作业（7）答案

1.思考题

（6）什么是临界区？什么是临界资源?什么是竞争条件?

临界资源：并发进程中共享变量所代表的资源；一次只能供一个进程使用的资源。

临界区：并发进程中，各个程序中访问临界资源的那部分程序（代码）。

系统中的若干个进程因相互争夺独占型资源时所产生的制约关系，称这些进程之间为竞争关系。这些独占型资源为竞争条件。

（7）试述临界区管理的基本原则。

一次至多允许一个进程停留在相关的临界区内

一个进程不能无限止地停留在临界区内

一个进程不能无限止地等待进入临界区

（17）试述产生死锁的必要条件。

a) 互斥条件(mutal exclusion)：进程互斥使用临界资源

b) 占有和等待条件(hold and wait)：进程在申请新资源得不到满足而等待时，不释放已经占有资源

c) 不剥夺条件(no preemption)：一个进程不能抢夺其他进程占有的资源

d) 循环等待条件(circular wait)：存在一个循环等待链，每个进程分别等待它前一个进程所持有的资源，造成永远等待

（18）列举死锁的各种防止策略。

策略(1)：使资源可同时访问而非互斥。（破坏互斥条件）

有些资源可以采取此策略，如：只读文件、时钟、磁盘等，有些资源不能采取，如：可写文件、键盘、磁带机等。

策略（2）：采用静态分配策略（破坏占有和等待)

静态分配是指一个进程必须在执行前就申请它所要的全部资源，并且直到它所要的资源都得到满足后才开始执行。如果有一种资源不能满足进程，即使其它资源都空闲，也不分配给进程，而让进程等待。

策略(3)：采用剥夺式调度方法（适用于内存和处理器资源）（破坏不剥夺条件）

当进程在申请资源未获准许的情况下,应主动释放已经保持了的资源，以后再去重新申请。这被认为使进程已经占有的资源被剥夺了。

策略（4）：采用层次分配策略（破坏循环等待条件）

资源被分成多个层次，当进程得到某一层的一个资源后，它只能再申请较高层次的资源

当进程要释放某层的一个资源时，必须先释放占有的较高层次的资源

当进程得到某一层的一个资源后，它想申请该层的另一个资源时，必须先释放该层中的已占资源

2.应用题

（2）两个进程 P1 和 P2 并发执行，其程序代码分别如下。

```
P1(){
    while(true){
        ①k = k * 2;
        ②k = k + 1;
    }
}
```

```
P2(){
    while(true){
        ③print k;
        ④k = 0;
    }
}
```

若令 k 的初值为 5，在进程 P1 执行了两个循环后，进程 P1 和 P2 又并发执行了一个循环。写出该过程中可能的打印值，并指出其中与时间有关的错误。

解答如下：

将四条语句编号如程序中所示：

进程 P1 执行了两个循环后，k=23

接下来，进程 P1 和 P2 并发执行一个循环，可能的执行顺序有：

①②③④ 打印值为：47

③④①② 打印值为：23

①③②④ 打印值为：46

①③④② 打印值为：46

③①②④ 打印值为：23

③①④② 打印值为：23

与时间有关的错误：结果不唯一!!

（7）有如下两个优先级相同的进程 P1 和 P2，已知信号量 S1 和 S2 的初值均为 0，试问 P1、P2 并发执行后 x、y、z 的值各为多少？

```
P1(){
    ①y = 1;
    ②y = y + 3;
    V(S1);
    ③z = y + 1;
    P(S2);
```

```
P2(){
    ⑤x = 1;
    ⑥x = x + 5;
    P(S1);
    ⑦x = x + y;
    V(S2);
```

④y = z + y;
}

⑧z = z + x;
}

解答：

将各个语句编号如程序中所示：

语句①②⑤⑥没有关系，所以并发执行不会产生结果不唯一，执行完之后的结果为：x =6, y = 4;

接下来根据 PV 操作的作用，可能的执行顺序有：

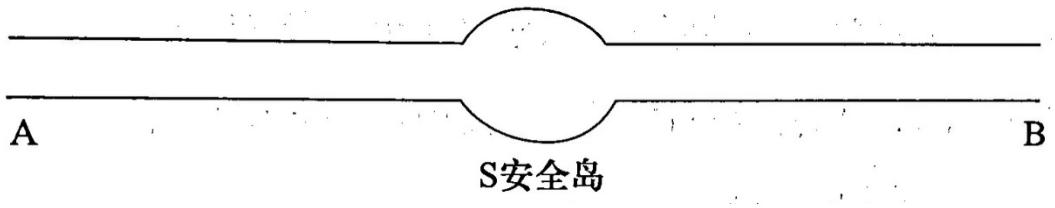
语句执行顺序	x 的值	y 的值	z 的值
③⑦④⑧	10	9	15
③⑦⑧④	10	19	15
⑦③④⑧	10	9	15
⑦③⑧④	10	19	15
⑦⑧③④	10	9	5

综上，x，y，z 有三组值，分别是：x=10，y=9，z=15

x=10，y=19，z=15

x=10，y=9，z=5

（15）现有一个如题图所示的小巷，除安全岛可容 2 人暂时停身外，仅能容 1 人通过，若 A、B 两端都允许行人进出，试使用信号量与 PV 操作设计一个算法，让两端行人顺利通过小巷。



这个题目主要考虑的不能产生死锁。

A 端设置一个信号量 a1=1，每个 A 端的人都要申请，通过整个巷子之后释放（否则不能满足安全岛人数限制）；

B 端设置一个信号量 b1=1，每个 B 端的人都要申请，通过整个巷子之后释放（否则不能满足安全岛人数限制）；

安全岛最多两个人，由（a1+b1=2）控制；

需要使用 S-A，设置一个信号量 a2=1，通过前申请，通过后释放；

需要使用 S-B，设置一个信号量 b2=1，通过前申请，通过后释放；；

A-S 和 B-S 这个可以同时通过；

算法如下：

semaphore a1,a2,b1,b2; a1=1;a2=1;b1=1;b2=1; cobegin	

Process PA(){
P(a1);
P(a2);
行人通过 A-S 到达安全岛;
V(a2);
P(b2);
行人通过 S-B;
V(b2);
V(a1);
}
	Process PB(){
	P(b1);
	P(b2);
	行人通过 B-S 到达安全岛;
	V(b2);
	P(a2);
	行人通过 A-S;
	V(a2);
	V(b1);
	}

coend	

（17）有一个阅览室，读者进入时必须先在一张登记表上登记，此表为每个座位列出一个表目，包括座位号、姓名，读者离开时要注销登记信息；假如阅览室共有 100 个座位。试用信号量和 PV 操作实现用户进程的同步算法。

进程：

每个读者 i 进出阅览室的一次活动为一个进程，进程之间无合作关系。

信号量设计

mutex：用于读者之间对登记表登记、注销登记信息时的互斥信用量，初值为 1，每个读者在进入时申请，登记完登记表释放；每个读者离开时申请，注销登记信息完毕之后释放；

empty：阅览室容量的信号量，初值为 100；每个读者进入阅览室之前申请，退出阅览室时释放；

算法如下：

semaphore mutex, empty; mutex=1; empty=100; cobegin

process reader_i()//i=1,2,…..
{ P(empty); //申请阅览室座位
P(mutex); //申请对登记表信息进行互斥地写

登记登记表信息；
V(mutex);　//释放登记表互斥信号量
 	
读者进入阅览室
 	
P(mutex);　//申请对登记表信息进行互斥地写
注销登记表信息；
V(mutex); //释放登记表互斥信号量
V(empty);　　//释放阅览室座位

coend

（21）一个经典的同步问题：吸烟者问题（Patil，1971 年）。三位吸烟者在同一个房间内，还有一位香烟供应者。为了制造并抽掉香烟，每位吸烟者需要三样东西：烟草、纸和火柴，供应者有丰富的货物提供。三位吸烟者中，第一个人有自己的烟草，第二个人有自己的纸，第三个人有自己的火柴。供应者随机地将两样东西放在桌子上，允许一位吸烟者吸烟。当吸烟者吸完烟后唤醒供应者，供应者再将两样东西放在桌子上，唤醒另一位吸烟者。试采用信号量和 PV 操作编写他们同步工作的程序。

设四个进程：S，A，B，C

S 为供应者，A，B，C 为三个吸烟者进程，A 拥有烟草，B 拥有烟纸，C 拥有火柴

本问题属于生产者-消费者模型

信号量设计

mutex：用于生产者-消费者同步的信号量，初值为 1

SA：A 进程需要的信号量，初值为 0

SB：B 进程需要的信号量，初值为 0

SC：C 进程需要的信号量，初值为 0

供应者 S：每次申请 mutex，根据放的原材料释放不同的信号量；

吸烟者 A：每次申请属于自己的信号量 SA，释放 mutex

吸烟者 B：每次申请属于自己的信号量 SB，释放 mutex

吸烟者 C：每次申请属于自己的信号量 SC，释放 mutex

同步算法如下：

<table>
<tr><td colspan="4">semaphore mutex，SA，SB，SC；
mutex=1; SA=0; SB=0; SC=0；
cobegin</td></tr>
<tr><td>process S()
{
while（1）{
P(mutex);
if(paper&&match) (SA);
else if(tobacco&&match)
V(SB);
else V(SC);
}
}</td><td>process A()
{
while（1）{
P(SA);
从桌子上取走两种原材料；
V(mutex)；
卷烟吸烟；
}
}</td><td>process B()
{
while（1）{
P(SB);
从桌子上取走两种原材料；
V(mutex)；
卷烟吸烟；
}
}</td><td>process C()
{
while（1）{
P(SC);
从桌子上取走两种原材料；
V(mutex)；
卷烟吸烟；
}
}</td></tr>
<tr><td colspan="4">coend</td></tr>
</table>

（24）试用信号量和 PV 操作实现睡眠的理发师问题：理发店里有一位理发师、一把理发椅和 n 把供等候理发的顾客坐的椅子。要求：①如果没有顾客，理发师便在理发椅上睡觉；②一个顾客到来时，他必须叫醒理发师；③如果理发师正在理发时又有顾客来到，如果有空椅子可坐，就坐下来等待，否则就离开。

全局变量：waiting，表示顾客等候的顾客数

信号量设计：

一组同步信号量：barbers 和 customers

barbers：由理发师发送给顾客（V 操作），顾客申请该信号量（P 操作）

customers：由顾客发送给理发师（V 操作），理发师申请该信号量（P 操作）

互斥信号量 mutex：理发师和顾客对对全局变量 waiting 进行读写操作时需要申请/释放的互斥信号量

同步算法如下：

```
int waiting=0;   //等候理发顾客数
int CHAIRS=N; //为顾客准备的椅子数
semaphore customers, barbers, mutex;
```

customers=0; barbers=0; mutex=1; cobegin	

process barber( ) {
while(true) {
P(customers); //有顾客吗？若无顾客，理发师睡眠
P(mutex); //若有顾客时，进入临界区
waiting--; //等候顾客数少一个
V(barbers); //理发师准备为顾客理发
V(mutex); //退出临界区
cut_hair(); //理发师正在理发(非临界区)
}
}
	process customer_i( ) {
	P(mutex); //进入临界区
	if(waiting<CHAIRS) { //有空椅子吗
	waiting++; //等候顾客数加 1
	V(customers); //唤醒理发师
	V(mutex); //退出临界区
	P(barbers); //理发师忙，顾客坐下等待
	get_haircut(); //否则顾客坐下理发
	}
	else V(mutex); //人满了，走吧！
	}

coend	

（26）假设系统有 n 个进程共享 m 个资源，已知每个进程一次只能申请或释放一个资源，且每个进程最多需要 m 个资源，所有进程的资源需求总数少于 m+n 个，证明系统此时不会产生死锁。

证明：根据进程的资源需求总数少于 m+n 个，有：nm<m+n （1）

对不等式（1）进行变换有：nm-n <m， （2）

即， n(m-1) <m （3）

不等式（3）左边 n(m-1)可以表示每个进程都已经获得比最多需求少 1 个资源。m 为整数，说明当处于这种情况下，系统资源至少剩余 1 个，则可以满足一个进程的剩余需求，该进程可以运行完，接下来所有进程都可以运行完，因此，系统不会死锁。

（27）设当前的系统状态如下表所示，此时 Avavailable=（1，1，2）。试回答下列问题：①计算各个进程还需要的资源数 C_{ki}-A_{ki}。②此时系统是否处于安全状态，为什么？③进程 P_2 发出请求向量 $request_2$（1，0，1），系统能把资源分配给它吗？④若在进程 P_2 申请资源后，P1 发出请求向量 $request_1$（1，0，1），系统能把资源分配给它吗？⑤若在进程 P1 申请资源后，P3 发出请求向量 $request_3$（0，0，1），系统能把资源分配给它吗？

进程	Claim	Allocation
	R1 R2 R3	R1 R2 R3
P1	3 2 2	1 0 0
P2	6 1 3	5 1 1
P3	3 1 4	2 1 1
P4	4 2 2	0 0 2

解答如下：

①计算各个进程还需要的资源数 C_{ki}-A_{ki}。

进程	C_{ki}-A_{ki}
	R1 R2 R3
P1	2 2 2
P2	1 0 2
P3	1 0 3
P4	4 2 0

②此时系统是否处于安全状态，为什么？

(a) 初始化：CurrentAvailable=Available=（1，1，2）

(b) CurrentAvailable 可以满足进程 P2 还需要的资源数，因此 P2 可以运行完，归还资源：CurrentAvailable=（1，1，2）+（5，1，1）=（6,2,3）

(c) CurrentAvailable=（6,2,3）可以满足所有进程的还需要资源数，因此可以按照进程 P1，P3，P4 的顺序进行调度，所有进程都可以运行完。

综上，存在进程调度序列 P2，P1，P3，P4，按照这个顺序，所有进程都可以执行完，所以系统处于安全状态。

③进程 P_2 发出请求向量 $request_2$（1，0，1），系统能把资源分配给它吗？

$request_2$（1，0，1）<Available（1，1，2）

$request_2$（1，0，1）< C_{k2}-A_{k2}（1，0，2）

系统进行预分配之后，修改资源数如下：

Available=（1，1，2）-（1，0，1）=（0,1,1）

进程	Claim	Allocation	C_{ki}-A_{ki}
	R1 R2 R3	R1 R2 R3	R1 R2 R3
P1	3 2 2	1 0 0	2 2 2
P2	6 1 3	6 1 2	0 0 1
P3	3 1 4	2 1 1	1 0 3
P4	4 2 2	0 0 2	4 2 0

接下来进行安全性检查

(a) 初始化：CurrentAvailable=Available=（0,1,1）

(b) CurrentAvailable 可以满足进程 P2 还需要的资源数，因此 P2 可以运行完，归还资源：CurrentAvailable=（0，1，1）+（6，1，2）=（6,2,3）

(c) CurrentAvailable=（6,2,3）可以满足所有进程的还需要资源数，因此可以按照进程 P1，P3，P4 的顺序进行调度，所有进程都可以运行完。

综上，存在进程调度序列：P2，P1，P3，P4，按照这个顺序，所有进程都可以执行完，所以系统处于安全状态，系统能把资源分配给 P2。

④若在进程 P_2 申请资源后，P1 发出请求向量 $request_1$（1，0，1），系统能把资源分配给它吗？

由于 $request_1$（1，0，1）>Available（0,1,1），系统资源已经不足，进程 P1 等待。

⑤若在进程 P1 申请资源后，P3 发出请求向量 $request_3$（0，0，1），系统能把资源分配给它吗？

$request_3$（0，0，1）<Available（0,1,1）

$request_3$（0，0，1）< C_{k3}-A_{k3}（2，1，1）

系统进行预分配之后，修改资源数如下：

Available=（0，1，1）-（0，0，1）=（0,1,0）

进程	Claim	Allocation	C_{ki}-A_{ki}
	R1 R2 R3	R1 R2 R3	R1 R2 R3
P1	3 2 2	1 0 0	2 2 2
P2	6 1 3	6 1 2	0 0 1
P3	3 1 4	2 1 2	1 0 2
P4	4 2 2	0 0 2	4 2 0

初始化：CurrentAvailable=Available=（0,1,0）,此时已经不能满足任何进程还需要的资源数，因此系统不会将资源分配给进程 P3，重新修改资源数如下：

进程	Claim	Allocation	C_{ki}-A_{ki}
	R1 R2 R3	R1 R2 R3	R1 R2 R3
P1	3 2 2	1 0 0	2 2 2
P2	6 1 3	6 1 2	0 0 1
P3	3 1 4	2 1 1	1 0 3
P4	4 2 2	0 0 2	4 2 0