操作系统第七次作业

1

> 在有m个进程的系统中出现死锁时，死锁进程的个数k应该满足的条件是什么?

k需要满足的条件:$2 \leq k \leq m$

原因:

1. 出现死锁至少需要两个进程,因此$k \geq 2$
2. 进程数不会超过操作系统的进程数,因此$k \leq m$

2

> 什么是死锁？产生死锁的原因是什么?

死锁的定义:
多个进程因竞争资源而造成一种僵局,若无外力作用,这些进程都将无法向前推进

原因:

1. 系统资源的竞争
2. 进程推进顺序非法

3

> 假设一个系统有m个相同类型的资源被n个进程共享，进程每次只请求或释放一个资源。试证明只要符合下面两个条件，系统就不会发生死锁。
>
> - 每个进程需要资源的最大值在$1 \sim m$之间。
> - 所有进程需要资源的最大值的和小于$m+n$。

证明过程:

条件1:每个进程需要资源的最大值在$1 \sim m$之间。

条件2:所有进程需要资源的最大值的和小于$m+n$。

结论:系统发生死锁

反证法:假设条件1,2均满足,但是系统发生了死锁

设第i个进程所需要的资源数量为$Resource(i)$, 记为R_i
根据条件1, 我们可以得知$1 \leq R_i \leq m$
根据条件2, 我们可以得知$\Sigma R_i < m + n$
我们可得$n \leq \Sigma R_i < m + n$
$0 \leq \Sigma R_i - m < n$
意味着$\forall i, R_i \geq 1$, 此时剩下的资源数$\Sigma R_i - m$仍然小于进程数n
所以表明系统中必然至少有一个资源是空闲的，可以被某个等待资源的进程使用
由于$\forall i$, i运行的条件为$m - (\Sigma R - i - n) > 0$, 即$\Sigma R_i - m < n$
而且上面我们已经证明了$\forall i, \Sigma R_i - m < n$
因此至少有一个进程会运行, 因此系统未发生死锁.
综上, 满足条件1, 条件2, 结论是不可能的, 因此在满足条件1和条件2时, 系统一定不会发生死锁
$Q.E.D$

4

> 考虑一个系统在某一时刻的状态：
>
> $$Allocation = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 3 & 5 & 4 \\ 0 & 6 & 3 & 2 \\ 0 & 0 & 1 & 4 \end{pmatrix}$$
> $$Max = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 7 & 5 & 0 \\ 2 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \\ 0 & 6 & 5 & 6 \end{pmatrix} \tag{1}$$
> $$Avilable = \begin{pmatrix} 1 & 5 & 2 & 0 \end{pmatrix}$$
>
> 使用银行家算法回答下列问题：
>
> 1. Need 矩阵的内容是怎样的?
> 2. 系统是否处于安全状态?
> 3. 如果从进程 P1发来一个请求（0, 4, 2, 0），这个请求能否立刻被满足?

1. Need 矩阵的内容

$$Allocation = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 3 & 5 & 4 \\ 0 & 6 & 3 & 2 \\ 0 & 0 & 1 & 4 \end{pmatrix}$$

$$Max = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 7 & 5 & 0 \\ 2 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \\ 0 & 6 & 5 & 6 \end{pmatrix}$$

$$Avilable = (1 \quad 5 \quad 2 \quad 0)$$

$$Need = Max - Allocation = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 7 & 5 & 0 \\ 2 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \\ 0 & 6 & 5 & 6 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 3 & 5 & 4 \\ 0 & 6 & 3 & 2 \\ 0 & 0 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 \\ 0 & 6 & 4 & 2 \end{pmatrix}$$

2. 是否安全

初始$Available = (1 \quad 5 \quad 2 \quad 0)$

需求矩阵$Need = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 \\ 0 & 6 & 4 & 2 \end{pmatrix}$

安全序列生成步骤：

1. $work = (1 \quad 5 \quad 2 \quad 0)$, $\text{work} \geq \text{Need}[0]$, 将$P_0$放入安全队列
更新$work$为$(1 \quad 5 \quad 3 \quad 2)$（加上$P_0$释放的资源）

2. $work = (1 \quad 5 \quad 3 \quad 2)$, $\text{work} \geq \text{Need}[2]$, 将$P_2$放入安全队列
更新$work$为$(2 \quad 8 \quad 8 \quad 6)$（加上$P_2$释放的资源）

3. $work = (2 \quad 8 \quad 8 \quad 6)$, $\text{work} \geq \text{Need}[1]$, 将$P_1$放入安全队列
更新$work$为$(3 \quad 8 \quad 8 \quad 6)$（加上$P_1$释放的资源）

4. $work = (3 \quad 8 \quad 8 \quad 6)$, $\text{work} \geq \text{Need}[3]$, 将$P_3$放入安全队列
更新$work$为$(3 \quad 14 \quad 11 \quad 8)$（加上$P_3$释放的资源）

5. $work = (3 \quad 14 \quad 11 \quad 8)$, $\text{work} \geq \text{Need}[4]$, 将$P_4$放入安全队列
更新$work$为$(3 \quad 14 \quad 12 \quad 12)$（加上$P_4$释放的资源）

3.

$Requset_1 = (0 \quad 4 \quad 2 \quad 0)$
$Need_1 = (0 \quad 7 \quad 5 \quad 0)$
$Available_1 = (1 \quad 5 \quad 2 \quad 0)$
检查条件 1: 请求 $\leq$ 需求
即 $(0 \quad 4 \quad 2 \quad 0) \leq (0 \quad 7 \quad 5 \quad 0)$
条件 1 满足。
检查条件 2: 请求 $\leq$ 当前可用资源
即 $(0 \quad 4 \quad 2 \quad 0) \leq (1 \quad 5 \quad 2 \quad 0)$
条件 2 也满足。
因此，进程$P1$的请求可以立即被满足。