2.3 操作系统调度

2.3.1 基本概念
2.3.2 单核调度
2.3.3 实时系统调度
2.3.4 多核调度
2.3.5 操作系统实例

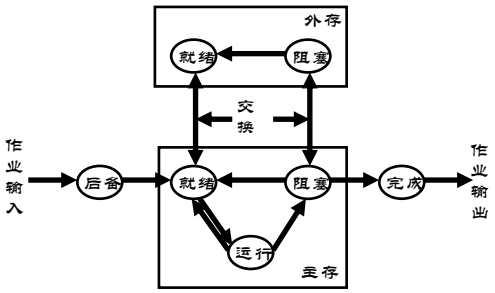
2.3.1 基本概念(1)

- 什么是调度
  - Q1：程序员小明想在单核计算机上运行一个需要执行30分钟的机器学习程序，同时还向打开播放器收听音乐
    - 轮流执行
  - Q2：如果仅有8个处理器，如何运行168个任务？
    - 下一个要执行的任务是哪一个？
    - 执行该任务的CPU是哪一个？
    - 每个任务执行多长时间？

2.3.1 基本概念(2)

- 处理机调度级别
  - 高级调度（作业调度）
    - 选择哪个作业进入就绪队列
    - 调度不频繁
  - 低级调度（进程调度）
    - 选择哪个进程可以占有CPU
    - 调度频繁
  - 中级调度（交换调度）
    - 进程换入换出
    - 提高内存利用率和系统吞吐量

2.3.1 基本概念(3)



作业状态及其转换

2.3.1 基本概念(4)

- 调度方式
  - 非剥夺方式
    - 实现简单，系统开销小，适合于批处理系统
  - 剥夺方式
    - 优先级/时间片，分时系统/实时系统
- 进程调度的功能
  - 记录系统中各进程的执行状况
  - 选择进程真正占有CPU
  - 进行进程上下文的切换

2.3.1 基本概念(5)

- 调度时机
  - 现行进程完成执行或由于某种错误而中止运行
  - 正在执行的进程提出I/O请求，等待I/O完成
  - 分时系统中按照时间片轮转调度策略，分配给进程的时间片用完
  - 优先级调度策略中，进程有更高优先级进程变为就绪
  - 进程执行了某种操作原语，如阻塞原语或唤醒原语时，可能引起进程调度

2.3.1 基本概念(6)

- 处理机调度准则
  - CPU利用率高
  - 吞吐量高
  - 周转时间/平均周转时间短
  - 等待时间/平均等待时间短
  - 响应时间短
  - 批处理系统：增加系统吞吐量和提高系统资源的利用率
  - 分时系统：保证每个分时用户的响应时间

106

2.3.2 单核调度

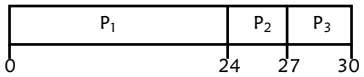
2.3.2.1 FCFS调度算法
2.3.2.2 SJF调度算法
2.3.2.3 高响应比优先调度算法
2.3.2.4 优先级调度算法
2.3.2.5 时间片轮转调度算法
2.3.2.6 多级队列调度算法
2.3.2.7 多级反馈队列调度算法

107

2.3.2.1 FCFS调度算法(1)

- 采用非剥夺调度方式
- 既可用于作业调度，也可用于进程调度
- 示例：3个进程的到达顺序及运行时间为

进程：	P_1	P_2	P_3
运行时间：	24	3	3

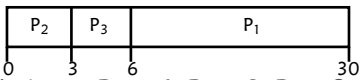


等待时间： $P_1 = 0; P_2 = 24; P_3 = 27$
平均等待时间： $(0 + 24 + 27)/3 = 17$

108

2.3.2.1 FCFS调度算法(2)

- 若进程到达顺序为P_2, P_3, P_1，则



等待时间： $P_1 = 6; P_2 = 0; P_3 = 3$
平均等待时间： $(6 + 0 + 3)/3 = 3$

- 简单，但效率不高
- 有利于长作业（进程），不利于短作业（进程），容易被大作业（进程）垄断，使得平均等待时间很长

109

2.3.2.2 SJF调度算法(1)

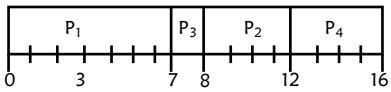
- 考虑作业（进程）运行时间
- 在长期调度中频繁使用
- 既可采用非剥夺调度方式，也可采用剥夺调度方式（Shortest-Remaining-Time-First，SRTF）
- 示例：4个进程的到达时间及运行时间为

进程	到达时间	运行时间
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

110

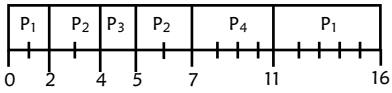
2.3.2.2 SJF调度算法(2)

- 非剥夺方式



平均等待时间 $(0 + 6 + 3 + 7)/4 = 4$

- 剥夺方式



平均等待时间 $(9 + 1 + 0 + 2)/4 = 3$

111

2.3.2.2 SJF调度算法(3)

- 优点
  - SJF是一种最优算法，它可以获得最小平均等待时间，提高系统吞吐量
- 缺点
  - 对长作业不利，容易产生“饥饿”现象
  - 未考虑作业的紧迫程度
- 困难
  - 难以确定进程的执行时间

2.3.2.3 高响应比优先调度算法(1)

- 优点
  - 兼顾了运行时间短和等待时间长的作业（结合FCFS和SJF方法），优先运行短作业和等待时间足够长的长作业
- 缺点
  - 较复杂，系统开销大

$$\text{响应比} = \frac{\text{作业等待时间} + \text{作业估计运行时间}}{\text{作业估计运行时间}} = 1 + \frac{\text{作业等待时间}}{\text{作业估计运行时间}}$$

2.3.2.3 高响应比优先调度算法(2)

- 响应比计算
  - 若作业等待时间相同，则作业估计运行时间越短，响应比越高，因此有利于短作业
  - 若作业估计运行时间相同，则作业等待时间越长，响应比越高，此时算法即FCFS
  - 对长作业来说，其响应比将随等待时间的增加而提高，当等待时间足够长时，响应比将足够高，从而可以获得处理机，避免产生“饥饿”现象

2.3.2.4 优先级调度算法(1)

- 为每个进程设定一个优先级
- 既可采用非剥夺调度方式(批处理系统)
- 也可采用剥夺调度方式（实时系统）
- 优先级设定
  - 进程类型：系统进程/用户进程
  - 进程对资源的需求
    - 申请资源较多的进程，优先级较低
  - 用户要求
- 优先级与优先数

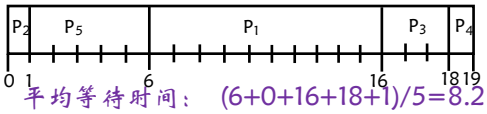
2.3.2.4 优先级调度算法(2)

- 优先级类型
  - 静态优先级
    - 进程创建时确定，整个运行期间保持不变
    - 不能反映进程特点，调度性能差
    - 容易导致“饥饿”，即不能调度低优先级进程
  - 动态优先级
    - 随着进程的推进或等待时间的增加而改变

2.3.2.4 优先级调度算法(3)

- 示例：5个进程的运行时间及优先级如下

进程	运行时间	优先数
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

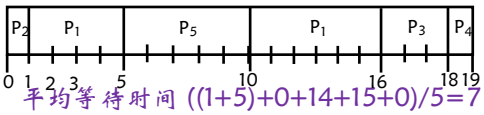


平均等待时间：(6+0+16+18+1)/5=8.2

2.3.2.4 优先级调度算法(4)

- 示例：5个进程的运行时间及优先级如下

进程	到达时间	运行时间	优先数
P1	0	10	3
P2	0	1	1
P3	2	2	4
P4	3	1	5
P5	5	5	2



平均等待时间 ((1+5)+0+14+15+0)/5=7

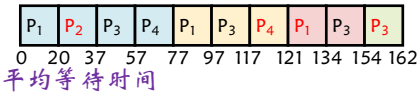
2.3.2.5 时间片轮转调度算法(1)

- 用于分时系统
- 采用剥夺调度方式
- 时间片的确定
  - 既要保证系统各个用户进程及时地得到响应，又不要由于时间片太短而增加调度的开销，降低系统的效率

2.3.2.5 时间片轮转调度算法(2)

- 示例：时间片为20，4个进程运行时间为

进程	运行时间
P1	53
P2	17
P3	68
P4	24

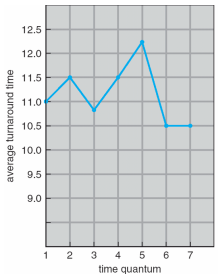


平均等待时间

((57+24)+20+(37+40+17)+(57+40))/4=73

((134-53-0)+(37-17-0)+(162-68-0)+(121-24-0))/4=73

2.3.2.5 时间片轮转调度算法(3)



process	time
P_1	6
P_2	3
P_3	1
P_4	7

周转时间随时间片大小变换

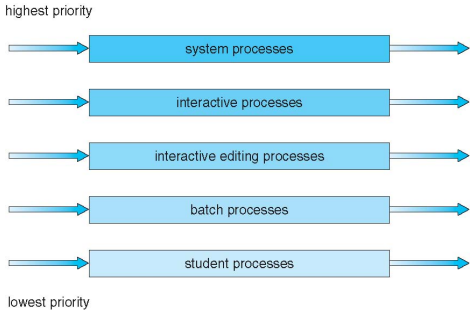
2.3.2.6 多级队列调度算法(1)

- 综合考虑各种类型进程的需要
  - 终端型作业用户
    - 交互性好，响应时间短
  - 短批处理作业用户
    - 周转时间短，在较短时间内完成
  - 长批处理作业用户
    - 不会出现长期得不到处理的“饥饿”现象

2.3.2.6 多级队列调度算法(2)

- 将就绪队列分成多个独立队列
- 根据进程的属性（内存大小、进程优先级、进程类型），一个进程被永久地分配到一个队列
- 每个队列有自己的调度算法
- 队列之间进行调度
  - 固定优先级抢占调度

2.3.2.6 多级队列调度算法(3)



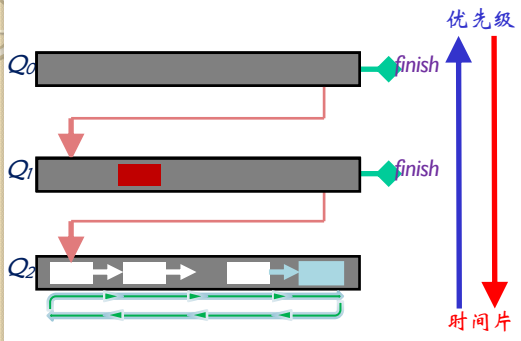
124

2.3.2.7 多级反馈队列调度算法(1)

- 设置多个就绪队列，并为其赋予不同的优先级和时间片，高优先级队列中进程的时间片较小
- 新进程放入第一队列尾，按FCFS算法调度
- 若该进程在指定时间片内未完成，调度程序将该进程转入第二队列尾，按FCFS算法调度，依次类推
- 仅当第一队列空时，调度程序才调度第二队列中的进程运行，依次类推
- 当处理机为某级队列服务时，有高优先级进程进入，则采用剥夺方式为高优先级进程服务

125

2.3.2.7 多级反馈队列调度算法(2)



126

2.3.2.7 多级反馈队列调度算法(3)

- 调度时考虑的问题
  - 队列数目
  - 每个队列采用的调度算法
  - 初始状态时，每个进程处于哪个队列

127