操作系统精髓与设计原理课后答案

第 1 章　　计算机系统概述

1.1 列出并简要地定义计算机的四个主要组成部分。

主存储器，存储数据和程序；算术逻辑单元，能处理二进制数据；控制单元，解读存储器中的指令并且使他们得到执行；输入/输出设备，由控制单元管理。

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：其他模块（I/O，存储器）中断处理器正常处理过程的机制。

1.5 多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 内存层次的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。

1.7 什么是高速缓冲存储器？

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 列出并简要地定义 I/O 操作的三种技术。

可编程 I/O：当处理器正在执行程序并遇到与 I/O 相关的指令时，它给相应的 I/O 模块发布命令（用以执行这个指令）；在进一步的动作之前，处理器处于繁忙的等待中，直到该操作已经完成。

中断驱动 I/O：当处理器正在执行程序并遇到与 I/O 相关的指令时，它给相应的 I/O 模块发布命令，并继续执行后续指令，直到后者完成，它将被 I/O 模块中断。如果它对于进程等待 I/O 的完成来说是不必要的，可能是由于后续指令处于相同的进程中。否则，此进程在中断之前将被挂起，其他工作将被执行。

直接存储访问：DMA 模块控制主存与 I/O 模块间的数据交换。处理器向 DMA 模块发送一个传送数据块的请求，（处理器）只有当整个数据块传送完毕后才会被中断。

1.9 空间局部性和临时局部性间的区别是什么？

空间局部性是指最近被访问的元素的周围的元素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 2 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？

内核是操作系统最常使用的部分，它存在于主存中并在特权模式下运行，响应进程调度和设备中断。

2.3 什么是多道程序设计？

多道程序设计是一种处理操作，它在两个或多个程序间交错处理每个进程。

2.4 什么是进程？

进程是一个正在执行的程序，它被操作系统控制和选择。

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，防止互相干涉数据和存储空间。

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。

2.7 解释实地址和虚地址的区别。

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。

实地址指的是主存中的地址。

2.8 描述轮循调度技术。

轮循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成可以同时运行的多个线程。

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 通常那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。（表 3.1）

3.3 对于图 3.6 中的进程模型，请简单定义每个状态。

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组中释放出的进程，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 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 列出挂起态进程的 4 个特点。

1.进程不能立即执行。

2.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。

3.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。

4.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？

1.给新进程分配一个唯一的进程标识号。2.给进程分配空间。3.初始化进程控制块。4.设置正确的连接。5.创建或扩充其他的数据结构。

3.12 中断和陷阱有什么区别？

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。

时钟终端，I/O 终端，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、对称多处理和微内核

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，这些元素中那些可能属于线程控制块，那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。处理器状态信息：这些信息通常只与进程有关。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。

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？

资源所有权和调度/执行。

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，加速执行和模块化程序结构。

4.5 哪些资源通常被一个进程中的所有线程共享？

例如地址空间，文件资源，执行特权等。

4.6 列出用户级线程优于内核级线程的三个优点。

1.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。

2.调用可以是应用程序专用的。一个应用程序可能倾向于简单的轮询调度算法，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。

3.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。

1.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。

2.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 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。

4.9 简单定义图 4.8 中列出的各种结构。

SIMD：一个机器指令控制许多处理部件步伐一致地同时执行。每个处理部件都有一个相关的数据存储空间，因此，每条指令由不同的处理器在不同的数据集合上执行。

MIMD：一组处理器同时在不同的数据集上执行不同的指令序列。主/从：操作系统内核总是在某个特定的处理器上运行，其他处理器只用于执行用户程序，还可能执行一些操作系统实用程序。

SMP：内核可以在任何处理器上执行，并且通常是每个处理器从可用的进程或线程池中进行各自的调度工作。集群：每个处理器都有一个专用存储器，而且每个处理部件都是一个独立的计算机。

4.10 列出 SMP 操作系统的主要设计问题。

同时的并发进程或线程，调度，同步，存储器管理，可靠性和容错。

4.11 给出在典型的单体结构操作系统中可以找到且可能是微内核操作系统外部子系统中的服务和功能。

设备驱动程序，文件系统，虚存管理程序，窗口系统和安全服务。

4.12 列出并简单解释微内核设计相对于整体式设计的七个优点。

一致接口：进程不需要区分是内核级服务还是用户级服务，因为所有服务都是通过消息传递提供的。

可扩展性：允许增加新的服务以及在同一个功能区域中提供多个服务。

灵活性：不仅可以在操作系统中增加新功能，还可以删减现有的功能，以产生一个更小、更有效的实现。

可移植性：所有或者至少大部分处理器专用代码都在微内核中。因此，当把系统移植到一个处理器上时只需要很少的变化，而且易于进行逻辑上的归类。

可靠性：小的微内核可以被严格地测试，它使用少量的应用程序编程接口（API），这就为内核外部的操作系统服务产生高质量的代码提供了机会。

分布式系统支持：微内核通信中消息的方向性决定了它对分布式系统的支持。

面向对象操作系统环境：在微内核设计和操作系统模块化扩展的开发中都可以借助面向对象方法的原理。

4.13 解释微内核操作系统可能存在的性能缺点。

通过微内核构造和发送信息、接受应答并解码所花费的时间比一次系统调用的时间要多。

4.14 列出即使在最小的微内核操作系统中也可以找到的三个功能。

低级存储器管理，进程间通信（IPC）以及 I/O 和中断管理。

4.15 在微内核操作系统中，进程或线程间通信的基本形式是什么？

消息。

第 5 章　并发性：互斥和同步

5.1 列出与并发相关的四种设计问题

进程间的交互，共享资源之间的竞争，多个进程的同步问题，对进程的处理器时间分配问题

5.2 列出并发的三种上下文

多个应用程序，结构化应用程序，操作系统结构

5.3 执行并发进程的最基本要求是什么？

加强互斥的能力

5.4 列出进程间的三种互相知道的程度，并简单地给出各自的定义。

进程间互相不知道对方：这是一些独立的进程，他们不会一起工作。

进程间间接知道对方：这些进程并不需要知道对方的进程 ID 号，但他们共享访问某些对象，如一个 I/O 缓冲区。

进程间直接知道对方：这些进程可以通过进程 ID 号互相通信，用于合作完成某些活动。

5.5 竞争进程和合作进程进程间有什么区别。

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种控制问题，并简单地给出各自的定义。

互斥：竞争进程仅可以访问一个临界资源（一次仅有一个进程可以访问临界资源），并发机制必须满足一次只有一个进程可以访问临界资源这个规则。

死锁：如果竞争进程需要唯一的访问多于一个资源，并且当一个进程控制着一个进程，且在等待另一个进程，死锁可能发生。

饥饿：一组进程的一个可能会无限期地拒绝进入到一个需要资源，因为其他
成员组成垄断这个资源。

5.7 列出对互斥的要求。

1.必须强制实施互斥：在具有关于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。

2.一个在临界区停止的进程必须不干涉其他进程。

3.绝不允许出现一个需要访问临界区的进程被无限延迟的情况，即不会饿死或饥饿。

4.当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。

5.对相关进程的速度和处理器的数目没有任何要求和限制。

6.一个进程驻留在临界区中的时间是有限的。

5.8 在信号量上可以执行什么操作。

1.一个信号量可以初始化成非负数。

2.wait 操作使信号量减 1，如果值为负数，那么进程执行 wait 就会受阻。3signal 操作使信号量增加 1，如果小于或等于 0，则被 wait 操作阻塞的进程被解除阻塞。

5.9 二元信号量与一般信号量有什么区别。

二元信号量只能取 0 或 1，而一般信号量可以取任何整数。

5.10 强信号量与弱信号量有什么区别。

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。

5.12 对于消息，有阻塞和无阻塞有什么区别？

发送者和接收者任一方阻塞则消息传递需要等待，都无阻塞则不需等待。

5.13 通常与读者-写者问题相关联的有哪些条件？

1.任意多的读进程可以同时读这个文件

2.一次只有一个写进程可以往文件中写

3.如果一个写进程正在往文件中写时，则禁止任何读进程读文件。

第 6 章　　并发性：死锁和饥饿

6.1 给出可重用资源和可消费资源的例子。

可重用资源：处理器，Ⅰ/O通道，主存和辅存，设备以及诸如文件，数据库和信号量之类的数据结构。

可消费资源：中断，信号，消息和Ⅰ/O缓冲区中的信息。

6.2 可能发生死锁所必须的三个条件是什么？

互斥，占有且等待，非抢占。

6.3 产生死锁的第 4 个条件是什么？

循环等待。

6.4 如何防止占有且等待的条件？

可以要求进程一次性地请求所有需要的资源，并且阻塞这个资源直到所有请求都同时满足。

6.5 给出防止无抢占条件的两种方法。

第一种，如果占有某些资源的一个进程进行进一步资源请求被拒绝，则该进程必须释放它最初占用的资源，如果有必要，可再次请求这些资源和另外的资源。

第二种，如果一个进程请求当前被另一个进程占有的一个资源，则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？

可以通过定义资源类型的线性顺序来预防。如果一个进程已经分配到了R类型的资源，那么它接下来请求的资源只能是那些排在R类型之后的资源类型。

6.7 死锁避免，检测和预防之间的区别是什么？

死锁预防是通过间接地限制三种死锁必要条件的至少一个或是直接地限制循环等待的发生来避免死锁的出现。死锁避免允许可能出现的必要条件发生，但是采取措施确保不会出现死锁的情况。而死锁检测允许资源的自由分配，采取周期性的措施来发现并处理可能存在的死锁情况。

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、逻辑组织和物理组织。

7.2 为什么需要重定位进程的能力？

通常情况下，并不能事先知道在某个程序执行期间会有哪个程序驻留在主存中。此外还希望通过提供一个巨大的就绪进程池，能够把活动进程换入和换出主存，以便使处理器的利用率最大化。在这两种情况下，进程在主存中的确切位置是不可预知的。

7.3 为什么不可能在编译时实施内存保护？

由于程序在主存中的位置是不可预测的，因而在编译时不可能检查绝对地址来确保保护。并且，大多数程序设计语言允许在运行时进行地址的动态计算（例如，通过计算数组下标或数据结构中的指针）。因此，必须在运行时检查进程产生的所有存储器访问，以便确保它们只访问了分配给该进程的存储空

间。

7.4 允许两个或多个进程访问进程的某一特定区域的原因是什么？

如果许多进程正在执行同一程序，则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。同样，合作完成同一任务的进程可能需要共享访问同一个数据结构。

7.5 在固定分区方案中，使用大小不等的分区有什么好处？

通过使用大小不等的固定分区：1.可以在提供很多分区的同时提供一到两个非常大的分区。大的分区允许将很大的进程全部载入主存中。2.由于小的进程可以被放入小的分区中，从而减少了内部碎片。

7.6 内部碎片和外部碎片有什么区别？

内部碎片是指由于被装入的数据块小于分区大小而导致的分区内部所浪费的空间。外部碎片是与动态分区相关的一种现象，它是指在所有分区外的存储空间会变成越来越多的碎片的。

7.7 逻辑地址、相对地址和物理地址间有什么区别？

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化成物理地址。相对地址是逻辑地址的一个特例，是相对于某些已知点（通常是程序的开始处）的存储单元。物理地址或绝对地址是数据在主存中的实际位置。

7.8 页和帧之间有什么区别？

在分页系统中，进程和磁盘上存储的数据被分成大小固定相等的小块，叫做页。而主存被分成了同样大小的小块，叫做帧。一页恰好可以被装入一帧中。

7.9 页和段之间有什么区别？

分段是细分用户程序的另一种可选方案。采用分段技术，程序和相关的数据被划分成一组段。尽管有一个最大段长度，但并不需要所有的程序的所有段的长度都相等。

第 8 章　　虚拟内存

8.1 简单分页与虚拟分页有什么区别？

简单分页：一个程序中的所有的页都必须在主存储器中程序才能正常运行，除非使用覆盖技术。

拟内存分页：不是程序的每一页都必须在主存储器的帧中来使程序运行，页在需要的时候进行读取。

8.2 解释什么是抖动。

虚拟内存结构的震动现象，在这个过程中处理器大部分的时间都用于交换块，而不是执行指令。

8.3 为什么在使用虚拟内存时，局部性原理是至关重要的？

可以根据局部性原理设计算法来避免抖动。总的来说，局部性原理允许算法预测哪一个当前页在最近的未来是最少可能被使用的，并由此就决定候选的替换出的页。

8.4 哪些元素是页表项中可以找到的元素？简单定义每个元素。

帧号：用来表示主存中的页来按顺序排列的号码。

存在位（P）：表示这一页是否当前在主存中。

修改位（M）：表示这一页在放进主存后是否被修改过。

8.5 转移后备缓冲器的目的是什么？

转移后备缓冲器（TLB）是一个包含最近经常被使用过的页表项的高速缓冲存储器。它的目的是为了减少从磁盘中恢复一个页表项所需的时间。

8.6 简单定义两种可供选择的页读取策略。

在请求式分页中，只有当访问到某页中的一个单元时才将该页取入主存。

在预约式分页中，读取的并不是页错误请求的页。

8.7 驻留集管理和页替换策略有什么区别？

驻留集管理主要关注以下两个问题：（1）给每个活动进程分配多少个页帧。（2）被考虑替换的页集是仅限在引起页错误的进程的驻留集中选择还是在主存中所有的页帧中选择。

页替换策略关注的是以下问题：在考虑的页集中，哪一个特殊的页应该被选择替换。

8.8 FIFO 和 Clock 页替换算法有什么区别？

时钟算法与 FIFO 算法很接近，除了在时钟算法中，任何一个使用位为一的页被忽略。

8.9 页缓冲实现的是什么？

（1）被替换出驻留集的页不久又被访问到时，仍在主存中，减少了一次磁盘读写。

（2）被修改的页以簇的方式被写回，而不是一次只写一个，这就大大减少了 I/O 操作的数目，从而减少了磁盘访问的时间。

8.10 为什么不可能把全局替换策略和固定分配策略组合起来？

固定分配策略要求分配给一个进程的帧的数目是确定的，当一个进程中取入一个新的页时，这个进程驻留页集中的一页必须被替换出来（保持分配的帧的数目不变），这是一种局部替换策略。

8.11 驻留集和工作集有什么区别？

一个进程的驻留集是指当前在主存中的这个进程的页的个数。一个进程的工作集是指这个进程最近被使用过的页的个数。

8.12 请求式清除和预约式清除有什么区别？

在请求式清除中，只有当一页被选择用于替换时才被写回辅存；

在预约式清除中，将这些被修改的多个页在需要用到它们所占据的页帧之前成批的写回辅存。

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：决定加入到待执行的进程池中；

中程调度：决定加入到部分或全部在主存中的进程集合中；

短程调度：决定哪一个可用进程将被处理器执行。

9.2 在交互式操作系统中，通常最重要的性能要求是什么？

反应时间

9.3 周转时间和响应时间有什么区别？

周转时间是一个要求花费在系统上的包括等待时间和服务时间的总的时间。响应时间对一个交互进程，这是指从提交一个请求到开始接受响应之间的时间间隔。通常进程在处理该请求的同时，就开始给用户产生一些输出。

9.4 对进程调度，较小的优先级值表示较低的优先级还是较高的优先级？

在 UNIX 和许多其他系统中，大的优先级值表示低优先级进程。许多系统，比如 WINDOWS，刚好相反，大数值表示高优先级。

9.5 抢占式和非抢占式调度有什么区别？

非抢占：在这种情况下，一旦进程处于运行态，他就不断执行直到终止，或者为等待 I/O 或请求某些操作系统服务而阻塞自己。

抢占：当前正在运行的进程可能被操作系统中断，并转移到就绪态。关于抢占的决策可能是在一个新进程到达时，或者在一个中断发生后把一个被阻塞的进程置为就绪态时，或者基于周期性的时间中断。

9.6 简单定义 FCFS 调度。

当每个进程就绪后，它加入就绪队列。当当前正在运行的进程停止执行时，选择在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

以一个周期性间隔产生时钟中断，当中断产生时，当前正在运行的的进程被置于就绪队列中，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。

这是一个非抢占的策略，其原则是下一次选择所需处理时间最短的进程。

9.9 简单定义最短剩余时间调度。

最短剩余时间是针对 SPN 增加了抢占机制的版本。在这种情况下，调度器总是选择预期剩余时间最短的进程。当一个新进程加入到就绪队列时，他可能比当前运行的进程具有更短的剩余时间，因此，只

有新进程就绪，调度器就可能抢占当前正在运行的进程。

9.10 简单定义最高响应比优先调度。

在当前进程完成或被阻塞时，选择 R 值最大的就绪进程。R=(w+s)/s,w 等待处理器的时间，s 期待的服务时间。

9.1 1 简单定义反馈调度。

调度基于抢占原则并且使用动态优先级机制。当一个进程第一次进入系统时，它被放置在 RQ0。当它第一次被抢占后并返回就绪状态时，它被防止在 RQ1。在随后的时间里，每当它被抢占时，它被降级到下一个低优先级队列中。一个短进程很快会执行完，不会在就绪队列中降很多级。一个长进程会逐级下降。因此，新到的进程和短进程优先于老进程和长进程。在每个队列中，除了在优先级最低的队列中，都使用简单的 FCFS 机制。一旦一个进程处于优先级最低的队列中，它就不可能再降低，但是会重复地返回该队列，直到运行结束。

第 10 章　　多处理器和实时调度

10.1 列出并简单定义五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：在一个单独应用中的并行处理或多任务处理；

粗粒度：在多道程序环境中并发进程的多处理；

非常粗粒度：在网络节点上进行分布处理，以形成一个计算环境；

无约束粒度：多个无关进程。

10.2 列出并简单定义线程调度的四种技术。

加载共享：进程不是分配到一个特定的处理器，而是维护一个就绪进程的全局队列，每个处理器只要空闲就从队列中选择一个线程。这里使用术语加载共享来区分这种策略和加载平衡方案，加载平衡是基于一种比较永久的分配方案分配工作的。

组调度：一组相关的线程基于一对一的原则，同时调度到一组处理器上运行。

专用处理器分配：在程序执行过程中，每个程序被分配给一组处理器，处理器的数目与程序中的线程的数目相等。当程序终止是，处理器返回到总的处理器池中，可供分配给另一个程序。

动态调度：在执行期间，进程中线程的数目可以改变。

10.3 列出并简单定义三种版本的负载分配。

先来先服务（FCFS）：当一个作业到达时，它的所有线程都被连续地放置在共享队列末尾。当一个处理器变得空闲时，它选择下一个就绪线程执行，直到完成或阻塞。

最少线程数优先：共享就绪队列被组织成一个优先级队列，如果一个作业包含的未调度线程数目最少，则给它指定最高的优先级。具有同等优先级的队列按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阻塞。

可抢占的最少线程数优先：最高的的优先级给予包含的未被调度的线程数目最少的作业。刚到达的作业如果包含的线程数目少于正在执行的作业，它将抢占属于这个被调度作业的线程。

10. 硬实时任务和软实时任务有什么区别？

硬实时任务指必须满足最后期限的限制，否则会给系统带来不可接受的破坏或者致命的错误。

软实时任务也有一个与之相关联的最后期限，并希望能满足这个期限的要求，但是这并不是强制的，即使超过了最后期限，调度和完成这个任务仍然是有意义的。

10.5 周期性实时任务和非周期性实时任务有什么区别？

非周期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。而对于周期任务，这个要求描述成“每隔周期 T 一次”或“每隔 T 个单位”。

10.6 列出并简单定义对实时操作系统的五方面的要求。

可确定性：在某中程度上是指它可以按固定的、预先确定的时间或时间间隔执行操作。

可响应性：它关注的是在知道中断之后操作系统未中断提供服务的时间

用户控制：用户应该能够区分硬实时任务和软实时任务，并且在每一类中确定相对优先级。实时系统还允许用户指定一些特性，例如使用分页还是进程交换、哪一个进程必须常驻主存、使用何种磁盘算法、不同的优先级的进程各有哪些权限等。

可靠性 ：可靠性必须提供这样一种方式，以继续满足实时最后期限。

故障弱化操作：故障弱化操作指系统在故障时尽可能多的保存其性能和数据的能力。

10.7 列出并简单定义四类实时调度算法。

静态表驱动法：执行关于可行调度的静态分析。分析的结果是一个调度，它用于确定在运行时一个任务何时必须开始执行。

静态优先级驱动抢占法：同样，执行一个静态分析，但是没有制定调度，而且用于给任务指定优先级，使得可以使用传统的优先级驱动的抢占式调度器。

基于动态规划调度法：在运行是动态地确定可行性，而不是在开始运行前离线的确定（静态）。一个到达的任务，只有当能够满足它的时间约束时，才可以被接受执行。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。系统试图满足所有的最后期限，并终止任何已经开始运行但错过最后期限的进程。

10.8 关于一个任务的哪些信息在实时调度是非常有用？

就绪时间：任务开始准备执行的时间。对于重复或周期性的任务，这实际上是一个事先知道的时间序列。而对于非周期性的任务，或者也事先知道这个时间，或者操作系统仅仅知道什么时候任务真正就绪。

启动最后期限：任务必须开始的时间。

完成最后期限：任务必须完成的时间。典型的实时应用程序或者有启动最后期限，或者有完成最后期限，但不会两者都存在。

处理时间：从执行任务直到完成任务所需的时间。在某些情况下，可以提供这个时间，而在另外一些情况下，操作系统度量指数平均值。其他调度系统没有使用这个信息。

资源需求：任务在执行过程中所需的资源集合（处理器以外的资源）。

优先级：度量任务的相对重要性。硬实时任务可能具有绝对的优先级，因为如果错过最后期限则会导致系统失败。如果系统无论如何也要继续运行，则硬实时任务和软实时任务可以被指定相关的优先级，以指导调度器。

子任务结构：一个任务可以分解成一个必须执行的子任务和一个可选的子任务。只有必须执行的子任务拥有硬最后期限。

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。

可编程 I/O：处理器代表进程给 I/O 模块发送给一个 I/O 命令，该进程进入忙等待，等待操作的完成，然后才可以继续执行。

中断驱动 I/O：处理器代表进程向 I/O 模块发送一个 I/O 命令，然后继续执行后续指令，当 I/O 模块完成工作后，处理器被该模块中断。如果该进程不需要等待 I/O 完成，则后续指令可以仍是该进程中的指令，否则，该进程在这个中断上被挂起，处理器执行其他工作。

直接存储器访问（DMA）：一个 DMA 模块控制主存和 I/O 模块之间的数据交换。为传送一块数据，处理器给 DMA 模块发送请求，只有当整个数据块传送完成后，处理器才被中断。

11.2 逻辑 I/O 和设备 I/O 有什么区别？

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一般 I/O 功能，允许它们根据设备标识符以及诸如打开、关闭、读、写之类的简单命令与设备打交道。

设备 I/O：请求的操作和数据（缓冲的数据、记录等）被转换成适当的 I/O 指令序列、通道命令和控

制器命令。可以使用缓冲技术，以提高使用率。

11.3 面向块的设备和面向流的设备有什么区别？请举例说明。

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

面向流的设备以字节流的方式输入输出数据，其末使用块结构。终端、打印机通信端口、鼠标和其他指示设备以及大多数非辅存的其他设备，都属于面向流的设备。

11.4 为什么希望用双缓冲区而不是单缓冲区来提高 I/O 的性能？

双缓冲允许两个操作并行处理，而不是依次处理。典型的，在一个进程往一个缓冲区中传送数据（从这个缓冲区中取数据）的同时，操作系统正在清空（或者填充）另一个缓冲区。

11.5 在磁盘读或写时有哪些延迟因素？

寻道时间，旋转延迟，传送时间

11.6 简单定义图 11.7 中描述的磁盘调度策略。

FIFO:按照先来先服务的顺序处理队列中的项目。

SSTF: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:磁头臂仅仅沿一个方向移动，并在途中满足所有未完成的请求，直到它到达这个方向上最后一个磁道，或者在这个方向上没有其他请求为止。接着反转服务方向，沿相反方向扫描，同样按顺序完成所有请求。

C-SCAN:类似于 SCAN，

11.7 简单定义图 7 层 RAID。

0：非冗余

1：被镜像；每个磁盘都有一个包含相同数据的镜像磁盘。

2：通过汉明码实现冗余；对每个数据磁盘中的相应都计算一个错误校正码，并且这个码位保存在多个奇偶校验磁盘中相应的文件。

3：交错位奇偶校验 ；类似于第二层，不同之处在于 RAID3 为所有数据磁盘中同一位置的位的集合计算一个简单的奇偶校验位，而不是错误校正码。

4：交错块分布奇偶校验；对每个数据磁盘中相应的条带计算一个逐位奇偶。

5：交错块分布奇偶校验；类似于第四层，但把奇偶校验条带分布在所有磁盘中。

6：交错块双重分布奇偶校验；两种不同的奇偶校验计算保存在不同磁盘的不同块中。

11.8 典型的磁盘扇区大小是多少？

512 比特

第12章　文件管理

12.1 域和记录有什么不同？

域（field）是基本数据单位。一个域包含一个值。

记录（record）是一组相关的域的集合 ，它可以看做是应用程序的一个单元。

12.2 文件和数据库有什么不同？

文件（file）是一组相似记录的集合，它被用户和应用程序看做是一个实体，并可以通过名字访问。

数据库（database）是一组相关的数据集合，它的本质特征是数据元素间存在着明确的关系，并且可供不同的应用程序使用。

12.3 什么是文件管理系统？

文件管理系统是一组系统软件，为使用文件的用户和应用程序提供服务。

12.4 选择文件组织时的重要原则是什么？

访问快速，易于修改，节约存储空间，维护简单，可靠性。

12.5 列出并简单定义五种文件组织。

堆是最简单的文件组织形式。数据按它们到达的顺序被采集，每个记录由一串数据组成。

顺序文件是最常用的文件组织形式。在这类文件中，每个记录都使用一种固定的格式。所有记录都具有相同的长度，并且由相同数目、长度固定的域按特定的顺序组成。由于每个域的长度和位置已知，因此只需要保存各个域的值，每个域的域名和长度是该文件结构的属性。

索引顺序文件保留了顺序文件的关键特征：记录按照关键域的顺序组织起来。但它还增加了两个特征：用于支持随机访问的文件索引和溢出文件。索引提供了快速接近目标记录的查找能力。溢出文件类似于顺序文件中使用的日志文件，但是溢出文件中的记录可以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。其结果是对记录的放置位置不再有限制，只要至少有一个索引的指针指向这条记录即可。此外，还可以使用长度可变的记录。

直接文件或散列文件：直接文件使用基于关键字的散列。

12.6 为什么在索引顺序文件中查找一个记录的平均搜索时间小于在顺序文件中的平均搜索时间？

在顺序文件中，查找一个记录是按顺序检测每一个记录直到有一个包含符合条件的关键域值的记录被找到。索引顺序文件提供一个执行最小穷举搜索的索引结构。

12.7 对目录执行的典型操作有哪些？

搜索，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什么关系？

路径名是由一系列从根目录或主目录向下到各个分支，最后直到该文件的路径中的目录名和最后到达的文件名组成。工作目录是一个这样的目录，它是含有用户正在使用的当前目录的树形结构。

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)，知道(knowledge)，执行(execution)，读(reading)，追加(appending)，更新(updating)，改变保护（changing protection），删除（deletion)。

12.10 列出并简单定义三种组块方式。

固定组块（fixed blocking)：使用固定长度的记录，并且若干条完整的记录被保存在一个块中。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。

可变长度跨越式组块（variable-length spanned blocking)：使用长度可变的记录，并且紧缩到块中，使得块中没有未使用空间。因此，某些记录可能会跨越两个块，通过一个指向后继块的指针连接。

可变长度非跨越式组块（variable-length unspanned blocking)：使用可变长度的记录，但并不采用跨越的方式。如果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。

连续分配是指在创建文件时，给文件分配一组连续的块。链式分配基于单个的块，链中的每一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有一个一级索引，分配给该文件的每个分区在索引中都有一个表项。

第13章　　网 络

13.1 网络访问层的主要功能是什么？

网络层主要关注在两个端系统（服务器、工作站）之间的数据交换，以及端系统间的物理网络。

13.2 传输层的任务是什么？

传输层关注的是数据的可靠性和保证数据能正确到达接收端应用程序。

13.3 什么是协议？

协议是定义了用来管理两个单元间进行数据交换的一系列规则的集合。

13.4 什么是协议体系结构？

这是一种实现通信功能的软件结构。典型地，协议结构包含了一个分层化的协议集，并且每个层中有一个或多个协议。

13.5 什么是 TCP/IP？

传输控制协议/互联网协议（TCP/IP）是两个最初为网际互连提供低层支持而设计的协议。TCP/IP 协

也被广泛应用于涉及由美国防卫部门和因特尔团体发展的比较广泛的协议集。

13.6 使用套接字接口的目的是什么？

套接字接口是一个能够编写程序的 API，从而利用 TCP/IP 协议程序建立一个用户端和服务器之间的通信。

第 14 章　分布式处理、客户/服务器和集群

14.1 什么是客户/服务器计算？

客户/服务器计算是一个网络环境，在这个网络环境中包含着客户机和服务器，并由服务器来响应客户机的请求。

14.2 客户/服务器计算与任何其他形式的分布式数据处理的区别是什么？

1、在用户的本地系统上为该用户提供界面友好的应用程序，这样做可使系统具有更高的可靠性。这使得用户可以在很大程度上控制对计算机的使用方式和时间，并使得部门级管理者具有响应本地需求的能力。

2、尽管应用是分散开的，但仍然强调公司数据库的集中以及很多网络管理和使用功能的集中。这使公司的管理者能够对计算信息系统的投资总额进行总体控制，并提供互操作，以使多系统能够配合起来。同时，减少了各部门和单位在维护这些复杂的计算机设施时的开销，使他们能够选择他们需要的各种类型的机器和接口来访问那些数据和信息。

3、对于用户组织和厂商来说，他们有一个共同的承诺事项，即使系统开放和模块化。这意味着用户在选择产品和混和使用来自众多厂商的设备时具有很多选择。

4、网络互联是操作的基础，网络管理和网络安全在组织和操作信息系统中具有很高的优先权。

14.3 像 TCP/IP 这样的通信结构在客户/服务器计算环境中的作用是什么？

它是使客户端和服务器能够协同工作的通信软件。

14.4 讨论将应用程序定位在客户上、服务器上或分开定位在客户和服务器上的基本原理。

基于服务器的处理：这种配置的基本原理是用户工作站最适宜于提供良好的用户界面，并且数据库和应用程序很容易在中心系统上维护。尽管用户获得了良好界面的好处，但是，这种配置类型并不总能有效提高处理效率或系统支持的实际商业功能上有本质的改变。基于客户的处理：它使用户能够使用适应本地需要的应用。合作处理：这种配置类型可以比其他客户/服务器方式为用户提供更高的生产效率和更高的网络效率。

14.5 什么是胖客户和瘦客户，两种方法在基本原理上的差别是什么？

胖客户：这是基于客户的处理，而大部分的软件都集中在客户端。胖客户模型的主要优点是它充分利用了桌面功能，分担了服务器上的应用处理并使它们更加有效，不容易产生瓶颈。

瘦客户：这是基于服务器的处理，而大部分的软件都集中在服务器。这种方式更近似地模拟了传统的以主机为中心的方式，常常是使将公司范围的应用程序从大型机环境迁移到分布式环境的途径。

14.6 给出将 pros 和 cons 用于胖客户和瘦客户策略的建议。

胖客户：胖客户模型的主要优点是它充分利用了桌面功能，分担了服务器上的应用处理并使它们更加有效，不容易产生瓶颈。新增加的功能很快就超出了桌面机器的处理能力，迫使公司进行升级。如果模型扩充超出了部门的界限，合并了很多用户，则公司必须安装高容量局域网来支持在瘦服务器和胖客户之间进行大量的传输。最后，维护、升级或替换分布于数十台或数百台桌面机的应用程序将变得非常困难。瘦客户：这种方式更近似地模拟了传统的以主机为中心的方式，常常是使将公司范围的应用程序从大型机环境迁移到分布式环境的途径。但是它不能提供和胖客户策略一样的灵活性。

14.7 解释三层客户/服务器体系结构的基本原理。

中间层机器基本上是位于用户客户和很多后端数据库服务器之间的网关。中间层机器能够转换协议，将对一种类型的数据库查询映像为另一种类型数据库的查询。另外，中间层机器能够融合来自不同数据源的结果。最后，中间层机器因介于两个层次之间而可以充当桌面应用程序和后端应用程序之间的网关。

14.8 什么是中间件？

中间件是在上层应用程序和下层通信软件和操作系统之间使用标准的编程接口和协议。它提供统一的方式和方法来跨越各种平台访问系统资源。

14.9 既然具有像 TCP/IP 这样的标准，为什么还需要中间件？

TCP/IP 不提供 API 和中间层协定来支持应用于不同的硬件和操作系统的多种应用程序平台。

14.10 列出消息传递的阻塞原语和无阻塞原语的优缺点。

无阻塞原语为进程提供了对消息传递机制高效而灵活的使用，这种方法的缺点是难于测试和调试使用这些原语的程序。问题的不可再现性与时间顺序相关性往往导致产生很多奇怪而麻烦的问题。阻塞原语有与无阻塞原语相反的优缺点。

14.11 列出远程过程调用的非永久性和永久性绑定的优缺点。

非永久绑定：因为连接需要维持两端的状态信息，因此需要消耗资源，非永久绑定类型用于保存这些资源。另一方面，建立连接所带来的开销使非永久绑定对同一个调用者频繁调用远程过程的情况不太适用。

永久绑定：对于对远程过程进行多次重复调用的应用程序，永久绑定保持着逻辑连接，并支持使用同一连接进行一系列的调用和返回。

14.12 列出同步远程过程调用和异步远程过程调用的优缺点。

同步远程过程调用易于理解和编程，因为它的行为是可以预期的。然而，它未能发挥分布式应用中固有的全部并行性。这就限制了分布式应用所能具有的交互性，降低了性能。为了提供更大的灵活性，各种异步远程过程调用机制已经得到实现，以获得更大程度的并行性而同时又保留了远程过程调用的通俗性和简易性。异步远程过程调用并不阻塞调用者，应答也可以在需要它们时接收到，这使客户在本地的执行可以与对服务器的调用并行进行。

14.13 列出并简短定义四种不同的构建集群的方法。

被动等待：当主服务器出现故障时，由从服务器来接管。

分离服务器：各服务器具有独自的磁盘，数据可连续地从主服务器复制至从服务器。

各服务器连接到磁盘：所有服务器都连接到同一磁盘，但每台服务器仍拥有自己的磁盘，一旦某台服务器发生故障，则其磁盘被其他服务器接管。

共享磁盘：多台服务器同时共享对磁盘的访问。

第 15 章　分布式进程管理

15.1 讨论实现进程迁移的原因。

负载共享：通过将进程从负载较重的系统迁移到负载较轻的系统，负载就会得到平衡，从而提高整体性能。通信性能：可以将交互密集的多个进程移动到同一节点上，以减少因为它们之间的交互而带来的通信开销。同样，当一个进程在某些文件或某组文件上执行数据分析，且文件的大小比进程要大很多时，将该进程移动到数据端也许是更有利的。可用性：需要长时间运行的进程，在得到错误的预先通知时，或者在预定的关机时间之前，为了能够存活下来，可能需要迁移到其他机器中。如果操作系统提供了这样的通知，则那些需要继续运行的进程可以迁移到另一个系统上，或者保证在稍后的某个时间在当前系统上能重新启动。特殊功能的使用：进程的迁移可以充分利用特定节点上独特的硬件或软件功能。

15.2 在进程迁移过程中，进程地址空间是如何处理的？

下列策略可能被采用：Eager（all）：在迁移时转移整个地址空间。

预先复制（precopy）：进程继续在源节点上执行，而地址空间已经复制到了目标节点上。在预先复制的过程中，源节点上的某些页有可能又被修改，这些页必须被复制第二次。

Eager（dirty）：仅仅转移那些位于主存中且已被修改了的地址空间的页。虚地址空间的所有其他块将在需要时才转移。

基于引用的复制（copy-on-reference）：这是 Eager（dirty）的变体，只有在引用某页时，该页才被

取入。

刷新（flushing）：通过把脏页写回磁盘，该进程的页可以从源机器的主存中清除。这样，在需要时可以从磁盘访问到页，而不是从源节点的存储器中访问。

15.3 抢占式和非抢占式进程迁移的动机是什么？

非抢占式进程迁移对于负载平衡是很有用的，它的优点是能够避免全面性进程迁移的开销，缺点是该方法对于负载分布的突然变化反应不佳。

15.4 为什么不可能确定真正的全局状态？

因为系统之间的通信延迟，不可能在系统范围内维护一个所有系统都随时可用的时钟。而且，维护一个中央时钟并让所有本地时钟与之保持精确同步，这在技术上也是不现实的，因为经过一段时间后，在各个本地时钟之间就会产生一些偏差，这将导致同步的丢失。

15.5 集中式算法和分布式算法所实行的分布式互斥有何区别？

在完全集中式算法中，一个节点被指定为控制节点，它控制对所有共享对象的访问。当任何进程请求对一个临界资源进行访问时，就向本地资源控制进程发送一个请求，这个进程接着向控制节点发送一条请求消息，当共享对象可用时，将返回一条许可消息。当进程结束使用资源后，向控制节点发送一条释放消息。

在分布式算法中，互斥算法涉及到每个离散的实体之间的同步合作。

15.6 定义两种类型的分布式死锁。

在资源分配中产生的死锁以及由于消息通信而产生的死锁。