河北师范大学软件学院
Software College of Hebei Normal University

操作系统实践

实验04 避免死锁（1）

软件学院 基础组

本节知识点

1 实验目的
2 实验准备
3 实验内容
4 问题思考

本节知识点

1 实验目的
2 实验准备
3 实验内容
4 问题思考

1 实验目的

1、理解进程产生死锁原因，了解为什么要进行死锁的避免

2、掌握银行家死锁避免算法过程与实现

3、加深对算法理论到算法实现的思想的认识

本节知识点

1 实验目的

2 实验准备

3 实验内容

4 问题思考

2 实验准备

2.1 产生死锁的原因

- 竞争资源。当系统中供多个进程共享的资源（不可剥夺性资源）如打印机、公用队列等，其数目不足以满足诸进程的需要时，会引起诸进程对资源的竞争而产生死锁。
- 进程间推进顺序非法。进程在运行过程中，请求和释放资源的顺序不当，也同样会导致产生死锁。

2 实验准备

2.2 产生死锁的四个必要条件

- ①互斥条件：指进程对所分配到的资源进行排他性使用，即在一段时间内某资源只由一个进程占用。如果此时还有其他进程请求该资源，则请求着只能等待，直至占有该资源的进程用毕释放。
- ②请求和保持条件：指进程已经保持了至少一个资源，但又提出了新的资源请求（该资源又被其他进程占有，此时请求进程阻塞，但又对自己已获得的其他资源保持不放）。

2 实验准备

- ③不剥夺条件：指进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。
- ④环路等待条件：指在发生死锁时，必然存在一个进程——资源的环形链。即进程集合{P_0,P_1,P_2,...,P_n}中的P_0正在等待一个P_1占用的资源；P_1正在等待一个P_2占用的资源，……，P_n正在等待一个已被P_0占用的资源。

2 实验准备

2.3 处理死锁的基本方法

①预防死锁

②避免死锁

③检测死锁

④解除死锁

2 实验准备

2.4 避免死锁

该方法同样是属于事先预防的策略，但它并不须事先采取各种限制措施去破坏产生死锁的四个必要条件，而是在资源的动态分配过程中，用某种方法去防止系统进入不安全状态，从而避免发生死锁。这种方法只须事先加以较弱的限制条件，便可获得较高的资源利用率及系统吞吐量，但在实现上有一定的难度。目前在较完善的系统中，常用此方法来避免发生死锁。

2 实验准备

2.5 安全状态

安全状态，是指系统能按某种进程顺序（P_0, P_1, P_2, …, P_n）来为每个进程P_i分配其所需资源，直至满足每个进程对资源的最大需求，使每个进程都可顺利地完成。如果系统无法找到这样一个安全序列，则称系统处于不安全状态。

2 实验准备

2.6 存储形式

- 结构体，数组
- PCB中要包含资源信息：需要每项资源个数、已占用每项资源个数、还需要每项资源个数

2 实验准备

```
typedef struct{
    int MaxNum[RESOURCE_NUM];
    //需要每项资源个数
    int AllocationNum[RESOURCE_NUM];
    //已占用每项资源个数
    int NeedNum[RESOURCE_NUM];
    //还需要的每项资源个数
}ResourceList;
```

2 实验准备

```
typedef struct
{
        char Name[NAME_MAXSIZE];  //进程名
        ResourceList resList;         //资源清单
}PCB;
```

本节知识点

1 实验目的

2 实验准备

3 实验内容

4 问题思考

3 实验内容

3.1 实验思路

先对用户提出的请求进行合法性检查，即检查请求是否大于需要的，是否大于可利用的。若请求合法，则进行预分配，对分配后的状态调用安全性算法进行检查。若安全，则分配；若不安全，则拒绝申请，恢复到原来的状态，拒绝申请。

3 实验内容

3.2 银行家算法步骤

设$Request_i$是进程P_i的请求向量。$Request_i[j] = K$，表示P_i需K个R_j类型的资源。当P_i发出资源请求后，系统按下述步骤进行检查：

- 1. 如果$Request_i[j] <= Need[i,j]$，便转向步骤2；否则认为出错，因为它所需要的资源数已超过它所宣布的最大值。
- 2. 如果$Request_i[j] <= Available[j]$，便转向步骤3；否则，表示尚无足够资源，$P_i$需等待。

3 实验内容

- 3. 系统试探着把资源分配给进程Pi，并修改下面数据结构中的数值：

$$Available[j] = Available[j] - Request_i[j];$$

$$Allocation[i,j] = Allocation[i,j] + Request_i[j];$$

$$Need[i,j] = Need[i,j] - Request_i[j];$$

- 4. 系统执行安全性算法，检查此次资源分配后系统是否出于安全状态以决定是否完成本次分配。

3 实验内容

➢5. 若安全，则分配资源；

否则，不分配资源

Available[j] = Available[j] + $Request_i$[j];

Allocation[i,j] = Allocation[i,j] - $Request_i$[j];

Need[i,j] = Need[i,j] + $Request_i$[j];

P_i需等待。

3 实验内容

3.2 安全性算法

- 1. 设置两个向量：工作向量work：表示系统可提供给进程继续运行所需的各类资源数目，含有m个元素的一维数组，初始时，work = Available;

  Finish : 含n个元素的一维数组，表示系统是否有足够的资源分配给n个进程，使之运行完成。开始时先令Finish[i] = false (i=1..n); 当有足够资源分配给进程i时，再令Finish[i] = true。

3 实验内容

- 2. 从进程集合中找到一个能满足下述条件的进程：

  Finish[i] = false; Need[i,j] <= work[j]; 若找到，执行步骤3，否则执行步骤4。

- 3. 当进程P_i获得资源后，可顺利执行，直至完成，并释放出分配给它的资源，故应执行：

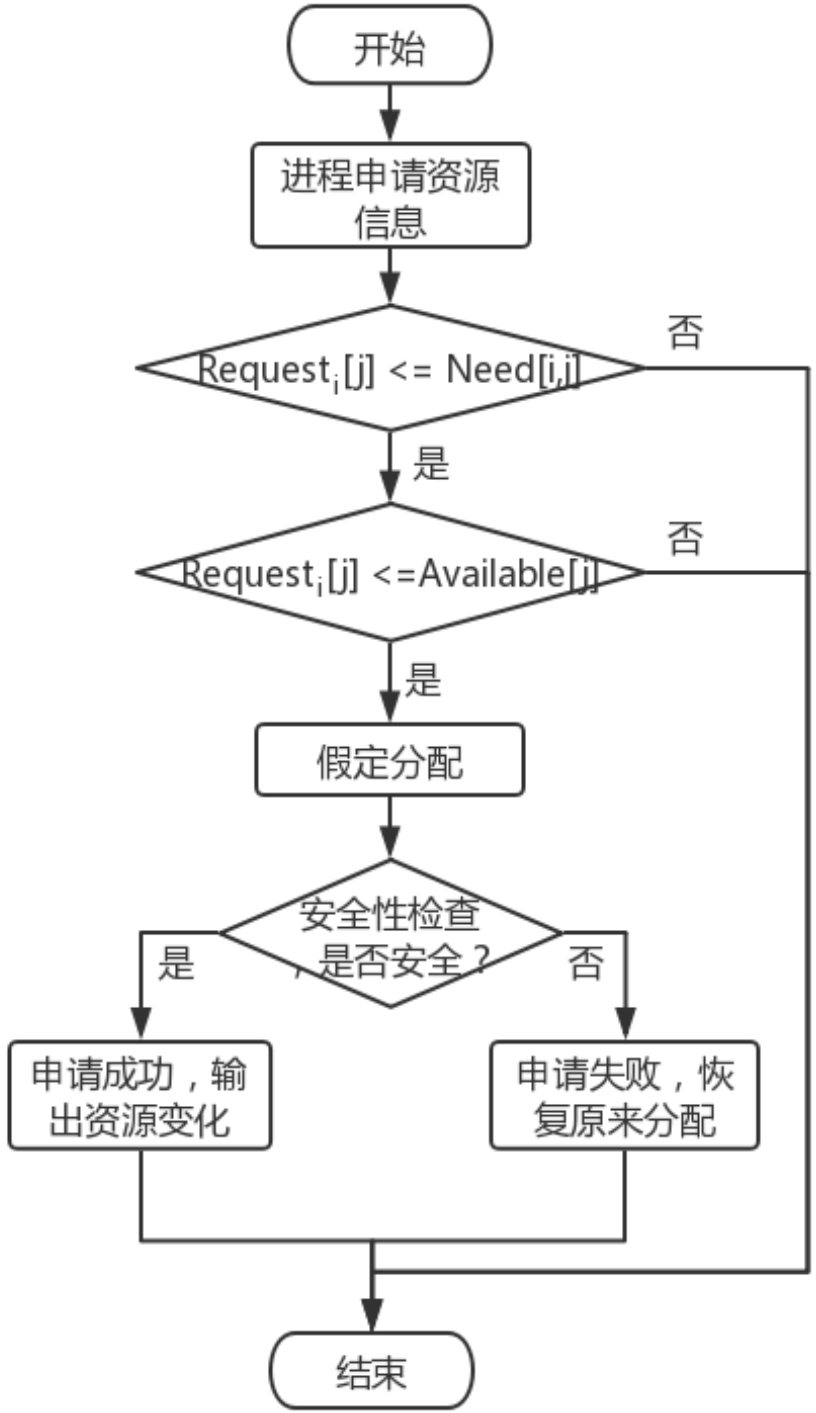
  work[j] = work[i] + Allocation[i,j] ;

  Finish[i] = true;

  goto step 2;

3 实验内容

- 4. 如果所有进程的Finish[i] = true都满足，则表示系统处于安全状态；否则，系统处于不安全状态。


开始
进程申请资源信息
Request$_i$[j] <= Need[i,j]
否
是
Request$_i$[j] <=Available[j]
否
是
假定分配
安全性检查
是否安全？
是
否
申请成功，输出资源变化
申请失败，恢复原来分配
结束

本节知识点

1 实验目的
2 实验准备
3 实验内容
4 问题思考

4 问题思考

1、进程资源信息事先全部安排好，并不是运行态的中间结果。（作业）

2、进程资源信息不涉及界面交互。

谢谢观赏！