进程的同步与互斥问题

版权所有
韩芳溪
山东大学计算机科学与技术学院
hfx@sdu.edu.cn

1、wait 与 signal 为什么要设计成原语？

2、一个输入进程向一个缓冲区中输入数据，另一个输出进程从缓冲区中取出数据输出。缓冲区中每次只能存放一个数。

3、三个进程共享一个缓冲区。一个计算进程送数；一个加工进程取出加工，然后将加工结果再送回缓冲区；一个输出进程将加工后的数据取出打印。缓冲区中每次只能存放一个数。

4、三个进程共享一个缓冲区。一个负责向缓冲区送数；一个取偶数输出，另一个取奇数输出。缓冲区中每次只能存放一个数。

5、四个进程共享一个缓冲区，一个送偶数，一个送奇数，一个取偶数，一个取奇数。缓冲区中每次只能存放一个数。

6、围棋问题：数量相等的黑子与白子混在一起，利用两个进程分开。一个进程拣白子，另一个进程拣黑子。要求：

（1）一个进程拣了一个子，必须让另一个进程拣子；即两个进程应交替拣子；

（2）假定先拣黑子。

7、要求下列四条语句正确执行

s1：a:=x+y;

s2：b:=z+1;

s3：c:=a-b;

s4：w:=c+1;

将其抽象成前趋图，然后解决；

该问题也可以衍生出四个进程之间的相互制约。（***举例三个进程之间的相互制约***）

若以线段表示进程，转换成前趋图的形式。

8、有一个仓库，可以存放 X 与 Y 两种产品，仓库的存储空间足够大，但要求：

（1）每次只能存入一种产品（X 或 Y）；

（2）-N<A 产品数量 – B 产品数量<M；

其中，N 和 M 是正整数。试用“存放 A”和“存放 B”和 wait、signal 描述产品 A 与产品 B 的入库过程。

9、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。发送和接收工作遵循如下规则：

（1）每个进程发送一个消息，写入一个缓冲区，缓冲区大小与消息长度一样；

（2）对每一个消息，B1，B2，……，Bn2 都需各接收一次，读入各自的数据区中；

（3）m 个缓冲区都满时，发送进程等待，没有可读取的消息时，接收进程等待。

试用 wait 与 signal 操作组织正确的发送和接收操作。

10、有一个仓库存放两种零件 A 和 B，最大库容为各为 m 个。有一个车间不断地取 A 和 B 进行装配，每次各取一个。为避免零件锈蚀，*遵循先入库者先出库的原则*。有两组供应商分别不断地供应 A 和 B（每次一个）。为保证齐套和合理库存，当某种零件的数量比另一种的数量超过 n（n<m）个时，暂停对数量大的零件的进货，集中补充数量少的零件。试用 wait 和 signal 正确实现之。

11、某高校计算机系开设网络课并安排上机实习。假定机房共有 2m 台机器，有 2n 个学生选该课，规定：

（1）每两个学生组成一组，各占一台机器，协同完成上机实习；

（2）只有一组两个学生到齐，并且此时机房有空闲机器时，该组学生才能进入机房；

（3）上机实习由一名教师检查，检查完毕，一组学生同时离开机房。

试用 wait 和 signal 正确实现之。

12、对于读者写者问题，

（1）说明进程间的相互制约关系，应设哪些信号量？

（2）用 wait 和 signal 写出其同步算法。

（3）修改上述算法，使它对写者优先，即一旦有写者到达，后续的读者都必须等待，而无论是否有读者在读文件。

13、司机与售票员问题

在公共汽车上，司机和售票员的工作流程如下所示。为保证乘客安全，司机和售票员应密切配合协调工作。请用 wait、signal 操作来实现司机与售票员之间的同步。

司机－>（loop）{ 启动车辆－>正常行车－>到站停车 }

售票员－>（loop）{ 上乘客－>关车门－> 售票－> 开车门－>下乘客 }

14、汽车过桥问题（两套信号量，两个读者进程即可，即每个读者既是本方的读者，又充当对方的写者）

15、考虑一个无限长的消息队列的同步问题；

16、Sleeping Barber Problem

17、Cigarette Smoker’s Problem

18、某数据采集与处理系统由一个数据采集进程与一个数据处理进程组成，它们共享一个缓冲区，

（1）描述两进程之间的制约关系；

（2）请利用记录型信号量机制和 wait、signal 操作解决这两个进程的同步问题，写出相应的算法描述；

19、某媒体播放器由一组循环使用的缓冲区及两个并发的播放进程与接收进程组成，其中，

(1) 8 个缓冲区构成一个循环链表，用于缓存要播放的媒体流；

(2) 接收进程负责从服务器端接收欲播放的媒体流，并依次放入缓冲区中；

(3) 播放进程依次从缓冲区中取出媒体流播放；

请利用信号量机制和 wait、signal 操作解决这两个进程的同步问题，写出相应的算法描述；

20、为某临界区设置一把锁 W，当 W＝1 时表示关锁，当 W＝0 时表示锁已经打开。试写出开锁原语与关锁原语，并利用他们实现互斥。

21、在生产者－消费者问题中，交换两个 signal 操作测次序会出现什么结果？交换两个 signal 操作呢？说明理由。

22、设有三个进程 A、B、C，其中 A 与 B 构成一对 P－C 问题，共享一个由 n 个缓冲区组成的缓冲池；B 与 C 构成一对 P－C 问题，共享一个由 m 个缓冲区组成的缓冲池。试用记录型信号量机制及 wait 与 signal 操作实现他们的同步。

23、有一阅览室，共有 100 个座位。读者进入时必须先在一张登记表上登记，该表为每一作为列一个目录，包括座号与读者姓名。读者离开时要销掉登记内容。试用记录型信号量机制及 wait 与 signal 操作描述读者之间的同步。

参考答案:（注：可能有错误，仅供参考，欢迎批评指正）

1、wait 与 signal 为什么要设计成原语？

对于 wait 操作，分析当 s=1 时两个进程并发时引起不该阻塞的进程进入阻塞状态。

对于 signal 操作，分析当 s=-1 时两个进程并发时引起该唤醒的进程没有被唤醒。

2、一个输入进程从键盘接收数据，送入一个缓冲区中，另一个输出进程从缓冲区中取出数据输出。（假定缓冲区开始是空的，且每次只能存放一个数据）

```
Var mutex, in, out:semorphore:=1,1,0;
    Buffer:integer;
Begin
  Parbegin
    Input:begin
            repeat
              从磁盘获取数据;
              wait(in);                  //解释同步在前，互斥在后，否则可能引起死锁
              wait(mutex);               //discussion:   mutex is not necessary.
              add data to the buffer;
              signal(mutext);
              signal(out);
            until false
          end
    Output:begin
            repeat
              wait(out);
              wait(mutex);
              add data to the buffer;
              signal(mutext);
              signal(in);
            until false
          end
  Parend
```

3、三个进程共享一个缓冲区。一个计算进程送数；一个加工进程取出加工，然后将加工结果再送回缓冲区；一个输出进程将加工后的数据取出打印。缓冲区中每次只能存放一个数。

设定四个信号量。（注：mutext 在此不是必须的）

```
Var mutex, s1,s2,s3:semorphore:=1,1,0,0;
    Buffer:integer;
```

```
Begin
  Parbegin
    Input:begin
          repeat
            从磁盘获取数据；
            wait(s1);
            wait(mutex);          //discussion:   mutex is not necessary.
            add data to the buffer;
            signal(mutext);
            signal(s2);
          until false
        end
    compute::begin
          repeat
            从磁盘获取数据；
            wait(s2);
            wait(mutex);          //discussion:   mutex is not necessary.
            add data to the buffer;
            signal(mutext);
            signal(s3);
          until false
        end
    Output:begin
          repeat
            从磁盘获取数据；
            wait(s3);
            wait(mutex);          //discussion:   mutex is not necessary.
            add data to the buffer;
            signal(mutext);
            signal(s1);
          until false
        end

  Parend
```

4、三个进程共享一个缓冲区。一个负责向缓冲区送数；一个取偶数输出，另一个取奇数输出。缓冲区中每次只能存放一个数。

设定四个信号量。（注：mutext 在此不是必须的）

```
Var mutex, s1,s2,s3:semorphore:=1,1,0,0;   //解释信号量的功能
    Buffer:integer;
Begin
  Parbegin
    Input:begin
            repeat
              wait(s1);
              wait(mutex);              //discussion:   mutex is not necessary.
              add X to the buffer;
              signal(mutext);
              if X is an even number then
                signal(s2);   //inform the process to get an even number
              else
                signal(s3);     //inform the process to get an odd number
            until false
          end
    get_even:begin
            repeat
              wait(s2);
              wait(mutex);              //discussion:   mutex is not necessary.
              get an even number from the buffer;
              signal(mutext);
              signal(s1);
            until false
          end
    get_odd:begin
            repeat
              wait(s3);
              wait(mutex);              //discussion:   mutex is not necessary.
              get an odd number from the buffer;
              signal(mutext);
              signal(s1);
            until false
          end
  Parend
```

5、四个进程共享一个缓冲区，一个送偶数，一个送奇数，一个取偶数，一个取奇数。缓冲区中每次只能存放一个数。

设定四个信号量。（注：mutext 在此不是必须的）

```
Var mutex, s1,s2,s3:semorphore:=1,1,0,0;   //解释信号量的功能
    Buffer:integer;
Begin
  Parbegin
    Input_even:begin
            repeat
               wait(s1);
               wait(mutex);              //discussion:   mutex is not necessary.
               add X to the buffer;
               signal(mutext);
               signal(s2);   //inform the process to get an even number
            until false
          end
    Input_odd:begin
            repeat
               wait(s1);
               wait(mutex);              //discussion:   mutex is not necessary.
               add X to the buffer;
               signal(mutext);
               signal(s3);     //inform the process to get an odd number
            until false
          end
    get_even:begin
            repeat
               wait(s2);
               wait(mutex);              //discussion:   mutex is not necessary.
               get an even number from the buffer;
               signal(mutext);
               signal(s1);
            until false
          end
    get_odd:begin
            repeat
               wait(s3);
               wait(mutex);              //discussion:   mutex is not necessary.
               get an odd number from the buffer;
               signal(mutext);
```

```
            signal(s1);
          until false
        end
  Parend
```

6、围棋问题：数量相等的黑子与白子混在一起，利用两个进程分开。一个进程拣白子，另一个进程拣黑子。要求：

（1）一个进程拣了一个子，必须让另一个进程拣子；即两个进程应交替拣子；

（2）假定先拣黑子。

强调这是一个同步问题，而不是一个互斥问题。

```
Var   black,white :semorphore:=1,0;     //get a black chessman first
    Buffer:integer;
Begin
  Parbegin
    Get_black_chessman:begin
          repeat
            wait(black);
            Get a black chessman from the box;
            signal(white);
          until false
        end
    Get_white_chessman:begin
          repeat
            wait(white);
            Get a black chessman from the box;
            signal(black);
          until false
        end
  Parend
```

7、要求下列四条语句正确执行

s1：a:=x+y;

s2：b:=z+1;

s3：c:=a-b;

s4：w:=c+1;

将其抽象成前趋图，然后解决；

该问题也可以衍生出四个进程之间的相互制约。

8、有一个仓库，可以存放 X 与 Y 两种产品，仓库的存储空间足够大，但要求：

（1）每次只能存入一种产品（X 或 Y）；

（2）-N<A 产品数量 – B 产品数量<M；

其中，N 和 M 是正整数。试用“存放 A”和“存放 B”和 wait、signal 描述产品 A 与产品 B 的入库过程。

分析：表达式“-N<A 产品数量 – B 产品数量<M”可以分解成两个表达式：“-N<A 产品数量 – B 产品数量”和“A 产品数量 – B 产品数量<M”，即“存放 A”的操作次数不能比“存放 B”的次数少 N-1 次，“存放 A”的操作次数不能比“存放 B”的次数多 M-1 次；

注：应该是一个程序；

```
Var    mutex,sa,sb :semorphore:=1,M-1,N-1;
Begin
  Parbegin
    Input:begin
            Repeat
              Get product X;
              IF X=A THEN
                begin
                  wait(sa);
                  wait(mutext);
                  put product X into the depository;
                  signal(mutext);
                  signal(sb);
                end
              ELSE IF X=B THEN
                begin
                  wait(sb);
                  wait(mutext);
                  put product X into the depository;
                  signal(mutext);
                  signal(sa);
                end
            until false
          end
  Parend
```

9、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。发送和接收工作遵循如下规则：

（1）每个进程发送一个消息，写入一个缓冲区，缓冲区大小与消息长度一样；

（2）对每一个消息，B1，B2，……，Bn2 都需各接收一次，读入各自的数据区中；

（3）m 个缓冲区都满时，发送进程等待，没有可读取的消息时，接收进程等待。

试用 wait 与 signal 操作组织正确的发送和接收操作。

说明：

1、该问题与传统的生产者—消费者问题不同的是，每个缓冲区只需写一次，但需读 n2 次。

2、信号量 mutext 实现诸进程对缓冲区的互斥访问

3、信号量 empty 代表空缓冲区的数目

4、对每一个缓冲区设置 n2 个信号量，即信号量数组 full[1..m,1..n2]，当生产者在一个缓冲区写入消息后，对该缓冲区信号量数组均执行 signal 操作，通知各消费者可以从该缓冲区中取走消息；当所有的消费者均从该缓冲区取走消息后，设定该缓冲区为空；

5、缓冲池中的缓冲区利用指针为 I（生产者）与 j（消费者）；

```
Var    mutex,empty, full[1..m,1..n2] :semorphore;
       Mutex_c:semaphor:=1;
       i,j,k,h,c:Integer;     //
       get_buffer_count [1..m]:array of integer;
       M:message;
       Buffer[0..m-1]:array of message;
Begin
  I:=0;
  J:=0;
  Mutex:=1;
  Empty:=m;
  For k:=1 to m do
    For h:=1 to n2 do
    Begin
      Full[k,h]:=0;
    end;
  For k:=1 to m do
    get_buffer_count[k]=1;
  Parbegin
    Ai(M):   begin
             repeat
               wait(empty);   //申请一个空缓冲区；
               wait(mutex);   //互斥访问缓冲池
               I:=(I+1) mod m;
               Buffer[I]:=m;
               signal(mutex);
               For k:=1 to m do
```

```
                For h:=1 to n2 do
                Begin
                   V(full[k,h]);
                end;
            until false
        end
Bi:   begin
            repeat
               wait(full[i, get_buffer_count[i]]);
               wait(mutex);
               Get a message M from the buffer[j] and saved it into private buffer;;
               signal(mutex);
               wait(mutex_c);   //实现对 get_buffer_count[i]的互斥访问
               get_buffer_count[i]:= get_buffer_count[i]+1;
               if get_buffer_count[i]=n2+1 then
                  begin
                     get_buffer_count[k]=1;
                     j:=(j+1) mod m;   //下次 n2 个消费者从下一个缓冲区取消息;
                     signal(empty);
                  end;
               signal(mutex_c);
            until false
        end
Parend
```

10、有一个仓库存放两种零件 A 和 B，最大库容为各为 m 个。有一个车间不断地取 A 和 B 进行装配，每次各取一个。为避免零件锈蚀，遵循先入库者先出库的原则。有两组供应商分别不断地供应 A 和 B（每次一个）。为保证齐套和合理库存，当某种零件的数量比另一种的数量超过 n（n<m）个时，暂停对数量大的零件的进货，集中补充数量少的零件。试用 wait 和 signal 正确实现之。

该题的控制关系有 4 个：

A 的数量 – B 的数量 <= n

B 的数量 – A 的数量 <= n

A 的数量 <= m

B 的数量 <= m

注意：下面的答案不能保证“遵循先入库者先出库的原则”，若要满足该要求，可以分别为产品 A 及产品 B 设置相应的缓冲区，设置两个指针跟踪入库的过程。取货物时可以用同一个指针跟踪出库过程，因为 A 与 B 必须配对出现。

```
Var    mutex, availa,fulla,sa ,availb,fullb,sb :semorphore;
```

```
Begin
   Mutex:=1;
   Availa:=m;
   Fulla=0;
   Sa:=n;
   Availb:=m;
   Fullb:=0;
   Sb:=n;

   Parbegin
      Input_A:begin
            Repeat
                  begin
                     wait(availa);
                     wait(sa);
                     wait(mutext);
                     put product A into the depository;
                     signal(mutext);
                     signal(fulla);
                     signal(sb);
                  end
            until false
         end
      Input_B:begin
            Repeat
                  begin
                     wait(availb);
                     wait(sb);
                     wait(mutex);
                     put product B into the depository;
                     signal(mutex);
                     signal(fullb);
                     signal(sa);
                  end
            until false
         end
      get_product:begin
            Repeat
```

```
            begin
                wait(fulla);
                wait(fullb);
                wait(mutex);
                get product A and B from the depository;
                signal(mutex);
                signal(availa);
                signal(availb);
            end
        until false
    end
Parend
```

11、某高校计算机系开设网络课并安排上机实习。假定机房共有 2m 台机器，有 2n 个学生选该课，规定：

（1）每两个学生组成一组，各占一台机器，协同完成上机实习；

（2）只有一组两个学生到齐，并且此时机房有空闲机器时，该组学生才能进入机房；

（3）上机实习由一名教师检查，检查完毕，一组学生同时离开机房。

试用 wait 和 signal 正确实现之。

```
Var    student,computer,enter,finished,check :semorphore;
Begin
Student:=0;
Computer:=2m;
Enter:=0;
Finished:=0;
Check:=0;
  Parbegin
    Monitor:begin
        Repeat
            begin
                wait(student);        //等待学生到达
                wait(student);        //等待另一学生到达
                signal(enter);        //允许学生进入
                signal(enter);        //允许另一学生进入
            end
        until false
    end
Student:begin
```

```
        Repeat
            begin
                signal(student);        //有学生到达
                wait(computer);       //获取一台计算机
                wait(enter);           //等待允许进入
                Do it with partner;
                signal(finished);     //实习完成
                wait(check);          //等待教师检查
                signal(computer);    //释放计算机资源
            end
        until false
    end
Teacher:begin
        Repeat
            begin
                wait(finished);       //等待学生完成
                wait(finished);       //等待另一学生完成
                check the work;
                signal(check);        //完成检查
                signal(check);        //完成检查
            end
        until false
    end
Parend
```

12、对于读者写者问题，

（1）说明进程间的相互制约关系，应设哪些信号量？

（2）用 wait 和 signal 写出其同步算法。

（3）修改上述算法，使它对写者优先，即一旦有写者到达，后续的读者都必须等待，而无论是否有读者在读文件。

解答：

（1）与（2）参见教材；

（3）增加信号量 w，用以在写进程到达时封锁后续的读者进程。

```
Var    rmutex,wmutex,w :semorphore:=1,1,1;
       Readcount:integer:=0;
Begin
  Parbegin
    reader:begin
```

```
        Repeat
            Begin
                wait(w);              //尽管有读者在读，但封锁刚到来的读者
                wait(rmutex);
                if readcount=0 then wait(wmutext);
                readcount:=readcount+1;
                signal(rmutex);
                signal(w);
                 perform read operation;
                wait(rmutext);
                readcount:=readcount-1;
                if readcount=0 then signal(wmutext);
                signal(rmutex);
            end
        until false
    end
writer:begin
        Repeat
            Begin
                Wait(w);
                wait(wmutex);
                 perform write operation;
                signal(wmutex);
                signal(w);
            end
        until false
    end
Parend
```

13、司机与售票员问题

在公共汽车上，司机和售票员的工作流程如下所示。为保证乘客安全，司机和售票员应密切配合协调工作。请用 wait、signal 操作来实现司机与售票员之间的同步。

司机－>（loop）{ wait(run) 启动车辆－>正常行车－>到站停车 signal(stop)}

售票员－>（loop）{ 上乘客－>关车门－> signal(run) 售票－> wait(stop) 开车门－>下乘客 }

14、汽车过桥问题（两套信号量，两个读者进程即可，即每个读者既是本方的读者，又充当对方的写者）

15、考虑一个无限长的消息队列的同步问题；

与经典生产者－消费者问题的区别在于生产者放入消息时不需要检查是否有空闲缓冲区；消费者取走消息后也就不需要做相应的 signal 操作。

生产者：

wait(mutex);

放入消息；

signal(mutex);

signal(full);

消费者：

wait(full);

wait(mutex);

取出消息；

signal(mutex);

消费；

16、The Sleeping Barber Problem

常量：CHAIRS　//椅子的个数

变量：int waiting=0; //记录等待服务的顾客数

信号量：

customers=0;　//等待的顾客数

barber=0;　//理发师是否准备好进行服务

mutex=1;　//互斥信号量（实现 waiting 的互斥访问）

Barber:

```
while (true)  {
    wait(customers);  //如果没有顾客，睡眠（等待顾客）
    wait(mutext);      //进入临界区，获得 waiting 的访问权
    waiting=waiting-1;
    signal(barber);    //理发师准备好可以服务
    signal(mutex);     //释放 waiting 的访问权
    cut-hair;          //理发
    }
```

Customer:

```
wait(mutex);
if (waiting < CHAIRS)  {    //如果没有空闲位，离开
    waiting=waiting +1;
    signal(customers);      //如果需要的话唤醒理发师（如果没有顾客，理发师睡眠）
    signal(mutex);
    wait(barber);  //如果理发师没有准备好，睡眠
    get_hair cut;   //坐下，接受服务
```

```
} else                  // 理发店已满，离开
{
   signal(mutex);
   leaving;
}
```

17、The Cigarette's Problem

```
semaphore tobacco_paper          = 0 // waiting for tobacco and paper
semaphore tobacco_matches        = 0 // waiting for tobacco and matches
semaphore paper_matches          = 0 // waiting for paper and matches
```

agent:

```
while( true )
        {
        pick a random number from 1-3
           {
           if random number is 1
                {
                                // put these two    ingredient on table
                                signal( tobacco_paper )
                }
           else if random number is 2
                {
                        // put these two    ingredient on table
                        signal( tobacco_matches )    // put on table
                }
           else if random number is 3
                {
                        // put these two    ingredient on table
                        signal( paper_matches )    // put on table
                }
                }

        wait( doneSmoking )
```

```
    } /* while */

smokers:
    // the smoker that has matches
    while( true )
        {
            wait( tobacco_paper ); /* picks up tobacco and paper */

            // roll cigarette and smoke

            signal( doneSmoking );
            }

    // the smoker that has paper
    while( true )
        {
            wait(tobacco_matches ) ; /* picks up tobacco and match */

            // roll cigarette and smoke

            signal( doneSmoking );
            }

    // the smoker that has tobacco
    while( true )
        {
            wait(paper_matches ) ; /* picks up paper and match */

            // roll cigarette and smoke

            signal( doneSmoking );
            }
```