第三章测试练习

一、名词解释

1．作业：在一次应用业务处理过程中，从输入开始到输出结束，用户要求计算机所做的有关该业务处理的全部工作称为一个作业。

2．周转时间：从作业被提交给系统开始，到作业完成为止的这段时间间隔。

3．带权周转时间：带权周转时间：作业的周转时间 T 与系统为它提供服务的时间 Ts 之比，即 W= T/Ts。

4．死锁：一组并发进程，它们共享系统的某些资源，该组过程中每个进程都已经占有了部分资源，但在都不释放自己已经占有资源的情况下要求获得被其他进程已经占有的资源，从而造成它们相互等待，永远不能继续推进的一种状态。

5．临界资源：”

二、选择题

1．作业调度是从处于（　　）状态的队列中选取作业投入运行，（周转时间）是指作业进入系统到作业完成所经过的时间间隔，（时间片轮转）算法不适合作业调度。

2．如果为每一个作业只建立一个进程，则为了照顾短作业用户，应采用（短作业优先）；为照顾紧急作业的用户，应采用（基于优先权的剥夺调度算法）；为能实现人机交互作用应采用（时间片轮转法）；为了兼顾短作业和长时间等待的作业应采用（高响应比优先）；为了使短作业、长作业及交互型作业用户都比较满意应采用（多级反馈队列调度算法）；为了使作业的平均周转时间最短应采用（短作业优先）算法。

3．//系统产生死锁是指（若干进程等待被其他进程所占用而又不可能被释放的资源）。//产生死锁的基本原因是（　　）和（　　），产生死锁的四个必要条件是互斥条件、（　　）、不剥夺条件和（　　）。

4．下述解决死锁的方法中，属于死锁预防策略的是（资源有序分配法），属于死锁避免策略的

是（银行家算法）。

5. 死锁的预防是通过破坏产生死锁的四个必要条件来实现的。下列方法中，(一次性分配策略）破坏了"请求与保持"条件，（资源有序分配策略）破坏了"循环等待"条件。

三、填空题

//1. 作业的输入方式包括（联机输入)、(脱机输入)、(直接耦合方式）和（SPOOLING 方式)。

2．作业在其生存期间会经历（后备)、(提交)、执行以及（完成）等状态。

3．处理机调度的类型分为（低级调度)、中级调度和（高级调度)。其中，中级调度又称为（中程调度）和（交换调度)。

4．优先数的确定分为（静态优先数）和（动态优先数）两种；在最高响应比优先算法中，响应比相应于（动态优先数)。

//5. 根据响应时间分类，可以将实时系统分为 (强实时系统)、(弱实时系统) 和一般实时系统。

6．死锁的处理方法包括（预防死锁)、(避免死锁)、(检测死锁）和（解除死锁)。

四、判断题

1.（　　）系统处于不安全状态必然会导致死锁。

2.（　　）竞争可同时共享的资源，不会导致系统进入死锁状态。(死锁：竞争不可剥夺资源或临时性资源可能引起死锁)

3.（　　）计算作业的优先权应高于 I/O 型作业的优先权。

（一般来说，I / O 型作业的优先权是高于计算型作业的优先权，这是由于 I / O 操作需要及时完成，它没有办法长时间保存所要输入 / 输出的数据）

4.（　　）资源要求多的作业，其优先权应高于资源要求少的作业。

（作业的优先权与作业的长短或者是系统资源要求的多少没有必然的关系）

5.（　　）在动态优先权时，随着进程执行时间的增加，其优先权降低。

6.（　　）预防死锁设置的限制条件比避免死锁严格，不利于进程的并发执行。

7．（　）实时系统的输出结果的正确性仅仅依赖于结果的正确性。

（输出结果的正确性不仅取决于计算所形成的逻辑结果，还要取决于结果产生的时间）

8．（　）在多级反馈队列调度算法中，优先权越高的队列，其执行的时间片越短。

9．（　）响应比是等待时间与要求服务的时间之比。（响应时间/要求服务的时间=（等待时间+要求服务时间）/要求服务的时间）

10．（　）作业的概念一般用于早期批处理系统和现在的大型机、巨型机系统中，对于微机和工作站系统一般不使用作业的概念。

第三章测试练习(课后习题)

1. 有三类资源 A(17)、B(5)、C(20)。有 5 个进程 P_1~P_5。T_0时刻系统状态如下：

	最大需求	已分配
P_1	5 5 9	2 1 2
P_2	5 3 6	4 0 2
P_3	4 0 11	4 0 5
P_4	4 2 5	2 0 4
P_5	4 2 4	3 1 4

(1) T_0时刻是否为安全状态，给出安全系列。

(2) T_0时刻，P_2：Request(0,3,4)，能否分配，为什么?

(3)在(2)的基础上 P_4：Request(2,0,1)，能否分配，为什么?

(4)在(3)的基础上 P_1：Request(0,2,0)，能否分配，为什么?

解： (1) T_0时刻

	最大需求	已分配	Need
P_1	5 5 9	2 1 2	3 4 7
P_2	5 3 6	4 0 2	1 3 4
P_3	4 0 11	4 0 5	0 0 6
P_4	4 2 5	2 0 4	2 2 1
P_5	4 2 4	3 1 4	1 1 0

T_0时刻 Available（A,B,C）= Available（2,3,3）。T_0时刻的安全性：

	Work	Need	已分配	Work+已分配	Finish
P_4	2 3 3	2 2 1	2 0 4	4 3 7	True
P_5	4 3 7	1 1 0	3 1 4	7 4 11	True
P_1	7 4 11	3 4 7	2 1 2	9 5 13	True
P_2	9 5 13	1 3 4	4 0 2	13 5 15	True
P_3	13 5 15	0 0 6	4 0 5	17 5 20	True

在 T_0时刻存在一个安全序列{P_4,P_5,P_1,P_2,P_3},故 T_0时刻是安全状态。

（2）在 T_0时刻 Request(0,3,4)<Available（2,3,3）,所以不能分配

（3）

①P_4： Request(2,0,1)<$Need_4$（2,2,1)

②P_4：Request(2,0,1)< Available（2,3,3）

③系统先假定可为 P_4 分配资源，并修改 Available（A,B,C）= Available（0,3,2）；$Need_4$=（0,2,0）；

如下

	最大需求	已分配	Need
P_1	5 5 9	2 1 2	3 4 7
P_2	5 3 6	4 0 2	1 3 4
P_3	4 0 11	4 0 5	0 0 6
P_4	4 2 5	4 0 5	0 2 0
P_5	4 2 4	3 1 4	1 1 0

④用安全算法检查系统是否安全

T_0时刻的安全性：

	Work	Need	已分配	Work+已分配	Finish
P_4	0 3 2	0 2 0	4 0 5	4 3 7	True
P_5	4 3 7	1 1 0	3 1 4	7 4 11	True
P_1	7 4 11	3 4 7	2 1 2	9 5 13	True
P_2	9 5 13	1 3 4	4 0 2	13 5 15	True
P_3	13 5 15	0 0 6	4 0 5	17 5 20	True

由所进行的安全性检查得知，可以找到一个安全序列{P_4,P_5,P_1,P_2,P_3}。因此，系统是安全的，可以将 P_4 所申请的资源分配给它。

（4）

①P_1：Request(0,2,0) <$Need_1$（3,4,7）

②P_1：Request(0,2,0) < Available（0,3,2）

③系统先假定可为 P_1 分配资源，并修改 Available（A,B,C）= Available（0,1,2）；$Need_1$=（3,2,7）；

如下

	最大需求	已分配	Need
P_1	5 5 9	2 3 2	3 2 7
P_2	5 3 6	4 0 2	1 3 4
P_3	4 0 11	4 0 5	0 0 6
P_4	4 2 5	4 0 5	0 2 0
P_5	4 2 4	3 1 4	1 1 0

④用安全算法检查系统是否安全

因为当前 Available（0,1,2）不能满足任何进程的需求，所以不能将 P_1 所申请的资源分配给它。

2. 在银行家算法中，若出现下述资源分配情况。

试问：

Process	Allocation	Need	Available
P_0	0,0,3,2	0,0,1,2	1,6,2,2
P_1	1,0,0,0	1,7,5,0	
P_2	1,3,5,4	2,3,5,6	
P_3	0,3,3,2	0,6,5,2	
P_4	0,0,1,4	0,6,5,6	

（1）该状态是否安全?

（2）若进程 P_2 提出请求 Request（1,2,2,2）后，系统能否将资源分配给它?

解：（1）

	Work	Need	Allocation	Work+ Allocation	Finish
P_0	1,6,2,2	0,0,1,2	0,0,3,2	1,6,5,4	True
P_3	1,6,5,4	0,6,5,2	0,3,3,2	1,9,8,6	True
P_1	1,9,8,6	1,7,5,0	1,0,0,0	2,9,8,6	True
P_2	2,9,8,6	2,3,5,6	1,3,5,4	3,12,13,10	True
P_4	3,12,13,10	0,6,5,6	0,0,1,4	3,12,14,16	True

该状态安全，因为存在安全状态{P_0,P_3,P_1,P_2,P_4}

（2）

①P_2：Request（1,2,2,2）<$Need_2$（2,3,5,6）

②P_2：Request（1,2,2,2）< Available（1,6,2,2）

③系统先假定可为 P_2 分配资源，并修改 Available 的值为 Available（0,4,0,0）；$Need_2$=（1,1,3,4）

④用安全算法检查系统是否安全

	Work	Need	Allocation	Work+ Allocation	Finish
P_0	0,4,0,0	0,0,1,2	0,0,3,2	0,4,3,2	True
$P_?$	0,4,3,2				

此时当前 Available（0,4,3,2）不能满足任何进程（P_1, P_2, P_3, P_4）的需求，所以不能将 P_2 所申请的资源分配给它。