习题一参考答案（P26）

1-2 批处理系统和分时系统各具有什么特点？为什么分时系统的响应比较快？

答：在批处理系统中操作人员将作业成批装入计算机并由计算机管理运行，在程序的运行期间用户不能干预，因此批处理系统的特点是：用户脱机使用计算机，作业成批处理，系统内多道程序并发执行以及交互能力差。在分时系统中不同用户通过各自的终端以交互方式共同使用一台计算机，计算机以“分时”的方法轮流为每个用户服务。分时系统的主要特点是：多个用户同时使用计算机的同时性，人机问答方式的交互性，每个用户独立使用计算机的独占性以及系统响应的及时性。分时系统一般采用时间片轮转的方法，使一台计算机同时为多个终端用户服务，因此分时系统的响应比较快。

1-4 什么是多道程序设计技术？试述多道程序运行的特征？答：多道程序设计技术是指同时把多个作业（程序）放入内存并允许它们交替执行和共享系统中的各类资源；当一道程序因某种原因（如 I/O 请求）而暂停执行时，CPU 立即转去执行另一道程序。多道程序运行具有如下特征：

- 多道：计算机内存中同时存放几道相互独立的程序。
- 宏观上并行：同时进入系统的几道程序都处于运行过程中，它们先后开始了各自的运行，但都未运行完毕。
- 微观上串行：从微观上看，内存中的多道程序轮流或分时地占有处理机，交替执行。

1-7 已讲过（略）

习题二参考答案（P43）

2-1 什么是核态？什么是用户态？

答：核态是指系统程序执行时，机器所处的状态。

用户态是指用户程序执行时，机器所处的状态。

2-2 为什么必须区分二态？系统如何区分出二态？

答：二态是指核态（系统程序执行时处理机所处的状态）和用户态（用户程序执行时处理机所处的状态）。

操作系统是计算机系统中最重要的系统软件，为了能正确地进行管理和控制，其本身是不能被破坏的。为此，系统应能建立一个保护环境，因此系统必须区分处理机的工作状态。系统中有两类程序在运行，它们的任务是不同的，系统程序是管理和控制者，用户程序是被管理和被控制的对象，因此应将它们运行时处理机的工作状态区分出来，即系统必须区分二态。

2-5 按中断的功能来分，中断有哪几种类型？

答：按中断的功能来分，中断有如下五种类型：

- **I/O** 中断
- 外中断
- 硬件故障中断
- 程序性中断
- 访管中断

2-8 什么是程序状态字？在微机中它一般由哪两个部分组成？

答：程序状态字是指反映程序执行时机器所处的现行状态的代码。

在微机中它一般由指令计数器（**PC**）和处理机状态寄存器（**PS**）。

2-9 什么是向量中断？什么是中断向量？

答：向量中断是指当中断发生时，由中断源自己引导处理机进入中断服务程序的中断过程。

中断向量就是存储该类型中断服务例行程序的入口地址和处理器状态字的存储单元。

2-12 什么是操作系统虚拟机？

答：操作系统是最基本的系统软件，它是硬件功能的第一层扩充。配置了操作系统的计算机称为操作系统虚拟机。扩充了的计算机除了可以使用原来裸机提供的各种基本硬件指令，还可以使用操作系统增加的许多其它指令。

3-1 用户与操作系统的接口是什么？一个分时系统提供什么接口？一个批处理系统又提供什么接口？

答：用户与操作系统的接口是指操作系统提供给用户与计算机打交道的外部机制。

一个分时系统提供的接口有系统功能调用和键盘操作命令。

一个批处理系统提供的接口有系统功能调用和作业控制语言。

3-2 计算机对用户算题任务的加工过程一般分哪几个作业步？各作业步之间的关系如何？用自己的上机体会说明。

答：计算机对用户算题任务的加工过程一般分四个作业步：编辑、编译、连接和运行。

各作业步之间的关系如下：

❖前一个作业步的结果是下一个作业步的操作对象；

❖一个作业步的成功完成依赖于上一个作业步的成功完成。

3-3 什么是系统调用？对操作系统的服务[illegible]程序调用有什么区别？

答：系统调用是操作系统提供给编程人员的唯一接口。编程人员利用系统调用，在源程序一级动态请求和释放系统资源，调用系统中已有的系统功能来完成那些与机器硬件部分相关的工作以及控制程序的运行速度等。因此，系统调用像一个黑箱子那样，对用户屏蔽了操作系统的具体功能，只提供有关的功能。系统调用与一般过程调用的主要区别如下：

系统调用程序是在核心态执行，调用它们需要一个类似于硬件中断处理的中断处理机制来提供系统服务。（也可按上课时讲的三个方面来阐述）

3-5 简述系统调用的执行过程。

答：系统调用命令的具体格式因系统而异，但由用户程序进入系统调用的步骤及执行过程大体相同：

首先，将系统调用命令所需的参数（如功能号）或参数区首址装入指定寄存器；然后，在用户程序中适当的位置安排一条调用系统功能指令。至于系统调用命令的功能号，有的系统直接在调用指令中给出，有的系统则把它作为系统调用命令的参数，在调用时放入指定寄存器。

当用户程序执行到调用系统功能的指令时，就转到系统调用的处理程序执行。其过程如下：

（1）为执行系统调用命令做准备，即将用户程序的“现场”保存起来，同时把系统调用命令的编号等参数放入约定的存储单元。（2）根据系统调用命令的编号查找系统调用入口表，找到相应系统功能调用子程序的入口地址，然后转到该子程序执行。当系统调用命令执行完毕，相应的结果通常返回给参数，这些参数放在约定的存储单元里。

（3）系统调用命令执行完毕后的处理，包括恢复用户程序执行的“现场”信息，同时把系统调用命令的返回参数或参数区首址放入指定的寄存器中，以供用户程序使用。

习题四参考答案（P97）

4-3 什么是进程？进程与程序的主要区别是什么？

答：进程，即是一个具有一定独立功能的程序关于某个数据集合的一次活动。

进程与程序的主要区别是：

（1）程序是指令的有序集合，是一个静态概念，其本身没有任何运行的含义，进程是程序在处理机上的一次执行过程，是一个动态概念。

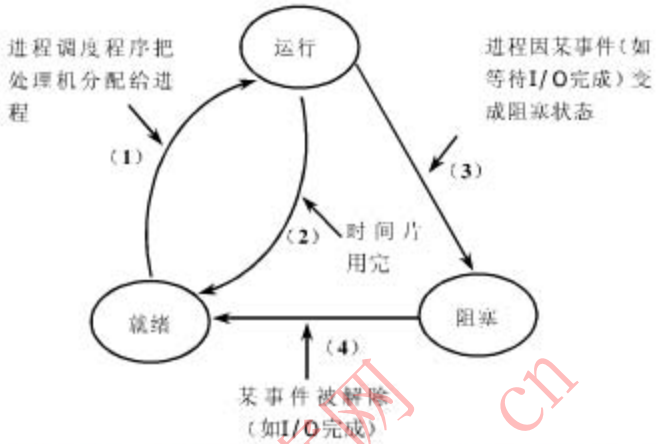
（2）程序作为软件资料可长期保存，而进程是有生命期的，因创建而产生、因调度而执行、因得不到资源而暂停、因撤消而消亡。

（3）程序是记录在介质上指令的有序集合，而进程则由程序、数据和进程控制块 3 部分组成。

（4）进程与程序之间无一一对应关系。不同的进程可以包含同一程序，同一程序在执行中也可以产生多个进程。（5）进程是一个独立的运行单位，也是系统进行资源分配和调度的独立单位。而程序无此概念。

4-6 进程有哪几个基本状态？试画出进程状态变迁图，并标明发生变迁的可能原因。

答：进程有三个基本状态：运行状态、就绪状态和等待状态（又称阻塞、挂起、睡眠）。



4-9 我们用进程流图来描述一个作业程序执行的先后次序。试用信号灯的 **P**、**V** 操作实现如图 **4.22(a)**、**(b)** 所示进程之间的同步，并写出程序描述。

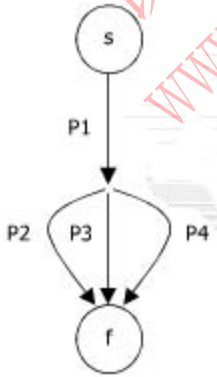


图4.22(a)

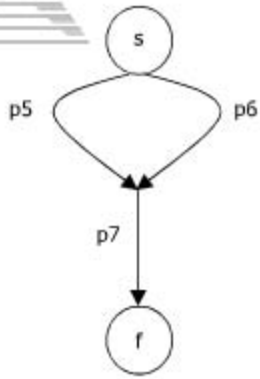


图4.22(b)

(a)解：Main(){

int s12=0,s13=0,s14=0;　cobegin

P1;

```
        P4;
        coend
}
P1(){
    p1 execute;
    V(s12);
    V(s13);
    V(s14);
}
P2(){
    P(s12);
    p2 execute;
}
P3(){
    P(s13);
    p3 execute;
}
P4(){
    P(s14);
    p4 execute;
}
```

(b)略

4-11 如图 4.24 所示，有一个计算进程和两个缓冲区 s、t（其大小为每次存放一个记录）。get 进程负责不断地把输入记录送入缓冲区 s 中，copy 进程负责从缓冲区 s 中取出记录复制到缓冲区 t 中，而 put 进程负责从缓冲区 t 中取出记录打印。试用 P、V 操作实现这三个进程之间的同步，并写出程序描述。

解：Main(){

int sa=1,sb=0;// sa 表示缓冲区 S 是否为空，sb 表示是否为满。

int ta=1,tb=0;// ta 表示缓冲区 T 是否为空，tb 表示是否为满。

cobegin

get;

copy;

put;

coend

}

Get(){

while(1){

P(sa);

input data to buffer S;

V(sb);

}

```
}
Copy (){
   while(1){
      P(sb);
      copy data from buffer S;
      V(sa);
      P(ta);
      input copy-data to buffer T;
      V(tb);}
}
Put(){
   while(1){
      P(tb);
      output data to buffer S;
      V(ta);
   }
}
```

4-12 什么是进程的互斥与同步？同步和互斥这两个概念有什么联系和区别？

答：在操作系统中，当一个进程进入临界区使用临界资源时，另一个进程必须等待，当占用临界资源的进程退出临界区后，另一进程才被允许去访问此临界资源。我们称进程之间的这种相互制约关系为互

斥。

进程同步是指多个相关进程在执行次序上的协调。这些进程相互合作，在一些关键点上可能需要互相等待或互通消息。

4-13 在一个实时系统中，有两个进程 **p** 和 **q**，它们是循环运行的。循环进程 **p** 每隔 **1** 秒钟由脉冲寄存器（**REG**）获得输入，并把它累计到一个整型变量（**W**）中，同时清除脉冲寄存器。循环进程 **q** 则每隔 **1** 小时输出这个整形变量的内容并把它复位。系统提供标准的 **I/O** 过程 **input** 和 **output**，并提供系统调用命令 **delay(seconds)**。试拟定出这两个进程并发活动的程序描述。

解：设置一个互斥信号量 **mutex** 用来实现进程 **p** 和 **q** 对共享变量 **W** 的互斥使用。

```
Main(){
   int mutex=1;
   int W=0;
   cobegin
      p;
      q;
   coend
}

p(){
```

```
    while(1){
        delay(1);
        P(mutex);
        W=W+input(REG);
        V(mutex);
        REG=0;
    }
}
q(){
    while(1){
        delay(3600);
        P(mutex);
        output(W);
        W=0;
        V(mutex);
    }
}
```

4-18 什么是线程？线程和进程有什么区别？

答：线程有时也称为轻量级进程，它是比进程更小的活动单位，它是进程中的一个执行路径。一个进程可以有多个执行路径即线程。

线程和进程的主要区别如下：

（1）线程是进程的一个组成部分。一个进程可以有多个线程，而且至少有一个可执行的线程。（2）进程是资源分配的基本单位，它拥有自己的地址空间和各种资源。线程是处理机调度的基本单位，它只能和其他线程共享进程的资源，而本身并不具有任何资源。（3）进程的多个线程都在进程的地址空间内活动。这样，在以线程为单位进行处理机调度和切换时，由于不发生资源变化特别是地址空间的变化，因此切换时间较短。而以进程为单位进行处理机调度和切换时，由于涉及到资源转移及现场保护等问题，将导致切换时间变长和资源利用率降低。（4）线程和进程一样，都有自己的状态和相应的同步机制。但是，由于线程没有自己单独的程序和数据空间，因而不能像进程的程序和数据那样交换到外存去。

（5）进程的调度和控制大多由操作系统的内核完成，而线程的控制既可以由操作系统内核完成，也可以由用户控制完成。

习题五参考答案（P117）

5-4 三个进程共享四个同类资源，这些资源的分配与释放只能一次一个。已知每一进程最多需要两个资源，试问：该系统会发生死锁吗？为什么？

答：该系统不会发生死锁。

因为最坏情况是每个进程都占有一个资源，申请第二个资源，而此时系统中剩下一个资源，不管这个资源分给哪个进程，都能满足它的资源要求，因此它能在有限时间内运行结束从而释放它所占有的两个资源，这两个资源又可以分配给另外两个进程，使它们能够运行结束，所以系统不会发生死锁。

5-5 p 个进程共享 m 个同类资源，每一个资源在任一时刻只能供一个进程使用，每一进程对任一资源都只能使用一有限时间，使用完便立即释放。并且每个进程对该类资源的最大需求量小于该类资源的数目。设所有进程对资源的最大需求数目之和小于 p+m，试证：该系统中不会发生死锁。

证明：假设每个进程最多请求X_i（$1 \leqslant i \leqslant p$）个资源，则根据题意有：

$$X_1+X_2+\cdots+X_{p-1}+X_p<p+m$$

$$\Rightarrow X_1+X_2+\cdots+X_{p-1}+X_p-p<m$$

$$\Rightarrow (X_1-1)+(X_2-1)+\cdots+(X_{p-1}-1)+(X_p-1)<m$$

$$\Rightarrow (X_1-1)+(X_2-1)+\cdots+(X_{p-1}-1)+(X_p-1)+1<m+1$$

$$\Rightarrow (X_1-1)+(X_2-1)+\cdots+(X_{p-1}-1)+(X_p-1)+1 \leqslant m$$

这说明在最坏情况下，每个进程均还差一个资源，而此时系统中还有一个没被分配的可用资源。将它分配给任何一个进程，都可以使该得到全部资源的进程运行结束而释放其占有的资源，并将释放的资源分配给其它的进程，使其它进程都能运行结束，系统不会发生死锁。

证毕

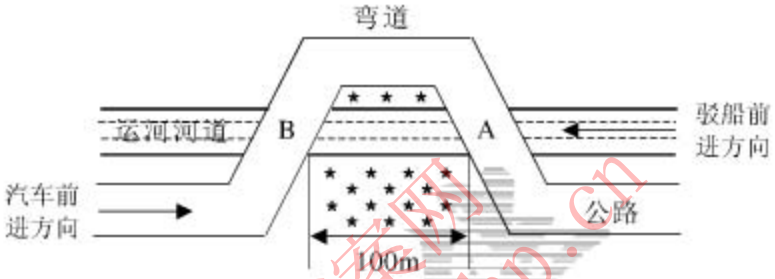
5-6 图 5.7 表示一带闸门的运河，其上有两架吊桥。吊桥坐落在一条公路上，为使该公路避开一块沼泽地而令其横跨运河两次。运河和公路的交通都是单方向的。运河上的基本运输由驳船担负。在一艘驳船接近吊桥 A 时就拉汽笛警告，

若桥上无车辆，[illegible]对吊桥 **B** 也按同样次序处理。

（1） 一艘典型驳船的长度为 **200** 米，当它在河上航行时是否会产生死锁？若会，其理由是什么？

（2） 如何能克服一个可能的死锁？请提出一个防止死锁的办法。

（3） 如何利用信号灯上的 **P**、**V** 操作实现车辆和驳船的同步？



答：（1）驳船长 **200** 米，当驳船通过了 **A** 桥，其船头到达 **B** 桥，请求 **B** 桥吊起，而此时它的尾部仍占据 **A** 桥。若这个时候 **B** 桥上及 **B** 桥到 **A** 桥之间的公路上都被汽车占据，而汽车又要求通过 **A** 桥。这样驳船和汽车都无法前进，形成死锁的局面。

（2）可以有以下两种方法：

①资源的静态分配。即进程把它所需要的所有资源在运行前提前申请，系统把它所需要的全部资源一次性都分配给它。也就是说，这时把 **A** 桥和 **B** 桥看成一个资源。打破了产生死锁的四个必要条件之一的部分分配条件。

②可以规定资源按序申请和分配，从而破坏了死锁的循环等待条件，防止死锁的发生。规定如下：**B** 桥的序号小于 **A** 桥的序号，驳船和汽车都必须先申请序号小的资源 **B** 桥，申请得到满足后，再申请序号大的资源 **A** 桥。

（3）算法如下：

①设置两个互斥信号量 **mutexa**，**mutexb**，用来实现驳船和汽车对 **A** 桥和对 **B** 桥的互斥使用；设置一个共享变量 **count**，用来记录当前占用 **A** 桥和 **B** 桥的汽车数并设置互斥信号量 **mutex**，用来实现汽车对共享变量 **count** 的互斥访问。

```
Main( ){
   int mutexa, mutexb, mutex, count
   mutexa=1;
   mutexb=1;
   mutex=1;
   count=0;
   cobegin
       bargei; //i=1,2,…,m
       carj; //j=1,2,…,n
   coend
}

bargei(){
   ……
   P(mutexb);
   P(mutexa);
   吊起 B 桥;
   吊起 A 桥;
   驶船通过 A 桥;
   放下 A 桥;
   驶船通过 B 桥;
   放下 B 桥;
   V(mutexa);
   V(mutexb);
   ……
}

carj(){
   ……
   P(mutex);
```

```
    count++;
    if(count==1)
    {
        P(mutexb);
        P(mutexa);
    }
    V(mutex);
    汽车通过 B 桥;
    汽车通过 AB 段公路;
    汽车通过 A 桥;
    P(mutex);
    count--;
    if(count==0)
    {
        V(mutexb);
        V(mutexa);
    }
    V(mutex);
    ……
}
```


②设置两个互斥信号量 **mutexa**，**mutexb**，用来实现驳船和汽车对 **A** 桥和对 **B** 桥的互斥使用；设置两个共享变量 **counta** 和 **countb**，分别用来记录 **A** 桥和 **B** 桥上的汽车数并设置互斥信号量 **mutex1** 和 **mutex2**，用来实现汽车对共享变量 **counta** 和 **countb** 的互斥访问。

```
Main( ){
    int mutexa, mutexb, mutex1, mutex2, counta, countb;
    mutexa=1;
    mutexb=1;
    mutex1=mutex2=1;
```

```
    counta=countb=0;
    cobegin
        bargei; //i=1,2,…,m
        carj; //j=1,2,…,n
    coend
}

bargei(){
    ……
    P(mutexb);
    吊起 B 桥;
    P(mutexa);
    吊起 A 桥;
    驶船通过 A 桥;
    放下 A 桥;
    V(mutexa);
    驶船通过 B 桥;
    放下 B 桥;
    V(mutexb);
    ……
}

carj(){
    ……
    P(mutex2);
    countb++;
    if(countb==1)
        P(mutexb);
    V(mutex2);
```

汽车通过 B 桥;

```
P(mutex2);
countb--;
if(countb==0)
    V(mutexb);
V(mutex2);
```

汽车通过 AB 段公路;

```
P(mutex1);
counta++;
if(counta==1)
    P(mutexa);
V(mutex1);
```

汽车通过 A 桥;

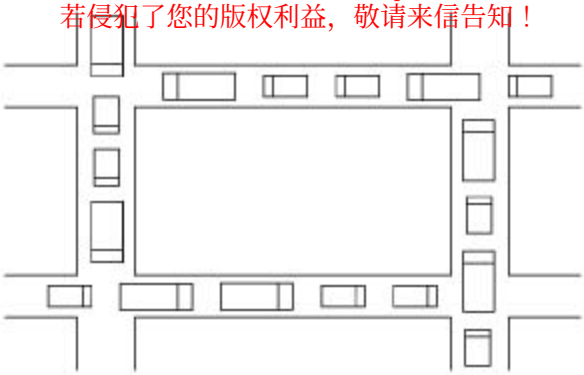
```
P(mutex1);
counta--;
if(counta==0)
    V(mutexa);
V(mutex1);
……
}
```

5-7 讨论图 **5.8** 描述的交通死锁的例子（设各方向上的汽车是单线、直线行驶）：

（**1**）对于产生死锁的四个必要条件中的哪些条件在此例中是适用的？

（**2**）提出一个简单的原则，它能避免死锁。

（**3**）若用计算机实现交通自动管理，请用信号灯上的 **P**、**V** 操作来实现各方向上汽车行驶的同步。



答：（1）路口是共享资源。

① 互斥条件：路口必须互斥使用，即汽车对它所需要的路口是排他性控制的。

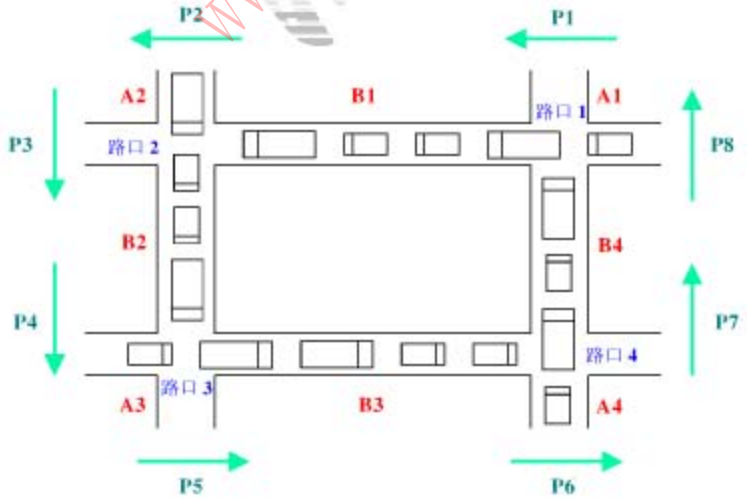
② 不剥夺条件：汽车一旦占用了路口，除非自己让出路口，别人无权剥夺。

③ 部分分配条件：每个方向的车队都占有一个路口，同时因申请新路口而等待。

④ 环路等待条件：占有路口的车都在等待其它车占有的路口，循环等待。

（2）可以在每个路口设置红绿灯进行控制：绿灯亮时，南北方向的车可以通行，东西方向的车禁止通行；当红灯亮时，东西方向的车可以通行，而南北方向的车禁止通行。

（3）分析如下



其中：**A1** 表示从东向西预通过路口 **1** 的车队；**B1** 表示位于路口 **1** 和路口 **2** 之间的车队，即从东向西预通过路口 **2** 的车队；**A2** 表示从北向南预通过路口 **2** 的车队；**B2** 表示位于路口 **2** 和路口 **3** 之间的车队，即从北向南预通过路口 **3** 的车队；**A3** 表示从西向东预通过路口 **3** 的车队；**B3** 表示位于路口 **3** 和路口 **4** 之间的车队，即从西向东预通过路口 **4** 的车队；**A4** 表示从南向北预通过路口 **4** 的车队；**B4** 表示位于路口 **4** 和路口 **1** 之间的车队，即从南向北预通过路口 **1** 的车队。**P1** 进程控制车队 **A1** 中的汽车从东向西依次通过路口 **1**；**P2** 进程控制车队 **B1** 中的汽车从东向西依次通过路口 **2**；**P3** 进程控制车队 **A2** 中的汽车从北向南依次通过路口 **2**；**P4** 进程控制车队 **B2** 中的汽车从北向南依次通过路口 **3**；**P5** 进程控制车队 **A3** 中的汽车从西向东依次通过路口 **3**；**P6** 进程控制车队 **B3** 中的汽车从西向东依次通过路口 **4**；**P7** 进程控制车队 **A4** 中的汽车从南向北依次通过路口 **4**；**P8** 进程控制车队 **B4** 中的汽车从南向北依次通过路口 **1**。其中，**P1**、**P3**、**P5**、**P7** 相当于 **4** 个生产者，**P2**、**P4**、**P6**、**P8** 相当于 **4** 个消费者，**4** 个路口之间的 **4** 段公路相当于 **4** 个有限缓冲区。

因此需要设置 **4** 对同步信号量：**emptyB1** 和 **fullB1**，**emptyB2** 和 **fullB2**，**emptyB3** 和 **fullB3**，**emptyB4** 和 **fullB4**。其中，**emptyBi**（**i=1,2,3,4**）表示车队 **Bi** 所在的公路段还能容纳多少辆汽车（假设车队 **Bi** 所在的公路段最多可以容纳 **NBi** 辆汽车，**i=1，2，3，4**）；**fullBi**（**i=1,2,3,4**）表示车队 **Bi** 所在的公路段上现在已有的汽车数。显然，**emptyBi**（**i=1,2,3,4**）的初始值等于 **NBi**（**i=1,2,3,4**），**fullBi**（**i=1,2,3,4**）的初始值等于 **0**。除此之外，还设置 **4** 个互斥信号灯 **mutexBi**（**i=1、2、3、4**），用来实现对 **Bi** 车队的互斥使用，初始值均为 **1**。

还需要设置 **4** 个互斥信号灯 **mutexi**（**i=1、2、3、4**），用来实现汽车对路口 **i** 的互斥使用，其初始值均为 **1**。

另外，还设置了 **4** 个计数器 **countA1**，**countA2**，**countA3**，**countA4** 用来分别记录当前 **A1** 车队、**A2** 车队、**A3** 车队和 **A4** 车队中的汽车数，初始值均为 **0**。

```
Main(){
  int mutex1,mutex2,mutex3,mutex4;
  int mutexB1,mutexB2,mutexB3,mutexB4;
  int emptyB1, emptyB2, emptyB3, emptyB4;
  int fullB1, fullB2, fullB3, fullB4;
```

```
    mutex1=1;mutex2=1;mutex3=1;mutex4=1;
    mutexB1=1;mutexB2=1;mutexB3=1;mutexB4=1;
    emptyB1=NB1;emptyB2=NB2;emptyB3=NB3;emptyB4=NB4;
    fullB1=0; fullB2=0; fullB3=0; fullB4=0;
    cobegin
        P1;   P2;   P3;   P4;   P5;   P6;   P7;   P8;
    coend
}

P1(){
    while(1){
        if(一辆汽车达到车队 A1)
            countA1++;
        if(countA1>0){
            P(emptyB1);
            P(mutexB1);
            P(mutex1);
            A1 车队中的一辆汽车从东向西通过路口 1;
            V(mutex1);
            V(mutexB1);
            V (fullB1);
            countA1--;
        }
    }
}

P2(){
    while(1){
        P(fullB1);
```

```
        P(mutexB1);
        P(mutex2);
        B1 车队中的一辆汽车从东向西通过路口 2;
        V(mutex2);
        V(mutexB1);
        V (emptyB1);
    }
}

P3(){
    while(1){
        if(一辆汽车达到车队 A2)
            countA2++;
        if(countA2>0){
            P(emptyB2);
            P(mutexB2);
            P(mutex2);
            A2 车队中的一辆汽车从北向南通过路口 2;
            V(mutex2);
            V(mutexB2);
            V (fullB2);
            countA2--;
        }
    }
}

P4(){
    while(1){
        P(fullB2);
        P(mutexB2);
```

```
        P(mutex3);
        B2 车队中的一辆汽车从北向南通过路口 3;
        V(mutex3);
        V(mutexB2);
        V (emptyB2);
    }
}

P5(){
    while(1){
        if(一辆汽车达到车队 A3)
            countA3++;
        if(countA3>0){
            P(emptyB3);
            P(mutexB3);
            P(mutex3);
            A3 车队中的一辆汽车从西向东通过路口 3;
            V(mutex3);
            V(mutexB3);
            V (fullB3);
            countA3--;
        }
    }
}

P6(){
    while(1){
        P(fullB3);
        P(mutexB3);
        P(mutex4);
```

```
        B3 车队中的一辆汽车从西向东通过路口 4;
        V(mutex4);
        V(mutexB3);
        V (emptyB3);
    }
}

P7(){
    while(1){
        if(一辆汽车达到车队 A4)
            countA4++;
        if(countA4>0){
            P(emptyB4);
            P(mutexB4);
            P(mutex4);
            A4 车队中的一辆汽车从南向北通过路口 4;
            V(mutex4);
            V(mutexB4);
            V (fullB4);
            countA4--;
        }
    }
}

P8(){
    while(1){
        P(fullB4);
        P(mutexB4);
        P(mutex1);
```

```
        B4 车队中的一辆汽车从南向北通过路口 1；
        V(mutex1);
        V(mutexB4);
        V (emptyB4);
    }
}
```

补充作业：

画出资源分配图，判断此状态是否为安全状态？如果是，则找出安全序列；在此基础上

1. P_2 申请（1，0，2）能否分配？为什么？

2. P_5 申请（3，3，0）能否分配？为什么？

3. P_1 申请（0，2，0）能否分配？为什么？

	已分配的资源			最大需求量		
	A	B	C	A	B	C
P_1	0	1	0	7	5	3
P_2	2	0	0	3	2	2
P_3	3	0	2	9	0	2
P_4	2	1	1	2	2	2
P_5	0	0	2	4	3	3

剩余资源	A	B	C
	3	3	2

习题五参考答案（P117）

5-4 三个进程共享四个同类资源，这些资源的分配与释放只能一次一个。已知每一进程最多需要两个资源，试问：该系统会发生死锁吗？为什么？

答：该系统不会发生死锁。

因为最坏情况是每个进程都占有一个资源，申请第二个资源，而此时系统中剩下一个资源，不管这个资源分给哪个进程，都能满足它的资源要求，因此它能在有限时间内运行结束从而释放它所占有的两个资源，这两个资源又可以分配给另外两个进程，使它们能够运行结束，所以系统不会发生死锁。

5-5 p 个进程共享 m 个同类资源，每一个资源在任一时刻只能供一个进程使用，每一进程对任一资源都只能使用一有限时间，使用完便立即释放。并且每个进程对该类资源的最大需求量小于该类资源的数目。设所有进程对资源的最大需求数目之和小于 p+m。试证：在该系统中不会发生死锁。

证明：假设每个进程最多请求X_i（$1 \leqslant i \leqslant p$）个资源，则根据题意有：

$$X_1+X_2+\cdots+X_{p-1}+X_p<p+m$$

$$\Rightarrow X_1+X_2+\cdots+X_{p-1}+X_p-p<m$$

$$\Rightarrow (X_1-1)+(X_2-1)+\cdots+(X_{p-1}-1)+(X_p-1)<m$$

$$\Rightarrow (X_1-1)+(X_2-1)+\cdots+(X_{p-1}-1)+(X_p-1)+1<m+1$$

$$\Rightarrow (X_1-1)+(X_2-1)+\cdots+(X_{p-1}-1)+(X_p-1)+1 \leqslant m$$

这说明在最坏情况下，[illegible]均[illegible]资源，而此时系统中还有一个没被分配的可用资源。将它分配给任何一个进程，都可以使该得到全部资源的进程运行结束而释放其占有的资源，并将释放的资源分配给其它的进程，使其它进程都能运行结束，系统不会发生死锁。

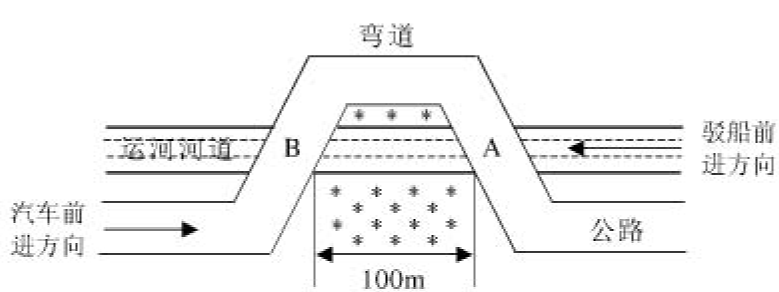
证毕

5-6 图 5.7 表示一带闸门的运河，其上有两架吊桥。吊桥坐落在一条公路上，为使该公路避开一块沼泽地而令其横跨运河两次。运河和公路的交通都是单方向的。运河上的基本运输由驳船担负。在一艘驳船接近吊桥 A 时就拉汽笛警告，若桥上无车辆，吊桥就吊起，直到驳船尾部通过此桥为止。对吊桥 B 也按同样次序处理。

（1）一艘典型驳船的长度为 200 米，当它在河上航行时是否会产生死锁？若会，其理由是什么？

（2）如何能克服一个可能的死锁？请提出一个防止死锁的办法。

（3）如何利用信号灯上的 P、V 操作实现车辆和驳船的同步？



答：（1）驳船长200米，当驳船通过了A桥，其船头到达B桥，请求B桥吊起，而此时它的尾部仍占据A桥。若这个时候B桥上及B桥到A桥之间的公路上都被汽车占据，而汽车又要求通过A桥。这样驳船和汽车都无法前进，形成死锁的局面。

（2）可以规定资源按序申请和分配，从而破坏了死锁的循环等待条件，防止死锁的发生。规定如下：B桥的序号小于A桥的序号，驳船和汽车都必须先申请序号小的资源B桥，申请得到满足后，再申请序号大的资源A桥。

（4）算法如下：

设置两个互斥信号量mutexa，mutexb，用来实现驳船和汽车对A桥和对B桥的互斥使用；设置两个共享变量counta和countb，分别用来记录A桥和B桥上的汽车数并设置互斥信号量mutex1和mutex2，用来实现汽车对共享变量counta和countb的互斥访问。

```
Main( ){
   int mutexa, mutexb, mutex1, mutex2, counta, countb;
   mutexa=1;
   mutexb=1;
   mutex1=mutex2=1;
   counta=countb=0;
   cobegin
      bargei; //i=1,2,…,m
      carj; //j=1,2,…,n
```

```
    coend
}

bargei(){
    ……
    P(mutexb);
    吊起 B 桥;
    P(mutexa);
    吊起 A 桥;
    驳船通过 A 桥;
    放下 A 桥;
    V(mutexa);
    驳船通过 B 桥;
    放下 B 桥;
    V(mutexb);
    ……
}

carj(){
    ……
    P(mutex2);
    countb++;
```

```
if(countb==1)
   P(mutexb);
V(mutex2);
```

汽车通过 B 桥；

```
P(mutex2);
countb--;
if(countb==0)
   V(mutexb);
V(mutex2);
```

汽车通过 AB 段公路；

```
P(mutex1);
counta++;
if(counta==1)
   P(mutexa);
V(mutex1);
```

汽车通过 A 桥；

```
P(mutex1);
counta--;
if(counta==0)
   V(mutexa);
V(mutex1);
……
```

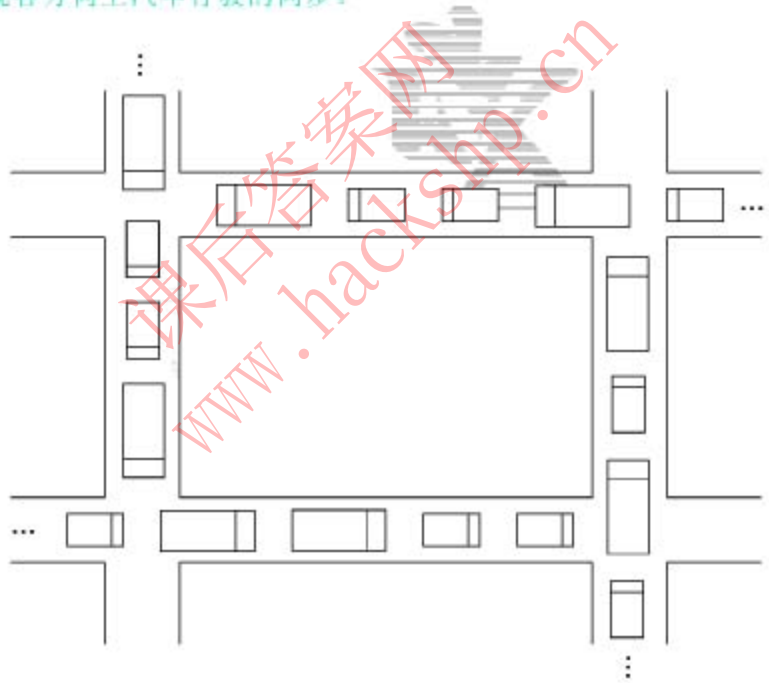
}

5-7 讨论图 **5.8** 描述的交通死锁的例子（设各方向上的汽车是单线、直线行驶）：

（**1**）对于产生死锁的四个必要条件中的哪些条件在此例中是适用的？

（**2**）提出一个简单的原则，它能避免死锁。

（**3**）若用计算机实现交通自动管理，请用信号灯上的 **P**、**V** 操作来实现各方向上汽车行驶的同步。



答：（**1**）路口是共享资源。

① 互斥条件：路口必须互斥使用，即汽车对它所需要的路口是排他性控制的。

② 不剥夺条件：车辆不得强行剥夺已占有的路口，别人无权剥夺。

③ 部分分配条件：每个方向的车队都占有一个路口，同时因申请新路口而等待。

④ 环路等待条件：占有路口的车都在等待其它车占有的路口，循环等待。

（2）可以在每个路口设置红绿灯进行控制：绿灯亮时，南北方向的车可以通行，东西方向的车禁止通行；当红灯亮时，东西方向的车可以通行，而南北方向的车禁止通行。

（3）设置 4 个互斥信号灯 mutexi（i=1、2、3、4），用来实现汽车对每个路口的互斥使用；8 个进程，4 个生产者，4 个消费者，4 对同步信号量。

```
Main(){
  int mutex1,mutex2,mutex3,mutex4;
  int SA1,SB1,SA2,SB2,SA3,SB3,SA4,SB4;
  mutex1=mutex2=mutex3=mutex4=1;
  SA1=SA2=SA3=SA4=1;
  SB1=SB2=SB3=SB4=0;
  cobegin
    P1;
    P2;
    P3;
```

```
        P4;
        P5;
        P6;
        P7;
        P8;
    coend
}

P1(){
    while(1){
        P(SA1);
        P(mutex1);
        通过一辆汽车；
        V(mutex1);
        P(SB1);
    }
}

P2(){
    while(1){
        P(SB1);
        P(mutex2);
```

```
        通过一辆汽车；
        V(mutex2);
        P(SA1);
    }
}
P3(){
    while(1){
        P(SA2);
        P(mutex2);
        通过一辆汽车；
        V(mutex2);
        P(SB2);
    }
}

P4(){
    while(1){
        P(SB2);
        P(mutex3);
        通过一辆汽车；
        V(mutex3);
        P(SA2);
```

```
    }
}

P5(){
    while(1){
        P(SA3);
        P(mutex3);
        通过一辆汽车；
        V(mutex3);
        P(SB3);
    }
}

P6(){
    while(1){
        P(SB3);
        P(mutex4);
        通过一辆汽车；
        V(mutex4);
        P(SA3);
    }
}
```

```
P7(){
   while(1){
      P(SA4);
      P(mutex4);
      通过一辆汽车；
      V(mutex4);
      P(SB4);
   }
}

P8(){
   while(1){
      P(SB4);
      P(mutex1);
      通过一辆汽车；
      V(mutex1);
      P(SA4);
   }
}
```

补充作业：

画出资源分配图，[illegible]是，则找出安全序列：在此基础上

1. P_2 申请（1，0，2）能否分配？为什么？

2. P_5 申请（3，3，0）能否分配？为什么？

3. P_1 申请（0，2，0）能否分配？为什么？

	已分配的资源			最大需求量		
	A	B	C	A	B	C
P_1	0	1	0	7	5	3
P_2	2	0	0	3	2	2
P_3	3	0	1	9	0	2
P_4	2	1	1	2	2	2
P_5	0	0	2	4	3	3

剩余资源	A	B	C
	3	3	2

答：上课时已给出提示。略

6-2 某系统进程调度状态变迁图如图 **6.5** 所示（设调度方式为非剥夺方式），请说明：

（**1**）什么原因将引起发生变迁 **2**、变迁 **3**、变迁 **4**？

（**2**）当观察系统中所有进程时，能够看到某一进程产生的一次状态变迁能引起另一进程作一次状态变迁，在什么情况下，一个进程的变迁 **3** 能立即引起另一个进程发生变迁 **1**？

（**3**）下述因果变迁是否可能发生？如果可能的话，在什么情况下发生？

（a）**3->1**；（b）**3->2**；（c）**2->1**

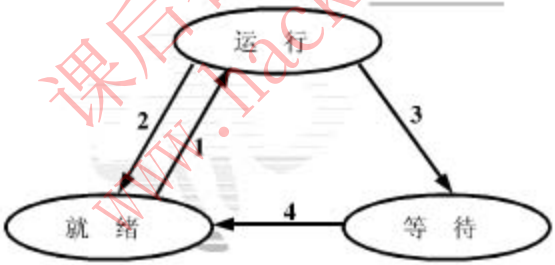


图 **6.5**

解答：（**1**）当运行进程在分得的时间片内未完成，时间片到将发生变迁 **2**；

当运行进程在执行过程中，需要等待某事件的发生才能继续向下执行，此时会发生变迁 **3**；

当等待进程等待的事件发生了，将会发生变迁 **4**。

（2）正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时会立即引起一个就绪进程被调度执行的变迁 1。

（3）a. 3->1 的因果变迁可能发生

正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时必然引起一个就绪进程被调度执行的变迁 1。

b. 3->2 的因果变迁不可能发生。

c. 2->1 的因果变迁必然发生

正运行的进程因时间片到变为就绪状态的变迁 2，必然引起一个就绪进程被调度执行的变迁 1。

6-3 若题 2 中所采用的调度为可剥夺式，请回答题 2 中提出的问题：

（1）什么原因将引起发生变迁 2、变迁 3、变迁 4？

（2）当观察系统中所有进程时，能够看到某一进程产生的一次状态变迁能引起另一进程的一次状态变迁，在什么情况下，一个进程的变迁 3 能立即引起另一个进程发生变迁 1？

（3）下述因果变迁是否可能发生？如果可能的话，在什么情况下发生？

（a）3->1；（b）3->2；（c）2->1

解答：（1）当运行进程在分得的时间片内未完成，时间片到将发生变迁 2；或者新创建一个进程或一个等待进程变成就绪，它具有比当前进程更高的优先级，也将发生变迁 2。

当运行进程在执行过程中，需要等待某事件的发生才能继续向下执行，此时会发生变迁 3。

当等待进程等待的事件发生了，将会发生变迁 4。

（2）正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时会立即引起一个就绪进程被调度执行的变迁 1。

（3）a. 3->1 的因果变迁可能发生

正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时必然引起一个就绪进程被调度执行的变迁 1。

b. 3->2 的因果变迁不可能发生。

c. 2->1 的因果变迁必然发生

正运行的进程因时间片到变为就绪状态的变迁 2，必然引起一个就绪进程被调度执行的变迁 1。或者新创建一个进程或一个等待进程变成就绪，它具有比当前进程更高的优先级发生的变迁 2，必然引起调度一个具有更高优先级就绪进程执行的变迁 1。

6-4 某系统的进程状态变迁图如图 6.6 所示（设该系统的进程调度方式为非剥夺式），请说明：

（1）一个进程发生变迁 3 的原因是什么？发生变迁 2、变迁 4 的原因又是什么？

（2）下述因果变迁是否会发生，如果有可能的话，在什么情况下发生？

（a）2->1；（b）3->2；（c）4->5；（d）4->2；（e）3->5

（3）根据此状态变迁图叙述该系统的调度策略、调度效果。

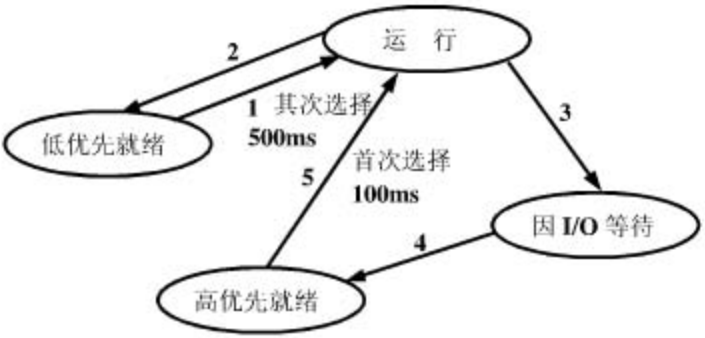


图 6.6

解答：（1）当运行进程在执行过程中，需要等待某事件的发生才能继续向下执行，此时会发生变迁 3。

当运行进程在分得的时间片（100ms 或 500ms）内未完成，时间片 100ms 或时间片 500ms 到将发生变迁 2。

当等待进程等待的事件发生了，将会发生变迁 4。

（2）a. 2->1 的因果变迁可能发生

当运行进程在分得的时间片（100ms 或 500ms）内未完成，时间片 100ms 或时间片 500ms 到发生的变迁 2，在高优先就绪队列为空时，必然引起低优先就绪队列中的一个就绪进程被调度执行的变迁 1。

b. 3->2 的因果变迁不可能发生

c. 4->5 的因果变迁可能发生

在高优先就绪队列为空时调度算法中，当一进程从等待状态变为就绪状态的变迁 **4**，在该进程的优先级最高且系统采用抢占式调度时，就会引起该进程被调度执行的变迁 **5**。

或者在当前运行进程是原低优先就绪队列中的一个进程且高优先就绪队列为空时，若系统采用可抢占方式，则当一进程从等待状态变为就绪状态的变迁 **4**，就会引起该进程被调度执行的变迁 **5**。

d. **3->2** 的因果变迁不可能发生

e. **3->5** 的因果变迁可能发生

正在运行的进程因等待某事件的发生而变为等待状态的变迁 **3**，在高优先就绪队列非空时必然引起一个就绪进程被调度执行的变迁 **5**。

（**3**）调度策略：

首先调度高就绪队列中的进程（一般由 **I/O** 型进程或短进程组成）投入运行（给高优先就绪队列中的进程分配的时间片大小为 **100ms**），只有当高就绪队列中的所有进程全部运行完毕或因等待某事件发生处于阻塞状态，高就绪队列中没有进程可运行时，才调度低优先就绪队列中的进程（一般由计算型进程或长进程组成）（给低优先就绪队列中的进程分配的时间片大小为 **500ms**）。若一个运行进程时间片（**100ms** 或 **500ms**）到还未完成就进入低优先就绪队列。若某进程在运行期间因等待某事件发生而进入阻塞队列，则当其所等待事件完成后，它将进入高优先就绪队列。

调度效果：这种算法优先照顾了 **I/O** 量大的进程或短进程。

6-7 在单道批处理系统中，分别采用先来先服务调度算法和最短作业调度算法进行调度，哪一种算法调度性能好些？请完成表6.5中未填写的各项。

表 6.5

作业	提交时间	执行时间	开始时间	完成时间	周转时间（分钟）	带权周转时间（分钟）
1	10:00	120				
2	10:10	60				
3	10:15	15				
平均周转时间 t= 平均带权周转时间 w=						

解答：（1）先来先服务调度算法

作业	提交时间	执行时间	开始时间	完成时间	周转时间（分钟）	带权周转时间（分钟）
1	10:00	120	10:00	12:00	120	1
2	10:10	60	12:00	13:00	170	2.83
3	10:15	15	13:00	13:15	180	12
平均周转时间 t=（120+170+180）/3=156.7 平均带权周转时间 w=（1+2.83+12）/3=5.3						

（2）最短作业优先调度算法

作业	提交时间	执行时间	开始时间	完成时间	周转时间（分钟）	带权周转时间（分钟）
1	10:00	120	10:00	12:00	120	1
2	10:10	60	12:15	13:15	185	3.08
3	10:15	15	12:00	12:15	120	8
平均周转时间 t=（120+185+120）/3=141.7 平均带权周转时间 w=（1+2.83+12）/3=4						

6-2 某系统进程调度状态变迁图如图 **6.5** 所示（设调度方式为非剥夺方式），请说明：

（**1**）什么原因将引起发生变迁 **2**、变迁 **3**、变迁 **4**？

（**2**）当观察系统中所有进程时，能够看到某一进程产生的一次状态变迁能引起另一进程作一次状态变迁，在什么情况下，一个进程的变迁 **3** 能立即引起另一个进程发生变迁 **1**？

（**3**）下述因果变迁是否可能发生？如果可能的话，在什么情况下发生？

（**a**）**3->1**；（**b**）**3->2**；（**c**）**2->1**

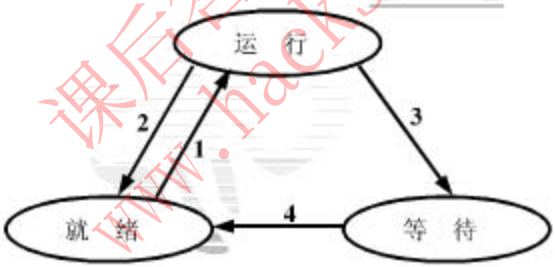


图 **6.5**

解答：（**1**）当运行进程在分得的时间片内未完成，时间片到将发生变迁 **2**；

当运行进程在执行过程中，需要等待某事件的发生才能继续向下执行，此时会发生变迁 **3**；

当等待进程等待的事件发生了，将会发生变迁 **4**。

（2）正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时会立即引起一个就绪进程被调度执行的变迁 1。

（3）a3->1 的因果变迁可能发生

正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时必然引起一个就绪进程被调度执行的变迁 1。正运行的进程因时间片到变为就绪状态的变迁 2，必然引起一个就绪进程被调度执行的变迁 1。

6-3 若题 2 中所采用的调度为可剥夺式，请回答题 2 中提出的问题：

（1）什么原因将引起发生变迁 2、变迁 3、变迁 4？

（2）当观察系统中所有进程时，能够看到某一进程产生的一次状态变迁能引起另一进程的一次状态变迁，在什么情况下，一个进程的变迁 3 能立即引起另一个进程发生变迁 1？

（3）下述因果变迁是否可能发生？如果可能的话，在什么情况下发生？

（a）3->1；（b）3->2；（c）2->1

解答：（1）当运行进程在分得的时间片内未完成，时间片到将发生变迁 2；或者新创建一个进程或一个等待进程变成就绪，它具有比当前进程更高的优先级，也将发生变迁 2。

当运行进程在执行过程中，等待某事件的发生才能继续向下执行，此时会发生变迁 3。

当等待进程等待的事件发生了，将会发生变迁 4。

（2）正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时会立即引起一个就绪进程被调度执行的变迁 1。

（3）a3->1 的因果变迁可能发生

正在运行的进程因等待某事件的发生而变为等待状态的变迁 3，在就绪队列非空时必然引起一个就绪进程被调度执行的变迁 1。正运行的进程因时间片到变为就绪状态的变迁 2，必然引起一个就绪进程被调度执行的变迁 1。或者新创建一个进程或一个等待进程变成就绪，它具有比当前进程更高的优先级发生的变迁 2，必然引起调度一个具有更高优先级就绪进程执行的变迁 1。

6-4 某系统的进程状态变迁图如图 **6.6** 所示（设该系统的进程调度方式为非剥夺式），请说明：

（1）一个进程发生变迁 3 的原因是什么？发生变迁 2、变迁 4 的原因又是什么？

（2）下述因果变迁是否会发生，如果有可能的话，在什么情况下发生？

（a）2->1；（b）3->2；（c）4->5；（d）4->2；（e）3->5

（3）根据此状态变迁图叙述该系统的调度策略、调度效果。

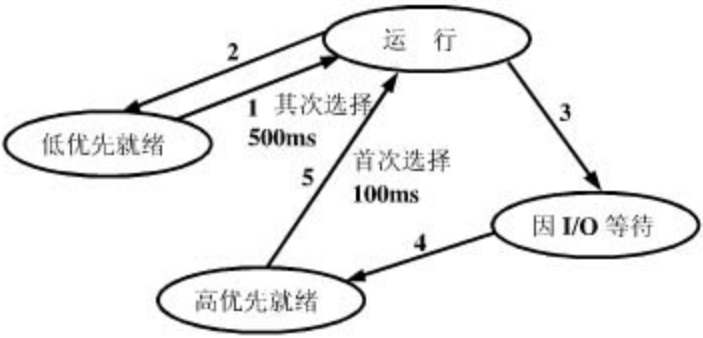


图 6.6

解答：（1）当运行进程在执行过程中，需要等待某事件的发生才能继续向下执行，此时会发生变迁 3。

当运行进程在分得的时间片（100ms 或 500ms）内未完成，时间片 100ms 或时间片 500ms 到将发生变迁 2。

当等待进程等待的事件发生了，将会发生变迁 4。

（2）a2->1 的因果变迁可能发生

当运行进程在分得的时间片（100ms 或 500ms）内未完成，时间片 100ms 或时间片 500ms 到发生的变迁 2，在高优先就绪队列为空时，必然引起低优先就绪队列中的一个就绪进程被调度执行的变迁 1。在高优先就绪队列采用优先级调度算法时，当一进程从等待状态变为就绪状态的变迁 4，在该进程的优先级最高且系统采用抢占式调度时，就会引起该进程被调度执行的变迁 5。

当一进程从等待状态变为就绪状态时，则引起该进程被调度执行的变迁 **5**。

正在运行的进程因等待某事件的发生而变为等待状态的变迁 **3**，在高优先就绪队列非空时必然引起一个就绪进程被调度执行的变迁 **5**。

（**3**）调度策略：

首先调度高就绪队列中的进程（一般由 **I/O** 型进程或短进程组成）投入运行（给高优先就绪队列中的进程分配的时间片大小为 **100ms**），只有当高就绪队列中的所有进程全部运行完毕或因等待某事件发生处于阻塞状态，高就绪队列中没有进程可运行时，才调度低优先就绪队列中的进程（一般由计算型进程或长进程组成）（给低优先就绪队列中的进程分配的时间片大小为 **500ms**）。若一个运行进程时间片（**100ms** 或 **500ms**）到还未完成就进入低优先就绪队列。若某进程在运行期间因等待某事件发生而进入阻塞队列，则当其所等待事件完成后，它将进入高优先就绪队列。

调度效果：

这种算法优先照顾了 **I/O** 量大的进程或短进程。

6-7 在单道批处理系统中，有下列三个作业用先来先服务调度算法和最短作业调度算法进行调度，哪一种算法调度性能好些？请完成表 **6.5** 中未填写的各项。

表 6.5

作业	提交时间	执行时间	开始时间	完成时间	周转时间（分钟）	带权周转时间（分钟）
1	10:00	120				
2	10:10	60				
3	10:15	15				
平均周转时间 t= 平均带权周转时间 w=						

解答：（1）先来先服务调度算法

作业	提交时间	执行时间	开始时间	完成时间	周转时间（分钟）	带权周转时间（分钟）
1	10:00	120	10:00	12:00	120	1
2	10:10	60	12:00	13:00	170	2.83
3	10:15	15	13:00	13:15	180	12
平均周转时间 t=（120+170+180）/3=156.7 平均带权周转时间 w=（1+2.83+12）/3=5.3						

（2）最短作业优先调度算法

作业	提交时间	执行时间	开始时间	完成时间	周转时间（分钟）	带权周转时间（分钟）
1	10:00	120	10:00	12:00	120	1
2	10:10	60	12:15	13:15	185	3.08
3	10:15	15	12:00	12:15	120	8
平均周转时间 t=（120+185+120）/3=141.7 平均带权周转时间 w=（1+2.83+12）/3=4						

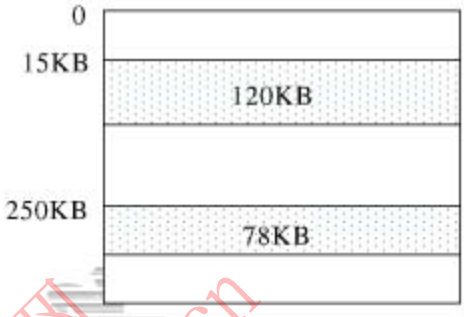
习题七参考答案（P167）

7-7 如图 **7.39** 所示，主存中有两个空白区。现有这样一个作业序列：

作业 **1**　要求　**50KB**

作业 **2**　要求　**60KB**

作业 **3**　要求　**70KB**



若用首次适应算法和最佳适应算法来处理这个作业序列，试问哪一种算法可以分配得下，为什么？

答：用首次适应法首先把 **120KB** 的空白区分配 **50KB** 的空间给作业 **1**，分割后还剩 **70KB** 的空白区，再将其分配给作业 **2**，剩下 **10KB** 的空白区。起始地址为 **250KB** 的空白区（**78KB**）可以满足作业 **3** 的需求，分割后还剩 **8KB** 的空白区。因此首次适应法可以吞吐此作业序列。

用最佳适应法，则先分配 **78KB** 的空白区给作业 **1**，还剩 **28KB** 的空白区，不能满足作业 **2** 的需求，因此分配 **120KB** 的空白区给作业 **2**，还剩 **60KB** 的空白区。此时系统中有大小为 **28KB** 和 **60KB** 的两个空白区，它们均不能满足作业 **3** 的需求。因此最佳适应法不能吞吐此作业序列。

7-8 已知主存有 **256KB** 容量，其中 **OS** 占用低址 **20KB**，可以有这样一个作业序列：

作业 1	要求	80KB
作业 2	要求	16KB
作业 3	要求	140KB
作业 1	完成	
作业 3	完成	
作业 4	要求	80KB
作业 5	要求	120KB

试用首次适应算法和最佳适应算法分别处理上述作业序列（在存储分配时，从空白区高址处分割作为已分配区），并完成以下各步：

（1）画出作业 1、2、3 进入主存后，主存的分配情况。

（2）作业 1、3 完成后，画出主存分配情况。

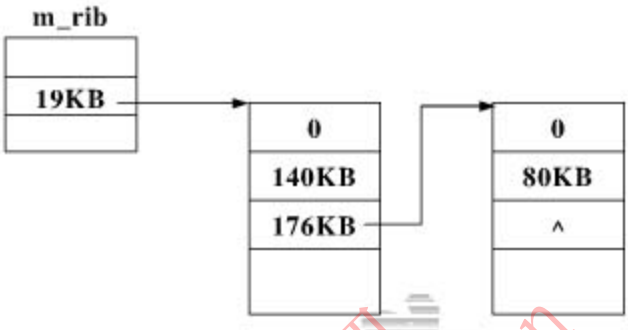
（3）画出两种算法中空白区的分区描述器信息（假定分区描述器所需占用的字节数已包含在作业所要求的主存容量中）及空白区链接情况。

（4）哪种算法对该作业序列而言是合适的？

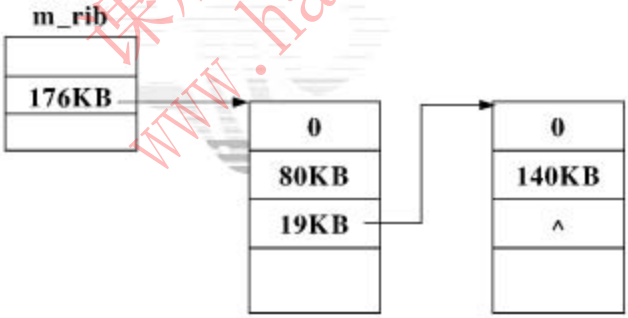
答：（1）作业 1、2、3 进入主存后，主存的分配情况如下图所示：

（2）作业 1、3 完成后，主存的分配情况如下图所示：

（3）首次适应法中空白区的分区描述器信息及空白区链接情况如下所示：



最佳适应法中空白区的分区描述器信息及空白区链接情况如下所示：



（4）若采用首次适应法，则应将起始地址为 **19KB** 的空白区（大小为 **140KB**）分配给作业 **4**，还剩下 **96KB** 空白区。此时系统中有两个空白区，它们的大小分别为 **96KB** 和 **80KB**，都不能满足作业 **5** 的需求。所以这种方法对该作业序列是不合适的。

若采用最佳适应法，则此无特起始地址为 **176KB** 的空白区（大小为 **80KB**）分配给作业 **4**。此时系统中还有一个空白区，即起始地址为 **19KB**，大小为 **140KB** 的空白区，它可以满足作业 **5** 的需求（**120KB**）。因此最佳适应法对该作业序列是合适的。

7-10 已知主存容量为 **64KB**，某一作业 **A** 的地址空间如图 **7.40** 所示，它的 **4** 个页面（页面大小为 **1KB**）**0**、**1**、**2**、**3** 被分配到主存的 **2**、**4**、**6**、**7** 块中，要求并回答

（**1**）画出作业 **A** 的页面映射表。

（**2**）当 **200** 号单元处有一条指令“**mov r1,[3500]**”执行时，如何进行正确的地址变换，以使 **3500** 处的内容 **12345** 装入 **r1** 中？。

答：（**1**）作业 **A** 的页面映射表如下图所示：

作业 **A** 的页表

页号	物理块号
0	**2**
1	**4**

（2）因为每页大小为 **1KB=1024** 字节，而 **3500=3*1024+428**，可知逻辑地址 **3500** 对应的页号为 **3**，页内地址为 **428**。根据页号检索页表可知对应的物理块号为 **7**，所以物理地址为：**7*1024+428=7596**。

补充作业

1.对于一个利用快表的分页系统中，假设 CPU 一次访问内存的时间为 1us，访问快表的时间可忽略不计，如果 85% 的地址映射可直接通过快表完成，那么进程完成一次内存读写的平均有效时间是多少？

答：进程完成一次内存读写的平均有效时间为：

t=0.85*1+(1-0.85)*(1+1)=1.15us

2.在一个分页存储管理系统中，逻辑地址长度为 16 位，页面大小为 4096 字节，现有一个逻辑地址为 2F6AH，且将 0、1、2 页依次存放在物理块 5、10、11 中，问相应的物理地址为多少？

答：该作业的页面映射表如下图所示：

页　表

页号	物理块号
0	**5**
1	**10**
2	**11**

页面大小为 4096=2^{12}字节，可知 16 位的逻辑地址中低 12 位为页内地址，而高 4 位为页号。

2F6AH=0010 1111 0110 1010B

页号为 2，检索页表可得其对应的物理块号为 11，则对应的物理地址为：1011 1111 0110 1010B=0BF6AH

3.设有一个页式存储管理系统，向用户提供的逻辑地址空间最大为 16 页，每页为 2048 字节，内存总共有 8 个物理块，问逻辑地址至少应该为多少位？内存空间为多大？

答：因为逻辑地址空间最大为 16 页，共需要 4 位二进制表示页号（16=2^4）；每页为 2048 字节，需要 11 位表示页内地址（2048=2^{11}）。因此逻辑地址至少为 15 位。

因为物理块大小与页面相同，故块内地址需 11 位表示；又因为内存中有 8 个物理块，需要 3 位表示块号。因此物理地址需要 14 位表示，即内存空间大小为 2^{14}=16KB。

8-1 什么是“设备独立性”？引入这一概念有什么好处？

解答：所谓设备独立性是指，用户在编制程序时所使用的设备与实际使用的设备无关，也就是在用户程序中仅使用逻辑设备名。

引入设备独立性，可使应用程序独立于物理设备。独立性可使用户程序独立于某一特定的物理设备。此时，进程只需用逻辑设备名去请求使用某类设备。当系统中有多台该类设备时，系统可将其中的任一台设备分配给请求进程，而不必局限于某一指定设备。这样，可以显著地提高资源的利用率和可适应性。

独立性还可以使用户程序独立于设备类型。例如，在进行输出时，既可以利用显示终端进行输出，也可以利用打印机进行输出。有了这种适应性，就可以很方便地实现输出重定向。类似地可以实现输入重定向。

8-4 什么是缓冲？引入缓冲的原因是什么？

解答：缓冲是两种不同速度的设备之间传输信息时平滑传输过程的常用手段。

引入了缓冲技术的原因有：

（1）为了进一步缓和 **CPU** 和 **I/O** 设备之间速度不匹配的矛盾。

（2）为了减少中断次数和 **CPU** 的中断处理时间。如果没有缓冲，慢速 **I/O** 设备每传送一个字节就要产生一个中断，**CPU** 必须处理该

中断。如果采用了缓冲，则慢速 I/O 设备将缓冲区填满时，才向 CPU 发出中断，从而减少了中断次数和 CPU 的中断处理时间。

（3）为了解决 DMA 或通道方式下数据传输的瓶颈问题。DMA 或通道方式都适用于成批数据传输，在无缓冲的情况下，慢速 I/O 设备只能一个字节一个字节的传送信息，这造成了 DMA 或通道方式数据传输的瓶颈。缓冲区的设置适应了 DMA 或通道方式的成批数据传输方式，解决了数据传输的瓶颈问题。

8-5 常用的缓冲技术有哪些？

解答：常用的缓冲技术有双缓冲、环形缓冲和缓冲池。

引入双缓冲可以提高处理机与设备之间的并行操作程度。例如，输入设备先将第一个缓冲区装满数据，在输入设备向第二个缓冲区装数据时，处理机就可以从第一个缓冲区中取出数据进行处理；当第一个缓冲区的数据处理完毕，若第二个缓冲区已经装满数据，则处理机又可以从第二个缓冲区中取出数据进行处理，而输入设备又可向第一个缓冲区装填数据。

为了在 CPU 与外设对信息的操作速度相差甚远时仍能得到良好并行效果，可以采用环形缓冲技术。环形缓冲技术是在主存中分配一组大小相等的存储区作为缓冲区，并将这些缓冲区链接起来，每个缓冲区中有一个指向下一个缓冲的指针，最后一个缓冲区的指针指向第一个缓冲区，这样 n 个缓冲区就成了一个环形。此外，系统中有个缓冲区链首指针指向第一个缓冲区。环形缓冲区用于输入输出时，还需要两个指针 in 和 out。其中，in 指向第一个空缓冲区；out 指向第

一个装满数据的缓冲区。输入时，把数据输入到 **in** 所指的空缓冲区中，然后 **in** 模取后移一位，指向下一个空缓冲区；输出时，从 **out** 所指的满缓冲区中取出数据，然后 **out** 模取后移一位，指向下一个满缓冲区。

缓冲池是由若干个大小相等的缓冲区组成的。缓冲池中的每一个缓冲区都是由系统统一管理和动态分配。当某个进程需要使用缓冲区时便提出申请，由系统将缓冲区分配给它，当进程不再使用缓冲区时，就将缓冲区归还给缓冲池。这样，就可以用少量的缓冲区为更多的进程服务。缓冲池通常将缓冲区排成 **3** 个队列：空闲缓冲区队列、输入缓冲区队列和输出缓冲区队列。

8-8 什么是独占设备？对独占设备如何分配？

解答：独占设备是指在一段时间内只允许一个用户进程访问的设备。系统一旦把这类设备分配给某进程后，便由该进程独占直到使用完后释放。多数低速 **I/O** 设备都属于独占设备。如打印机就是典型的独占设备。

独占设备应采用独占分配方式，即将一个独占设备分配给某进程后便一直由它独占，直到该进程完成或释放该设备时，系统才能将该设备分配给其他进程。

8-9 什么是共享设备？如何进行共享分配？

解答：共享进程是指在一段时间内允许多个进程同时访问的设备。如磁盘就是典型的共享设备，若干个进程可以交替地从磁盘上读写信息。

对共享设备可将其同时分配给多个进程使用。共享分配方式显著提高了设备的利用率，但对设备的访问需进行合理的调度。

8-10 什么是虚拟设备技术？什么是虚拟设备？如何进行虚拟分配？

解答：所谓虚拟设备技术，是在一类物理设备上模拟另一个物理设备的技术，是将独占设备转换为共享设备的技术。目前最广泛流行的虚拟设备技术是 **SPOOLing** 技术。

虚拟设备是指通过虚拟技术将一台独占设备变换为若干台逻辑设备，供若干个用户进程使用，通常把这种经过虚拟技术处理后的设备成为虚拟设备。引入虚拟设备的目的是为了克服独占设备所具有的速度较慢、资源的利用率较低的缺点，以提高设备的利用率。

虚拟分配是针对虚拟设备而言的。当进程申请独占设备时，由系统分配给它共享设备（如磁盘）上的一部分存储空间；当进程要与设备交换信息（以输出为例）时，系统就将要交换的信息存放到这部分存储空间中；在适当的时候，系统再将存储空间中的信息传送到独占设备上。