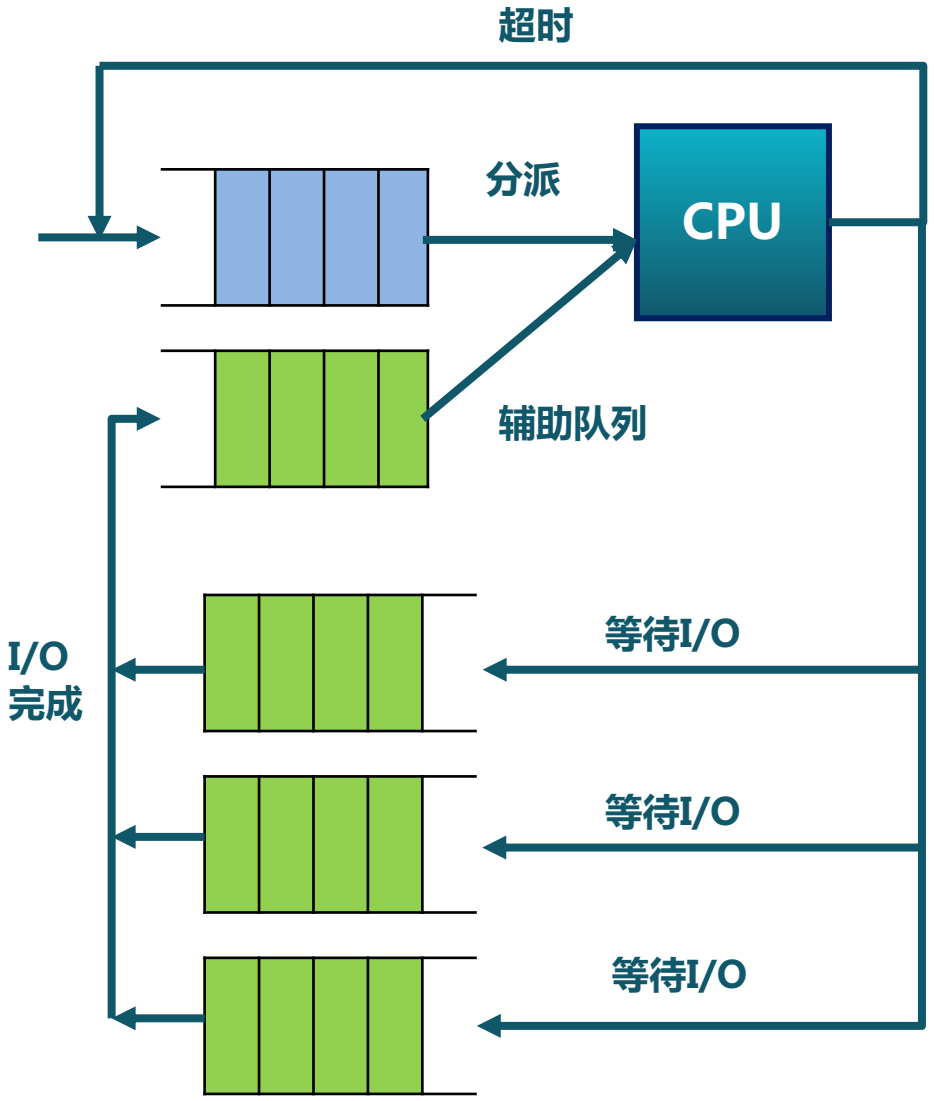
- 假设上下文切换开销为0，不同时间片以及FCFS对应的调度效果

时间片	P_1	P_2	P_3	P_4	平均等待时间
RR(q=1)	84	22	85	57	62
RR(q=5)	82	20	85	58	61.25
RR(q=8)	80	8	85	56	57.25
RR(q=10)	82	10	85	68	61.25
RR(q=20)	72	20	85	88	66.25
BestFCFS	32	0	85	8	31.25
WorstFCFS	68	145	0	121	83.5

=最短时间优先

虚拟轮转算法（Virtual Round Robin）

- 引入辅助队列
  - FIFO（先入先出）
- I/O密集型进程
  - 进入辅助队列（而不是就绪队列）以备调度
- 引入优先级
  - 辅助队列比就绪队列有更高的优先级

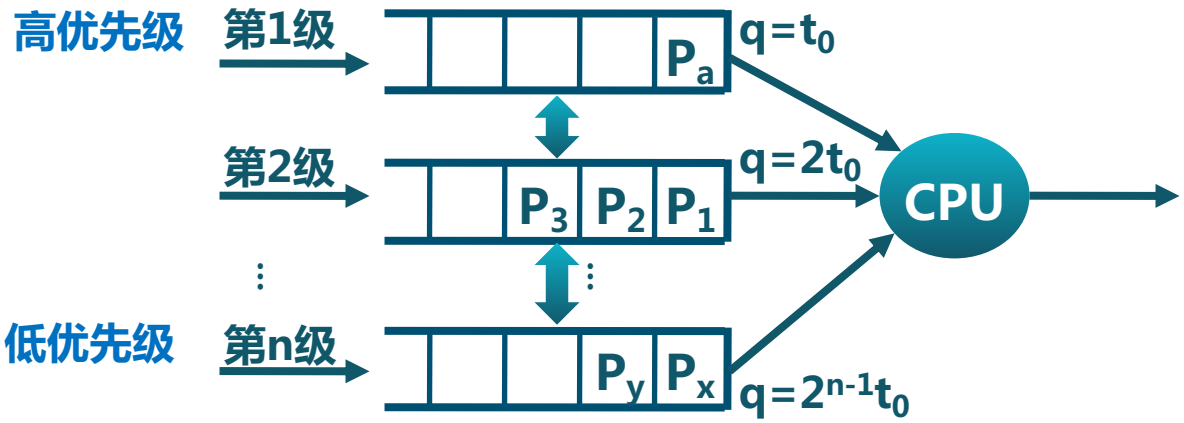


多级队列（MLQ）与优先级调度

- 将就绪队列分为多个独立的子队列，每个队列可有自己的调度算法
  - 前台任务（交互式）-RR，后台任务（批处理）-FCFS
- 队列之间
  - 最高优先级优先
    - 先调度高优先级，再调度低优先级
    - 可能导致饥饿：低优先级进程一直得不到执行
  - 潜在问题：优先级反转
    - 高优先级进程所需资源被低优先级进程持有，等待低优先级执行
    - 中优先级进程优先执行，表现为优先级高于高优先级进程
    - 解决方法：高优先级进程由于等待资源被阻塞时，自动提升低优先级进程的优先级（优先级继承）

多级反馈队列(MLFQ)算法

- 进程可在不同队列间动态移动的多级队列算法，实现优先级动态调整
- 特征
  - 每一级队列分配一个时间片，时间片大小随优先级级别增加而减小
  - 进程默认进入第1级队列。如果进程在当前队列中累计运行的时间超过最大运行时间，则降到下一个优先级队列中
  - CPU密集型进程的优先级下降很快，I/O密集型进程停留在高优先级
  - 进程等待时间过长时，将进程提升至较高优先级，避免饥饿



公平共享调度(FSS, Fair Share Scheduling)

- 用户A运行任务A-1和A-2，用户B运行任务B-1，如果要让用户A和B平分CPU资源，则应该采取哪种调度算法？
  - RR？
  - 优先级调度？
- 考虑用户对系统资源使用的比例，进行配额管理
  - 假设总份额为100，用户A和B各被分配的份额是50
  - 任务A-1和任务A-2各占用户A的份额的一半
  - 任务B-1使用用户B的所有份额

公平共享调度(FSS, Fair Share Scheduling)

- FSS基于份额控制用户对系统资源的访问
  - 份额管理（假设份额为100）
    - 按一定比例（proportion）在不同用户间分配份额
    - 需要为突发任务预留份额（reservation）
    - 可以为用户设置份额上限（limit）
- 优先级和份额的区别
  - 优先级表明任务执行的先后，可以优化周转时间、响应时间，但无法确保任务能获得应得的资源比例
  - 份额对应任务使用的资源比例
  - 当用户关注的不是周转时间、响应时间，而是关注在总资源中占用的比例时，应该使用FSS调度

彩票调度

- 彩票方法
  - 一种FSS策略的实现
  - 给每个作业一定数量的彩票（份额）
  - 随机抽取一张中奖彩票（winning ticket），中奖的作业获得CPU
  - 有较多彩票的作业能获得的调度机会多，随着调度次数增加，每个作业的彩票数量占比趋近调度次数占比
  - 为了近似SRTCF，给短作业更多的彩票
  - 为了避免“饥饿”，给每个作业至少一张彩票

调度算法总结

- 调度算法比较

算法	适用系统	平均响应时间	公平性	潜在问题
FCFS	批处理	长	可能造成饥饿	
STCF	批处理	短	长进程可能饥饿	需要预测作业执行时间
SRTCF	批处理	短	长进程可能饥饿	需要预测作业执行时间
轮询算法	交互式	短，IO进程响应时间较长	公平对待	时间片小会导致吞吐率低
虚拟轮询算法	交互式	短	对IO密集进程友好	
多级队列优先级调度	交互式和批处理共存	低优先级队列任务响应时间长	可能造成饥饿	优先级反转
多级反馈队列	交互式和批处理共存	短	兼顾长短作业	
彩票算法	交互式和批处理共存	短	兼顾长短作业，考虑资源比例	

实时调度

- 实时：确定性、可预测性
- 两种类型的实时
  - 硬实时（hard deadline）
    - 必须满足，否则会导致错误
  - 软实时（soft deadline）
    - 大多时候满足，没有强制性
- 实时任务

- 周期性实时任务

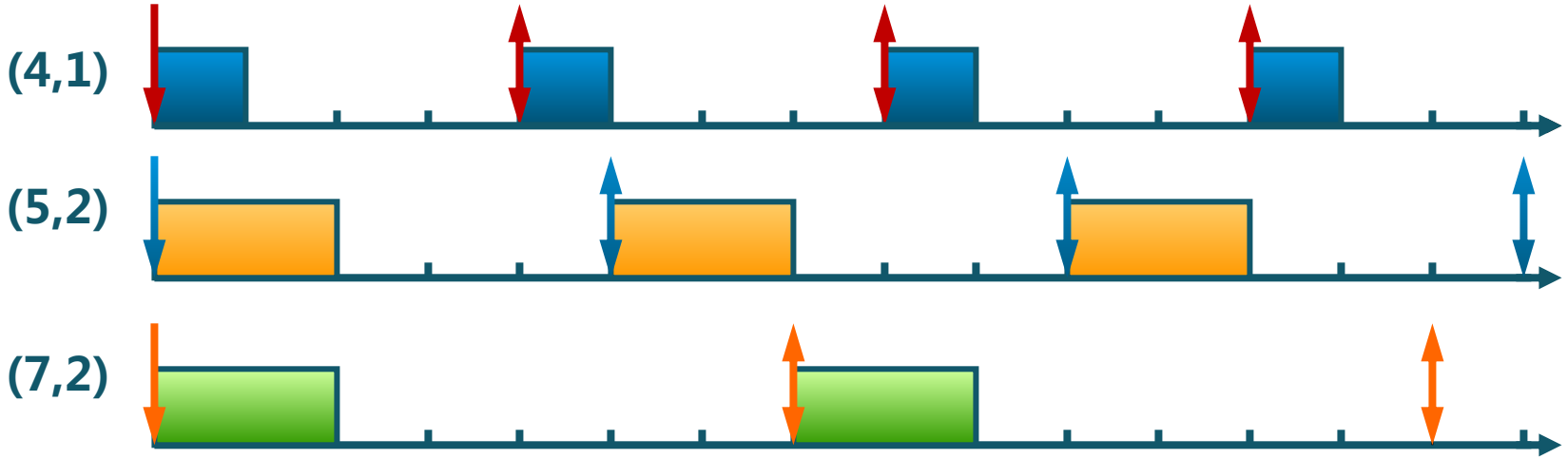
实时调度

- 准入控制（Admission Control）
  - 只有当系统能够保证所有进程的实时性的前提下，新的实时进程才会被接纳/准入
  - 如果满足下面的条件，作业就是可调度的：

$$\sum \frac{C_i}{T_i} \leq 1$$

其中C_i = 计算时间，T_i = 周期

- **示例：1/4 + 2/5 + 2/7 = 131/140 < 1**



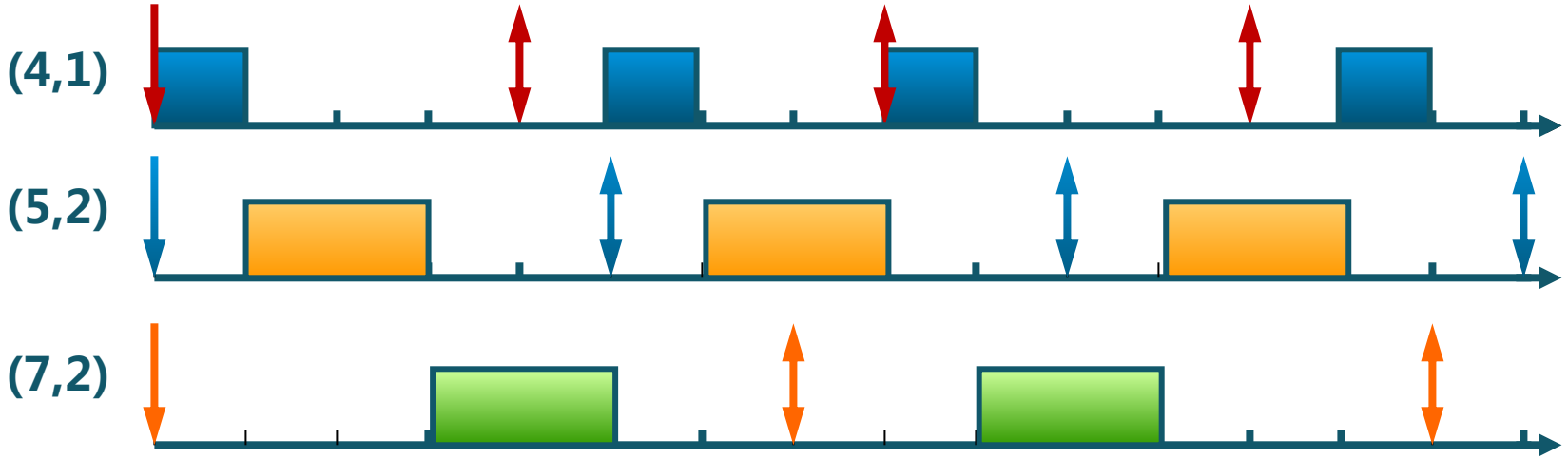
实时调度

- 准入控制（Admission Control）
  - 只有当系统能够保证所有进程的实时性的前提下，新的实时进程才会被接纳/准入
  - 如果满足下面的条件，作业就是可调度的：

$$\sum \frac{C_i}{T_i} \leq 1$$

其中C_i = 计算时间，T_i = 周期

- **示例：1/4 + 2/5 + 2/7 = 131/140 < 1**



最早截止时间优先调度（Earliest Deadline First Scheduling, EDF）

- 假设
  - 当进程运行时，它会提供其需要完成运行的截止时间
  - 不一定是周期性进程
- EDF的基本思想
  - 根据截止时间对就绪的进程进行排序
  - 运行列表中的第一个进程（最早截止时间优先）
  - 当新的进程就绪时，并且其截止时间快来临时，它会抢占当前进程，保证任务的实时性
  - 和SRTCF的区别？
- 例子
  - P1需要在第30s之前结束，P2需要在第40s之前结束，P3需要在第50s之前结束
  - P1先运行

扩展：BSD调度

- 基于多级反馈队列和优先级动态调整
- 优先级每秒都会重新计算
  - P_i = base + CPU_i / 2 + nice, where CPU_i = CPU_{i-1} / 2
  - P值越小，优先级越高
  - base是进程的基础优先级
  - CPU_i是进程在第i段时间间隔的利用率
  - 不同程序可以分配不同的基础优先级和nice值

扩展：Linux中的调度

- 分时共享调度（非实时任务）
  - Complete Fairness Schedule（CFS）
  - 根据nice值（-20到19），设定每个进程的优先级
  - 内核根据进程的nice值计算virtual time，nice值为0时，virtual runtime和物理运行时间一致，nice值越大，virtual runtime越大，但实际运行物理时间小，反之，实际运行物理时间大
  - 通过每个进程的虚拟时间（virtual time）来选择被调度的进程，virtual time越小的进程越先调度
- 实时调度（实时任务）
  - 软实时：尽量使进程在限定的时间前完成，不保证总能满足
  - FIFO策略和Round Robin策略

总结

- 调度基础
  - 抢占式调度与非抢占式调度
  - 中断在调度中的作用
- 调度算法
  - 优化问题，不同的系统决定了不同的调度目标
  - 批处理系统
    - FCFS简单易实现
    - STCF和SRTCF可以获得最小的平均响应时间
  - 交互式系统
    - 轮询算法及改进算法有利于公平性，较小的时间片有利于提高I/O利用率
  - 并存系统
    - 多队列与优先级调度及其变种，普遍存在于多个系统中
    - 彩票调度考虑的资源占用比例
  - 实时调度依赖于准入控制