1. 死锁和饥饿的定义，用自己的语言阐释。

死锁的定义：

在一组进程发生死锁的情况下，这组死锁进程中的每一个进程，都在等等待另一个死锁进程所占有的资源。或者说每个进程等待的时间是该组中其他进程释放所占有的资源。但由于所有这些进程都无法运行，因此它们谁也不释放资源，致使没有任何一个进程可被唤醒。这样这组进程只能无期限地等待下去。如果一组进程中的每一个进程都在等待仅有该组进程中的其它进程才能引发的事件，那么这组进程是死锁的。（教材P107）

饥饿的定义：

饥饿是指系统不能保证某个进程的等待时间上界，从而使该进程长时间等待，当等待时间给进程推进和响应带来明显影响时，称发生了进程饥饿。当饥饿到一定程度的进程所赋予的任务即使完成也不再具有实际意义时称该进程被饿死。（来源<https://blog.csdn.net/qq_15037231/article/details/56842836>）

个人理解：

死锁是一组相互占用所需要资源的进程，大家都占有一部分资源不肯放手，同时在等待其他进程释放自己所需要的另一部分资源，这个状态僵持着，所以这组进程中没有进程可以顺利执行，所有进程都无限制地等待着。

饥饿也是进程无法顺利执行，但是不需要一定是一组相互占用着所需资源的进程，可以是一个进程，饥饿的原因是长时间无上限的等待，等待资源调度。

1. 银行家算法练习。



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Max | Allocation | Need | Available |
| P1 | 5 5 9 | 2 1 2 | 3 4 7 | 2 3 3 |
| 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 | 2 2 1 |
| P5 | 4 2 4 | 3 1 4 | 1 1 0 |

1. T0时刻的安全性：利用安全性算法对T0时刻资源分配进行分析可知，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。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Work | Need | Allocation | Work+Allocation | Finish |
| 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（0，3，4），系统按银行家算法进行检查：

Request（0，3，4）< Need（1，3，4）

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

（3）P4请求资源（2，0，1），系统按银行家算法进行检查：

Request（2，0，1）< Need（2，2，1）

Request（2，0，1）< Available（2，3，3）

系统先假定可为P4分配资源，修改P4的Max，Allocation值和Available值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Max | Allocation | Need | Available |
| 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 |

利用安全性算法检查此时系统是否安全

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Work | Need | Allocation | Work+Allocation | Finish |
| 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 |

由安全性检查知，可以找到一个安全序列{P4,P2,P3,P5,P1}，故系统是安全的，可以立即将P4所申请的资源分配给它。

（4）P1请求资源（0，2，0），系统按银行家算法进行检查：

Request（0，2，0）< Need（3，4，7）

Request（0，2，0）> Available（0，3，2），不能分配，让P1等待。