**南大OS 1999年参考答案**

1

(1)

批处理系统着眼于让处理器和外围设备同时保持忙碌，提高作业的吞吐率和整个系统的效率。其关键机制是：在响应一个作业的处理结束信号时，处理器将在主存中驻留的不同作业间切换；并具有用户脱机工作，成批处理作业，多道并发执行的特点。

(2)

分时交互系统的设计目标是为用户提供方便的程序开发和调试环境和快速响应交互式用户的命令请求，但又要支持多用户同时工作，以降低系统的成本。其关键机制是：采用时间片轮转法，让处理器在多个交互式用户间多路复用；

(3)

实时操作系统监控控制对象并能做出及时反应，例如，数据库的查询和修改应用或生产过程控制实时应用，实时应用有更加严格的时间限制，可靠性高、响应及时，但资源利用率低。其关键机制是：事件驱动机制，当系统接受来自外部的事件后，快速分析这些事件，驱动实时任务在规定的响应时间完成相应处理和控制。

(4)

分布式操作系统具备进程通信、资源共享、并行运算、网络管理等基本功能，用于管理分布式计算机系统。它的主要优点和特点是：坚定性强、扩充容易、可靠性好、维护方便和效率较高。

2

(1)

分段是信息的逻辑单位，由源程序的逻辑结构所决定，用户可见，段长可根据用户需要来规定，段起始地址可以从任何主存地址开始。在分段方式中，源程序（段号，段内位移）经连结装配后仍保持二维(地址)结构。

分页是信息的物理单位，与源程序的逻辑结构无关，用户不可见，页长由系统确定，页面只能以页大小的整倍数地址开始。在分页方式中，源程序（页号，页内位移）经连接装配后变成了一维(地址)结构。

(2)

若干个逻辑记录合并成一组，写入一个块叫记录成组，把逻辑记录从块中分离出来的操作叫记录的分解。记录成组和分解处理不仅节省存储空间，还能减少输入输出操作次数，提高系统效率。

(3)

首先，能提高设备的利用率，其次，作业执行中不再和低速的设备联系，也就减少了作业等待输入输出数据的时间，从而，缩短了作业全部的执行时间。此外，还具有能增加多道程序的道数，增加作业调度的灵活性等优点。可见，操作系统提供了外围设备联机同时操作功能后，系统的效率会有很大提高。

(4)

第一，可防止死锁，死锁的预防是系统预先确定一些资源分配策略，进程按此规定来申请和使用资源，保证死锁的一个必要条件不会满足，使得系统不发生死锁。采用的方法有：静态资源分配法、层次或按序资源分配法、剥夺式资源调度法、共享资源使用法等。第二，可避免死锁，死锁的避免涉及到动态地分析和检测新的资源请求和资源的分配情况，以确保系统始终处于安全状态，如采用银行家算法。第三，可使用死锁检测和解除法。

3

1. Job1从投入到运行完成需80ms，Job2从投入到运行完成需90ms，Job3从投入到运行完成需90ms。
2. CPU空闲时间段为：60ms至70ms，80ms至90ms。所以CPU利用率为(90-20)/90=77.78%。
3. 设备I1空闲时间段为：20ms至40ms，故I1的利用率为(90-20)/90=77.78%。设备I2空闲时间段为：30ms至50ms，故I2的利用率为(90-20)/90=77.78%。

4

var mutex1,mutex2:semaphore;

empty1,empty2;semaphore;

full1,full2:semaphore;

F1,F2:array[0..9] of product;

In1,in2,out1,out2:integer;

mutex1:=mutex2:=1;

empty1:=empty2:=10;

full1:=full2:=0;

in1:=in2:=out1:=out2:=0;

main( )

{

cobegin

{process 生产车间A

begin

repeat

produce product A;

P(empty1);

P(mutex1);

F1[in1]:= product A ;

in1:=(in1+1) mod 10;

V(mutex1);

V(full1);

Until false;

end;

process 生产车间B

begin

repeat

produce product B;

P(empty2);

P(mutex2);

F2[in2]:= product B;

in2:=(in2+1) mod 10;

V(mutex2);

V(full2);

Until false;

end;

process 装配车间

begin

repeat

P(full1);

P(full2);

P(mutex1);

Get the product A from F1[out1];

out1:=(out1+1) mod 10;

V(mutex1);

V(empty1);

P(mutex2);

Get the product B from F2[out2];

out2:=(out2+1) mod 10;

V(mutex2);

V(empty2);

Install the product A and B;

Until false;

end;

coend

}

5

TYPE sharefile =MONITOR

VAR numbersum,n:integer;

SF:codition;

DEFINE startread,endread;

USE wait,signal,check,release;

procedure startread(var number:integer:);

begin

check(IM);

L: if (number+numbersum)≥K then {wait(SF,IM); goto L;}

numbersum:=numbersum+number;

release(IM);

end

procedure endread(var number:integer;);

begin

check(IM);

numbersum:=numbersum-number;

signal(SF,IM);

release(IM);

end

begin

numbersum:=0;

end.

main()

{cobegin

process-i();

coend

}

process-i()

var number:integer;

begin

number:=进程读文件编号;

startread(number);

read F;

endread(number);

end.

**南大OS 2000年参考答案**

一、

Program Status Word，Program Control Block，

First Come First Service，Communicating Sequential Processes，

Least Recently Used，Direct Memory Access，Remote Procedure Call，

Network Of Workstation，Distributed Shared Memory，

Simultaneous Peripheral Operations On Line

二、1 (b) (c)

2 (b)和(c) ，(a)

3 (c) (d)

4 (a) (c) (d)

5 (a) (b) (c) (d)

6 (b) (c) (d)

三、7 **× ，8 √ ，9 ×，10 × ，11 × ，12 √**

四、

13

优先级调度的基本原则是：照顾时间紧迫的作业使其得到及时处理、照顾会话型作业使其快速响应、照顾I/O繁忙型作业，使外设充分保持忙碌。故I/O为主的作业对应的进程优先级高，而以计算为主的作业对应的进程优先级低。

14

一是关中断时间过长会影响系统效率，限制了处理器交叉执行程序的能力；二是关中断方法不适用于多CPU系统，因为，在一个处理器上关中断，并不能防止进程在其他处理器上执行相同临界段代码；三负责开关中断交给用户完成，加重了其负担。

15

按序分配策略把系统的所有资源安排一个顺序，按顺序给每个资源一个编号，规定每个进程申请两个以上资源时，总是先申请编号小的再申请编号大的资源。这样，在进程集合中总存在某个进程，它占有了己申请资源中编号最大的资源，因而，它不再能申请其他资源，当它运行结束，就可以释放占用的全部资源。剩余的进程集合中又会有一个进程此时占有己申请资源中编号最大的资源，那么，它也能运行结束。以此类推，最终所有进程都能运行结束，故系统不会发生死锁。实质上，按序分配通过破坏死锁的循环等待条件而防止死锁。

16

线程是操作系统进程中能够独立执行的实体（控制流），是处理器调度和分派的基本单位。线程是进程的组成部分，每个进程内允许包含多个并发执行的实体（控制流），这就是多线程。同一个进程中的所有线程共享进程获得的主存空间和资源，可并发或并行地执行，而进程之间的并发执行演变为不同进程的线程之间的并发执行。

17

均为3次。FIFO淘汰了页面1、2、3，LRU淘汰了页面3、4、5。

五、

18

可能amount为10元、20元或30元。只要定义一个信号量S，其初值为1。兄弟两人的取钱前，结束后增加P(S)与V(S)即可。

19

## TYPE FMSD = MONITOR

## var plate ARRAY[0,1] of (apple, orange);

## count:integer; flag0,flag1:boolean;

## SP, SS, SD : condition;

## define put, get;

## use wait, signal, check, release；

# 

## procedure put(var fruit:(apple, orange));

## begin

## check(IM);

## if (count==2) then wait(SP,IM)；

## else { if (flag0==false) then

## plate [0]:= fruit; flag0:=true;

## else plate[1]:=fruit; flag1:=true;

## count:=count+1;

## if (fruit==orange) then signal(SS,IM)；

## else signal(SD,IM);

## }

## release(IM);

## end;

## procedure get(var fruit:(apple,orange),x:plate);

## begin

## check(IM);

## if (count= =0) or plate<>fruit

## then begin

## if (fruit= =orange) then wait(SS,IM)；

## else wait(SD,IM)；

## end;

## count:=count-1;

## if (flag0 & plate[0]= =fruit) then

## { x := plate[0];flag0:=false;}

## else { x:= plate[1]; flag1:=false;}

## signal(SP, IM);

## release(IM);

## end;

## begin

## count:=0;flag0:=false;flag1:=false; SP := 0; SS := 0;

## SD := 0；plate[0]:=plate[1]:=null;

## end;

## main()

## {cobegin

## process father

## begin

## while (1)

## {

## prepare the apple;

## call FMSD.put(apple);

## ……

## }

## end;

## process mother

## begin

## while (1)

## {

## prepare the orange;

## call FMSD.put(orange);

## ……

## }

## end;

## process son

## begin

## while (1)

## { call FMSD.get(orange,x);

## eating thr orange;

## ……

## }

## end;

## process daughter

## begin

## while (1)

## { call FMSD.get(apple,x);

## eating the apple;

## ……

## }

## end;

## coend

## }

**南大OS 2001年参考答案**

一、（17）分

1．线程：线程又称轻量级进程，是指进程中的一条执行路径。

目录：文件系统中的所有文件控制的有序集合称为目录。

死锁：若一个进程集合中的每一个进程都在等待只能由本集合中的另一个进程才能引发的事件，则称这组进程处于死锁。

窗口：把用户的工作显示在计算机屏幕的一个矩形区域中，允许用户在矩形区域中操作应用软件和文件，该矩形区域就称为“窗口”。

2．（E）在系统资源中的潜在的并发活动增加了。

3．（D）每个请求在系统上运行的平均时间增加了。

4．（A） (B)

二、（18分）

1．

（1）进程：系统进行资源管理和保护的单位，与中级别调度相关的实体。

内核级线程：进程的一条执行路径，操作系统进行处理器调度的实体。

用户级线程：进程的一条执行路径，操作系统不知道它的存在，在执行时映射到内核级线程上，用户调度的实体。

（2）划分成实时优先级层次和交互式优先级层次，其中实时优先级层次较高。

实时优先级层次包括多个优先级，可以组织成多个就绪线程队列，也可以组织成一个优先队列；可以采用抢占式优先数调度策略，如果分配时间片，应该较长。

交互式优先级层次可以划分成3个就绪线程队列，按照优先从高到低依次为访问字符设备的就绪线程队列、访问块设备的就绪线程队列，时间片到的就绪线程队列，优先级较高的就绪线程队列具有较短的时间片。

（注：时间片一般为100ms数量级）

2．

最短查找时间优先算法中总跨越：1+3+4+11+4+5=28个柱面。

电梯调度算法中总跨越：1+4+4+5+16+4=34个柱面

分析两种算法可以发现，最短查找时间优先法使得整个系统效率较电梯调度算法为高，但仔细分析发现，该算法可能会在某种情况下使得单个请求发生“饿死”现象。比如存在访问200柱面的请求时，该请求将很难在有限时间内得到满足，而电梯调度度算法在单方向上是“最短查找时间优先”，而绝对的单向移动能保证某个请求的“响应及时”，避免“饿死”。

三、答案（25分）**(本题的详细分析、解答见模拟考试题)**

**南大OS 2002年参考答案**

一、（每题2分 共10分）

（1）A，D （2）D （3）B （4）B，C （5）A，D

二、（每题3分 共12分）

（1）没有。进程是资源分配和保护的单位，线程是处理器调度的单位，而挂起是由于平顺系统的资源负载引起的，只适用于进程。

（2）给每个进程（线程）分配若干张彩票，调度时根据随机数进行抽签，让中签的进程（线程）占有处理器（或使用其他资源）运行。

（3）创建文件、打开文件、关闭文件、读文件、写文件、撤销文件

（4）当归还分区与其他空闲分区不相邻，空闲分区数增加一个；当归还分区与一个空闲分区上相邻或下相邻，空闲分区数不变；当归还分区与两个空闲分区分别上相邻和下相邻，空闲分区数减少一个。

三、（9分）

BUF

A

X mod 3=0

X mod 3=1

X mod 3=2

X

B

C

D

VAR BUF:integer;

S,S1,S2,S3:semaphore;

S:=1; S1:=0; S2:=0; S3:=0;

cobegin

process A process C

X:integer; SUM2:integer;

begin begin

L：从文件中读入一个整数到X； SUM2:=0；

P(S); L2:P(S2);

BUF:=X; SUM2:=SUM2+BUF;

if X MOD 3=0 then V(S1); V(S);

if X MOD 3=1 then V(S2); GOTO L2;

if X MOD 3=2 then V(S3); end;

GOTO L; process D

end; SUM3:integer;

process B begin

SUM1:integer; SUM3:=0;

begin L3:P(S3);

SUM1:=0; SUM3:=SUM3+BUF;

L1:P(S1); V(S);

SUM1:=SUM1+BUF; GOTO L3;

V(S); end;

GOTO L1;

end;

coend.

四、（9分）

## type PQR=MONITOR

## var buf:integer;

## SP,SQ,SR:condition;

## turn:{p,q,r};

## DEFINE PPUT,QGET,QPUT,RGET;

## USE wait,signal,check,release;

# 

## procedure PPUT(var data:integer;);

## begin

## check(IM);

## if turn !=p then wait(SP,IM);

## turn:=q;

## buf:=data;

## signal(SQ,IM);

## release(IM);

## end

# 

## Procedure QGET(var data:integer;);

## begin

## check(IM);

## if turn !=q tnen wait(SQ,IM);

## data:=buf

## release(IM);

## end

# 

## procedure QPUT(var data:integer;);

## begin

## check(IM);

## turn:=r;

## buf:=data;

## signal(SR,IM);

## release(IM);

## end

# 

## Procedure RGET(var data:integer;);

## begin

## check(IM);

## if turn !=r tnen wait(SR,IM);

## turn:=p;

## data:=buf

## signal(SP,IM);

## release(IM);

## end

## begin

## SP:=0;SQ:=0;SR:=0;turn:=p;

## end

# 

## main()

## { cobegin

## process P

## x:integer;

## begin

## LP:从文件读入一个数据到x;

## PPUT(x);

## goto LP;

## end

## 

## process Q

## x:integer;

## begin

## LQ:QGET(x);

## 加工处理x;

## QPUT(x);

## goto LQ;

## end

## process R

## x:integer;

## begin

## LR:RGET(x);

## 打印x;

## goto LR;

## end

## coend

## }

**南大OS 2003年参考答案**

一、（每个0.5）共2分

1．（拼写错误可酌情少扣或不扣）

（1）Reundant Array of Independent Disks

（2）Light Weight Process

（3）Interprocess Communication

（4）Inverted Page Table

2．（概念回答有误要扣1-1.5分，关系未答可扣0.5分）

（1）计算机系统的安全性和可靠性是两个概念，可靠性指硬件系统正常持续运行的程度，目标为反故障；安全性是指不因人为疏漏和蓄谋作案而导致信息资源被泄露、篡改和破坏，目标是反泄露。

关系：可靠性是安全性的基础，安全性比可靠性更为复杂。

（2）死锁是因进程竞争资源，但系统拥有资源的数量有限，或并发进程推进的顺序不当而造成的一种永远等待资源的僵局。而饥饿是指每个资源占用者都在有限时间内释放占用的资源，但申请进程仍然长时间得不到资源的现象。

关系：产生饥饿现象并不意味着一定产生死锁。

（3）并发进程的执行会产生相互制约关系：一种是进程之间竞争使用独占型资源，只能让它们逐个使用，这种现象称互斥，是一种竞争关系，也称间接的制的关系。另一种是进程之间协同完成任务，在关键点上等待另一进程发来的消息，以便协同一致，是一种协作关系，也称直接的制的关系。

关系：本质上，互斥是一种特殊的同步，因为它也是进程之间的执行次序上的一种协调。

（4）文件目录记录文件的管理和说明信息，如文件名字、长度、属性、外存位置等信息，用于对单个文件的控制。把同一个卷上的若干文件的文件目录组成一个单独的文件，由于它是由文件目录组成的，故称目录文件，它用于对整个文件系统的管理。

二、（未答到要点扣大部份分）

1．Unix存在两类进程：系统进程和用户进程。前者在核心态下执行OS代码，后者在用户态下执行用户程序。当用户进程因中断或系统调用进入内核态，此时发生了模式切换，但进程上下文切换没有发生。这时，系统进程开始执行，而这两个进程（用户/系统进）使用同一个PCB，实质是一个进程，仅执行程序不同、映射的物理地址空间不同、使用的堆栈不同OS例行程序和数据放在共享地址空间，可被所有用户共享。

2．（1）小内存中运行大程序。（2）程序的局部性原理。（3）部分装入和部分对换。主存/辅存独立编址统一使用。（4）动态重定位设施及快速磁盘。（5）计算机地址总线宽度。

3．可设计成高、低优先级两个队列。分时作业进入高优先级队列，采用短时间片的时间片轮转法调度。当高优先级队列空时，调度低优先级队列的批作业，并给予较长的时间片。

4．保护系统程序或用户程序不受其它用户程序的破坏，采用的方法是对计算机系统设置不同工作状态或者说处理器具有不同的工作模式，采用两态模式时常称：管态和目态。运行在管态下的程序比运行在目态下的程序有更多的访问权，例如，对内存的使用权和运行或其它用户程序的目的。有的计算机处理器可工作在多种状态下，也有的提供了保护环设施，都能更好地进行状态隔离。

在多道程序环境中，空间隔离能使每个用户进程被限制在分配的存储空间工作，如果一个进程错误地写信息到另一个进程的地址空间，会导致严重的后果。每个用户进程的内存空间可以通地虚拟存储技术来实现内存保护，提供有效的内存隔离，这时操作系统能确保每个页面或每个段只被其所属进程或授权进程访问。空间隔离技术能保证系统程序和用户程序的安全性。

三、答：

（1），Unix操作系统中的文件是多级索引的流式文件，一个1M的文件占用2048块，将该文件遍历一遍至少需要2048次I/O交换，即2048次结束中断。此外，2048块中，前10块的地址已在内存，但第11块至第138块的地址索引块需要读入，第139至2048块的二级索引块需要读入，共需3次I/O交换，因此，总共需要2051次磁盘完成中断。

（2），LRU算法：

3

2

2

3

2

3

1

2

5

1

2

5

1

2

5

4

2

5

4

2

5

4

2

5

2

3

5

2

3

5

2

3

F F F F F F F

中断率：7/12

FIFO算法：

3

2

2

3

2

3

1

2

3

1

5

3

1

5

2

1

5

2

4

5

2

4

5

2

4

3

5

2

3

5

2

3

F F F F F F F F F

中断率：9/12

Clock算法：

3\*

2\*

2\*

3\*

2\*

3\*

1\*

2\*

3

1

5\*

2\*

1

5\*

2\*

4\*

5\*

2\*

4\*

5\*

2

4

3\*

2

4

3\*

2\*

5\*

3\*

2\*

5\*

3\*

F F F F F F F F

中断率：8/12

在页面调度算法中，OPT算法是理论最优但无法实现，LRU算法性能几乎OPT一样，但是实现相当困难，系统开销非常大，此外，虽然FIFO算法简单易行，但是性能较差。相比上述算法，Clock算法是LRU算法的变种，通过为每一块附加一个附加位记录该内存块的使用情况，以较小的开销接近了LRU的性能，故此Clock算法相对而言应用比较广泛。

四、答：

（1），临界区管理原则有：a)一次只能一个进程进入临界区；b)等待进入临界区的时间必须是有限的；c)临界区空闲时，访问临界区的请求应立即响应；d)进程在临界区中的驻留时间必须有限。

（2）该题中的互斥现象有两点：第一，因为简易桥是单向通过，因此东西方向过桥的车辆对桥的使用是互斥的，第二，同方向上可能会同时有多辆车辆过桥，为了正确释放桥的使用，必须设置桥上车辆的计数器，对计数器的使用也是需要互斥的。该题中的同步现象有：当桥上已有4辆车辆时，同方向上的第5辆之间存在同步现象。

（3）据上分析，信号量应该有4个：S，初值为1，代表桥的互斥使用的信号量；Scounteast, 初值为1，代表由东向西方向行驶的桥上的车辆计数器的互斥使用；Scountwest,初值为1，代表由西向东方向行驶的桥上的车辆计数器的互斥使用；Scount4，初值为4，代表桥上车辆的计数信号量，用于同步管理。

（4）算法如下：

var S, Scounteast, Scountwest, Scount4;:semaphore;

S:=1;Scounteast:=1;Scountwest:=1;Scount4:=4;

Counteast, Countwest:integer;

Counteast:=0;Countwest:=0;

Cobegin

process east(i)

begin

P(Scounteast);

if Counteast=0 then P(S);

Counteast:=Counteast+1;

V(Scounteast);

P(Scount4);

上桥；过桥；下桥；

V（Scount4）;

P(Scounteast);

Counteast:=Counteast-1;

if Counteast=0 then V(S);

V(Scounteast);

end;

process west(i)

begin

P(Scountwest);

if Countwest=0 then P(S);

Countwest:=Countwest+1;

V(Scountwest);

P(Scount4);

上桥；过桥；下桥；

V(Scount4);

P(Scountwest);

Countwest:=Countwest-1;

if Countwest=0 then V(S);

V(Scountwest);

end;

coend.

解题的基本思路如下：

●关于桥的互斥使用：互斥管理只发生在：同方向上的第一辆车在上桥前需要查看桥的状态是否为“可用”，此时，可以使用P(S) 操作；类似，同方向上的最后一辆车下桥时，必须释放桥的使用，此时，可以使用V(S)操作。

●为判断桥上车辆数目，必须设置计数器Counteast和Countwest.，而上述两变量则是为新引入的共享变量，必须互斥使用，因此，设置两个信号量Scounteast和Scountwest，使用PV操作进行管理。

●桥的最大负荷为4辆，实际上类似于“读者—写者问题”中共享有四个缓冲区的缓冲池，这是一种同步管理。根据题意，同步管理应该设置在“上桥”和“下桥”时，否则会出现等待车辆可能没有及时过桥的错误。

**南大OS 2004年参考答案**

一、（16分）

1．（6分）同步/互斥：互斥是指若干个进程要使用一个共享资源时，任何时刻最多允许一个进程去使用，其它进程必须等待，直到占有者释放该资源。

同步是指两个以上的进程基于某个条件来协调他们的活动。互斥是同步的特例。

死锁：一组进程处于死锁状态是指如果在一个进程集合中的每个进程都在等待只能由该集合中的其他一个进程才能引发的事件，则称该组进程此刻发生了死锁。

Redundant Array of Independent Disks，独立磁盘冗余阵列。一种大容量外存储系统。策略是用一组较小容量、独立、可并行工作的磁盘组成存储阵列，借助冗余技术，实现数据的多组织、分布存储，从而能够并行处理单个或多个I/O请求，提高系统性能和效率。

2．（5分）答：从实现的角度看，线程可以分为用户级、内核级和混和级三种。

其中：

（1）用户级线程基于用户空间运行的线程库支撑，线程的管理由应用程序来进行。这种实现方法的优点是线程调度开销较小、调度算法可以应用相关，灵活性强。缺点是同一进程内的某个线程被阻塞后，该进程将被完全阻塞，降低了并行度。

（2）内核级线程基于内核提供的API，线程的管理工作由内核进行。这种实现方式的优点是线程调度在内核进行，线程并发度大大提高，缺点在于调度需要进行系统模态转换，切换开销较大。

（3）混和级线程同时提供了上述两种支撑方法，在设计得当的前提下，可以结合上述两种方式的优点、摒弃它们的缺点。

3．（5分）答：加密技术是一种将明文转化为密文米保护暴露在未受保护介质上的原文的技术，其基本原理是定义一个加密函数encrypt和一个解密函数decrypt:

decrypt(key’,encrypt(Key,plain text))=plain text

其中，key是加密密钥，key’是解密密钥。加密机制的一般模型是：

明文P

密文P

明文P

加密密钥key

解密密钥key

加密 解密算解 解密算

算法E 法D

二、(12分)**(本题的详细分析、解答见模拟考试题)**

三、（12分）

该题中描述的指挥官和士兵的关系本质上是一个生产者与三个消费者问题。无线电传输过程可以理解为一个存放指令的缓冲区，只是该“缓冲区”只要写一次，但要读三次。因为signal不是最后一个操作，必须用Hoare方法求解，题解如下：

TYPE BattleCommand=monitor

var SS\_A,SS\_B,SS\_C,Sa,Sb,Sc:semaphore;

SS\_A-count,SS\_B\_count.SS\_C\_count:integer;

Sa\_count,Sb\_count,Sc\_count:integer;

soldierA, soldierB,soldierC:boolean;

buffer:instruction;

procedure sendCommand

begin

if soldierA then wait(Sa,Sa\_count,IM);

if soldierB then wait(Sb,Sb\_count,IM);

if soldierC then wait(Sc,Sc\_count,IM);

Buffer:=instruction;

SoldierA:=soldierB:=soldierC:=true;

Signal(SS\_A,SS\_A\_count,IM);

signal(SS\_B,SS\_B\_count,IM);

signal(SS\_C,SS\_C\_count,IM);

end;

procedure ACK\_SoldierA

begin

if soldierA=false then wait(SS\_A,SS\_A\_count,IM);

else soldierA:=false;

take the instruction from buffer;

signal(Sa,Sa\_count,IM);

end;

procedure ACK\_SoldierB

begin

if soldieB=false then wait(SS\_B,SS\_B\_count,IM);

else soldierB:=false;

take the instruction from buffer;

signal(Sb,Sb\_count,IM);

end

procedure ACK\_SoldierC

begin

if soldierC=false then wait(SS\_C,SS\_C\_count,IM);

else soldierC:=false;

take the command from buffer;

signal(Sc,Sc\_count,IM);

end

begin

soldierA:=soldierB:=soldierC:=false;

end

cobegin

｛ process commander

begin

repeat

produce a instruction;

P(IM.mutex);

Call BattleCommand.sendCommand();

If IM.next-count>0 then V(IM.next);

Else V(IM>mutex);

…

until false;

end

process soldierA

begin

repeat

P(IM.mutex);

Call Battlecommand.ACK\_SoldierA();

If IM.next-count>0 then V(IM.next);

Else V(IM>mutex);

…

until false;

end

process soldierAB

begin

repeat

P(IM.mutex);

Call Battlecommand.ACK\_SoldierB();

If IM.next-count>0 then V(IM.next);

Else V(IM>mutex);

…

until false;

end

process soldierC

begin

repeat

P(IM.mutex);

call Battlecommand.ACK\_SoldierC();

If IM.next-count>0 then V(IM.next);

Else V(IM>mutex);

…

until false;

end

}

coend.