操作系统第一次作业

1. 找出死锁和饥饿的定义和区别（通俗解释）。

死锁：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。(百度百科，该词条由[“科普中国”百科科学词条编写与应用工作项目](http://www.cast.org.cn/)审核)

通俗讲就好比几个人进厕所，位置少于人数，占到位置的人不想上，想上的人没有占到位置（有没有位置是随机分配），陷入僵局，就是死锁。

饥饿：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，称为发生了进程饥饿。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。（百度知道）

通俗讲就好比一个人很饿，在排队买饭（站在队伍第一个），中途总有人来插队，插到最前面，一直都轮不到第一个人，很久之后他就饿死了。

相同点：两者都是由于竞争资源引起的。

不同点：

（1）从[进程状态](https://www.baidu.com/s?wd=%E8%BF%9B%E7%A8%8B%E7%8A%B6%E6%80%81&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4rjnvnyc3m1bYPH9BujD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWDsPj6dPjmY)考虑，死锁进程都处于等待状态，忙等待(处于运行或就绪状态)的进程并非处于等待状态，但却可能被饿死；

（2）死锁进程等待永远不会被释放的资源，饿死进程等待会被释放但却不会分配给自己的资源，表现为等待时限没有上界(排队等待或忙式等待)；

（3）死锁一定发生了循环等待，而饿死则不然。这也表明通过资源分配图可以检测死锁存在与否，但却不能检测是否有进程饿死；

（4）死锁一定涉及多个进程，而饥饿或被饿死的进程可能只有一个。

（5）在饥饿的情形下，系统中有至少一个进程能正常运行，只是饥饿进程得不到执行机会。而死锁则可能会最终使整个系统陷入死锁并崩溃。（百度知道）

通俗讲好比，死锁是大家一起等都没有进展，饥饿是在进展但是一直排不到它。  
2. (1)是安全的，安全序列是{P4，P2，P3，P5，P1}

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

可以找到一个安全序列 {P4，P2，P3，P5，P1}，故系统是安全的

(2) T0时刻 P2：Request（0，3，4）≤Need2（1，3，4）；

P2：Request（0，3，4）> Available（2，3，3），所以让P2等待。

(3)可以分配

P4：Request（2，0，1）≤Need4（2，2，1）；P4：Request（2，0，1）≤Available（2，3，3）；

设为P4分配资源，有关数据则变为： Allocation4（4，0，5）， Need4(0，2，0)，Available（0，3，2），进行安全性算法检查

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

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)不分配

Allocation4（4，0，5）， Need4(0，2，0)，Available（0，3，2）；

P1：Request（0，2，0）≤Need1（3，4，7）；P1：Request（0，2，0）≤Available（0，3，2）；

设为P1分配资源，有关数据则变为：

Allocation1（2，3，2）， Need1(3，2，7)，Available（0，1，2），进行安全性算法检查

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

由上图，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给P1。