**2016011974 刘喆**

**操作系统第一次课后练习**

**练习一：区分“死锁”、“饿死”**

**要求：**

**1. 分别给出 “死锁”、“饿死”的定义，并标明出处；**

**2. 通俗说明 “死锁“、”饿死“的区别。**

1.（1）死锁：

In concurrent computing, a **deadlock** is a state in which each member of a group is waiting for some other member to take action, such as sending a message or more commonly releasing a lock. (from Wikipedia)

**死锁**是指两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。（来自百度百科）

（2）饿死：

In computer science, **starvation** is a problem encountered in concurrent computing where a process is perpetually denied necessary resources to process its work. **Starvation** may be caused by errors in a scheduling or mutual exclusion algorithm, but can also be caused by resource leaks, and can be intentionally caused via a denial-of-service attack such as a fork bomb. (from Wikipedia)

进程饥饿指当等待时间给进程推进和响应带来明显影响，当饥饿到一定程度的进程在等待到即使完成也无实际意义的时候称为**饿死**。（来自百度百科）

2. （1）通俗来说，死锁就是每个进程都捧着手上已经有的资源不放手，又想拿别的进程手上的资源，从而导致所有进程僵持不下，系统头很大不知道要怎么协调的情况；而饿死就是某进程干巴巴等一个资源但是这个资源就是轮不到它吃，时间一长，他就饿死掉去了的情况。

（2）死锁和饿死的不同点：

①死锁进程等待别的进程永远不会松手的资源，饿死进程等待会被别的进程释放但却不会分配给自己的资源。

③死锁一定是你等我我等他他等你，一堆进程循环等待，而饿死则不然，我饿死了可能跟别的进程都没什么关系。

**练习二：**

**有三类资源A（17）、B（5）、C（20）。有5个进程P1-P5。**

**T0时刻系统状态如下：**

|  |  |  |
| --- | --- | --- |
|  | **最大需求** | **已分配** |
| **P1** | **5 5 9** | **2 1 2** |
| **P2** | **5 3 6** | **4 0 2** |
| **P3** | **4 0 11** | **4 0 5** |
| **P4** | **4 2 5** | **2 0 4** |
| **P5** | **4 2 4** | **3 1 4** |

**（1）T0时刻是否为安全状态，给出安全序列；**

**（2）T0时刻，P2：Request(0,3,4)，能否分配，为什么？**

**（3）在（2）的基础上，P4：Request(2,0,1)，能否分配，为什么？**

**（4）在（3）的基础上，P1：Request(0,2,0)，能否分配，为什么？**

（1）T0时刻的安全性：利用安全性算法对T0时刻的资源分配情况进行分析（如下图所示）可知，在T0时刻存在着一个安全序列 {P4, P2, P3, P5, P1}，故系统是安全的。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 资源  进程 | Work | Need | Allocation | Work+Allocation | Finish |
| A B C | A B C | A B C | A B C |
| P4 | 2 3 3 | 2 2 1 | 2 0 4 | 4 3 7 | true |
| P2 | 4 3 7 | 1 3 4 | 4 0 2 | 8 3 9 | true |
| P3 | 8 3 9 | 0 0 6 | 4 0 5 | 12 3 14 | true |
| P5 | 12 3 14 | 1 1 0 | 3 1 4 | 15 4 18 | true |
| P1 | 15 4 18 | 3 4 7 | 2 1 2 | 17 5 20 | true |

（2）P2请求资源：P2发出请求向量Request(0,3,4)，系统按银行家算法进行检查：

①Request(0,3,4)≤Need(1,3,4)；

②Request(0,3,4)＞Available(2,3,3)，让P2等待。

（3）P4请求资源：P4发出请求向量Request(2,0,1)，系统按银行家算法进行检查：

①Request(2,0,1)≤Need(2,2,1)；

②Request(2,0,1)≤Available(2,3,3)；

③系统先假定可为P4分配资源，并修改Available，Allocation和Need向量，由此形成的资源变化情况如下表括号中所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 资源  进程 | Max | Allocation | Need | Available |
| A B C | A B C | A B C | A B C |
| P1 | 5 5 9 | 2 1 2 | 3 4 7 | 2 3 3  (0 3 2) |
| P2 | 5 3 6 | 4 0 2 | 1 3 4 |  |
| P3 | 4 0 11 | 4 0 5 | 0 0 6 |  |
| P4 | 4 2 5 | 2 0 4  (4 0 5) | 2 2 1  (0 2 0) |  |
| P5 | 4 2 4 | 3 1 4 | 1 1 0 |  |

④再利用安全性算法检查此时系统是否安全，如下表所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 资源  进程 | Work | Need | Allocation | Work+Allocation | Finish |
| A B C | A B C | A B C | A B C |
| P4 | 0 3 2 | 0 2 0 | 4 0 5 | 4 3 7 | true |
| P2 | 4 3 7 | 1 3 4 | 4 0 2 | 8 3 9 | true |
| P3 | 8 3 9 | 0 0 6 | 4 0 5 | 12 3 14 | true |
| P5 | 12 3 14 | 1 1 0 | 3 1 4 | 15 4 18 | true |
| P1 | 15 4 18 | 3 4 7 | 2 1 2 | 17 5 20 | true |

由所进行的安全性检查得知，可以找到一个安全序列 {P4, P2, P3, P5, P1}。因此，系统是安全的，可以立即将P4所申请的资源分配给它。

（4）P1请求资源：在（3）的基础上，P4已顺利获取所申请的资源，由此形成的资源变化情况如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 资源  进程 | Max | Allocation | Need | Available |
| A B C | A B C | A B C | A B C |
| P1 | 5 5 9 | 2 1 2 | 3 4 7 | 0 3 2 |
| P2 | 5 3 6 | 4 0 2 | 1 3 4 |  |
| P3 | 4 0 11 | 4 0 5 | 0 0 6 |  |
| P4 | 4 2 5 | 4 0 5 | 0 2 0 |  |
| P5 | 4 2 4 | 3 1 4 | 1 1 0 |  |

在（2）的基础上，考虑P2发出的请求向量Request(0,3,4)，系统按银行家算法进行检查：

①Request(0,3,4)≤Need(1,3,4)；

②Request(0,3,4)＞Available(0,3,2)，让P2继续等待；

继而考虑P1发出请求的向量Request(0,2,0)，系统按银行家算法进行检查：

①Request(0,2,0)≤Need(3,4,7)；

②Request(0,2,0)≤Available(0,3,2)；

③系统先假定可为P1分配资源，并修改Available，Allocation和Need向量，由此形成的资源变化情况如下表括号中所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 资源  进程 | Max | Allocation | Need | Available |
| A B C | A B C | A B C | A B C |
| P1 | 5 5 9 | 2 1 2  （2 3 2） | 3 4 7  （3 2 7） | 0 3 2  （0 1 2） |
| P2 | 5 3 6 | 4 0 2 | 1 3 4 |  |
| P3 | 4 0 11 | 4 0 5 | 0 0 6 |  |
| P4 | 4 2 5 | 4 0 5 | 0 2 0 |  |
| P5 | 4 2 4 | 3 1 4 | 1 1 0 |  |

④此时可用资源Available(0,1,2)已不能满足任何进程的需要，故系统进入不安全状态，此时系统不分配资源。