第一章 操作系统概述课后习题

1. 硬件将处理机划分为两种状态，即管态和目态，这样做给操作系统设计带来什么好处 ？答：便于设计安全可靠的操作系统。管态和目态是计算机硬件为保护操作系统免受用户程序的干扰和破坏而引入的两种状态。通常操作系统在管态下运行，可以执行所有机器指令；而用户程序在目态下运行，只能执行非特权指令。如果用户程序企图在目态下执行特权指令，将会引起保护性中断，由操作系统终止该程序的执行，从而保护了操作系统。

2. 何为特权指令?举例说明之。如果允许用户执行特权指令，会带来什么后果？ 答：只能在管态下才能执行的指令称为特权指令。如开关中断、置程序状态寄存器等。如果允许用户执行特权指令，它将不仅影响当前运行的程序，而且还有可能影响操作系统的正常运行，甚至整个系统。

3. 中断向量在机器中的存储位置是由硬件确定的，还是由软件确定的 ？答：中断向量在机器中的位置是由硬件确定的。例如，在 INTEL 80x86 CPU 中，内存空间**0x00000——0x003ff**为中断向量空间。

4. 中断向量的内容是由操作系统程序确定的还是由用户程序确定的？ 答：中断向量的内容是由操作系统程序确定的。向量的内容包括中断处理程序的入口地址和程序状态字（中断处理程序运行环境），中断处理程序是由操作系统装入内存的，操作系统将根据装入的实际地址和该中断处理程序的运行环境来填写中断向量。

5. 中断向量内的处理机状态位应当标明是管态还是目态 ？为什么？ 答：应当标明是管态。 该状态由系统初试化程序设置，这样才能保证中断发生后进入操作系统规定的中断处理程序。

6. 中断和程序并发之间的关系是什么？ 答：中断是程序并发的必要条件。如果没有中断，操作系统不能获得系统控制权，无法按调度算法对处机进行重新分配，一个程序将一直运行到结束而不会被打断。

7. 说明“栈”和“堆”的差别. 答：栈是一块按后进先出（FIFO）规则访问的存储区域，用来实现中断嵌套和子程序调用的参数和返回断点。而堆虽然是一块存储区域，但是对堆的访问是任意的，没有后进先出的要求，堆主要用来为动态变量分配存储空间。

8. 何为系统栈？何为用户栈？系统栈有何用途？用户栈有何用途? 答：系统栈是内存中操作系统空间的一个固定区域；用户栈是内存中用户空间的一个区域。系统栈的作用：(1)保存中断现场，对于嵌套中断，被中断程序的现场信息依次压入系统栈，中断返回时逆序弹出；**(2)**保存操作系统子程序间相互调用的参数、返回值、返回点、以及子程序的局部变量。用户栈的作用：用于保存用户进程的子程序间相互调用的参数、返回值、返回点、以及子程序的局部变量。

9. 用户堆栈段的长度为何无法确定 ？答：用户堆栈段的长度主要取决于两个因素：（1）用户进程（线程）中子程序（函数）之间的嵌套调用深度；（**2**）子程序参数和局部变量的数量及类型；（**3**）动态变量的使用。这些在进程（线程）运行前无法确定，由此导致用户堆栈段的长度

无法预先准确确定。

10堆栈段的动态扩充为何可能导致进程空间的搬迁? 答：堆栈段的扩充需要在原来进程空间大小的基础上增添新的存储区域，而且通常要求与原来存储区域连续。由于原存放位置处可扩展的区域可能已经被其它进程占用，故可能需要将整个进程空间搬迁到另外一个区域，以实现地址空间扩展要求。

11. 何谓并行 ？何谓并发? 在单处理机系统中，下述并行和并发现象哪些可能发生，哪些不会发生 ?(1) 进程与进程之间的并行;(2) 进程与进程之间的并发;(3) 处理机与设备之间的并行;(4) 处理机与通道之间的并行;(5) 通道与通道之间的并行;(6) 设备与设备之间的并行;答：所谓并行是指同一时刻同时进行，进程并行需要多处理器的支持；所谓并发，是指在一段时间内，多个进程都在向前推进，而在同一时刻，可能只有一个进程在执行，多个进程轮流使用处理器。 在单处理器传统中，可能发生的并行和并发现象如下： (2) 进程与进程之间的并发。例如，在Windows操作系统中，**mp3**播放进程和**Word**字处理进程可以并发执行，这样用户就可以边听音乐边写文章了。 (3) 处理机与设备之间的并行。例如，当处理机进行科学运算时，打印机可以打印文档。 (4) 处理机与通道之间的并行。通道程序的执行可与处理机的操作并行。 (5) 通道与通道之间的并行。通常一个系统中有多个通道，这些通道可以并行地执行相应的通道程序。 (6) 设备与设备之间的并行。例如打印机打印文档时，磁带机在输入数据。

12. 何谓作业? 它包括哪几个部分? 各部分用途是什么 ?答：所谓作业是指用户要求计算机系统为其完成的计算任务的集合。一个作业通常包括程序、程序所处理的数据以及作业说明书。程序用来完成特定的功能，数据是程序处理的对象，作业说明书用来说明作业处理的步骤。

13. 从透明性和资源共享两方面，说明网络操作系统与分布式操作系统之间的差别。 答：从透明性上看，分布式操作系统优于网络操作系统。网络用户能够感觉到所访问的资源是在本地还是在远地；而在分布式系统中，用户感觉不到所访问的资源是否在本地，分布式操作系统掩盖了资源在地理位置上的差异。 从资源共享上看 ，分布式操作系统比网络操作系统能共享更多的资源。在网络操作系统中，一个计算任务不能由一台主机任意迁移到另外一台主机上运行；而在分布式操作系统中，所有作业可以由一台主机任意迁移到另外一台主机上处理，即可实现处理机资源的共享，从而达到整个系统的负载平衡。

14. 为什么构成分布式系统的主机一般都是相同的或兼容的?答：这样更有利于进程的动态迁移。如果主机不兼容，则在一台主机上能运行的进程，因所用指令系统不同，在另一台主机上可能无法运行，导致进程难于在不同主机间迁移，使得分布式系统难于实现负载平衡。构成分布式系统的主机一般都是相同的或兼容的。

15. 为什么嵌入式操作系统通常采用微内核结构? 答：嵌入式操作系统与一般操作系统相比具有比较明显的差别 ：(1)嵌入式操作系统规模一般较小，因为一般硬件配置较低，而且对操作系统提供的功能要求也不高。**(2)**应用领域差别大，对于不同的应用领域其硬件环境和设备配置情况有明显差别。 所以，嵌入式操作系统一般采用微内核 （ micro kernel ）结构,包括如下基本功能: **(1)**处理机调度；**(2)**基本内存管理；**(3)**通讯机制；**(4)**电源管理。在这些基本成分之上可进行扩展，以适应不同应用目标。

第二章 进程、线程与作业课后习题

1. 为何引入多道程序设计？在多道程序系统中，内存中作业的道数是否越多越好？请说明原因。 答：引入多道程序设计技术是为了提高计算机系统资源的利用率。在多道程序系统中，内存中作业的道数并非越多越好。一个计算机系统中的内存、外设等资源是有限的，只能容纳适当数量的作业，当作业道数增加时，将导致对资源的竞争激烈，系统开销增大，从而导致作业的执行缓慢，系统效率下降。

2. 什么是进程？进程具有那些主要特性？比较进程与程序之间相同点与不同点．答：进程是具有一定独立功能的程序关于一个数据集合的一次执行活动。特性：并发性、动态性、独立性、交往性、异步性和结构性。联系：程序是进程的组成部分，一个进程存在的目的就是执行其所对应的程序。区别：程序是静态的，而进程是动态的；进程是有生存期的，而程序没有；一个程序可对应多个进程，而一个进程只能对应一个程序。

3. 有人说，用户进程所执行的程序一定是用户自己编写的。 这种说法对吗？如不对举例说明之。 答：这种说法不对。例如， C编译程序以用户进程身份运行，但C编译程序并不是用户自己编写的。此外还有字处理程序等工具软件。

4. 什么是进程上下文？进程上下文包括那些成分？那些成分对目态程序是可见的？答： 在UNIX System V中，将进程的物理实体与支持进程运行的物理环境合称为进程上下文(process context)，进程上下文包括三个组成部分：• 用户级上下文。是由用户进程的程序块、用户数据块（含共享数据块）和用户堆栈组成的进程地址空间。**·系统级上下文。包括进程控制块、内存管理信息、进程环境块，以及系统堆栈等组成的进程地址空间 •寄存器上下文。由程序状态字寄存器、各类控制寄存器、地址寄存器、通用寄存器、用户堆栈指针等组成。其中用户级上下文及部分寄存器上下文对目态程序是可见得。**

5. 进程一般具有哪三个主要状态？举例说明状态转换的原因。 答：进程在其生存期内可能处于如下三种基本状态之一 ：(1) 运行态(Run)：进程占有处理机资源， 正在运行。 显然，在单处理机系统中任一时刻只能有一个进程处于此种状态；(2) 就绪态(Ready)： 进程本身具备运行条件，但由于处理机的个数少于可运行进程的个数，暂未投入运行。即相当于等待处理机资源；(3) 等待态(Wait)： 也称挂起态(Suspended)、封锁态(Blocked)、睡眠态(Sleep)。 进程本身不具备运行条件，即使分给它处理机也不能运行。 进程正等待某一个事件的发生，如等待某一资源被释放，等待与该进程相关的 I/O 传输的完成信号等。 进程的三个基本状态之间是可以相互转换的。具体地说，当一个就绪进程获得处理机时，其状态由就绪变为运行 ；当一个运行进程被剥夺处理机时，如用完系统分给它的时间片，或出现高优先级别的其它进程，其状态由运行变为就绪;当一个运行进程因某事件受阻时，如所申请资源被占用，启动 I/O 传输未完成，其状态由运行变为等待 ；当所等待事件发生时，如得到申请资源， I/O 传输完成，其状态由等待变为就绪。

6. 有几种类型进程队列？每类各应设置几个队列？答：有三种类型进程队列：就绪队列（整个系统一个）、等待队列（每个等待事件一个）和运行队列（在单CPU系统中只有一个）

7. 线程控制块 TCB 中一般应包含那些内容？ 答：一般 TCB 中的内容较少，因为有关资源分

配等多数信息已经记录于所属进程的 PCB中．TCB中的主要信息包括： 线程标识、线程状态、调度参数、现场、链接指针 ，其中现场信息主要包括通用寄存器、指令计数器 PC以及用户栈指针．对于操作系统支持的线程，TCB中还应包含系统栈指针。

8. 同一进程中的多个线程有那些成分是共用的，那些成分是私用的？答：共用的成分有：堆、数据和程序代码；私用的成分有：线程控制块、寄存器和用户栈。

9. 比较用户级线程与系统级线程间在以下方面的差别和各自的优缺点。 (1) 创建速度；(2) 切换速度；(3)并行性；(4) TCB 的存储位置 答：用户级线程由系统库支持．线程的创建和撤销，以及线程状态的变化都由库函数控制并在目态完成，与线程相关的控制结构 TCB 保存在目态空间并由运行系统维护。 由于线程对操作系统不可见， 系统调度仍以进程为单位，核心栈的个数与进程个数相对应用户级别线程的优点在于： (1)线程不依赖于操作系统，可以采用与问题相关的调度策略，灵活性好；(2)同一进程中的线程切换不需进入操作系统，因而实现效率较高。缺点在于：(1)同一进程中的多个线程不能真正并行，即使在多处理机环境中；(2)由于线程对操作系统不可见，调度在进程级别，某进程中的一个线程通过系统调用进入操作系统受阻，该进程的其它线程也不能运行 。 核心级别线程通过系统调用由操作系统创建，线程的控制结构 TCB 保存于操作系统空间，线程状态转换由操作系统完成，线程是 CPU 调度的基本单位。 另外由于系统调度以线程为单位，操作系统还需要为每个线程保持一个核心栈 。 核心级线程的优点是并发性好，在多 CPU 环境中同一进程中的多个线程可以真正并行执行 。 核心级别线程的缺点是线程控制和状态转换需要进入操作系统完成，系统开销比较大。 10.何谓作业步？作业何时转为进程?答：作业步：作业中一个相对独立的处理步骤作业进入内存，根据作业步的要求建立进程。11. 分析作业、进程、线程三者之间的关系.答：联系：一个作业包含多个进程，一个进程包含多个线程；区别：作业是向计算机提交任务的任务实体，而进程是执行实体，是资源分配的基本单位，线程是处理机调度的基本单位。12. 何谓系统开销? 试举三个例子说明之　　运行操作系统程序对系统进行管理而花费的时间和空间，如：作业调度、进程调度、进程切换等。

第三章 中段与处理机调度课后习题

1. 试说明下述概念之间的联系与差别：(1) 系统调用命令 **(2)** 访管指令 **(3)** 广义指令　　答：访管指令由指令码和访管中断号两部分组成。 即 ：　　SVC n ——— ①　　其中 SVC(SuperVisor Call)为指令码，表明为访管指令；**n** 为访管中断号，其值是一整数，具体表示何种访问要求。中断发生时，硬件中断装置将访管中断号 **n** 送入旧的程序状态字内的中断码字段，访管中断总控程序由系统堆栈中将其取出，并据此转入对应的服务程序。　　在实际使用时，用户程序与操作系统之间还需要相互传递参数和返回值。如此，用户使用访管指令的一般形式为 ：　　准备参数　　SVC n　　取返回值 ——— ②　　根据具体访管要求约定，参数及返回值可以通过寄存器传递，也可以通过内存传递。对于后者，操作系统必须能够访问进程空间。　　通常将②称为系统调用命令，它除访管指令外，还有准备参数和取返回值。为了使用方便，在高级语言中一般将其写为与过程调用相类似的形式，即： 返回值＝系统调用名称(参数**1**,参数**2**, … ,参数 **m**); ——— ③ 当然，编译程序会将③翻译成形如②的形式。其中系统调用名称对应①，不同的系统调用名称对应不同的整数 n 。 在有的书中，也将③称为代表②的宏指令或广义指令。

2. 为什么说中断是进程切换的必要条件，但不是充分条件？ 答： 发生进程切换时一定发生中断。系统由一个运行进程转去运行另外一个进程，前提条件是必须进入操作系统，即处于系统态，因为处于用户态运行的进程不可能将cpu的使用权直接交给另一个进程，而中断是从用户态转换为系统态的必要条件。即中断是进程切换的前提（必要）条件。 但发生中断时未必发生进程切换。如果中断处理完后原进程不再具有继续运行的条件，则一定会发生进程切换；反之，如果中断处理完后原进程仍具有继续运行的条件，则可能会发生进程切换，也可能不发生进程切换，这与处理机调度策略有关。

3. 试分析中断与进程状态转换之间的关系。 答：进程状态转换是由内核控制的，如果一个进程的状态发生了改变，则在新旧状态之间一定发生了处理机状态由目态到管态的转换，而中断是处理机状态由目态转换到管态的必要条件，所以中断也是进程状态转换的必要条件。

4. 中断发生时，旧的PSW和PC为何需要压入系统栈？ 答：保存断点信息，以便中断结束后接着原来的程序断点处继续执行。通常来说中断处理程序的最后一条指令是中断返回指令，该指令从系统栈顶弹出断点信息。如果未将PSW和PC压入系统栈，则中断返回指令弹出的不是中断前的断点信息，而是不确定的信息，这将导致系统处于不确定的状态，严重的情况会使系统崩溃。

5. 何谓中断向量？用户能否修改中断向量的值？ 答：当中断事件发生时，中断装置根据中断类别自动地将中断处理程序所对应的 PSW和PC送入程序状态字和指令计数器中，如此便转移到对应的中断处理程序。这个转移类似于向量转移，因而PSW和PC 被称为中断向量。 用户不能修改中断向量的值，因为修改中断向量是特权指令，普通用户程序不能执行特权指令。另外，如果允许用户修改中断向量的值，那么，用户就可以将中断向量与一段病毒程序联系起来，使中断发生时便执行病毒程序，从而破坏计算机系统。

6. 中断向量的存储位置是否可由程序改变？为什么？中断向量的值是如何确定的？ 答：中断向量的存储位置是由硬件确定的，不能由程序改变。中断发生后，中断装置按照中断类型到内存指定位置取出中断向量。例如，在IBM PC系统中，地址000~03FF是中断向量空间。操作系统的设计者根据各中断事件处理程序的存储位置及运行环境确定对应中断向量的值，系统启动时由初始化程序将该值填入指定位置。

7. 有人说，中断发生后硬件中断装置保证处理机进入管态，这种说法准确吗？说明理由。 答：这种说法不准确。中断发生后，硬件中断装置负责引出中断处理程序，中断处理程序是否运行于管态取决于PSW中的处理机状态位，该位的值是操作系统初始化时设置的，只有在初试化程序正确设置该状态位的前提下，才能保证中断后系统进入管态。

8. 为什么在中断处理过程中通常允许高优先级别的中断事件中途插入， 而不响应低优先级别的中断事件?答：(1)从逻辑上来说，高优先级别中断源所对应的事件比低优先级别中断源所对应的中断事件急迫；(2)由于硬件中断类型是有限的，这样做实际上也就限制了中断嵌套的深度。防止中断层数无限增长，甚

至系统栈溢出。

9. 为什么说“关中断”会影响系统的并发性? 答：考虑单处理机系统。在单处理机系统中，并发是通过将处理机轮流分配给多个进程而实现的，这个分配是由操作系统中处理机调度程序完成的。中断是进程切换的必要条件，如果关了中断，则操作系统无法获得处理机的控制权，也就无法使多个进程分时共享处理机。在关中断期间，一个进程独占处理机。所以说“关中断”会影响系统的并发性。

10. 假如关中断后操作系统进入了死循环，会产生什么后果?答：因为操作系统进入了死循环，并且处于关中断，即不能响应中断，也就不能从死循环中退出。系统不响应任何外部干预事件，系统表现为“死机”。

11. 为什么不允许目态程序执行关中断指令及中断屏蔽指令? 答：开关中断指令和中断屏蔽指令属于特权指令，一般用户无权访问。如果允许用户使用，用户关中断后可能影响系统对内部或外部事件的响应，也会使操作系统无法获得系统控制权。

12. 如果没有中断，是否能够实现多道程序设计？为什么?答：不能。因为一个程序一旦被调度执行，将一直执行下去，中间不可能被打断，不可能达到多个进程交替执行的并发目的。

13. 下列中断源哪些通常是可以屏蔽的，哪些通常是不可屏蔽的?（1）I/O 中断； **(2)** 访管中断；**(3)** 时钟中断；**(4)** 掉电中断答：（1）I/O中断可以屏蔽；**(2)** 访管中断不可以屏蔽；**(3)** 时钟中断可以屏蔽；**(4)** 掉电中断不可以屏蔽。

14. 下列中断事件哪些可由用户自行处理？哪些只能由操作系统中断服务程序统一处理？为什么?（1）溢出；**(2)** 地址越界；**(3)** 除零；**(4)** 非法指令；**(5)** 掉电答：一般来说，只影响应用程序自身的中断，可以由用户自行处理，包括：（1）溢出；**(3)** 除零。可能影响其它用户或操作系统的中断只能由操作系统中断服务程序统一处理，包括：**(2)** 地址越界；**(4)** 非法指令；**(5)** 掉电。

15. 如果中断由用户程序自行处理，为何需要将被中断程序的断点由系统堆栈弹出并压入用户堆栈? 答：中断发生时，被中断程序的现场信息已被压入系统堆栈中。而中断续元运行于目态，它执行完毕后将由用户堆栈区中恢复现场。 为此，操作系统在转到中断续元之前还应当将系统堆栈中的现场信息弹出并压入用户堆栈中，否则中断续元执行完毕后将无法恢复现场返回断点。

16. 对于下面中断与进程状态转换之间的关系各举两个例子说明之: （1）一定会引起进程状态转换的中断事件； （2）可能引起进程状态转换的中断事件.答：定会引起进程状态转换的中断事件：当前运行进程终止、应用程序启动I/O传输并等待**I/O**数据、运行程序申请当前被占用的某一资源。可能引起进程状态转换的中断事件：时钟中断事件可能引起进程状态转换，例如对于时间片轮转进程调度算法，

若时钟中断发生后，当前进程的时间片已用完，则将发生进程切换；否则不发生进程切换。

17. 若在 T1 时刻进程 P1 运行，T2 时刻进程 P2 运行,且 P1 ≠ P2 ,则在时刻 T1 和时刻 T2 期间之内一定发生过中断。 这种说法对吗 ？为什么？ 答：这种说法对。 如果在时刻 T1进程P1在运行，在时刻T2进程P2在运行，且P1≠P2，则说在时刻T1和时刻T2之间发生了进程切换。这说明在时刻T1和时刻T2之间执行了处理机调度程序，而 处理机调度程序是操作系统低层中的一个模块，在系统运行的过程中，除非显式地调用到该模块，否则系统不会由运行一个进程转去运行另外一个进程，就是说不会发生进程切换。只有进入操作系统，即处于系统态,才有可能调用到处理机调度， 因为处于用户态运行的用户程序不可能直接调用操作系统中的任何模块。中断是系统由用户态转换为系统态的必要条件。据此，假如在时刻T1与时刻T2之间发生了进程切换，则在时刻 T1 与时刻 T2 之间一定发生过中断。

18. 进程切换时，上升进程的PSW和PC为何必须由一条指令同时恢复？ 答：中断向量中程序状态字PSW与指令计数器PC的内容必须由一条指令同时恢复，这样才能保证系统状态由管态转到目态的同时，控制转到上升进程的断点处继续执行。如果不同时恢复，则只能（1）先恢复PSW再恢复PC，在恢复PSW后已经转到目态，操作系统恢复PC的使命无法完成；（2）先恢复PC再恢复PSW，PC改变后转到操作系统另外区域（因为PSW仍为系统状态），PSW无法恢复。

19. 某系统采用可抢占处理机的静态优先数调度算法，请问何时会发生抢占处理机的现象？ 答：当一个新创建的进程或一个被唤醒进程的优先数比正在运行进程的优先数高时，可能发生抢占处理机现象。

20. 在实时系统中，采用不可抢占处理机的优先数调度算法是否适宜？为什么？答：不适宜。一旦一个低优先数、需要大量CPU时间的进程占用处理机，就会一直运行，直到运行结束，或者直到因某事件而阻塞。在此之前，即使高优先数的紧急任务到达，也得不到处理，因而可能延误对重要事件的响应和处理。

21. 在分时系统中，进程调度是否只能采用时间片轮转算法？为什么？ 答：分时系统的特点是要求响应速度及时，除 RR算法之外，还可以采用可剥夺CPU的动态优先数调度算法。如经典UNIX的处理机调度算法，由于负反馈性质，算法也可以保证响应速度。

22. 有人说，在采用等长时间片轮转处理机调度算法的分时操作系统中，各终端用户所占有处理机的时间总量是相同的. 这种说法对吗？为什么？答：这种说法不对。因为处理机是分配给进程（线程）的，而不同终端用户可能有不同数量的进程，一个拥有较多数量进程的终端显然比拥有较少数量进程的终端获得CPU的时间要多。

23. 对于下述处理机调度算法分别画出进程状态转换图。 (1) 时间片轮转算法; (2) 可抢占处理机的优先数调度算法; (3) 不可抢占处理机的优先数调度算法。 答：

(1) 时间片轮转算法	(2) 可抢占处理机的优先数调度算法
	;

(3) 不可抢占处理机的优先数调度算法	

24. 举出两个例子说明操作系统访问进程空间的必要性. 答：例（1）：进程执行输出操作时，通过系统调用进入系统，由操作系统将待输出的数据由进程空间取出送给指定的外部设备，为此操作系统必须访问用户进程空间。例（2）：当发生可由用户自己处理的中断事件时，操作系统在转到中断续元之前应当将系统堆栈中的现场信息弹出并压入用户堆栈中，为此操作系统也必须访问进程空间。25．根据进程和线程的组成说明进程调度和线程调度各需要完成哪些工作。 答：进程调度： (1)地址映射寄存器；(2)用户栈指针；(3)通用寄存器；(4)PSW与PC。 线程调度： (1)用户栈指针；(2)通用寄存器；(3)PC。 26. 系统资源利用率与系统效率是否一定成正比? 如不是, 举例说明之.答：不是，如程序的并发执行时，并发要有个度，并发执行的程序过多，虽然系统资源利用率提高，但是，由于竞争过于激烈，切换过于频繁，系统开销大，反而会使系统效率降低。 27. 设有周期性实时任务集如下表所示，用 EDF 算法和 RMS 算 法是否可以调度？画出相应的 Gantt 图。

任务	发生周期*Ti*	处理时间*Ci*
A	30	10
B	40	15
C	50	5

答：由于 ，因而采用EDF算法一定可以调度，其Gantt图为：

```
    A1          B1        C1    A2          B2          C2    A3          B3     A4    C3
    10          15         5    10          15           5    10          15     10     5
0       10              25   30      40              55   60      70   80      95   105   110       120
```

由于 ，因而采用RMS算法不可调度。

第四章 互斥、同步与通讯课后习题答案

1．何谓与时间有关的错误？举例说明之。 答：并发进程的执行实际上是进程活动的某种交叉，某些交叉次序可能得到错误结果。由于具体交叉的形成与进程的推进速度有关，而速度是时间的函数，因而将这种错误称为与时间有关的错误。例如，两个并发进程的程序如下：

```
int n=0; main(){ 创建进程 A； 创建进程 B； };
A ( ) { while(1){ n++; } };                    B(){ while(1){ printf(n); n=0; } };
```

假设进程 A被部署在公园入口的终端上，用来记录进入公园的人数，进程B被部署在公园的控制中心，用来输出一段时间内进入公园的总人数。进程A和进程B共享全局变量n。如果在进程B执行完打印语句后被进程A打断，进程A执行了若干次变量自增语句，之后进程B接着执行清0语句，那么进程A对n的累加丢失了，相当于进程B被打断的这段时间内进入公园的人没有被记录下来。发生与时间有关的错误。

2．有人说，假设两个进程之间没有共享内存,则二者之间没有公共变量,这种说法准确吗? 说明原因.

答：如果只从用户空间考虑，这种说法是正确的。但从操作系统的角度来说并不准确。两个没有公共内存的用户进程可能同时（宏观）进入操作系统，并访问操作系统空间中的公共变量。

3. 何谓忙式等待 ？是否还有其它方式的等待？比较它们之间的联系和差别。 答：不进入等待状态的等待称为忙式等待。另一种等待方式是阻塞式等待，进程得不到共享资源时将进入阻塞状态，让 出 CPU 给其他进程使用。忙等待和阻塞式等待的相同之处在于进程都不具备继续向前推进的条件，不同之处在于处于忙等待的进程不主动放弃 **CPU** ， 尽 管 **CPU** 可 能被剥夺，因而是低效的；而处于阻塞状态的进程主动放弃 **CPU** ，因而是高效的。

4. 下列进程互斥方法哪些存在忙式等待问题？ （1）软件：面包店算法 **(2)** 硬件：**TS**指令 **(3)** 关中断指令答：（1）、（**2**）存在忙等待问题。

5. 为何开关中断进程互斥方法仅在单 CPU 系统中是有效的？ 答：关中断方法不适用于多 CPU 系统，因为关中断只能保 证 **CPU** 不由一个进程切换到另外一个进程，从而防止多个进程并发地进入公共临界区域。但即使关中断后，不同进程仍可以在不同 **CPU** 上并行执行关于同一组共享变量的临界区代码。

6. 在多处理机系统中，软件互斥方法是否有效？为什么？答：依然有效。多处理机并行与单处理并发之间的差别在于程序交叉的粒度，单处理机机环境中进程交叉发生在指令之间，多处理机环境中进程交叉发生在指令周期之间。由于纯软件互斥算法并不依赖特殊的硬件指令(如**test_and_set**)，指令之间的交叉与指令周期之间的交叉结果相同。

7. 试分析临界区域的大小与系统并发性之间的关系。 答：关于同一组变量的临界区域是不能并发执行的代码，临界区越大，并发性越差，因而编写并发程序应尽量减小临界区域的大小。

8. 设CR1是关于一组共享变量**SV1**的临界区域,**CR2**是关于另外一组共享变量**SV2**的临界区域，当进程**P1**进入**CR1**时,进程**P2**是否可以进入**CR2?** 为什么**?**答：进程互斥是指多个进程不能同时进入关于同一组共享变量的临界区，否则可能发生于时间有关的错误。SV1与**SV2**不是同一组共享变量，所以当进程**P1**进入**CR1**时，进程**P2**可以进入**CR2**。

9. Lamport 面包店互 斥算法是否会出现饿死情况 ？答：不会，该算法是公平的。假定系统中共有 n 个进程，每个想要进入临界区域的进程（线程）在最坏的情况下需要等待其它 n -1个进程进入并离开临界区域之后即可获得进入临界区域的机会，因而存在(忙式)等待的上界。

10. 试用信号灯和PV操作实现临界区语句: **region <共享变量〉** do 〈语句**>**答：semaphore mutex=1; … **p(mutex);**〈语句〉 **v(mutex);**

…11. 由 V 操 作唤醒的进程是否一定能够直接进入运行状态？举例说明之。 答：否。一般来说，唤醒是将进程状态由等待状态变成就绪状态，而就绪进程何时获得处理机则是由系统的处理机调度策略确定的。如果采用抢占式优先级调度算法，并且被唤醒的进程是当前系统中优先级最高的进程，那么该进程将被调度执行，该进程的状态变成运行态。如果该进程不是系统中优先级最高的进程或系统采用其它调度算法，那么该进程不会被调度执行，其状态将维持在就绪态。

12. 设S1和S2为两个信号灯变量，下列八组P、V操作哪些可以同时进行？哪些不能同时进行？为什么？(1) P(S1), P(S2) (2) P(S1), V(S2)(3) V(S1), P(S2) (4) V(S1), V(S2) (5) P(S1), P(S1) (6) P(S2), V(S2)(7) V(S1), P(S1) (8) V(S2), V(S2)答： (1),(2),(3),(4)可以同时进行；（5），（6），（7），（8）不可以同时进行。因为如果将信号灯变量看作共享变量，则P、V操作作为其临界区，为了不发生于时间有关的错误，须保证多个进程不能对同一个信号灯变量同时执行P、V操作。

13. 对于生产者—消费者问题，假设缓冲区是无界的，试用信号灯与 PV 操作给出解法。 答：由于是无界缓冲区，所以生产者不会因得不到缓冲区而被阻塞。

```
semaphore  mutex_in=1;  semaphore  mutex_out=1;
semaphore product=0; int in=0 ,  out=0;
生产者：  while(1){ 生产一个物品；  P(mutex_in); 消费者：  while(1){ P(product); P(mutex_out); 物品
buffer[in]=物品；  in++; v(mutex_in); V(product); }  =buffer[out]; out++; V(mutex_out); }
```

14. 设有一个可以装A、B两种物品的仓库，其容量无限大，但要求仓库中A、B两种物品的数量满足下述不等式:-M≤A物品数量－B物品数量≤N其中M和N为正整数．试用信号灯和PV操作描述A、B两种物品的入库过程.解：若只放入A，而不放入B，则A产品最多可放入N次便被阻塞；若只放入B，而不放入A，则B产品最多可放入M次便被阻塞；每放入一次A,放入产品B的机会也多一次；同理，每放入一次B,放入产品A的机会也多一次.Semaphore mutex=1， sa=N， sb=M;产品A进程： 产品B进程 :while(1){ while(1){p(sa);
p(sb);p(mutex); p(mutex);A产品入库； B产品入库 ； V(mutex); V(mutex);V(sb);
V(sa);} }

15. 试用信号灯与 PV 操作实现司机与售票员之间的同步问题。设公共汽车上有一个司机和一个售票员，其活动如下图所示。 为了安全起见，显然要求 ：(1)关车门后方能启动车辆; (2)到站停车后方能开车门。亦即“启动车辆”这一活动应当在“关车门”这一活动之后，“开车门”这一活动应当在“到站停车”这一活动之后。 semaphore s1=0,s2=0; 司机： 售票员：
while(1){ while(1){ p(s1); 关车门； 启动车辆；
v(s1); 正常行驶； 售票； 到站停车； p(s2); v(s2);
开车门； } }

16. 设有A、B、C三组进程，它们互斥地使用某一独占型资源R，使用前申请，使用后释放．资源分配原则如下： (1) 当只有一组申请进程时，该组申请进程依次获得R； (2) 当有两组申请进程时，各组申请进程交替获得R，组内申请进程交替获得R； (3) 当有三组申请进程时，各组申请进程轮流获得R，组内申请进程交替获得R.试用信号灯和PV操作分别给出各组进程的申请活动程序段和释放活动程序段.int free=1;//设备状态标志semaphore mutex=1;semaphore qa=qb=qc=0; //各组等待队列int counta=countb=countc=0;//等待队列长度

A组申请：P(mutex);if(free==1){free=0;V(mutex);}else{counta++;V(mutex);P(qa);}

A组释放：if(countb>0){countb--;V(qb);}else{if(countc>0){countc--;V(qc);}else{if(counta>0){counta--V(qa);}else{ free=1;}}}

A组进程活动可以给出B组和C组进程活动。

17. 设自行车生产线上有一只箱子，其中有 N 个位置（N ≥ 3)，每个位置可存放一个车架或一个车轮；又设有三个工人，其活动分别为:

工人 1活动： do { 加工一个车架 ； 车架放入箱中 ； } while (1)

工人 2活动： do { 加工一个车轮 ； 车轮放入箱中 ； } while (1)

工人 3活动： do { 箱中取一车架 ； 箱中取二车轮 ； 组装为一台车 ； } while (1)

试分别用信号灯与 PV 操作、管程、会合实现三个工人的合作，要求解中不含死锁。 一、用信号灯与 PV 操作实现三个工人的合作 首先不考虑死锁问题，工人 1与工人3、工人2与工人3构成生产者与消费者关系，这两对生产/消费关系通过共同的缓冲区相联系。从资源的角度来看，箱字中的空位置相当于工人1和工人2的资源，而车架和车轮相当于工人3的资源。定义三个信号灯如下： semaphore empty=N; semaphore wheel=0; semaphore frame=0; 三位工人的活动分别为：

工人 1活动： do { 加工一个车架 ； P(empty); 车架放入箱中 ； V(frame); } while (1)

工人 2活动： do { 加工一个车轮 ； P(empty); 车轮放入箱中 ； V(wheel); } while (1)

工人 3活动： do{P(frame); 箱中取一车架 ； V(empty); P(wheel); P(wheel); 箱中取二车轮 ； V(empty); V(empty); 组装为一台车 ; } while (1)

分析上述解法易见，当工人 1推进速度较快时，箱中空位置可能完全被车架占满或只留有一个存放车轮的位置，而当此时工人3同时取2个车轮时将无法得到，而工人2又无法将新加工的车轮放入箱中；当工人2推进速度较快时，箱中空位置可能完全被车轮占满，而当此时工人3同取车架时将无法得到，而工人1又无法将新加工的车架放入箱中 。上述两种情况都意味着死锁。 为防止死锁的发生，箱中车架的数量不可超过 N-2，车轮的数量不可超过N-1，这些限制可以用两个信号灯来表达。 semaphore s1=N-2; semaphore s2=N-1; 如此，可以给出不含死锁的完整解法如下：

工人 1活动： do { 加工一个车架 ； P(s1); P(empty); 车架放入箱中 ； V(frame); } while (1)

工人 2活动： do { 加工一个车轮 ； P(s2); P(empty); 车轮放入箱中 ； V(wheel); } while (1)

工人 3活动： do{P(frame); 箱中取一车架 ； V(empty); V(s1); P(wheel); P(wheel); 箱中取二车轮 ； V(empty); V(empty); V(s2); V(s2); 组装为一台车 ; } while (1)

详细描述还应考虑对箱子单元的描述以及访问互斥问题。建议车架放在箱子的一端，车轮放在箱子的另

一端，车架与车轮都采用后进先出的管理方式。Semaphore s1=N-2，s2=N-1，**mutex=1**;int in1=0，in2=N-1;int buf[N];

工人 1活动： do { 加工一个车架 ； P(s1); P(empty);P(mutex);Buf[in1]=车架; in1=in1+1; V(mutex); V(frame); } while (1)

工人 2活动： do { 加工一个车轮 ； P(s2); P(empty); P(mutex);Buf[in2]= 车 轮 ； in2=in2-1; V(mutex);V(wheel); } while (1)

工 人 3 活 动 ： do { P(frame); P(mutex);Temp1=Buf[in1-1] ； in1=in1-1; V(mutex); V(empty); V(s1); P(wheel); P(wheel); P(mutex);Temp2=Buf[in2+1];in2=in2+1;Temp3=Buf[in2+1];In2=in2+1;V(mutex);V(empty); V(empty); V(s2); V(s2); 组装为一台车 ; } while (1)

18. 一座小桥(最多只能承重两个人)横跨南北两岸，任意时刻同一方向只允许一人过桥，南侧桥段和北侧桥段较窄只能通过一人，桥中央一处宽敞，允许两个人通过或歇息．试用信号灯和PV操作写出南、北两岸过桥的同步算法．

semaphore load=2;semaphore north=1;semaphore south=1;

tosouth(){P(load);P(north); 过 北 段 桥 ; 到 桥 中 间 ;V(north);P(south); 过 南 段 桥 ; 到 达 南 岸 V(south);V(load);}

tonorth(){P(load);P(south); 过 南 段 桥 ; 到 桥 中 间 V(south);P(north); 过 北 段 桥 ; 到 达 北 岸 V(north);V(load);}

19. 某寺庙，有小和尚、老和尚若干．庙内有一水缸，由小和尚提水入缸，供老和尚饮用．水缸可容纳 30 桶水，每次入水、取水仅为1桶，不可同时进行。水取自同一井中，水井径窄，每次只能容纳一个水桶取水。设水桶个数为5个，试用信号灯和 PV 操作给出老和尚和小和尚的活动。

semaphore empty=30; // 表示缸中目前还能装多少桶水，初始时能装 30 桶水 semaphore full=0; // 表示缸中有多少桶水，初始时缸中没有水 semaphore buckets=5; // 表示有多少只空桶可用，初始时有 5 只桶可用 semaphore mutex_well=1; // 用于实现对井的互斥操作 semaphore mutex_bigjar=1; // 用于实现对缸的互斥操作 semaphore mutex_bucket=1; //用于实现对桶的互斥操作

young_monk() { while(1){ P(empty); P(buckets);P(mutex_bucket); 取一个桶； V(mutex_bucket); go to the well; P(mutex_well); get water; V(mutex_well); go to the temple; P(mutex_bigjar); pure the water into the big jar; V(mutex_bigjar); V(buckets); V(full); } }

old_monk() { while(){ P(full); P(buckets); P(mutex_bucket); get a bucket; V(mutex_bucket); P(mutex_bigjar); get water; V(mutex_bigjar); V(buckets); V(empty); } }

20. 设系统中有5台类型相同的打印机，依次编号为1～5。 又设系统中有n个使用打印机的进程，使用前申请，使用后释放。 每个进程有一个进程标识，用于区别不同的进程。 每个进程还有一个优先数，不同进程的优先数各异。当有多个进程同时申请时，按照进程优先数由高到低的次序实施分配。 试用信号灯和PV操作实现对于打印机资源的管理，即要求编写如下函数和过程:(1) 函数 require(pid，pri): 申请一台打印机。参数pid为进程标识，其值为1到n的整数; pri为进程优先数，其值为正整数; 函数返回值为所申请到打印机的编号，其值为1到5的整数;(2) 过程 return(prnt): 释放一台打印机。参数prnt为所释放打印机的编号，其值为1到5的整数。

解：#define N 5Int flag[N+1];//flag[0]表示可用打印机数,//flag表示第i号打印机的状态（1<=i<=N），0表示占用，1表示空闲PCB *queue=NULL;//进程阻塞队列semaphore mutex_flag=1;//用于对flag数组的互斥操作semaphore mutex_queue=1;//用于对阻塞队列的互斥操作

int require(int pid,int priority){P(mutex_flag);if(flag[0]>0){flag[0]--;for(int i=1;i<N+1;i++)if(flag==1){flag=0;break;}V(mutex_flag);return i;}else{V(mutex_flag);p(mutex_queue);将进程pid按其优先数插入到等待队列queue中；V(mutex_queue);}}

return(int print){P(mutex_flag);if(queue==NULL){flag[0]++;flag[print]=1;V(mutex_flag);}else{V(mutex_flag);p(mutex_queue);将print分配给queue队首进程；queue下移；V(mutex_queue);}}

26. 关于读者/写者问题,有人给出如下改进解法:semaphore r_w_w, mutex, s；(初值均为1)int count；(初值为0)

读者活动：P(s);P(mutex);count++;if (count==1)P(r_w_w);V(mutex);V(s);{读操作}P(mutex);count--;If (count==0)V(r_w_w);V(mutex);

写者活动：P(s);P(r_w_w);{写操作}V(r_w_w);V(s);

分析上述改进算法的调度效果。答：由于s以及读者和写者对s的操作，读者和写者都不会无限等待，因而算法不会出现饿死现象，是一个公平的解法。

第五章 死锁与饥饿课后习题答案

1. 下面关于死锁问题的叙述哪些是正确的，哪些是错误的，说明原因. (1) 参与死锁的所有进程都占有资源； (2) 参与死锁的所有进程中至少有两个进程占有资源； (3) 死锁只发生在无关进程之间;(4) 死锁可发生在任意进程之间.答：说法（1）是错误的，应该是参与死锁的所有进程都等待资源。说法(2)正确。 参与死锁的进程至少有两个，设为p1,p2，p1占有资源r1而等待资源r2，p2占有资源r2而等待资源r1。说法(3)错误。死锁也可能发生在相关进程之间。说法(4)正确，死锁既可能发生在相关进程之间，也可能发生在无关进程之间。即死锁可发生在任意进程之间。

2. 试证明当每个资源类中仅有一个资源实例时，资源分配图中的环路是死锁的充要条件。证明：先证明充分条件：用反证法，假设每个资源类中仅有一个资源实例时，资源分配图的环路是可约简的，那么说明环路外至少有一个非孤立且没有请求边的进程节点pk占有一个资源类rk中的一个实例,而rk中的另外一个实例被环路中某个进程占有。说明rk中有两个以上的资源实例，与前提矛盾。所以说，每个资源类中仅有一个资源实例时，资源分配图的环路是不可约简的，根据死锁定理，得出结论每个资源类中仅有一个资源实例时，资源分配图若存在环路就产生死锁。 再证明必要条件：若死锁产生，则存在一个

循环等待进程序列<p1, p2, p3……pn>，进程**p1**正等待资源类r_{k1}中唯一的一个实例，而r_{k1}又被进程**p2**所占用；进程**p2**正等待资源类r_{k2}中唯一的一个实例，而r_{k2}又被进程**p3**所占用；……；进程**pn**正等待资源类r_{kn}中唯一的一个实例，而r_{kn}又被进程**p1**所占用。能看出，画出的资源分配图存在环路。

3.　　什么叫饥饿？什么叫饿死？什么叫活锁？举例说明之．答：在一个动态系统中，资源请求与释放是经常性发生的进程行为．对于每类系统资源，操作系统需要确定一个分配策略，当多个进程同时申请某类资源时，由分配策略确定资源分配给进程的次序。 资源分配策略可能是公平的(fair)，能保证请求者在有限的时间内获得所需资源；资源分配策略也可能是不公平的(unfair)，即不能保证等待时间上界的存在。 在后一种情况下，即使系统没有发生死锁，某些进程也可能会长时间等待．当等待时间给进程推进和响应带来明显影响时，称发生了进程饥饿(starvation)，当饥饿到一定程度的进程所赋予的任务即使完成也不再具有实际意义时称该进程被饿死(starve to death)。在忙式等待条件下发生的饥饿，称为活锁．考虑一台打印机分配的例子，当有多个进程需要打印文件时，系统按照短文件优先的策略排序，该策略具有平均等待时间短的优点，似乎非常合理，但当短文件打印任务源源不断时，长文件的打印任务将被无限期地推迟，导致饥饿以至饿死。

4.　　死锁与饿死之间有何相同点和不同点?答：饿死与死锁有一定联系：二者都是由于竞争资源而引起的，但又有明显差别，主要表现在如下几个方面：(1) 从进程状态考虑，死锁进程都处于等待状态，忙式等待(处于运行或就绪状态)的进程并非处于等待状态，但却可能被饿死；(2) 死锁进程等待永远不会被释放的资源，饿死进程等待会被释放但却不会分配给自己的资源，表现为等待时限没有上界(排队等待或忙式等待)；(3) 死锁一定发生了循环等待，而饿死则不然。这也表明通过资源分配图可以检测死锁存在与否，但却不能检测是否有进程饿死；(4) 死锁一定涉及多个进程，而饥饿或被饿死的进程可能只有一个。饥饿和饿死与资源分配策略(policy)有关，因而防止饥饿与饿死可从公平性考虑，确保所有进程不被忽视，如**FCFS**分配算法。

5.　　何谓银行家算法的保守性? 举例说明之.答：银行家算法的保守性是指银行家算法只给出了进程需要资源的最大量，而所需资源的具体申请和释放顺序仍是未知的，因而银行家只能往最坏处设想．例如：书中举例p119页。例**5.5**。

6.　　能否给出避免死锁的充要性算法? 为什么 **?**答：目前关于避免死锁的算法，如银行家算法是充分性算法，即确保系统时刻处于安全状态，这是在系统已知每个进程所需资源最大量的条件下可以给出的最好结果。如果系统不仅知道每个进程所需资源的最大量，而且知道进程有关资源的活动序列，在这个更强的条件下，则可以给出避免死锁的充要性算法（读者可以证明），但其复杂度是很高(NP完全)的。而且由于程序中分支和循环的存在，事先给出进程有关资源的命令序列一般是不可能的。

7.　设有一个T型路口，其中A、B、C、D处各可容纳一辆车，车行方向如下图所示，试找出死锁并用有

序分配法消除之. 要求资源编号合理.解：(1)E方向两台车分别位于A和B；S方向一台车位于C；W方向一台车位于D。(2)S方向两台车分别位于B和C；E方向一台车位于A；W方向一台车位于D。 设位置资源C、B、A、D的编号从低到高依次为1、2、3、4，管理四个位置的信号量分别为s1,s2,s3,s4，信号量的初值均为1。车辆活动如下：

```
semaphore s1=1,s2=1,s3=1,s4=1;
W：直行 P(s1);// 按序申请 P(s4); 驶入 D;驶入 C; V(s4); 驶出 C; V(s1);
E：左转 P(s2); 驶入 B;P(s3)； 驶入 A; V(S2) P(s4)； 驶入 D; V(s3)； 驶出 D;V(s4);
S：左转 P(s1); 驶入 C;P(s2)； 驶入 B; V(S1) P(s3)； 驶入 A; V(s2)； 驶出 A;V(s3);
```

8. 设系统中仅有一个资源类，其中共有M个资源实例，使用此类资源的进程个数共有N个，它们所需资源最大量总和为Σ，试证明发生死锁的必要条件是Σ3M+N. 答：证明：假定发生死锁，那么Alloc(1)+Alloc(2)+…+Alloc(N)=M，(Alloc(i)表示第i进程已分配的资源量)。Need(1)+Need(2)+…Need(N) 3N。(Need(i)表示第i进程还需要的资源量)所以，发生死锁时，所有进程所需资源的总量3ΣM+N。

9. 在银行家算法中，若出现如下资源分配情况：

	Allocation	Need	Available
	A B C D	A B C D	A B C D
P0:	0 0 3 2	0 0 1 2	1 6 2 3
P1:	1 0 0 0	1 7 5 0	
P2:	1 3 5 4	2 3 5 6	
P3:	0 3 3 2	0 6 5 2	
P4:	0 0 1 4	0 6 5 6	

试问：(1)当前状态是否安全？ (2)如果进程P2提出安全请求Request[2]=(1,2,2,2),系统能否将资源分配给它？说明原因. 解：（1）当前状态是安全状态。运行安全性检查算法如下： 1）Work = Available；Finish = false； 2）寻找满足如下条件的i：Finish[i]==false并且Need[i]≤Work[i]； 如果不存在，则转步骤4)； 3）Work = Work + Allocation[i]；Finish[i] = true； 转步骤2）4）如果对于所有i，Finish[i] = true，则系统处于安全状态，否则处于不安全状态。 令Work = Available=（1, 6, 2, 3）运行安全性检测算法，Finish[0]=false并且Need[0]=(0 0 1 2)<Work,则Work = Work + Allocation[0]=（1, 6, 2, 3）+（0, 0, 3, 2）=（1, 6, 5, 5）；Finish[0] = true；Finish[3]=false并且Need[3]=（0, 6, 5, 2）<Work,则Work = Work + Allocation[3]=（1, 6, 5, 5）+（0, 3, 3, 2）=（1, 9, 8, 7）；Finish[3]= true；Finish[4]=false并且Need[4=（0, 6, 5, 6）<Work,则Work = Work + Allocation[4]=（1, 9, 8, 7）+（0, 0, 1, 4 ）=（1, 9, 9, 11）； Finish[4] = true； Finish[1]=false并且Need[1]=（1, 7, 5, 0）<Work,则Work = Work + Allocation[4]=(1, 9, 9, 11）+(1, 0, 0, 0)=(2, 9, 9, 11)；Finish[1] = true；Finish[2]=false并且Need[2]=（2, 3, 5, 6）<Work,则Work = Work + Allocation[4]=（2, 9, 9, 11）+（1, 3, 5, 4 ）=（3, 12, 14, 15）；Finish[2] = true； 可以找到一个安全进程序列<p_0，p_3，p_4，p_1,p_2>，它使Finish=*true*，*对于所有0≤i≤4，因而可以断言系统当前处于安全状态.*（2）运行银行家算法，由于Request[2]=（1, 2, 2, 2）£Need[2]=（2, 3, 5, 6），因而请求合法。进一步，Request[2]=（1, 2, 2, 2）£Available=（1, 6, 2, 3），故该请求是可以满足的。假设将资源分配给p2，则系统状态变为：

	Allocation	Need	Available
	A B C D	A B C D	A B C D
P0:	0 0 3 2	0 0 1 2	0 4 0 1
P1:	1 0 0 0	1 7 5 0	
P2:	2 5 7 6	1 1 3 4	
P3:	0 3 3 2	0 6 5 2	
P4:	0 0 1 4	0 6 5 6	

运行安全性检测算法，Work=Available=（0, 4, 0, 1），

Finish=false，此时所有Need£Work均不成立，结果Finish[i[均为false，不存在安全进程序列，系统处于不安全状态。系统将取消资源分配并恢复原来状态，进程p2等待。

10．某系统采用死锁检测手段发现死锁，设系统中资源类集合为{A，B，C}，资源类A中共有8个实例，资源类B中共有6个实例，资源类C中共有5个实例．又设系统中进程集合为{p1,p2,p3,p4,p5,p6}，某时刻系统状态如下：

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
p1:	1	0	0	0	0	0	2	2	1
p2:	3	2	1	0	0	0			
p3:	0	1	2	2	0	2			
p4:	0	0	0	0	0	0			
p5:	2	1	0	0	3	1			
p6:	0	0	1	0	0	0			

(1)　在上述状态下系统依次接受如下请求：Request[1]=(1,0,0)；Request[2]=(2,1,0)；Request[4]=(0,0,2)．给出系统状态变化情况，并说明没有死锁．答：如果系统只是接受请求，但是没有分配资源给进程，那么系统状态变为：

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
p1:	1	0	0	1	0	0	2	2	1
p2:	3	2	1	2	1	0			
p3:	0	1	2	2	0	2			
p4:	0	0	0	0	0	2			
p5:	2	1	0	0	3	1			
p6:	0	0	1	0	0	0			

在该状态下运行死锁检测算法，可以找到一个进程序列<p4,p1,p2,p3,p5,p6>，它使Finish=true，对于所有1≤i≤6，因而可以断言系统当前没有进入死锁状态。(2)　在由(1)所确定的状态下系统接收如下请求：Request[1]=(0,3,1)，说明此时已发生死锁，并找出参与死锁的进程．答：设在（1）的状态下系统接收如下请求：Request[1]=(0,3,1)，则系统状态变为：

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
p1:	2	0	0	0	3	1	1	2	1
p2:	3	2	1	2	1	0			
p3:	0	1	2	2	0	2			
p4:	0	0	0	0	0	2			
p5:	2	1	0	0	3	1			
p6:	0	0	1	0	0	0			

在该状态下运行死锁检测算法，找不到一个进程序列使Finish=true，对于所有1≤i≤6，因为存在i∈{1，2，3，5}，使Finish=false，因而可以断言系统已经进入死锁状态，进程p1,p2,p3,p5卷入死锁．

11．设有7个简单资源：A、B、C、D、E、F、G。其申请命令分别为a、b、c、d、e、f、g；释放命令分别为a-、b-、c-、d-、d-、f-、g-；又设系统中有P1、P2、P3三个进程，其活动分别为：P1活动：a b a- b- e f g e- f- g-　P2活动：b c b- c- d a d- a-　P3活动：c d c-d- e g f e- f- g-　试分析当P1、P2、P3并发执行时，是否有发生死锁的可能性，并说明原因。解：不会有发生死锁的可能性。　在本题中，进程p1和p2都使用的资源集合是{a,b}，由于进程p2在申请a之前已经释放了b，不存在占有b并且申请a的情况，所以进程p1和p2之间不满足死锁的四个必要条件，不会产生死锁；进程p1和p3都使用的资源集合是{e,f,g}，进程p1和p3都是先申请资源e，这两个进程同时申请资源，那么只能有一个进程先获得e，另一个进程将因为得不到e而阻塞，获得e的进程将进一步顺利获得资源f和g，从而运行结束，释放资源e,f和g，唤醒另一个进程运行。可见，进程p1和p3之间不会产生死锁；进程p2和p3都使用的资源集合是{c,d}，由于进程p2在申请d之前已经释放了c，不存在占有c并且申请d的情况，所以进程p2和p3之间不满足死锁的四个必要条件，不会产生死锁。综上所述，当P1、P2、P3并发执行时，没有发生死锁的可能性。

第六章 存储管理课后习题答案

1．考虑下述存储管理方式中，进程空间和逻辑空间的编址情况：（1）界地址存储管理方式，进程空间的首地址；（2）页式存储管理，进程空间的首地址；（3）段式存储管理，进程空间各段的首地址；（4）段页式存储管理，进程空间各段的起始地址．答：（1）界地址存储管理方式，

进程空间的首地址从0开始编址；（2）页式存储管理，进程空间的首地址从0开始编址，而逻辑空间划分为若干个页面，每个页面的起始地址是逻辑页号乘以页面大小；（3）段式存储管理，进程空间各段的首地址从0开始编址；（4）段页式存储管理，进程空间各段的起始地址从0开始编址.

2. 对于如下存储管理方式来说，进程地址空间各是几维的？（1）页式；（2）段式；（3）段页式答：（1）页式的进程地址空间是一维的　（2）段式的进程地址空间是二维的　（3）段页式的进程地址空间是二维的

3. 在页式存储管理中，页的划分对用户是否可见？在段式样存储管理中，段的划分对用户是否可见？在段页式存储管理中，段的划分对用户是否可见？段内页的划分对用户是否可见？答：(1)在页式存储管理中，分页对于用户是透明的，一个进程由若干个页构成，所有页的长度相同；(2)在段式存储管理中，分段对于用户是可见的，一个进程由若干个段构成，各个段的长度可以不同，一个段恰好对应一个程序单位。(3)在段页式存储管理中，段的划分对用户是可见的，段内页的划分对用户是透明的，一个段由若干个页构成，所有页的长度相同。

4. 为什么空闲页面链适合管理内存空间，而不适合管理外存空间？答：空闲页面链是将所有的空闲页面连成一个链，分配时可取链头的页面，去配时可将被释放的页面连入链头。此种方法适用于内存页面的分配，但对于外存页面的分配因分配和去配均需执行一次I/O传输，速度较慢。特别是当要申请多个页面时，需要进行多次I/O传输，分配效率太低。

5. 在某些虚拟页式存储管理系统中，内存永远保持一个空闲页面，这样做有什么好处？答：在内存没有空闲页架的情况下，需要按照置换算法淘汰一个内存页架，然后读入所缺页面，缺页进程一般需要等待两次I/O传输时间。若内存总保持一个空闲页架，当发生页故障时，所缺页面可以被立即调入内存，缺页进程只需等待一次I/O传输时间。读入后立即淘汰一个内存页面，此时可能也需执行一次I/O传输，但对缺页进程来说不需等待，因而提高了响应速度。

6. 为何引入多级页表？多级页表是否影响速度？答：随着内存空间和进程空间的快速增长，页表越来越大，单级页表的存放遇到困难，为此常将页表分为多级存放，即引入多级页表。多级页表会降低地址映射的速度，但通过快表可以将效率保持在合理的范畴内。

7. 与传统页表相比，倒置页表有什么优势？答：传统页表是面向进程虚拟空间的，即对应进程的每个逻辑页面设置一个表项，当进程的地址空间很大时，页表需占用很多的存储空间，造成浪费．与经典页表不同，反置页表是面向内存物理页架的，即对应内存的每个物理架设置一个表项，表项的序号就是物理页架号f，表项的内容则为进程标识pid与逻辑页号p的有序对．系统只需设置一个反置页表，为所有进程所共用.

8．允许进程空间逻辑页号不连续带来的好处是什么？ 答：可以给同一进程内的多个线程预留足够的栈空间，而又不浪费实际内存页架。

9. 比较段式存储管理与页式存储管理的优点和缺点．答：页式存储管理优缺点： (1) 静态等长存储分配简单，有效地解决了内存碎片问题； (2) 共享和保护不够方便。 段式存储管理优缺点： (1) 动态异长存储分配复杂，存在碎片问题； (2) 共享与保护方便； (3) 可以实现动态链接和动态扩展。

10． 举例说明段长动态增长的实际意义．答：允许段长动态增长对于那些需要不断增加或改变新数据或子程序的段来说很有好处。例如，分配给进程的栈空间大小，通常预先无法准确估计，若分配过少可能不够用，分配过多则造成浪费。在栈可以动态增长的情况下，系统开始可以为进程分配一个基本长度的栈空间，这个长度浪费很小。若进程运行时发生栈溢出，通过中断可以进行动态扩展。

11. 在段式存储管理中，段的长度可否大于内存的长度？在段页式存储管理中呢？答：在段式存储管理中，段的长度不能大于内存的长度，因为一个独立的段占用一段连续的内存空间，内存分配是以段为单位进行的，如果一个段的长度大于内存的长度，那么该段将无法调入内存。在段页式存储管理中，段的长度可以大于内存的长度。因为内存分配的单位是页，一个段内逻辑上连续的页面，可以分配到不连续的内存页面中，不要求一个段的所有逻辑页都进入内存。

12．共享段表的用途何在？答：共享段表的用途主要有如下两个：(1)用来寻找共享段：根据进程首次访问某段的名称在共享段表中查找，可以得知该段是否已在内存；(2)确保一个共享段只有一组描述信息：共享段的地址、长度等信息在共享段表中仅记录一次，防止在多个进程段表中重复登记所带来的维护困难。共享段表用来实现段的共享和保护，该表中记录所有共享段。多个进程共享同一段时，这些进程段表中的相应表目指向共享段表中的同一个表目。

13. 具有两级页表的页式存储管理与段页式存储管理有何差别？答：具有两级页表的页式存储管理的地址空间依然是一维的，页的划分对于进程来说是透明的。而段页式存储管理的地址空间是二维的，段的划分用户能感觉到。

14．何谓请调？何谓预调？为何在预调系统中必须辅以请调？ 答：P157。

15. 段的动态连接给共享带来什么问题？如何解决？答：动态连接提高了系统的效率，但也带来一些问题，主要是对于段共享的影响。代码段共享的必要条件是该段在运行过程中不修改自身，即要求是“纯代码”(pure code)，而动态连接需要修改连接字，这与共享的要求相矛盾。解决这个问题的一种方法是将代码段分为“纯段”和“杂段”两个部分，即将连接字等可

修改的内容存放在“杂段”中，而将其它内容放在“纯段”中。“杂段”不共享，“纯段”可共享。

16．在虚拟段页式存储管理中，考虑段的共享与段长度的动态变化，连接中断如何处理？答：由段名查本进程的段名—段号对照表及共享段表，经判断可分为如下三种情形:(1) 所有进程都未连接过（共享段表、段名-段号对照表均无）： 查文件目录找到该段； 为该段建立页表，将该段由文件全部读入swap空间，部分读入内存，填写页表；为该段分配段号，填写段名-段号对照表；如该段可共享，填写共享段表，共享记数置1；填写段表；根据段号及段内地址形成无障碍指示位的一般间接地址。(2) 其它进程连接过但本进程未连接过(共享段表有，段名-段号对照表无)： 为该段分配段号；填写段名-段号对照表，填写段表(指向共享段表)，共享段表中共享记数加1；根据段号及段内地址形成无障碍指示位的一般间接地址。(3) 本进程已连接过(共享段表无，段名-段号对照表有)：根据段号及段内地址形成无障碍指示位的一般间接地址。这里，段内地址由两部分构成，即逻辑页号和页内地址。

第七章　文件系统

1.举例说明何种文件长度是固定不变的，何种文件长度是动态变化的。 答：某些系统可执行程序，如shell、vi的长度通常是固定不变的；而用户正在编辑的文本文件或源代码文件的长度通常是动态变化的。

2.比较文件名、文件号、文件描述符之间的关系。答：文件名是文件的外部名字，通常是一个符号名(字符串)，同一文件可以有多个文件名(如通过link)。文件号是文件的内部名字，通常是一个整数，文件号与文件具有一对一的关系。文件描述符是文件打开时返回的整数(入口地址)，对应用户打开文件表(如UNIX中的u_ofile)中的一个入口。同一文件可以被多个用户同时打开，此时返回的文件描述符一般不同。同一文件也可以被同一用户多次打开，每次打开时返回的文件描述符一般也不同。

3.将文件控制块被分为两部分有何好处？此时目录项中包含那些成分？答：将文件的FCB划分为次部和主部两部分具有如下两个主要的优点： (1) 提高查找速度：查找文件时，需用欲查找的文件名与文件目录中的文件名字相比较。文件目录是存于外存的，需要以块为单位将其读入内存。 由于一个FCB包括许多信息， 一个外存块中所能保存的FCB个数较少，这样查找速度较慢。将FCB分为两部分之后，文件目录中仅保存FCB的次部，一个外存块中可容纳较多的FCB，从而大大地提高了文件的检索速度。(2) 实现文件连接：所谓连接就是给文件起多个名字，这些名字都是路径名，可为不同的用户所使用。次部仅包括一个文件名字和一个标识文件主部的文件号，主部则包括除文件名字之外的所有信息和一个标识该主部与多少个次部相对应的连接计数。当连接计数的值为0时，表示一个空闲未用的FCB主部。

4．文件在使用之前为何需要打开？多个进程共享同一文件时，其FCB为何在内存中只能保持一个副本？答：由于文件目录和文件一起存放在外存上，当存取文件时，必须先到外存中读取文件目录信息，从中得获得文件的存放地址，然后再去存取文件。这样一来，文件信息的存取将花费很多时间。如果将整个文件目录放入内存，又要占用很大内存空间，不可取。所以，文件在使用之前需要打开，目的是将系统中当前使用的文件的有关信息复制到系统打开表、用户打开文件表中，以建立用户和这个文件的联系。多个进程共享同一文件，即各自的用户打开文件表目对应系统打开文件表中的同一入口处。在系统打开

文件表的这一入口处的表目存放的唯一的FCB主部的信息，即文件号、共享计数、修改标志。这样做的好处是FCB在内存中只有一个副本，当任何一个进程对文件的操作导致FCB内容变化时，内存中的FCB内容及时得到更新，当所有进程都不再需要该文件时，即当最后一个进程关闭该文件时，才将FCB的内容回写到外存上。这样做可以减少I/O交换次数，提高系统效率。若在系统打开文件表中又存在该文件的FCB的另一副本，不但占用内存空间，另外无法更好实现多个进程共享同一文件。

5. 使用文件描述符存取打开文件与直接使用文件名相比有何优点？ 答：首先，文件名是一个字符串，操作速度慢且占空间大，而文件描述符为一整数，其处理效率明显高于字符串。其次，文件被打开后其控制信息（FCB）被缓冲到内存系统空间，文件描述符作为用户打开文件表中的入口地址直接与内存 FCB 建立起联系，而文件名无法做到这一点。

6. 用户打开文件表中包含那些内容？为何不能将其合并到系统打开表中？ 答：用户打开文件表中包含以下内容：

文件描述符	打开方式	读写指针	系统打开文件表入口

由于文件是可共享的，多个进程可能会同时打开同一文件，而其打开方式可能是不同的，当前的读写位置通常也是不一样的。如果将这些信息合并到系统打开文件表中，就会导致一个共享文件占用多个系统打开文件表表目，这些表目的大部分内容是重复的。当一个进程对文件的操作导致FCB内容变化时，该进程关闭文件时就要将FCB回写到外存。增加了内外存传输的次数，也容易导致FCB内容的不一致。因此，通常将打开方式和读写指针记录在另外一个表，即用户打开文件表中。

7. 说明对于如下文件操作命令，文件管理系统如何进行合法性检查。（1）打开文件 （2）读写文件 （3）删除文件 答：（1）打开文件 ：根据打开方式、共享说明和用户身份检查访问合法性；（2）读写文件：根据用户打开文件表中所记录的打开方式和存取方式核查访问的合法性；（3）删除文件：根据共享说明和用户身份检查访问合法性。

8. 采用文件连接技术后，文件名与文件是否一对一？文件号与文件是否一对一？文件描述符与文件是否一对一？ 答：文件名与文件是多对一。文件号与文件是一对一。文件描述符与文件是多对一。

9. 对于 Hash 文件结构，回答下述顺序探查法解决冲突方面的问题。（1）对于一个非空闲记录来说，其键值 key 的杂凑值 hash（key）是否一定与该记录地址 addr 相同？（2）当一记录的冲突记数为0时，该记录是否一定空闲？（3）当一记录空闲时，该记录的冲突记数是否一定为0？答：（1）不一定，当前面记录发生冲突时，可能在顺序探查时占用本记录。（2）不一定，可能存放冲突的记录。（3）不一定，本记录被删除后，仍可能有其它记录其 hash（key）为本记录入口地址。

10. 何谓文件连接？如何实现文件连接？答：文件连接就是给文件起多个名字，这些名字都是路径名，可为不同的用户所使用。实现文件连接时，要根据已存在的文件路径名查找目录，得到被连接文件的文件号，将该文件号与欲连接的文件名字合起来构成一个新的目录项，并填入到欲连接文件路径

第八章 设备管理课后习题答案

1. 简略叙述I/O操作的演变过程：查询方式→中断方式→通道方式，并分析对于多道程序设计所带来的影响.答：I/O操作最早为查询方式，将待传输的数据放入I/O寄存器并启动设备，然后反复测试设备状态寄存器直至完成。采用这种方式，处理机与设备之间是完全串行的。伴随设备中断处理机的能力，产生了中断I/O方式。CPU在启动设备后，可从事其它计算工作，设备与CPU并行，当设备I/O操作完成时，向CPU发送中断信号，处理机转去进行相应处理，然后可能再次启动设备传输。中断使多道程序设计成为可能：一方面中断使操作系统能够获得处理机控制权，另一方面通过I/O中断可以实现进程状态的转换。中断使处理机与设备之间的并行成为可能，但I/O操作通常以字节为单位，当设备很多时对处理机打扰很多，为此人们设计了专门处理I/O传输的处理机—通道。通道具有自己的指令系统，可以编写通道程序，一个通道程序可以控制完成许多I/O传输，只在通道程序结束时，才向处理机发生一次中断。

2. 通道与DMA之间有何共同点？有何差别？答：相同点：都以内存为中心；支持块传输。二者差别在于：通道控制器具有自己的指令系统，一个通道程序可以控制完成任意复杂的I/O传输，而DMA并没有指令系统，一次只能完成一个数据块传输。

3. 下述设备的物理地址各是几维的？为什么在I/O操作之前通常使用一维地址？一维地址与多维地址之间的对应关系是由硬件确定的，还是由操作系统确定的？(1) 磁盘组 (2) 磁带 (3) 光盘答：(1) 磁盘组的物理地址是三维的 (2) 磁带的物理地址是一维的 (3) 光盘的物理地址是二维的。为了屏蔽设备的具体特性，在I/O操作之前通常使用一维地址，一维地址与三维地址之间的对应关系通常由操作系统设计者来确定

4. 磁带机为何不适合用作共享型设备?答：磁带机属于启停型设备，即使用时开启，不用时停止。用来记录信息的磁带一般很长，磁带上块号顺序编址，块的随机定位涉及磁带的机械运动，速度很慢，随机访问不连续的磁带块会造成较大的时间开销，这决定磁带属于顺序访问设备。顺序访问设备不适合用作共享型设备。

5. 在下述三种类型通道中，哪种类型支持通道程序的并发执行？ (1) 字节多路通道 (2) 数据选择通道 (3) 数组多路通道答：(1) 字节多路通道(byte multiplexor channel)：通道中含有许多非分配型子通道，每个子通道所连接的 I/O 设备以字节为单位，分时间地与通道交换数据，主要用于连接低速 I/O 设备，通道程序可以并发执行。(2) 数组选择通道(block selector channel)：其所连的 I/O 设备是以块为单位与通道交换数据。用于连接多台高速设备，但其中只有一个分配型子通道，在一段时间内只能执行一道通道程序。(3) 数组多路通道 (block multiplexor channel) ：所连外部设备以块为单位与通道交换数据。 用于连接多台高速设备，允许几个通道程序分时并行工作。

6. 用户申请独占型设备为何不指定具体设备, 而仅指定设备类别? 答: **p209-2107.** 为何不允许用户程序直接执行设备驱动指令?答: (1)系统中的设备可能被多个进程所共享，例如磁盘就是这样的设备。如果允许用户程序直接执行设备驱动指令，那么就有可能损坏设备。(2)设备操作涉及很复杂的驱动过程，一般用户编写驱动程序是很大的负担，操作系统的目标是方便用户使用计算机系统，因而提供标准驱动程序。

8. 何谓"磁道歧视"? 假设每个磁道各有一个磁头, 是否还存在磁道歧视问题?答: **p212** 假设每个磁道各有一个磁头, 不存在磁道歧视问题

9. 处理机与通道之间是如何通讯的? 通道与处理机之间呢?答: 通道与处理机之间相对独立，通道程序的执行可与处理机的操作并行; 因为一个系统中可能有多个通道，这些通道也可并行地执行相应的通道程序。通常，通道程序形成之后，处理机将通道程序的起始地址放到内存指定单元处，然后执行通道启动指令使通道开始工作。通道被启动之后由指定单元取来通道程序的起始地址, 并放入通道地址字CAW中，由此依次地执行各条通道指令。当通道程序执行完毕，或执行到通道结束指令时，产生通道中断信号，该信号发给处理机，处理机响应中断后取出中断字，分析中断原因并进行相应的中断处理。

10. 说明下列术语之间的对应关系 (1) I/O设备 (2) I/O驱动程序 (3) I/O进程
答: 完成I/O操作功能的程序叫做I/O驱动程序，设备驱动程序与设备类型是一一对应的。 I/O进程可分为三种方式实现: a.每类设备设一个专门的I/O进程,且该进程只能在系统状态下执行。 b.整个系统设一个I/O进程，全面负责系统的数据传送工作 c.每类设备设一个专门的I/O进程,但该进程即可以在系统状态下执行也可以在用户态下执行

11. 什么叫缓冲(buffering)? 缓冲与高速缓存(caching)有何差别?答: 利用存储区缓解数据到达速度与离去速度不一致而采用的技术称为缓冲，此时同一数据只包含一个拷贝。例如，操作系统以缓冲方式实现设备的输入和输出操作主要是缓解处理机与设备之间速度不匹配的矛盾，从而提高资源利用率和系统效率。 缓存是为提高数据访问速度而将部分数据由慢速设备预取到快速设备上的技术，此时同一数据存在多个拷贝。例如，远程文件的一部分被取到本地。 当然，在有些情况下，缓冲同时具有缓存的作用。例如 UNIX 系统对于块型设备的缓冲区，在使用时可保持与磁盘块之间的对应关系，既有缓冲的作用也有缓存的作用，通过预先读与延迟写技术，进一步提高了 I/O 效率 。

12. 与为每个设备配置一个(或若干个)缓冲区相比, 采用可为多个设备共用的缓冲池有何优点? 答: **p214**

13. 在系统中缓冲区空间总长度固定的前提下，一个缓冲区过大或过小各有何优点

和缺点?答：缓冲区过大会造成资源浪费（平均浪费半个缓冲区容量)，但是能减少I/O传输次数；缓冲区过小则会因I/O传输次数增多而增加系统开销，另外缓冲区过小会引缓冲链指针过多而浪费缓冲空间。

14. 假设要修改某一磁盘块上的一部分，而其它部分保持原内容不变，应当如何做?答：首先将该磁盘块内容读入内存缓冲区，然后在内存中修改相关内容，最后将修改后的缓冲区完整地回写到磁盘中。

15. 某磁盘组共有200个柱面，由外至内依次编号为0,…,199．I/O请求以10,100,191,31,20,150,32的次序到达，假定引臂当前位于柱面98处，对FCFS,SSTF,SCAN,C-SCAN,LOOK,C-LOOK 引臂调度算法分别给出寻道示意图，并计算总移动量．对SCAN和LOOK算法，假定引臂当前移动方向由外向内．对LOOK算法假定回扫方向由内向外．（略）

16. 假设没有输入井，可采用何种作业调度算法?答：没有输入井，可采用先到先服务的作业调度算法。

17. 为什么提出RAID技术？RAID技术如何提高I/O性能？答: 这是因为硬件技术发展具有一定的不平衡性．处理机速度提高很快，通常以指数数量级别，大约每隔18个月速度提高一倍，而磁盘速度的提高相对极其缓慢。磁盘的速度瓶颈是磁头引臂的移动速度。解决问题的出路是并行部件和并行存取技术，多磁盘同时操作。RAID是一个物理磁盘的集合，作为一个逻辑磁盘管理和使用。数据被分散存于多个物理磁盘上，校验等冗余信息用于提高信息存取的可靠性。

18. RAID level5是常用的RAID级别，分析其性能和实现代价；答：空间利用率（N-1）/N，利用率较高　　可靠性高，访问速度快，有容错能力　　RAID5的控制比较复杂，尤其是利用硬件对磁盘阵列的控制，因为这种方式的应用比其他的RAID level要掌握更多的事情，更多的输出／入需求，既要速度快，又要处理数据，计算校验值，做错误 校正等，所以实现代价较大

19. 独占型设备利用率低的原因何在 ? 虚拟技术为何能提高独占型设备的利用率 ? 输入型和输出型虚拟设备各是如何实现的 ?答：用户直接使用独占型设备的方式是在申请命令与释放命令之间进程将一直占用所申请到的独占型设备，这种设备使用方式有如下两个主要缺点： (1)由于独占型设备速度较慢，进程在执行使用命令时需要花费较长时间等待 I/O 传输完成，因而影响进程推进速度； (2)由于在各个使用命令之间可能夹杂着与该设备无关的操作(如计算、操作其它设备等)，进程在占有该设备的期间内不一定一直使用该设备，因而降低了设备的利用率。 为克服上述缺点，引入了虚拟设备。利用共享型设备实现的数量较多、速度较快的独占型设备称为虚拟设备。其基本思想是在独占型设备与内存进程之间加入一个共享型设备

作为过渡，因为共享型设备速度很快，所以进程 I/O 传输所需等待时间较短，提高了进程推进速度。另外由于信息在独占型设备与共享型设备之间的传输是连续进行的，即独占型设备被占用期间一直被使用，因而提高了设备资源的利用率。 (1）输入型虚拟设备实现 对于输入型虚拟设备来说，信息的流向是由独占型设备到共享型设备，再由共享型设备到进程空间。假定用于输入的独占型设备为读卡机，用于实现虚拟设备的共享型设备是磁盘。 对于进程所发出的申请命令、使用命令及释放命令，系统所需完成的工作如下： 申请：分配一台虚拟设备 (盘区)，分配一台实设备(读卡机)，将信息由实设备(读卡机)连续地传输到虚拟设备(盘区)，释放实设备(读卡机)。 使用：将信息由虚拟设备 (盘区)传输到进程空间。释放：收回虚拟设备 (盘区)。 (2）输出型虚拟设备实现 对于输出型虚拟设备来说，信息的流向是由进程空间到共享型设备，再由共享型设备到独占型设备 . 假定用于输出的独占型设备为打印机，用于实现虚拟设备的共享型设备是磁盘. 对于进程所发出的申请命令、使用命令及释放命令，系统所需完成的工作如下：申请：分配一台虚拟设备 (盘区)。使用：将信息由进程空间传输到虚拟设备 (盘区)。释放：分配一台实设备 (打印机)，信息全部由虚拟设备传输到实设备 (打印机)，收回实设备 (打印机)，收回虚拟设备 (盘区域)。20.什么是稳定存储器？为什么需要稳定存储器？如何实现稳定存储器？答案：p222