**第三章作业**

班级：   07812201     学号：  1820221050      姓名：   丘绎楦

3-7. 设系统有n+1个进程，其中有n个发送进程和1个接收进程。其中A1、A2、**…**、An通过一个单缓冲区分别不断地向进程B发送消息，B不断地取走缓冲区中的消息，而且必须取走发来的每一个消息。刚开始时，缓冲区为空。试用P、V操作正确实现进程之间的同步。

设两个信号量s1和s2，s1用来制约发送进程，初始值为1；s2用来制约接收进程，初始值为0。

进程Ai 执行的过程为： 进程B执行的过程：

begin begin

准备消息 P(s2)

P(s1) 从缓冲区接收消息

消息写入缓冲区 V(s1)

V(s2) 释放消息

end end

3-8. 有一容量为100的循环缓冲区，有多个并发执行进程通过该缓冲区进行通信。为正确管理缓冲区，系统设置了两个读写指针，分别为OUT和IN。IN和OUT的值如何反应缓冲区为空还是满的情况？

缓冲区大小为100，因此当（IN + 1）% 100 = OUT 时，说明缓冲区已满。

当IN = OUT 时，缓冲区为空。

B5

B4

B3

B2

B1

OUT IN

3-9. 有一阅览室，共有100个座位。为了很好利用它，读者进入时必须先在登记表上进行登记，该表表目设有座位号和读者姓名，离开时再将其登记项擦除。试问：  
(1)  为描述读者的动作，应编写几个程序？应设几个进程？它们之间的关系是什么？  
(2) 试用P、V操作描述读者之间的同步算法。

假定 mutex：互斥信号量，用来制约个读者互斥地进行登记，其初始值为1；

empty：同步信号量，用来制约各读者能同时进入阅览室的数量，初始值为100.

登记过程： 擦除过程： 读者看成进程：

begin begin begin

p(empty) p(mutex) 调用登记过程

p(mutex) 擦除登记 进入阅览室

登记 v(mutex) 准备离开

v(mutex) v(empty) 调用擦除过程

end end end

3-14. 假定系统有N个进程共享M个同类资源，规定每个进程至少申请一个资源，每个进程的最大需求不超过M，所有进程的需求总和小于M+N。为什么在这种情况下也绝不会发生死锁？试证明。

假定会死锁，则根据死锁定义，N个进程之间相互等待，至少需要N个单位资源，又系统M个资源已分完，故所有进程需求总和大于或等于M+N，这与题中的所有进程需求总和小于M+N矛盾，故假设不成立。因此不会发生死锁。

3-15. 设有8个进程M1、M2、…、M8，它们有如图所示的优先关系，试用P、V操作实现这些进程间的同步。

M1： M2: M3: M4:

…… P(s12); P(s13); P(s14);

V(s12); …… …… ……

V(s13); V(s26); V(s36); V(s47);

V(s14); …… V(s38); ……  
----------------------------------------------------------------

M5: M6: M7: M8:

…… P(s26); P(s47); P(s38);

V(s57); P(s36); P(s57); P(s78);

…… …… …… ……

V(s78);

3-16. 设有5个哲学家，共享一张放有5把椅子的圆桌。每人分得1把椅子。但是，桌子上总共只有5把叉子。哲学家们在肚子饥饿时才试图分两次从两边捡起2把叉子就餐。条件：

a. 每人只有拿到2把叉子时，哲学家才能吃饭；  
b. 如果叉子已在他人手上，则该哲学家必须等到他人吃完之后才能拿到叉子。  
c. 任性的哲学家在自己未拿到2把叉子吃饭之前，绝不放下自己手中的叉子。  
试问：(1) 什么情况下5个哲学家全部吃不上饭？  
           (2) 描述一种避免有人饿死（永远拿不到2个叉子）的算法。

当5各哲学家各拿了1把叉子时，则全部吃不上饭。

假定一个信号量表示一把叉子，则存在5个信号量，

Int fork[0]=fork[1]=…=fork[4]=1

第i个哲学家吃饭过程：

do {

p(mutex);

p(fork[i]);

p(fork[(i+1)mod5]);

v(mutex);

吃饭;

v(fork[i]);

v(fork[(i+1)mod5]);

} while(1);

3-17. 在本章前面的读者与写者问题中，是用读者优先解决问题的。试分别用读者与写者公平竞争（无写者时，读者仍遵循多个读者可以同时读）、写者优先的算法解决这个问题。

**公平竞争：**

rmutex互斥共享、rwmutex读写互斥，读写进程在x上排队。

int rmutex = 1, rwmutex = 1, readcount = 0;

读者：

begin:

p(x);

p(rmutex);

if(readcount == 0) then p(rwmutex);

end if

++readcount;

v(rmutex);

v(x);

read data;

p(rmutex);

--readcount;

if(readcount == 0) then v(rwmutex);

end if;

v(rmutex);

…

end

写者：

begin:

p(x);

p(rwmutex);

write data;

v(rwmutex);

v(x);

…

end

**写者优先**：

int readcount, writecount;

semaphore rmutex = 1, wmutex = 1, rwmutex = 1, x = 1, z = 1;

读者：

While(1){

p(z);

p(x);

p(rmutex);

++readcount;

if(readcount == 1) then p(rwmutex);

end if

v(rmutex);

v(x);

v(z);

read data;

p(rmutex);

--readcount;

if(readcount == 0) then v(rwmutex);

end if;

v(rmutex);

}

写者：

while(1)

p(wmutex);

++writecount;

if(writecount == 1) then p(x);

end if;

v(wmutex);

p(rwmutex);

write data;

v(rwmutex);

p(wmutex);

--writecount;

if(writecount == 0) then v(x);

end if;

v(wmutex);

}

3-18. 有一个理发师、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，则理发师便坐在椅子上睡觉，但一个顾客到来时，必须唤醒理发师，请求理发；如果理发师正在理发时，又有顾客到来，只要有空椅子，他就坐下来等待，如果没有空椅子，他就离开。请为理发师和顾客各编写一段程序来描述他们的同步过程。

int s1 = 0, s2 = n;

顾客进程： 理发师进程：

begin begin

p(s2); p(s1);

v(s1); 给顾客理发;

坐等理发; v(s2);

end end

3-19. 假定系统中只有一类资源，进程一次只申请一个单位的资源，且进程申请的资源数不会超过系统拥有的资源总数。假定进程申请的资源总数为2，且系统资源总数如下，问下列哪些情况会发生死锁？

|  |  |  |
| --- | --- | --- |
| 序号 | 进程数 | 系统资源总数 |
| (1) | 1 | 2 |
| (2) | 2 | 2 |
| (3) | 2 | 3 |
| (4) | 3 | 3 |
| (5) | 3 | 5 |
| (6) | 4 | 5 |

(2) 和 (4) 会发生死锁

3-20. 系统有同类资源10个，进程P1、P2和P3需要该类资源的最大数量分别为8、6、7。它们使用资源的次序和数量如下表所示。

|  |  |  |
| --- | --- | --- |
| 次序 | 进程 | 申请量 |
| 1 | P1 | 3 |
| 2 | P2 | 2 |
| 3 | P3 | 4 |
| 4 | P1 | 2 |
| 5 | P2 | 2 |
| 6 | P1 | 3 |
| 7 | P3 | 3 |
| 8 | P2 | 2 |

(1)  试给出采用银行家算法分配资源时，进行第5次分配后各进程的状态及各进程占用资源情况。  
(2)  在以后的申请中，哪次的申请可以得到最先满足？给出一个进程完成序列。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P1/剩余 | P2/剩余 | P3/剩余 | 系统剩余 |
| 1 | 3/5 |  |  | 7 |
| 2 |  | 2/4 |  | 5 |
| 3 |  |  | 4（不安全） |  |
| 4 | (3+2)/3 |  |  | 3 |
| 5 |  | 2（不安全） |  |  |
| 6 | (3+2+3)/0 |  |  | 0（8） |
| 7 |  |  | 4/3 | 4 |
| 8 |  | (2+2)/2 |  | 2 |

1. P1占有5个资源，剩余3个资源请求，运行态。

P2占有2个资源，剩余4个资源请求，阻塞态。

P3占有0个资源，剩余7个资源请求，阻塞态。

系统剩余3个资源

1. P1的请求最先满足。进程完成序列：P1，P2，P3.

3-21. 考虑某一系统，它有4类资源R1、R2、R3、R4，有5个并发进程P0、P1、P2、P3、P4。请按照银行家算法回答下列问题：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *Allocation* | | | | *Max* | | | | *Available* | | | |
| *A* | *B* | *C* | *D* | *A* | *B* | *C* | *D* | *A* | *B* | *C* | *D* |
| *P0* | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | 1 | 5 | 0 | 2 |
| *P1* | 1 | 0 | 0 | 0 | 1 | 7 | 5 | 0 |
| *P2* | 1 | 3 | 5 | 4 | 2 | 3 | 5 | 6 |
| *P3* | 0 | 6 | 3 | 2 | 0 | 6 | 5 | 2 |
| *P4* | 0 | 0 | 1 | 4 | 0 | 6 | 5 | 6 |

(1)  各进程的最大资源请求和已分配的资源矩阵及系统当前的剩余资源向量如下图所示，计算各进程的剩余资源请求向量组成的矩阵。  
(2)  系统当前处于安全状态吗？  
当进程P2申请的资源分别为(1,0,0,1)时，系统能立即满足吗？

（1）

最大需求矩阵： 分配矩阵： 剩余请求矩阵：

剩余资源向量：（1 5 0 2）

（2）当前系统处于安全状态。

系统剩余资源向量为（1 5 0 2），p2剩余请求向量为（1 0 0 1），因（1 5 0 2）>（1 0 0 1），

所以当p2提出请求时，能满足。