作业 7

贾城昊
2021K8009929010

7.1 某系统存在4个进程和5份可分配资源，当前的资源分配情况和最大需求如下表所示。求满足安全状态下X的最小值。请写出解题分析过程。

	Allocated	Maximum	Available
process A	5 4 2 5 3	5 5 2 5 5	0 x 1 0 0
process B	3 1 3 2 5	3 3 4 2 5	
process C	2 0 3 4 1	6 0 6 4 1	
process D	4 2 3 5 2	10 2 4 6 11	

解：

首先，我们可以写出 need matrix：

	Allocated	Maximum	Needs	Available
Process A	5 4 2 5 3	5 5 2 5 5	0 1 0 0 2	0 x 1 0 0
Process B	3 1 3 2 5	3 3 4 2 5	0 2 1 0 0	
Process C	2 0 3 4 1	6 0 6 4 1	4 0 3 0 0	
Process D	4 2 3 5 2	10 2 4 6 11	6 0 1 1 9	

- If x = 0:
  - 那么 0 0 1 0 0 资源可以分配，没有进程可以得到足够的资源能够进行。
  - 所以此时不是一个安全状态。

- If x = 1:
  - 那么 0 1 1 0 0 资源可以分配，没有进程可以得到足够的资源能够进行。
  - 所以此时不是一个安全状态。
- If x = 2:
  - 那么 0 2 1 0 0 资源可以分配，进程 B 可以得到足够的资源进行。
  - 进程 B 结束后有 3 3 4 2 5 资源可以分配 ， 进程 A 可以得到足够资源进行。
  - 进程 A 结束后有 8 7 6 7 8 资源可以分配 ， 进程 C 可以得到足够资源进行。
  - 进程 A 结束后有 10 7 9 11 9 资源可以分配 ， 进程 D 可以得到足够资源进行。
  - 所以此时是一个安全状态

所以满足安全状态下 X 的最小值为 2

7.2 两进程A和B各需要数据库中的3份记录1、2、3，若进程A以1、2、3的顺序请求这些资源， 进程B也以同样的顺序请求这些资源，则将不会产生死锁。但若进程B以3、2、1的顺序请求这些资源，则可能会产生死锁。这3份资源存在6种可能的请求顺序，其中哪些请求顺序能保证无死锁产生？请写出解题分析过程。

解：

如果 A 和 B 一开始请求相同的记录，只有其中一个会得到它。这意味着另一个将被阻塞，直到第一个完成。因此，在这种状态下不可能发生死锁。

如果 A 和 B 首先请求两个不同的记录，它们可能各自得到一个记录。这意味着它们都将被阻塞，因为它们都不能得到另一个得到的记录。因此，在这种状态下可能出现死锁。

在不丧失一般性的前提下，我们可以假设 A 请求的记录顺序为 1、2、3。那么 B 有 6 种可能的请求顺序:

1, 2, 3　　1, 3, 2　　2, 1, 3　　2, 3, 1　　3, 1, 2　　3, 2, 1

如前所述，只有两种组合保证不会死锁(1,2,3 和 1,3,2)。因此，只有 1/3 的组合保证不会死锁。

7.3 设有两个优先级相同的进程 T1，T2 如下。令信号量 S1，S2 的初值为 0，已知 z=2，试 问 T1，T2 并发运行结束后 x=?y=?z=?

线程 T1

```
y:=1;
y:=y+2;
V(S1);
z:=y+1;
P(S2);
y:=z+y;
```

线程 T2

```
x:=1;
x:=x+1;
P(S1);
x:=x+y;
V(S2);
z:=x+z;
```

注:请分析所有可能的情况，并给出结果与相应执行顺序。

解:

根据信号 S1 和 S2，两个进程的运行顺序有下列要求:

- P2 在第 3 行要等到 P1 执行完第 3 行后才能继续执行;
- P1 在第 5 行要等到 P2 执行完第 5 行后才能继续执行;
-

以 P1 的第 3 行执行和 P2 的第 5 行执行两个时间点，将进程执行过程分为 A、B、C 三段:

- P1 的第 1、2 行一定在 A 段执行，P2 的第 1、2 行可能在 A 或 B 段执行；
- P1 的第 4 行可能在 B 或 C 段执行，P2 的第 3、4 行一定在 B 段执行；
- P1 的第 5、6 行一定在 C 段执行，P2 的第 6 行一定在 C 段执行。

所以可能在 A 段执行的代码有 P1 的第 1、2 行，P2 的第 1、2 行；可能在 B 段执行的代码有 P1 的第 4 行，P2 的第 1、2、3、4 行；可能在 C 段执行的代码有 P1 的第 4 、5、6 行，P2 的第 6 行；

观察语句间的相关性：

- P1 的第 1、2 行与 P2 的第 1、2 行之间不相关，A 段代码执行顺序**不会**影响结果；
- P1 的第 4 行与 P2 的第 1、2、3、4 行之间不相关，B 段的代码执行顺序**不会**影响结果。
- P1 的第 4、6 行与 P2 的第 6 行之间存在相关性，C 段的代码执行顺序**会**影响结果。

完成 A、B 两段一定能够完成的指令后（P1 的 1、2、3 行，P2 的 1、2、3、4、5 行 ），有 x = 5、y = 3、z = 2。

P1 的第 4、6 行与 P2 的第 6 行，有 3 种完成顺序：

- P1 的第 4 行，P1 的第 6 行，P2 的第 6 行：x = 5、y = 7、z = 9；
- P1 的第 4 行，P2 的第 6 行，P1 的第 6 行：x = 5、y = 12、z = 9；
- P2 的第 6 行，P1 的第 4 行，P1 的第 6 行：x = 5、y = 7、z = 4。

综上，并发运行结束后，x、y、z 的值共有 3 种可能：

- x = 5、y = 7、z = 9；
- x = 5、y = 12、z = 9；
- x = 5、y = 7、z = 4。

对应的执行顺序上面以及给出。

7.4 在生产者-消费者问题中，假设缓冲区大小为 5，生产者和消费者在写入和读取数据时都会更新写入/读取的位置 offset。现有以下两种基于信号量的实现方法，

方法一

```
Class BoundedBuffer {
    mutex = new Semaphore(1);
    fullBuffers = new Semaphore(0);
    emptyBuffers = new Semaphore(n);
}
BoundedBuffer::Deposit(c) {
    emptyBuffers->P();
    mutex->P();
    Add c to the buffer;
    offset++;
    mutex->V();
    fullBuffers->V();
}
BoundedBuffer::Remove(c) {
    fullBuffers->P();
    mutex->P();
    Remove c from buffer;
    offset--;
    mutex->V();
    emptyBuffers->V();
}
```

方法二：

```
Class BoundedBuffer {
    mutex = new Semaphore(1);
    fullBuffers = new Semaphore(0);
    emptyBuffers = new Semaphore(n);
}
BoundedBuffer::Deposit(c) {
     mutex->P();
    emptyBuffers->P();
    Add c to the buffer;
    offset++;
    fullBuffers->V();
    mutex->V();
}
```

```
BoundedBuffer::Remove(c) {
    mutex->P();
    fullBuffers->P();
    Remove c from buffer;
    offset--;
    emptyBuffers->V();
    mutex->V();
}
```

请对比分析上述方法一和方法二，哪种方法能让生产者、消费者进程正常运行，并说明分析原因。

解：

方法一可以让生产者、消费者进程正常运行，方法二则可能会出现死锁。对于方法二，若缓冲区为空，Remove 申请到了锁，此时 Remove 不会继续执行，因为信号量 fullBuffers 为 0，同时 Deposit 也不会继续执行，因为信号量 mutex 也为 0，此时生产者和消费者进程都不能正常运行。

而方法一 Deposite 只有当 emptyBuffers 不为 0 才能申请到 mutex，Remove 也只有在 fullBuffers 不为 0 才能申请到 mutex，所以申请到 mutex 的进程必定能运行并释放资源，所以可以正常运行。