概述
- 计算机的四个组成部分：处理器 内存 输入 / 输出模块 系统总线
- 定义处理器寄存器的两种类别：用户可见寄存器 控制和状态寄存器
- 多个中断的处理方式：
  - 正在处理一个中断时，禁止再发生中断
  - 定义中断优先级
- 内存层次各个元素间的特征：价格，容量和访问时间
- 两种局部性区别：
  - 空间局部性：最近被访问的元素的周围将来可能会被访问；利用更大的缓冲块并且在存储器控制逻辑中加入预处理机制
  - 时间局部性：近被访问的元素将来可能会被再次访问；在高速缓冲存储器中保留最近使用的指令及数据
操作系统概述
- 作系统设计的三个目标：方便 有效 扩展的能力
- 进程的组成：可执行的程序 需要的相关数据 执行上下文
- 地址的区别：
  - 实地址：指的是主存中的地址，实际的主存储器的地址，即物理空间
  - 虚地址：指的是存在于虚拟内存中的地址，可能在磁盘中或者主存中
- 时间片轮询调度技术：所有的进程存放在一个环形队列中并按固定循序依次激活
- 对称多处理器操作系统设计的关键问题：并发调度 同步 内存管理 可靠性和容错性
- API 侧重于向上层提供服务，而系统调用则侧重于通过软中断向下层的内核发出一个明确的请求
- 中断与当前正在运行的进程无关的某些类型的外部事件相关，陷阱与当前正在运行的进程所产生的错误或异常条件相关
进程
- 创建进程的事件：新的批处理作业 交互登录 提供一项服务而创建 由现有的进程派生
- 进程控制块中的三类信息：进程标识 处理器状态信息 进程控制信息
- 程模型中的状态：新建态 就绪态 运行态 阻塞态 挂起态 退出态
- 进程的几个定义：正在执行的程序 程序实例 由处理器执行的实体 执行的指令、当前状态和相关的系统资源表征的活动单元
- 创建一个新进程所执行的步骤：
  - 分配一个唯一的进程标识号
  - 分配空间
  - 初始化进程控制块
  - 设置正确的连接
  - 创建或扩充其他的数据结构
- 进程映像：程序 + 数据 + 栈 + 属性
- 进程标识信息：存储在 PCB 中的数字标识符，包括：进程 ID，父进程 ID，用户 ID
- 把操作系统作为一组系统进程来实现的优点是：
  - 鼓励模块化操作系统设计原理，使模块间接口最小且最简单
  - 非关键系统功能可简单的用独立的进程来实现
  - 在多处理器和多机环境中很有用
- 用户态到内核态切换的三种方式：系统调用 异常 外设中断
线程
- 线程间的模式切换比进程间的模式切换开销更低的原因：包含的状态信息更少
- 单用户多处理系统中使用线程场景：前台和后台操作 异步处理 加速执行 模块化程序结构
- 一个进程中的所有线程共享的资源：地址空间 文件资源 执行特权
- 用户级线程优于内核级线程的优点：
  - 线程切换不需要内核级模式的特权
  - 调用可以是应用程序专用的
  - 可以在任何操作系统中运行
- 用户级线程较之内核级线程的缺点：
  - 许多系统调用都会引起阻塞
  - 不能利用多处理器技术
- 线程的优点：
  - 创建和终止的时间少于进程
  - 在内线程间切换的时间少于进程
  - 提高了不同执行程序间的通信效率
- 基本操作：派生，阻塞，解除阻塞，结束
- 线程同步：同步线程的活动是它们互不干扰且不破坏数据结构，如两个线程向一个链表加入元素，则可能会丢失
并发
- 程间的三种互相知道的程度：
  - 相互不知道对方的存在 是竞争关系
  - 间接知道对方的存在 合作行为
  - 直接知道对方的存在 合作行为
- 三个控制问题：
  - 互斥 访问一个不可共享的资源(临界资源)
  - 死锁 每个进程都在等待另一个被对方占用的资源
  - 饥饿 一个进程被无限地拒绝访问资源
- 管程的特点：
  - 外部过程不能访问局部数据变量
  - 一个进程通过调用管程的一个过程进入管程
  - 只能有一个进程在管程中执行,任何其他进程都被阻塞
- 处理死锁的三种方法：预防、检测和避免
- 并发的原理：
  - 全局资源的共享充满了危险
  - OS 很难对资源进行最优化分配
  - 定位程序设计错误非常困难
死锁
- 产生死锁的必要条件：互斥 占有且等待 不可抢占 循环等待(是前三者的结果)
- 死锁避免的优点：无须死锁预防的抢占和回滚进程，且与死锁预防相比限制较少。
- 死锁避免的限制：
  - 必须实现声明每个进程请求的最大资源
  - 所讨论的进程必须是无关的，即它们的执行顺序必须没有同步要求的限制
  - 分配的资源数量必须是固定的
  - 在占有资源时，进程不能退出
- 死锁检测：不限制资源访问或约束进程行为，只要有可能就会给进程分配其所需要的资源，操作系统周期性的执行一个算法来检测前面的条件
内存管理
- 内存管理满足的需求：重定位 保护 共享 逻辑组织 物理组织
- 页表项中能找到的典型元素：页号 进程标志符 控制位 链指针
- 固定分区方案简单但有缺点：
  - 限制了系统中活动进程的数量
  - 小作业不能有效地利用分区空间
- 伙伴系统：二分法确定分配块的大小，一定是整个内存空间的二的幂次分之一
虚拟内存
- 进程执行过程中任何时刻都在内存中的部分称为进程的常驻集，当访问一个不再内存中 (驻留集) 的逻辑地址，会产生一个中断，操作系统会将此进程置于 阻塞态 ，为此操作系统产生一个 磁盘 I/O 读请求。
- 虚拟内存的开销受到系统抖动的影响，会导致处理器大部分的时间都用于交换块而非执行指令。
调度
- 三种类型的处理器调度： 长程调度 中程调度 短程调度
- 最重要的性能要求是：响应时间
- 周转时间和响应时间的区别：
  - 周转时间：一个进程从提交到完成
  - 响应时间：提交一个请求到开始接收响应
- 简单定义调度策略：
  - FCFS 调度：进程就绪后加入就绪队列。当前正运行的进程停止执行时，选择就绪队列中存在时间最长的进程运行。
  - 轮转调度：周期性地产生时钟中断，正运行的进程会放置到就绪队列中，基于 FCFS 策略选择下一个
  - 最短进程优先调度：下次选择预计处理时间最短的进程(非抢占策略)
  - 最短剩余时间调度：总是选择预期剩余时间最短的进程
  - 最高响应比优先调度：当前进程完成或被阻塞时，选择 R 值最大的就绪进程。基于对归一化周转时间的估计。
  - 反馈调度：基于抢占原则并使用动态优先级机制，基于每个进程的执行历史把它们分配到各个队列中
文件管理
- 文件组织的选择原则：快速修改 易于修改 节约存储空间 维护简单 可靠性
- 基本文件组织：
  - 堆：是简单的文件组织形式，数据按它们到达的顺序被收集
  - 顺序文件：每条记录都使用固定的格式，所有记录具有相同的长度，并由相同数量、长度固定的域按特定顺序组成。每条记录的第一个域通常是关键域
  - 索引顺序文件：用于支持随机访问的文件索引和溢出文件。索引顺序文件极大的减少了访问单条记录的时间
- 记录组块：定长组块 变长跨越式组块 变长非跨越式组块

1.1 列出并简要定义计算机的四个组成部分。
处理器：控制计算机的操作，执行数据处理功能。
内存：也叫主存储器，存储数据和程序。
输入 / 输出模块：在计算机和外部环境之间移动数据。
系统总线：在处理器、内存和输入输出间提供通信的设施。
1.2 定义处理器寄存器的两种主要类别。
用户可见寄存器：优先使用这些寄存器，可以使机器语言或者汇编语言的程序员减少对主存储器的访问次数。对高级语言而言，由优化编译器负责决定把哪些变量应该分配给主存储器，一些高级语言，如 C 语言，允许程序言建议编译器把哪些变量保存在寄存器中。
控制和状态寄存器：用以控制处理器的操作，且主要被具有特权的操作系统例程使用，以控制程序的执行。
1.3 一般而言，一条机器指令能指定的四种不同操作是什么？
处理器 - 寄存器：数据可以从处理器传送到存储器，或者从存储器传送到处理器。
处理器 - I/O：通过处理器和 I/O 模块间的数据传送，数据可以输出到外部设备，或者从外部设备输入数据。
数据处理：处理器可以执行很多关于数据的算术操作或者逻辑操作。
控制：某些指令可以改变执行顺序。
1.4 什么是中断？
中断是指计算机运行过程中，出现某些意外情况需主机干预时，机器能自动停止正在运行的程序并转入处理新情况的程序，处理完毕后又返回原被暂停的程序继续运行。
1.5 多个中断的处理方式是什么？
处理多中断有两种方法。
第一种方法是当正在处理一个中断时，禁止再发生中断。
第二种方法是定义中断优先级，允许高优先级的中断打断低优先级的中断处理器的运行。
1.6 内存层次各个元素间的特征是什么？
存储器的三个重要特性是：价格，容量和访问时间。并且各层次从上到下，每“位”价格降低，容量递增，访问时间递增。
1.7 什么是高速缓存？
高速缓冲存储器是比主存小而快的存储器，用以协调主存跟处理器，作为最近储存地址的缓冲区。
1.8 多处理器系统和多核系统的区别是什么？
多处理器系统（Multiprocessor Systems）是指包含两台或多台功能相近的处理器，处理器之间彼此可以交换数据，所有处理器共享内存，I/O 设备，控制器，及外部设备，整个硬件系统由统一 - 的操作系统控制，在处理器和程序之间实现作业、任务、程序、数组极其元素各级的全面并行。
多内核（multicore chips）是指在一枚处理器（chip）中集成两个或多个完整的计算引擎（内核）。
1.9 空间局部性和时间局部性的区别是什么？
空间局部性是指最近被访问的元素的周围在不久的将来可能会被访问。
临时局部性（即时间局部性）：是指最近被访问的元素在不久的将来可能会被再次访问。
1.10 开发空间局部性和时间局部性的策略是什么？
空间局部性的开发是利用更大的缓冲块并且在存储器控制逻辑中加入预处理机制。
时间局部性的开发是利用在高速缓冲存储器中保留最近使用的指令及数据，并且定义缓冲存储的优先级。
第二章 操作系统概述 ¶
2.1 操作系统设计的三个目标是什么？
方便：操作系统使计算机更易于使用。
有效：操作系统允许以更有效的方式使用计算机系统资源。
扩展的能力：在构造操作系统时，应该允许在不妨碍服务的前提下有效地开发、测试和引进新的系统功能。
2.2 什么是操作系统的内核？
操作系统内核是计算机上最低层的软件，提供计算机最核心的功能，比如：进程管理、内存管理、I/O 管理、文件管理、网络管理等。
2.3 什么是多道程序设计？
两个或两个以上程序在计算机系统中同处于开始到结束之间的状态，这就称为多道程序设计。也就是在计算机内存中同时存放几道相互独立的程序，使它们在管理程序控制之下，相互穿插的运行。多道程序技术运行的特征：多道、宏观上并行、微观上串行。
2.4 什么是进程？
进程由三部分组成：
一段可执行的程序。
程序所需要的相关数据（变量、工作空间、缓冲区等）。
程序的执行上下文（也称进程状态）。
2.5 操作系统是怎么使用进程上下文的？
执行上下文又称为进程状态，是操作系统用来管理和控制所需的内部数据。这种内部信息和进程是分开的，因为操作系统信息不允许被进程直接访问。上下文包括操作系统管理进程以及处理器正确执行进程所需要的所有信息，包括各种处理器寄存器的内容，如程序计数器和数据寄存器。它还包括操作系统使用的信息，如进程优先级以及进程是否在等待特定 I/O 事件的完成。
2.6 列出并简要介绍操作系统的五种典型存储管理职责。
进程隔离：操作系统必须保护独立的进程，防止互相干扰各自的存储空间，包括数据和指令。
自动分配和管理：程序应该根据需要在存储层次间动态的分配，分配对程序员是透明的。因此，程序员无需关心与存储限制有关的问题，操作系统有效的实现分配问题，可以仅在需要时才给作业分配存储空间。
支持模块化程序设计：程序员应该能够定义程序模块，并动态地创建、销毁模块，动态地改变模块大小。
保护和访问控制：不论在存储层次中的哪一级，存储器的共享都会产生一个程序访问另一个程序存储空间的潜在可能性。当某个特定的应用程序需要共享时，这是可取的。但在其他时候，他可能会威胁到程序的完整性，甚至威胁到操作系统本身。
长期存储：许多应用程序需要在计算机关机后长时间的保存信息。
2.7 解释实地址和虚地址的区别。
实地址：指的是主存中的地址，实际的主存储器的地址，对应主存空间，即物理空间。
虚地址：指的是存在于虚拟内存中的地址，它有时候在磁盘中，有时候在主存中。
2.8 描述时间片轮询调度技术。
轮询调度是一种调度算法，所有的进程存放在一个环形队列中并按固定循序依次激活。因为等待一些事件（例如：等待一个子进程或者一个 I/O 操作）的发生而不能被处理的进程将控制权交给调度器。
2.9 解释单体内核和微内核的区别。
单体内核是一个提供操作系统应该提供的功能的大内核，包括调度、文件系统、网络、设备驱动程序、存储管理等。内核的所有功能成分都能够访问它的内部数据结构和程序。典型情况下，这个大内核是作为一个进程实现的，所有元素都共享相同的地址空间。
微内核是一个小的有特权的操作系统内核，只提供包括进程调度、内存管理和进程间通信等基本功能，要依靠其他进程担当起和操作系统内核联系作用。
2.10 什么是多线程？
多线程技术是指把执行一个应用程序的进程划分为可以同时运行的多个线程。
2.11 列出对称多处理器操作系统设计时要考虑的关键问题。
并发（concurrent）进程或线程：内核程序应可重入，以使多个处理器能同时执行同一段内核代码。当多个处理器执行内核的相同或不同部分时，为避免数据损坏和无效操作，需要妥善管理内核表和数据结构。
调度：任何一个处理器都可以执行调度，这既增加了执行调度策略的复杂度，也增加了保证调度相关数据结构不被损坏的复杂度。如果使用的是内核级多线程方式，就存在将同一进程的多个线程同时调度在多个处理器上的可能性。
同步：因为可能会存在多个活跃进程访问共享地址空间或共享 I/O 资源的情况，因此必须认真考虑如何提供有效的同步机制这一问题。 同步用来实现互斥及事件排序。
内存管理：多处理器上的内存管理要处理单处理器上内存管理涉及的所有问题。另外，操作系统还要充分利用硬件提供的并行性来实现最优性能。不同处理器上的分页机制必须进行调整，以实现多处理器共享页或段时的数据一致性，执行页面置换。物理页的重用是我们关注的最大问题，即必须保证物理页在重新使用前不能访问到它以前的内容。
可靠性和容错性：出现处理器故障时，操作系统应能妥善地降低故障的影响。调度器和操作系统的其他部分必须能识别出发生故障的处理器，并重新组织管理表。
第二部分 进程 ¶
第三章 进程描述和控制 ¶
3.1 什么是指令跟踪？
指令跟踪是指为该进程而执行的指令序列。
3.2 通常哪些事件会导致创建一个进程？
新的批处理作业；
交互登录；
操作系统因为提供一项服务而创建；
由现有的进程派生。
3.3 简要定义图 3.6 所示进程模型中的每种状态。
运行态：该进程正在执行。
就绪态：进程做好了准备，只要有机会就会开始执行。
阻塞态：进程在某些事件发生前不能执行，如 I/O 操作完成。
新建态：刚刚创建的进程，操作系统还没有把它加入到可执行进程组中。
退出态：操作系统从可执行进程中释放出的进程，或者是因为它自身停止了，或者是因为某种原因被取消。
3.4 被抢占一个进程是什么意思？
处理器为了执行另外的进程而终止当前正在执行的进程，这就叫进程抢占。
3.5 什么是交换，其目的是什么？
交换是指把主存中某个进程的一部分或者全部内容转移到磁盘。当主存中没有处于就绪态的进程时，操作系统就把一个阻塞的进程换出到磁盘中的挂起队列，从而使另一个进程可以进入主存执行。
3.6 为什么图 3.9（b）中有两个阻塞态？
有两个独立的概念：进程是否在等待一个事件（阻塞与否）以及进程是否已经被换出主存（挂起与否）。为适应这种 2*2 的组合，需要两个阻塞态和两个挂起态。
3.7 列出挂起状态进程的 4 个特点。
进程不能立即执行。
进程可能是或不是正在等待一个事件。如果是，阻塞条件不依赖于挂起条件，阻塞事件的发生不会使进程立即被执行。
为了阻止进程执行，可以通过代理把这个进程置于挂起态，代理可以是进程自己，也可以是父进程或操作系统。
除非代理显式地命令系统进行状态转换，否则进程无法从这个状态中转移。
3.8 对于哪类实体，操作系统为了管理它而维护其信息表？
内存、I/O、文件 和 进程。
3.9 列出进程控制块中的三类信息。
进程标识，处理器状态信息，进程控制信息。
3.10 为什么需要两种模式（用户模式和内核模式）？
用户模式下可以执行的指令和访问的内存区域都受到限制。这是为了防止操作系统受到破坏或者修改。而在内核模式下则没有这些限制，从而使它能够完成其功能。
3.11 操作系统创建一个新进程所执行的步骤是什么？
给新进程分配一个唯一的进程标识号。
给进程分配空间。
初始化进程控制块。
设置正确的连接。
创建或扩充其他的数据结构。
3.12 中断和陷阱有什么区别？
在程序运行过程中，系统出现了一个必须由 CPU 立即处理的情况，此时，CPU 暂时中止程序的执行转而处理这个新的情况的过程就叫做中断。中断与当前正在运行的进程无关的某些类型的外部事件相关，如完成一次 I/O 操作。
陷阱指的是当异常或者中断发生时，处理器捕捉到一个执行线程，并且将控制权转移到操作系统中某一个固定地址的机制。陷阱与当前正在运行的进程所产生的错误或异常条件相关，如非法的文件访问。
3.13 举出中断的三个例子。
时钟中断，I/O 中断，内存失效。
3.14 模式切换和进程切换有什么区别？
发生模式切换可以不改变当前正处于运行态的进程的状态。
发生进程切换时，一个正在执行的进程被中断，操作系统指定另一个进程为运行态。进程切换需要保存更多的状态信息。
第四章 线程 ¶
4.1 表 3.5 列出了在一个没有线程的操作系统中进程控制块的基本元素。对于多线程系统，这些元素中哪些可能属于线程控制块，哪些可能属于进程控制块？
这对于不同的系统来说通常是不同的，但一般来说，进程是资源的所有者，而每个线程都有它自己的执行状态。关于表 3.5 中的每一项的一些结论如下：
进程标识：进程必须被标识，而进程中的每一个线程也必须有自己的 ID。
进程控制信息：调度和状态信息主要处于线程级；数据结构在两级都可出现；进程间通信和线程间通信都可以得到支持；特权在两级都可以存在；存储管理通常在进程级；资源信息通常也在进程级；
处理器状态信息：又名现场信息，一个进程在运行时存放在处理器现场中的各种信息。这些信息通常只与进程有关。
进程控制信息，用于管理和调度一个进程。常用的控制信息包括：
进程的调度相关信息，如进程状态、等待事件和等待原因、进程优先级、队列指引元等
进程组成信息，如正文段指针、数据段指针
引进程间通信相关信息，如消息队列指针、信号量等互斥和同步机制
进程在辅存储器内的地址
CPU 资源的占用和使用信息，如时间片余量、进程已占用 CPU 的时间、进程已执行的时间总和，记账信息
进程特权信息，如在内存访问和处理器状态方面的特权
4.2 请列出线程间的模式切换比进程间的模式切换开销更低的原因。
包含的状态信息更少。
4.3 在进程概念中体现出的两个独立且无关的特点是什么？
资源所有权：进程包括存放进程映像的虚拟地址空间；回顾第 3 章的内容可知，进程映像是程序、数据、栈和进程控制块中定义的属性集。进程总具有对资源的控制权或所有权，这些资源包括内存、I/O 通道、I/O 设备和文件等。操作系统提供预防进程

间发生不必要资源冲突的保护功能。
调度 / 执行: 进程执行时采用一个或多程序 (见图 1.5) 的执行路径 (轨迹)，不同进程的执行过程会交替进行。因此，进程具有执行态 (运行、就绪等) 和分配给其的优先级，是可被操作系统调度和分派的实体。
4.4 给出在单用户多处理系统中使用线程的四个例子。
前台和后台操作
异步处理
加速执行
模块化程序结构。
4.5 哪些资源通常被一个进程中的所有线程共享?
进程中的所有线程共享该进程的状态和资源，例如地址空间，文件资源，执行特权等。
4.6 列出用户级线程由于内核级线程的三个优点。
由于所有线程管理数据结构都在一个进程的用户地址空间中，线程切换不需要内核级模式的特权，因此，进程不需要为了线程管理而切换到内核模式，这节省了在两种模式间进行切换 (从用户模式到内核模式；从内核模式返回用户模式) 的开销。
调用可以是应用程序专用的。一个应用程序可能倾向于就按单的轮询调度算法，而另一个应用程序可能倾向于基于优先级的调度算法。调度算法可以去适应应用程序，而不会扰乱底层的操作系统调度器。用户级线程可以在任何操作系统中运行，不需要对底层内核进行修改以支持用户级线程。线程库是一组供所有应用程序共享的应用级软件包。
4.7 列出用户级线程相对于内核级线程的两个缺点。
在典型的操作系统中，许多系统调用都会引起阻塞。因此，当用户级线程执行一个系统调用时，不仅这个线程会被阻塞，进程中的所有线程都会被阻塞。在纯粹的用户级进程策略中，一个多线程应用程序不能利用多处理器技术。内核一次只把一个进程分配给一个处理器，因此一次进程中只能有一个线程可以执行。
4.8 定义 “套管” (jacketing)。
Jacketing 通过调用一个应用级的 I/O 例程来检查 I/O 设备的状态，从而将一个产生阻塞的系统调用转化为一个不产生阻塞的系统调用。
第五章 并发性: 互斥与同步 ¶
5.1 列出与并发相关的 4 个设计问题。
操作系统必须能够跟踪不同的进程，这可使用进程控制块来实现。
操作系统必须为每个活动进程分配和释放各种资源。有时，多个进程想访问相同的资源。这些资源包括:
处理器时间: 这是调度功能。
存储器: 大多数操作系统使用虚存方案。文件。I/O 设备。
操作系统必须保护每个进程的数据和物理资源，避免其他进程的无意干扰，这涉及与存储器、文件和 I/O 设备相关的技术。
一个进程的功能和输出结果必须与执行速度无关 (相对于其他并发进程的执行速度)。
5.2 产生并发的三种上下文是什么?
多应用程序: 多道程序设计技术允许在多个活动的应用程序间动态共享处理器时间。
结构化应用程序: 作为模块化设计和结构化程序设计的扩展，一些应用程序可被有效地设计成一组并发进程。
操作系统结构: 同样的结构化程序设计优点适用于系统程序，且我们已知操作系统自身常常作为一组进程或线程实现。
5.3 执行并发进程的最基本要求是什么?
最基本要求是增加进程间的 == 互斥 == 能力。
5.4 列出进程间的三种互相知道的程度，并简要给出各自的定义。
进程之间相互不知道对方的存在: 这是一些独立的进程，它们不会一起工作。关于这种情况的最好例子是多个独立进程的多道程序设计，可以是批处理作业，也可以是交互式会话，或者是两者的混合。尽管这些进程不会一 - 起工作，但操作系统需要知道它们对资源的竞争情况 (competition)。例如，两个无关的应用程序可能都想访问同一个磁盘、文件或打印机。操作系统必须控制对它们的访问。
进程间接知道对方的存在: 这些进程并不需要知道对方的进程 ID，但它们共享某些对象，如一个 I/O 缓冲区。这类进程在共享同一个对象时会表现出合作行为 (cooperation)。
进程直接知道对方的存在: 这些进程可通过进程 ID 互相通信，以合作完成某些活动。同样，这类进程表现出合作行为。
5.5 竞争进程和合作进程间有何区别?
竞争进程需要同时访问相同的资源，像磁盘，文件或打印机。合作进程要么共享访问一个共有的资源，像一个内存访问区，要么就与其他进程相互通信，在一些应用程序中或活动上进行合作。
5.6 列出与竞争进程相关的三个控制问题，并简要给出各自的定义。
互斥 (mutual exclusion): 假设两个或更多的进程需要访问一个不可共享的资源，如打印机。在执行过程中，每个进程都给该 I/O 设备发命令，接收状态信息，发送数据和接收数据。我们把这类资源称为临界资源 (critical resource)，使用临界资源的那部分程序称为程序的临界区 (critical section)。一次只允许有一 一个程序在临界区中，这一点非常重要。
死锁 (deadlock): 考虑两个进程 P1 和 P2，以及两个资源 R1 和 R2，假设每个进程为执行部分功能都需要访问这两个资源，那么就有可能出现下列情况: 操作系统把 R1 分配给 P2，把 R2 分配给 P1，每个进程都在等待另一个资源，且在获得其他资源并完成功能前，谁都不会释放自己已拥有的资源，此时这两个进程就会发生死锁。
饥饿 (starvation): 假设有三个进程 (P1、P2 和 P3)，每个进程都周期性地访问资源 R。考虑这种情况，即 P1 拥有资源，P2 和 P3 都被延迟，等待这个资源。当 P1 退出其临界区时，P2 和 P3 都允许访问 R，假设操作系统把访问权授予 P3，并在 P3 退出临界区之前 P1 又要访问该临界区，若在 P3 结束后操作系统又把访问权授予 P1，且接下来把访问权轮流授予 P1 和 P3，那么即使没有死锁，P2 也可能被无限地拒绝访问资源。
5.7 列出对互斥的要求。
要提供对互斥的支持，必须满足以下要求:
必须强制实施互斥: 在与相同资源或共享对象的临界区有关的所有进程中，一次只允许一个进程进入临界区。一个在非临界区停止的进程不能干涉其他进程。
绝不允许出现需要访问临界区的进程被无限延迟的情况，即不会死锁或饥饿。
没有进程在临界区中时，任何需要进入临界区的进程必须能够立即进入。
对相关进程的执行速度和处理器的数量没有任何要求和限制。
一个进程驻留在临界区中的时间必须是有限的。
5.8 在信号量上可以执行什么操作?
为达到预期效果，可把信号量视为 - 一个值为整数的变量，整数值上定义了三个操作:
一个信号量可以初始化成非负数。
semWait 操作使信号量减 1。若值变成负数，则阻塞执行 semWait 的进程，否则进程继续执行。
semSignal 操作使信号量加 1。若值小于等于零，则被 semwait 操作阻塞的进程解除阻塞。
5.9 二元信号量和一般信号量有何区别?
二元信号量可以初始化为 0 或 1。
semWaitB 操作检查信号的值。若值为 0，则进程执行 semWaitB 就会受阻。若值为 1，则将值改为 0，并继续执行该进程。
semSignalB 操作检查是否有任何进程在该信号上受阻。若有进程受阻，则通过 semWaitB 操作，受阻的进程会被唤醒; 若没有进程受阻，则值设置为 1。
5.10 强信号量和弱信号量有何区别?
强信号量要求在信号量上等待的进程按照先进先出的规则从队列中移出。弱信号量没有此规则。
5.11 什么是管程?
管程是由一个或多个过程，一个初始化序列和局部数据组成的软件模块，其主要特定如下:
局部数据变量只能被管程的过程访问，任何外部过程都不能访问。
一个进程通过调用管程的一个过程进入管程。
在任何时候，只能有一个进程在管程中执行，调用管程的任何其他进程都被阻塞，以等待管程可用。
5.12 关于消息，阻塞和无阻塞有何区别?
发送者和接收者任一方阻塞则消息传递需要等待，都无阻塞则不需要等待。
5.13 与读者 / 写者问题相关的条件通常有哪些?
存在一个多个进程共享的数据区，该数据区可以是一个文件，一块内存等; 有些进程 (reader) 只能读取这个数据区中的数据，有些进程 (writer) 直往数据区中写数据。此外，还必须满足以下条件:
任意数量的读进程可以同时读这个文件。
一次只有一个写进程可以写文件。
若一次写进程正在写文件，则禁止任何读进程读文件。
第六章 并发: 死锁与饥饿 ¶
6.1 给出可重用资源和可消耗资源的例子。
可重用资源是指一次仅供 - 一个进程安全使用且不因使用而耗尽的资源。进程得到资源单元并使用后，会释放这些单元供其他进程再次使用。可重用资源的例子包括处理器、I/O 通道、内存和外存、设备，以及诸如文件、数据库和信号量之类的数据结构。
可消耗资源是指可被创建 (生产) 和销毁 (消耗) 的资源。某种类型可消耗资源的数量通常没有限制，无阻塞生产进程可以创建任意数量的这类资源。消费进程得到 - 一个资源时，该资源就不再存在。可消耗资源的例子有中断、信号、消息和 I/O 缓冲区中的信息。
6.2 产生死锁的三个必要条件是什么?
互斥。一次只有一个进程可以使用一个资源。其他进程不能访问已分配给其他进程的资源。
占有且等待。当一个进程等待其他进程时，继续占有已分配的资源。
不可抢占。不能强行抢占进程已占有的资源。
6.3 产生死锁的 4 个条件是什么?
循环等待。存在一个闭合的进程链，每个进程至少占有此链中下一个进程所需的一个资源。
6.4 如何防止占有且等待条件?
为预防占有且等待的条件，可以要求进程一次性地请求所有 需要的资源，并阻塞这个进程直到所有请求都同时满足。这种方法有两个方面的低效性。首先，一个进程可能被阻塞很长时间，以等待满足其所有的资源请求。而实际上，只要有一部分资源，它就可以继续执行。其次，分配给一个进程的资源可能会在相当长的一段时间不会被该进程使用，且不能被其他进程使用。另一个问题是一个进程可能事先并不知道它所需要的所有资源。
6.5 给出防止不可抢占条件的两种方法。
占有某些资源的一个进程进一步申请资源时若被拒绝，则该进程必须释放其最初占有的资源，必要时可再次申请这些资源和其他资源。
一个进程请求当前被另一个进程占有的一个资源时，操作系统可以抢占另一个进程，要求它释放资源。(只有在任意两个进程的优先级都不同时，这种方案才能预防死锁)。
6.6 如何防止循环等待条件?
循环等待条件可通过定义资源类型的线性顺序来预防。若一 个进程已分配了 R 类型的资源，则其接下来请求的资源只能是那些排在 R 类型之后的资源。为证明这种策略的正确性，我们给每种资源类型指定一个下标。 当 i<j 时，资源 R 排在资源 R，前面。现在假设两个进程 A 和 B 死锁，原因是 A 获得 R 并请求 R，而 B 获得 R，并请求 R，那么这个条件不可能，因为这意味着 i<j 且 j<i 。
6.7 死锁避免、 检测和预防之间的区别是什么?
处理死锁通常有三种方法: 预防、检测和避免。死锁预防通过确保不满足死锁的一个必要条件来避免发生死锁。操作系统总是同意资源请求时，需要进行死锁检测。操作系统必须周期性地检查死锁，并采取行动打破死锁。死锁避免涉及分析新的资源请求，以确定它是否会导致死锁，且仅当不可能发生死锁时才同意该请求。
第三部分 内存 ¶
第七章 内存管理 ¶
7.1 内存管理需要满足哪些需求? 重定位 保护 共享 逻辑组织 物理组织
7.2 为何需要重定位进程的能力?
在多道程序设计系统中，可用的内存空间通常被多个进程共享。通常情况下，程序员事先并不知道在某个程序执行期间会有其他哪些程序驻留在内存中。此外，我们还希望提供一个巨大的就绪进程池，以便把活动进程换入或换出内存，进而使处理器的利用率最大化。程序换出到磁盘中后，下次换入时要放到与换出前相同的内存区域会很困难。相反，我们需要把进程重定位 (relocate) 到内存的不同区域。
7.3 为何不可能在编译时实施内存保护?
由于程序在内存中的位置不可预测，因而在编译时不可能检查绝对地址来确保保护。此外，大多数程序设计语言允许在运行时进行地址的动态计算。因此必须在运行时检查进程产生的所有内存访问。
7.4 允许两个或多个进程访问内存某一特定区域的原因是什么?
任何保护机制都必须具有一定的灵活性，以允许多个进程访问内存的同一部分。例如，多个进程正在执行同一个程序时，允许每个进程访问该程序的同一个副本，要比让每个进程有自己单独的副本更有优势。合作完成同一个任务的进程可能需要共享访问相同的数据结构。因此，内存管理系统在不损害基本保护的前提下，必允许对内存共享区域进行受控访问。我们将会看到用于支持重定位的机制也支持共享。
7.5 在固定分区方案中，使用大小不等的分区有何好处?
可缓解因程序太大而无法放到固定大小的分区和因程序太小产生大量内部碎片的问题。
7.6 内部碎片和外部碎片有何区别?
内部碎片: 由于装入的数据块小于分区大小，因而导致分区内部存在空间浪费，这种现象称为内部碎片 (internal fragmentation)。
外部碎片: 随着时间的推移，内存中形成了越来越多的碎片，内存的利用率随之下降。这种现象称为外部碎片 (external fragmentation)，指在所有分区外的存储空间变成了越来越多的碎片，这与前面所讲的内部碎片正好对应。
7.7 逻辑地址、相对地址和物理地址有何区别?
逻辑地址 (logical address) 是指与当前数据在内存中的物理分配地址无关的访问地址，在执行对内存的访问之前必须把它转换为物理地址。
相对地址 (relative address) 是逻辑地址的一个特例，它是相对于某些已知点 (通常是程序的开始处) 的存储单元。
物理地址 (physical address) 或绝对地址是数据在内存中的实际位置。
7.8 页和页框有何区别?
页 (page): 一页指一系列的线性地址和包含于其中的数据。
页框 (pageframe): 分页单元认为所有的 RAM 被分成了固定长度的页框每个页框可以包含一页，也就是说一个页框的长度和一个页的长度是一样的。页框是内存的一部分，是一个实际的存储区域。页只是一组数据块，可以存放在任何页框中。 也就是说页对应的是线性 (虚拟) 地址的东西，而页框对应的是物理地址，是实际的存储区域。
7.9 页和段有何区别?
页大小相等，段可以大小不等; 分页对程序员来说是透明的，而分段通常是可见的。
第八章 虚拟内存 ¶
8.1 简单分页与虚拟内存分页有何区别?
进程运行时，简单分页的所有页必须都在内存中，除非使用了覆盖技术，虚存分页并非所有页都须在内存页框中，仅在需要时才读入页，把一页读入内存可能需要把另一页写出到磁盘。
8.2 什么是抖动?
当操作系统读取一块内存时，它必须把另一块换出。如果一块正好在将要用到之前换出，操作系统就不得不很快地把它取回。这类操作通常会导致一种称为系统抖动 (thrashing) 的情况。这样会使处理器的大部分时间都用于交换块而非执行指令。
8.3 为何在使用虚拟内存时，局部性原理至关重要?
局部性原理描述了一个进程中程序和数据引用的集簇倾向。因此，假设在很短的时间内仅需要进程的一部分块是合理的。同时，还可以对将来可能会访问的块进行猜测，从而避免系统抖动。局部性原理表明虚拟内存方案是可行的。
8.4 哪些元素是页表项中能找到的典型元素? 简单定义每个元素。
页号: 虚拟地址的页号部分。
进程标志符: 使用该页的进程。页号和进程标志符共同标志 - 个特定进程的虚拟地址空间的一页。
控制位: 该域包含一些标记， 比如有效、访问和修改，以及保护和锁定信息。
链指针: 若某项没有链项，则该域为空 (或用一个单独的位来表示)。否则，该域包含链中下一项的索引值 (0~2^m -1 之间的数字)。
8.5 转换检测缓冲区的目的是什么?
原则上，每次虚存访问都可能会引起两次物理内存访问: 一次取相应的页表项，另一次取需要的数据。因此，简单的虚拟内存方案会导致内存访问时间加倍。为克服这个问题，大多数虚拟内存方案都为页表项使用了一个特殊的高速缓存，通常称为转换检测缓冲区 (TranslationLookaside Buffer, TLB)。
8.6 简单定义两种可供选择的页面读取策略。
请求分页，只有当访问到某页中的一个单元时才将该页取入内存。若内存管理的其他策略比较合适，将发生下述情况: 当一个进程首次启动时，会在一段时间出现大量的缺页中断: 取入越来越多的页后，局部性原理表明大多数将来访问的页都是最近读取的页。因此，在一段时间后错误会逐渐减少，缺页中断的数量会降到很低。
预先分页，读取的页并不是缺页中断请求的页。预先分页利用了大多数辅存设备 (如磁盘) 的特性，这些设备有寻道时间和合理的延迟。若一 个进程的页连续存储在辅存中，则一次读取许多连续的页要比隔一段时间读取一 页有效。 当然，若大多数额外读取的页未引用到， 则这个策略是低效的。
8.7 驻留集管理和页面置换策略有何区别?
驻留集管理的概念为: (1) 给每个活动进程分配多少页框。 (2) 计划置换的页集是局限于那些缺页中断的进程，还是局限于所有页框都在内存中的进程。 置换策略的概念为: 在计划置换的页集中，选择换出哪一页。
8.8 FIFO 和时钟页面置换算法有何联系?
最简单的时钟策略需要给每个页框关联一个称为使用位的附加位。FIFO 和时钟页面置换算法都会扫描缓冲区，FIFO 直接置换出最老的页，而在时钟策略中会跳过使用位为 1 的页框。
8.9 页缓冲实现什么功能?
为提升性能，不丢弃置换出的页，被置换出的页任然驻留在内存中。因此，若进程访问该页，则可迅速返回该进程的驻留集，且代价很小。
8.10 为什么不能把全局置换策略和固定分配策略组合起来?
因为固定分配策略下，驻留集的大小是固定不变的，从内存中移出的一页必须由同一个进程的另一页置换。
8.11 驻留集和工作集有何区别?
驻留集表示进程在内存中的页集，工作集表示进程在过去的一段时间中被访问到的页集。
8.12 请求式清除和预约式清除有何区别?
请求式清除 (demand cleaning)，只有当一页被选择用于置换时才被写回辅存;
预约式清除 (precleaning) 策略则将这些已修改的多页在需要使用它们所占据的页框之前成批写回辅存。
第四部分 调度 ¶
第九章 单处理器调度 ¶
9.1 简要描述三种类型的处理器调度。
长程调度: 决定加入待执行进程池。
中程调度: 决定加入部分或全部位于内存中的进程集合。
短程调度: 决定可用 I/O 设备处理哪个进程挂起的 I/O 请求。
9.2 在交互式操作系统中，通常最重要的性能要求是什么?
响应时间
9.3 周转时间和响应时间有何区别?
周转时间指一个进程从提交到完成之间的时间间隔，包括实际执行时间和等待资源 (包括处理器资源) 的时间; 响应时间指从提交一个请求到开始接收响应之间的时间间隔。
9.4 对于进程调度，较小的优先级值是表示较低的优先级还是表示较高的优先级?
对于 UNIX 和许多其他操作系统中，优先级数值越大，表示的进程优先级越低。某些系统如 Windows 的用法正好相反，即大数值表示高优先级。
9.5 抢占式调度和非抢占式调度有何区别?
非抢占: 在这种情况下，一旦进程处于运行状态，就会不断执行直到终止，进程要么因为等待 I/O，要么因为请求某些操作系统服务而阻塞自己。
抢占: 当前正运行进程可能被操作系统中断，并转换为就绪态。一个新进程到达时，或中断发生后把一个阻塞态进程置为就绪态时，或出现周期性的时间中断时，需要进行抢占决策。
9.6 简单定义 FCFS 调度。
每个进程就绪后，会加入就绪队列。当前正运行的进程停止执行时，选择就绪队列中存在时间最长的进程运行。
9.7 简单定义轮转调度。
这种算法周期性地产生时钟中断，出现中断时，当前正运行的进程会放置到就绪队列中， 然后基于 FCFS 策略选择下一个就绪作业运行。
9.8 简单定义最短进程优先调度。
这是一个非抢占策略，其原则是下次选择预计处理时间最短的进程。
9.9 简单定义最短剩余时间调度。
最短剩余时间是在 SPN 中增加了抢占机制的策略。在这种情况下，调度程序总是选择预期剩余时间最短的进程。
9.10 简单定义最高响应比优先调度。
当前进程完成或被阻塞时，选择 R 值最大的就绪进程。调度决策基于对归一化周转时间的估计。
9.11 简单定义反馈调度。