# 第二章作业

班级： 07112002 学号： 1120200822 姓名： 郑子帆

## 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。

答：设置两个信号量s1, s2。

s1用来制约发送进程A1, ..., An，初值为1。

s2用来制约接收进程B，初值为0。

n个发送进程A1, ..., An和1个接收进程B的同步算法描述如下：

1. int s1=1, s2=0;
2. A1, ... , An:
3. begin
4. 准备消息
5. P(s1);
6. 将消息加入缓冲区
7. V(s2);
8. end
9. B:
10. begin
11. P(s2);
12. 从缓冲区取出消息
13. V(s1);
14. 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？

答：

当IN = OUT时，说明缓冲区已空；

当(IN+1)%100 = OUT时，说明缓冲区已满。

**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。

答：

1. 应编写3个程序，包含登记、擦除过程和读者进程调用登记、擦除过程的程序，我们将读者看成进程，进程数量由读者数量决定，所有读者在一个阅览室中，所以所有进程都运行一样的程序。
2. 设置两个同步信号量empty, mutex。

empty为同步信号量，用来制约同时在阅览室的读者数量，初值为100。

mutex为互斥信号量，用来制约读者互斥地修改登记表，初值为1。

同步算法描述如下：

1. empty=100, mutex=1;
2. 登记过程：
3. begin
4. p(empty), p(mutex)
5. 登记
6. v(mutex)
7. end
8. 擦除过程：
9. begin
10. p(mutex)
11. 找到之前自己登记的，擦除
12. v(mutex)
13. v(empty)
14. end
15. 进程：
16. begin
17. 调用登记过程
18. 找到座位阅读
19. 调用擦除过程
20. 离开阅览室
21. 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。

答：用反证法。

如果会发生死锁，那么将M个资源分配完后，根据死锁的定义，N个进程互相等待，且每个进行至少申请一个资源，那么每个进程至少还要申请1个资源以形成死锁。那么，此时N个进程共需申请M+N个资源，与题中“所有进程的需求总和小于M+N”矛盾。

故假设不成立，所以这种情况下不会发生死锁。

#### 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。

答：

设9个互斥信号量，设初值为0，按题目图中的前驱关系可以得到下面的P、V操作代码。

1. semaphore a12=a13=a14=a26=a36=a38=a47=a57=a78=0   *//初始化信号量*
2. M1(){
3. ...;
4. V(a12); V(a13); V(a14);
5. }
6. M2(){
7. P(a12); *// M1已经运行完成*
8. ...;
9. V(a26);
10. }
11. M3(){
12. P(a13); *// M1已经运行完成*
13. ...;
14. V(a36); V(a38);
15. }
16. M4(){
17. P(a14); *// M1已经运行完成*
18. ...;
19. V(a47);
20. }
21. M5(){
22. ...;
23. V(a57);
24. }
25. M6(){
26. P(a26); P(a36); *// M2、M3已经运行完成*
27. ...;
28. }
29. M7(){
30. P(a47); P(a57); *// M4、M5已经运行完成*
31. ...;
32. V(a78);
33. }
34. M8(){
35. P(a38); P(a78); *// M3、M7已经运行完成*
36. ...;
37. }

**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。

#### 答：

1. 当每个哲学家手上都只有1把叉子时，都吃不上饭。
2. 设6个信号量，5个信号量f[0]~f[4]表示5个叉子，mutex互斥信号表示当前是否有人在拿叉子。每个哲学家在拿叉子时不让其他人拿，并且自己一下拿两个叉子，算法如下：
3. f[0]=f[1]=f[2]=f[3]=f[4]=mutex=1;
4. 第 i 个哲学家的进程：
5. while(1){
6. P(mutex);
7. P(f[i]);
8. P(f[(i+1)mod5]);
9. V(mutex);
10. 吃饭
11. V(f[i]);
12. V(f[(i+1)mod5]);
13. }

#### 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。

答：

1. 公平竞争
2. readcount=0;
3. wmutex=1, remux=1, k=1;
5. reader( ): begin
6. P(k);
7. P(rmutex);
8. if(readcount == 0){
9. P(wmutex);
10. }
11. readcount++;
12. V(rmutex);
13. V(k);
14. 读
15. P(rmutex);
16. readcount--;
17. if(readcount == 0){
18. V(wmutex);
19. }
20. V(rmutex);
21. end
23. writer( ): begin
24. P(k);
25. P(wmutex);
26. 写
27. V(wmutex);
28. V(k);
29. end
30. 写者优先
31. startwr = 0, NowRW = 0, Rmutex = 0, Wmutex = 0;
32. readercount = 0, writercount = 0;
33. begin
34. P(rmutex);
35. P(NowRW);
36. readercount++;
37. if (readercount == 1){
38. P(startwr);
39. }
40. V(NowRW);
41. V(rmutex);
42. 读
43. P(rmutex);
44. readercount--;
45. if (readercount == 0){
46. V(startwr);
47. }
48. V(rmutex);
49. end
50. writer(): begin
51. P(wmutex);
52. writercount++;
53. if (writercount == 1){
54. P(NowRW);
55. }
56. V(NowRW);
57. V(wmutex);
58. P(startwr);
59. 写
60. V(startwr);
61. P(wmutex);
62. writercount--;
63. if (writercount == 0){
64. V(NowRW);
65. }
66. V(wmutex);
67. end

#### 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。

答：设同步信号量empty和互斥信号量mutex。

empty用来制约坐在椅子上的顾客数量，初值为n。

mutex用来制约理发师同时只能为一位顾客理发，初值为1。

同步算法如下：

1. empty=n, mutex=1
2. 顾客进程：
3. P(empty);
4. 坐椅子等理发
5. 理发
6. V(mutex);
7. 理发师进程：
8. P(mutex);
9. 理发
10. V(empty);

#### 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 |

答：(2)和(4)会发生死锁，因为在这两种情况下，每个进程依次申请到一个单位的资源，此时每个进程都只分配到了一个进程，再次轮到第一个进程申请时，则虚等待，而这也形成了所有进程相互等待的情况，形成了死锁。

而对于(1)(3)(5)(6)而言，在每个进程都分配到一个单位的资源后，此时仍有多余单位的资源待分配，所以可以有进程总共分配到2个单位的资源，完成后释放资源，资源再被分配给其他的进程，以此循环，所以不会形成死锁。

#### 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。

答：

1. 第三次分配时，P3不安全，收回。

第五次分配时，P2的2个资源不安全，收回。

故第五次分配后：

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最先满足，在第六次分配后得到满足。

进程完成序列为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？ (3)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？

#### 答：

1. 剩余请求矩阵Need=Max-Allocation=[]
2. 是安全状态。

P0先完成，释放资源，剩余资源向量为(1,5,1,4)

之后P2完成，释放资源，剩余资源向量为(2,8,6,8)

剩下进程P4 P3 P1均可完成，故是安全的。

1. 此时P2剩余请求向量为(1,0,0,2)<(1,5,0,2)，故能立即满足。