进程调度

- 2.2.1 处理机调度的概念
  - 概念 —— 按照某种算法选择一个进程将处理机分配给它
  - 调度的三个层次
    - 高级调度（作业调度） —— 将外存的作业放入内存中 —— 发生频率最低
    - 中级调度（内存调度） —— 按照某种策略决定将哪个处于挂起状态的进程重新调入内存 —— 中等 —— 就绪挂起->就绪
    - 低级调度（进程处理机调度） —— 按照某种策略从就绪队列中选取一个进程，将处理机分配给他 —— 最低
  - 挂起 —— 阻塞->阻塞挂起
  - 三级调度的联系，对比
    - 高级调度
      - 外存-->内存（面向作业）
      - 发生频率：最低
    - 中级调度
      - 外存-->内存（面向进程）
      - 发生频率：中等
    - 低级调度
      - 内存-->CPU
      - 发生频率：最高
  - 补充
    - 进程的挂起态 —— 为减轻系统负载，提高系统资源利用率，暂时不执行的进程会被调到外存从而发为挂起态
    - 七状态模型
- 2.2.2 调度的时机，切换与过程
- 2.2.3 进程调度的方式
  - 剥夺调度方式（抢占式） —— 只能由当前运行的进程主动放弃CPU
  - 非剥夺调度方式（非抢占式） —— 可由操作系统剥夺当前进程的CPU使用权
- 2.2.4 调度算法的评价指标
  - CPU利用率 —— 也有题目会让我们计算某设备的利用率
  - 系统吞吐量 —— 系统吞吐量=（总共完成了多少道作业）/总共花了多长时间
  - 周转时间
    - 周转时间 —— 作业提交到作业完成所经历的时间
      - 作业在外存后备队列上的等待作业调度的时间
      - 进程在就绪队列上等待进程调度的时间
      - 进程在CPU上执行的时间
      - 进程在阻塞队列上的等待时间
    - 平均周转时间 —— 各作业周转时间之和/作业数
    - 带权周转时间 —— 作业周转时间/作业实际运行的时间
    - 平均带权周转时间 —— 各作业的带权周转时间/作业数
    - 注意
      - 这里被服务的时间不仅仅是用CPU的时间，还包括IO操作的时间
      - 因此 等待时间就是周转时间-运行时间-IO操作时间
  - 等待时间
    - 进程/作业处于等待处理机状态的时间之和
      - 等待时间越长，用户满意度越低
      - 衡量一个调度算法的优劣，常常只需要简单地考察等待时间
    - 平均等待时间即各个进程/作业 等待时间的平均值
  - 响应时间 —— 从用户提交到首次产生响应所用的时间 —— 在交互系统中，一般采用响应时间作为衡量调度算法的重要准则之一
- 2.2.5 典型调度算法
  - （1）FCFS 先来先服务
    - 算法思想 —— 公平
    - 算法规则 —— 按照作业/进程到达队列的先后顺序进行服务
    - 用于作业/进程调度
    - 不可剥夺算法
    - 优点
      - 算法简单
      - 公平
    - 缺点
      - 效率低
      - 带权周转时间很大
      - 对长作业，CPU繁忙型作业有利
      - 对短作业，I/O繁忙型作业不利 —— 每次IO结束都需重新排队
    - 是否会导致饥饿 —— 不会
    - 其他
      - 不能作为分时系统和实时系统的主要调度策略
      - 常常结合在其他调度策略中使用，例如：在使用优先级作为调度策略的系统中，往往对多个具有相同优先级的进程按"先来先服务"原则处理
  - （2）SJF 短作业（进程）优先
    - 算法思想 —— 追求最少的平均等待时间、最少的平均周转时间、最少的平均带权周转时间
    - 算法规则 —— 最短的作业/进程优先/得到服务（所谓最短，是指要求服务时间最短）
    - 用于作业/进程调度
      - 短作业 —— 从后备队列选择一个或若干给计运行时间最短的作业
      - 短进程 —— 从就绪队列中选择一个给计运行时间最短的进程
      - 或者执行时间由用户提供
    - 不可剥夺算法 —— 也有可剥夺的版本-最短剩余时间优先算法 —— 每当有新进程加入就绪队列时就需要调度，如果新进的进程剩余的时间比当前运行的进程剩余时间要短，则当新进程到达时发生调度，当前运行进程重新回到就绪队列。另外，当一个进程完成时也需要调度
    - 优点 —— 一"最短的"平均等待时间、平均周转时间
    - 缺点
      - 不公平，对短作业有利，多长作业不利
      - 可能产生饥饿现象
      - 未考虑作业的紧迫程度
    - 是否会导致饥饿 —— 会
    - 其他 —— 长作业长期不被调度，出现饥饿现象 —— 注意区分死锁
      - 前者是调度策略问题
      - 后者是系统环形等待
    - 附加题：2.2.5 调度算法
  - （3）HRRN 高响应比优先
    - 算法思想 —— 综合考虑作业/进程的等待时间和要求服务的时间
    - 算法规则 —— 每次调度时先计算各个作业/进程的响应比，选择响应时间高的作业/进程为其服务 —— 响应比=（等待时间+要求服务时间）/要求服务时间
    - 用于作业/进程调度
    - 非抢占式算法
    - 优点 —— 综合考虑了等待时间和运行时间，避免了长作业饥饿问题
    - 是否会导致饥饿 —— 会
  - 以上三种小结
    - 对于算法（1）（2）（3），如果是非抢占式的，进程/就绪一般发生在当前进程执行结束或者当前进程进行IO操作
    - 这几种算法主要关心对用户的公平性、平均周转时间、平均等待时间等评价系统整体性能的指标，不关心响应时间，也不区分任务的紧急程度，因此交互性很差，只适用于早期的批处理系统
  - （4）优先级调度算法
    - 算法思想 —— 根据任务的紧急程度来决定处理顺序
    - 算法规则 —— 每个作业/进程都有各自的优先级，调度时选择优先级最高的进程/作业
    - 用于作业/进程调度 —— I/O调度也可
    - 非抢占式（进程执行结束再调度） —— 也有抢占式（就绪队列发生变化就调度）
    - 优点 —— 用优先级区分紧急程度、重要程度，适用于实时操作系统
    - 缺点 —— 可能会导致饥饿
    - 是否会导致饥饿 —— 会
    - 其他
      - 同级服从FCFS
      - 就绪队列组织方式
        - 不同优先级对应不同队列
        - 优先级由高到低动态排列
          - 数组存储
          - 链表存储
          - 高度平衡二叉树
          - 大顶堆
      - 优先级分类
        - 静态优先级
        - 动态优先级
  - （5）时间片轮转调度算法
    - 算法思想 —— 公平地、轮流地为各个进程服务
    - 算法规则 —— 按照进程到达就绪队列的顺序，轮流让各个进程执行一个时间片，执行完后将进程重新放到就绪队列队尾重新排队 —— 规定若当前进程A执行完一个时间片后，并且又来了新进程B，应该先让B先进入就绪队列。
    - 用于进程调度
    - 抢占式算法
    - 优点 —— 公平响应快，适用于分时操作系统
    - 缺点 —— 由于频繁的进程切换，降低了CPU的利用率，不区分任务的紧急程度
    - 是否会导致饥饿 —— 不会
    - 其他
      - 当q=1时间片用完，p2刚好到就绪队列时，一般认为q排在p2后面
      - 时间片过大退化为FCFS
      - 时间片过小导致实际用于进程执行的时间比例减少
  - （6）多级反馈队列调度算法
    - 算法思想 —— 时间片轮转调度算法和优先级调度算法的综合与发展
    - 算法规则 —— 略
    - 用于进程调度
    - 抢占式算法 —— 略
    - 优点
      - 提高系统吞吐量且缩短平均周转时间，照顾短进程
      - 不必是事先给计进程的执行时间
    - 是否会导致饥饿 —— 会
    - 其他
      - 当q=1时间片用完，p2刚好到就绪队列时，一般认为q排在p2后面
      - 时间片过大退化为FCFS
      - 时间片过小导致实际用于进程执行的时间比例减少
  - 4、5、6小结 —— 这些算法更注重系统的响应时间、公平性、平衡性等指标，能较好地满足交互式系统的需求，因此适合交互式系统

XMind：ZEN | 试用模式