Homework 4

3150103823 韩熠星

1. 假设系统由相同类型的9个资源被4个进程共享，试分析每个进程最多可以请求多少个资源数时该系统仍无死锁?

A. 1　B. 2　C.3　D.4

C

2. 死锁现象并不是计算机系统独有的，例如，除_____之外，下列三种案例都是死锁的体现。
A. 大桥大修，桥上只有一个车道通行
B. 高速公路大堵车，因为桥被台风吹跨了
C. 两列相向行使的列车在单轨铁路线上迎面相遇
D. 两位木匠订地板，一位只握一把斧头，而另一位没有榔头，却有钉子

B

3. 在下列描述中，发生进程通信上的死锁
A. 某一时刻，发来的消息传给进程P1，进程P1传给进程P2，进程P2得到的消息传给进程P3，则P1、P2、P3三进程。
B. 某一时刻，进程P1等待P2发来的消息，进程P2等待P3发来的消息，而进程P3又等待进程P1发来的消息，消息未到，则P1、P2、P3三进程。
C. 某一时刻，发来的消息传给进程P3，进程P3再传给进程P2，进程P2再传给进程P1，则P1、P2、P3三进程。
D. 某一时刻，发来的消息传给进程P2，进程P2再传给进程P3，进程P3再传给进程P1，则P1、P2、P3三进程。

B

4. 在哲学家进餐问题中，若仅提供5把叉子，则同时要求进餐的人数最多不超过时，一定不会发生死锁。
A．2　B. 3　C. 4　D. 5

C

5. 以下叙述中正确的是____。
A.进程调度程序主要是按一定算法从阻塞队列中选择一个进程，将处理机分配给它
B.预防死锁的发生可以通过破坏产生死锁的4个必要条件之一来实现，但破坏互斥条件的可能性不大
C.进程进入临界区时要执行开锁原语
D. P、V操作可以防止死锁的发生

B

6. 在下面哪种情况下，系统出现死锁。
A．计算机系统发生了重大故障
B．有多个阻塞的进程正在等待键盘的输入
C．若干进程因竞争资源而无休止地相互等待他方释放已占有的资源
D．资源数大大小于进程数或进程同时申请的资源数大大超过资源总数

C

7. 预防死锁是通过破坏死锁四个必要条件中的任何一个来实现的，下面关于预防死锁的说法中，错误的是___。
A．破坏“非抢占”条件目前只适用于内存和处理器资源
B．可以采用共享等策略来破坏“互斥”条件
C．破坏“请求和保持”条件可以采用静态分配策略或规定进程申请新的资源前首先释放已经占用的资源
D．采用资源编号并规定进程访问多个资源时按编号次序顺序申请的办法可以破坏“环路等待”条件，从而防止死锁的出现

A

8. 设m为同类资源数，n为系统中并发进程数。当n个进程共享m个互斥资源时，每个进程的最大需求是w；则下列情况会出现系统死锁的是。

A.m=2,n=1,w=2　　B.m=2,n=2,w=1　　C.m=4,n=3,w=2　　D.m=4,n=2,w=3

D

9. 使用银行家算法来避免死锁的操作系统是____。

A. Windows XP　　B. Linux　　C. FreeBSD UNIX　　D. A、B、C都不是

D

> 银行家算法（Banker's Algorithm）是一个避免死锁（Deadlock）的著名算法，是由艾兹格·迪杰斯特拉在1965年为**T.H.E系统**设计的一种避免死锁产生的算法。它以银行借贷系统的分配策略为基础，判断并保证系统的安全运行。

10. 死锁与安全状态的关系是____。

A. 死锁状态有可能是安全状态　　B. 安全状态也可能是死锁状态

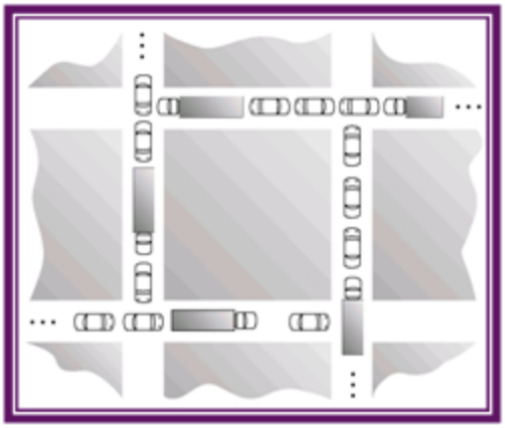
C. 不安全状态必定产生死锁　　D. 死锁状态一定是不安全状态

D

11. 假设有如下图所示的交通死锁情况：

(1) 说明产生死锁的4个必要条件在此处成立。

(2) 给出一个避免死锁的简单规则。



（1）在此处，产生死锁的四个必要条件为：

1.互斥：每个车道的每段道路只能被一辆车占用。

2.请求与保持：每个车队占用了一个车道，并请求前方的车道，即使需等待前方车道上的车队驶离，它仍将持有已占用的车道。

3.不抢占：在前方的车道被其它车队占用时，因为是单车道，而其它车队又不会后退，所以无法从其它车队处抢占车道。

4.环路等待：向东行驶的车队等待向北行驶的车队让出车道，向北行驶的车队等待向西行驶的车队让出车道，向西行驶的车队等待向南行驶的车队让出车道，而向南行驶的车队则等待向东行驶的车队让出车道。**故存在一循环等待链。**

（2）增加约束条件：只有前方两个路口都空闲时，才能占用第一个路口。即可避免死锁的产生。

12. 设有一系统在某时刻的资源分配情况如下：

进程号	已分配资源	最大请求资源	剩余资源
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

请问：

(1) 系统中各进程尚需资源数各是多少?

(2) 当前系统安全吗?

(3) 如果此时进程P1提出资源请求（0，4，2，0），系统能分配给它吗?

(1) 各进程尚需资源数 = 最大请求资源 - 已分配资源。结果如下表所示：

	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

(2) **系统是安全的** 可以找到一个安全序列：<P0, P2, P3, P4, P1>

(3) 分配资源给P1后，剩余资源为（1, 1, 0, 0）。可以找到一个安全序列<P0, P2, P3, P4, P1>。所以系统是安全的

7.1 Consider the traffic deadlock depicted in Figure 7.1.

a. Show that the four necessary conditions for deadlock indeed hold in this example.

b. State a simple rule for avoiding deadlocks in this system.

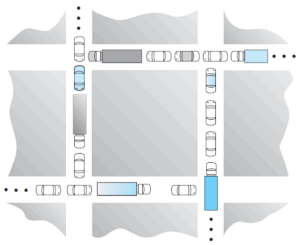


Figure 7.1 Traffic deadlock for Exercise 7.1.

Same as question 11

7.2 Consider the deadlock situation that could occur in the dining-philosophers problem when the philosophers obtain the chopsticks one at a time. Discuss how the four necessary conditions for deadlock indeed hold in this setting. Discuss how deadlocks could be avoided by eliminating any one of the four conditions.

Deadlock is possible because the four necessary conditions hold in the following manner:

1. Mutual exclusion is required for chopsticks
2. The philosophers tend to hold onto the chopstick in hand while they wait for the other chopstick
3. There is no preemption of chopsticks in the sense that a chopstick allocated to a philosopher cannot be forcibly taken away
4. There is a possibility of circular wait.

Deadlocks could be avoided by overcoming the conditions in the following manner:

1. Allow simultaneous sharing of chopsticks
2. Have the philosophers relinquish the first chopstick if they are unable to obtain the other chopstick,
3. Allow for chopsticks to be forcibly taken away if a philosopher has had a chopstick for a long period of time
4. Enforce a numbering of the chopsticks and always obtain the lower numbered chopstick before obtaining the higher numbered one.

7.6 Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock-free.

Suppose the system is deadlocked.

Each process is holding one resource and is waiting for one more. But there are three processes and four resources, one process must be able to obtain two resources. This process requires no more resources and, therefore it will return its resources when done.

So the system is deadlock-free.

7.7 Consider a system consisting of m resources of the same type, being shared by n processes. Resources can be requested and released by processes only one at a time. Show that the system is deadlock free if the following two conditions hold:

a. The maximum need of each process is between 1 and m resources
b. The sum of all maximum needs is less than $m + n$

From the conditions, we can get that:

$$\sum\nolimits_{i=1}^nNeed_i+ \sum\nolimits_{i=1}^nAllocation_i<m+n=\sum\nolimits_{i=1}^nMax_i$$
$$\sum\nolimits_{i=1}^nNeed_i<n$$

So there exists a process p_i such that $Need_i = 0$. Since $Max_i ≥ 1$ it follows that p_i has at least one resource that it can release.

That's why the system cannot be in a deadlock state.

7.11 Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix *Need*?

b. Is the system in a safe state?

c. If a request from process P_1 arrives for (0,4,2,0), can the request be granted immediately?

Same as question 12