第三章应用题参考答案

第三章（应用题部分） 2, 3, 5, 6, 17, 24, 29, 39, 47。

2 设有 n 个进程共享一个互斥段，如果：

(1)每次只允许一个进程进入互斥段；

(2)每次最多允许 m 个进程（m≤n）同时进入互斥段。

试问：所采用的信号量初值是否相同？信号量值的变化范围如何？

答：所采用的互斥信号量初值不同。

(1) 互斥信号量初值为 1，变化范围为[-n+1，1]。

当没有进程进入互斥段时，信号量值为 1；当有 1 个进程进入互斥段但没有进程等待进入互斥段时，信号量值为 0；当有 1 个进程进入互斥段且有一个进程等待进入互斥段时，信号量值为-1；最多可能有 n-1 个进程等待进入互斥段，故此时信号量的值应为-(n-1)也就是-n+1。

(2) 互斥信号量初值为 m，变化范围为[-n+m，m]。

当没有进程进入互斥段时，信号量值为 m；当有 1 个进程进入互斥段但没有进程等待进入互斥段时，信号量值为 m-1；当有 m 个进程进入互斥段且没有一个进程等待进入互斥段时，信号量值为 0；当有 m 个进程进入互斥段且有一个进程等待进入互斥段时，信号量值为-1；最多可能有 n-m 个进程等待进入互斥段，故此时信号量的值应为-(n-m)也就是-n+m。

3、有两个优先级相同的进程 P1 和 P2，各自执行的操作如下，信号量 S1 和 S2 初值均为 0。试问 P1、P2 并发执行后，x、y、z 的值各为多少？**(答案有 3 种情况)**

```
P1( ) {                P2( ) {
y=1;                    x=1;
y=y+3;                  x=x+5;
V(S1);                  P(S1);
z=y+1;                  x=x+y;
P(S2);                  V(S2);
y=z+y;                  z=z+x;
}                       }
```

答：现对进程语句进行编号，以方便描述。

```
y=1;        ①          x=1;        ⑤
y=y+3;      ②          x=x+5;      ⑥
V(S1);                 P(S1);
z=y+1;      ③          x=x+y;      ⑦
P(S2);                 V(S2);
y=z+y;      ④          z=z+x;      ⑧
```

①、②、⑤和⑥是不相交语句，可以任何次序交错执行，而结果是唯一的。接着无论系统如何调度进程并发执行，当执行到语句⑦时，可以得到 x=10，y=4。按 Bernstein 条件，语句③的执行结果不受语句⑦的影响，故语句③执行后得到 z=5。最后，语句④和⑧并发执行，这时得到了两种结果为：

语句④先执行：x=10，y=9，z=15。

语句⑧先执行：x=10，y=19，z=15。

此外，还有第三种情况，语句③被推迟，直至语句⑧后再执行，于是依次执行以下三个语句：

```
z=z+x;
z=y+1;
y=z+y;
```

这时 z 的值只可能是 y+1=5，故 y=z+y=5+4=9，而 x=10。

第三种情况为：x=10，y=9，z=5。

5 有一阅览室，读者进入时必须先在一张登记表上登记，该表为每一座位列出一个表目，包括座号、姓名，读者离开时要注销登记信息；假如阅览室共有 100 个座位。试用：(1）信号量和 P、V 操作；(2）管程，来实现用户进程的同步算法。

答：(1) 使用信号量和 P、V 操作：

```
struct {char name[10] ;
        int number;
    }A[100];
    semaphore mutex,seatcount;
    int i;mutex=1;seatcount=100;
  for(int i=0;i<100;i++ )
    {A[i].number=i;A[i].name=null;}
cobegin
process readeri(char readername[ ]) {                    //(i=1,2,…)
        P(seatcount);
        P(mutex);
          for (int i=1; i< 100 ;i++)
            if (A[i].name==null ) A[i].name=readername;
            reader get the seat number =i;                /*A[i].number*/
        V(mutex)
        {进入阅览室，座位号 i，座下读书};
        P(mutex);
          A[i].name=null;
        V(mutex);
        V(seatcount);
        离开阅览室;
       }
coend.
```

(2) 使用管程实现：

```
type readbook=MONTOR {
  semaphore R;
   int R_count, i, seatcount;
    char name[100] ;
    seatcount=0;
  InterfaceModule IM;
  DEFINE readbook( ),readerleave( );
```

```
    USE enter(), leave(),wait( ), signal( );
  void readercome(char readername[ ]) {
        enter (IM);
          if (seatcount>=100) wait(R,R_count,IM);
          seatcount=seatcount+1;
          for (int i=0;i<100;i++) {
           if (name[i]==null ) name[i]=readername;
          }
          get the seat number=i;
        leave(IM );
   }
  void readerleave(char readername) {
       enter(IM);
          seatcount--;
          for(int i=0;i<100;i++)
                if (name[i]==readername) name[i]=null;
          signal(R,R_count,IM);
       leave(IM);
  }
    cobegin
    process reader i ( ) {            //i=1,2.…
          readbook.readercome(readername);
          read the book;
          readbook.readerleave(readername);
          leave the readroom;
      }
coend
```

6　在一个盒子里，混装了数量相等的黑白围棋子。现在用自动分拣系统把黑子、白子分开，设分拣系统有二个进程 P1 和 P2，其中 P1 拣白子；P2 拣黑子。规定每个进程每次拣一子；当一个进程在拣时，不允许另一个进程去拣；当一个进程拣了一子时，必须让另一个进程去拣。试写出两进程 P1 和 P2 能并发正确执行的程序。

答 1：实质上是两个进程的同步问题，设信号量 S1 和 S2 分别表示可拣白子和黑子，不失一般性，若令先拣白子。

```
   semaphore S1,S2;
S1=1;S2=0;
cobegin
```

process P1( ) {
while(true) {
P(S1);
拣白子
V(S2);
}
}
	process P2( ) {
	while(true) {
	P(S2);
	拣黑子
	V(S1);
	}
	}

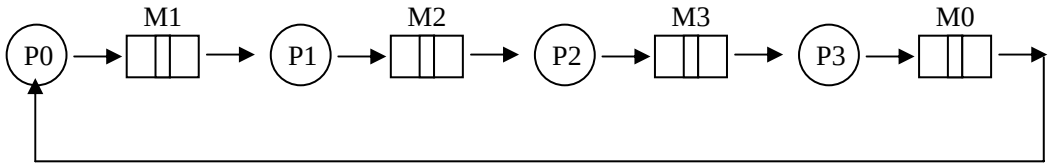
```
coend
```

答 2：(若给出的管程方法正确，也给满分)

```
type pickup_chess= MONITOR    {
     bool flag; flag=true;
     semaphore S_black,S_white;
     int S_black_count,S_white_count;
     InterfaceModule IM;
    DEFINE pickup_black ,pickup_white
    USE enter,leave,wait,signal；
  void pickup_black ( ) {
            enter(IM);
              if (flag) wait(S_black,S_black_count,IM);
              flag=true;
              pickup a black;
              signal(S_white,S_white_count,IM);
            leave(IM );
     }

void pickup_white( ) {
            enter(IM);
              if(!flag) wait(S_white,S_white_count,IM);
              flag=false;
              pickup a white;
              signal(S_black,S_black_count,IM);
            leave(IM);
     }
          cobegin
              process_B( ); process_W( );
          coend
     process_B( ) {
              pickup_chess.pickup_black( );
              other;
        }
    process_W( ) {
              pickup_chess.pickup_white( );
              other;
     }
```

17、四个进程 Pi（i=0…3）和四个信箱 Mj（j=0…3），进程间借助相邻信箱传递消息，即 Pi 每次从 Mi 中取一条消息，经加工后送入 M(i+1)mod4，其中 M0、M1、M2、M3 分别可存放 3、3、2、2 个消息。初始状态下，M0 装了三条消息，其余为空。试以 P、V 操作为工具，写出 Pi（i=0…3）的同步工作算法。



答：

semaphore mutex1,mutex2,mutex3,mutex0;
mutex1=mutex2=mutex3=mutex0=1;
semaphore empty0,empty1,empty2,empty3;
empty0=0;empty1=3;empty2=2;empty3=2;
semaphore full0,full1,full2,full3;
full0=3;full1=full2=full3=0; // M0 装了三条消息，其余为空。此处赋初值 full0=3，其余为 0。
int in0,in1,in2,in3,out0,out1,out2,out3;
in0=in1=in2=in3=out0=out1=out2=out3=0;
cobegin

process P0( ) {
while(true) {
P(full0);
P(mutex0);
{从 M0[out0]取一条消息};
out0=(out0+1) % 3;
V(mutex0);
V(empty0);
{加工消息};
P(empty1);
P(mutex1);
{消息存 M1[in1]};
in1=(in1+1) % 3;
V(mutex1);
V(full1);
}
}
	process P1( ) {
	while(true) {
	P(full1);
	P(mutex1);
	{从 M1[out1]取一条消息};
	out1=(out1+1) % 3;
	V(mutex1);
	V(empty1);
	{加工消息};
	P(empty2);
	P(mutex2);
	{消息存 M2[in2]};
	in2=(in2+1) % 2;
	V(mutex2);
	V(full2);
	}
	}

process P2( ) {
while(true) {
P(full2);
P(mutex2);
{从 M2[out2]取一条消息};
out2=(out2+1) % 2;
V(mutex2);
V(empty2);
加工消息;
	process P3( ) {
	while(true) {
	P(full3);
	P(mutex3);
	{从 M3[out3]取一条消息};
	out3=(out3+1) % 2;
	V(mutex3);
	V(empty3);
	{加工消息};

P(empty3);
P(mutex3);
{消息存 M3[in3]};
in3=(in3+1) % 2;
V(mutex3);
V(full3);
}
}
	P(empty0);
	P(mutex0);
	{消息存 M0[in0]};
	in0=(in0+1) % 3;
	V(mutex0);
	V(full0);
	}
	}

coend

24　系统有 A、B、C、D 共 4 种资源，在某时刻进程 P0、P1、P2、P3 和 P4 对资源的占有和需求情况如表，试解答下列问题：

Process	Allocation	Claim	Available
	A B C D	A B C D	A B C D
P_0	0 0 3 2	0 0 4 4	1 6 2 2
P_1	1 0 0 0	2 7 5 0	
P_2	1 3 5 4	3 6 10 10	
P_3	0 3 3 2	0 9 8 4	
P_4	0 0 1 4	0 6 6 10	

1)系统此时处于安全状态吗？

2)若此时进程 P2 发出 request1(1, 2, 2, 2)，系统能分配资源给它吗？为什么？

答：(1)系统处于安全状态，存在安全序列：P0，P3，P4，P1，P2。

	CurrentAvail	C_{ki}-A_{ki}	Allocation	CurrentAvail+allocation	Possible
	A B C D	A B C D	A B C D	A B C D	
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

(2)不能分配，否则系统会处于不安全状态。

若执行 P2 发出的 request1(1, 2, 2, 2)，此时 Available=(0, 4, 0, 0)，不能满足后续分配。

29　进程 A1、A2、…、An1 通过 m 个缓冲区向进程 B1、B2、…、Bn2 不断地发送消息。发送和接收工作符合以下规则：

(1) 每个发送进程每次发送一个消息，写进一个缓冲区，缓冲区大小与消息长度相等；

(2) 对每个消息，B1、B2、…、Bn2 都需接收一次，并读入各自的数据区内；

(1) 当 M 个缓冲区都满时，则发送进程等待，当没有消息可读时，接收进程等待。

试用信号量和 PV 操作编制正确控制消息的发送和接收的程序。

答：本题是生产者－消费者问题的一个变形，一组生产者 A1，A2，…An1 和一组消费者 B1，B2，…Bn2 共用 m 个缓冲区，每个缓冲区只要写一次，但需要读 n2 次。因此，可以把这一组缓冲区看成 n2 组缓冲区，每个发送者需要同时写 n2 组缓冲区中相应的 n2 个缓冲区，而每一个接收者只需读它自己对应的那组缓冲区中的对应单元。

应设置一个信号量 mutex 实现诸进程对缓冲区的互斥访问；两个信号量数组 empty[n2] 和 full[n2]描述 n2 组缓冲区的使用情况。其同步关系描述如下：

```
  semaphore mutex,empty[n2],full[n2];
  int i;mutex=1;
  for(i=0;i<n2;i++) { empty[i]=m; full[i]=0; }
main( ) {
cobegin
       A1( );
       A2( );
       ┇
      An1 ();
      B1 ();
      B2 ();
       ┇
      Bn2( );
coend
}
```

void send ( ) { /*进程 Ai 发送消息*/
for (int i=0;i<n2;i++)
P(empty[i]);
P(mutex);
将消息放入缓冲区；
V（mutex）;
for(int i=0;i<n2;i++)
V(full[i]);
}
	Ai ( ) { /*发送进程 A1,A2,…An1 的程序类似，这里给出进程 Ai 的描述*/
	while (true) {
	send ( );
	┇
	}
	}

void receive(i) { /*进程 Bi 接收消息*/
P(full[i]);
P(mutex);
{将消息从缓冲区取出}；
V(mutex);
V(emtpy[i]);
}
	Bi ( ) { /*接收进程 B1,B2,…Bn2 的程序类似，这里给出进程 Bi 描述*/
	while (true) {
	receive(i);
	┇
	}
	}

39 一组生产者进程和一组消费者进程共享九个缓冲区，每个缓冲区可以存放一个整数。生产者进程每次一次性向 3 个缓冲区写入整数，消费者进程每次从缓冲区取出一个整数。请用：(1)信号量和 P、V 操作，(2)管程，写出能够正确执行的程序。

答：(1)信号量和 P、V 操作。

```
var   int buf[9];
       int count,getptr,putptr;
       count=0;getpt=0;putpt=0;
```

```
    semaphoreS1,S2,SPUT,SGET;
    S1=1;S2=1;SPUT=1;SGET=0;
main( ) {
   cobegin
    producer-i();consumer-j();
   coend
}
```

process producer-i( ) {
while(true) {
{生产 3 个整数};
P(SPUT);
P(S1);
buf[putptr]=整数 1;
putptr=(putptr+1) % 9;
buf[putptr]=整数 2;
putptr=(putptr+1) % 9;
buf[putptr]=整数 3;
putptr=(putptr+1) % 9;
V(SGET);
V(SGET);
V(SGET):
V(S1);
}
}
	process consumer-j( ) {
	int y;
	while(true) {
	P(SGET);
	P(S2);
	y=buf[getptr];
	getptr=(getptr+1) % 9;
	count++;
	if (count==3)
	{count=0;V(SPUT);}
	V(S2);
	{consume the 整数 y};
	}
	}

(2) 管程/生产者消费者。

```
TYPE    get_put = MONITOR
          int buf [9];
          int count,getptr,putptr;
          semaphore SP,SG;
          int SP_count,SG_count;
         count:=0;getptr:=0;putptr:=0;
InterfaceModule IM;
DEFINE put, get;
USE wait, signal, enter, leave;
```

void put(int a1,int a2,int a3) {
enter(IM);
if (count>6) wait(SP,SP_count,IM);
count=count+3;
buf[putptr]=a1;
putptr=(putptr+1) % 9;
buf[putptr]=a2;
putptr=(putptr+1) % 9;
buf[putptr]=a3;
putptr=(putptr+1) % 9;
signal(SG,SG_count,IM);
	void get(int b) {
	enter(IM);
	if (count= =0) wait(SG,SG_count,IM);
	b=buf[getptr];
	getptr=(getptr+1) % 9;
	count=count++;
	if (count < 7) signal(SP,SP_count, IM);
	else if (count > 0) signal(SG,SG_count,IM);
	leave(IM);
	}

leave(IM); }	

```
cobegin
   process producer-i( ) {
      while(true) {
      {生产 3 个整数};
     get-put.put(a1,a2,a3);
   }
}
process consumer-j( ) {
      while(true) {
      get-put.get(b)
      {consume the  整数 b};
      }
   }
coend
```

47　设儿童小汽车生产线上有一只大的储存柜，其中有 N 个槽（N 为 5 的倍数且其值≥5），每个槽可存放一个车架或一个车轮。设有三组生产工人，其活动如下：

组 1 工人的活动	组 2 工人的活动	组 3 工人的活动
L1：加工一个车架； 车架放入柜的槽中。	L2：加工一个车轮； 车轮放入柜的槽中。	L3：在槽中取一个车架； 在槽中取四个车轮， 组装为一台小汽车。
goto L1:;	goto L2:;	goto L3:;

试用(1)信号量及 P,V 操作，(2)管程方法正确实现这三组工人的生产合作工作。

解：(1) 信号量及 P，V 操作

将柜子的 N 个槽口分为两部分：N/5 和 4N/5，分别装入车架和车轮

```
   车架  box1[N/5];
   车轮  box2[4*N/5] ;
   semaphore mutex1,mutex2,S1,S2,S3,S4;
   int counter,in1,in2,out1,out2;
   S1=N/5;S2=4N/5;S3=S4=0;
   counter=in1=in2=out1=out2=0;
   mutex1=mutex2=1;
 cobegin
   process worker1( ) {
         while(true) {
            加工一个车架;
            P(S1);
            P(mutex1);
            {车架放入 box1(in1)};
              in1=(in1+1) % (N/5);
              V(mutex1);
```

```
                V(S3);
                }
        }
    process worker2( ) {
            while(true) {
            加工一个车轮;
              P(S2);
              P(mutex2);
              {车轮放入 box2(in2)};
                in2=(in2+1) % (4*N/5);
                counter=counter+1;
                if(counter==4)    {counter=0;V(S4);}
                V(mutex2);
                }
        }
    process worker3( ) {
            while(true) {
              P(S3);
              P(mutex1);
              取车架从 box1(out1);
              out1=(out1+1) % (N/5);
              V(mutex1)
              V(S1);
              P(S4);
              P(mutex2);
              取车轮从 box2(out2);
                out2=(out2+1) % (4*N/5);
              取车轮从 box2(out2);
                out2=(out2+1) % (4*N/5);
              取车轮从 box2(out2);
                out2=(out2+1) % (4*N/5);
              取车轮从 box2(out2);
                out2=(out2+1) % (4*N/5);
                V(S2);
                V(mutex2);
                装配车子;
                }
        }
coend
```

(2)用管程方法.汽车厂。

```
type    produce_toy_car=monitor
          车架 box1[N/5];
          车轮 box2[4*N/5];
          semaphore S1,S2,S3,S4;
          int S1_count,S2_count,S3_count,S4_count;
          int counter1,counter2,count,in1,in2,out1,out2;
          counter1=counter2=count=in1=in2=out1=out2=0;
InterfaceModule IM;
DEFINE put1,put2,take;
USE wait,signal,enter,leave;
   void put1( ) {
      enter(IM);
       if(counter1==N/5)    wait(S1,S1_count,IM);
       {车架放入 box1(in1)};
       in1=(in1+1) % (N/5);
       counter1=counter1+1;
       signal(S3,S3_count,IM);
       leave(IM);
     }
   void put2( ) {
          enter(IM);
           if(counter2==(4*N/5))    wait(S2,S2_count,IM);
          车轮放入 box2(in2);
          in2=(in2+1) % (4*N/5);
          counter2=counter2+1;
          count=count+1;
          if(count==4)    {count=0; signal(S4,S4_count,IM);}
          leave(IM);
       }
     void take( ) {
           enter(IM);
          if(counter1==0)    wait(S3,S3_count,IM);
          取车架从 box1(out1);
           out1=(out1+1) % (N/5);
           counter1=counter1-1;
           if (counter2<4)    wait(S4,S4_count,IM);
           取车轮从 box2(out2);
          out2=(out2+1) % (4*N/5);
          取车轮从 box2(out2);
          out2=(out2+1) % (4*N/5);
          取车轮从 box2(out2);
          out2=(out2+1) % (4*N/5);
          取车轮从 box2(out2);
```

```
    out2=(out2+1) % (4*N/5);
    counter2=counter2-4;
    signal(S1,S1_count,IM);
    signal(S2,S2_count,IM);
    leave(IM);
}
```