# 第七讲 进程管理与单处理器调度

## 第二节 单处理器调度

向勇 陈渝 李国良 任炬

2023年春季

## 提纲

# 1. 处理机调度概念

- 处理机调度的时机和策略
- 比较调度算法的准则

2. 调度算法

# CPU资源的时分复用

- 进程切换：CPU资源的当前占用者切换
  - 保存当前进程在PCB中的执行上下文(CPU状态)
  - 恢复下一个进程的执行上下文
- 处理机调度
  - 从就绪队列中挑选下一个占用CPU运行的进程
  - 从多个可用CPU中挑选就绪进程可使用的CPU资源
- 调度器：挑选就绪进程的内核函数
- 调度策略
  - 依据什么原则挑选进程?

# 调度时机

- 内核执行调度的条件
  - 进程从运行状态切换到等待/就绪状态
  - 进程被终结了
- 非抢占系统
  - 当前进程主动放弃CPU时
- 可抢占系统
  - 中断请求被服务例程响应完成时

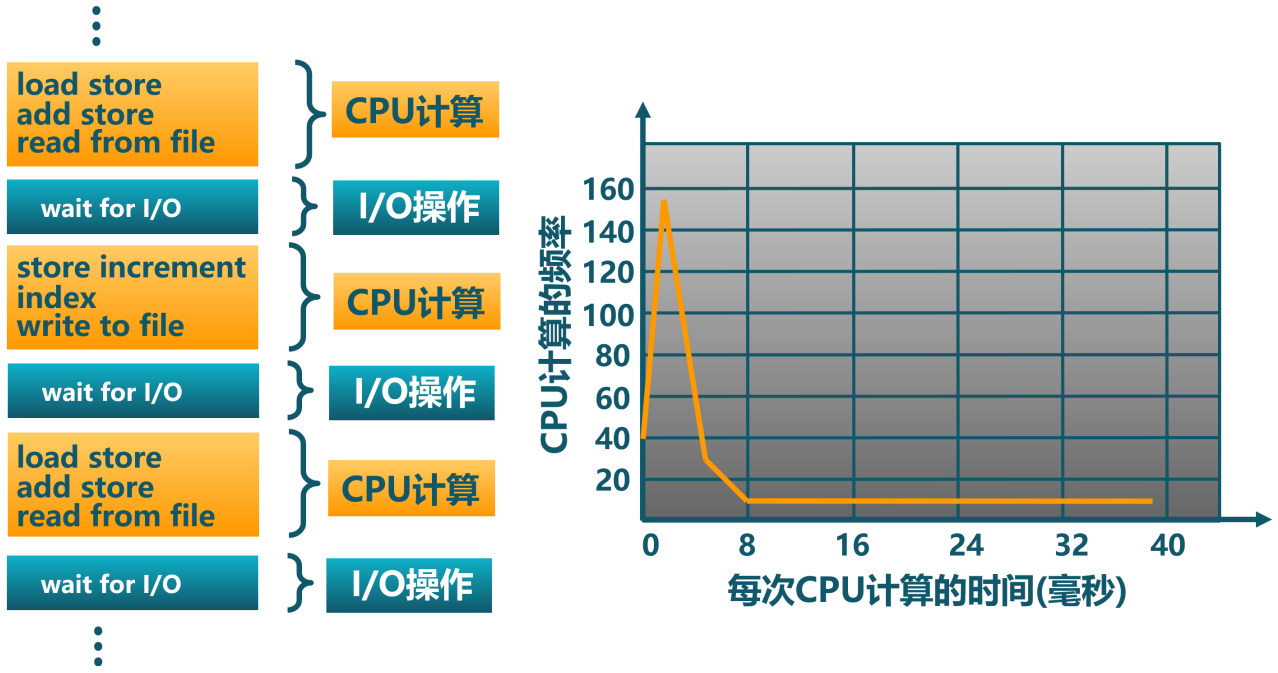
# 调度策略

确定如何从就绪队列中选择下一个执行进程

- 要解决的问题
  - 通过什么样的准则来选择?
  - 挑选就绪队列中的哪一个进程?
- 调度算法
  - 在内核调度中实现的调度策略
- 比较调度算法的准则
  - 哪一个策略/算法较好?

# 处理机资源的使用模式

- 进程在CPU计算和I/O操作间交替
  - 每次调度决定在下一个CPU计算时将哪个工作交给CPU
  - 在时间片机制下，进程可能在结束当前CPU计算前被迫放弃CPU



## 提纲

1. 处理机调度概念

- 处理机调度的时机和策略

## 比较调度算法的准则

2. 调度算法

# 比较调度算法的准则

- CPU使用率：CPU处于忙状态的时间百分比
- 吞吐量：单位时间内完成的进程数量
- 周转时间：进程从初始化到结束(包括等待)的总时间
- 就绪等待时间：就绪进程在就绪队列中的总时间
- 响应时间：从提交请求到产生响应所花费的总时间
- 公平：进程占用相同的资源，如CPU时间等

# 比较调度算法的吞吐量与延迟准则

- 调度算法的要求： 希望“更快”的服务
- 什么是更快?
  - 传输文件时的高带宽，调度算法的高吞吐量
  - 玩游戏时的低延迟，调度算法的低响应延迟
  - 这两个因素相互影响
- 与水管的类比
  - 低延迟：喝水的时候想要一打开水龙头水就流出来
  - 高带宽：给游泳池充水时希望从水龙头里同时流出大量的水，并且不介意是否存在延迟

# 比较调度算法的响应时间准则

- 减少响应时间
  - 及时处理用户的输入请求，尽快将输出反馈给用户
- 减少平均响应时间的波动
  - 在交互系统中，可预测性比高差异低平均更重要
- 低延迟调度改善了用户的交互体验
  - 如果移动鼠标时，屏幕中的光标没动，用户可能会重启电脑
- 响应时间是操作系统的计算延迟

# 比较调度算法的吞吐量准则

- 增加吞吐量
  - 减少开销（操作系统开销，上下文切换）
  - 系统资源的高效利用（CPU，I/O设备）
- 减少就绪等待时间
  - 减少每个就绪进程的等待时间
- 操作系统需要保证吞吐量不受用户交互的影响
  - 即使存在许多交互任务
- 吞吐量是操作系统的计算带宽

# 比较调度算法的公平准则

一个用户比其他用户运行更多的进程时，公平吗？怎么办?

- 公平的定义
  - 保证每个进程占用相同的CPU时间
  - 保证每个进程的就绪等待时间相同
- 公平通常会增加平均响应时间

## 提纲

1. 处理机调度概念

# 2. 调度算法

- 先来先服务算法FCFS、短作业优先算法SJF
- 最短剩余时间算法SRT、最高响应比优先算法HRRN
- 时间片轮转算法RR
- 多级队列调度算法MQ、多级反馈队列算法MLFQ
- 公平共享调度算法FSS

# 先来先服务调度算法FCFS

FCFS: First Come, First Served

- 依据进程进入就绪状态的先后顺序排列
- 进程进入等待或结束状态时，就绪队列中的下一个进程占用CPU
- 指标
  - FCFS算法的周转时间

# 先来先服务调度算法示例

- 示例：3个进程，计算时间分别为12,3,3

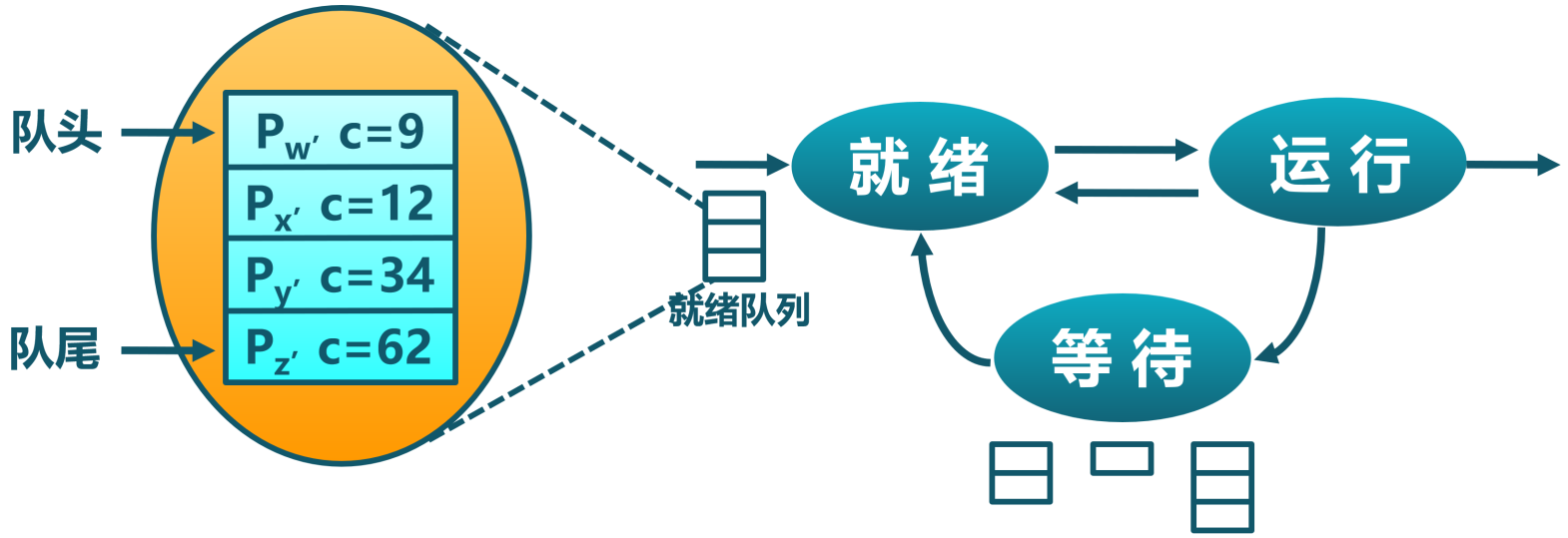
# 先来先服务调度算法的特征

- 优点：简单
- 缺点
  - 平均等待时间波动较大
  - 短作业/任务/进程可能排在长进程后面
  - I/O资源和CPU资源的利用率较低
    - CPU密集型进程会导致I/O设备闲置时，I/O密集型进程也等待

# 短作业优先调度算法SJF

Short Job First

- 选择就绪队列中执行时间最短作业/进程占用CPU进入运行状态
- 就绪队列按预期的执行时间来排序

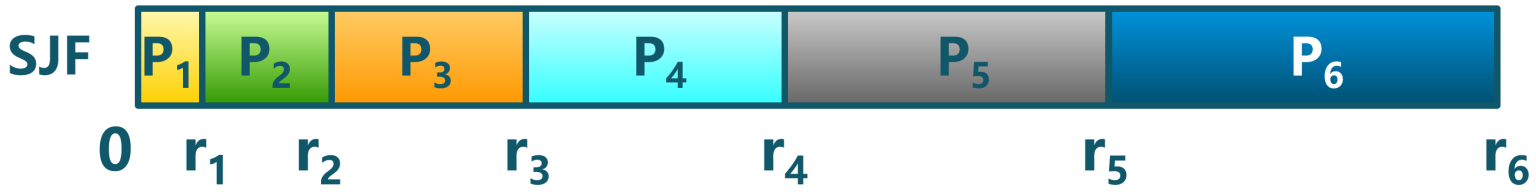


# 短作业优先调度算法的特征

## 具有最优平均周转时间

修改作业/进程执行顺序可能减少平均等待时间吗?

**SJF算法中一组进程的平均周转时间**




周转时间=$(r_1+r_2+r_3+r_4+r_5+r_6)/6$


周转时间=$(r_1+r_2+r_4-c_3+r_5-c_3+r_4+c_4+c_5+r_6)/6$

$=(r_1+r_2+r_3+r_4+r_5+r_6+(c_4+c_5-2c_3))/6$

# 短作业优先调度算法的特征

- 可能导致饥饿
  - 连续的短作业/进程流会使长作业/进程无法获得CPU资源
- 需要预知未来
  - 如何预估下一个CPU计算的持续时间?
  - 简单的解决办法：询问用户
    - 用户欺骗就杀死相应进程
    - 用户不知道怎么办?

# 短作业优先算法的执行时间预估

- 用历史的执行时间来预估未来的执行时间

$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$，其中$0 \leq \alpha \leq 1$

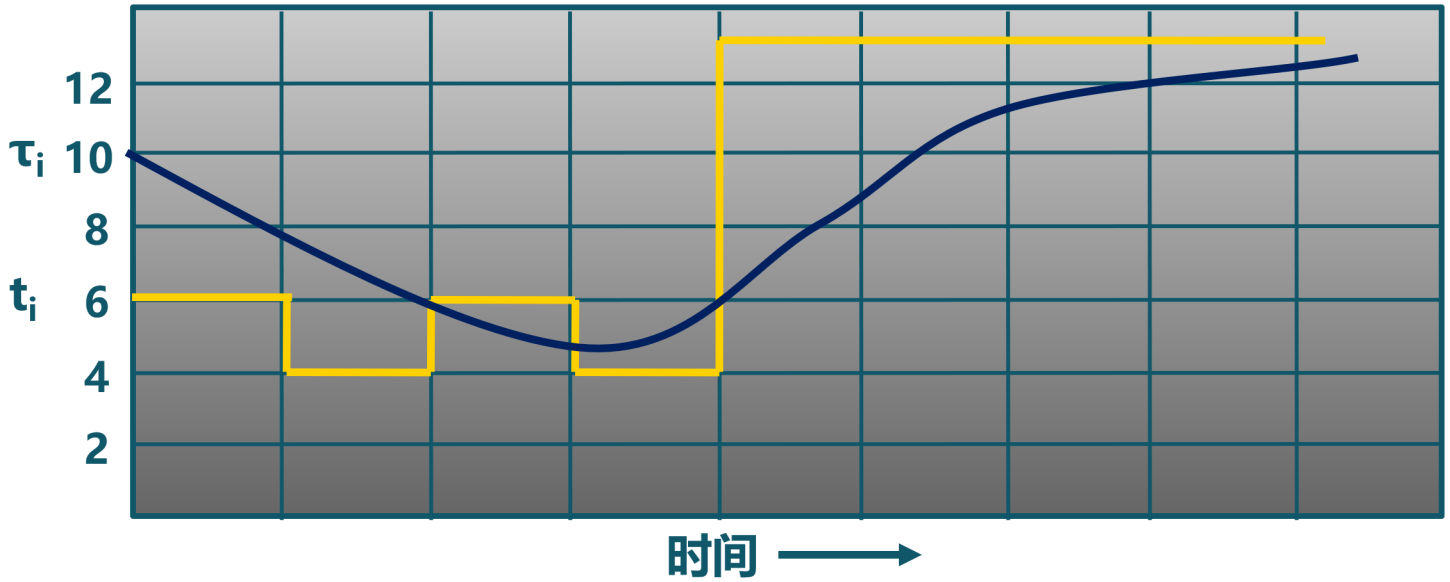
$t_n$ -- 第n次的CPU计算时间

$\tau_{n+1}$ -- 第n+1次的CPU计算时间预估

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\alpha t_{n-1} + (1-\alpha)(1-\alpha)\alpha t_{n-2} + \ldots$$

# 短作业优先算法的执行时间预估

- 执行时间预估




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 实际CPU执行时间 ($t_i$) | | 6 | 4 | 6 | 4 | 13 | 13 | 13 | … |
| 预估CPU执行时间($\tau_i$) | 10 | 8 | 6 | 5 | 6 | 9 | 11 | 12 | … |

# 最短剩余时间算法SRT

Shortest Remaining Time, SRT

- SRT支持抢占调度机制，即有新的进程就绪，且新进程的服务时间小于当前进程的剩余时间，则转到新的进程执行。

# 最高响应比优先算法HRRN

Highest Response Ratio Next，HRRN

- 高响应比优先调度算法主要用于作业调度
- 该算法是对FCFS调度算法和SJF调度算法的一种综合平衡，同时考虑每个作业的等待时间和估计的运行时间
- 在每次进行作业调度时，先计算后备作业队列中每个作业的响应比，从中选出响应比最高的作业投入运行。

# 最高响应比优先算法HRRN

- 选择就绪队列中响应比R值最高的进程

$$\mathrm{R} = \ (\mathrm{w} + s)/s$$

w: 就绪等待时间(waiting time)
s: 执行时间(service time)

- 在短作业优先算法的基础上改进
- 关注进程的等待时间
- 防止无限期推迟

## 提纲

1. 处理机调度概念
2. 调度算法

- 先来先服务算法FCFS、短作业优先算法SJF
- 最短剩余时间算法SRT、最高响应比优先算法HRRN

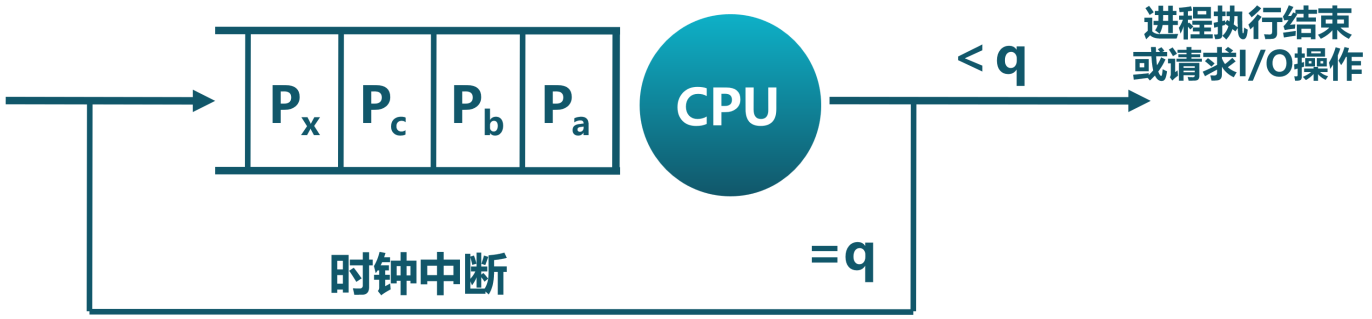
# 时间片轮转算法RR

- 多级队列调度算法MQ、多级反馈队列算法MLFQ
- 公平共享调度算法FSS

# 时间片轮转算法RR

RR, Round-Robin

- 时间片
  - 分配处理机资源的基本时间单元
- 算法思路
  - 时间片结束时，按FCFS算法切换到下一个就绪进程
  - 每隔(n – 1)个时间片进程执行一个时间片q



# 时间片轮转算法示例

**示例: 4个进程的执行时间如下**

| | |
|---|---|
| P1 | 53 |
| P2 | 8 |
| P3 | 68 |
| P4 | 24 |

**执行过程如下:**


**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** = $(72+20+85+88)/4=66.25$

# 时间片轮转算法的时间片长度参数

- RR算法开销： 额外的上下文切换
- 时间片太大
  - 等待时间过长，极限情况退化成FCFS
- 时间片太小
  - 反应迅速，但产生大量上下文切换
  - 大量上下文切换开销影响到系统吞吐量
- 时间片长度选择目标
  - 选择一个合适的时间片长度
  - 经验规则：维持上下文切换开销处于1%以内

# 比较FCFS和RR

示例: 4个进程的执行时间如下

| | |
|---|---|
| P1 | 53 |
| P2 | 8 |
| P3 | 68 |
| P4 | 24 |

假设上下文切换时间为零

FCFS和RR各自的平均等待时间是多少?

| 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 |

## 提纲

1. 处理机调度概念
2. 调度算法

- 先来先服务算法FCFS、短作业优先算法SJF
- 最短剩余时间算法SRT、最高响应比优先算法HRRN
- 时间片轮转算法RR

## 多级队列调度算法MQ、多级反馈队列算法MLFQ

- 公平共享调度算法FSS

# 多级队列调度算法MQ

MQ, MultiQueue

- 就绪队列被划分成多个独立的子队列
  - 如：前台进程(交互)子队列、后台进程(批处理)子队列
  - 同一优先级的进程属于某个队列，且不能跨越队列
- 每个队列拥有自己的调度策略
  - 如：前台进程–RR、后台进程–时间片大的RR/FCFS
- 规则1：如果A的优先级 > B的优先级，运行A（不运行B）。
- 规则2：如果A的优先级 = B的优先级，轮转运行A和B。

# 多级队列调度算法MQ

- 队列间的调度
  - 固定优先级
    - 先处理前台(交互)进程，然后处理后台进程
    - 可能导致饥饿
  - 时间片轮转
    - 每个队列都得到一个确定的能够调度其进程的CPU总时间
    - 如：80%CPU时间用于前台进程，20%CPU时间用于后台进程

# 多级反馈队列调度算法MLFQ

MLFQ, Multi-Level Feedback Queue

- 1962年，MIT教授Corbato首次提出多级反馈队列，应用于兼容时分共享系统（CTSS-Compatible Time-Sharing System）
- 解决两方面的问题
  - 如何在不知道工作要运行多久的情况下，优化周转时间
  - 如何降低响应时间，给交互用户很好的交互体验

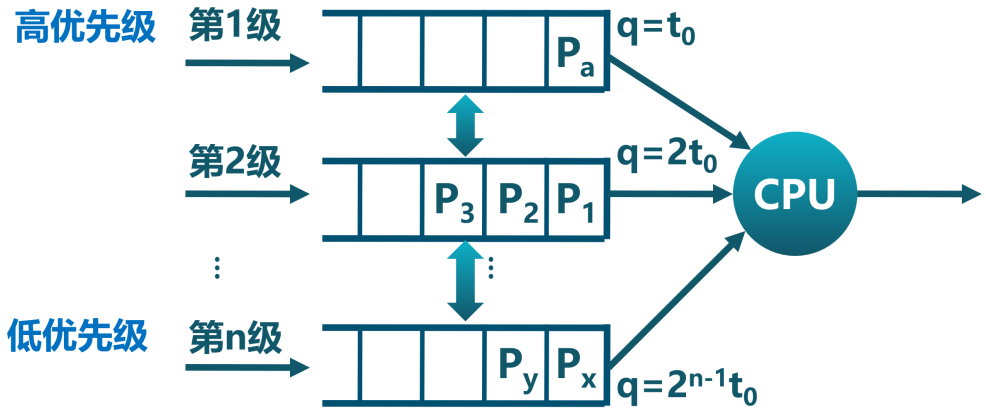
# 多级反馈队列调度算法MLFQ

- 关键问题：没有完备的知识如何调度?
  - 对进程工作长度未知情况下，如何构建能同时减少响应时间和周转时间的调度程序?
- 启发：从历史中学习
  - 用历史经验预测未来
- 继承Multi Queue的调度规则
  - 如果A的优先级 > B的优先级，运行A（不运行B）
  - 如果A的优先级 = B的优先级，轮转/FIFO运行A和B

# 多级反馈队列调度算法MLFQ

基本调度规则

- 工作进入系统时，放在最高优先级（最上层队列）
- 如进程在当前的时间片没有完成，则降到下一个优先级
- 如果工作在其时间片以内主动释放CPU，则优先级不变
- 时间片大小随优先级级别增加而增加

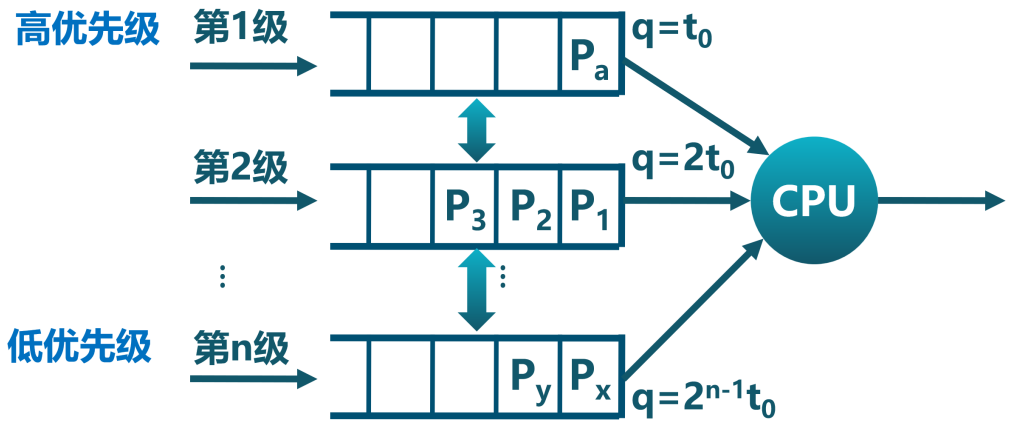


# 三个优先级队列的MLFQ调度例子

- CPU密集型进程首先进入最高优先级队列；
- 执行1ms时间片后，调度器将进程的优先级减1，进入次高优先级队列；
- 执行2ms时间片后，进入系统的最低优先级队列，一直留在那里，按4ms时间片执行。

# 多级反馈队列调度算法MLFQ

- MLFQ算法的特征
  - 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
- 潜在问题
  - CPU密集型进程会饥饿
  - 恶意进程会想办法留在高优先级



# 多级反馈队列调度算法MLFQ

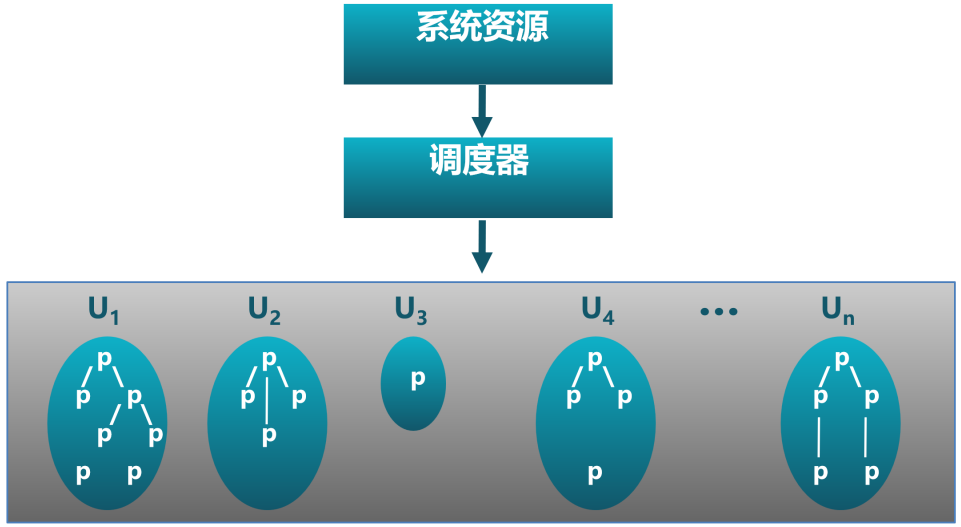
基本调度规则

- 如果A的优先级 > B的优先级，运行A（不运行B）
- 如果A的优先级 = B的优先级，轮转/FIFO运行A和B
- 工作进入系统时，放在最高优先级（最上层队列）
- 一旦工作用完了其在某一层中的时间配额（无论中间主动放弃了多少次CPU），就降低其优先级（移入低一级队列）
- 经过一段时间S，就将系统中所有工作重新加入最高优先级队列

# 公平共享调度算法FSS

FSS, Fair Share Scheduling

- 控制用户对系统资源的访问
  - 不同用户拥有多个进程
  - 按用户优先级分配资源
  - 保证不重要的用户无法垄断资源
  - 未使用的资源按比例分配



# 调度算法的特征

- 先来先服务算法
  - 平均等待时间较差
- 短作业优先算法
  - 平均周转时间最小
  - 需要精确预测计算时间
  - 不允许抢占；可能导致饥饿
- 最短剩余时间算法
  - 对短作业优先算法的改进，允许抢占
  - 可能导致饥饿

# 调度算法的特征

- 最高响应比优先算法
  - 基于短作业优先调度，不可抢占
  - 同时考虑每个作业的等待时间和估计的运行时间
- 时间片轮转算法
  - 公平，但是平均等待时间较差
- 多级反馈队列算法
  - 多种算法的集成
- 公平共享调度算法
  - 公平是第一要素

# 小结

1. 处理机调度概念

- 处理机调度的时机和策略、比较调度算法的准则

2. 调度算法

- FCFS、SJF、SRT、HRRN
- RR
- MQ、MLFQ、FSS