

处理器基本概念

处理器管理负责**管理、调度和分配计算机系统的处理器资源**，并控制程序的执行。

无论在应用程序，还是系统程序，最终都要在处理器上执行以实现其功能。处理器管理的优劣将直接影响到计算机系统的性能。

程序状态字寄存器

程序状态字PSW：指示处理器状态、控制指令的执行顺序、保留和指示与运行程序相关的各种信息。主要作用是方便实现程序状态的保护与恢复。每一个程序都有一个与其执行相关的PSW，每个处理器都设置一个程序状态寄存器。一个程序占有处理器执行时，它的PSW将占有程序状态字寄存器。

PSW包括：程序计数器、指令寄存器、条形码、中断位、中断允许位、中断屏蔽位、处理器模式位、主存保护位等，记录当前程序的动态信息。

操作系统的运行机制

特权指令与非特权指令

- 特权指令：可能涉及到改变处理器状态等的敏感操作。如内存清零。权限高，不允许用户程序使用。
- 非特权指令：如普通的运算指令，加减乘除，权限低。

用户态与内核态

- 用户态（目态）：只能执行非特权指令
- 内核态（管态）：能执行特权指令和非特权指令

两种处理器状态存储在在PSW中的处理器模式位。

内核程序与应用程序

- 操作系统内核程序：特权高，可使用全部机器指令
- 运行于OS环境的普通应用程序：权限低，只能使用资源管理命令之外的指令

宏内核与微内核

- 宏内核：资源管理+支撑功能。*LINUX, UNIX*
- 微内核：支撑功能。*Windows NT*

操作系统内核需要运行在内核态，操作系统的非内核功能运行在用户态。

处理器状态转换

- 内核态->用户态

执行一条特权指令——修改PSW的标志位为“用户态”。操作系统将主动让出CPU使用权。

- 用户态->内核态

由“中断”引发，硬件自动完成变态过程，触发中断信号意味着操作系统将强行夺回CPU的使用权。

只有中断才能使处理器从用户态向内核态转换。例如：程序主动同步请求OS服务，执行系统调用

原语及其原子性

执行关中断指令，不再检查中断信号，直到执行开中断指令后才恢复检查。

中断

处理器对系统中或系统外发生的异步事件的响应。

中断是多道程序得以实现以及设备管理的基础。

中断处理程序是I/O系统最底层，是整个I/O系统的基础。

只要发生中断，就意味着OS介入，参加管理工作。

中断的类型

- 内中断（陷入、trap）：由CPU内部事件所引起的中断。
 - 硬件故障中断事件：电源故障、主存故障：保护现场、停止工作、估计恢复
 - 程序性中断事件：算数异常、指令异常、终止进程指令、虚拟地址异常：报告用户、终止进程、重新执行指令。
 - 自愿性中断：系统调用：陷入指令->执行系统调用服务例程。访管中断。
- 外中断：CPU对I/O设备发来的中断信号的一种响应。
 - I/O中断事件
 - 外部中断事件：时钟、间隔时钟、设备、键鼠信号、关机重启等

上面五种中断事件，除了自愿性中断，全都是强迫性中断。

系统调用

程序接口由一组系统调用组成。用户通过系统调用请求内核为其服务。

发出系统调用请求的在用户态，对系统调用的相应处理在内核态。

陷入指令（trap指令/访管指令）是唯一一个只能在用户态下执行，不在内核态执行的指令。

凡是与共享资源有关的操作，会直接影响到其他进程的操作，就一定需要操作系统介入，就需要通过系统调用来实现。

进程

进程的概念

- 程序是指令的有序集合，其本身没有任何运行的含义，是一个静态的概念。由程序段和数据两部分组成。

进程是程序在处理机上的一次执行过程，是一个动态的概念。由程序段、数据和PCB组成。
- 同一程序可以对应多个进程，进程更能真实地描述并发，而程序不能
- 进程具有创建其他进程的功能，而程序没有
- 进程是动态的，而进程实体（**进程映像**）是静态的。进程实体反映了进程在某一时刻的状态。

- 进程与PCB——对应，PCB是系统感知进程存在的唯一标志。

PCB

PCB的作用

- 作为独立运行基本单位的标志
- 能实现间断性运行方式
- 提供进程管理所需要的信息
- 提供进程调度所需要的信息
- 实现与其他进程的同步和通信

PCB中的信息

- 进程描述信息
 - 进程标识符，唯一，通常是一个整数
 - 进程名，通常基于可执行文件名
 - 用户标识符
- 进程控制和管理信息
 - 当前状态
 - 优先级
 - 代码执行入口地址
 - 程序的外存地址
 - 运行统计信息（执行时间、页面调度）
 - 进程间同步和通信；阻塞原因
 - 进程的队列指针
 - 进程的消息队列指针
- 资源分配清单（资源使用情况）
 - 虚拟地址空间的现状
 - 打开文件列表
- 处理器相关信息（CPU现场保护信息）
 - 寄存器值（通用、程序计数器PC、状态PSW、地址包括栈指针）
 - 指向赋予该进程的段/页表的指针

PCB的组织形式

系统将所有PCB组织在一起，并把它们放在内存的固定区域，构成了PCB表。

PCB表的大小决定了系统中最多可同时存在的进程个数，称为系统的**并发度**。

- 链接方式

相同状态的进程PCB组成一个链表，不同状态对应多个不同的链表。
- 索引方式

相同状态的进程，分别设置各自的PCB索引表，表明PCB在PCB表中的地址。

进程的特征

- 动态性：进程是程序的执行
- 并发性：多个进程可同存于内存中，能在一段时间内同时运行
- 独立性：独立运行的基本单位，独立获得资源和调度的基本单位
- 异步性：各进程按各自独立的不可预知的速度向前推进
- 结构性：每个进程都配置一个PCB，进程由程序段、数据段和PCB组成

进程的状态

创建

进程正在被创建，操作系统为进程分配资源、初始化PCB

- 一般步骤：
 - 申请一个空白PCB，在PCB中填写用于控制和管理进程的信息
 - 分配资源
 - 设置为就绪状态并插入就绪队列中
- 目的：
 - 保证进程的调度在创建通过完成后进行，以确保对进程控制的完整性
 - 增加管理的灵活性
- 允许一个进程创建另一个进程。
 - 进程图：描述进程间关系的一棵有向树。
 - 引起创建进程的四类典型事件：
 - 用户登录
 - 作业调度
 - 提供服务
 - 应用请求

运行

正在运行的进程所处的状态，拥有CPU

就绪

存在于处理机调度队列中的那些进程，它们已经准备就绪，一旦得到CPU，就立即可以运行。

阻塞

若一进程正在等待某一事件发生（如等待I/O），这时，即使给它CPU，它也无法运行（阻塞/等待/睡眠/封锁状态）。

当一个进程所期待的某一时间尚未出现时，该进程调用阻塞原语**将自己阻塞**。进程阻塞是进程自身的一种主动行为。

终止

进程正在从系统中撤销，操作系统会回收进程拥有的资源、撤销PCB

- 一般步骤
 - 等待操作系统进行善后处理
 - 将PCB清零，将PCB空间返还系统
- 终止原因
 - 正常结束
 - 异常结束：越界错、保护错、非法指令、特权指令错、运行超时、等待超时、算术运算错、I/O故障
 - 外界干预：操作员或操作系统干预、父进程请求、因父进程终止
- 终止过程
 - 根据被终止进程的标识符，从PCB集合中检索出该进程的PCB
 - 若被终止进程正处于执行状态，应立即终止该进程的执行，并置调度标志为真，用于指示该进程被终止后应重新进行调度
 - 若该进程还有子孙进程，还应将其所有子孙也都予以终止，以防它们成为不可控的进程
 - 将进程所拥有的资源交给父进程或系统进程
 - 释放PCB

进程的操作

挂起

将内存中的一部分进程转移到磁盘中。

当内存中所有进程阻塞时，操作系统可将一进程置为挂起状态并交换到磁盘，再调入另一进程执行。

- 活动就绪（Readya）：未被挂起的就绪状态。此时的进程可以接受调度。处于Readys状态的进程用激活原语Active激活后，进程转变为Readya。
- 静止就绪（Readys）：用suspend原语将活动就绪挂起后，变为静止就绪。
- 活动阻塞（Blockeda）：未被挂起的阻塞状态。处于Blockeds状态的进程用激活原语激活后，进程变为Blockeda。
- 静止阻塞（Blockeds）：用suspend原语将活动阻塞挂起后，变为静止阻塞。

挂起的原因

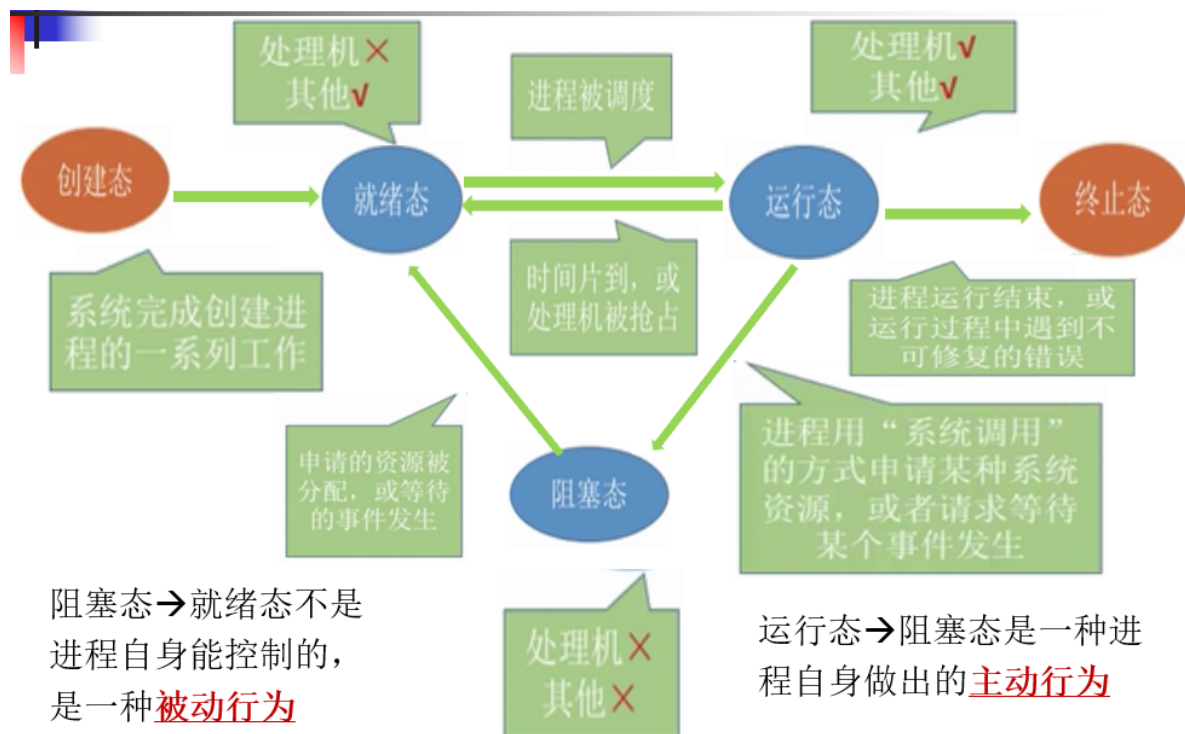
原因	说明
交换	OS 需要释放足够的内存空间以调入就绪进程执行
其他 OS 原因	OS 可能会挂起一个后台进程或怀疑引起问题的进程
交互要求	用户可能为了调试或与资源连接而要求挂起进程
中断	一个进程可能是周期性地执行的，那么它在等待下一次执行时如被挂起
父进程请求	父进程有时希望挂起某个后代进程以检查或修正挂起进程，或协调多个后代进程的运行

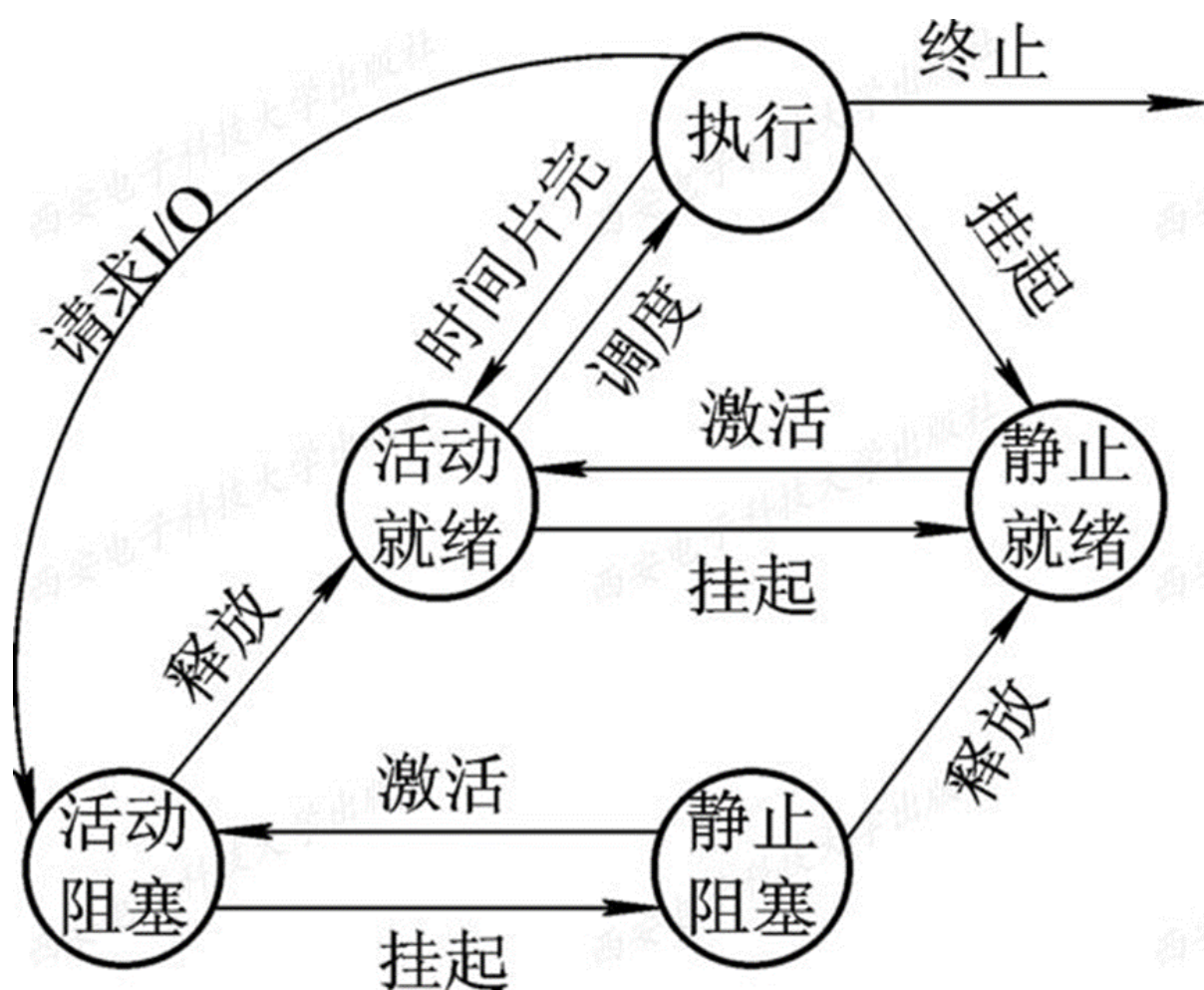
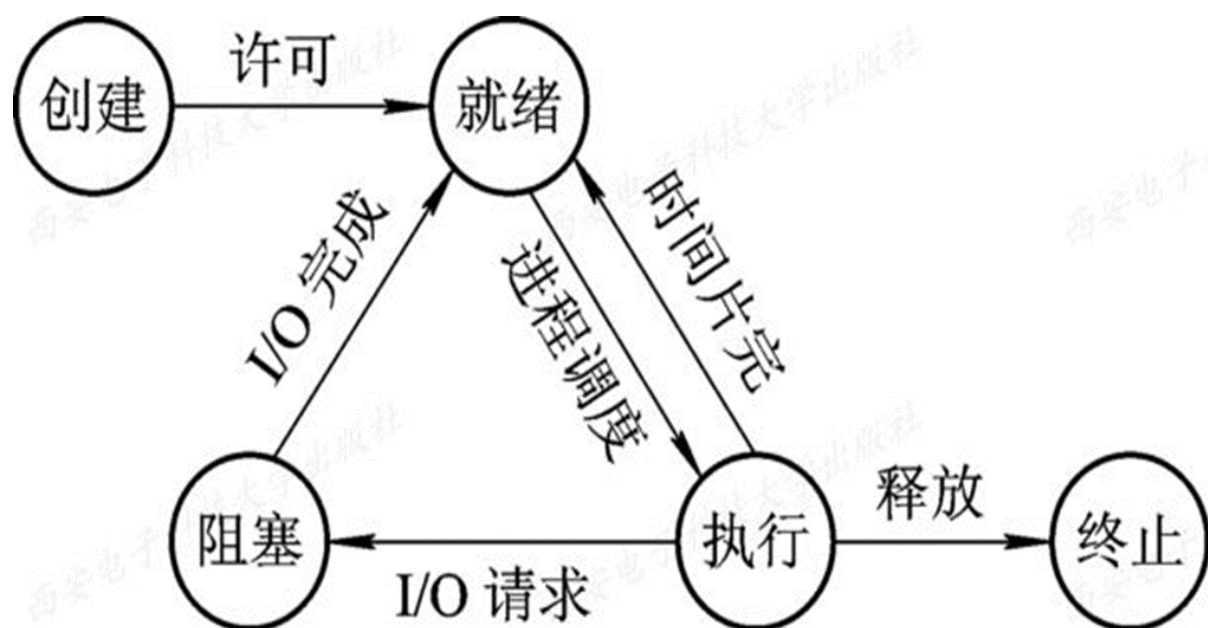
激活

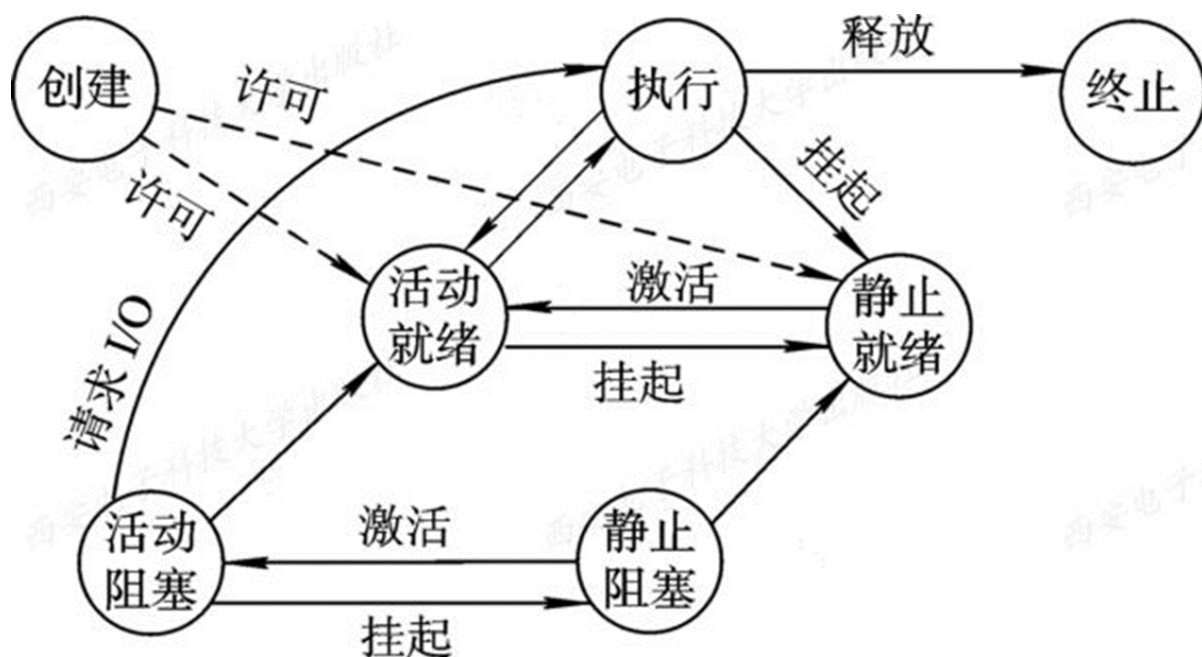
active;静止就绪->活动就绪；静止阻塞->活动阻塞。

唤醒

阻塞的进程，由它的合作进程用唤醒原语唤醒它。







注：上图应该是执行->活动阻塞，原图箭头不对。

进程的通信

指进程之间的信息交换。

- 共享存储
 - 互斥访问，操作系统只负责提供共享空间和同步互斥工具（P/V操作）
 - 基于数据结构的共享：共享空间里放一个长度为10的数组。速度慢、限制多、是一种低级通信方式。
 - 基于存储区的共享：在内存中画出一块共享存储区，数据的形式、存放位置都由进程控制，而不是操作系统。速度更快，是高级通信方式。**通信效率最高。**
- 信息传递
 - 以格式化的消息为单位，通过操作系统提供的“发送消息/接收消息”两个原语进行数据交换。
 - 直接通信方式：消息直接挂到接收进程的消息缓冲队列上
 - 间接通信方式：消息先发送到中间实体（**信箱**）中
- 管道通信
 - 管道是用于连接读写进程的一个共享文件，又名pipe文件。是在内存中开辟一个大小固定的缓冲区。
 - 管道只能采用半双工通信，某一个时间段内只能实现单向的传输。如果要实现双向同时通信，则需要设置两个管道。
 - 各进程要互斥地访问管道。
 - 数据以字符流的方式写入管道，当管道写满时，写进程将被阻塞，等待读进程将数据取走。当读进程将数据全部取走后，管道变空，读进程被阻塞。
 - 如果没写满，就不允许读；如果没读空，就不允许写。
 - 数据一旦被读出，就从管道中抛弃，这意味着读进程最多只能有一个，否则可能会有读错数据的情况。
- 客户机/服务器系统

进程的同步与互斥

并发程序的设计特点：间断性、失去封闭性、不可再现性。

Bernstein条件

如果两个程序 P_1 和 P_2 能够满足：

$$(R(P_1) \cap W(P_2)) \cup (R(P_2) \cap W(P_1)) \cup (W(P_1) \cap W(P_2)) = \emptyset$$

则它们可以并发执行，且具有可再现性。

临界区管理

需要互斥使用的资源称为临界资源，访问临界资源的代码称为临界区。

在进入区判断是否可以进入临界区，并设置临界区使用标志；在退出区清除临界区使用标志，并负责唤醒阻塞队列中的一个进程。

进程互斥遵循的原则

- 空闲让进
- 忙则等待
- 有限等待
- 让权等待：当进程不能进入临界区时立即释放处理机

信号量机制

记录型信号量：

- $s.value > 0$, 表示有 $value$ 个资源可用
- $s.value = 0$, 表示无资源可用
- $s.value < 0$, 则 $|s.value|$ 表示等待队列中的进程个数

P/V操作

每次P操作（wait操作）

- 意味着进程请求一个单位资源
- 信号量的值-1
- 若使信号量的值变为负数，则执行P操作的进程被阻塞

每次V操作（signal操作）

- 意味着进程释放一个单位资源
- 信号量的值+1
- 若信号量的值不是正数，表示依然有进程在等待该类资源，解除等待队列队首的阻塞进程。

缺少P就不能保证临界资源的互斥访问；缺少V会导致资源永不被释放，等待进程永不被唤醒。

对任何信号量的PV操作必须配对，同一进程中的多对PV操作只能嵌套，不能交叉。

进程同步

设置同步信号量S，初始值为0。这个S代表“某种资源”。刚开始没有这种资源，P2需要这种资源，又只能由P1产生这种资源。

在前操作之后执行V，在后操作之前执行P。

经典进程同步与互斥问题

生产者-消费者问题

多生产者-多消费者问题

哲学家就餐问题

读者写者问题

读者优先：除非有写者在写文件，否则读者不需要等待。可能导致饥饿。

写者优先：只要有一个写者申请写数据，就不再允许新的读者进入读数据。这样，写者只需等优先于它到来的读者完成其读数据任务，而不用等待其后到来的读者。解决了饥饿问题，但降低了并发程度，使系统的性能较差。

死锁

多个进程因竞争共享资源而造成的一种僵局。

产生死锁的必要条件

- 互斥条件：资源独占，竞争的资源一次只能被一个进程使用
- 请求和保持（占有且等待）：当一个进程占有有一些资源，同时又申请新的资源，如果新资源申请失败，进程将占有资源且等待。
- 不剥夺条件：进程已占有的资源不能被其他进程剥夺
- 环路等待条件：必有一进程-资源的环形链。环路中的进程形成等待链。

前三个为必要不充分条件，最后为充分条件。合起来就是充分必要条件。

死锁4种处理方法：忽略、检测与恢复、避免和预防，对于死锁的处理从宽到严，同时系统并发性由大到小。

处理死锁的基本方法

死锁预防

破坏死锁出现的条件。

- 实现简单。
- 严格限制了系统资源的分配和使用，会降低系统资源利用率和吞吐量。

对应上面三个必要条件：

- 破坏请求和保持（预先静态分配法）
 - 在所有进程开始之前，一次性申请其在运行过程中所有资源
 - 需要预知未来，编程困难

- 许多资源分配很长时间才使用，资源利用率低。可能饥饿。
 - 先活动运行初期所需的资源，再逐步释放已分配且用毕的全部资源，再请求新的资源
- 破坏不可抢占
 - 当一个已保持了某些不可被抢占资源的进程，提出新的资源请求而得不到满足时，它必须释放掉已经保持的所有资源。
 - 实现复杂
 - 增加系统开销，降低吞吐量
- 破坏循环等待
 - 对系统所有资源类型进行线性排序，并赋予不同的序号。若进程已经占有序号较高的资源，又想请求序号低的资源，需要先释放所有具有相同或者更高序号的资源后，才能申请序号低的资源。
 - 系统中各类资源序号须相对稳定，限制新设备增加。
 - 资源浪费
 - 限制用户简单自主编程

死锁避免

对每个资源请求进行判断，如果造成死锁就拒绝。

- 资源利用率和吞吐量较高
- 实现困难

安全状态：系统按照某种进程顺序来为每个进程分配其所需的资源，直至最大需求，使每个进程都能顺利完成，则为安全状态。

银行家算法

资源分配算法

- Available可用资源向量： m ， m 类资源，每个元素代表这类资源可用资源数目
- Max最大需求矩阵： (n,m) ， n 个进程对于 m 类资源的最大需求
- Allocation已分配矩阵： (n,m) ，每一类资源已分配给每一进程的资源数
- Need需求矩阵： (n,m) ，每一进程还需要的各类资源数量
 1. 先判断申请资源是否小于声明的最大量
 2. 再判断当前资源够不够分给它
 3. 试分配。修改Available,Allocation,Need
 4. 执行安全性算法，检测分配后系统是否处于安全状态

安全性算法（找到一个安全序列）

1. 为所有进程检查是否存在需求小于可分资源数量的进程。
2. 如果可分，其顺利执行完之后会返还资源，将可分资源数加上已分配给该进程的资源数
3. 否则看别的进程
4. 直到所有进程都看完，如果所有进程都可以分配，则系统处于安全状态

死锁检测与解除

- 一旦死锁出现，让一些进程回滚，让出资源。
- 检测死锁方法对系统资源的分配不加限制，只要有则可以分配。

线程

引入线程后，线程是**CPU调度**的基本单位（资源分配和调度的基本单位仍是进程）。在多CPU环境下，各个线程也可以分派到不同的CPU上并行地执行。

线程具有三种基本状态：就绪、执行和阻塞。一般不具有挂起状态。因为线程共享进程的资源，如果挂起一个进程，其所属的全部进程必将被挂起。

一个进程包括一个或多个线程。

关于cpu运行时间，内核级线程看线程数，用户级线程看进程数


处理机调度

调度的层次

- 高级调度（作业调度、长程调度、宏观调度）
 - 对外存上的作业进行调度，调入内存并建立PCB，分配必要的资源
 - 每个作业只调入一次，调出一次
 - 接纳多少个作业：取决于多道程序度
 - 接纳哪些作业：取决于调度算法
- 中级调度（内存调度、交换调度）
 - 将进程在内存和外存间换进换出
 - 决定将哪个处于挂起状态的进程重新调入内存
 - PCB常驻内存
 - 一个进程可能会被多次调出调入内存
- 低级调度（进程调度、短程调度）
 - 将哪一个就绪进程分配到CPU，并将CPU实际分配给这个进程
 - 运行频率高

调度层次	所做操作	调度时机	发生频率	对进程状态的影响
高级调度 (作业调度)	按照某种规则，从后备队列中选择合适的作业将其调入内存，并为其创建进程。	外存→内存 (面向作业)	最低	创建态→就绪态
中级调度 (内存调度)	按照某种规则，从挂起队列中选择合适的进程将其数据调回内存。	外存→内存 (面向进程)	中等	挂起态→就绪态 (阻塞挂起→阻塞态)
低级调度 (进程调度)	按照某种规则，从就绪队列中选择一个进程为其分配处理机。	内存→CPU (面向进程)	最高	就绪态→运行态

进程在操作系统**内核程序临界区**中不能进行调度与切换 

进程处于**临界区**时不能进行处理机调度 

调度算法的评价指标

- CPU资源利用率
- 系统吞吐量：总共完成了多少道作业/总共花了多少时间
- 周转时间：从作业提交给系统开始到作业完成为止这段时间
 - 带权周转时间：作业周转时间/作业实际运行时间
- 等待时间：进程/作业处于等待处理机状态时间之和
 - 等待I/O完成不计入等待时间
- 响应时间：从用户提交请求到首次产生响应所用的时间

设计原则：

- 批处理系统：平均周转时间短、系统吞吐量高、处理机利用率高
- 分时系统：响应时间快、均衡性好（时间片轮转法）
- 实时系统：截止时间的保证、可预测性好

调度算法

先来先服务FCFS

选择最先进入队列的作业，直到进程完成或阻塞。

- 非抢占
- 不饥饿
- 有利于长作业、CPU繁忙的作业，不利于短作业
- 适用于作业调度和进程调度

短作业优先调度SJF/SPF

选择估计运行时间最短的作业

- 非抢占
- 会饥饿
- 有利于短作业，可提高系统的吞吐量，降低作业平均等待时间
- 适用于作业调度和进程调度

最短剩余时间优先调度SRTN

抢占式的SJF，优先运行期望剩余时间最短的作业（优先级最高）

- 抢占
- 轮转时间性能比SJF好
- 需要记录已执行的时间，增加了开销

优先级调度（静态优先级）

分为非抢占式和抢占式

最高响应比优先调度HRRN（动态优先级）

$$\text{优先权} = \frac{\text{等待时间} + \text{要求服务的时间}}{\text{要求服务的时间}} = \frac{\text{响应时间}}{\text{要求服务时间}}$$

- 很难准确估计进程的执行时间
- 每次调度之前都需要计算响应比，增加了系统开销

时间片轮转调度算法RR

- 只应用于进程调度

多级反馈队列算法MLFQ

根据进程执行历史调度

实时调度

- 提供必要的调度信息
- 系统处理能力强
- 采用抢占式调度机制
- 具有快速切换机制

实时调度算法

- 通常采用抢占式的优先数高者优先
- 松弛度 = 必须完成时间 - 本身运行时间 - 当前时间（还剩的时间-已经运行了的时间）

习题

1. 程序性中断事件是由处理器执行机器指令出错或异常引起的，下面哪个事件属于程序性中断：

- A. 键盘/鼠标信号中断
 - B. 间隔时钟中断
 - C. 非法指令、地址越界等指令异常
 - D. 关机/重启动中断
2. 下列不属于**强迫性中断**事件的是
- A. 访管中断
 - B. 外部中断
 - C. 程序性中断
 - D. 硬件故障中断
3. 操作系统在中断处理服务程序完成之后，将()
- A. 选择刚被中断的程序执行
 - B. 选择另一中断服务程序执行
 - C. 等待下一事件发生
 - D. 按调度程序选择某程序执行
4. 系统调用的目的是()
- A. 请求系统服务
 - B. 中止系统服务
 - C. 转移系统服务
 - D. 释放系统资源
5. 一个进程映像是()
- A. 由协处理器执行的一个程序
 - B. 一个独立的程序+数据集
 - C. PCB结构与程序和数据的组合
 - D. 一个独立的程序
6. 在单处理器系统中，若同时存在10个进程，则处于阻塞队列中的进程最多有()几个。
- A. 1
 - B. 8
 - C. 9
 - D. 10
7. 下列选项中，导致创建新进程的操作是()
- I. 用户登录成功 II. 设备分配 III. 启动程序执行
- A. I和II
 - B. II和III
 - C. I和III
 - D. I、II、III
8. 一个计算机系统中，进程的最大数目主要受到()限制。
- A. 内存大小
 - B. 用户数目
 - C. 打开文件的数目
 - D. 外部设备数量
9. 信箱通信属于：
- A. 直接通信
 - B. 间接通信
 - C. 低级通信

- D. 信号量
10. 下列进程间通信方式中，通信效率最高的是：
- A. 共享存储区
 - B. 共享数据结构
 - C. 管道
 - D. 邮箱
11. 下列关于进程和线程的说法中正确的是：
- A. 一个进程可以包含一个或多个线程
 - B. 一个线程可以属于多个进程
 - C. 线程又称为轻型进程，因为线程都比进程小
 - D. 由于线程拥有自己的私有资源，所以线程之间可以无约束地并行执行
12. 关于多线程和多进程，下面描述错误的是：
- A. 多进程里，子进程可获得父进程的所有堆栈和数据
 - B. 线程会与同进程的其他线程共享进程数据，但拥有自己的栈空间
 - C. 同一进程下的线程切换更快，因为它们在同一地址空间内
 - D. 在多线程里，每个子进程有自己的地址空间，因此相互之间通信不如进程灵活和方便
13. 以下关于用户线程的描述，错误的是（ ）
- A. 用户线程由用户线程库进行管理
 - B. 用户线程的创建和调度需要内核的干预
 - C. 操作系统无法调度用户线程
 - D. 内核看不到用户线程
14. 同一个进程中的线程，不可以共享（ ）。
- A. 公有数据
 - B. 打开文件列表
 - C. 堆栈
 - D. 代码
15. 某分时系统采用多对一线程模型，内存中有10个进程并发运行，其中9个进程中只有一个线程，另外一个进程A拥有11个线程，则A获得的CPU时间占总时间的（ ）
- A. $1/20$
 - B. 0
 - C. $1/10$
 - D. 1
16. 某操作系统不支持内核级线程但支持用户级线程，且处理器采用时间片轮转调度算法。该系统现有进程A和进程B，且进程A拥有1个线程、进程B拥有100个线程，那么进程A执行时间一般应为进程B执行时间的（ ）。
- A. $1/100$ 倍
 - B. 1倍
 - C. 50倍
 - D. 100倍
17. 某操作系统支持内核级线程，且处理器采用时间片轮转调度算法。该系统现有进程A和进程B，且进程A拥有20个线程、进程B拥有2个线程，那么进程A执行时间一般应为进程B执行时间的：
- A. 10倍
 - B. 5倍
 - C. $1/5$ 倍
 - D. 1倍

18. 对信号量S的 wait 原子操作定义中, 使进程进入相应阻塞队列等待的条件是:

- A. $S > 0$
- B. $S = 0$
- C. $S < 0$
- D. $S \neq 0$

19. 在操作系统中, 对信号量S的 signal 原子操作定义中, 唤醒一个等待进程的条件是:

- A. $S > 0$
- B. $S < 0$
- C. $S \geq 0$
- D. $S \leq 0$

20. 进程P1和P2均包含并发执行的线程, 部分伪代码描述如下:

// 进程P1

```
int x=0;
```

```
Thread1 ( )
```

```
{   int a;
```

```
    a=1;   x += 1;
```

```
}
```

```
Thread2 ( )
```

```
{   int a;
```

```
    a=2;   x += 2;
```

```
}
```

// 进程P2

```
int x=0;
```

```
Thread3 ( )
```

```
{   int a;
```

```
    a=x;   x += 3;
```

```
}
```

```
Thread4 ( )
```

```
{   int b;
```

```
    b=x;   x += 4;
```

```
}
```

下列选项中, 需要互斥执行的操作是 ()

- A. $a=1$ 与 $a=2$
- B. $a=x$ 与 $b=x$
- C. $x+=1$ 与 $x+=2$
- D. $x+=1$ 与 $x+=3$

21. Windows系统中的线程普遍采用的多线程模型是一对一模型。

- A. 对
- B. 错

22. 设有一个售票大厅, 可容纳50人购票。如果厅内不足50人, 则允许进入, 超过则在厅外等候; 售票员某时是能给一个购票者提供服务, 购票者买完票后离开。请回答:

- (1) 购票者之间是同步关系还是互斥关系?
- (2) 用P、V操作描述购票者的工作过程。

23. 有一个阅览室, 共有100个座位, 读者进入时必须先在一张登记表上登记, 该表为一个座位列一个表目, 包括座位号和读者姓名。读者离开时要把表上登记的内容撤销掉, 使用P、V操作描述读者进程的同步结构。

24. 在9个生产者、6个消费者共享容量为8的缓冲区的生产者-消费者问题中，互斥使用缓冲区的信号量初始值为（ ）
- A. 1
B. 9
C. 6
D. 8

25. 使用信号量机制解决独木桥问题。

某条河上只有一座独木桥，以便行人过河。现在河的两边都有人要过桥，按照下面的规则过桥。为了保证过桥安全，请用P、V操作分别实现正确的管理。

过桥的规则是：同一方向的可连续过桥，某方向有人过桥时另一方向的人要等待。

26. 作业是用户提交的，进程是系统自动生成的，除此之外，两者的区别是（ ）
- A. 两者执行不同的程序段
B. 前者以用户任务为单位；后者以操作系统控制为单位。
C. 前者是批处理的，后者是分时的
D. 后者是可以并发执行，前者则不同

27. 某系统正在执行三个进程P1,P2和P3，各进程的计算(CPU)时间和I/O时间比例如表所示。

进程	计算时间	I/O 时间
P1	90%	10%
P2	50%	50%
P3	15%	85%

为提高系统资源利用率，合理的进程优先级设置应为（ ）

- A. P1>P2>P3
B. P3>P2>P1
C. P2>P1=P3
D. P1>P2=P3
28. 具有两道作业的批处理系统，作业调度采用短作业优先SJF调度算法，进程调度采用抢占式优先数调度算法。作业运行情况如表。其中作业的优先数即进程的优先数，**优先数越小，优先级越高**（忽略其他系统开销）。1) 列出所有作业进入内存的时间及结束的时间（以分钟为单位）(1)~(8)，请按空格顺序填空。2) 计算各作业的周转时间(9)~(12)；计算平均周转时间(13)。请按空格顺序填空。

作业	到达时间	运行时间	优先数	进入内存时间	结束时间	周转时间
1	8:00	40min	5	(1)	(2)	(9)
2	8:20	30min	3	(3)	(4)	(10)
3	8:30	50min	4	(5)	(6)	(11)
4	8:50	20min	6	(7)	(8)	(12)

29. 在单CPU和两台I/O (I1,I2)设备的多道程序设计环境下，同时投入三个作业。它们的执行轨迹如下：Job1: I2(30ms)、 CPU(10ms)、 I1(30ms)、 CPU(10ms)、 I2(20ms)

Job2: I1(20ms)、 CPU(20ms)、 I2(40ms)

Job3: CPU(30ms)、 I1(20ms)、 CPU(10ms)、 I1(10ms)

如果CPU、I1和I2都能并行工作，优先级从高到低为Job1、Job2和Job3，优先级高的作业可以抢占优先级低的作业的CPU，但不抢占I1和I2。

试求：

(1)Job1从投入到完成分别所需的时间 (1);

Job2从投入到完成分别所需的时间 (2);

Job3从投入到完成分别所需的时间 (3)

(2)从投入到完成CPU的利用率 (4) ; (小数点后保留一位)

从投入到完成I/O设备I1 的利用率 (5) ; (小数点后保留一位)

从投入到完成I/O设备I2 的利用率 (6) 。 (小数点后保留一位)

30. 某计算机系统中有10个A类资源，有K个进程竞争使用，每个进程最多需要3个A类资源。该系统可能会发生死锁的K的最小值是 () 。

31. 设m为同类资源R的数目，n为系统中的并发进程数。当n个进程共享m个互斥资源R时，每个进程对R的最大需求是w；则下列情况会出现死锁的是 。

A. m=4, n=3, w=2

B. m=4, n=2, w=3

C. m=2, n=1, w=2

D. m=2, n=2, w=1

32. 设系统中有5个进程P1、P2、P3、P4、P5，有三种类型的资源A、B、C。其中A资源的数量是17，B资源的数量是5，C资源的数量是20。T0时刻系统状态如表所示

资源分配表

进程	已分配资源数量 (Allocation)			最大资源需求量 (Max)			仍然需求资源数 (Need)		
	A	B	C	A	B	C	A	B	C
P1	2	1	2	5	5	9			
P2	4	0	2	5	3	6			
P3	4	0	5	4	0	11			
P4	2	0	4	4	2	5			
P5	3	1	4	4	2	4			

(1)计算每个进程还可能需要的资源，并填入表的“仍然需求资源数”栏中。

(2)T0时刻系统是否处于安全状态？为什么？

(3)如果T0时刻进程P2又有新的资源请求(0,3,4)，是否实施资源分配？为什么？

(4)如果T0时刻进程P4又有新的资源请求(2,0,1)，是否实施资源分配？为什么？

(5)在(4)的基础上，若进程P1又有新的资源请求(0,2,0)，是否实施资源分配？为什么？

答案

1~5 CADCC

6.D 可以都不执行也不就绪。最少有0个。

7.C 设备分配通过设置相应数据结构实现，不需要创建进程

8~14 ABAADBD

15.C 多对一线程模型，指多个用户级线程映射到一个内核线程。进程A虽有11个线程，但对应1个内核线程。因此CPU按照内核线程来实现CPU分配和调度，因此为1/10。

20.C P1和P2两个进程中的x是互相独立的互不干扰，只需要考虑进程内部两个并发执行的线程之间是否存在互斥操作即可。

在P1中，a分别各个线程中自定义，互不干涉，无论先执行谁，都是Thread1中的a=1，Thread2中的a=2。对于Thread1中的x先执行的话是1，后执行的话是3。Thread2中的x先执行的话是2，后执行的话是3，因此x+=1与x+=2存在互斥。

在P2中，a,b分别各个线程中自定义，互不干涉，无论先执行谁，都是Thread3中的a=0，Thread4中的b=0。对于Thread3中的x先执行的话是3，后执行的话是7。Thread4中的x先执行的话是4，后执行的话是7，因此x+=3与x+=4存在互斥。

21.A

22.

```

1  semaphore empty=50; // 50个人共享售票大厅，设信号量empty初值为50，如果50个都被申请
   empty<0，购票者不能进入大厅
2  semaphore mutex=1; //购票只能互斥进行，设信号量mutex初值为1，售票员，某时只能为一个购票
   者提供服务
3  void buyer() //购票者进程
4  {
5      P(empty);    //申请进入大厅，占用1个名额
6      P(mutex);    //申请购票权
7      购票;
8      V(mutex);    //释放购票权
9      V(empty);    //退出大厅，释放1个名额
10 }

```

23.

```

1  semaphore empty=100; //100个位置共享使用
2  semaphore mutex=1;  //1个表格互斥使用
3  void reader() //读者进程
4  {
5      P(empty);    //申请座位
6      P(mutex);    //申请表
7      登记填表;
8      V(mutex);    //释放表
9      读书;
10
11     P(mutex);    //申请表
12
13     撤销登记内容;
14     V(mutex);    //释放表
15     V(empty);    //释放座位
16 }

```

24.A

25.

```

1  semaphore SA,SB,mutex;
2  SA=1; SB=1; mutex=1;
3  int countA=0, countB=0;
4  void Process_A()

```

```

5  {
6      wait(SA);
7      if (countA == 0)    {
8          wait(mutex);
9      }
10     countA+=1;
11     signal(SA);
12     过独木桥;
13     wait(SA);
14     countA -=1;
15     if (countA==0) {
16         signal(mutex);
17     }
18     signal(SA);
19 }
20 void Process_B()
21 {
22     wait(SB);
23     if (countB==0){
24         wait(mutex);
25     }
26     countB+=1;
27     signal(SB);
28     过独木桥;
29     wait(SB);
30     countB-=1;
31     if (countB==0) {
32         signal(mutex);
33     }
34     signal(SB);
35 }

```

26.B

27.B 为合理设置进程优先级，需综合考虑CPU时间和IO时间。通常IO型作业优先权高于计算型作业优先权，因为IO操作需要及时完成，它无法长时间保存要输入输出数据，所以选择IO繁忙型作业优先权高。

28.作业是调入内存，进程是抢占CPU

- 1) 具有两道作业的批处理系统，内存只存放两道作业，它们采用抢占式优先级调度算法竞争 CPU，而将作业调入内存采用的是短作业优先调度。8:00，作业 1 到来，此时内存和处理机空闲，作业 1 进入内存并占用处理机；8:20，作业 2 到来，内存仍有一个位置空闲，因此将作业 2 调入内存，又由于作业 2 的优先数高，相应的进程抢占处理机，在此期间 8:30 作业 3 到来，但内存此时已无空闲，因此等待。直至 8:50，作业 2 执行完毕，此时作业 3、4 竞争空出的一道内存空间，作业 4 的运行时间短，因此先调入，但它的优先数低于作业 1，因此作业 1 先执行。到 9:10 时，作业 1 执行完毕，再将作业 3 调入内存，且由于作业 3 的优先数高而占用 CPU。所有作业进入内存的时间及结束的时间见下表。

作业	到达时间	运行时间	优先数	进入内存时间	结束时间	周转时间
1	8:00	40min	5	8:00	9:10	70min
2	8:20	30min	3	8:20	8:50	30min
3	8:30	50min	4	9:10	10:00	90min
4	8:50	20min	6	8:50	10:20	90min

- 2) 平均周转时间为 $(70 + 30 + 90 + 90)/4 = 70\text{min}$ 。

29.(1) 110ms

(2) 90ms

(3) 110ms

(4) 72.7%

(5) 72.7%

(6) 81.8%

30.D

31.B $m > n \cdot (w-1)$ 则不会死锁

32.(1)

3 4 7

1 3 4

0 0 6

2 2 1

1 1 0

(2)

处于安全状态

存在安全序列：P4->P5->P1->P2->P3

(3)不实施。在work数组中，C资源可用数量为3，不能满足P2进程对C资源4的请求。

(4)实施。首先试分配，此时可用资源变成[0, 3, 2]，并执行安全性算法。试分配后，存在安全序列：P4->P5->P1->P2->P3。因此可以分配资源。（在第二问中，安全序列中第一个就是P4，可以直接得出可以分配资源给P4）

(5)不实施。首先试分配，并执行安全性算法。此时找不出一个安全序列，使得试分配后系统处于安全状态。因此应该不实施分配，让进行P1等待。