《操作系统——精髓与设计原理（第八版）》复习题答案

第一部分 背景知识

第1章 计算机系统概述

1.1 列出并简要定义计算机的 4 个主要组成部分.

处理器：控制计算机的操作，执行数据处理功能。

内存：存储数据和程序。

输入/输出总线：在计算机和外部环境之间移动数据。

系统总线：在处理器、内存和输入输出模块间提供通信的设施。

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的操作是什么？

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？

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，作为最近存储地址的缓冲区。

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、数组及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。

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。

时间局部性是指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2章 操作系统概述

2.1 操作系统设计的三个目标？

方便：操作系统是计算机更容易使用。

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 什么是操作系统内核？

操作系统内核时计算机上最底层的软件，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）；

程序执行的上下文（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。进程切换过程包括保存当前进程的上下文和载入切换到的进程的上下文。

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、动态地改变模块的大小。

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。操作系统必须允许一部

分内存可以有各种用户以各种方式进行访问。

长期存储：许多应用程序需要在计算机关机后长时间保存信息。

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，及物理空间。

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（等待一个紫禁城或者一个 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。

内核单体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？

线程：可分派的工作单元。它包括处理器上下文环境（包含程序计数器和栈指针）和栈中自己的数据区域。线程顺序执行，并且是可中断的。

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，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性，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设计的所有问题。另外，操作系统还要充分利用迎接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3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 简要定义进程状态模型的每种状态。

运行态：该进程正在执行。

就绪态：进程做好了准备，只有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道可执行进程组中。

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？

交换是指把主存中某个进程的一部分或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 为什么有两个阻塞态？

进程是否阻塞、是否挂起。阻塞/挂起和就绪/挂起。

3.7 列出挂起状态进程的 4 个特点。

进程不能立即执行。

进程可能是或者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。

除非代理显式的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修改。而在内核模式下这没有这些限制，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；

给新进程分配空间；

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 暂时终止程序的执行转而处理这个新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。

陷阱指当异常或中断发生时，处理器捕捉到一个执行的线程，并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4章 线程

4.1 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，那些可能属于进程控制块？

对于不同的操作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。

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。

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。

处理器状态信息：这些信息通常只与进程有关。

4.2 请列出线程间的状态切换比进程间的状态切换开销更低的原因。

在已有进程中创建一个新线程的时间，远小于创建一个全新进程的时间。

终止线程要比终止进程所花的时间少。

统一进程内线程间切换的时间，要少于进程间切换的时间。

线程提高了不同程序间通信的效率。在多数操作系统中。独立进程间的通信需要内核介入，以提供保护和通信所需的机制。但是，由于同一个进程中的多个线程共享文件和内存，因此它们无需调用内核就可以互相通信。

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。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的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的执行路径，不同进程的执行过程会交替执行。因此，进程具有执行态和分配给其的优先级，是可以被操作系统分派的实体。

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？

进沉重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 列出用户级线程相对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的开销。

调用可以是应用程序专用的。一个应用程序可能倾想要简单的轮转调度算法，而另一个应用程序可能更适合于优先级调度算法。为了不依赖底层的操作系统调度程序，可以做到为应用程序量身定做调度算法。

用户级线程可以在任何操作系统下运行，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”技术。

把一个产生阻塞的系统调用转化为一个非阻塞的系统调用。

第5章并发性：互斥和同步

5.1 列出与并发相关的 4 个设计问题。

进程间通信、资源共享与竞争（如内存、文件、I/O 访问）、多个进程活动的同步以及给进程分配处理器时间。

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？

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。

进程间互相不知道对方的存在：这是一些独立的进程，它们不会一起工作。关于这种情况最好的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，或者两者的混合。尽管这些进程不会一起工作，但是操作系统需要知道它们对资源的竞争情况。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 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。

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)：一次只允许一个程序在临界区。

死锁（deadlock）：每个进程都在等待另一个资源，且获得其他资源并完成功能前，谁都不会释放自己拥有的资源。

饥饿（starvation）：非死锁情况下，某些进程被无限的拒绝访问资源。

5.7 列出对互斥的要求。

强制排他：在具有相同资源或共享对象的临界区的所有进程中，一次只允许一个进程进入临界区。

充分并发：一个在非临界区停止的进程必须不干涉其他进程。

空闲让进：没有进程在临界区中时，任何需要访问临界区的进程必须能够立即进入。

有限等待：绝不允许出现一个需要访问临界区的进程被无限延迟。

满足异步：相关进程的执行速度和处理机数目没有任何要求或限制。

让权等待：当进程不能进入临界区，应当立即释放处理机，防止进程忙等待。

5.8 在信号量上可以执行什么操作？

一个信号量可以初始化成非负数。

semWait 操作使信号量减 1。若值变为负数，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 0 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？

二元信号量的值只能是 0 或 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冲队列中移出。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，其主要特点如下：

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？

发送者和接收者任何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？

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。

若一个写进程正在写文件，则禁止任何读进程读文件。

第6章 并发：死锁和饥饿

6.1 给出可重用资源和可消耗资源的例子。

可重用资源的例子包括处理器、I/O 通道，内存和外存、设备，以及文件、数据库和信号量之类的数据结构；

可消耗资源的例子有中断、信号、消息和 I/O 缓冲区。

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有的资源。

6.3 产生死锁的 4 个条件是什么？

前三个条件都只是死锁的必要条件而非充分条件。要产生死锁还需第四个条件：

循环等待：存在一个闭合的进程链，每个进程至少占由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？

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。

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。

6.6 如何防止循环等待条件？

通过定义资源类型的线性顺序来预防。

6.7 死锁避免、检测和预防之间的区别是什么？

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7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，进而使处理器地利用率最大化。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？

程序在内存中的位置不可预测。

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？

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。

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而产生大量内部碎片问题。

7.6 内部碎片和外部碎片有何区别？

内部碎片：固定分区中，由于装入的数据块小于分区大小，导致分区内存在空间浪费。

外部碎片：动态分区中，指所在分区外的存储空间变成了越来越多的碎片。

7.7 逻辑地址、相对地址和物理地址有何区别？

逻辑地址：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：是逻辑地址的一个特例，它是相对于某些已知点的存储单元。

物理地址：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。

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。通常段比页大。逻辑地址表示分页是一维的、分段是二维的。

第8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，简单分页所有页必须都在内存中，除非使用了覆盖技术，虚拟分页并非所有页都必须在内存页框中，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？

处理器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。页号和进程标识符共同标志一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。否则，该域包含链中下一项的索引值。

8.5 转换检测缓冲区的目的是什么?

为了克服简单的虚拟内存方案导致内存访问时间加倍的问题。原则上，每次虚存访问都可能会引起超两次物理内存的访问：一次取相应的页表项，另一次取需要的数据。

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。

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的特性，这些设备有寻道时间和合理的延迟。

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：

给每个活动进程分配多少个页框。

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 和时钟页面置换策略有何区别？

时钟策略需要给每个页框关联一个称为使用位的附加位，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？

被置换的页仍然留在内存中。

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？

为了保持驻留集的大小固定，从内存中移出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：只有当一页被选择用于置换时才被写回辅存

预约式清除：将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9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受相应之间的时间间隔。

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。

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？

抢占式调度：当前正运行进程可能被操作系统中断，并转换为就绪态。一个新的进程到达时，或中断发生后把一个阻塞态进程置为就绪态时，或出现周期性的时钟中断时，需要进行抢占决策。

非抢占式调度：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。

9.6 简单定义 FCFS 调度。

每个进程就绪后，会加入就绪队列。当前运行的进程停止执行，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。

9.11 简单定义反馈调度。

调度基于抢占原则并使用动态优先级机制。

第10章 多处理器调度和实时调度

10.1 列出并简单定义 5 种不同级别的同步粒度。

细粒度：单指令流中固有的并行。

中等粒度：一个单独应用中的并行处理或多任务处理。

粗粒度：多道程序环境中并发进程的多处理。

极粗粒度：在网络节点上进行分布式处理，形成一个计算环境

无约束：多个无关进程

10.2 列出并简单定义线程调度的 4 种技术。

先来先服务（FCFS）：一个作业到达时，其所有线程都被连续地放在共享队列末尾。一个处理器变得空闲时，会选择下一个就绪线程执行，直到完成或被阻塞。

最少线程数优先：共享就绪队列被组织成一个优先级队列，一个作业包含的未调度线程的数量最少时，给它指定最高的优先级。具有相同优先级的队列按照作业到达的顺序排队。和 FCFS 一样，被调度的线程一直运行到完成或被阻塞。

可抢占的最少线程数优先：最高优先级给予具有最少未被调用线程数最少的作业。若刚到大的作业所包含的线程数少于正在执行作业的线程数，它将抢占属于这个被调度作业的线程。

10.4 硬实时任务和软实时任务有何区别？

硬实时任务：是指必须满足最后期限的任务，否则会给系统带来不可接受的破坏或致命的错误。

软实时任务：也有一个与之相连的最后期限，且希望能满足这一期限的要求，但不强制，即使超过了最后期限，调度和完成这个任务仍是有意义的。

10.5 周期性实时任务和非周期性实时任务有何区别？

非周期性实时任务有一个必须结束或开始的最后期限，或者有一个关于开始时间和结束时间的约束条件。

周期性实时任务：这个要求可以描述为“每隔周期 T 一次”或“每隔 T 个单位”。

10.6 列出并简单定义实时操作系统的 5 方面要求。

可确定性：某种程度上是指它可以按照固定的、预先确定的时间或时间间隔执行操作。

可响应性：在知道中断后，操作系统为中断提供服务时间。

用户控制：允许用户细粒度的控制任务优先级。

可靠性：避免性能的损失和降低。

故障弱化操作：系统在故障时尽可能多的去保存其性能和数据的能力。

10.7 列出并简单定义 4 类实时调度算法。

静态表调度算法：执行关于可行调度的静态分析。分析的结果是一个调度，它确定在运行时一个任务何时需开始执行。

静态优先级抢占调度法：执行一个静态分析，但未制定调度，而是通过给任务指定优先级，使得可以使用传统的基于优先级的抢占式调度程序。

基于动态规划的调度法：在运行时动态的确定可行性，而不是在开始运行前离线地确定。到达地任务仅能在满足其他时间约束时，才可接受并执行。可行性分析的结果是一个调度或规划，可用于确定何时分派这个任务。

动态尽力调度法：不执行可行性分析。系统试图满足所有地最后期限，并终止任何已经开始但错过最后期限的进程。

10.8 一个任务的哪些信息在实时调度时非常有用？

就绪时间、启动最后期限、完成最后期限、处理时间、资源需求、优先级、子任务结构。

第五部分 输入输出和文件

第11章 I/O管理和磁盘调度

11.1 列出并简单定义执行I/O的三种技术。

程序I/O：处理器代表一个进程给I/O模块发送一个I/O命令；该进程进入忙等待，直到操作完成才能继续执行。

中断驱动I/O：处理器代表进程向I/O模块发出一个I/O命令。有两种可能性：若来自进程的I/O指令是非阻塞的，则处理器继续执行发出I/O命令的进程的后续指令。若I/O指令是阻塞的，则处理器执行的下一条指令来自操作系统，它将当前的进程设置为阻塞态并调度其他进程。

直接存储器访问（DMA）：一个DMA模块控制内存和I/O模块之间的数据交换。为传送一块数据，处理器给DMA模块发出请求，且只有在整个数据块传送结束后，它才被中断。

11.2 逻辑I/O和设备I/O有何区别？

逻辑I/O：逻辑I/O模块把设备当作一个逻辑资源来处理，它并不关心实际控制设备的细节。逻辑I/O模块代表用户进程管理的普通I/O功能，允许用户进程根据设备标识符及诸如打开、关闭、读、写之类的简单指令与设备打交道。

设备I/O：请求的操作和数据被转换为适当的I/O指令序列、通道命令和控制器指令。可以使用缓冲技术来提高利用率。

11.3 面向块的设备和面向流的设备有何区别？各举一些例子。

面向块的设备将信息保存在块中，块的大小通常是固定的，传送过程中一次传送一块。通常可以通过块号访问数据。磁盘和USB智能卡都是面向块的设备。

面向流的设备以字节流的的方式输入/输出数据，它没有块结构。终端、打印机、通信端口、鼠标和其他指示设备以及其他大多数非辅存设备，都属于面向流的设备。

11.4 为什么希望用双缓冲而非单缓冲来提高I/O的性能？

对于面向块的传送，我们可以粗略地估计执行时间为max[C,T]。因此，若C小于等于T，则有可能使面向块的设备全速运行；另一方面，若C大于T，则双缓冲能确保该进程不需要等待I/O。

11.5 在磁盘读或写时有哪些延迟因素？

寻道时间

旋转延迟

存取时间

传输时间

11.6 简单定义各种磁盘调度策略。

先进先出：按顺序处理队列中的项目。

最短服务时间优先：选择使磁头臂从当前位置开始移动最少的磁盘 I/O 请求。

SCAN：要求磁头臂仅沿一个方向移动，并在途中满足所有未完成的请求，直到它到达这个方向上的最后一个磁道。

C-SCAN：把扫描限定在一个方向上。因此，当访问到沿某个方向的最后一个磁道时，磁头臂返回到磁盘相反方向末端的磁道。

11.7 简单定义 7 个 RAID 级别。

RAID 0：条带化，非冗余；

RAID 1：镜像、被镜像；

RAID 2：并行访问、通过汉明码实现冗余；

RAID 3：并行访问、交错位奇偶校验；

RAID 4：独立访问、交错块奇偶校验；

RAID 5：独立访问、交错块分布奇偶校验；

RAID 6：独立访问、交错块双重分布奇偶校验。

11.8 典型的磁盘扇区大小是多少？

512 字节。

第12章 文件管理

12.1 域和记录有何不同？

域（field）是基本的数据单元。一个域包含了一个值，如雇员的名字、日期或传感器读取的值。域可通过其长度和数据类型来描述。域的长度可以是定长的或变长的，具体取决于文件的设计。对于后一种情况，域通常包含两个或三个子域：要保存的实际值、域名，以及某些情况下的域长度。在其他情况下，域之间特殊的分隔符暗示了域的长度。

记录（record）是一组相关域的集合，可以视为应用程序的一个单元。例如，一条雇员记录可能包含以下域：名字、社会保险号、工作类型、雇用日期等。同样，记录的长度可以是定长的或变长的，具体取决于设计。如果一条记录中的某些域是变长的，或记录中域的数量可变，则该记录是变长。对于后一种情况，每个域通常都有一个域名。对于这两种情况，整条记录通常都包含一个长度域。

12.2 文件和数据库有何不同？

文件是一组相似记录的集合，它被用户和应用程序视为一个实体，可以通过名字访问。文件有唯一的一个文件名，可被创建或删除。

数据库是一组相关数据的集合，其本质特征是数据元素间存在着明确的关系，且可供不同的应用程序使用。数据库自身由一种或多种类型的文件组成。通常，数据库管理系统是独立于操作系统的，尽管它可能会使用某些文件管理程序。

12.3 什么是文件管理系统？

文件管理系统是一组系统软件，它为使用文件的用户和应用程序提供服务。典型情况下，文件管理系统是用户或应用程序访问文件的唯一方式。它可使得用户或程序员不需要为每个应用程序开发专用软件，并为系统提供控制最重要资源的方法。

12.4 选择文件组织时的重要原则是什么？

快速访问
易于修改
节约存储空间
维护简单
可靠性

12.5 列出并简单定义 5 种文件组织。

堆：是最简单的文件组织形式。数据按它们到达的顺序被收集，每条记录由一串数据组成。

顺序文件：是最常用的文件组织形式。每条记录都使用一种固定的格式。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的顺序组成。

索引顺序文件：保留了顺序文件的关键特征（按照关键域的顺序组织）。增加了两个特征:用于支持随机访问的文件索引和溢出文件。溢出文件类似于顺序文件中使用的日志文件，但溢出文件中的记录可根据它前面记录的指针进行定位。

索引文件：只能通过索引来访问记录。

直接文件或散列文件：开发直接访问磁盘中任何一个地址已知的块的能力。

12.6 为何在索引顺序文件中查找一条记录的平均时间小于在顺序文件中的平均时间？

索引提供了快速接近目标记录的查找能力。

12.7 对目录执行的典型操作有哪些？

查找、创建文件、删除文件、显示目录、修改目录。

12.8 路径名和工作目录有何关系？

一系列目录名和最后到达的文件名组成了该文件的路径名。对于交互用户或进程而言，总有一个当前路径与之相关联，通常称为工作目录。

12.9 可以授予或拒绝的某个特定用户对某个特定文件的访问权限通常有哪些？

无（none）——不知道文件的存在
知道（knowledge）——知道文件的存在及属主
执行（execution）——可加载并执行但不能复制
读（read）—— 查看/复制内容
追加（append）——（在文件尾）添加数据
更新（update）——修改/删除/添加数据
改变保护（change protection）——（文件所有者）修改授予其他用户的访问权限
删除（deletion）——从文件系统中去掉

12.10 列出并简单定义三种组块方法。

定长组块：使用定长的记录，且若干完整的记录保存在一个块中。在每个块的末尾可能会有一些未使用的空间，称为内部碎片。

变长跨域式组块：使用变长的记录，并紧缩到块中，使得块中不存在未使用的空间。因此，某些记录可能会跨越两个块，两个块通过指向一个后续块的指针连接。

变长非跨越式组块：使用变长记录，但并不采用跨域方式。若下一条记录比块中剩余的未使用空间大，则无法使用这一部分，因此在大多数块中都会有未使用的空间。

12.11 列出并简单定义三种文件分配方法。

连续分配：指在创建文件时，给文件分配一组连续的块。

链式分配：基于单个块，链中的每块都包含指向下一块的指针。

索引分配：每个文件在文件分配表中都有一个索引。分配给该文件的每个分区在索引中都有一个表项。