计算机操作系统主观题

1．采用多道程序设计的主要优点是什么?

答：在单道运行方式下，每当程序发出 I/O 请求时，CPU 便处于等待 I/O 完成的状态，致使 CPU 空闲。多道程序设计考虑到作业的运行规律是交替使用 CPU 和 I/O，故将多道程序同时保存于系统中，使各作业对 CPU 和 I/O 的使用在时间上重叠，提高了 CPU 和 I/O 设备的利用率。

2．操作系统是随着多道程序设计技术的出现逐步发展起来的，要保证多道程序的正确运行、在技术上要解决哪些基本问题?

答：多道程序设计技术能有效提高系统的吞吐量和改善资源利用率。实现多道程序系统时，由于主存中总是同时存在几道作业，因而需要妥善解决下述几个问题：

（1）处理机管理问题。应如何分配被多道程序共享的处理机，以使处理机既能满足各程序运行的需要又有较高的利用率；当把处理机分配给某程序后，应何时收回处理机。

（2）内存管理问题。如何为每道程序分配必要的内存空间，使它们各得其所又不致因相互重叠而丢失信息；应如何防止因某道程序出现异常情况而破坏其他程序。

（3）设备管理问题。系统中可能有多种类型的 I/O 设备供多道程序共享，应如何分配这些 I/O 设备；如何做到既方便用户对设备的使用。又能提高设备的利用率。

（4）文件管理问题。在现代计算机系统中，通常都存放着大量的程序和数据信息，应如何组织信息才能便于用户使用并能保证数据信息的安全性和一致性。

3．实现多道程序系统的最主要硬件支持是什么？

答：最主要硬件支持是中断系统和通道技术。

（1）很多进程的切换是由时钟中断引起的，尤其是分时系统。用户程序进行系统调用时通过软中断来实现，如 TRAP　通道和外设的操作也要向操作系统发送中断。

（2）在多道程序系统中，当 CPU 要求在主存和外设间传输数据时，通过发出 I/O 指令命令通道工作，通道独立地在内存和外设间进行数据传输，I/O 操作完成后，通道以中断方式通知 CPU，从而实现了 CPU 计算与 I/O 操作的并行。

4．叙述操作系统在计算机系统中的位置。

答：操作系统是运行在计算机硬件系统上的最基本的软件。它控制和管理着所有的系统硬件(CPU、主存、各种硬件部件和外部设备等)，也控制和管理着所有的软件(系统程序和用户进程等)　操作系统为计算机使用者提供了一种良好的操作环境，也为其他各种应用系统提供了最基本的支撑环境。现代操作系统是一个复杂的软件系统. 它与计算机硬件系统有着千丝万缕的联系，也与用户有着密不可分的关系，它在计算机系统中位于计算机硬件和计算机用户之间，如图 1.2 所示。紧挨着硬件的就是操作系统，它通过系统核心程序对计算机系统中的几类资源进行管理，如处理机、存储器、输入／输出设备、数据与文档资源、用户作业等，并向用户提供若干服务，通过这些服务将所有对硬件的复杂操作隐藏起来，为用户操供一个透明的操作环境。操作系统是最基本的系统软件。操作系统的外层是其他系统软件，用户可以直接通过系统软件层与计算机打交道，也可以建立各类应用软件和应用系统，通过它们来解决用户的问题。

由此可见，操作系统是介于计算机硬件和用户之间的一个接口。

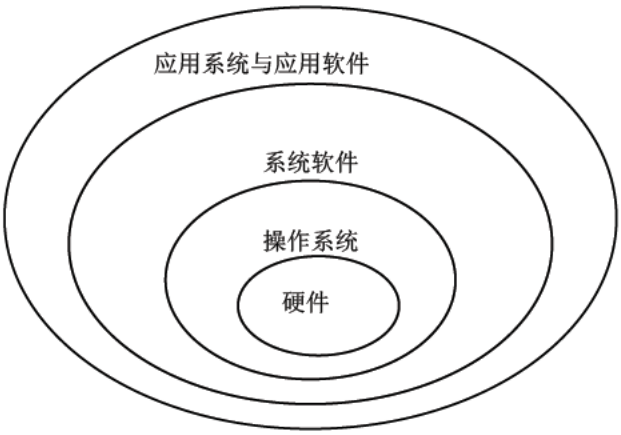

图 1.2 操作系统在计算机系统中的位置

5．处理机为什么要区分管态和目态（系统态和用户态）？

答：为了防止操作系统及关键数据受到用户程序有意或无意的破坏，通常将处理机的执行状态分成管态和目态（系统态和用户态）两种。处于目态执行的程序的操作要受到限制，不能去执行特权指令，访问操作系统区域和其他程序的区域，这就防止了用户程序对操作系统和其他用户程序的破坏。操作系统的内核通常是运行在管态（系统态）的，目态（用户态）的程序通过系统调用接受管态程序运行的服务

目态下的进程能存取它们自己的指令与数据，但不能存取内核指令和数据或其他进程的指令和数据。然而，管态下的进程能够使用所有指令、资源，并具有改变 CPU 状态的能力。在目态下执行的进程没有执行特权指令的能力，在目态下执行特权指令会引起错误。

从目态转换为管态的惟一途径是中断；从管态到目态的转换通过修改程序状态字来实现。

6．网络操作系统与分布式操作系统的区别？

答：在计算机网络中，可根据网络结构、通信方式和资源管理方法配置网络操作系统和分布式操作系统。在配置了网络操作系统的计算机网络中，各计算机没有主次之分；网络中任意两台计算机可以进行信息交换；网络 OS 中的用户使用自己的机器可以访问网络上别的机器的资源，通过网络将很多的机器连接起来，共享软硬件资源，但是整个系统对用户来说是分散的、不透明的。而分布式计算机是由多台计算机组成的一种特殊的计算机网络，分布式操作系统能使系统中的若干台计算机相互协作完成一个共同的任务，使一个程序分布在几台计算机上并行执行、互相协作得出最终的计算结果，但是整个系统对用户是透明的，用户面对整个 OS 就好像使用一个自己的机器一样。

7. 多用户分时系统如何克服多道批处理系统的缺点 ？

答:尽管多道批处理系统已经大大地提高了计算机系统的资源利用率，但是它的致命缺点是缺少交互性。怎样才能使系统既具有交互性又不使资源的利用率降低？资源利用率与交互性是一对矛盾。如果一台计算机能够连接多个操作台（终端），允许多个用户同时在操作台上操作，每个操作台上的用户执行一个程序，就有多个程序进入系统，导致在计算机的内

存中就装入了多个程序，形成多个程序的并发执行,通过并发程序的分时执行，确保每个用户的操作计算机终端就好像单独操作一台计算机一样。这样就避免了只有一个操作台时，大量的计算机的时间被一个用户的大量浪费，同时又克服多道批处理系统非交互性的缺点。

8.AB 两个程序，程序 A 按顺序使用 CPU10s，使用设备甲 10s，使用 CPU5s，使用设备乙 10s ,最后使用 CPU10s。程序 B 按顺序使用设备甲 10s，使用 CPU10s，使用设备乙 10s，使用 CPU5s，使用设备乙 10s，问：

（1）在顺序环境下先执行程序 A 再执行程序 B，CPU 的利用率是多少？

（2）在多道程序环境下，CPU 的利用率是多少？

答：（1）程序 A 和程序 B 顺序执行，程序 A 执行完毕，程序 B 才开始执行。两个程序共耗时 90s，其中占用 CPU 的时间为 40s，因此顺序执行时 CPU 的利用率为 44.4%

（2）在多道程序环境下，两个程序并发执行，执行情况如图 1.3 所示，两个程序共耗时 50s，其中占用 CPU 的时间为 40s，故此时 CPU 的利用率为 40/50=80%。

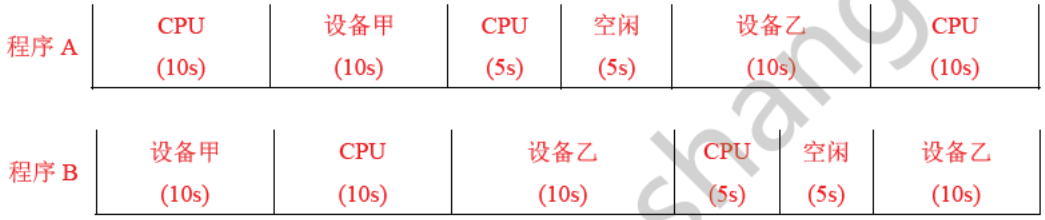

图 1.3 多道环境下 A、B 执行示意图

9．试对分时系统和实时系统进行比较。

分析与解答

我们可以从以下几个方面对这两种操作系统进行比较。

（1）从多路性看，实时信息处理系统与分时系统一样都能为多个用户服务。系统按分时原则为多个终端用户服务；而对实时控制系统，则表现为经常对多路现场信息进行采集以及对多个对象或多个执行机构进行控制。

（2）从独立性看 实时信息处理系统与分时系统一样，每个用户各占一个终端，彼此独立操作，互不干扰 因此用户感觉就像他一人独占计算机；而实时控制系统中信息的采集和对对象的控制都是彼此互不干扰的。

（3）从及时性看，实时信息系统对响应时间的要求与分时系统类似，都是以人们所能接受的等待时间来确定；而实时控制系统的响应时间则是以控制对象所能接受的延时来确定的。

（4）从交互性看，分时系统是一种通用性系统，主要用于运行终端用户程序，因此它具有较强的交互能力；而实时系统虽然也有交互能力，但其交互能力不及前者。

（5）从可靠性看，分时系统也要求系统可靠，相比之下，实时系统则要求系统高度可靠。

10．一个分层结构操作系统由裸机，用户，CPU 调度和 P、V 操作，文件管理，作业管理，内存管理，设备管理，命令管理等部分组成。试按层次结构的原则从内到外将各部分重新排列。

分析与解答

采用分层结构方法可以将操作系统的各种功能分成不同的层次，即将整个操作系统看成是由若干层组成，每一层都提供一组功能，这些功能只依赖于该层以内的各层次，最内层部分是机器硬件本身提供的各种功能。操作系统的这种层次结构如图 1.1 所示，同机器硬件紧挨着的是操作系统内核，它是操作系统的最里一层。内核包括中断处理、设备驱动、处理机调度以及进程控制和通信等功能，其目的是提供一种进程可以存在和活动的环境。内核以外各层依次是存储管理层、I/O 管理层、文件管理层和作业管理层。它们提供各种资源管理功能并为用户提供各种服务。命令管理是操作系统提供给用户的接口层，因而在操作系统的最外层。

从上述分析可知，按层次结构原则从内到外依次为：裸机，CPU 调度和 P、V 操作，内存管理，设备管理，文件管理，作业管理，命令管理，用户。

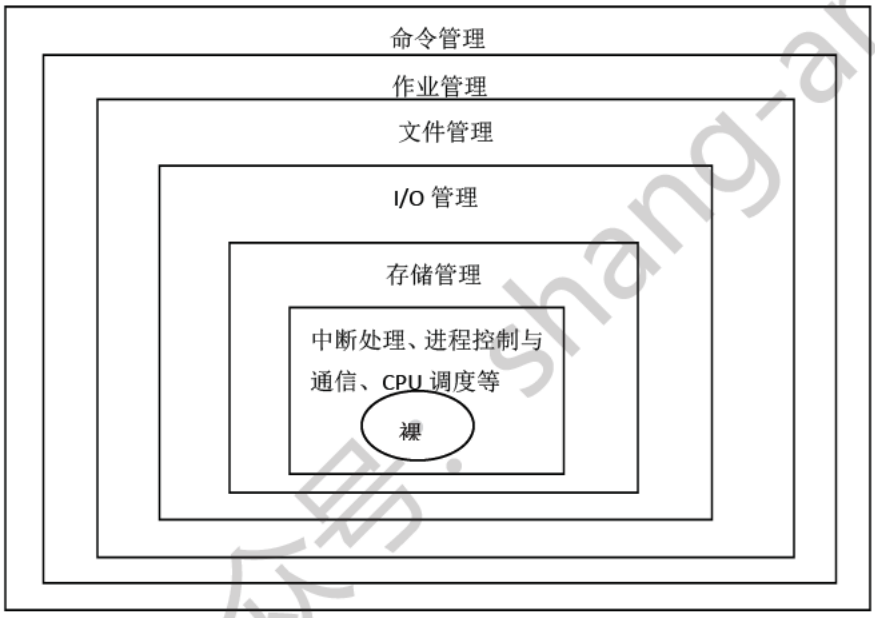

图 1.1 操作系统的层次结构

11．操作系统具有哪些特征?它们之间有何关系?

分析与解答

操作系统的特征有并发、共享、虚拟和异步性（不确定性）。它们的关系如下：

（1）并发和共享是操作系统最基本的特征。为了提高计算机资源的利用率，操作系统必然要采用多道程序设计技术，使多个程序共享系统的资源，并发的执行。

（2）并发和共享互为存在的条件。一方面，资源的共享以程序（进程）的并发执行为条件，若系统不允许程序并发执行，自然不存在资源的共享问题；另一方面，若系统不能对资源共享实施有效管理，协调好各个进程对共享资源的访问，也必将影响到程序的并发执行，甚至根本无法并发执行。

（3）虚拟以并发和共享为前提条件。为了使并发进程能更方便、更有效地共享资源，操作系统经常采用多种虚拟技术来在逻辑上增加 CPU 和设备的数量以及存储器的容量，从而解决众多并发进程对有限系统资源的竞争问题。

（4）异步性（不确定性）是并发和共享的必然结果。操作系统允许多个并发进程共享资源、相互合作，使得每个进程的运行过程受到其他进程的制约，系统中的每个程序何时执行，多个程序间的执行顺序以及完成每道程序所需的时间是不确定的，因而也是不可预知的。

12. 分时系统需要使用下面哪些成份:

①多道程序设计技术 ②作业说明书 ③终端命令解释程序 ④中断处理

⑤优先级调度 ⑥系统调用

分析与解答 ① ③ ④ ⑥

13. 你认为下列哪些指令在核心态下执行？

①屏蔽所有中断 ②读时钟日期 ③设置时钟 ④改变存储映像图

⑤存取某地址单元的内容 ⑥停机

分析与解答：

操作系统程序在核心态下运行，发生系统调用时都转入核心态运行，系统调用大致分为如下几类：（1）文件操作：打开/删除文件，读写文件，建立文件；（2）资源申请：申请/释放存储空间，申请/释放外围设备；（3）控制：正常/异常结束，返回断点/指定点；（4）信息维护：设置、获取日期时间、设置获取文件属性等。

所以应在核心态下执行的指令是① ② ③ ④ ⑥

14. 为什么说直到中断和通道技术出现，多道程序概念才变为有用？

分析与解答：

通道是一种专业 I/O 处理机，它一旦被启动就独立于 CPU 运行，故做到了输入输出与 CPU 并行工作，但早期 CPU 向通道发询问指令来了解通道工作是否完成，若未完成，则循环询问，无法做到 CPU 与 I/O 设备真正并行工作。

中断是在输入输出结束或硬件发生某种故障时，由相应硬件（即中断机构）向 CPU 发出信号，CPU 立即停止手头的工作而转向处理中断请求，待处理完中断后再继续原来手头的工作。

CPU 启动通道，通道工作结束时，通过中断机构向 CPU 发中断请求。

所以说，直到中断和通道技术出现，多道程序概念才变为有用。

15. 进程和线程的主要区别是什么?

答：主要从调度、并发性、系统开销、拥有资源等方面来比较线程和进程：

（1）调度。在传统的操作系统中，独立调度、分派的基本单位是进程。引入线程的操作系统中，则把线程作为调度和分派的基本单位。

（2）并发性。在引入线程的操作系统中，不仅进程之间可以并发执行，而且在一个进程中的多个线程之间亦可并发执行，因而使操作系统具有更好的并发性，从而能更有效地使用系统资源和提高系统吞吐量。

（3）拥有资源。不论是传统的操作系统，还是设有线程的操作系统，进程都是拥有资源的一个独立单位，它可以拥有自己的资源。一般地说，线程自己不拥有系统资源(只有一点运行时必不可少的资源)，但它可以访问其隶属进程的资源。

（4）系统开销。由于在创建、撤销或切换进程时，系统都要为之分配或回收资源，保存 CPU 现场。因此，操作系统所付出的开销将显著地大于创建、撤销或切换线程时的开销。在进程切换时，涉及到整个当前进程 CPU 环境的保存及新调度到的进程 CPU 环境的设置；而线程切换时，只需保存和设置少量寄存器内容，因此开销很少。另外，由于同一进程内的多个线程共享进程的地址空间，因此，这些线程之间的同步与通信非常容易实现，甚至无需操作系统的干预。

16. 什么是进程控制块？它有什么作用？

答：进程控制块 PCB 是一个记录进程属性信息的数据结构，是进程实体的一部分，是

操作系统最重要的数据结构。

当操作系统要调度某进程执行时，需要从该进程的 PCB 中查询其现行状态和优先级调度参数；在调度到某进程后，要根据其 PCB 中保存的处理机状态信息去设置和恢复进程运行的现场，并根据其 PCB 中的程序和数据的内存地址来找到其程序和数据；进程在执行过程中，当需要与其他进程通信时，也要访问其 PCB；当进程因某种原因而暂停执行时，又需要将断点的现场信息保存在其 PCB 中。系统在建立进程的同时就建立了该进程的 PcB，在撤销一个进程时也就撤销其 PCB。由此可知，操作系统根据 PCB 来对并发执行的进程进行控制和管理，PCB 是进程存在的惟—标志。

17．用户级线程和内核支持线程有何区别？

答：两者的区别是：

（1）内核支持线程是 OS 内核可感知的，而用户级线程是 OS 内核不可感知的

（2）用户级线程的创建、撤消和调度不需要 OS 内核的支持，是在语言（如 Java）这一级处理的；而内核支持线程的创建、撤消和调度都需 OS 内核提供支持，而且与进程的创建、撤消和调度大体是相同的。

（3）用户级线程执行系统调用指令时将导致其所属进程被中断，而内核支持线程执行系统调用指令时，只导致该线程被中断。

（4）在只有用户级线程的系统内，CPU 调度还是以进程为单位，处于运行状态的进程中的多个线程，由用户程序控制线程的轮换运行；在有内核支持线程的系统内，CPU 调度则以线程为单位，由 OS 的线程调度程序负责线程的调度。

（5）用户级线程的程序实体是运行在用户态下的程序，而内核支持线程的程序实体则是可以运行在任何状态下的程序。

18．按序分配是防止死锁的一种策略。什么是按序分配？为什么按序分配可以防止死锁？

答：按序分配是适应于动态分配的一种分配方法。为了避免产生死锁，系统将所有资源进行编号，并规定进程请求资源时，严格按照设备编号的大小，比如由小到大的顺序进程申请。如果某进程第 n 号资源没有获得，则进程不能请求第 j（j>n）号资源。（系统也可以规定由大到小的请求次序。）

按序分配可以破坏环路等待条件。因为在进程发生死锁时，必然存在一个“进程-资源”的环形链。要排除循环的产生，让进程 Pi 先请求较小编号的资源，如果不能满足就不准请求较大编号的。这样一来，较小编号的资源如用于其他进程，Pi 进程申请不到，也就不能申请高号资源，因此形成不了循环，也就不会产生死锁。

19．何谓临界区？下面给出的两个进程互斥的算法是安全的吗？为什么？

```
#define true;
#define false;
int flag[2];
flag[1]=flag[2]=false;
enter-crtsec(i)
int i;
{
while (flag[1-i]);
```

```
flag[i]=true;
}

leave-crtsec(i)
int i;
{
flag[i]=false;
}

process i:
…
enter-crtsec(i);
In critical section;
Leave-crtsec(i);
```

答：一次仅允许一个进程使用的资源称为临界资源，在进程中对临界资源访问的程序段称为临界区。从概念上讲，系统中各进程在逻辑上是独立的，它们可以按各自的速度向前推进。但由于它们共享某些临界资源，因而产生了临界区问题。对于具有临界区问题的并发进程，它们之间必须互斥，以保证不会同时进入临界区。

这种算法不是安全的。因为，在进入临界区的 enter crtsec()不是一个原子操作，如果两个进程同时执行完其循环(此前两个 flag 均为 false)，则这两个进程可同时进入临界区。

20．某车站售票厅，任何时刻最多可容纳 20 名购票者进入，当售票厅中少于 20 购票者时，则厅外的购票者可立即进入，否则需在外面等待。若把一个购票者看作一个进程，请用 P、V 操作管理这些并发进程时，应怎样定义信号量？写出信号量的初值以及信号量各种取值的含义。

答：设置一个信号量 S，表示售票厅里还可以容纳的人数，初值为 20。每个购票者的描述如下：

```
Semaphore S=20;
Buyer()
{
wait(S);
进入售票厅；
购票；
退出售票厅；
signal(S);
}
```

21．理发店里有一位理发师、一把理发椅和 N 把供等候理发的顾客坐的椅子。如果没有顾客，理发师便在理发椅上睡觉。当一个顾客到来时，它必须叫醒理发师。如果理发师正在理发时又有顾客来到，如果有空椅上可坐，顾客就坐下来等待，否则就离开理发店。

答：本题使用 3 个信号量和一个控制变量：控制变量 waiting 用来记录等候理发的顾客数，初值为 0；信号量 customers 用来记录等候理发的顾客数，并用作阻塞理发师进程，初值为 0；信号量 barbers 用来记录正在等候顾客的理发师数，并用作阻塞顾客进程，初值为 0；信号量 mutex 用于互斥，初值为 1。同步算法描述如下：

```
semaphore   customers=0；/*等候理发的顾客数*/
semaphore   barberers=0；/*等候顾客的理发师数*/
semaphore   mutex=1；
int   waiting=0；
main()
{ cobegin
   barbers();
   customers();
   coend
}
barber()
{
while(true)
{
 Wait(customers)；        /*是否有等候的顾客*/
 Wait(mutex);
Waiting=waiting-1；      /*顾客数减 1*/
Signal(barbers)；        /*理发师开始理发*/

Signal(mutex)；
理发；
}
}
Customer()
{
Wait(mutex)；
If(waiting<N)
 {
Waiting=waiting+1；      /*若有空椅子，等候的顾客数加 1*/
   Signal(customers)；
   Signal(mutex)；
   Wait(b  rbers)；
   坐下等候服务；
   }
   Else
   {
signal(mutex)；          /*无空椅子则离开*/
   }
   }
```

22．设公共汽车上，司机和售票员的活动分别为：司机的活动为启动车辆，正常行车，到站停车；售票员的活动为关车门，售票，开车门。用 wait、signal 操作实现他们间的协调操作。

答：司机和售票员在车上的操作的同步关系为：只有售票员关门后，司机才能启动开始

行驶汽车；只有司机停车后，售票员才能开门让乘客上下车；可见，售票员关车门后，要向司机发开车信号，司机接到开车信号后才能启动车辆。在汽车正常行驶过程中售票员售票，到站时司机停车，售票员在车停后开车门，让乘客上下车。因此司机启动车辆的动作必须与售票员的动作取得同步；售票员开车门的动作也必须同司机停车取得同步。

根据同步规则以及操作流程确定信号量的个数是 2 个，S1 和 S2。

S1 的表示是否允许司机启动汽车，初值为 0；S2 表示是否允许售票员开门，初值为 0。

```
semaphore S1=0;
semaphore S2=0;
main()
{
cobegin
   driver();
   busman();
coend
}
driver()
   {
      While (true)
      { wait(S1);
         启动车辆；
         正常行车；
         到站停车；
         signal(S2);
       }
   }
busman ()
      {
       While (true)
      { 关车门
         signal(S1);
         售票
         w  it(S2);
         开车门；
         上下乘客；
      }
   }
```

23．多个进程共享一个文件，其中只读文件的称为读者，只写文件的称为写者，读者可以同时读，但写者只能独立写，并要求对写着优先，即一旦有写者到达，后续的读者必须等待，而不论是否有读者在读文件。

答：本题是经典读者写者问题，只是增加了一些限制条件，即算法对写者优先。为了提高写者的优先级，在原先经典读者写者问题的算法上，增加一个信号量 s，用于在写进程到达后封锁后续的读者，算法描述如下：

```
semaphore   rmutex= 1;
semaphore   wmutex =1;
semaphore   s =1;
int         readcount=0;
 reader( )
{
    While(1)
    {
        wait(s);
        wait(rmutex);
         if   (readcount= =0)
              wait(wmutex);
              readcount++;
              signal(rmutex);
              signal(s);
                 …                                    …
              perform read operation;
                …                                   …
              wait(rmutex);
              readcount--;
            if (readcount= =0)
             signal(wmutex);
              signal(rmutex);
        }
}
 writer()
{
  while(1)
  {
      wait（s）
      wait(wmu ex);
      perform write operation;
      signal(wmutex);
      signal(s);
  }
}
main()
{
  cobegin
  {
    reader();
    writer();
  }
 coend
```

}

24．一个计算机系统中拥有 6 台打印机，现有 N 个进程竞争使用，每个进程要求两台，试问，N 的值如何选取时系统中绝对不会出现死锁？

答：本题的考核要点是资源竞争与死锁问题。已知系统中的每个进程需要两台打印机，那么在最坏的情况下，各进程都占用了其中的一台，而且都在请求自己所需的另一台。如果此时系统尚有多余的一台，那么就可以满足其中一个进程运行完毕。因此说，如果 N 的值达到 5，每个进程分得一台打印机，还有一台剩余的打印机，把这台打印机分给任何一个进程，都能让这个进程运行完毕。当该进程运行完毕释放出它所占有的打印机后，又可以进一步满足其他进程。系统就不会出现死锁，因此 N 的值最大可取到 5。

25．有三个进程 P1、P2 和 P3 并发工作。进程 P1 需要资源 S3 和 S1 各一个；进程 P2 需用资源 S1 和 S2 各一个；进程 P3 需用资源 S2 和 S3 各一个，S1、S2、S3 的资源个数都是 1，回答：

（1）若对资源分配不加限制，会发生什么情况？为什么？

（2）为保证进程正确地工作，应采用怎样的资源分配策略？为什么？

答：（1）若对进程间的资源分配不加限制，可能会发生死锁。因为这样的分配可能导致进程间的“循环等待”，并且这种状态将永远持续下去。进程 P1、P2 和 P3 分别获得资源 S3、S1 和 S2，后再继续申请资源时都要等待。进程和资源会形成如下环路：

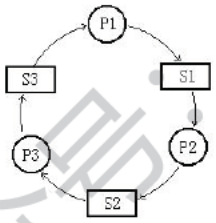

图 2.7　进程资源分配图

（2）为保证系统处于安全状态，应采用下面列举 3 种资源分配策略：

1）采用静态分配：由于执行前已获得所需的全部资源，故不会出现占有资源又等待的资源的现象（或不会出现循环等待资源现象）。

2）采用资源按序分配，避免出现循环等待资源的现象。

3）采用银行家算法进行分配资源前的检测。

26　现有两道作业同时执行，一道以计算为主，另一道以输入输出为主，你将怎样赋予作业进程占有处理器的优先级？为什么？

答：本题考核要点是，如何提高系统效率的问题。我们知道，以计算为主的进程运行期间，将主要集中在 CPU 的计算上，较少使用外部设备。而以输入输出为主的进程则主要集中在外部设备的 I/O 上，较少使用 CPU。因此让两个进程并发运行是可以提高系统效率的。不过它们的优先级应当设定合理。

如果计算进程的优先级高于或者等于输入输出进程的优先级，系统效率不会提高。因为计算进程一旦占用了 CPU 便忙于计算，使输入输出进程得不到运行机会，同样会使设备空闲，不能提高系统效率。

如果输入输出进程的优先级高于计算进程的优先级，系统效率就能够得到提高。因为输入输出操作是一种速度极慢的操作。若该项操作的优先级高，那么，当它完成一项输入输出操作后，便能立即获得 CPU，为下一次输入输出作准备工作，并启动外部设备。当设备被

启动起来后，它便主动让出 CPU，由系统将 CPU 交给计算机进程使用。从而获得较好的运行效率。

因此，将赋予以输入输出为主的进程优先级高。

27．有三个作业 A、B、C，到达时间和运行时间如下表。在单道批处理系统按照响应比高者优先算法进行调度，计算平均周转时间和平均带权周转时间。

作业号	到达时间	运行时间
A	8.5	1.5
B	9	0.4
C	9.5	1

答：响应比=1+作业等待时间/作业运行时间

在 8.5 时，只有作业 A 到达，系统将作业 A 投入运行。作业 A 运行 1.5 小时后于 10 时完成，这时作业 B、C 均已到达，此时作业 B、C 的响应比为

R_B=1+(10-9)/0.4=3.5

R_C=1+(10-9.5)/1=2.5

作业 B 的响应比高，所以 B 先运行，因此这 3 个作业的运行顺序是 A、B、C，运行情况如表所示：

作业号	到达时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	8.5	1.5	8 5	10	1.5	1
B	9	0.4	10	10.4	1.4	3.5
C	9.5	1	10.4	11.4	1.9	1.9
平均周转时间 T=(1.5+1.4+1.9)/3=1.6 平均带权周转时间 W=(1+3.5+1.9)/3=2.13						

28．假定某计算机系统有 R1 和 R2 两类可以使用资源(其中 Rl 有两个单位，R2 有一个单位)，它们被进程 P1 和 P2 所共享，且已知两个进程均以下列顺序使用两类资源：

申请 R1→申请 R2→申请 R1→释放 R1→释放 R2→释放 Rl

试求出系统运行过程中可能到达的死锁点，并画出死锁的资源分配图(或称进程一资源图)。

答：当两个进程都执行完第 1 步后，即进程 P1 和进程 P2 都申请了一个 R1 类资源时，系统进入不安全状态。随着两个进程的向前推进，无论哪个进程执行完第 2 步，系统都将进入死锁状态。可能达到的死锁点是进程 P1 占有一个单位的 Rl 类资源及一个单位的 R2 类资源，进程 P2 占有一个单位的 R1 类资源，此时系统内已无空闲资源，而两个进程都在保持已占有资源不释放的情况下继续申请资源，从而造成死锁；或进程 P2 占有一个单位的 Rl 类资源及一个单位的 R2 类资源，进程 P1 占有一个单位的 RI 类资源，此时系统内也已无空闲资源，而且两个进程都在保持已占有资源不释放的情况下继续申请资源，从而造成死锁。

假定进程 P1 成功执行了第 2 步，则死锁点的资源分配如图 2.8 所示所示。

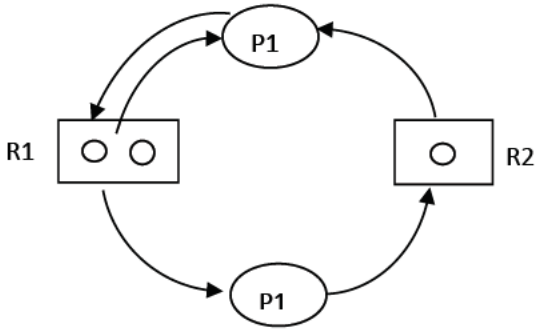

图 2.8 死锁点的资源分配图

29．当前系统中出现下述资源分配情况：

	Allocation	Need	Available
P0	0 0 3 2	0 0 1 2	1 6 2 2
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	

利用银行家算法，试问：

（1）该状态是否安全？

（2）如果进程 P2 提出资源请求 Request(1,2,2,2)后，系统能否将资源分配给它？

答：（1）利用银行家算法对此时刻的资源分配情况进行分析，安全性分析情况如下表：

	work	need	allocation	Work+allocation	finish
P0	1 6 2 2	0 0 1 2	0 0 3 2	1 6 5 4	true
P3	1 6 5 4	0 6 5 2	0 3 3 2	1 9 8 6	true
P4	1 9 8 6	0 6 5 6	0 0 1 4	1 9 9 10	true
P1	1 9 9 10	1 7 5 0	1 0 0 0	2 9 9 10	true
P2	2 9 9 10	2 3 5 6	1 3 5 4	3 12 14 14	true

此时存在一个安全序列{P0,P3,P4,P1,P2}，故该状态是安全的。

（2）P2 提出 Request(1,2,2,2)，按银行家算法进行检查：

Request(1,2,2,2)≤Need(2,3,5,6)

Request(1,2,2,2) ≤Available(1,6,2,2)

试探性分配并修改相应的数据结构，资源分配情况如下：

	Allocation	Need	Available
P0	0 0 3 2	0 0 1 2	0 4 0 0
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	

再利用安全性算法检查系统状态是否安全，可利用资源向量（0，4，0，0）已不能满足任何进程的需要，故系统进入不安全状态，所以系统不能将资源分配给进程 P2。