一、选择题

1. C 2. C 3.C 4.A 5.C 6.B

一、选择题

1.A 2.B 3.C 4.B 5.A 6.D 7.B 8.D

二、综合应用题

1. 答：该算法有错。一方面，首次访问临界资源必须是A进程，如果A进程永不要求访问临界资源，则不会执行V(S1)，那么B进程永远得不到操作临界资源的机会；另一方面，如果A进程首次访问了临界资源，那么接下来，只能两个进程交替使用临界资源，如果有一个进程不再访问临界资源，则另一个进程下次将得不到操作临界资源的机会。所以问题在于：本应进行互斥控制，却使用的是同步控制。改正如下：

设置信号mutex

mutex:=1;

Cobegin

A进程：

Begin

repeat

P(mutex);

CSA;

V(mutex);

until false

End

B进程：

Begin

repeat

P(mutex);

CSB;

V(mutex);

until false

End

Coend

2. 答：一次仅允许一个进程使用的资源称为临界资源，在进程中对临界资源访问的程序段称为临界区。

这种算法不是安全的。因为，进入区enter-crtsec()不是一个原子操作，如果两个进程同时执行完其循环(此前两个flag均为false)，则这两个进程可同时进入临界区。

3. 答：以下是两个进程并发执行的一种调度情况：

P0进程：

do{

blocked[0]=true;

②

④

⑥

while(turn!=0)

{

while(blocked[1]);

turn=0;

}

P0的临界区

blocked[0]=false;

P0的非临界区

}while(true);

P1进程：

do{

blocked[1]=true;

①

⑤

③

while(turn!=1)

{

while(blocked[0]);

turn=1;

}

P1的临界区

blocked[1]=false;

P1的非临界区

}while(true);

这个双进程的临界区问题算法是错误的。在上面的调度中：

（1）假设P1先获得CPU并进入临界区，此时出现中断；

（2）P0获得CPU，而P0也试图进入临界区，它只能在“while(blocked[1]);”处等待，无法进入，此时又发生中断；

（3）P1重新获得CPU，并在退出临界区后执行“blocked[1]=false;”，此时又出现中断；

（4）P0获得CPU，P0执行完“while(blocked[1]);”，但还没有执行“turn=0;”，此时又出现中断；

（5）P1得到CPU开始执行，它执行“while(turn!=1);”发现条件为假，于是进入临界区，此时再出现中断；

（6）P0得到CPU，P0执行完“turn=0;”后，也进入临界区。

此时P0进程和P1进程同时出现在各自的临界区中，都在访问临界资源，所以该双进程临界区问题算法没有满足“忙则等待”即互斥的条件，所以是错误的。

4. 答：semaphore a=b=c=d=e=f=g=h=0;

P1(){…;V(a);V(b);}

P2(){P(a);…;V(c);V(d);}

P3(){P(b);…;V(e);V(f);}

P4(){P(c);P(e);…;V(g);}

P5(){P(d);P(f);…;V(h);}

P6(){P(g);P(h);…;}

5. 答：确定P、V操作的位置：

司机操作中，是否关前门？没关则等待，这是一个P操作，P(Door1)；是否关后门？没关则等待，这是一个P操作，P(Door2)；

司机操作中，设立启动标志，通知前、后门售票员可以售票，这是两个V操作，V(T1) ，V(T2)；

司机操作中，设立停车标志，通知前、后门售票员可以开门，这是两个V操作，V(S1) ，V(S2)；

前门售票员售票操作中，汽车是否启动？没启动则等待，这是一个P操作，P (T1)。

前门售票员开门操作中，是否停车？没停则等待，这是一个P操作，P(S1)；

前门售票员关门操作中，设立关门标志，这是一个V操作，V(Door1)；

后门售票员售票操作中，汽车是否启动？没启动则等待，这是一个P操作，P (T2)。

后门售票员开门操作中，是否停车？没停则等待，这是一个P操作，P(S2)；

后门售票员关门操作中，设立关门标志，这是一个V操作，V(Door2)；

由于汽车初始状态处于行驶之中，所以Door1=Door2=0，T1= T2=0（不严格），S1= S2=0，所有信号量的取值范围都是-1~1。

int Door1=Door2=1;

int S1= S2=0;

int T1= T2=1;

main()

{

cobegin

driver();busserver 1();busserver 2();

coend

}

driver()

{ do while T

{ P(Door1)；P(Door2)；

启动车辆；

V(T1)；V(T2)；

正常行车；

到站停车；

V(S1)；V(S2);

}

}

busserver 1()

{ do while T

{

P(T1)；

售票；

P(S1)；

开前门；

关前门；

V(Door1)；

}

}

busserver 2()

{ do while T

{

P(T2)；

售票；

P(S2)；

开后门；

关后门；

V(Door2)；

}

}

6. 答：

var mutex=1,customer\_num=0:semaphore;

customer()

{

取一个号码；

p(mutex);

进入队列；

v(mutex);

v(customer\_num);

}

seller()

{

while(1)

{p(customer\_num);

p(mutex);

从队列中取一个号码；

v(mutex);

为持有该号码的顾客服务；

}

}

7. 答：

int count=0;

//记录理发店里的顾客数量

semaphore barber\_chair=1;

//理发椅一张

semaphore wait\_chair=3;

//顾客等待时坐的椅子三张

semaphore mutex=1;

//用于互斥访问count变量所用的信号量

semaphore ready=0;

//把准备好可以理发的顾客数量当成理发师所需的资源

semaphore finish=0;

//把理发师完成理发后发给顾客的信号作为理发中的顾客等待的资源

Barber(){

//理发师进程

while(1){

wait(ready);

//等待理发椅上的顾客准备好

barbering

signal(finish);

//告诉理发椅上的顾客理发完成

}

}

Customer(){

//顾客进程

while(1){

wait(mutex);

if(count>=4){

//如果理发店已经有四个顾客了

signal(mutex);

leave

}

else{

//理发店里顾客不足4个

count++;

//更新顾客人数

signal(mutex);

}

wait(wait\_chair);

//先请求坐等待时坐的椅子

wait(barber\_chair);

//再请求坐理发椅

signal(wait\_chair);

//坐上理发椅后让出等待时坐的椅子

signal(ready);

//告诉理发师I'm ready

be barbered

wait(finished);

//等待理发师完成理发

signal(barber\_chair);

//离开理发椅

wait(mutex);

count--;

//更新店里的顾客人数

signal(mutex);

}

}

8. 答：var seats,raaders,mutex:semaphore;

seats:=50;

readers:=0;

mutex:=1;

cobegin

procedureEnter

begin

while(TRUE)

begin

p(seats);

p(mutex);

填写登记表;

进入阅览室阅读;

v(mutex);

v(readers);

end

end

procedureLeave

begin

while(TRUE)

begin

p(readers);

p(mutex);

消掉登记;

离开阅览室;

v(mutex);

v(seats);

end

end

coend