第三章 处理机调度与死锁

3.1

3.1.0 作业和进程

- 作业：用户向计算机提交任务的**任务实体**
- 进程：完成用户任务的**执行实体**

> 没有作业，进程无事可干；没有进程，作业任务无法完成。

- 作业建立完毕，总在外存等待；进程总有一部分在内存
- 作业调度：使作业进入主存储器
- 进程调度：是作业进程占用处理器

3.1.1 处理机的层次

1. 高级调度（作业调度）

- 调度对象：作业
- 功能：
  - 接纳多少作业？ 接纳那些作业?
- 场景：
  - 批处理 yes
  - 分时系统 no
  - 实时系统 no

2. 低级调度（进程调度）

- 调度对象：进程 / 内核级线程
- 特点：频率很高
- 功能：
  - 保存处理机的现场信息
  - 按某种算法选取进程
  - 把处理机分配给进程
- 场景：三种OS里都有
- 3个基本机制
  - 排队器
  - 分派器
  - 上下文切换器
- 2种方式
  - 非剥夺调度方式：实现简单，开销小；不适于紧急任务。
    - **引起进程调度**：
    - 进程执行完毕，或因发生某事件而不能继续执行
    - 执行中的进程因提出I/O请求而暂停执行
    - 执行了某种原语操作：如P操作、Block原语

- 剥夺调度方式：适用于紧急任务；分时/实时系统
  - 优先权原则
  - 时间片原则
  - 短作业优先原则

3. 中级调度（内存调度）

- 调度对象：挂起状态进程
- 功能
  - 等待队列过长，放入外存挂起；再放回内存

补充：调度队列模型

- 1.仅有进程调度的调度队列模型
- 2.具有高级和低级调度的调度队列模型
- 3.同时具有三级调度的调度队列模型

仅有进程调度

- 原理
  - 当创建一个新进程时，将它放入队列末尾。
  - 按时间片轮转方式运行。
- 就绪队列形式：FIFO

具有高级和低级调度

- 原理
  - 从外存的后备队列中选择一批 作业调入内存，并为它们建立进程，送入就绪队列。
  - 然后才由进程调度算法选择一个进程，把处理机分配给该进程
- 就绪队列形式：优先权队列形式
- 设置多个阻塞队列

同时具有三级调度

- 当在OS中引入中级调度后，
  - 可以把进程的就绪状态分为内存就绪和外存就绪，
  - 把阻塞状态分为内存阻塞和外存阻塞两种状 态。
- 在调出操作的作用下，可使进程状态由内存就绪转变为外存就绪，由内存阻塞转变为外存阻塞；
- 在中级调度的作用下，又可使外存就绪转变为内存就绪。

3.1.2 调度算法评价和指标

- CPU利用率 = CPU有效工作时间 / （CPU有效工作时间 + CPU空闲等待时间）
- 系统吞吐量 = 总共完成多少作业 / 总的运行时间
- 周转时间 = 作业完成时间 - 作业提交时间
- 带权周转时间 = （作业完成时间 - 作业提交时间）/ 作业实际运行时间
- 平均带权周转时间 = 各作业带权周转时间之和 / 作业数
- 响应时间

3.2 调度算法

作业（进程）调度

- 批处理系统：短作业优先
- 分时系统：时间片流转

1. 先来先服务 FCFS

- 适用：作业/进程调度
- 作业：先到先得
- 进程：先到先得
- 利弊
  - 利于长作业，不利于短作业

2. 短作业（进程）优先调度算法 （SJ(P)F）

- 适用：作业/进程调度
- 作业：运行时间端的作业,优先调入内存运行。
- 进程：运行事件端的进程，先分配处理机
- 利弊
  - 有效降低作业的平均等待时间，提高系统吞吐量
  - 必须预知作业的运行时间
  - 对长作业不利
  - 无法实现人机交互
  - 未考虑紧迫性

3. 高响应比优先调度算法 （HRRN，Highest Response Ratio Next）

- 适用：作业/进程调度
- 优先级 = （等待时间+要求服务时间）/要求服务时间
- 响应比 = （等待时间+要求服务时间）/要求服务时间 = 响应时间/要求服务时间 = 优先级

4. 时间片轮转法

- 适用：**进程调度**
- 正常FCFS，时间片内未运行完，转回就绪态插入就绪队列未尾，CPU交给下一个进程
- 时间片大小的确定
  - 时间片略大于一次典型的交互所需要的时间
- 利弊
  - 抢占
  - 无缓急处理

5. 优先级调度算法

- 非抢占式：
  - 适用：批处理系统
  - 优点：满足紧迫作业
- 抢占式：
  - 选择优先级最高的，可打断。先来先运行

优先级的类型：

- 静态优先级
  - 确定优先级的依据
    - 进程类型：系统进程 > 用户进程
    - 进程对资源的需求
    - 用户要求
- 动态优先级

6. 多级反馈队列调度算法（调度机制要背！！！ 其他可以不背）

- 多个就绪队列，第1个就绪队列在时间片没有运行完，则放入就绪队列2，以此类推...一直到最后一个就绪队列。
- 能较好满足各种类型进程的需要。

3.4 **实时调度

3.4.1 实时调度的基本条件

1. 提供必要的信息

- 就绪时间
- 开始截止时间和完成截止时间： Only one
- 处理事件
- 资源要求
- 优先级

2. 系统处理能力强

3.5 死锁

> 死锁至少需要两个进程。
> 错误：所有情况都可能出现死锁。

- 死锁：卡死
- 饥饿：长期等不到

3.5.1 资源问题

- 可重用资源
- 可消耗资源/临时性资源
- 可抢占资源
  - 不会引起死锁
  - CPU、主存
- 不可抢占资源
  - 打印机、磁带机

3.5.2 产生死锁的原因

- 竞争资源
  - 竞争不可抢占性资源

    - 竞争可消耗资源
- 进程推进顺序不当
    - 进程推进顺序合法
    - 进程推进顺序不合法

3.5.3 定义、必要条件和处理方法

1. 定义

- 如果一组进程中的每一个进程都在等待仅由改组进程中的其他进程才能引发的事件。
- 至少有两个及两个以上的进程

2. 产生死锁的必要条件（非常重要！！！）

- **互斥条件**：一个在用，另一个只能等
- **请求和保持条件**：拿了还要
- **不可抢占条件**：没用完，不还
- **循环等待条件**：持续等待

3. 处理死锁的基本方法

- **预防死锁**：较严格，破坏2，3，4条件
- **避免死锁**（**银行家算法**）：较宽松，从源头防进入
- **死锁的检测**
- **死锁接触**

3.6 预防死锁

- 一般破坏2，3，4条件，互斥条件需要存在

3.6.1 破坏“请求和保持”条件

- 原：源资源不释放，同时请求新资源(得不到的大家都得不到)

1. 第一种协议

- 运行前一次性申请完所有资源
- 优：**简单、安全**
- 缺：**资源浪费严重，进程易延迟**

2. 第二种协议

- 只运行最初需要的资源，运行过程中释放用完的全部资源，才能再申请。
- 优：**提高设备利用率，减少进程发生饥饿的概率**

3.6.2 破坏“不可抢占”条件

- 新资源拿不到，立即释放所有资源（得不到，让大家得到）
- 缺：
    - 复杂

- 仅对易保存资源使用
- 增加系统开销，降低系统吞吐量
- 导致饥饿

3.6.3 破坏“循环等待”条件

- 顺序资源分配法：必须按编号递增的顺序请求资源。（从小到大申请）
- 优
  - **资源利用率**和**系统吞吐量**都有较明显的改善
- 缺
  - 为资源编号，**限制新设备的增加**；
  - 进程使用**设备顺序与申请顺序不同**，**浪费资源**；
  - **限制用户编程自由。**

3.7 避免死锁

3.7.1 系统安全状态

- 安全状态：一定不是死锁
- 不安全状态：可能进入，但不是一定是死锁

3.7.2 银行家算法 p112

1. 数据结构

- **可利用资源向量 Available**
  - 系统有多少
- **最大需求矩阵 Max**
  - 进程最多要多少
- **分配矩阵 Allocation**
  - 进程已经有多少
- **需求矩阵 Need**
  - 进程还要多少
- Need = Max - Allocation

2. 银行家算法

- 请求资源 <= 目前需求 **Need**？
  - No, 需求的资源大于它所宣布的最大值，出错
- 请求资源 <= 当前可用最大值 **Available**？
  - No, 无足够资源，需等待
- 尝试分配资源
- 判断安全性

3. 安全性算法

- **设置两个向量**
  - Work变量：系统可以给进程的资源；最初Work = Available
  - Finish变量

- **找满足以下条件的进程**
  - Finish = false
  - Need <= Work'
- **找到后**
  - Work = Work + Allocation
  - Finish = TRUE
  - 继续找
- **没找到**
  - 安全：Finish 都等于TRUE
  - 不安全

3.8 死锁检测与接触

死锁检测

- 资源分配图
- **死锁定理**（容易考！！）
  - 是充分条件
  - 死锁 = 资源分配图 不可完全简化

3.8.2 死锁的解除

- 抢占资源
- 终止/撤销进程
  - 终止所有进程
  - 逐个终止进程
  - 进程回退法