Homework 3

3150103823 韩熠星

1. 设与某资源关联的信号量初值为3，当前值为1，若M表示该资源的可用个数，N表示等待资源的进程数，则M、N分别是____

A．0、1　　B．1、0　　C．1、2　　D．2、0

B：初始值3，当前1，也就是有两个进程正在占用资源，还可以进入一个，没有进程等待。

2. 有两个进程P1和P2描述如下:

```
shared data:
int counter = 6;

P1 :
    Computing;
    counter=counter+1;

P2 :
    Printing;
    counter=counter-2;
```

两个进程并发执行，运行完成后，counter的值不可能为。

A. 4　　B. 5　　C. 6　　D. 7

C

3. 在执行V操作时，当信号量的值，应释放一个等待该信号量的进程___。

A．小于0　　B．大于0　　C．小于等于0　　D．大于等于0

C： v 操作的作用是归还资源，将所申请的资源数加一，然后判断资源数是否小于等于 0，若小于等于 0 说明有进程阻塞在当前资源上，唤醒一个当前资源链表中的进程。

4. 在消息缓冲通信方式中，临界资源为_____。

A.发送进程　　B.消息队列　　C.接收进程　　D.信箱

B

5. 有9个生产者，6个消费者，共享容量为8的缓冲区。在这个生产者-消费者问题中，互斥使用缓冲区的信号量mutex的初值应该为___。

A．1　　B．6　　C．8　　D．9

A

6. 在操作系统中，信号量表示资源，其值_____。

A.只能进行加减乘除运算来改变　　B.进行任意的算术运算来改变

C.只能进行布尔型运算来改变　　D.仅能用初始化和P、V操作来改变

D

7. 在解决进程间同步和互斥机制中，有一种机制是用一个标志来代表某种资源的状态，该标志称为____。
A. 共享变量　　B. flag　　C. 信号量　　D. 整型变量

C

8. 下列哪一个问题只包含进程互斥问题?
A. 田径场上的接力比赛
B. 两个进程都要使用打印机
C. 一个生产者和一个消费者通过一个缓冲区传递产品
D. 公共汽车上司机和售票员的协作

B

9. 下列哪种方法不能实现进程之间的通信?
A. 共享文件　　B. 数据库　　C. 全局变量　　D. 共享内存

A

10. 我们把在一段时间内，只允许一个进程访问的资源，称为临界资源，因此，我们可以得出下列论述，请选择一条正确的论述。
A. 对临界资源是不能实现资源共享的。
B. 对临界资源，应采取互斥访问方式，来实现共享。
C. 为临界资源配上相应的设备控制块后，便能被共享。
D. 对临界资源应采取同时访问方式，来实现共享。

D

11. 在生产者和消费者问题中，信号量mutex，empty，full的作用是什么？如果对调生产者进程中的两个wait操作和两个signal操作，则可能发生什么情况?

信号量mutex：是保证各生产者进程和消费者进程对缓冲池的互斥访问。

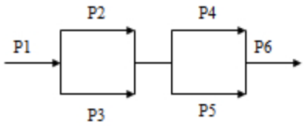
信号量empty和full：资源信号量，它们分别对应于缓冲池中的空闲缓冲区和缓冲池中的产品，生产者需要通过 wait(empty)来申请使用空闲缓冲区，而消费者需要通过 wait(full)才能取得缓冲中的产品.**所以这两个信号量起着同步生产者和消费者的作用，它们保证生产者不会将产品存放到满缓冲区中，而消费者不会从空缓冲区中取产品。**

如果将两个 wait 操作，即 wait(full)和 wait(mutex)互换位置， 或者 wait(empty)和 wait(mutex)互换位置，都可能引起死锁。

> 如果系统中缓冲区全满时，若一生产者进程先执行了 wait(mutex)操作并获得成功，当再执行wait(empty)操作时，它将因失败而进入阻塞状态，它期待消费者执行signal(empty)来唤醒自己，在此之前，它不可能执行signal(mutex)操作，从而使企图通过 wait(mutex)进入自己的临界区的其他生产者和所有的 消费者进程全部进入阻塞状态，系统进入死锁状态。

将signal(full)和 signal(mutex)互换位置，或者 signal(empty) 和 signal(mutex)互换位置，则不会引起死锁，其影响只是使某个临界资源的释放略为推迟一些。

12. 一组合作进程，执行顺序如下图。请用wait、signal操作实现进程间的同步操作。



各进程的执行顺序

可设置8个信号量 a、b、c、d、e、f、g、h，它们的初值均为0，而相应的进程可描述为（其中“…”表示进程原来的代码）：

```
main()
cobegin{
  Process P1() {  …;        signal(a);  signal(b);             }
  Process P2() {  wait(a);  …;          signal(c);  signal(d);  }
  Process P3() {  wait(b);  …;          signal(e);  signal(f);  }
  Process P4() {  wait(c);  wait(e);    …;          signal(g);  }
  Process P5() {  wait(d);  wait(f);    …;          signal(h);  }
  Process P6() {  wait(g);  wait(h);    …;                      }
}coend
```

13. 试从“互斥”（mutual exclusion）、"空闲让进"（progress）、"有限等待"(bounded waiting)三方面讨论程序中用软件方法解决二个进程互斥访问临界区问题。
下述关于双进程临界区问题的算法（对编号为id的进程）是否正确：

```
do{
blocked[id]=true;
while(turn !=id)
      {
while(blocked[1-id]);
turn=id;
      }
编号为id的进程的临界区
          blocked[id]=false;
编号为id的进程的非临界区
   } while (true)
```

其中，布尔型数组blocked[2]初始值为为{false,false}，整型turn初始值为0，id代表进程编号（0或1）。请说明它的正确性，或指出错误所在

1. 互斥 P1,P2不会同时进入临界区，满足互斥条件。

2. 有空进让 设开始无进程在临界区中，P0执行`turn=0`进入临界区，当P0退出临界区时，执行`blocked[0]=flase`，使P1得以进入临界区，P1先执行的情况类似，所以满足有空进让的原则。

3. 有限等待 假定进程P0在临界区执行，进程P1申请进入临界区，则因进程P0会在有限时间内执行完并退出临界区，然后，将执行`blocked[0]=flase`，这使得进程P1因`blocked[0]`值为`0`,而使`turn`值变为`1`,立即可进入临界区。因而，能满足有限等待的原则。

综上所述，该算法可以解决双进程临界区问题。

14. 三个进程P1、P2、P3互斥使用一个包含N（N>0）个单元的缓冲区。P1每次用produce()生成一个正整数并用put()送入缓冲区某一个空单元中；P2每次用getodd()从该缓冲区中取出一个奇数并用countodd()统计奇数个数；P3每次用geteven()从该缓冲区中取出一个偶数并用counteven()统计偶数个数。请用信号量机制实现这三个进程的同步与互斥活动，并说明所定义的信号量的含义。要求用伪代码描述。

```
semaphore mutex=1;                  // 用于互斥访问缓冲区
semaphore odd=0,even=0;
semaphore empty=N;
main()
cobegin{
    Process P1()
    while(True)
    {
        x=produce();                //生成一个数
        P(empty);                   //判断缓冲区是否有空单元
        P(mutex);                   //缓冲区是否被占用
        Put();
        V(mutex);                   //释放缓冲区
        if(x%2==0)
            V(even);                //如果是偶数，向P3发出信号
        else
            V(odd);                 //如果是奇数，向P2发出信号
    }
    Process P2()
    while(True)
    {
        P(odd);                     //收到P1发来的信号，已产生一个奇数
        P(mutex);                   //缓冲区是否被占用
        getodd();
        V(mutex);                   //释放缓冲区
        V(empty);                   //向P1发信号，多出一个空单元
        countodd();
    }
    Process P3()
    while(True)
    {
        P(even);                    //收到P1发来的信号，已产生一个偶数
        P(mutext);                  //缓冲区是否被占用
        geteven();
        V(mutex);                   //释放缓冲区
        V(empty);                   //向P1发信号，多出一个空单元
        counteven();
    }
}coend
```