### 第三章

作者：佚名 来源：网络

1、 有三个并发进程：R 负责从输入设备读入信息块，M 负责对信息块加工处理；P 负责打印输出信息块。今提供；

l ）一个缓冲区，可放置 K 个信息块；

2 ）二个缓冲区，每个可放置 K 个信息块；

试用信号量和P 、V 操作写出三个进程正确工作的流程。答：

1. ) var B : array [ 0 , k-1 ] of item ; sread : semaPhore : = k ;

smanage : semaPhore : = 0 ; swrite : semaphore : = 0 ; rptr : integer : = O ; mptr : integer : = O ; wptr ：integer : = 0 ;

x : item cobegin

process reader ; process manager ; process writer ; begin begin begin

LI : read a message intox ; L2 : P ( smanage ) ; L3 : P ( swnte ) ; P ( sread ) ; x:=B[mptr]; x:=B[swrite];

B[rptr]:=x; mptr:=(mptr+1) mod k; wptr:=(wptr+1) mod k; Rptr:=(rptr+1) mod k; manage the message in x; V(sread); V(smanage); B[mptr]:=x; print the message in x;

Goto L1; V(swrite); goto L3; End; goto L2; end;

End; coend

1. ) var A , B :array [ 0 , k -l ] of item ; sPut1 : semaphore:=k;

SPut2: semaPhore:=k; sget1 : semaPhore : = 0 ; sget2 : semaphore : = 0 ; put1 ：integer ：=O ; put2：integer : = 0 ; get1 ：integer ：=O ; get2 : integer : = O ; cobegin

process reader ; processn manager; process Writer ; begin begin begin

Ll : read a message into x ; L2 : P ( sgetl ) ; L3 : P ( sgetZ ) ; P ( SPut1 ) ; x : = A [ get1] ; x : = B [get2];

A [put1]:=x ; get1 ：(get1+1 ) mod k ; get2:=（get2 + l ) mod k ; Put1:=(put1+1) mod k; V(sput1); V(sput2);

V(sget1); manage the message into x; print the message in x; Goto L1; P(sput2); goto L3;

Put2:=(put2+1) mod k; V(sget2);

Goto L2; End; Coend

2 设有n 个进程共享一个互斥段，如果： ( 1 ）每次只允许一个进程进入互斥段；

( 2 ）每次最多允许m 个进程（m 簇n ）同时进入互斥段。

试问：所采用的信号量初值是否相同？信号量值的变化范围如何？ 答：所采用的互斥信号量初值不同。

1. ）互斥信号量初值为 1 ，变化范围为［-n＋l , 1 ］。

当没有进程进入互斥段时，信号量值为 1 ；当有 1 个进程进入互斥段但没有进程等待进入互斥段时，信号量值为 O ；当有 1 个进程进入互斥段且有一个进程等待进入互斥段时，信号量值为-1 ； 最多可能有n -1 个进程等待进入互斥段，故此时信号量的值应为-（n - 1 ）也就是-n+1 。

1. ）互斥信号量初值为 m ，变化范围为［-n＋m , m ］。

当没有进程进入互斥段时，信号量值为 m ；当有 1 个进程进入互斥段但没有进程等待进入互斥段时，信号量值为m - 1 ：当有m 个进程进入互斥段且没有一个进程等待进入互斥段时，信号量值为 0 ：当有m 个进程进入互斥段且有一个进程等待进入互斥段时，信号量值为一 l ；最多可能有n - m 个进程等待进入互斥段，故此时信号量的值应为-(n-m)也就是-n+m.

1. 有两个优先级相同的进程 P1 和P2，各自执行的操作如下，信号量 S1 和S2 初值均为 0。试问Pl 、P2 并发执行后，x 、y 、z 的值各为多少？

P1: P2:

Begin begin Y:=1; x:=1; Y:=y+3; x:=x+5; V(S1); P(S1); Z:=Y+1; X:X+Y; P(s2); V(S2);

Y:=z+y; z:=z+x;

End end

答：现对进程语句进行编号，以方便描述． P1 : P2 :

begin begin

y : = 1 ；① x :=1 ; ⑤ y :=y+3 ；② x ：x+5 ; ⑥ V(S1); P(S1);

Z:Y+1 ；③ x ：X＋Y ;⑦ P(s2); V(S2);

Y:=z+y; ④ z：=Z+X；⑧ End end

① 、② 、⑤ 和⑥ 是不相交语句，可以任何次序交错执行，而结果是唯一的。接着无论系统如何调度进程并发执行，当执行到语句⑦ 时，可以得到 x = 10 , y = 4 。按Bernstein 条件，语句③ 的执行结果不受语句⑦ 的影响，故语句③ 执行后得到z = 5 。最后，语句④ 和⑧ 并发执行，这时得到了两种结果为：

语句④ 先执行：x =10 , y =9 , z= 150

语句⑧ 先执行：x =10 , y =19 , z =15

此外，还有第三种情况，语句③ 被推迟，直至语句⑧ 后再执行，于是依次执行以下三个语句： 7 ：二z + X :

z : = y + 1 ;

y : ＝Z 十y ;

这时z 的值只可能是 y ＋1=5 ，故y =Z＋Y=5 + 4=9，而x = 10 。

第三种情况为：x = 10 ，Y=9 , Z = 5 。

1. 有一阅览室，读者进入时必须先在一张登记表上登记，该表为每一座位列出一个表目，包括座 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：l ）信号量和 P 、V 操作；2 ）管程，来实现用户进程的同步算法。

答：1 ）使用信号量和 P 、v 操作： var name ：array [ l „100]of A ;

A = record

number ：integer ; name：string ; end

for i : = 1 to 100 do {A [ i ].number ：i；A [ i ].name :null;} mutex , seatcount : semaphore ;

i : integer ；mutex : = l ; seatcount : = 100 ; cobegin

{

process readeri ( var readename：string ) （i=1 , 2 „)

{

P ( seatcount ) ; P （mutex ) ;

for i : = 1 to 100 do i++

if A [ i ].name＝null then A [ i ].name：readername； reader get the seat number=i；/\*A[I].number

V ( mutex )

进入阅览室，座位号 i ，座下读书； P ( mutex ) ;

A[i]name：null ; V （mutex ) ; V(seatcount);

离开阅览室；

}

}

coend

2 ）使用管程操作： TYPE readbook=monitor VAR R: condition ; I,seatcount ：integer;

name：array [ l:100] of string ; DEFINE rcadercome, readerleave ;

USE check , wait , signal , release ; Procedure readercome ( readername ) begin

check ( IM ) ;

if seatcount≥100 wait ( R,IM ) seatcount : = seatcount + 1 ;

for i=1 to 100 do i++

if name[i] ==null then name[i]:= readername; get the seat number = i ;

release ( IM ) ; end

procedure readerleave ( readername ) begin

check ( IM ) ; seatcount--;

for i = 1 to 1 00 do i++

if name［i ］readername then name［i］:null; release ( IM ) ;

end begin

seatcount : = 1OO ; name:＝null ; end

cobegin

{

process readeri ( i = 1 , 2 ．„ ） begin

readercome ( readername）; read the book ;

readerleave ( readername）; leave the readroom;

end

}

coend.

5. 在一个盒子里，混装了数量相等的黑白围棋子· 现在用自动分拣系统把黑子、白子分开，设分拣系统有二个进程 P1 和P2 ，其中P1 拣白子；P2 拣黑子。规定每个进程每次拣一子；当一个进程在拣时，不允许另一个进程去拣；当一个进程拣了一子时，必须让另一个进程去拣．试写 出两进程P1 和P2 能并发正确执行的程序。

答 1 ：实质上是两个进程的同步问题，设信号量 s1 和s2 分别表示可拣白子和黑子，不失一般性，若令先拣白子。

var S1 , S2 : semaphore; S1 : = l; S2 ：=0;

cobegin

{

process P1 begin repeat

P( S1 ) ;

拣白子

V ( S2 ) ;

until false ;

end process P2 begin repeat

P ( S2 ) ;

拣黑子

V (S1 ) ;

until false ; end

}

coend . 答 2 :

TYPE pickup-chess = MONITOR VAR flag : boolean ;

S-black , s-white : codition ; DEFINE pickup-black , pickup-white ; USE wait,signal , check , release ; procedure pickup-black ;

begin check(IM ) ;

if flag then wait(s-black,IM ) ; flag : ＝true;

pickup a black; signal(S-white,IM); release ( IM ) ; end

procedure pickup-white ; begin

check ( IM ) ;

if not flag then wait(S-white,IM ); flag :=false ;

pickup a white ; signal ( S-black,IM ) ; release ( IM ) ;

end begin

flag:=true ; end

main ( )

{ cobegin process -B ( ) ;

process -W ( ) ; coend

}

process-B ( ) begin

pickup-chess.pickup-black ( ) ; other ;

end

process-W ( ) begin

pickup-chess.pickup-white( ) ; other ;

end

1. 管程的同步机制使用条件变量和 wait 及signal ，尝试为管程设计一种仅仅使用一个原语操作的同步机制。

答：可以采用形如 waituntil ＜条件表达式＞的同步原语。如 waituntil ( numbersum + number

< K ) 表示进程由于条件不满足而应等待，当进程号累加和小于 K 时，系统应唤醒该进程工作．

1. 设公共汽车上，司机和售票员的活动分别如下： 司机的活动：启动车辆：正常行车；到站停车。售票员的活动：关车门；售票；开车门。

在汽车不断地到站、停车、行驶过程中，这两个活动有什么同步关系？用信号量和 P 、V 操作实现它们的同步。

答：在汽车行驶过程中，司机活动与售票员活动之间的同步关系为：售票员关车门后，向司机发 开车信号，司机接到开车信号后启动车辆，在汽车正常行驶过程中售票员售票，到站时司机停车， 售票员在车停后开门让乘客上下车。因此，司机启动车辆的动作必须与售票员关车门的动作取得 同步；售票员开车门的动作也必须与司机停车取得同步。应设置两个信号量：S1 、S2 ;S1 表示是否允许司机启动汽车（其初值为 0 ) ;S2 表示是否允许售票员开门（其初值为 0 ）。用P 、v 原语描述如下：

var S1 , S2 : semaphore ;

S1=0；S2=0；

cobegin

{

driver ( ) ;

busman ( ) ;

}

coend driver ( ) begin

while ( 1 ) { P ( S1 )

启动车辆；正常行车；到站停车； V ( S2 ) ;

}

end busman ( ) begin

while ( 1 ) { 关车门；

V ( 51 )

售票;

P ( S2 )

开车门； 上下乘客；

}

end

8、一个快餐厅有 4 类职员：( l ）领班：接受顾客点菜；( 2 ）厨师：准备顾客的饭菜；( 3 )

包工：将做好的饭菜打包；( 4 ）出纳员：收款并提交食品。每个职员可被看作一个进程，试用一种同步机制写出能让四类职员正确并发运行的程序。

答：典型的进程同步问题，可设四个信号量 51 、S2 、S3 和S4 来协调进程工作。var S1 , S2 ,S3 , S4 : semaphore ;

S1 : = 1 ;S2 ：=S3 : = S4 : = 0 ;

cobegin

{ process P1 begin repeat

有顾客到来； P ( S1 )；

接受顾客点菜； V ( 52 )；

untile false； end

process P2 begin repeat

P (S2 ) ;

准备顾客的饭菜； v ( S3 ) ;

untile false ; end

process P3 begin repeat

P (S3 ) ;

将做好的饭菜打包； V ( S4 ) ;

untile false ; end

process P4

begin repeat P( 54 ) ;

收款并提交食品；V ( 51 ) ; ufltile false ;

end

}

coend .

9、在信号量S 上作P 、v 操作时，S 的值发生变化，当 S> 0、S=0、S< 0 时，它们的的物理意义是什么？

答：S 的值表示它代表的物理资源的使用状态：S > 0 表示还有共享资源可供使用。S 阅表示共享资源正被进程使用但没有进程等待使用资源。S < 0 表示资源已被分配完，还有进程等待使用资源。

10 ( 1 ）两个并发进程并发执行，其中，A 、B 、C 、D 、E 是原语，试给出可能的并发执行路径。

Process P Process Q begin begin

1. ; D ;
2. ; E ;
3. ; end :

end ;

( 2 ）两个并发进程P1 和P2 并发执行，它们的程序分别如下： P 1 P2

repeat repeat k:=k×2 ; print k ; k:=k+1 ; k:=0 ;

until false ; until false ;

若令k 的初值为 5 ，让 P1 先执行两个循环，然后，P1 和P2 又并发执行了一个循环，写出可能的打印值，指出与时间有关的错误。

答：

( 1 ）共有 10 种交错执行的路径：

A 、B 、C 、D 、E; A 、B 、D 、E 、C; A 、B 、D 、C 、E ; A 、D 、B 、E 、C; A 、D 、B 、C 、E; A 、D 、E 、B 、C ;

D 、A 、B 、E 、C; D 、A 、B 、C 、E; D 、A 、E 、B 、C ; D 、E 、A 、B 、C 。( 2 ）把语句编号，以便于描述：

P1 P2

repeat repeat

k:=k×2 ；① printk ；③ k:=k+l ；② k:=0 ；④ until false ; until false ;

l ) K 的初值为 5 ，故P1 执行两个循环后，K = 23 。

2 ）语句并发执行有以下情况：

① 、② 、③ 、④ ，这时的打印值为：47

③ 、④ 、① 、② ，这时的打印值为：23

① 、③ 、② 、④ ，这时的打印值为：46

① 、③ 、④ 、② ，这时的打印值为：46

③ 、① 、② 、④ ，这时的打印值为：23

③ 、① 、④ 、② ，这时的打印值为：23

由于进程P1 和P2 并发执行，共享了变量 K ，故产生了‘结果不唯一’。

1. 证明信号量与管程的功能是等价的： ( l ）用信号量实现管程；

( 2 ）用管程实现信号量。

答：( 1 ）用信号量实现管程；

Hoare 是用信号量实现管程的一个例子，详见课文内容。下面介绍另一种简单方法：每一个管程都对应一个mutex ，其初值为 1 ，用来控制进程互斥调用管程。再设一个初值为 0 的信号量， 用来阻塞等待资源的进程。相应的用信号量实现的管程库过程为：

Var mutex,c:semaphore ; mutex:=1 ; c:=0 ;

void enter-monitor ( ) /\*进入管程代码，保证互斥P ( mutex ) ;

}

void leave-monitor-normally ( )/\*不发信号退出管程

{

V ( mutex ) ;

}

void leave-with-sigal(c) /\*在条件c 上发信号并退出管程，释放一个等待 c 条件的进程。｛注意这时没有开放管程，因为刚刚被释放的进程己在管程中。

V ( c ) ;

}

void wait(c) /\*等待条件c ，开放管程

{

V ( mutex ) ;

P (c) ;

}

( 2 ）用管程实现信号量。TYPE semaphore=monitor VAR S ; condition ; C:integer ;

DEFINE P , V ;

USE check , wait , signal , release ; procedure P

begin

check ( IM ) ; C:= C-1 :

if C < 0 then wait ( S,IM ) ; release ( IM ) ;

end

procedure V begin

check ( IM ) :

C : = C + 1 ;

if C≤0 then signal ( S,IM ) ; release ( IM ) ;

end begin C:=初值; End.

1. 证明消息传递与管程的功能是等价的： ( 1 ）用消息传递实现管程；

( 2 ）用管程实现消息传递。

答：( 1 ）用消息传递实现管程；

用消息传递可以实现信号量（见 13 ( 2 ) ) ，用信号量可以实现管程（见 11 (1 ) ) ，那么， 把两种方法结合起来，就可以用用消息传递实现管程。

( 2 ）用管程实现消息传递。TYPE mailbox=monitor

VAR r , k , count:integer ;

buffer ：array[0„n-1] of message ; full , empty:condition ;

DEFINE add , get ;

USE check , wait , signal , release ; procedure add ( r ) ;

begin

check ( IM ) ;

if count=n then wait ( full,IM ) ; buffer [r]:=message ;

r:＝(r+1) mod n count:=count + 1 ;

if count = 1 then sighal ( empty , IM ) ; release ( IM ) ;

end

procedure get ( m ) ; begin

check ( IM ) ;

if count = 0 then wait ( empty , IM ) ; m:=buffer [ k 」；

count : = count-1 ;

if count＝n-1 then signal ( full , IM ) ; release ( IM ) ;

end

begin

r:= 0 ; k:= 0 ; count:=0 ; end

1. 证明信号量与消息传递是等价的： ( 1 ）用信号量实现消息传递；

( 2 ）用消息传递实现信号量。

答：( l ）用信号量实现消息传递；

1. ）把消息队列组织成一个共享队列，用一个互斥信号量管理对该队列的入队操作和出队操作.
2. ）发送消息是一个入队操作，当队列存储区满时，设计一个同步信号量阻塞 send 操作。
3. ）接收消息是一个出队操作，当队列存储区空时，设计另一个同步信号量阻塞 receive 操作。( 2 ）用消息传递实现信号量。

l ）为每一个信号量建立一个同步管理进程，它包含了一个计数器，记录信号量值；还为此信号量设立一个等待进程队列

1. ）应用进程执行 P 或V 操作时，将会调用相应 P 、V 库过程。库过程的功能是：把应用进程封锁起来，所执行的 P 、V 操作的信息组织成消息，执行 send 发送给与信号量对应的同步管理进程，之后，再执行 receive 操作以接收同步管理进程的应答。
2. ）当消息到达后，同步管理进程计数并查看信号量状态。如果信号量的值为负的话，执行 P 操作的应用进程被阻塞，挂到等待进程队列，所以，不再要送回答消息。此后，当 V 操作执行完后， 同步管理进程将从信号量相应队列中选取一个进程唤醒，并回送一个应答消息。正常情况下，同 步管理进程回送一个空应答消息，然后，解锁执行 P 、V 操作的应用程序。
3. 使用（1）消息传递，( 2 ）管程，实现生产者和消费者问题。答：( 1 ）见课文 ch3 3.5.4 节。

（2 ）见课文Ch3 3.4.3 节。

1. 试利用记录型信号量和 P 、V 操作写出一个不会出现死锁的五个哲学家进餐问题的算法。答： var forki:array [0„4] of semaphore ;

forki:=1 ; cobegin

{

process Pi /\* i = 0 , 1 , 2 , 3 \*/ begin

L1 :

思考：

P(fork[i]) ; / \* i =4,P(fork [0]) \* / P(fork[i+1] mod 5) / \* i =4P（fork [4]）\* / 吃通心面；

V (fork[i] ;

V (fork([i+1] mod 5 ) ; goto L1 ;

end ;

}

coend ;

1. Dijkstra 临界区软件算法描述如下：

var flag ：array[0„n] of (idle,want-in ，in\_cs ) ; turn:integer ; tune:0 or 1 or „ or , n-1 ; process Pi(i=0,1，„,n-1)

var j ; integer ; begin

repeat repeat

flag [i] :want\_in ; while turn≠1 do

if flag[turn]==idle then turn:=i ; flag[i]:= ip\_cs ;

j:=0 ;

while (j < n ) & (j==1 or flag[j] ≠in\_cs ) do j:=j + 1 ;

until j≥n :

critical section ; flag [i]:=idle ;

„„

until false ; end .

试说明该算法满足临界区原则。

答：为方便描述，把 Dijkstra 程序的语句进行编号： repeat

flag[i]:=want\_in ；① while turn≠i do ②

if flag[trun]==idle then turn:=i ；③ flag[i]: = in\_cs ；④

j:= O ;

while(j < n ) & (j==1 or flag[j] ≠in\_cs ）⑤ do j:=j + 1 ; @

until j≥n ; critical section ; flag[i] :=idle ；⑦

„

( l ）满足互斥条件

当所有的巧都不在临界区中，满足 flag[j]≠in\_cs（对于所有j , j≠i ）条件时，Pi 才能进入它的临界区，而且进程 Pi 不会改变除自己外的其他进程所对应的 flag[j]的值。另外，进程 Pi 总是先置自己的flag[j]为in\_cs 后，才去判别 Pj 进程的flag[j]的值是否等于in\_cs 所以，此算法能保证n 个进程互斥地进入临界区。

( 2 ）不会发生无休止等待进入临界区

由于任何一个进程 Pi 在执行进入临界区代码时先执行语句① ，其相应的flag[i]的值不会是idle 。注意到flag[i]＝in\_cs 并不意味着turn 的值一定等于i 。我们来看以下情况，不失一般性，令turn 的初值为 0，且 P0 不工作，所以，flag[turn]=flag[0]=idle。但是若干个其他进程是可能同时交替执行的，假设让进程 Pj(j=l , 2 , „n-l）交错执行语句① 后（这时flag[j]=want\_in），再做语句② （第一个while 语句），来查询flag[turn]的状态。显然，都满足turn≠i ，所以，都可以执行语句③ ，让自己的turn 为j 。但 turn 仅有一个值，该值为最后一个执行此赋值语句的进程号，设为 k 、即turn=k (1≤k≤n -1 ）。接着，进程

Pj(j=1,2,„n-l ) 交错执行语句④ ，于是最多同时可能有 n-1 个进程处于in\_cs 状态，但不要忘了仅有一个进程能成功执行语句④ ，将加m 置为自己的值。

假设｛P1 , P2 ，„ Pm ｝是一个己将flag[i] 置为in\_cs ( i =1,2,„,m ) ( m ≤n -1）的进程集合，并且已经假设当前 turn=k ( 1≤k≤m ) ，则Pk 必将在有限时间内首先进入临界区。因为集合中除了Pk 之外的所有其他进程终将从它们执行的语句⑤（第二个while 循环语句）退出， 且这时的j 值必小于n ，故内嵌 until 起作用，返回到起始语句① 重新执行，再次置 flag [ i ]

= want\_in ，继续第二轮循环，这时的情况不同了，flag[turn] =flag[ k] 必定≠idle （而为in\_cs ）。而进程Pk 发现最终除自身外的所有进程 Pj 的flag[j]≠in\_cs ，并据此可进入其临界区。

1. 另一个经典同步问题：吸烟者问题(patil , 1971 ）。三个吸烟者在一个房间内，还有一个香烟供应者。为了制造并抽掉香烟，每个吸烟者需要三样东西：烟草、纸和火柴，供应者有丰富货 物提供。三个吸烟者中，第一个有自己的烟草，第二个有自己的纸和第三个有自己的火柴。供应 者随机地将两样东西放在桌子上，允许一个吸烟者进行对健康不利的吸烟。当吸烟者完成吸烟后 唤醒供应者，供应者再把两样东西放在桌子上，唤醒另一个吸烟者。试采用：( 1 ）信号量和 P 、v 操作，( 2 ）管程编写他们同步工作的程序。答：( 1 ）用信号量和P 、v 操作。

vars , S1 ,S2 , S3 ; semaphore ; S:=1 ; S1:=S2:=S3:=0 ;

fiag1 , flag2 , fiag3 : Boolean ; fiag1:=flag2:=flag3:=true; cobegin

{

process 供应者begin

repeat P(S) ;

取两样香烟原料放桌上，由 flagi 标记； / \* nago1 、nage2 、nage3 代表烟草、纸、火柴if flag2 & flag3 then V(S1) ; / ＊供纸和火柴

else if flag1 & fiag3 then V(S2 ) ; / ＊供烟草和火柴else V(S3) ; / ＊供烟草和纸

untile false ; end

process 吸烟者 1 begin

repeat P(S1) ;

取原料； 做香烟； V(S) ;

吸 香 烟 ； untile false ;

process 吸烟者 2 begin

repeat

P (S2 ) ;

取原料； 做香烟； V(S) ;

吸 香 烟 ； untile false ;

process 吸烟者 3 begin

repeat

P (S3 ) ;

取原料； 做香烟； V ( S ) ;

吸 香 烟 ； untile false ; coend .

( 3 ）用管程。

TYPE mskesmoke=moonitor

VAR S, S1 ,S2 ,S3 : condition ; flag1 , flag2, flag3 : boolean DEFINE give , take1 , take2 , take3 ; USE check , wait , signal , release ; procedure give

begin

check ( IM ) ; 准备香烟原料；

if 桌上有香烟原料then wait( S , IM ) ; 把准备的香烟原料放桌上； if fiag2 & flag3 then signal ( S1 ,IM）;

if flag1 & flag3 then signal ( S2 ,IM ) ; else signal (S3 , IM ) ; release ( IM ) ;

end

procedure take1 begin check(IM):

if 桌上没有香烟原料then wait ( S1 ,IM）; else 取原料；

signal ( S , IM ) ; release ( IM ) ; end

procedure take2 begin

check ( IM ) :

if 桌上没有香烟原料 then wait(S2,IM); else 取原料；

signal ( S , IM ) ;

release （IM）; end

procedure take3 begin

check ( IM ) :

if 桌上没有香烟原料then wait(S3,IM); else 取原料

signal ( S ,IM ) ; release ( IM ) ; end

begin flag1:=flag2:=flag3:=true; end.

cobegin

{

process 供应者begin

repeat

Call makesmoke.give();

„„

until false ; end

process 吸烟者 1 begin

repeat

Call makesmoke.take1() ; 做香烟，吸香烟；

until false ; end

process 吸烟者 2 begin

repeat

Call makesmoke.take2() ; 做香烟，吸香烟；

until false ; end

process 吸烟者 3 begin

repeat

Call makesmke.take3(); 做香烟，吸香烟；

until false ; end

}

coend .

18、 如图所示，四个进程 Pi （i=0„ 3 ）和四个信箱Mj (j=0„ 3 ) ，进程间借助相邻信箱传递消息，即Pi 每次从Mi 中取一条消息，经加工后送入 M(i + 1) mod4 ，其中M0 、M1 、M2 、M3 ;可存放 3 、3 、2 、2 个消息。初始状态下，MO 装了三条消息，其余为空。试以 P 、V 为操作工具，写出Pi（i=0„3）的同步工作算法

答：

var mutexl , mutexZ , mutex3 ，mutex0 :semaphore; Mutex1＝nutex2:=mutex3:=mutex0:=1; Empty0,empty1,empty2, empty3; semaphore;

empty:=0 ; empty1:=3 ; empty:=2:=empty3:=2; full0 , full1 , full2 , full3:semphore ; full0:=3;full1:=full2:=full3:=0; in0,in1,in2,in3,out0 ,out2,out3,;intger; in0:=in1:＝in2:＝in3:=out0:=out1:=out2:=out3:=0; cobegin

{

process P0 begin repeat P(full0); P(mutex0);

从M0[out0]取一条消息； out0:=(out0+1) mod 3 ; V(mutex0);

V(empty0) ; 加工消息； P(empty1) ; P(mutex1) ;

消息已M1[in1];

In1:=(in1+1) mod 3;

V(mutex1) ;

V(full1 ) ;

untile false ; end

process P1 begin repeat

P ( full1 ) ; P ( mutex1 ) ;

从M1[out1]取一条消息； Out1:=(out1+1) mod 3 ; V(mutex1);

V(empty1);

加 工 消 息 ; P(empty2); P(mutex2 ) ; 消息己M2[in2];

In2:=(in2+1) mod 2; V(mutex2 ) ;

v ( full2 ) ; untile false ; end

process P2 begin repeat P(full2) ; P(mutex2 ) ;

从M2[out2]取一条消息； out2:=(out2 + l ) mod 2; V(mutex2) ;

V(empty2) ; 加工消息； P(empty3) ; P(mutex3) ;

消 息 己 M3[in3]; in3:=(in3+1) mod 2 ; V(mutex3) ; V(full3) ;

untile false ; end

process P3 begin repeat P(full3) ; P(mutex3) ;

从M3[out3] 取一条消息; out3:=(out3+1)mod 2;

V (mutex3) ; V (empty3) ; 加工消息；

P ( empty0 ) ; P ( mutex0 ) ; 消息己MO[in0];

In0:=(in0+1) mod 3 ; V(mutex0) ;

V(full0) ; untile false ; end

{

coend

19、有三组进程Pi 、Qj、Rk ，其中Pi 、Qj 构成一对生产者和消费者，共享一个由 M1 个缓区构成的循环缓冲池 buf1 。Qj、Rk 凡构成另一对生产者和消费者，共享一个由 M2 个缓冲区构成的循环缓冲池buf2 。如果 Pi 每次生产一个产品投入 buf1,Qj 每次从中取两个产品组装成一个后并投入buf2，Rk 每次从中取三个产品包装出厂. 试用信号量和 P 、V 操作写出它们同步工作的程序。

答：

var mutex1 , mutex2 , mutex3 : semaphore; empty1 , empty2 , full1 , full2 ; semaphore ;

in1 , in2 , out1 , out2 : integer ; counter1 , counter2:integer ; buffer1:array[0„M1-1] of item ; buffer2:array[0„M2-1]of item ; empty1:=M1 ; empty:=M2;

in1 : = in2 :=out1:=out2:=0 ; counter1:=counter2:=0 ; fun1:=full2:＝mutex1:=mutex2:=mutex3:=1;

cobegin

{

process Pi begin

L1:

P(empty1) ; P(mutex1 ) ;

put an item into buffer [in1] ; in1:=(in1+1) mod M1 ; counter++;

if counter1 = 2 then { counter1:=0;V(full1);} V(mutex) ;

goto L1; end

process Qj begin

L2:

P ( full2) ; P ( mutex1 ) ;

take an item from buffer1[out1]; out1:=(out1+1) mod M1;

take an item from buffer1[out1] ; out1:=(out1 + 1) mod M1 ;

V ( mutex1 ) ; V ( empty1 ) ;

V ( empty1 ) ; Process the products ; P ( emPty2) ;

P ( mutex2 ) ;

put an item into buffer2 [ in2 ] ; in2:=( in2 + l ) mod M2 ; counter2 + + ;

if counter2 = 3 then { counter2:=0 ;V( full2 ) ; } V ( mutex2) ;

goto L2 ; process Rk begin L3 :

P ( full2 ) ; P ( mutex2 ) ;

take an item from buffer2 [out2]; out2: = ( out2 + 1 ) mod M2 ; take an item from buffer2 [out2] ; out2:=( out2 + 1) mod M2 ;

take an item from buffer2 [out2]; out2:=(out2 + 1 ) mod M2 ;

v ( mutex2 ) ; V ( empty2 ) ; V ( empty2 ) ; V ( empty2 ) ;

packet the products ; goto L3 ;

end

}

coend

1. 在一个实时系统中，有两个进程 P 和Q ，它们循环工作。P 每隔 1 秒由脉冲寄存器获得输入， 并把它累计到整型变量 W 上，同时清除脉冲寄存器。Q 每隔 1 小时输出这个整型变量的内容并将它复位。系统提供了标准例程创 PUT 和OUT 卫UT 供拍，提供了延时系统调用 Delay ( seconds ）。试写出两个并发进程循环工作的算法。

答：

Var W ,V:integer; Mutex:semaphore; W:=0 ; V:=0 ;mutex:1;

cobegin { process P begin repeat P(mutex) ;

delay (1) ; V＝INPUT ;

W:=W + V ;

清除脉冲寄存器；

V (mutex) ; untile false ; end

process Q begin repeat

P ( mutex ) ; delay ( 60 ) ; OUTPUT ( W ) ; W : = 0 ;

V ( mutex ) ; untile false ;

}

coend .

1. 系统有同类资源m 个，被n 个进程共享，问：当 m > n 和m≤n 时，每个进程最多可以请求多少个这类资源时，使系统一定不会发生死锁？

答：当m≤n 时，每个进程最多请求 1 个这类资源时，系统一定不会发生死锁。当 m > n 时，如果m/n 不整除，每个进程最多可以请求”商＋1 ”个这类资源，否则为”商”个资源，使系统一定不会发生死锁？

1. N 个进程共享M 个资源，每个进程一次只能申请释放一个资源，每个进程最多需要 M 个资源， 所有进程总共的资源需求少于 M+N 个，证明该系统此时不会产生死锁。

答卜设max ( i ）表示第i 个进程的最大资源需求量，need ( i ）表示第i 个进程还需要的资源量，alloc ( i ）表示第i 个进程已分配的资源量。由题中所给条件可知：

max ( 1 ）+„+max( n ) = ( need (1)+„+need( n ))+((alloc(1)+„+alloc(n))<m+n

如果在这个系统中发生了死锁，那么一方面 m 个资源应该全部分配出去，alloc (1) ＋„+alloc ( n ）=m

另一方面所有进程将陷入无限等待状态。可以推出need(1)+„+need (n)< n

上式表示死锁发生后，n 个进程还需要的资源量之和小于n ，这意味着此刻至少存在一个进程 i , need ( i ) = 0 ，即它已获得了所需要的全部资源。既然该进程已获得了它所需要的全部资源， 那么它就能执行完成并释放它占有的资源，这与前面的假设矛盾，从而证明在这个系统中不可能 发生死锁。

答 2 ：由题意知道，n×m < m + n 是成立的， 等式变换n×( m - 1 ) + n < n + m

即n×(m-1) < m

于是有n×( m-1 ) + 1<m + 1 或n× ( m-1 ) + 1≤m

这说明当n 个进程都取得了最大数减 1 个即（m- 1 ）个时，这时至少系统还有一个资源可分配。故该系统是死锁无关的。

1. 一条公路两次横跨运河，两个运河桥相距 100 米，均带有闸门，以供船只通过运河桥。运河和公路的交通均是单方向的。运河上的运输由驳船担负。在一驳船接近吊桥 A 时就拉汽笛警告， 若桥上无车辆，吊桥就吊起，直到驳船尾 P 通过此桥为止。对吊桥 B 也按同样次序处理。一般典型的驳船长度为 200 米，当它在河上航行时是否会产生死锁？若会，说明理由，请提出一个防止

死锁的办法，并用信号量来实现驳船的同步。

答：当汽车或驳船未同时到达桥 A 时，以任何次序前进不会产生死锁。但假设汽车驶过了桥 A ， 它在继续前进，并且在驶过桥 B 之前，此时有驳船并快速地通过了桥 A ，驳船头到达桥B ，这时会发生死锁。因为若吊起吊桥 B 让驳船通过，则汽车无法通过桥 B ；若不吊起吊桥 B 让汽车通过，则驳船无法通过桥 B 。可用两个信号量同步车、船通过两座桥的动作。

var Sa , Sb : semaphore ; Sa:=Sb:=1 ;

cobegin

{

process 驳船begin

P(Sa ) ;

P(Sb ) ;

船过桥A 、B ; V(Sa ) ;

V(Sb ) ;

end

process 汽车begin

P ( Sa ) ;

P ( Sb ) ;

车过桥A 、B ; V ( Sa ) ;

V ( Sb ) ;

end

}

coend

1. Jurassic 公园有一个恐龙博物馆和一个花园，有 m 个旅客租卫辆车，每辆车仅能乘一个一旅客。旅客在博物馆逛了一会，然后，排队乘坐旅行车，挡一辆车可用喊飞它载入一个旅客，再绕 花园行驶任意长的时间。若 n 辆车都己被旅客乘坐游玩，则想坐车的旅客需要等待。如果一辆车己经空闲，但没有游玩的旅客了，那么，车辆要等待。试用信号量和 P 、V 操作同步m 个旅客和n 辆车子。

答：这是一个汇合机制，有两类进程：顾客进程和车辆进程，需要进行汇合、即顾客要坐进车辆 后才能游玩，开始时让车辆进程进入等待状态

var sc1 , sck , sc ，Kx,xc ，mutex : semaphore ; sck:=kx:=sc:=xc:=0；

sc1:=n ；mutex : = 1 ;

sharearea ：一个登记车辆被服务乘客信息的共享区； cobegin

process 顾客i ( i = 1 , 2 ，„ ） begin

P ( sc1 ) ; / ＊车辆最大数量信号量

P ( mutex ) ; / ＊封锁共享区，互斥操作

在共享区sharearea 登记被服务的顾客的信息：起始和到达地点，行驶时间V ( sck ) ; /\* 释放一辆车 ,即顾客找到一辆空车

P （Kx）; /\* 待游玩结束之后，顾客等待下车V ( sc1 ) ; /\*空车辆数加 1

End

Process 车辆j(j=1,2,3„) Begin

L:P(sck); /\*车辆等待有顾客来使用

在共享区sharearea 登记那一辆车被使用，并与顾客进程汇合； V(mutex); /\*这时可开放共享区，让另一顾客雇车

V(kx); /\*允许顾客用此车辆车辆载着顾客开行到目的地； V(xc); /\*允许顾客下车Goto L;

End coend

1. 今有k 个进程，它们的标号依次为 1 、2 、„ 、k ，如果允许它们同时读文件 file ，但必须满足条件：参加同时读文件的进程的标号之和需小于 K ，请使用：1 ）信号量与P 、v 操作，

2 ）管程，编写出协调多进程读文件的程序。答 1 : l ）使用信号量与P 、v 操作

var waits , mutex :semphore ; numbersum:integer:=0 ; wait:=0；mutex:=1 ;

cobegin

{

process readeri ( var number:integer ; ) begin

P(mutex ) ;

L:if numbersum+number≥ K then { V ( mutex ) ; P ( waits ) ; goto L ; } Then numbersum:numbersum+number;

V (mutex ) ; Read file ; P(mutex ) ;

numbersum: = numbersum-number ; V(waits ) ;

V(mutex ) ;

2 ）使用管程：

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

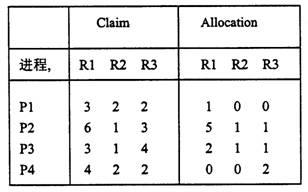
}

process-i()

var number : integer ; begin

number : ＝进程读文件编号； startread(number);;

read F ; endread(number) ; end

26、设当前的系统状态如下：系统此时 Available=(1,1,2):

l ）计算各个进程还需要的资源数 Cki - Aki ( 2 ）系统是否处于安全状态，为什么？

( 3 ) P2 发出请求向量request2 ( 1 , o , 1 ) ，系统能把资源分给它吗？

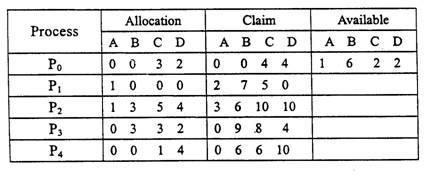
( 4 ）若在P2 申请资源后，若P1 发出请求向量req 够stl ( 1 ,0, l ) ，系统能把资源分给它吗？

( 5 ）若在P1 申请资源后，若P3 发出请求向量request3 ( 0 ,0，l ) ，系统能把资源分给它吗？

答：( 1 ) P1 , P2 , P3 , P4 的Cki . Aki 分别为：( 2 , 2 , 2 ）、（1 , 0 , 2 ）、（1 , 0 , 3 ）、（4 , 2 , 0 ) ( 4 ）系统处于安全状态，存在安全序：P2 , P1 , P3 , P4

( 5 ）可以分配，存在安全序列：P2 , P1 , P3 , P4 . ( 6 ）不可以分配，资源不足。

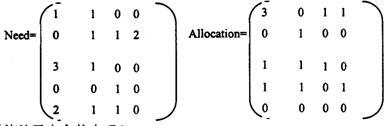
( 7 ）不可以分配，不安全状态。

1. 系统有A 、B 、C 、D 共 4 种资源，在某时刻进程 PO 、Pl 、PZ 、P3 和P4 对资源的占有和需求情况如表，试解答下列问题：

系统此时处于安全状态吗？

若此时P2 发出request2 ( 1 、2 、2 、2 ) ，系统能分配资源给它吗？为什么？ 答：( l ）系统处于安全状态，存在安全序列：P0, P3 , P4 , P1 , P2 。

( 2 ）不能分配，否则系统会处于不安全状态。

1. 把死锁检测算法用于下面的数据，并请问： Available=(1,0,2,0)

( l ）此时系统处于安全状态吗？

( 2 ）若第二个进程提出资源请求 request2( 0 , 0 , 1 , 0 ) 系统能分配资源给它吗？

(3）执行（2）之后，若第五个进程提出资源请求 request5( 0 ,0 ,1 ,0 )系统能分配资源给它吗？

答：( l ）此时可以找出进程安全序列：P4 , P1 , P5 , P2 , P3 。故系统处于安全状态。( 2 ）可以分配，存在安全序列：P4 , P1 , P5, P2 , P3 。

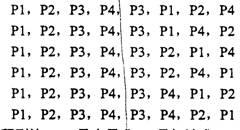
( 3 ）不可分配，系统进入不安全状态。

1. ）考虑一个共有巧 0 个存储单元的系统，如下分配给三个进程，P1 最大需求 70 ，己占有 25 ; 以P2 最大需求 60 ，己占有 40 ; P3 最大需求 60 ，己占有 45 。使用银行家算法，以确定下面的任何一个请求是否安全。（l ) P4 进程到达，P4 最大需求 60 ，最初请求 25 个。（2 ) P4 进

程到达，P4 最大需求 60 ，最初请求 35 。如果安全，找出安全序列；如果不安全，给出结果分配情况。

答：

( l ）由于系统目前还有 150-25-40-45=40 个单元，P4 进程到达，把 25 个单元分给它。这时系统还余 15 个单元，可把 15 个单元分给P3 ，它执行完后会释放 60 个单元。于是可供 P1 （还要 45 个单元）, P2 （还要 20 个单元）, P4(还要 35 个单元)任何一个执行。

安全序列为：

（1）P4 进程到达，P4 最大需求 60，最初请求 35 。如果把 35 个单元分给P4 ，系统还余 5 个单元，不再能满足任何一个进程的需求，系统进入不安全状态。

1. 有一个仓库，可存放X 、Y 两种产品，仓库的存储空间足够大，但要求：( l ）每次只能存入一种产品X 或Y , ( 2 ）满足-N＜X 产品数量-Y 产品数量＜M 。其中，N 和M 是正整数，试用信号量与P 、V 操作实现产品X 与Y 的入库过程。

答：本题给出的表达式可分解为制约条件：

-N < X 产品数量-Y 产品数量X 产品数量-Y 产品数量＜M

也就是说，X 产品的数量不能比 Y 产品的数量少 N 个以上，X 产品的数量不能比 Y 产品的数量多M 个以上。可以设置两个信号量来控制 X 、Y 产品的存放数量：

SX 表示当前允许X 产品比Y 产品多入库的数量，即在当前库存量和 Y 产品不入库的情况下，还可以允许SX 个X 产品入库；初始时，若不放 Y 而仅放X 产品，则SX 最多为M-1 个。

sy 表示当前允许Y 产品比x 产品多入库的数量，即在当前库存量和 x 产品不入库的情况下，还可以允许sy 个Y 产品入库．初始时，若不放 X 而仅放Y 产品，则sy 最多为N -1 个。当往库中存放入一个X 产品时，则允许存入 Y 产品的数量也增加 1 ，故信号量sy 应加 1 ：当往库中存放入一个Y 产品时，则允许存入 X 产品的数量也增加 1 ，故信号量sx 应加 1 .

var mutex : semaphore = 1 /\*互斥信号量＊/ sx , sy : semaphore;

sx = M-1 ; sy = = N - l ; cobegin

{

process X

｛repeat P(sx ) ;

P （mutex ) ; 将X 产品入库； V(mutex ) ;

V ( sy ) ; until false

}

process Y

{ repeat

P ( sy ) ;

P （mutex ) ; 将Y 产品入库； V （mutex ) ; V ( px ) ; until false

}

}

coend .

1. 有一个仓库可存放A 、B 两种零件，最大库容量各为 m 个。生产车间不断地取 A 和B 进行装配，每次各取一个．为避免零件锈蚀，按先入库者先出库的原则。有两组供应商分别不断地供应 A 和B ，每次一个。为保证配套和合理库存，当某种零件比另一种零件超过 n ( n < m ）个时， 暂停对数量大的零件的进货，集中补充数量少的零件．试用信号量与 P 、V 操作正确地实现它们之间的同步关系。

答：按照题意，应满足以下控制关系：A 零件数量-B 零件数量≤n ; B 零件数量-A 零件数量≤n : A 零件数量≤m ; B 零件数量≤m ．四个控制关系分别用信号量 sa 、sb 、empty1 和empty2 实施。为遵循先入库者先出库的原则，A 、B 零件可以组织成两个循形队列，并增加入库指针 in1 、in2 和出库指针out1 、out2 来控制顺序。并发程序编制如下：

Var empty1,empty2,full1,full2:semaphore; Mutex ,sa,sb:semaphore; In1,in2,out1,out2:integer; Buffer1,buffer2:array[0„m-1]of item; Empty1:=empty2:=m;

Sa:=sb:=n; In1:=in2=out1:=out2:=0; Cobegin

{

Process producerA

{repeat P(empty1); P(sa);

P(mutex); Buffer1[in1]:=A 零件; In1:=(in1+1)mod m; V(mutex);

V(sb);

V(full1); Untile false;

}

Process producer B

{repeat P(empty2); P(sb);

P(mutex); Buffer2[in2]:=B 零件; In2:=(in2+1)mod m; V(mutex);

V(sa);

V(full2); Untile false;

}

Process take

{repeat P(full1);

P(full2);

P(mutex);

Take from buffer1[out1] and buffer2[out2]中的A，B 零件； Out1:=(out1+1)mod m;

Out2:=(out2+1)mod m; V(mutex); V(empty1); V(empty2);

把A 和B 装配成产品； Until false

}

}

Coend.

1. 进程Al 、A2 、„、An1 通过m 个缓冲区向进程B1 、B2 、„ 、Bn2 不断地发送消息．发送和接收工作符合以下规则：

( l ）每个发送进程每次发送一个消息，写进一个缓冲区，缓冲区大小与消息长度相等； ( 2 ）对每个消息，Bl 、BZ 、二、BnZ 都需接收一次，并读入各自的数据区内；

( 3 ）当M 个缓冲区都满时，则发送进程等待，当没有消息可读时，接收进程等待． 试用信号量和PV 操作编制正确控制消息的发送和接收的程序。

答：本题是生产者一消费者问题的一个变形，一组生产者 A1 , A2 ，„ An1 和一组消费者B1 , B2 ，„ Bn2 共用m 个缓冲区，每个缓冲区只要写一次，但需要读 n2 次。因此，可以把这一组缓冲区看成n2 组缓冲区，每个发送者需要同时写 n2 组缓冲区中相应的 n2 个缓冲区，而每一个接收者只需读它自己对应的那组缓冲区中的对应单元。

应设置一个信号量 mutex 实现诸进程对缓冲区的互斥访问；两个信号量数组 empty[n2]和full[n2]描述n2 组缓冲区的使用情况．其同步关系描述如下：

var mutex , empty[n2],full[n2]:semaphore ; i :integer ; mutex=1 ;

for(i=0;i<=n2-1;i++)

{

empty[i]=m; Full[i]=0;

}

main ( )

{

cobegin A1 ( ) ;

A2 ( ) ;

„

An1 ( ) ;

B1 ( ) ;

B2 ( ) ;

„

Bn2 ( ) ;

coend

send ( ) / ＊进程Ai 发送消息＊/

{ int i ;

for （i=0;i<=n2-1;i++）; P(empty[i]);

P (mutex ) ;

将消息放入缓冲区； V （mutex ) ;

for(i=0；i<=n2-1;i++) V(full[i]);

}

receive (i) /＊进程Bi 接收消息＊/

{

P(full[i]); P(mutex);

将消息从缓冲区取出； v （mutex ) ;

v ( empy[i]) ;

Ai ( ) / ＊发送进程A1 , A2 ，„ An1 的程序类似，这里给出进程 Ai 的描述＊l {

{

While(1)

{

„

send ( ) ;

„

}

}

Bi ( ) /＊接收进程Bl , B2 ，„ BnZ 的程序类似，这里给出进程 Bi 描述＊/

{

while(i) (

„

receive ( i ) ;

„

}

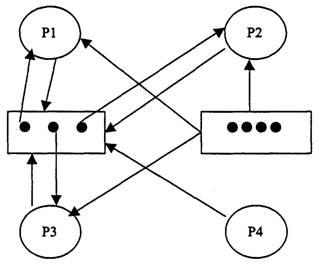
}

某系统有R1 设备 3 台，R2 设备 4 台，它们被Pl 、PZ 、P3 和P4 进程共享，且己知这 4 个进程均按以下顺序使用设备：

一申请Rl 一申请R2 一申请RI ～释放Rl 一释放R2 一释放Rl ( 1 ）系统运行中可能产生死锁吗？为什么？

( 2 ）若可能的话，请举出一种情况，并画出表示该死锁状态的进程一资源图．

答：( l ）系统四个进程需要使用的资源数为 Rl 各 2 台，R2 各 1 台。可见资源数不足，同时各进程申请资源在先，有可能产生死锁发生的四个条件，故系统可能产生死锁。( 2 ）当三个进程执行完申请资源 Rl ，开始执行申请资源 R2 时，第四个进程会因没有资源 Rl 而被阻塞。当三个进程执行完申请资源R2 后，系统还剩1 个R2 资源。而这三个进程因执行申请第二个资源Rl 而全部被阻塞，系统进入死锁。



1. f如图所示，左右两队杂技演员过独木桥，为了保证安全，请用 PV 操作和信号量来解决过独木桥问题。只要桥上无人，则允许一方的人过桥，待一方的人全部过完后，另一方的人才允许过桥。

答：

var wait ，mutex1 ，mutex2 , bridge1 , bridge2 : semaphore ; mutex1:＝mutex2:=bridgel:=bridge2:=1;wait:=0;

counter1 , counter2 : integer ; cobegin

{

process P 左 process P 右begin begin

P ( mutex1 ) ; P ( mutex2 ) ;

Count1 ++; count2 ++；

if count1 = 1 then P( wait ) ; if count2 = 1 then P( wait ) ; V ( mutex1 ) ; V( mutex2) ;

P(bridge1) ; P ( bridge2 ) ; 过独木桥； 过独木桥；

V ( bridge1) ; V( bridge2 ) ; P ( mutex1) ; P ( mutex2 ) ;

Count1-- ; count2--；

if count1 = 0 then V(wait) ; if count2 = 0 then P (wait) ; V ( mutex1) ; V (mutex2) ;

end ; end ;

} coend

1. 修改读者一写者的同步算法，使它对写者优先，即一旦有写者到达，后续的读者必须等待，而 无论是否有读者在读文件。（1 ）用信号量和 P 、v 操作实现；( 2 ）用管程实现。

答：（1）用信号量和 P、V 操作实现

为了提高写者的优先级，增加了一个信号量 S，用于在写进程到达后封锁后续的读者。其控制流程如下：

Var rmutex,wmutex,s:semaphore; Rmutex=1;wmutex=1;s=1; Count:integer:=0;

Main()

{cobegin Reader();

Writer(); Coend

}

Reader() Begin While(1)

{ P(s);

P(rmutex); If(count==0) P(wmutex); Count++;

V(rmutex); V(s);

读文件； P(rmutex); Count--;

If (count==0) v(wmutex); V(rmutex);

}

Writer() Begin While(1)

{ P(s);

P(wmutex); 写文件； V(wmutex); V(s);

}

End.

(2)用管程实现

TYPE read-write=monitor Var rc,wc:integer; R,W:condition;

DEPINE start-read , end-read , start-riter , end-writer; USE wait , signal , check , release ;

procedure start-read; begin

check ( IM ) :

if wc > 0 then wait ( R ,IM ) ; rc:=rc + 1;

signal ( R , IM ) ; release ( IM ) ; end ;

procedure end-read ; begin

check ( IM ) ; rc:=rc-1 ;

If rc=0 then signal ( W , IM ) ; release ( IM ) ;

end ;

procedure start-write ; begin

check ( IM ) ; wc:=wc + 1 ;

if rc > 0 or wc > 1 then wait ( W , IM ) : release ( IM ) ;

end ;

procedure end-write ; begin

check ( IM ) ; wc:=wc-1 :

if wc > 0 then signal ( W , IM ) ; else signal ( R , IM ) ;

release ( IM ) ; end ;

begin

rc:=0; wc:=0 ; R:=0 ; W:=0 ;

end . Cobegin { process P1 begin

„„

call read-writer.start-read;

„„

Read;

call read-riter.end-read ; end ;

process P2 begin

„„

Call read-writer.start-writer;

„„

Write;

„„

Call read-writer.end-write;

„„

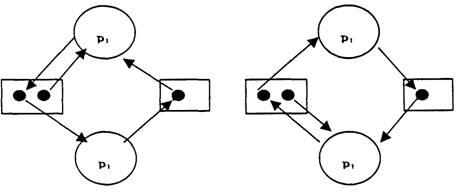
End;

}

Coend.

1. 假定某计算机系统有R1 和R2 两类可再使用资源（其中 R1 有两个单位，R2 有一个单位）， 它们被进程P1, P2 所共享，且已知两个进程均以下列顺序使用两类资源． →申请R1→申请R2→ 申请R1→释放R1→释放R2→释放R1→

试求出系统运行过程中可能到达的死锁点，并画出死锁点的资源分配图（或称进程→资源图）。 答：当两个进程都执行完第一步（都占用 R1）时，系统进入不安全状态。这时无论哪个进程执行完第二步，死锁都会发生。可能到达的死锁点：进程 P1 占有一个R1 和一个R2 ，而进程P2 占有一个R1 。或者相反。这时己形成死锁。进程--资源图为：



37、 某工厂有两个生产车间和一个装配车间，两个生产车间分别生产 A 、B 两种零件，装配车间的任务是把A 、B 两种零件组装成产品。两个生产车间每生产一个零件后都要分别把它们送到装配车间的货架Fl 、F2 上，F1 存放零件A , F2 存放零件B , Fl 和F2 的容量均为可以存放

10 个零件。装配工人每次从货架上取一个 A 零件和一个B 零件，然后组装成产品。请用：( l ） 信号量和P 、V 操作进行正确管理，( 2 ）管程进行正确管理．

答：( 1 ）信号量和P 、V 操作进行正确管理． var Fl , F2 : ARRAY [ 0„9 ] of item;

SP1 , SP2 , SI1 , SI2:seMaphore ; in1 , in2 ，outl ，outZ ：integer ; in1:=0;in2:=0;out1:=0；out2:=0； SP1:=10;SP2:=10;SI1:=0;SI2:=0;

Main()

{cobegin Producer1(); Producer2(); Installer() Coend

}

Process producer1() Begin

While(true)

{

Produce A 零件； P(SP1);

F1[in1]:A;

In1:=(in1+1) mod 10 V(SI1);

}

End

Process producer2() Begin

While(true)

{

Produce B 零件； P(SP2);

F2(in2):=B;

In2:=(in2+1) mod 10 V(SI2);

}

End

Process installer() Var product:item; Begin

While(true)

{ p(SI1);

Product1:=F1[out1]; Out1:=(out1+1) mod 10; V(SP1);

P(SI2);

Product2:=F2[out2]; Out2:=(out2+1) mod 10; V(SP2);

组装产品；

}

End

TYPE produceprodut=monitor

VAR F1 , F2 : ARRAY [ 0 „9 ] of item;

SP1 , SP2 , SG1 , SG2:semaphore;

SP1\_count1,SP2 count2 , SG1\_count,SG2\_count:integer; In1, in2 ，out1 ，out2:=integer ;

inc1 , inc2 : integer ; DEFINE put1 , put2 , get :

USE wait,signal; procedure put1( A ); begin

if inc1=10 then wait ( SP1 , SP1\_count , IM ); Inc1:=inc1 + 1 :

F1[in1]:= A ; in1:=(in1 + 1 ) MOD 10

signal ( SG1 , SG1\_count , IM ) ; end :

procedure put2 ( B ) :

begin

if inc2 =10 then wait ( SP2 , SP2\_count , IM ); Inc2 :=inc2 + 1 ;

F2 [in2]:=B; in2:=(in2 + 1 ) MOD 10

signal ( SG2 , SG2\_count , IM ) ; end ;

procedure get ( A , B ) ; begin

if inc1=0 then wait ( SG1 , SG1\_count , IM ) ; if inc2=0 then wait ( SG2 , SG2\_count , IM ) ; inc1:=inc1-1 ;

inc2:=inc2-1; A:F1[out1];

out1:=(out1 + 1 ) MOD 10 B:=F2[out2];

Out2 :=(out2 + 1 ) MOD 10

signal ( SP1 , SP1\_count , IM ) ; signal ( SP2 , SP2\_count , IM ) ; end ;

begin

in1:=0 ;in2:=0;out1:=0;out2:=0;inc1:=0;inc2:=0 ; SP1:=0;SP2:=0;SG1:=0;SG2:=0;

end. cobegin

{

process Produce1 begin while(true)

{produce A 零件； P(IM.mutex);

Call produceprodut.put1(A); If IM.next>0 then V(IM.next); Else V(IM,mutex);

}

End;

Process Produce2 Begin While(true)

{produce B 零件; P(IM.mutex);

Call produceprodut.put2(B); If (IM.next>0 then V(IM.next); Else V(IM,mutex);

}

Process consume Begin While(true)

{

P(IM.mutex);

Call produceprodut.get(A,B); If IM.next>0 then V(IM.next); Else V(IM,mutex);

组装产品；

}

End;

}

Coend.

1. 桌上有一只盘子，最多可以容纳两个水果，每次仅能放入或取出一个水果。爸爸向盘子中放苹 果(apple ) ，妈妈向盘子中放桔子(orange ) ，两个儿子专等吃盘子中的桔子，两个女儿专等吃盘子中的苹果．试用：( 1 ）信号量和P 、v 操作，( 2 ）管程，来实现爸爸、妈妈、儿子、女儿间的同步与互斥关系．

答：( l ）用信号量和P 、v 操作．

类似于课文中的答案，扩充如下：1 ）同步信号量初值为 2 ; 2 ）要引进一个互斥信号量 mutex , 用于对盘子进行互斥：3 ）盘子中每一项用橘子、苹果 2 个枚举值。

Var

plate ARRAY [ 0 , 1] of ( apple , orange ) ; flag0 , fiag1:=boolean ;

mutex : semaphore ;

sp : semaphore; / ＊盘子里可以放几个水果＊/

sg1 , sg2 : semaphore ; / ＊盘子里有桔子，有苹果\* / sp : = 2 ; / ＊盘子里允许放入二个水果＊/

sg1 :=sg2 :=0 ; / ＊盘子里没有桔子，没有苹果＊/ flag0:=flag1:=false ; mutex :=1 :

cobegin process son process father begin begin L3 : P (sg1 ) ;

L1 :削一个苹果； P( mutex ) ;

P ( sp ) ; if（flag0&flte[0]== 桔 子 ） then If(flag0==false) then else{x:=plate[1];flag1:=false;}

{ plate[0]:＝苹果；flag1:=true;} v(mutex); else {plate[1]:=苹果；flag1:=true;} V(sp) ; v (mutex ); 吃桔子；

v(sg2) goto L3; goto Ll ; end; end ;

process mother process daughter begin begin

L2 ：剥一个桔子； L4 : P ( 592 ) :

P ( sp ) ; P ( mutex )

P ( mutex ) ; if ( flag0 & plate [0]=＝苹果）then

if ( flag0==false ）then {x:=plate [01]; flag0:=false ; }

{plate[0]:＝桔子；flag0:=true;) else { x:==plate[1] ; flag1:=false ; } else {plate[1]:=＝桔子；flag1:＝true ; } V ( mutex ) ;

V (mutex) ; V ( sp ) ; V (sg1) ; 吃 苹 果 ； goto L2 ; goto L4; end ; end ;

coend .

( 2 ）用管程．

TYPE FMSD = MONITOR

VAR plate ARRAY [ 0 , 1 ] of ( apple , orange ) ; Count:integer ; flag0，flag1:boolean ;

SP ,SS , SD : codition ; DEFFINE 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(varfruit:(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)

{准备好苹果；

Call FMSD.put(apple);

„„

}

End;

Process mother Begin While(1)

{

准备好桔子；

Call FMSD.put(orange);

„„

}

End; Process son Begin While(1)

{call FMSD.get(orange,x); 吃取到的桔子；

„„

}

End;

Process daughter Begin

While(1)

{

Call FMSD.get(apple,x); 吃取到的苹果；

„„

}

End;

}

Coend

1. 一组生产者进程和一组消费者进程共享九个缓冲区，每个缓冲区可以存放一个整数。生产者进 程每次一次性向 3 个缓冲区写入整数，消费者进程每次从缓冲区取出一个整数。请用：( l ）信号量和P 、V 操作，( 2 ）管程，写出能够正确执行的程序。

答：( l ）信号量和P 、V 操作。

var buf : ARRAY [ 0„8 ] of integer ; count,getptr , putptr ：integer ; count:=0；getptr:=0;putptr:=0；

S1 , S2 , SPUT , SGET ; semaphore ; S1:=1 ; S2 :=1 ; SPUT : = 1 ; SGET :=0；

main（)

{ cobegin producer-i( ) ;

consumer-j( ) ; coend

}

process producer-i begin

L1 ：生产 3 个整数； P(SPUT);

P(S1);

Buf[putptr]:=整数 1;

Putptr:=(putptr+1)mod 9;

Buf[putptr]:＝ 整 数 2 ; putptr :=(puttr+1 ) MOD 9 buf[putptr]:=整数 3 ;

putptr:=(putptr+1) MOD 9; V ( SGET ) ;

v ( SGET ) ;

v ( SGET ) :

v ( S1 ) ;

goto L1 end

process consumer-j var y:integer ; begin

L2:P(SGET ) ;

P (S2 ) ;

y=buf[getptr] ; getptr:=(getptr + 1) MOD 9 ;

count:=count + 1; if count= 3 then begin

count:=0；

V ( SPUT ) ;

end

V (S2 ) ;

consume the 整数 y; goto L2 ;

end

( 2 ）管程。

TYPE get-put = MONITOR

VAR buf ARRAY [ 0 „8] of integer ; count , getptr , putptr:integer ; SP , SG ; codition

DEFINE put,get ;

USE wait ,signal , check , release ; Procedure put(var a1 , a2 , a3 :integer ; ) ; begin

check(IM）;

if ( coun>6）then wait(SP , IM ) ; count:count + 3 ;

buf[putptr]:=a1 ; putptr(put1+1 ) MOD 9; buf [putptr]:=a2； putptr:=(putptr+1) MOD 9 ; buf[putptr]:=a3; putptr:=(putptr+1) MOD 9 ; signal(SG,IM);

release(IM ) ; end ;

procedure get (b); begin

check(IM);

if ( count==0）then wait(SG,IM ) ; b:buf[getptr] ;

getptr:=(getptr + 1 ) MOD 9 ; count :=count + 1 ;

if count < 7 then signal ( SG,IM ) ; else if count > 0 then signal ( SG,IM ) ; release ( IM ) ;

end； begin

count:=0；getptr:=0;putptr:=0； SP:=0;SG:=0;

End; cobegin

{

process producer-i begin

L1 ：生产 3 个整数；

Call get-put.put(a1, a2 , a3 ) ; goto L1

end

process consumr-j var y:integer ; begin

L2 : call get-put.get(b) consume the 整 数 b ; goto L2;

end

}

coend

1. 设有三个进程P 、Q 、R 共享一个缓冲区，P 进程负责循环地从磁带机读入一批数据并放入缓冲区，Q 进程负责循环地从缓冲区取出 P 进程放入的数据进行加工处理并把结果放入缓冲区， R 进程负责循环地从缓冲区读出 Q 进程放入的数据并在打印机上打出。请用：( 1 ）信号量和 P 、v 操作，( 2 ）管程，写出能够正确执行的程序．

答：( 1 ）信号量和P 、v 操作var Sp , Sq , Sr : semaphore; Buf ：integer; SP:=1;SP:=Sr:=0;

Cobegin

{process P Begin Repeat

从磁带读入数据； P(SP);

Buf:=data; V(sq); Until false; End

Process Q Begin Repeat P(sq); Data:=buf;

加工处理data;

Buf:=data;

V(Sr);

Until false; End

Process R Begin Repeat P(Sr);

Data:=buf; V(sp);

打印数据

Until false; End

}

(2)管程

TYPE PQR=MONITOR

VAR buf:integer; SP,SQ,SR:codition; 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

Process QGET(var data:integer;); Begin

Check(IM);

If turn !=q then 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 then 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

41、下述流程是解决两进程互斥访问临界区问题的一种方法。试从“互斥”（mutual exclusinn ）、 “空闲让进(progress ）、“有限等待(bounded waiting ）等三方面讨论它的正确性。如果它是正确的，则证明之；如果它不正确，请说明理由。

Program attemp; Var c1,c2:integer;

Procedure p1; (/\*对第一个进程P1\*/) Begin

Repeat

Remain section 1; Repet

C1:=1-c2;

Until c2<>0;

Critical section; (/\*临界区\*/) C1:=1;

Until false End;

Procedure p2; (/\*对 另一个进程p2\*/) Begin

Repet

Remain section 2; Repeat

C2:=1-c1

Until c1<>0;

Critical section; (/\* 临界区\*/) C2:=1

Until false End;

Begin (/\*主程序\*/) C1:=1;

C2:=1;

Cobegin

P1;P2 (/\*两进程P1,P2 开始执行\*/) Coend

End

答：( 1 ）互斥

己知cl 和c2 的初值为 1 ，若进程 P1 执行到c1: = 1-c2 时，进程 P2 也同时执行c2 : = 1-c1 ．这样一来，c1 和c2 的值都变为 0，接着再各自执行，repeat---untile 循环语句c1: = 1-c2 和c2 :=1-c1 时， c1 和c2 就又都变回了 1。于是，P1 和P2 会同时进入临界区，不满足互斥条件。

( 2 ）有空让进

设开始无进程在临界区中，进程 P1 执行了c1 :=1-c2 ，由于c2 的初值为 1 ，这使得c1 的值变为 0 但c2 仍为 1 ，从而保证了P1 进入临界区。当 P1 退出临界区时，执行了 c1 :=1，使得P2 就可进入临界区。进程 P2 先执行的情况相似，能保证有空让进的原则。

( 3 ）有限等待

假定进程P1 在临界区执行，进程P2 申请进入临界区，则因进程 P1 会在有限时间内执行完并退出临界区，然后，将执行 c1 : = 1 ，这使得进程P2 因c1 值为 1 而立即可进入临界区。因而， 能满足有限等待的原则。

1. 分析下列算法是否正确，为什么？ repeat

key:=true; repeat

swap ( lock , key ) :

until key=false;

Critical section （/＊临界区\*/） Lock:=false;

Other code ; Until false;

答：由于 lock 的初值未定，如果它的值 false ，则可通过 swap 实现上锁操作。但如果 lock 的初值为true，那么，进程会永远等待而进不了临界区．

1. 以下并发执行的程序，仅当数据装入寄存器后才能加 1

Const n =50；

var tally :integer :

procedure total ( ) var count ：integer ; Begin

For count:=1 to n do tally:=tally+1 End;

Begin (/\*main program\*/) Tally:=0;

Cobegin Total();total() Coend; Writeln(tally); End.

给出该并发程序输出的 tally 值的上限和下限． 答：tally 值的上限和下限为 100 和 50 .

1. 举例说明下列算法不能解决互斥问题。var balocked ：array[ O„1] of boolean ; turn:0„1;

procedure P[id:integer]; begin

repeat blocked[id]:=true; while turn≠id do begin

while blocked [1-id] do Skip; turn: = id ;

end;

{critical section } blocked[id]:＝false :

{remainder } until false end;

begin

blocked [ 0 ]: blocked[1]:=false ; turn:=0;

cobegin P[0] ;P[1]

coend ; end.

答：为方便描述，把程序语句进行编号： Blocked[id]:=true; ①

while turn≠id do ② begin

while blocked[1-id] do skip; ③ Turn:=id; ④

End;

假设id=0，则 1-id =1 ，并且turn = 1 ．当进程P[id] 先执行① 置blocked[id]=true ：接着执行② 时，因为turn≠id 而进入到③ 执行．此时，因 blocked[1-id]为false （初值）， 故在③ 上不做空操作而打算去做④ 。麻烦的事情发生了，如果在 P[ id ] 执行④ 之前，系统又调度执行P[1-id ] , 而P [ 1-id] 在执行了① 置blocked[1-id]=true 之后，在执行② 时， 因发现turn =1-id ，故退出了while ，直接进入临界区。而这时 P[id ]继续执行④ ，虽然置turn=id 但已无法挡住P[1-id] 先己进入了临界区的事实，此后，P[ id ]也进入临界区。

所以，该算法不能解决互斥问题，它会让两个进程同时进入临界区。

1. 现有三个生产者P1 、P2 、P3 ，他们都要生产水，每个生产者都已分别购得两种不同原料， 待购得第三种原料后就可配制成桔子水，装瓶出售。有一供应商能源源不断地供应糖、水、桔子 精，但每次只拿出一种原料放入容器中供给生产者。当容器中有原料时需要该原料的生产者可取 走，当容器空时供应商又可放入一种原料。假定：生产者 P1 已购得糖和水；

生产者P2 已购得水和桔子精； 生产者P3 已购得糖和桔子精；

试用：1 ）管程，2)信号量与P 、v 操作，写出供应商和三个生产者之间能正确同步的程序． 答：1 ）管程．

TYPE makedrink = monitor

VAR S , S1 , S2 , S3 : condition ; container:item ;

DEFINE give , produce1 , produce2 , produce3 ; USE check , wait , signal , re lease ; procedure give

begin

Check ( IM ) ; take raw material ;

ifcontainer≠null then wait ( S , IM ) ; else container : = rawn materiai ;

if （container）=桔子精then singal ( s1 , IM ) ; eise if ( container)= 糖 then signal(S2 ，IM）; else signal ( S3 , IM ) ;

release ( IM ) ; end

procrdure produce1 begin

check ( IM ) ;

if ( c ontainer ）≠桔子精 then wait ( s1 , IM ) ; else { take the 桔子精 from container ；做桔子水；} signal ( S ，IM）;

re1ease ( IM ) ; end

procrdure produce2 begin

check(IM);

IF(CONTAINER)≠ 糖 then wait(S2,IM); Else{take the 糖 from container;做橘子水；} Signal(S,IM);

Release(IM); End

Procrdure produce3 Begin

Check(IM);

If(container)≠ 水 then wait(S3,IM); Else{take the 水 from container;做橘子水;} Signal(S,IM);

Release(IM); End

Begin

Container{糖，水，橘子精}; End

Cobegin

{

Process 供应商Begin

Repeat

„

Call makedrink.give();

„

Until false; End

Process P1 Begin repeat

„

Call makedrink.produce1();

„

Until false; End

Process P2 Begin Repeat

„

Call makedrink.produce2();

„

Until false; End

Process P3 Begin

Repeat

„

Call makedrink,produce3();

„

Until false; End

}

Coend.

2)信号量与P、V 操作

Var S,S1,S2,S3:=semaphore; S:=1,S1:=S2:=S3:=0;

Container{糖，水，橘子精}; Cobegin

{ process 供应商Begin

Repeat P(s);

Take raw material into container; If (container)=橘子精 then V(S1); Else if (container)=糖then V(s2); Else V(s3);

Until false; End

Process P1 Begin Repeat P(S1);

Take the 橘子精 from container; V(s);

做橘子水； Until false; End

Process P2 Begin Repeat P(s2);

Take the 糖from container; V(s);

做橘子水; Until false;

End process P3 begin repeat

P ( S3 ) ;

take the 水 from container; V ( S ) ;

做桔子水； untile false ; end

}

coend .

1. 有一材料保管员，他保管纸和笔若干。有 A 、B 两组学生，A 组学生每人都备有纸，B 组学生每人都备有笔．任一学生只要能得到其他一种材料就可以写信。有一个可以放一张纸或一支笔 的小盒，当小盒中无物品时，保管员就可任意放一张纸或一支笔供学生取用，每次允许一个学生 从中取出自己所需的材料，当学生从盒中取走材料后允许保管员再存放一件材料，请用：1 ）信号量与P 、v 操作，2 ）管程，写出他们并发执行时能正确工作的程序。

答：1 ）信号量与P 、v 操作。

var s , Sa . Sb , mutexa , mutexb : s emaphore ; s : =mutexa ：=mutexb : = 1 ; sa : = sb : = 0 ; box : ( PaPer , pen ) ;

cobegin

{

process 保管员begin

repeat

P ( S ) ;

take a material intobox ;

if ( box ) = Paper then V ( Sa ) ; else V( Sb ) ;

untile false ; end

Process A 组学生begin

repeat

P ( Sa ) ;

P ( mutexa ) ;

take the pen from box ; V ( mutexa ) ;

V ( S ) ;

write a letter; untile false ; end

Process B 组学生begin

repeat

P ( Sb ) ;

P ( mutexb ) ;

take the paper from box ; V ( mutexb ) ;

V ( S ) ;

wnte a letter ; untile false ; end

}

Coend .

2 ）管程。

TYPE paper&pen = monitor VARS , S1 , S2 : condition ; box : { paper.pen , null } DEFINE put , get1 , get2 ;

USE check , wait , signal , release ; procedure put

begin

Check ( IM ) ; take a material ;

if box ≠ null then wait ( S ,IM ) ; else box : = material ;

if ( box) = Pen then signal ( S1 , IM ) ; else signal (S2 , IM ) ;

release ( IM ) ; end

procrdure get1 begin

check ( IM ) ;

if ( box ) = null or ( box ）≠pen then wait ( S1 , IM ) ; else {take the Pen from box ; }

signal ( S , IM ) ; release ( IM ) ; end

procrdure get2 begin

check ( IM ) ;

if ( box ) = null or ( box ）≠paper then wait ( S2 , IM ) ; else { take the paper from box ; }

Signal ( S ,IM ); release ( IM ) ; end

begin

box : = null ; end

cobegin

Process 保管员begin

LI : Callp paper&Pen.put）; goto L1 end

Process A 组学生begin {

L2 : call paper&pen.get ( ) 写信；

goto L2 ; end

process B 组学生begin

L3 : call paper&pen.get ( ) 写信； goto L3 ;

end coend

1. 进程A 向缓冲区buffer 发消息，每当发出一消息后，要等待进程 B 、C 、D 都接收这条消息后，进程A 才能发新消息。试写出：( l ）用信号量和P 、v 操作，( 2 )monitor ，写出它们同步工作的程序。

答：( l ）用信号量和P 、v 操作。

本质上是一个生产者与三个消费者问题。缓冲区 buffer 只要写一次，但要读三次。可把 buffer 看作用三个缓冲块组成的缓冲区，故 sa 初值为 3 。

var Sa , Sb , Sc , Sd : semaphore ; Sa : = 3 ; Sb : = Sc : = Sd : = O ;

cobegin

{ process A begin repeat ;

P ( Sa ) ;

P ( Sa ) ;

P ( Sa ) ;

Send message to buffer ; V ( Sb ) ;

V ( Sc ) ;

V ( Sd ) ;

until false ; end

process B begin repeat

P ( sb ) ;

receive the message from buffer ; V ( Sa ) ;

until false ;

end Process C begin repeat

P ( Sc ) ;

receive the message from buffer ; V ( Sa ) ;

until false ; end

process D begin repeat

P ( Sd ) ;

receive the message from buffer ; V ( Sa ) ;

until false ; end

}

coend

( 2 ) monitor ·

TYPE send&receive=monitor

VAR SSb , SSc , SSd , Sb , Sc , Sd : selnaphore ; SSb\_count , SSc\_pount , SSd\_count : integer; Sb\_count , Sc\_count , Sd\_count ：integer;

fiagb , fiagc , fiagd : Boolean ; buffer : message ;

DEFINE sendmes receiveb receivec received ; USE wait , signal ;

procedure sendmes begin

if flagb then wait ( sb , Sb\_count ，IM）; if flagc then wait ( Sc , Sc\_count , IM ) ; if flagd then wait ( Sd , Sd\_count , IM ) ; buffer :=message ;

flagb : =flagc : =flagd : ＝true ; signal ( SSb , SSb\_count , IM ) ; signal ( SSc , SSc\_count , IM ) ; signal ( SSd , SSd\_count , lM ) ; end

procedure receiveb begin

if flagb = false then wait ( SSb , SSb\_count , IM ) ; else flagb : = false ;

signal ( Sb , Sb\_count , IM ) ;

end

procedure receivec begin

if flagc = false then wait ( SSc , SSc\_count , IM ) ; else flagb : = false ;

signal ( Sc , Sc\_count , IM ) ; release ( IM ) ;

end

procedure received begin

check ( IM ) ;

if flag=false then wait ( SSd , IM ) ; else flagb : = false ;

signal ( Sd , Sd\_count , IM ) ; release ( IM ) ;

end begin

flagb : = flagc : = flagd : = false ; end

cobegin

{ process A begin repeat

produce a message ; P ( IM.mutex ) ;

Call send&receive.sendmes() ;

If IM.next > O then V ( IM.next ) ; Else V ( IM.mutex ) ;

„

until false ; end

process B begin rpeat

P ( IM . mutex ) ;

Call send&receive . receiveb();

If IM . next > 0 then V ( IM . next ) ; Else V ( IM . mutex ) ;

„

until false ; end

process C begin repeat

P ( IM . nutex ) ;

Call send&receive . receiveco ; If IM . next > 0 thenV ( IM . next ) ; elseV ( IM . mutex ) ;

„

until false ; end

processD begin repeat

P ( IM . next ) ;

Call send&receive . receivedo ; If IM . next > 0 thenV （加．next ) ; elseV ( IM . mutex ) ;

until false ; end

}

Coend

1. 试设计一个管程来实现磁盘调度的电梯调度算法。答： type diskschedule = monitor

var headpos : integer ; direction ( up , down ) ; busy : boolean ;

S : array [0 „ 99]of condition ; DEFINE request , return ;

USE wait , signal , check , release ; procedure request ( var dest : integer ) ; begin

check ( IM ) ;

if busy then wait ( S[dest] , IM ) ; busy : ＝true ;

if ( headpos < dest ) or(headpos = dest&direction = up ) then direction : = up ;

else direction : = down ; headpos : = = dest ; release ( IM ) ;

end

procedure retum vari : integer ; begin

check ( IM ) ; busy : = false ;

if direction = = up / \* uP 为向里方向，即柱面号大的方向小 en begin\*/ i : = headpos ;

while ( i < 200 & S [ i ] = 0 ) do i : = i + 1 ; if i < 200 then Signal ( S [i] , IM ) ;

else begin / \* down 为向外方向，即柱面号小的方向 i ：角eadPos ;\*/ while ( i≥0 & S [i]＝0 ) do i : = i-1;

if i≥ 0 then signal ( S [i] , IM ) ; end

end

else begin / \* down 为向外方向，即柱面号小的方向 i : = h eadPos ; while ( i > 0 & S [ 1]= 0 ) do i : = i -l ;

ifi≥ 0 then signal ( S [ i]，IM ) ;

else begin / ＊即为向里方向，即柱面号大的方向 i : = = headPos ; while ( i < 200 & S [ i ] = 0 ) do i : = i + 1 ;

if i < 200 then signal ( S [ 1 ] , IM ) ; end

end

release ( IM ) ; begin

headpos : = 0 ; direction : ＝up ; busy : = false ;

S : = O ;

end . main()

{ cobegin process visit var k : integer ; begin

„

call diskschedul.Request(k) ;

„

访问第k 个柱面；

„

call diskschedul . Return ;

„

end coend .

1. 有P1 、P2s 、P3 三个进程共享一个表格 F , P1 对F 只读不写，P2 对F 只写不读，P3 对F 先读后写。进程可同时读 F ，但有进程写时，其他进程不能读和写。用（l ）信号量和P 、v 操作，( 2 ）管程编写三进程能正确工作的程序。

答：( 1 ）信号量和P 、v 操作。

这是读一写者问题的变种。其中，P3 既是读者又是写者。读者与写者之间需要互斥，写者与写者之间需要互斥，为提高进程运行的并发性，可让读者尽量优先。

var rmutex , wmutex : semaphore ; rnutex : = wmutex : = = 1 ; count : integer ; count : = 0 ; cobegin

{

process P1 begin repeat

P (rmutex ) ;

count : = count + 1 ;

if count= 1 then P( wmutex ) ; V ( rmutex ) ;

Read F ;

P ( rmutex ) ;

count : = count - 1 ;

if count=0 then V ( wmutex ) ; V ( rmutex ) ;

untile false ; end

process P2 begin repeat

P ( wmutex ) ; Write F ;

V ( wmutex); untile false ; process P3 begin

rpeat

P ( rmutex ) ;

count : = count + 1 ;

if count=1 then P ( wmutex ) ; V ( rmutex ) ;

Read F ;

P ( rmutex ) ; coUnt : = count-1 ;

if count = 0 then V( wmutex ); V ( rmutex ) ;

P ( wmutex ) ; Write F ; V(wmutex ) ; untile false ; end

}

coend

( 2 ）管程。

见课本读者写者问题的解。

50、现有 100 名毕业生去甲、乙两公司求职，两公司合用一间接待室，其中甲公司招收 10 人，

乙公司准备招收 10 人，招完为止。两公司各有一位人事主管在接待毕业生，每位人事主管每次只可接待一人，其他毕业生在接待室外排成一个队伍等待。试用信号量和 P 、v 操作实现人员招聘过程。

答：由于毕业生仅排成一队，故用如图的一个队列数据结构表示。在队列中不含甲、乙公司

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | A | A | B | Sm  A | B | Sn  B | A | „ |  |  |  |  |  |

都接待过的毕业生和己被录用的毕业生。只含标识为 A （被甲接待过）或只含标识为 B （被乙接待过）及无标识的毕业生队列。此外，sm 和Sn 分别为队列中甲、乙正在面试的毕业生 i ( i = 1 ,

2 ，„ ，100 ）标识、即此刻另一方不得面试该毕业生 i 。

K1 和K2 为甲、乙所录取的毕业生数，C1 、C2 为互斥信号量。注意，如果甲录取了一人，且该生没有被乙面试的话，则乙面试的毕业生将减 1 。办法是：如果甲录取了一人，且该生没有被乙面试可把乙的面试计数器 C2 加 1 （相当于乙己面试了他），从而，保证乙面试的人数值为 100 。反之对甲亦然。

var Sa , Sb , mutex : semaphore ; Sa : = Sb ：=mnutex : = 1 ;

C1 , C2 , K1 , K2 ：integer ; C1 : = C2 : = K1 : = K2 : = 0 ;

cobegin

{

process 甲公司begin

L1: P ( mutex ) ;

P ( Sa ) ;

C1 : = = C1 + 1 ; V ( Sa ) ;

If C1≤100 then

｛从标识为B 且不为Sn 或无标识的毕业生队列中选第 i 个学生，将学生i 标识为A 和Sm｝ V ( mutex ) ;

面试；

P ( mutex ) ; if 合格then

{ K1 : = K1 + 1 ;

if 学生i 的标识不含B then

{ P ( Sb ) ; C2 : = C2 + 1 ; V ( Sb ) ;

将学生i 从队列摘除；

}

else 将学生i 从队列摘除； else

if 学生i 的标识含B then 将学生i 从队列摘除; else 取消学生i 的Sm 标识；

V ( mutex ) ;

If ( K1 < 10 ) & ( C2 < 100 ) then goto L1 ;

}

process 乙公司begin

L2 : P ( mutex ) ;

P ( Sb ) ;

C2 : = C2 + 1 ; V ( Sb ) ;

if C2≤100 then

｛从标识为A 且不为sm 或无标识的毕业生队列中选第 i 个学生，将学生i 标识为B 和Sn｝ V ( mutex ) ;

面试；

P ( mutex ) ; if 合格then

{ K2 : = K2 + 1 ;

if 学生i 的标识不含A then

{ P(Sa)

C1 : = C1 + 1 ; V ( Sa ) ;

将学生i 从队列摘除；

}

else 将学生i 从队列摘除； else

if 学生i 的标识含A then 将学生i 从队列摘除； else 取消学生i 的Sn 标识；

V ( mutex ) ;

if ( K2 < 10 ) & ( c1 < 100 ）then goto L2 ; }

}

coend .

51 有一个电子转帐系统共管理 10000 个帐户，为了向客户提供快速转帐业务，有许多并发执行的资金转帐进程，每个进程读取一行输入，其中，含有：贷方帐号、借方帐号、借贷的款项数。 然后，把一款项从贷方帐号划转到借方帐号上，这样便完成了一笔转帐交易。写出进程调用Monitor ，以及MOnitor 控制电子资金转帐系统的程序。

答：

TYPE lock-account = monitor

VAR use : array [1 „ 10000] of Boolean ; / ＊该帐号是否被锁住使用标志S ：array [ 1 „ 10000 ] of condition ; / ＊条件变量

DEFINE lockaccount unlockaccount / ＊移出过程

USE wait , signal , check , release ; / ＊移入过程procedure lockaccount ( var i,j : integer )

Begin

Check ( IM )

if i > j then begin Temp:= i ;

1. : = j ;
2. : = temp ;

end ; / ＊层次分配，先占号码小的账号否则可能产生死锁if use [i] then wait(s[i].lockaccount,IM ) ;

else use [ i ] :=true ; / ＊锁住account ( i ) if use[j] then wait ( s[j].lockaccount ，IM）; else use [j］:=true ; / ＊ 锁 住 accounto ) Release ( IM ) ;

end ;

Proeedure unfockaccount ( var i:sinteger ; ) Begin

Check ( IM ) ;

use [ i ] : = sfalse ; signal(s[i].lock-account , IM ) ; Release ( IM ) ;

end begin

for i:= 1 ; to 10000 do use [i]:=false ; end .

main ( )

{

cobegin

Process transfer account begin

input a information line ;

get the account number i,j and 还款数x ; Lock-account.slockaccount ( i,j )

按锁住帐号account ( i ）和account(j）执行； A [j]:= A [j] - x ; A [i]:=A [i] + x ;

Lock-ccount.unlockaccount(i); Lock-account.unlockaccount(j); end ;

CoeDd .

52、某高校开设网络课程并安排上机实习，如果机房共有 2m 台机器，有 2n 个学生选课，规定： ( 1 ）每两个学生分成一组，并占用一台机器，协同完成上机实习；( 2 ）仅当一组两个学生到

齐，并且机房机器有空闲时，该组学生才能进机房；( 3 ）上机实习由一名教师检查，检查完毕， 一组学生同时离开机房。试用信号量和 P 、V 操作模拟上机实习过程。

答：

var mutex , enter:semaphore ; mutex : = 1 ; enter : = 0 ;

finish:=test:=rc:=0;computercounter:=2m; cobegin

{

process studenti ( i=1 , 2 ，„ ） begin

P ( computereounter ) ; / ＊申请计算机P ( mutex ) ;

rc ：rc＋1 ; / ＊学生互斥计数

if rc == 1 then { v ( mutex ) ; P ( enter ) ; } / ＊若只来一个学生，则在即 ter 上等待else { rc:= 0 ; V ( mutex ) ; V ( enter ) ; } s/ ＊到达一组中第二个学生，rc 清。是为下一组计数用学生进入机房，上机实习；

V ( finish ) ; / ＊告诉老师，实习结束P ( test ) ; / ＊等待老师检查实习结果V( computercounter ) ; / ＊归还计算机end

process teacher begin

P ( finish ) ; / ＊等第一个学生实习结束P ( finish ) ; / ＊等第二个学生实习结束检查实习结果；

V ( test ) ; / ＊第一个学生检查完成V ( test ) ; / ＊第二个学生检查完成end

}

coend .

53 某寺庙有小和尚和老和尚各若干人，水缸一只，由小和尚提水入缸给老和尚饮用。水缸可容水

10 桶，水取自同一口水井中。水井径窄，每次仅能容一只水桶取水，水桶总数为 3 个。若每次入、取水仅为 1 桶，而且不可同时进行。试用一种同步工具写出小和尚和老和尚入水、取水的活动过程。

答：互斥资源有水井和水缸，分别用 mutex1 和mutex2 来互斥。水桶总数仅 3 只，由信号量 count 控制，信号量empty 和full 控制入水和出水量。

var mutex1 , mutex2 : semaphore ; empty ，full : semaphore ;

count : integer ;

mutex1 : mutex2 : = 1 ; count : = 3 ; empty : = 10 ；full ：=0 ; cobegin

{

process 小和尚（打水）i ( i = 1 , 2 ，„ ） begin

repeat

P ( e mpty ) ; / ＊水缸满否？ P ( count ) ; / ＊取得水桶

P ( mutexl ) ; / ＊互斥从井中取水从井中取水；

V ( mutex1) ;

P ( mutex2) ; / ＊互斥使用水缸

倒水入缸；

V ( mutex2 ) ;

V ( count ) ; / ＊归还水桶v ( full ) ; / ＊多了一桶水untile false ;

end

process 老和尚（取水）j(j=1 , 2 ，„ ） begin

repeat

P ( full ) ; / ＊有水吗？ P ( count ) ; / ＊申请水桶

P ( inutex2 ) ; / ＊互斥取水从缸中取水；

V ( mutex2 ) ;

V ( count ) ; / ＊归还水桶

V ( empty ) ; / ＊水缸中少了一桶水untile false ;

end

}

coend .

54 在一个分页存储管理系统中，用 free[index] 数组记录每个页框状态，共有 n 个页框

( index=0 ，„ ，n - 1 ）。当 free[index]=true 时，表示第index 个页框空闲，free[index ]

= false 时，表示第index 个页框。试设计一个管程，它有两个过程 acquire 和return 分别负责分配和回收一个页框。

答：

TYPE framemanagement = monitor

VAR free ：array [ 0 „ n - 1 ] of Boolean ; waitcondition : codition ; i : integer ; DEFINE acquire , release ;

USE check , wait , signal ,return; procedure acquire ( var index : integer ; ) begin

check ( IM ) ;

for i : = 0 to n - 1 do

if free[i] then { free [i] : = false ; index : = i ; } else wait ( waiteondition , IM ) ;

release ( IM ) ; end

procedure return ( var index : integer ; ) begin

check ( IM ) ; free[index]:=true ;

signal ( waitcondition , IM ) ; release ( IM ) ;

end begin

for index : = 0 to n - 1 do free[index]:=true ; end

进程调用管程申请和归还页框的过程从略。

55、 AND 型信号量机制是记录型信号量的扩充，在 P 操作中增加了与条件“AND " ，故称“同时”P 操作和V 操作，记为 SP 和SV ( Simultaneous P 和V ）于是SP ( S1 , S2 ，„ ，Sn ）和VS ( S1 ; , S2 ，„ ，Sn ）其定义为如下的原语操作：

procedure SP ( vars , ，„ ，sn : semaphore ) begin

if S1 > = 1 ＆„ ＆Sn > = 1 then begin for i : = 1 to n do

Si := S1 - 1 ;

end

else begin

｛进程进入第一个遇到的满足 si < 1 条件的S1 信号量队列等待，同时将该进程的程序计数器地址回退，置为SP 操作处。｝;

end

procedure VP ( var S1 ，„ ,Sn:semaphore ) begin for i : = 1 to n do begin

Si : = S1 + 1 ;

{从所有s 。信号量等待队列中移出进程并置入就绪队列。｝; end

试回答AND 信号量机制的主要特点，适用于什么场合？

S 答：记录型信号量仅适用于进程之间共享一个临界资源的场合，在更多应用中，一个进程需要先获得两个或多个共享资源后，才能执行其任务。AND 型信号量的基本思想是：把进程在整个运行其间所要的临界资源，一次性全部分配给进程，待该进程使用完临界资源后再全部释放。只要 有一个资源未能分配给该进程，其他可以分配的资源，也不分配给他。亦即要么全部分配，要么 一个也不分配，这样做可以消除由于部分分配而导致的进程死锁。

56、试用AND 型信号量和SP 、SV 操作解决生产者一消费者问题。答：

Var B : array [ 0 ，„ k -1 ] of item ;

sput : semaphore ：= k ; / ＊指示有可用的空缓冲区的信号量sget : semaphore ：= 0 ; / ＊指示缓冲区有可用的产品信号量mutex : semaphore ：= 1 ; / ＊互斥信号量

sput : = k ; / ＊缓冲区允许放入的产品数sget : = 0 ; / ＊缓冲区内没有产品

in : integer ：= 0 ; out : Integer ：= 0 ; begin

cobegin

process producer\_i begin

L1 : produce a product ;

SP ( sput , mutex ) ; B [ in ]:= product ;

in ：=（in + 1 ) mod k ; SV ( mutex , sget ) ; goto L1 ;

end ;

process consumer\_j begin

L2 : SP ( sget , mutex ) ;

Product := B[out] ；

out : = [out + 1] mod k ; SV ( mutex , sput ) ; consume a product :

goto L2 ; end ; coend end

57、 试用AND 型信号量和SP 、SV 操作解决五个哲学家吃通心面问题。答： Var forki：array [ 0 „ 4 ] of semaphore ;

forki := 1 ; cobegin

process Pi / \* i = 0 , 1 , 2 , 3 \* / begin

L1 :

思考；

SP ( fork [ i ] ，fork [ i + 1 ] mod 5 ) ; / \* 1 = 4 时，SP ( fork [ 0 〕 ，fork [ 4 ] )

\* /

吃 通 心 面 ； V(fork[i],Vfork[i+1] mod 5); Goto L1;

End;

58、 如果AND 型信号量SP 中，并不把等待进程的程序计数器地址回退，亦即保持不变，则应该对AND 型信号量SV 操作做何种修改？

答：要保证进程被释放获得控制权后，能再次检测每种资源是否＞= 1 。故可在else 部分增加一条goto 语句，转向if 语句再次检测每种资源状况。

59、一般型信号量机制（参见汤子派等编著的计算机操作系统，西安电子科技大学出版社）

对AND 型信号量机制作扩充，便形成了一般型信号量机制，SP ( s1;,t1 , d1, ；„ ；sn , tn , dn ) 和SV ( s1 ,d1；„ sn,tn,dn）的定义如下：

procedure SP ( s1 , t1 , d1 ；„ ：sn , tn , dn ) var S1 ，„ ，Sn：semaphore ;

t1 : „ ，tn：integer ; dl ，„ ，dn : integer ; begin

if S1 > = t1 ＆„ ＆Sn ＞= Tn then begin

for i : = 1 to n do S1 : = S1 - di ;

end else

｛进程进入第一个遇到的满足 si < ti 条件的S1 信号量队列等待，同时将该进程的程序计数器地址回退，置为SP 操作处。｝;

end end

procedure SV ( S1 , d1；„ sn , dn ) var S1 ，„ Sn：semaphore ;

d1 ，„ dn：integer ; begin

for i : = 1 to n do begin S1:= S1 + di ;

｛从所有s 。信号量等待队列中移出进程并置入就绪队列。｝; end

end

其中，ti 为这类临界资源的阀值，di 为这类临界资源的本次请求数。试回答一般型信号量机制的主要特点，适用于什么场合？

答：在记录型和同时型信号量机制中，P 、V 或SP 、SV 仅仅能对信号量施行增 1 或减 1 操作， 每次只能获得或释放一个临界资源。当一请求 n 个资源时，便需要 n 次信号量操作，这样做效率很低。此外，在有些情况下，当资源数量小于一个下限时，便不预分配。为此，可以在分配之前， 测试某资源的数量是否大于阀值 t 。对AND 型信号量机制作扩充，便形成了一般型信号量机制。

60 下面是一般信号量的一些特殊情况：

* SP ( s , d , d )
* SP ( s , 1 , 1 )
* SP ( s , 1 , 0 )

试解释它们的物理含义或所起的作用。答：

* SP ( s , d , d ）此时在信号量集合中只有一个信号量、即仅处理一种临界资源，但允许每次可以申请d 个，当资源数少于 d 个时，不予分配。

1. sP ( s , 1 , 1 ）此时信号量集合已蜕化为记录型信号量（当 s > 1 时）或互斥信号量( s

= l 时）。

1. sP ( s , 1 , 0 ）这是一个特殊且很有用的信号量，当 s > = l 时，允许多个进程进入指定区域；当 s 变成 0 后，将阻止任何进程进入该区域。也就是说，它成了一个可控开关。

61、试利用一般信号量机制解决读者一写者问题·

答：对读者一写者问题作一条限制，最多只允许 m 个读者同时读。为此，又引入了一个信号量 L ， 赋予其初值为m ，通过执行 SP ( L , 1 , 1 ）操作来控制读者的数目，每当一个读者进入时， 都要做一次SP ( L , 1 , 1 ）操作，使L 的值减 1 。当有m 个读者进入读后，L 便减为 0 ， 而第m + 1 个读者必然会因执行 sP ( L , 1 , 1 ）操作失败而被封锁。

利用一般信号量机制解决读者一写者问题的算法描述如下： var m : integer ; / ＊允许同时读的读进程数

L : semaphore : = m ; / ＊控制读进程数信号量，最多 m W : semaphore : = 1 ;

begin cobegin

process reader begin

repeat

SP ( L , 1 , 1 ; W , 1 , 0 ) ;

Read the file ; SV ( L , 1 ) ;

until false ; end

process writer begin

Repeat

SP ( W , 1 , 1 ; L , rn , 0 ) ;

Write the file ; SV ( W , 1 ) ;

until false ; end

coend end .

上述算法中，SP ( w , 1 , 0 ）语句起开关作用，只要没有写者进程进入写，由于这时 w = 1 , 读者进程就都可以进入读文件。但一旦有写者进程进入写时，其 W = 0 ，则任何读者进程及其他写者进程就无法进入读写。sP ( w , 1 , 1 ; L , rn , 0 ）语句表示仅当既无写者进程在写（这时w = 1）、又无读者进程在读（这时 L = rn ）时，写者进程才能进行临界区写文件。

### 第四章

作者：佚名 来源：网络

1 在一个请求分页虚拟存储管理系统中，一个程序运行的页面走向是：

1. 、2 、3 、4 、2 、1 、5 、6 、2 、1 、2 、3 、7 、6 、3 、2 、1 、2 、3 、6 。

分别用FIFO 、OPT 和LRU 算法，对分配给程序 3 个页框、4 个页框、5 个页框和 6 个页框的情况下，分别求出缺页中断次数和缺页中断率。

答：

|  |  |  |  |
| --- | --- | --- | --- |
| 页框数 | FIFO | LRU | OPT |
| 3  4  5  6 | 16  14  12  9 | 15  10  8  7 | 11  8  7  7 |

只要把表中缺页中断次数除以 20，便得到缺页中断率。

1. 在一个请求分页虚拟存储管理系统中，一个作业共有 5 页，执行时其访问页面次序为：( 1 ) 1 、4 、3 、1 、2 、5 、1 、4 、2 、1 、4 、5

( 2 ) 3 、2 、1 、4 、4 、5 、5 、3 、4、3、2、1、5

若分配给该作业三个页框，分别采用 FIFO 和LRU 面替换算法，求出各自的缺页中断次数和缺页中断率。

答：( 1 ）采用FIFO 为 9 次，9 / 12 = 75 ％。采用LRU 为 8 次，8 / 12 = 67 ％。( 2 ）采用FIFO 和LRU 均为 9 次，9 / 13 = 69 ％。

1. 一个页式存储管理系统使用 FIFO 、OPT 和LRU 页面替换算法，如果一个作业的页面走向为：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ( l | ) | 2 | 、3 | 、2 | 、l | 、5 | 、2 | 、4 | 、5 | 、3 | 、2 | 、5 | 、2 | 。 |
| ( 2 | ) | 4 | 、3 | 、2 | 、l | 、4 | 、3 | 、5 | 、4 | 、3 | 、2 | 、l | 、5 | 。 |
| ( 3 | ) | 1 | 、2 | 、3 | 、4 | 、1 | 、2 | 、5 | 、l | 、2 | 、3 | 、4 | 、5 | 。 |

当分配给该作业的物理块数分别为 3 和 4 时，试计算访问过程中发生的缺页中断次数和缺页中断率。

答：( l ）作业的物理块数为 3 块，使用FIFO 为 9 次，9 / 12 = 75 ％。使用LRU 为 7 次，7

/ 12 = 58 ％。使用OPT 为 6 次，6 / 12 = = 50 ％。

作业的物理块数为 4 块，使用FIFO 为 6 次，6 / 12 = 50 ％。使用LRU 为 6 次，6 / 12 = 50 ％。使用OPT 为 5 次，5 /12 = 42 ％。

( 2 ）作业的物理块数为 3 块，使用FIFO 为 9 次，9 / 12 = 75 ％。使用LRU 为 10 次，10 /

12 = 83 ％。使用OPT 为 7 次，7/12 = 58 ％。

作业的物理块数为 4 块，使用 FIFO 为 10 次，10 / 12 = 83 ％。 使用LRU 为 8 次，8/12＝66％。使用OPT 为 6 次，6/12＝50%.

其中，出现了Belady 现象，增加分给作业的内存块数，反使缺页中断率上升。

4、在可变分区存储管理下，按地址排列的内存空闲区为：10K 、4K 、20K 、18K 、7K 、9K 、

12K 和 15K 。对于下列的连续存储区的请求：( l ) 12K 、10K 、9K , ( 2 ) 12K 、10K 、15K 、

18K 试问：使用首次适应算法、最佳适应算法、最差适应算法和下次适应算法，哪个空闲区被使用？

答：( 1 ）空闲分区如图所示。答

|  |  |
| --- | --- |
| 分区号 | 分区长 |
| 1  2  3  4  5  6  7  8 | 10K  4K  20K  18K  7K  9K  12K  15K |

* 1. 首次适应算法

12KB 选中分区 3 ，这时分区 3 还剩 8KB 。10KB 选中分区 1 ，恰好分配故应删去分区 1 。9KB 选中分区 4 ，这时分区 4 还剩 9KB 。

1. ）最佳适应算法

12KB 选中分区 7 ，恰好分配故应删去分区 7 。1OKB 选中分区 1 ，恰好分配故应删去分区 1 。

9KB 选中分区 6 ，恰好分配故应删去分区 6 。

1. ）最差适应算法

12KB 选中分区 3 ，这时分区 3 还剩 8KB 。1OKB 选中分区 4 ，这时分区 4 还剩 8KB 。9KB 选中分区 8 ，这时分区 8 还剩 6KB 。

1. ）下次适应算法

12KB 选中分区 3 ，这时分区 3 还剩 8KB 。10KB 选中分区 4 ，这时分区 4 还剩 8KB 。9KB 选中分区 6 ，恰好分配故应删去分区 6 。

( 2 ）原始分区情况同上图。

1. ）首次适应算法

12KB 选中分区 3 ，这时分区 3 还剩 8KB 。10KB 选中分区 1 ，恰好分配故应删去分区 1 。15KB 选中分区 4 ，这时分区 4 还剩 3KB 。最后无法满足 18KB 的申请，应该等待。

1. ）最佳适应算法

12KB 选中分区 7 ，恰好分配故应删去分区 7 。1OKB 选中分区 1 ，恰好分配故应删去分区 1 。

15KB 选中分区 8 ，恰好分配故应删去分区 8 。18KB 选中分区 4 ，恰好分配故应删去分区 4 。

1. ）最差适应算法

12KB 选中分区 3 ，这时分区 3 还剩 8KB 。10KB 选中分区 4 ，这时分区 4 还剩 8KB 。15KB 选中分区 8 ，恰好分配故应删去分区 8 。最后无法满足 18KB 的申请，应该等待。

1. ）下次适应算法

12KB 选中分区 3 ，这时分区 3 还剩 8KB 。1OKB 选中分区 4 ，这时分区 4 还剩 8KB 。15KB 选中分区 8 ，恰好分配故应删去分区 8 。最后无法满足 15KB 的申请，应该等待。

1. 给定内存空闲分区，按地址从小到大为：100K 、500K 、200K 、300K 和 600K 。现有用户进程依次分别为 212K 、417K 、112K 和 426K , ( l ）分别用first-fit 、best-fit 和worst-fit 算法将它们装入到内存的哪个分区？( 2 ）哪个算法能最有效利用内存？

答：按题意地址从小到大进行分区如图所示。

|  |  |
| --- | --- |
| 分区号 | 分区长 |
| 1  2  3  4  5 | 100KB  500KB  200KB  300KB  600KB |

( 1 ) 1）first-fit 212KB 选中分区 2 ，这时分区 2 还剩 288KB 。417KB 选中分区 5 ，这时分区 5 还剩 183KB 。112KB 选中分区 2 ，这时分区 2 还剩 176KB 。426KB 无分区能满足，应该等待。

1. ) best-fit 212KB 选中分区 4 ，这时分区 4 还剩 88KB 。417KB 选中分区 2 ，这时分区 2 还剩 83KB 。112KB 选中分区 3 ，这时分区 3 还剩 88KB 。426KB 选中分区 5 ，这时分区 5 还剩174KB 。
2. ) worst-fit 212KB 选中分区 5 ，这时分区 5 还剩 388KB 。417KB 选中分区 2 , 这时分区 2 还剩 83KB 。112KB 选中分区 5 ，这时分区 5 还剩 176KB 。426KB 无分区能满足，应该等待。

( 2 ）对于该作业序列，best-fit 算法能最有效利用内存

6、 一个 32 位地址的计算机系统使用二级页表，虚地址被分为 9 位顶级页表，11 位二级页表和偏移。试问：页面长度是多少？虚地址空间共有多少个页面？

答：由于 32-9 -11 ＝ 12 ，所以，页面大小为 4KB ，页面的个数为 220 个。

7、 一进程以下列次序访问 5 个页：A 、B 、C 、D 、A 、B 、E 、A 、B 、C 、D 、E ：假定使用FIFO 替换算法，在内存有 3 个和 4 个空闲页框的情况下，分别给出页面替换次数。

答：内存有 3 个和 4 个空闲页框的情况下，页面替换次数为 9 次和 10 次。出现了Belady 即现象，增加分给作业的内存块数，反使缺页中断率上升。

8、 某计算机有缓存、内存、辅存来实现虚拟存储器。如果数据在缓存中，访问它需要 Ans；如果在内存但不在缓存，需要 Bns 将其装入缓存，然后才能访问；如果不在内存而在辅存，需要 Cns 将其读入内存，然后，用 Bns 再读入缓存，然后才能访问。假设缓存命中率为（n-1) / n ，内存命中率为（m -1) / m ，则数据平均访问时间是多少？

答：

数据在缓存中的比率为：( n - 1 ) / n

数据在内存中的比率为：( 1 -（n - 1 ) / n )×( m - 1 ) / m = ( m - 1 ）/nm 数据在辅存中的比率为：( 1 -（n －1 ) / n )×( 1－（m －1 ) / m ）1/nm

故数据平均访问时间是＝( ( n- 1 ) / n ) × A + ( ( 1 -（n - 1 ) / n ) × ( m-1 ) / m )

× ( A + B ) + ( ( 1-（n -1 ) / n ) ×( 1-（m-1）/ m ) ) × ( A + B + C ) = A + B / n

+ C / nm

9、某计算机有 cache 、内存、辅存来实现虚拟存储器。如果数据在 cache 中，访问它需要 20ns ； 如果在内存但不在 cache ，需要 60ns 将其装入缓存，然后才能访问；如果不在内存而在辅存， 需要12us 将其读入内存，然后，用60ns 再读入cache ，然后才能访问。假设cache 命中率为0 .9 ， 内存命中率为 0.6 ，则数据平均访问时间是多少（ns )

答：506ns 。

1. 有一个分页系统，其页表存放在主存里，( 1 ）如果对内存的一次存取要 1.2 微秒，试问实现一次页面访问的存取需花多少时间？( 2 ）若系统配置了联想存储器，命中率为 80 % ，假定页表表目在联想存储器的查找时间忽略不计，试问实现一次页面访问的存取时间是多少？

答：(1) 2.4 微秒 （2 )0.8 × 1.2 + 0.2 × 2.4 = 0.76 + 0.45 = 1.24 微秒

1. 给定段表如下：

|  |  |  |
| --- | --- | --- |
| 段号 | 段首址 | 段长 |
| 0 | 219 | 600 |
| 1 | 2300 | 14 |
| 2 | 90 | 100 |
| 3 | 1327 | 580 |
| 4 | 1952 | 96 |

给定地址为段号和位移： 1 ) [ 0 , 430] 、2 ) [ 3 , 400 ］、3 ) [ 1 , 1 ］、4 ) [ 2 , 500 ］、

5 ) [ 4 , 42 ) ，试求出对应的内存物理地址。答：1) 649 2) 1 727 3) 2301 4）越界 5) 1994

12、 某计算机系统提供 24 位虚存空间，主存为 2 18 B ，采用分页式虚拟存储管理，页面尺寸为 1KB 。假定用户程序产生了虚拟地址 11123456 （八进制），而该页面分得块号为 100 ( 八进制），说明该系统如何产生相应的物理地址及写出物理地址。

答：虚拟地址 11123456 （八进制）转化为二进制为：

001 001 001 010 011 100 101 110

其中前面为页号，而后 10 位为位移：001 001 001 010 01-------1 100 101 110 。由于主存大

小为 218 B，页面尺寸为 1KB ，所以，主存共有 256 块。所以，块号为 100 （八进制）是合法地

址，于是，物理地址为 100 （八进制）与位移 1 100 101 110 并接，得到：八进制物理地址 001000000

1 100 101 110 = = 201456 （八进制）。

1. 主存中有两个空间区如图所示，

0K

|  |
| --- |
|  |
| 100K |
|  |
| 50K |
|  |

15K

125K

现有作业序列依次为：Job1 要求 30K ; Job2 要求 70K ; Job3 要求50K ；使用首次适应、最坏适应和最佳适应算法处理这个作业序列，试

问哪种算法可以满足分配？为什么？

答：首次适应、最坏适应算法处理这个作业序列可以满足分配，最佳适应算法不行。因为后者会 分割出无法使用的碎片，浪费内存，从而，不能满足所有作业的内存需求。

1. 设有一页式存储管理系统，向用户提供的逻辑地址空间最大为 16 页，每页 2048 字节，内存总共有 8 个存储块。试问逻辑地址至少应为多少位？内存空间有多大？

答：

逻辑地址 211×24 ，故为 15 位。内存大小为 23×211 = 214B = 16KB 。

15、在一分页存储管理系统中，逻辑地址长度为 16 位，页面大小为 4096 字节，现有一逻辑地址为ZF6AH ，且第 0 、1 、2 页依次存在物理块 10 、12 、14 号中，问相应的物理地址为多少？ 答：因为逻辑地址长度为 16 位，而页面大小为 4096 字节，所以，前面的 4 位表示页号。把 ZF6AH

转换成二进制为：00 10 1 1 11 0110 1010 ，可知页号为 2 。故放在 14 号物理块中，写成十六进制为：EF6AH 。

16 有矩阵：VAR A : ARRAY [ 1 „100 , 1 „100 ] OF integer；元素按行存储。在一虚存系统中，采用LRU 淘汰算法，一个进程有 3 页内存空间，每页可以存放 200 个整数。其中第 1 页存放程序，且假定程序已在内存。

程序A :

FOR i : = 1 TO 100 DO FOR j : = 1 TO 100 DO A [i，j ] : = 0 ;

程序B :

FOR j : = 1 TO 100 DO FOR i : = 1 TO 100 DO A [ i,j ] : = 0 ;

分别就程序A 和B 的执行进程计算缺页次数。

答：100 \* 100 = 10000 个数据，每页可以存放 200 个整数，故一共存放在 50 个第 99 行、第

100 行缺页中断为 5000 次。由于元素按行存储，第 1 行、第 2 行放在第 1 页，„ 第 99 行、第 100 行放在第 50 页。故对于程序 A ，缺页中断为 50 次。对于程序 B，缺页中断为 5000 次。17、一台机器有 48 位虚地址和 32 位物理地址，若页长为 8KB ，问页表共有多少个页表项？如果设计一个反置页表，则有多少个页表项？

答：因为页长 8KB 占用 13 位，所以，页表项有 235 个。反置页表项有 219 个。

1. 在虚拟页式存储管理中，为解决抖动问题，可采用工作集模型以决定分给进程的物理块数，有 如下页面访问序列：

„„ 2 5 1 6 3 3 7 8 9 1 6 2 3 4 3 4 3 4 4 4 3 4 4 3 „„

| △ t1 | | △ t2 |

窗口尺寸△ ＝9 ，试求t1 、t2 时刻的工作集。

答：t1 时刻的工作集为：{ l , 2 , 3 , 6 , 7 , 8 , 9 ｝。t 时刻的工作集为：{ 3 , 4 ｝。

1. 有一个分页虚存系统，测得 CPU 和磁盘的利用率如下，试指出每种情况下的存在问题和可采取的措施：( 1 ) CPU 利用率为 13 % ，磁盘利用率为 97 % ( 2 ) CPU 利用率为 87 % ，磁盘利用率为 3 % ( 3 ) CPU 利用率为 13 % ，磁盘利用率为 3 ％。

答：( 1 ）系统可能出现抖动，可把暂停部分进程运行。（2 ）系统运行正常，可增加运行进程数以进一步提高资源利用率。（3 ）处理器和设备和利用率均很低，可增加并发运行的进程数。20、在一个分页虚存系统中，用户编程空间 32 个页，页长IKB ，主存为 16KBo 如果用户程序有

10 页长，若己知虚页 0 、1 、2 、3 ，己分到页框 8 、7 、4 、10 , 试把虚地址OACSH 和IACSH 转换成对应的物理地址。

答：虚地址 OACSH 对应的物理地址为：12CSH 。而执行虚地址 IACSH 会发现页表中尚未有分配的页框而发生缺页中断，由系统另行分配页框。

1. 某计算机有 4 个页框，每页的装入时间、最后访问时间、访问位 R 、修改位D 如下所示（时间用时钟点数表示）:

page loaded last ref R D

|  |  |  |  |
| --- | --- | --- | --- |
| 0 126 | 279 | 0 | 0 |
| 1 230 | 260 | 1 | 0 |
| 2 120 | 272 | 1 | 1 |
| 3 160 | 280 | 1 | 1 |

分别用FIFO 、LRU 、二次机会算法分别淘汰哪一页？ 答：( 1 ) FIFO 淘汰page2 。

( 2 ) LRU 淘汰page1 。( 3 ）二次机会淘汰page1

1. 考虑下面的程序：for ( i = 0;i < 20 ; i++） For(j=0;j<10;j++)

a [ i ] : = a [i] ×j

试举例说明该程序的空间局部性和时间局部性。

答：当数组元素a [0] , a[1] ，„ ，a [ 19 ] 存放在一个页面中时，其空间局部性和时间局部性较好，也就是说，在很短时间内执行都挂行循环乘法程序，而且数组元素分布在紧邻连续的 存储单元中。当数组元素存放在不同页面中时，其时间局部性虽相同，但空间局部性较差，因为 处理的数组元素分布在不连续的存储单元中。

1. 一个有快表的请页式虚存系统，设内存访问周期为 1 微秒，内外存传送一个页面的平均时间为 5 毫秒。如果快表命中率为 75 % ，缺页中断率为 10 ％。忽略快表访问时间，试求内存的有效存取时间。

答：快表命中率为 75 % ，缺页中断率为 10 % ，所以，内存命中率为 15％。故内存的有效存取时间＝1×75 % + 2\*15%+( 5000+2) \*10%=501.25 微秒。

1. 假设某虚存的用户空间为IO24KB ，页面大小为 4KB ，内存空间为 512KB 。已知用户的虚页

10 、11 、12 、13 页分得内存页框号为 62 、78 、25 、36 ，求出虚地址OBEBC ( 16 进制） 的实地址（16 进制）是多少？

答：虚地址 0BEBC ( 16 进制）的二进制形式为：0000 1 011 1110 1011 1100 。由于页面大小为 4KB ，故其中后 12 位是位移，所以，虚地址的页号为：11 。查页表分得内存对应页框号为：

78 。己知内存空间为 512KB ，故内存共有 128 个页框，78 是合法物理块。把 78 化为 16 进制是 4E ，虚地址OBEBC ( 16 进制）的实地址（16 进制）是：4EEBC 。

25 ／某请求分页存储系统使用一级页表，假设页表全部放在主存内，:

1. ）若一次访问主存花 120ns ，那么，访问一个数据的时间是多少？
2. ）若增加一个快表，在命中或失误时需有 20ns 开销，如果快表命中率为 80 % ，则访问一个数据的时间为

答：1 ) 120ns\*2 = 240ns

2 ) ( 120 + 20 ) \*80 % +(120+120+20)\*20%=174ns

1. 设某系统中作业J . , JZ , J3 占用主存的情况如图。今有一个长度为 20k 的作业J4 要装入主存，当采用可变分区分配方式时，请回答：

( l ) J4 装入前的主存己分配表和未分配表的内容。

( 2 ）写出装入J4 时的工作流程，并说明你采用什么分配算法。10k 18k 30k 40k 54k70k

答：( 1 ）主存已分配表共有三项，由作业 j1 、j2 、j3 占用，长度依次为：10k 、30k 和 54k 未分配表共有三项：空闲区 1 、空闲区 2 和空闲区 3 ，长度依次为 18k 、40k 和 70k 。( 2 ） 作业J4 装入时，采用直接分配，搜索未分配表，空闲区 1 不能满足。所以，要继续搜索未分配表，空闲区 2 可以满足J4 的装入要求。

1. 考虑下列的段表：

段号始址段长: 段号 始址 段长0 200 500

1 890 30

2 120 100

3 1250 600

4 1800 88

对下面的逻辑地址，求物理地址，如越界请指明。l ) ＜0,480 > 2 ) < l ,25 > 3 ) < l ,14 > 4 ) < 2 , 200> 5 ) < 3 ,500 > 6 ) < 4 ,100 > .

答：l ) 680 ( 2 ) 915(3 ) 904(4 ）越界(5 ) 1750(6 ）越界。

28 请页式存储管理中，进程访问地址序序列为：10 , 11 , 104 , 170 , 73 , 305 , 180 , 240 ,

2 科，科 5 , 467 , 366。试问(1 ）如果页面大小为 100 ，给出页面访问序列。2 、讲程若分 3 个页框采用

FIFO 和LRU 替换算法，求缺页中断率？

答：l ）页面访问序列为 l , l , 2 , 2 , 1 , 4 , 2 , 3 , 3 , 5 , 5 , 4 。

2 ) FIFO 为 5 次，缺页中断率为 5 / 12 科 41.6 ％。LRU 为 6 次，缺页中断率为 6 / 12 = 50 ％。LRU 反比FIFO 缺页中断率高。

1. 假设计算机有 2M 内存，其中，操作系统占用 512K ，每个用户程序也使用 512K 内存。如果所有程序都有 70 ％的I/O 等待时间，那么，再增加 1M 内存，吞吐率增加多少？

答：由题意可知，内存中可以存放 3 个用户进程，而CPU 的利用率为：1-（70 % )3 , = 1 一（0 . 7 )3 = 65 . 7 ％。再增加 1M 内存，可增加 2 个用户进程，这时 CPU 的利用率为：1 -（70 % )5 ,

= 1 一（0 ．7）5＝83 . 2 ％。故再增加 1M 内存，吞吐率增加了：83 . 2 ％/65 . 7 ％-100 %

=27 ％。

1. 一个计算机系统有足够的内存空间存放 4 道程序，这些程序有一半时间在空闲等待 I/O 操作。问多大比例的CPU 时间被浪费掉了？

答：( 500 % )＝( l / 2 ) = 1 / 16 。

1. 如果一条指令平均需 1 微秒，处理一个缺页中断另需 n 微秒，给出当缺页中断每 k 条指令发生一次时，指令的实际执行时间。

答：( 1 +n/k)微秒。

1. 一台计算机的内存空间为 1024 个页面，页表放在内存中，从页表中读一个字的开销是 50Ons 。为了减少开销，使用了有 32 个字的快表，查找速度为 10Ons 。要把平均开销降到 20Ons 需要的快表命中率是多少？

答：设快表命中率是 x ，则内存命中率为 1-x。于是：500 ( 1-x）+ 100x = = 2 00 ，解方程得x=75 ％。

1. 假设一条指令平均需花 1 微秒，但若发生了缺页中断就需 2001 微秒。如果一个程序运行了

60 秒，期间发生了 15000 次缺页中断，若可用内存是原来的两倍，这个程序坛行需要多少时间？ 答：一个程序运行期间发生了 15000 次缺页中断，由于缺页中断处理花 2000 微秒（1 微秒是指令执行时间，于是这个程序缺页中断处理花了：2000 微秒米 1 5000 = 30 秒。占了运行时间 60 秒的一半。当可用内存是原来的两倍时，缺页中断次数减为一半，故有巧秒就能处理完。所以，这 个程序运行需要时间为：45 秒。

1. 在分页式虚存管理中，若采用 FIFO 替换算法，会发生：分给作业页面越多，进程执行时缺页中断率越高的奇怪现象。试举例说明这个现象。

答：见本章应用题 7 。

1. 假设一个任务被划分成 4 个大小相等的段，每段有 8 项的页描述符表，若页面大小一为 ZKB 。试问段页式存储系统中：( a ）每段最大尺寸是多少？伪）该任务的逻辑地址空间最大为多少？ ( c ）若该任务访问到逻辑地址空间 5ABCH 中的一个数据，试给出逻辑地址的格式。

答：段数 2 2 = 4 ，每段有 23 = 8 页，页大小为 211= ZKB 。（a ）故每段最大为 214B = 16KB 。伪）逻辑她曳匕勿风爆七尺 4 又、曰KB = 64KB 。

( c ）若该任务访问到逻辑地址空间 SABCH ，其二进制表示为： 0 101 1010 1011 1100

所以，逻辑地址表示为：01 011 010 1011 1100

SABCH 的逻辑地址为：第 1 段第 3 页，位移由后 11 位给出。

36.对已知某系统页面长 4KB ，页表项 4B ，采用多级页表映射 64 位虚地址空间。若限定最高层页表占 1 页，问它可以采用几级页表？

答：由于页面长 4KB ，页表项 4B ，故每页可· 包含IKB 个页表项。由于限定最高层页表占 1 页， 即它的页表项为 210 个；而每个页表项指向一页，每页又存放页表项个数为 210 个，依此类推，

最多可以采用砚巧取整为 6 级页表。

1. 在请求分页虚存管理系统中，若驻留集为 m 个页框，页框初始为空，在长为 p 的引用串中具有n 个不同页面n＞m ) ，对于FIFO、LRU 两种页面替换算法，试给出缺页中断的上限和下限， 并举例说明。

答：对于 FIFO 、LRU 两种页面替换算法，缺页中断的上限和下限：为 p 和n 。因为有 n 个不同页面，无论怎样安排，不同页面进入内存至少要产生一次缺页中断，故下限为 n 次。由于m<n ， 引用串中有些页可能进入内存后又被调出，而多次发生缺页中断。极端情况，访问的页都不在内 存，这样共发生了 p 次缺页中断。例如，当 vm =3 ，p=12 , n =4 时，有如下访问中：1 , 1 , 1 , 2 , 2 , 3 , 3 , 3 , 4 , 4 , 4 , 4 。缺页中断为下限 4 次。而访问串：2 , 3 , 4 , 1 , 2 ,

3 , 4 , 1 , 2 , 3 , 4 , 1 。缺页中断为上限 12 次。

1. 在请求分页虚存管理系统中，页表保存在寄存器中。若替换一个未修改过页面的缺页中断处理 需 8 毫秒，若替换一个己修改过页面的缺页中断处理需另加写盘时间 12 毫秒，内存存取周期为

1 微秒。假定 70 ％被替换的页面被修改过，为保证有效存取时间不超过 2 微秒，允许的最大缺

页中断率为多少？

答：设最大缺页中断率为 x ，则有：

( l - x ) \*1 微秒＋( 1 -70 % ) \*X\*8 毫秒＋70 % \*X \*( 8 + 12 ) = 2 微秒即得到-x +2400x + 14000x = 1 ，解得：x 约为 0 .00006 。

1. 若内存按地址递增次序有三个不邻接的空闲区 Fl 、F2 、F3 ，它们的大小分别是：50K 、120K 和 25K 。请给出后备作业序列，使得实施分配时：( l ）采用最佳适应算法效果好，但采用首次适应与最坏适应算法效果不好。（2 ）采用最环适应算法效果好，但采用首次适应与最佳适应算法效果不好。

答

( 1 ）采用最佳适应算法效果好，120 , 50 。

( 2 ）采用最环适应算法效果好，80 , 50 , 25 。

但采用首次适应与最坏适应算法效果不好。作业序列：25 但采用首次适应与最佳适应算法效果不好。作业序列：40 ,

1. 有两台计算机P1 和P2，它们各有一个硬件高速缓冲存储器 Cl 和 CZ ，且各有一个主存储器Ml 和M2。其性能为：

CI CZ Ml M2

存储容量 4KB 4KB 2MB 2MB

存取周期 60ns 80ns 1 us 0 9 us 若两台机器指令系统相同，它们的指令执行时间与存储器的平均存取周期成正比。如果在执行某个程序时，所需指令或数据在高速缓冲存储器中存取到的概率 P 是 0 . 7 ，试问：这两台计算机哪个速度快？当 P = 0 .9 时，处理器的速度哪个快？答：CPU 平均存取时间为：T = = T1+（1 -p）\*T2 , T1 为高速缓冲存储器存取周期，T2 为主存储器存取周期，p 为高速缓冲存储器命中率。

( 1 ）当p=0 . 7 时，

Pl 平均存取时间为：60 + ( 1 -0 . 7 ) \* 1 us = 360ns PZ 平均存取时间为：80 + ( 1 -0 . 7 ) \*0.9 us= 350ns 故计算机P2 比P1 处理速度快。

( 2 ）当p = 0 . 9 时，

P1 平均存取时间为：60 + ( 1 -0.9 ) \* 1 us = 160ns

PZ 平均存取时l ' ed 为：80 + ( l -0 . 9 ) \*0 .9 us = 170ns 故计算机P1 比P2 处理速度快。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FI | ( | 50 | ) |  |
|  | | | | |
| F2 | ( | 120 |  | ) |
| F3 | ( | 25 | ) |  |

### 第五章

作者：佚名 来源：网络

1 ，旋转型设备上信息的优化分布能减少为若干个拍服务的总时间．设磁鼓上分为 20 V 个区， 每区存放一个记录，磁鼓旋转一周需 20 毫秒，读出每个记录平均需用 1 毫秒，读出后经 2 毫秒处理，再继续处理下一个记录。在不知当前磁鼓位置的情况下：

( 1 ）顺序存放记录 1 、„ „ ，记录 20 时，试计算读出并处理 20 个记录的总时间；

( 2 ）给出优先分布 20 个记录的一种方案，使得所花的总处理时间减少，且计算出这个方案所花的总时间。

答：定位第 1 个记录需 10m s 。读出第 1 个记录，处理花 2ms ，这时已到了第 4 个记录，再转过 18 个记录（花 18ms ）才能找到记录 2 ，所以，读出并处理 20 个记录的总时间：10 + 3 + ( l

+ 2 + 18 ) \* 19 = 13 + 2 1 \* 19 =412ms

如果给出优先分布 20 个记录的方案为：1 , 8 , 15 , 2 , 9 , 16 , 3 , 10 , 17 , 4 , 11 , 18 ,

5 , 12 , 19 , 6 , 13 , 20 , 7 , 14 。当读出第 1 个记录，花 2ms 处理后，恰好就可以处理

记录 2 ，省去了寻找下一个记录的时间，读出并处理 20 个记录的总时间： 10+3+3\*19=13+247=260ms

2.现有如下请求队列：8 , 18 , 27 , 129 , 110 , 186 , 78 , 147 , 41 , 10 , 64 , 12 ：试

用查找时间最短优先算法计算处理所有请求移动的总柱面数。假设磁头当前位置下在磁道 1000

答：处理次序为：100 -110 -129 -147 -186 -78 -64 -41 -27 -18 -12 -10 -8 。移动的总柱面

数：264 。

3 上题中，分别按升序和降序移动，讨论电梯调度算法计算处理所有存取请求移动的总柱面数。答：升序移动次序为：100 -110 -129 -147 -186 -78 -64 -41 -27 -18 -12 -10 -8 。移动的总柱面数：264 。

降序移动次序为：100 -78 -64 -4l -27 -18 -12 -10 -8 -110 -129 -147 -186 。移动的总柱面

数：

4.某文件为连接文件，由 5 个逻辑记录组成，每个逻辑记录的大小与磁盘块大小相等，均为 512 字节，并依次存放在 50 、121 、75 、80 、63 号磁盘块上。现要读出文件的 1569 字节，问访问哪一个磁盘块？

答：80 号磁盘块

5 对磁盘存在下面五个请求：求！柱面号

答:最少调度次序 :5.3.2.1.和 4

6.有一具有 40 个磁道的盘面，编号为 0-39 ，当磁头位于第n 磁道时，顺序来到如下磁道请求： 磁道号：1 、36 、16 、34 、9 、12 ；试用l ）先来先服务算法FCFS 、2 ) 最短查找时间优先算法SSTF 、3 ）扫描算法 SCAN 等三种磁盘驱动调度算法，计算出它们各自要来回穿越多少磁道？

答：1 ) FCFs 为 111 (2 ) SSTF 为 61 (3 ) SCAN 为 60 （先扫地址大的请求），为 45 （先扫地址小的请求）。

1. 假定磁盘有 200 个柱面，编号O - 199 ，当前存取臂的位置在 143 号柱面上，并刚刚完成了125 号柱面的服务请求，如果请求队列的先后顺序是：86 , 147 , 91 , 177 , 94 , 150 , 102 , 175 , 130 ；试问：为完成上述请求，下列算法存取臂移动的总量是多少？并算出存取臂移动的顺序。

( 1 ）先来先服务算法FCFS;

( 2 ）最短查找时间优先算法 SSTF :

( 3 ）扫描算法SCAN 。( 4 ）电梯调度。

答：( l ）先来先服务算法 FCFS 为 565 ，依次为 143 -86 -147 -91 -177 -94 -150 -102 -175

-130 。( 2 ）最短查找时间优先算法 SSTF 为 162 ，依次为 143 -147 -150 -130 -102 -94 -91

-86 -175 -177 。

( 3 ）扫描算法SCAN 为 169 ，依次为 143 -147 -150 -175 -177 -199 -130 -102 -94 -91 -86 。

( 4 ）电梯调度为 125，依次为 143 -147 -150 -175 -177 -130-102 -94 -91 -86 。

1. 除FCFS 外，所有磁盘调度算法都不公平，如造成有些请求饥饿，试分析：( l ）为什么不公平？( 2 ）提出一种公平性调度算法。（3 ）为什么公平性在分时系统中是一个很重要的指标？ 答：( l ）对位于当前柱面的新请求，只要一到达就可得到服务，但对其他柱面的服务则不然。如SST 下算法，一个离当前柱面远的请求，可能其后不断有离当前柱面近的请求到达而得不到服务（饥饿）。

( 2 ）可划定一个时间界限，把这段时间内尚未得到服务的请求强制移到队列首部，并标记任何新请求不能插到这些请求前。对于 SSTF 算法来说，可以重新排列这些老请求，以优先处理。

( 3 ）可避免分时进程等待时间过长而拉长响应时间。

1. 若磁头的当前位置为 100 柱面，磁头正向磁道号减小方向移动。现有一磁盘读写请求队列，柱面号依次为：190 , 10 , 160 , 80 , 90 , 125 , 30 , 20 , 29 , 140 , 25 ．若采用最短寻道时间优先和电梯调度算法，试计算出各种算法的移臂经过的柱面数？答：采用 SSTF 处理次序为： 100 - 90 一 80 一 125 一 140 一 160 一 190 一 30 一 29 一 5 一 20 一 10 ，总柱面数为：3 10 。采用电梯调度处理次序为：100 - 90 一 80 一 30 一 29 一 25 一 20 。10 一 125 一 140 一 160 一

190 ，总柱面数为：270 。

10 若磁头的当前位置为 100 柱面，磁头正向磁道号增加方向移动。现有一磁盘读写请求队列， 柱面号依次为：23 , 376 , 205 , 132 , 19 , 61 , 190 , 398 , 29 , 4 , 18 , 40 。若采用先来先服务、最短寻道时间优先和扫描算法，试计算出各种算法的移臂经过的柱面数？一

答：采用先来先服务处理次序为：100 一 3 一 376 一 05 一 132 一 19 一 61 一 190 一 398 - 29

-4-18 -40 ，总柱面数为：15960

采用SSTF 处理次序为：100 一 132 一 190 一 05 一 61 -40 一 9 一 3 一 19 一 18 -4 一 376 一

398 ，总柱面数

处理次序为：100 -132 一 190 一 205 一 376 一 398 一 140 一 29 一 23 一 19 一 18 -4，总柱面数

1. 设有长度为L 个字节的文件存到磁带上，若规定磁带物理块长为 B 字节，试问：存放该文件需多少块？( 2 ）若一次启动磁带机交换 K 块，则存取这个文件需执行操作多少次？

( l ）求IJB ，如整除则需”商”个块数，否则为”商＋l ”个块数． ( 2 ）把上述结果再除以 K ，可求出存取这个文件需执行的拍操作次数。

1. 某磁盘共有 200 个柱面，每个柱面有 20 个磁道，每个磁道有 8 个扇区，每个扇区为 1024B ．如

果驱动程序接到访求是读出 606 块，计算该信息块的物理位置。: l ）每个柱面的物理块数为 20 XS = 160 块。

2 ) 606 / 160。得到商为 3 ，余数为 126 。故可知访求的物理位置在：第 3 个柱面（0 柱面开

始编号）的 126 物理块中。

1. 假定磁带记录密度为每英寸 800 字符，每一逻辑记录为 160 个字符，块间隙为 0 . 6 英寸。

今有 1 500 个逻辑记录需要存储，尝试：( 1 ）计算磁带利用率？( 2 ）1500 个逻辑记录占用多

少磁带空间？( 3 ）若要使磁带空间利用率不少于 50 % ，至少应以多少个逻辑记录为一组？

( 1 ）间隙可以存放的字符数是：800 x 0 . 6 = 480 个字符。这时磁带的利用率为：160 / ( 48

+160 ) = 25 %

( 2 ) 1500\* ( 480+160 ) / 800 = 1200 英寸。

( 3 ）设成组块因子为x ，则有： ( 160x ) / ( 480 + 160x ) >=50 %

x >3 ，因而，记录成组的块因子至少为 3 。

1. 假定磁带记录密度为每英寸 800 字符，每一逻辑记录为 200 字符，块间隔为 0 . 6 英寸。现

有 3200 个逻辑记录需要存储，如果不考虑存储记录，则不成组处理和以 8 个逻辑记录为一组的成组处理时磁带的利用率各是多少？两种情况下，3200 个逻辑记录需要占用多少磁带空间？

间隙可以存放的字符数是：800 \*0.6=480 个字符。

( l ）记录不成组时，一个逻辑记录占用一个物理块存储，这时磁带的利用率为：

200 / ( 480+200 )=29 % 占用磁带空间为：3200\* ( 4 800 + 200 ）/800 = 2720 英寸．( 2 ）

记录成组的块因子为 8 时，这时磁带的利用率为：200 \*8 / ( 4 800 + 200 \*8 ) =? 76 . 9 % 占用磁带空间为：3200 /8\*( 480+200\*8）/800 = l 040 英寸。

1. 一个软盘有 40 个柱面，查找移过每个柱面花 6ms ．若文件信息块零乱存放，则相邻逻辑块平均间隔 13 个柱面．但优化存放，相邻逻辑块平均间隔为 2 个柱面．如果搜索延迟为 100ms ，传输速度为每块 25ms ，现问在两种情况下传输 100 块文件信息各需多长时间。

答：非优化存放，读一块数据需要时间为：13 \*6 十 100 十 25 = = 203ms 因而，传输 100 块文件需：2O300ms 。优化存放，读一块数据需要时间为：2\*6 十 100 + 25 = = 137ms 因而，传输

100 块文件需：13700ms 。

16 磁盘请求以 10 、22 、20 、2 、40 、6 、38 柱面的次序到达磁盘驱动器，如果磁头当前位于柱面 20 。若查找移过每个柱面要花 6ms ，用以下算法计算出查找时间：1 ) F CFS , 2 ) 最短查找优先，3 ）电梯调度（正向柱面大的方向）.

答：

1. FCFS 查找时间次序为：20 、10 、22 、2 、40、6、38、、查找时间为 867ms
2. 最短查找优先查找次序为：20 、20 、22 ??10 、6 、2 、38 、40、查找时间为 360ms

3 ）电梯调度查找次序为：20 、20 、22 、38 、40 、10 、6 、2 ，查找时间为：348ms . 17 今假定在某移动臂磁盘上，刚刚处理了访问一信息，并且有下述请求序列等待访问磁盘

75 号柱面的请求，目前正在 80 号柱面读信息,并且有下请求序列等待访问磁盘:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 请求次序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 欲访问的柱面号 | 160 | 40 | 190 | 188 | 90 | 58 | 32 | 102 |

试用：( l ）电梯调度算法( 2 ）最短寻找时间优先算法分别列出实际处理上述请求的次序。答：( l ）电梯调度算法查找次序为：80 、90 、102 、160 、188 、190 、58 、40 、32 ，总柱面数为：268 .

( 2 ）最短查找优先查找次序为：80 、90 、102 、58 、40 、32 、160 、188 、190 总柱面数为：250 。

1. 计算机系统中，屏幕显示分辨率为 640x 480 ，若要存储一屏 256 彩色的图像，需要多少字节存储空间？

答：屏幕信息显示以象素为单位，分辨率为 640x 480 时，屏幕象素有 640X480 = = 300 x 210 个。当用 256 彩色显示时，每个象素用 8 位二进数表示（2 、256 ) ．因而，存储一屏

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

彩色的图像需要：8\*300\*210 位＝300\*210 字节＝= 300K 字节。

1. ?磁盘组共有n 个柱面，编号顺序为O 、1 、2 、„、n-1 ；共有 m 个磁头，编号顺序为 0 、

1 、2 、„、m -1 ：每个磁道内的 k 个信息块从 1 开始编号，依次为 1 、2 、„、k 。现用 x 表示逻辑磁盘块号，用 a , b , c 分别表示任一逻辑磁盘块的柱面号、磁头号、磁道内块号，则 x 与a 力，。可通过如下公式进行转换：

x = k\*m\*a 十k\*b + c

a = = ( x -l ) DIV (K\*M )

b = ( ( x -l ) MOD (K\*m ) ) DIVk

c = ( ( x -l ) MOD (K\*m ) )MOD k + l

若某磁盘组为n = 200 , m =20 , k =10 ，问：

( 1 ）柱面号为 185 ，磁头号为 12 ，道内块号为 5 的磁盘块的逻辑磁盘块号为多少？( 2 ）逻

辑磁盘块号为 1200 ，它所对应的柱面号、磁头号及磁道内块号为多少？( 3 ）若每一磁道内的信息块从。开始编号，依次为。、1 、„ 、k -1 ，其余均同题设，试写出 x 与a 、b 、c 之间的转换公式．

答：( 1 ）由上述公式可知，逻辑磁盘块号 x 为： x = k\*m\*a+k\*b+c =10\*20\*185+120+5= 37125

所以，柱面号为 185 ，磁头号为 12 ，道内块号为 5 的磁盘块的逻辑磁盘块号为：37125 。( 2 ） 由上述公式可知，

a=（X-1 ) DIV ( k \*m )=（1200-l ) DIV ( 10\*20=1199 DIV 200 = 5

b = ( ( x 一 1 ) MOD ( k \* m) ) DIV K=（( 1200 -1 ) MOD ( 10\*20 ) ) DIV 10

= = ( 1199 MOD 200 ) DIV 10 = =199 DIV 10 = 10

c = ( ( x-l ) MOD ( k \*m ) ) MOD k + l = ( ( 1200 一 1 ) MOD ( 10X20 ) ) MOD 10 + 1 =

= ( 1 199 MOD 200 ) MOD 10 + 1 = 199 MOD 10 + l = 9 + l = = 10

所以，逻辑磁盘块号为 1200 ，它所对应的柱面号是 5 、磁头号是 19 及磁道内块号为

( 3 ）转换公式为：

x = k\*m\*a 十k\*b + c + 1 A=(x-1)DIV(k\*m)

b = ( ( x 一 1 ) MOD (k\*m ) ) DIV K?

c = ( ( x 一 1 ) MOD ( k\*m ）MOD k

### 第六章

作者：佚名 来源：网络

1 ．磁带卷上记录了若干文件，假定当前磁头停在第 j 个文件的文件头标前，现要按名读出文件i ，试给出读出文件 i 的步骤。

答：由于磁带卷上的文件用“带标”隔开，每个文件的文件头标前后都使用了三个带标。 正常情况磁头应停在文件头标的前面，所以，只要计算带标的个数，就可找到所要文件。

1. ）当i>=j 时，要正走磁带，

步 1 组织通道程序正走磁带，走过“带标”个数为 3\* ( i –j)个步 2 组织通道程序读文件 i 的文件头标。

步 3 根据文件i 的文件头标信息，组织读文件信息。

1. ）当i < j 时，要反走磁带，

步 1 组织通道程序反走磁带，走过“带标”个数为 3 \*(j-i)个，同时还要后退一块，到达文件 i 头标前。

步 2 组织通道程序读文件 i 的文件头标。

步 3 根据文件i 的文件头标信息，组织读文件信息。

1. 假定令B ＝物理块长、R = ＝逻辑记录长、F ＝块因子。对定长记录（一个块中有整数个逻辑记录），给出计算 F 的公式。

答：F = [B/R]

1. ．某操作系统的磁盘文件空间共有 500 块，若用字长为 32 位的位示图管理盘空间，试问：( 1 ） 位示图需多少个字？( 2 ）第i 字第j 位对应的块号是多少？( 3 ）并给出申谕归还一块的工作流程。

答兰（1 ）位示图占用字数为 500 / 32 = 16 （向上取整）个字。( 2 ）第i 字第j 位对应的块号卜 32\*i + j 。

( 3 ）申请时自上至下、自左至有扫描位示图跳过为 1 的位，找到第一个迁到的 0 位，根据它是第i 字第j 位算出对应块号，并分配出去。归还时已知块号，块号／32 算出第i 字第j 位并把位示图相应位清O 。

1. ．若两个用户共享一个文件系统，用户甲使用文件 A 、B 、C 、D 、E ；用户乙要用到文件 A 、D 、E 、F 。己知用户甲的文件 A 与用户乙的文件 A 实际上不是同一文件；甲、乙两用户的文件D 和E 正是同一文件。试设计一可以采用二级目录或树形目录结构来解决难题。例如，

用户甲文件目录

用户名！文件目录始址

1. ．在 UNIX 中，如果一个盘块的大小为 IKB ，每个盘块号占 4 个字节，即每块可放 256 个地址。请转换下列文件的字节偏移量为物理地址：( l ) 9999 ; ( 2 ) 18000 ; ( 3 )420000 。

答：步 1 将逻辑文件的字节偏移量转换为文件的逻辑块号和块内偏移。方法是：将逻辑文件的字

节偏移量／盘块大小，商为文件的逻辑块号，余数是块内偏移。二步 2 将文件的逻辑块号转换为物理块号。使用多重索引结构，在索引节点中根据氰逻辑块号通过直接索引或间接索引找到对应 物理块号。

（1 、9000 LI ＝INT（9999 , 1024 ) = 9 Bl = MOD（9999 , 1024 ）783

履其逻辑块号为 9 ，故直接索引addrr81 中可找到物理块号。

（2 、15000 L2 = INT ( 15000 , 1024 )17 BI = = MOD（15000 , 1024 ) = = 592 会其逻辑块

号为 17 ，通过一次间接索引 addr [10]中可找到物理块号。

嚓（3 、420000 LI = ＝INT(420000 , 1024 )＝4 10 Bl MOD（9000 , 1024 ）=160 露其逻辑块

号为 410 ，通过二次间接索引 addr [ ll ］中可找到物理块号．

1. ．在UNIX/LINUX 系统中，如果当前目录是／usr / wang ，那么，相对路径为..／ast/xx 文件的绝对路径名是什么？

答：在U NIX/Linux 系统中，" / ' ’表示根目录，" . ”是指当前目录，“..”是指父目录。在本题中当前目录是 lusr / wang，故相对路径为..／ast /xxx 文件实际上是usr 目录下的文件， 故绝对路径名是呀厄叫沁以。

1. ．7 一个UNIX 文件F 的存取权限为：rwxr-x...,该文件的文件主uid = 12 , gid=1 ，另一个用户的uid = = 6 , gid = = 1 ，是否允许该用户执行文件 F

答：F 的存取权限为：rwxr-x...，表示文件主可对F 进行读、写及执行操作，同组用户可

对F 进行读及执行操作，但其他用户不能对 F 操作。因为另一用户的组标识符 gid 相同所以， 允许访问。

8.设某文件为连接文件，由 5 个逻辑记录组成，每个逻辑记录的大小与磁盘块大小相等，均为 512 字节，并依次存放在 50 、121 、75 、80 、63 号磁盘块上。若要存取文件的第 1569 逻辑字节处的信息，问要访问哪一个磁盘块？

1569 / 512 得到商为：3 ，余数为：33 。所以，访问的是互磁盘块的第 33 个字节。

1. 一个UNIX/Linux 文件，如果一个盘块的大小为 1KB ，每个盘块占 4 个字节，那么，若进程欲访问偏移为 263 168 字节处的数据，需经过几次间接？

答：UNIX 口Linux 文件系统中，直接寻址为 10 块，一次间接寻址为 256 块，二次间接寻址为2562 三次间接寻址为 2563 块。 偏移为 263 168 字节的逻辑块号是：263168 / 1024 = = 257 ．块

内偏移量＝263168 -257\*l024 = 0 。由于 10 < 257 < 256 + 10 ，故 263168 字节在一次间接寻

址内．

1. 设某个文件系统的文件目录中，指示文件数据块的索引表长度为 13 ，其中 0 到 9 项为直接寻址方式，后 3 项为间接寻址方式。试描述出文件数据块的索引方式；给出对文件第 n 个字节（设块长 512 字节）的寻址算法．

答：索引表长度为 13 ，其中O 到 9 项为直接寻址方式，后 3 项为一次、二次和三次间接寻址。

步 1 将逻辑文件的字节偏移量转换为文件的逻辑块号和块内偏移。方法是：将逻辑文件的字节偏

移量可盘块大小（5 12 ) ，商为文件的逻辑块号，余数是块内偏移．步 2 将文件的逻辑块号转换为物理块号。使用多重索引结构，在索引节点中根据逻辑块号通过直接索引或间接索引找到对 应物理块号。再判别逻辑块号在 10 块以内或以上，分别采用可直接寻址，一次、二次和三次间接寻址。

1. 设文件ABCD 为定长记录的连续文件，共有 18 个逻辑记录。如果记录长为 5 12B , 物理块长为 1024B ，采用成组方式存放，起始块号为 12 ，叙述第巧号逻辑记录读入内存缓冲区的过程。采用成组方式存放，块因子为 2 。由于共有 18 个逻辑记录，故占用了 9 个物理块，而巧号逻辑

记录占用的是第 1512 = 8 （向上取整）物理块。因为，是连续文件物理块也是连续，所以，该

逻辑记录占用的是 12 + 8 -1 = 19 块．所以，第 15 号逻辑记录读入内存缓冲区的过程如下：根

据块因子，计算占用的相对物理块号 8 ：根据起始块号为 12 ，计算出绝对物理块号 19 ；把物

理块号 19 读入内存缓冲区；把所要的逻辑记录分解出来。

1. 若某操作系统仅支持单级目录，但允许该目录有任意多个文件，且文件名可任意长，试问能 否模拟一个层次式文件系统？如能的话，如何模拟。

答：可以，文件名中可以用插入多个“/ ”来模拟文件分层。例如/usul /datafile/ data1 和

/user/datafil/data2 但在此操作系统中，这些仅仅是包含“/ ' ’的单个文件名。

1. 文件系统的性能取决于高速缓存的命中率，从高速缓存读取数据需要 lms ，从磁盘读取数据需要 40Ins 。若命中率为h ，给出读取数据所需平均时间的计算公式，并画出 h 从 0 到l 变化时的函数曲线。

答:读取数据所需平均时间 T=h\*l + 4\* ( 1 - h ) = h + 40\* ( 1 -h ）。

1. 有一个磁盘组共有 10 个盘面，每个盘面有 100 个磁道，每个磁道有 16 个扇区。若以扇区为

分配单位，现问：答:( 1 ）用位示图管理磁盘空间，则位示图占用多少空间？( 2 ) 若空白文件

目录的每个目录项占 5 个字节，则什么时候空白文件目录大于位示图？( 1 ）磁盘扇区总数为：

10\* 16\* 100 = 16000 个，故位示图占用 16000 / 8 = 2000 字节。( 2 ）己知空白文件目录的

每个目录项占 5 个字节，而位示图占用 2000 字节，也就是说 2000 字节可容纳 400 个文件目录项。当空白文件目录＞400 时，空白文件目录大于位示图。

1. 某磁盘共有 100 个柱面，每个柱面有 8 个磁头，每个盘面分 4 个扇区。若逻辑记录与扇区等

长，柱面、磁道、扇区均从。起编号。现用 16 位的 200 个字（0-199 ）来组成位示图来管理盘

空间。现问：( 1 ）位示图第 15 个字的第 7 位为。而准备分配给某一记录，该块的柱面号、磁

道号、扇区号是多少？( 2 ）现回收第 56 柱面第 6 磁道第 3 扇区，这时位示图的第几个字的第

几位应清 0

( 1 ）位示图第 15 个字的第 7 位对应的块号＝15 \* 16 （字长）+ 7 = 247 ，而块号 247 对应

的：柱面号＝= 247 / ( 8 \*4 )7 （从。编号，向下取整）

磁头号＝( 247 MOD 32 ) / 4 = = 5

扇区号＝= 2 47 MOD 32 MOD =3

( 2 ）块号＝柱面号x 柱面扇区数＋磁道号 x 盘扇区＋盘扇区＝5 6\* ( 8 \*4+6 \*4 + 3 = 1819 字号＝ 1819 /16=113

位号＝= 1819 MOD 16 = = 11

所以，回收第 56 柱面第 6 磁道第 3 扇区时，位示图的第 113 字的第n 位应清 0

1. 如果一个索引节点为 128B ，指针长 4B ，状态信息占用 68B ，而每块大小为 8KB 。问在索引节点中有多大空间给指针？使用直接、一次间接、二次间接和三次间接指针分别可表示多大的文 件？

答：由于索引节点为 128B ，而状态信息占用 68B ，故索引节点中用于磁盘指针的空间大小为： 128-68 = 60 字节。

一次间接、二次间接和三次间接指针占用三个指针项，因而直接指针项数为：60 / 4 -3 = 12 个。每块大小为 8KB ．所以，直接指针时：12 \*8192 = 98304B 。

一次间接指针时：8192 / 4 = 2 048 ，即一个磁盘块可装 2048 个盘块指针，2048\*8192 = 16MB 。二次间接指针时：2048 \* 2048 = 4M ，即二次间接可装 4M 个盘块指针，4M\* 8192 = 32GB 。三次间接指针时：2048 x 2048 \*2048 = 8G ，即三次间接可装 8G 个盘块指针,8G\* 8 192 = 16TB 。

1. ．设一个文件由 100 个物理块组成，对于连续文件、连接文件和索引文件，分别计算执行下列操作时的启动磁盘 I/O 次数（假如头指针和索引表均在内存中）: ( l ）把一块加在文件的开头；( 2 ）把一块加在文件的中间（第 51 块）; ( 3 ）把一块加在文件的末尾；( 4 ）从文件的开头删去一块；( 5 ）从文件的中间（第 51 块）删去一块；( 6 ）从文件的未尾删去一块。答

操作名称 连续文件 链接文件 索引文件

|  |  |  |
| --- | --- | --- |
| 加一块到文件开头 | 201 | 1 1 |
| 加一块到文件中间 | 101 | 51 1 |
| 加一块到文件末尾 | 1 2 | 1 |
| 从文件头删去一块 | 0 1 | 1 |

删去文件中间块 98 52 1

从文件尾删去一块 0 100 1