操作系统期末复习

乔铭宇

2020/12/25

题型

- 选择（30分）
- 填空（10分）
- 大题（60分）

Unit1

- OS作用
- 理解系统调用和库函数
- 多道程序设计的好处
- OS最基本特征：并发和共享
- 程序接口，命令接口

1.1.2 操作系统的作用

1.OS作为用户与计算机硬件系统之间的接口

OS作为用户与计算机硬件系统之间接口的含义是：OS处于用户与计算机硬件系统之间，用户通过OS来使用计算机系统。用户在OS帮助下，能够方便、快捷、安全、可靠地操纵计算机硬件和运行自己的程序。**注意：**OS是一个系统软件，因而这种接口是软件接口。

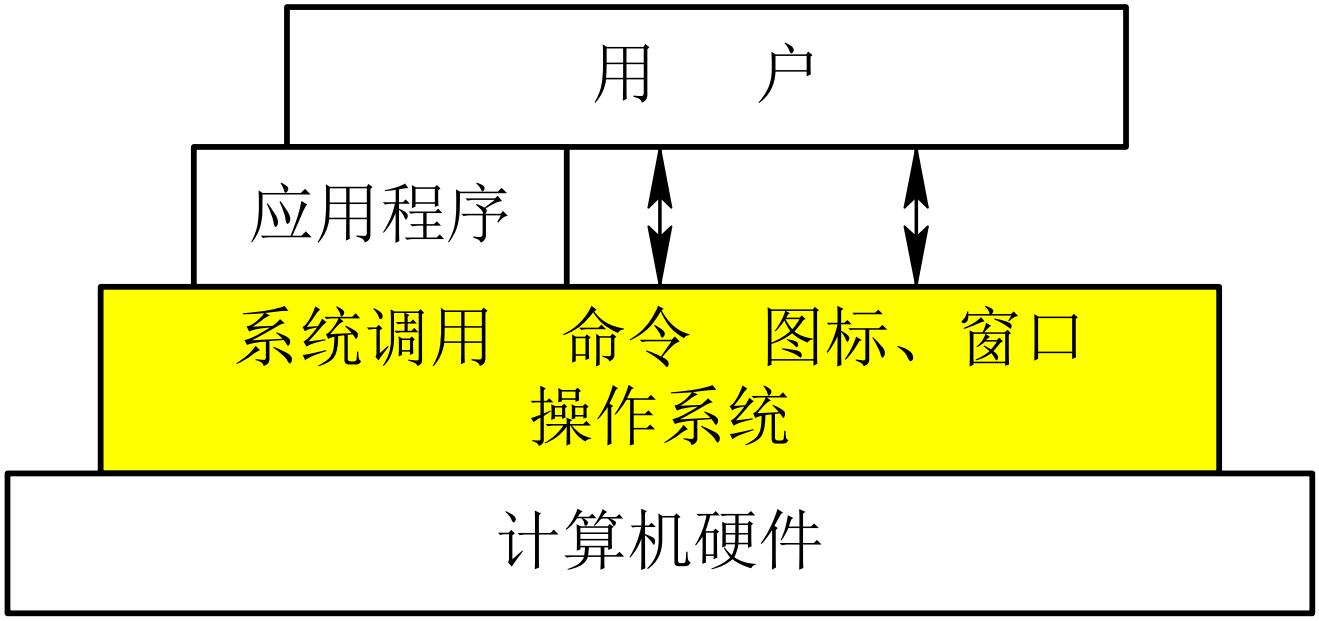


图 1-1 OS作为接口的示意图

1) 命令方式。这是指由OS提供了一组联机命令(语言)， 用户可通过键盘输入有关命令，来直接操纵计算机系统。

2) 系统调用方式。OS提供了一组系统调用，用户可在自己的应用程序中通过相应的系统调用，来操纵计算机。

3) 图形、窗口方式。用户通过屏幕上的窗口和图标来操纵计算机系统和运行自己的程序。

2. OS作为计算机系统资源的管理者

归纳起来可将资源分为四类：处理器、存储器、 I/O设备以及信息(数据和程序)。相应地，OS的主要功能也正是针对这四类资源进行有效的管理，即：

- 处理机管理， 用于分配和控制处理机；
- 存储器管理，主要负责内存的分配与回收；
- **I/O设备管理**，负责I/O设备的分配与操纵；
- 文件管理，负责文件的存取、共享和保护。

3. OS实现了对计算机资源的抽象

对于一台完全无软件的计算机系统(即裸机)，即使其功能再强，也必定是难于使用的。

如果我们在裸机上覆盖一层**I/O设备管理软件**，用户便可利用它所提供的I/O命令，来进行数据输入和打印输出。通常把覆盖了软件的机器称为扩充机器或虚机器。

在OS中引入多道程序设计技术可带来以下好处：

(1) 提高CPU的利用率。

(2) 可提高内存和I/O设备利用率。

(3) 增加系统吞吐量。

例题

【17考研28题】与单道程序系统相比，多道程序系统的优点是（D　　）

Ⅰ.CPU利用率高　　Ⅱ.系统开销小

Ⅲ.系统吞吐量大　　Ⅳ.I/O设备利用率高

A.仅Ⅰ、Ⅲ　　B.仅Ⅰ、Ⅳ

C.仅Ⅱ、Ⅲ　　D.仅Ⅰ、Ⅲ、Ⅳ

例题

【12年考研29题】一个多道批处理系统中仅有P1和P2两个作业，P2比P1晚5ms到达，它们的计算和I/O操作顺序如下：

P1：计算60ms，I/O 80ms， 计算20ms

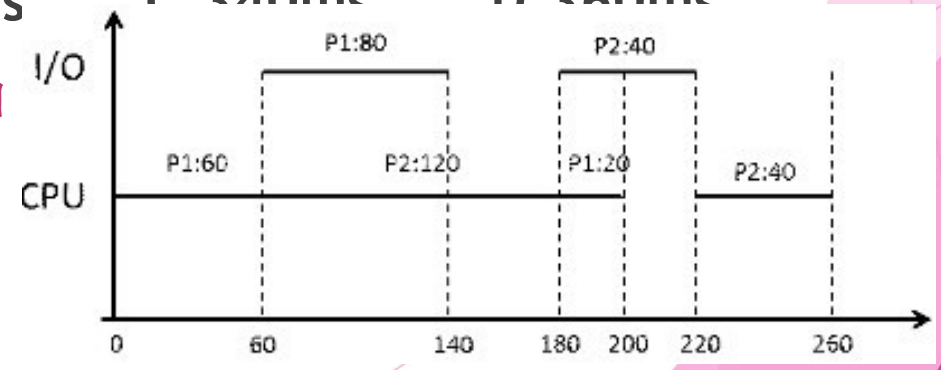
P2：计算120ms，I/O 40ms，计算40ms

若不考虑调度和切换时间，则完成两个作业需要的时间最少是（ B ）

- A.240ms　　B.260ms　　C.340ms　　D.360ms

分析：画出P1和P2的运行甘特图

最少时间为260ms。



3. 多道批处理系统的优缺点

(1) 资源利用率高。

(2) 系统吞吐量大。

(3) 平均周转时间长。

(4) 无交互能力。

4. 多道批处理系统需要解决的问题

(1) 处理机管理问题。

(2) 内存管理问题。

(3) I/O设备管理问题。

(4) 文件管理问题。

(5) 作业管理问题。

1.3 操作系统的基本特性

1.3.1 并发(Concurrence)

并行性是指两个或多个事件在同一时刻发生；

并发性是指两个或多个事件在同一时间间隔内发生。

单处理器多道批处理环境下：

- 多道：内存中同时存放几个作业；
- 宏观上并行运行：都处于运行状态，但都未运行完；
- 微观上串行运行：各作业交替使用CPU；

多个处理机环境下，可以并发执行的程序便可被分配到多个处理机上，实现并行执行

1.3.2 共享(Sharing)

在操作系统环境下，所谓共享是指系统中的资源可供内存中多个并发执行的进程(线程)共同使用。由于资源属性的不同，进程对资源共享的方式也不同，目前主要有以下两种资源共享方式。

1. 互斥共享方式

2. 同时访问方式

1.3 操作系统的基本特性

并发和共享是操作系统的两个最基本的特征，它们又是**互为存在**的条件。

一方面，**资源共享是以程序(进程)的并发执行为条件的**，若系统不允许程序并发执行，自然不存在资源共享问题；另一方面，**若系统不能对资源共享实施有效管理**，协调好诸进程对共享资源的访问，也必然影响到程序并发执行的程度，甚至根本**无法并发执行**。

1.4.5 操作系统与用户之间的接口

1. 命令接口

(1) 联机用户接口。这是为联机用户提供的，它由一组键盘操作命令及命令解释程序所组成。当用户在终端或控制台上每键入一条命令后，系统便立即转入命令解释程序，对该命令加以解释并执行该命令。在完成指定功能后，控制又返回到终端或控制台上，等待用户键入下一条命令。

(2) 脱机用户接口。该接口是为批处理作业的用户提供的，故也称为批处理用户接口。该接口由一组作业控制语言**JCL**组成。批处理作业的用户不能直接与自己的作业交互作用，只能委托系统代替用户对作业进行控制和干预。

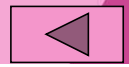
2. 程序接口

该接口是为用户程序在执行中访问系统资源而设置的，是用户程序取得操作系统服务的惟一途径。它是由一组系统调用组成，每一个系统调用都是一个能完成特定功能的子程序。

3. 图形接口

图形用户接口采用了图形化的操作界面， 用非常容易识别的各种**图标(icon)**来将系统的各项功能、各种应用程序和文件，直观地表示出来。用户可用**鼠标或通过菜单和对话框**，来完成对应用程序和文件的操作。

用户完全不必像使用命令接口那样去记住命令名及格式，从而把用户从繁琐且单调的操作中解脱出来。



作 业

- 有三个程序A,B,C，它们使用同一个设备进行IO 操作，并按A,B,C的优先次序进行。这三个程序的计算和IO 操作时间如表1-1所示。假设调度的时间可忽略不计，请分别画出单道程序环境和多道程序环境下（假设内存中可同时装入这三道程序），它们运行的时间关系图，并比较它们的总运行时间。
- 操作系统具有哪几大特征？其中最基本的特征是什么？

程序 操作	A	B	C
计算	30	60	20
I/O	40	30	40
计算	10	10	20

表1-1

Unit2（重点）

- 前驱图
- 进程的三种基本状态（必考）
- 原语，原子调度
- 空闲等待区
- P, V 原语的含义
- 同步，互斥描述前驱图
- 生产者—消费者（填P, V操作伪代码）
- 哲学家就餐，读者写者问题
- P， V原语必考
- 进程通信：通信方式，用P, V原语描述通信过程
- 管道
- 线程，PCB必考
- OS = 数据结构（结构体）+算法
- 感知线程的存在是有PCB存在
- 线程的三个基本状态（没有挂起）
- 第二章作业题

2.1.2 前趋图

前趋图(Precedence Graph)是一个有向无循环图，用于描述进程之间执行的前后关系。

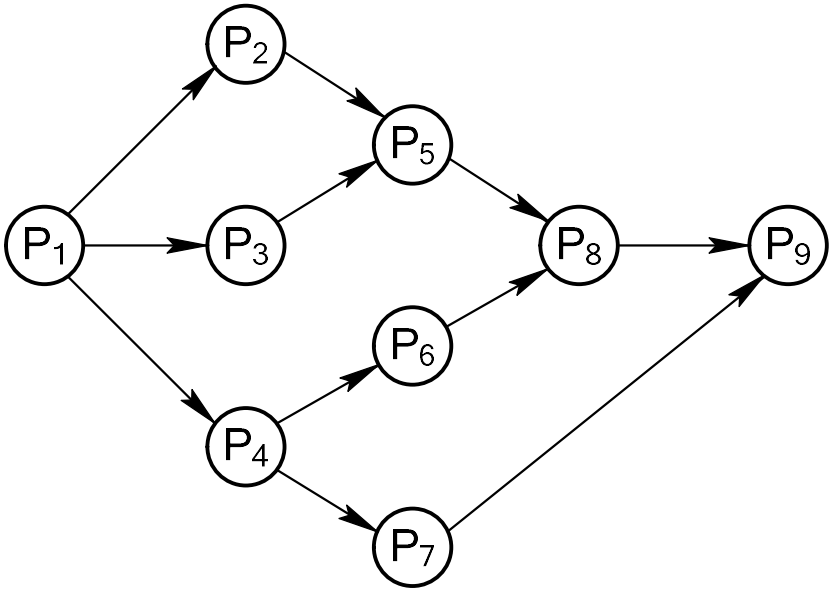
$\rightarrow=\{(P_i, P_j)|P_i$ must complete before P_j may start$\}$，如果$(P_i, P_j)\in\rightarrow$,可写成$\mathbf{P_i\rightarrow P_j}$，称$\mathbf{P_i}$是$\mathbf{P_j}$的直接前趋，而称$\mathbf{P_j}$是$\mathbf{P_i}$的直接后继。

没有前趋的结点称为初始结点(Initial Node)

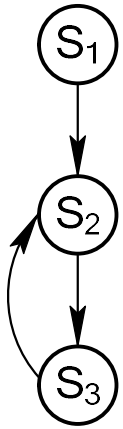
没有后继的结点称为终止结点(Final Node)。

每个结点还具有一个重量(Weight)，用于表示该结点所含有的程序量或结点的执行时间。

$I_i \rightarrow C_i \rightarrow P_i$和$S_1 \rightarrow S_2 \rightarrow S_3$



(a) 具有九个结点的前趋图



具有循环的图

图 2-2 前趋图

对于图 2-2(*a*)所示的前趋图， 存在下述前趋关系：

$P_1 \rightarrow P_2$, $P_1 \rightarrow P_3$, $P_1 \rightarrow P_4$, $P_2 \rightarrow P_5$, $P_3 \rightarrow P_5$, $P_4 \rightarrow P_6$, $P_4 \rightarrow P_7$, $P_5 \rightarrow P_8$, $P_6 \rightarrow P_8$, $P_7 \rightarrow P_9$, $P_8 \rightarrow P_9$

或表示为：

$$P=\{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9\}$$

$\rightarrow$={ (P1, P2), (P1, P3), (P1, P4), (P2, P5), (P3, P5), (P4, P6), (P4, P7), (P5, P8), (P6, P8), (P7, P9), (P8, P9)}

应当注意，前趋图中必须不存在循环，但在图2-2(*b*)中却有着下述的前趋关系：

$$S_2 \rightarrow S_3, S_3 \rightarrow S_2$$

例题

【11年考研32题】有两个并发执行的进程P1和进程P2，共享初值为1的变量x。P1对x加1，P2对x减1。加1和减1操作的指令序列分别如下所示：

```
//加1操作                          //减1操作
load R1,x//取x到寄存器R1中         load R2,x//取x到寄存器R2中
inc R1                            dec R2
store x,R1//将R1的内容存入x        store x,R2//将R2的内容存入x
```

两个操作完成后，x的值（ C ）

- A、可能为-1或3　　B、只能为1
- C、可能为0、1或2　　D、可能为-1、0、1或2

程序和进程的区别

程序	进程
静态的指令序列	进程是一个动态的概念。
程序没有并行特征	进程具有并行特征，是竞争系统资源的基本单位。
一个程序可对应多个进程	一个进程至少对应一个程序在工作（父子进程有可能两个进程对应一个程序）
程序是永久性的软件资源	暂存资源，动态过程

2. 进程的三种基本状态

1) 就绪(Ready)状态

2) 执行状态

3) 阻塞状态

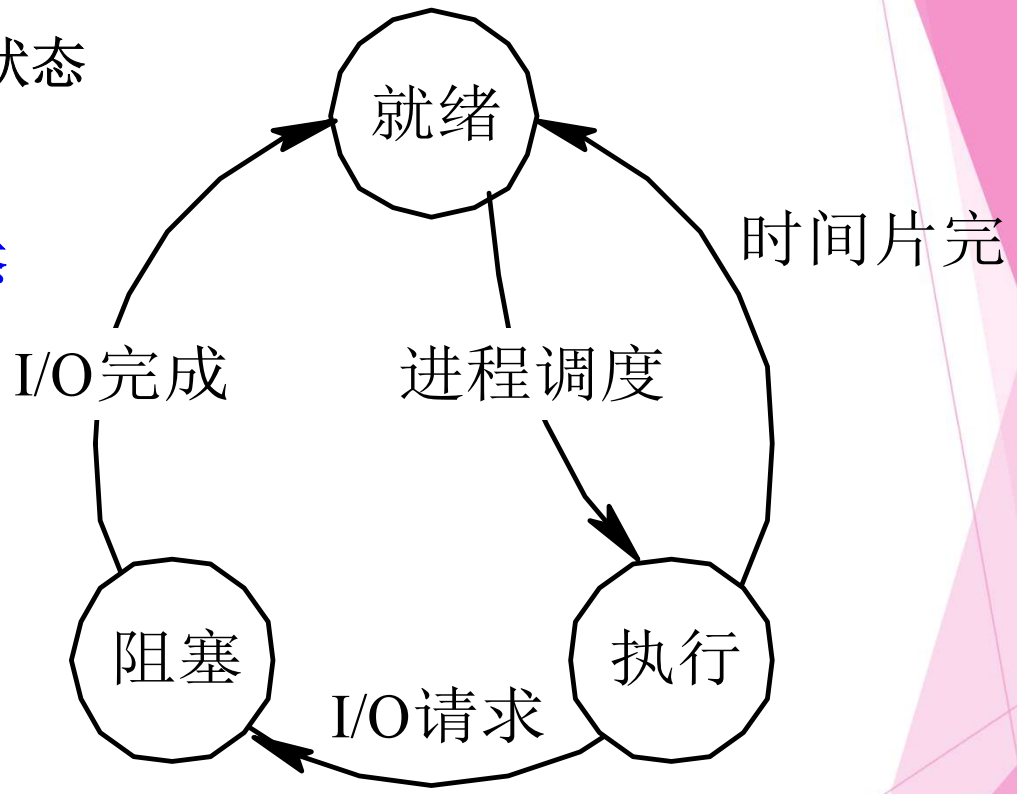
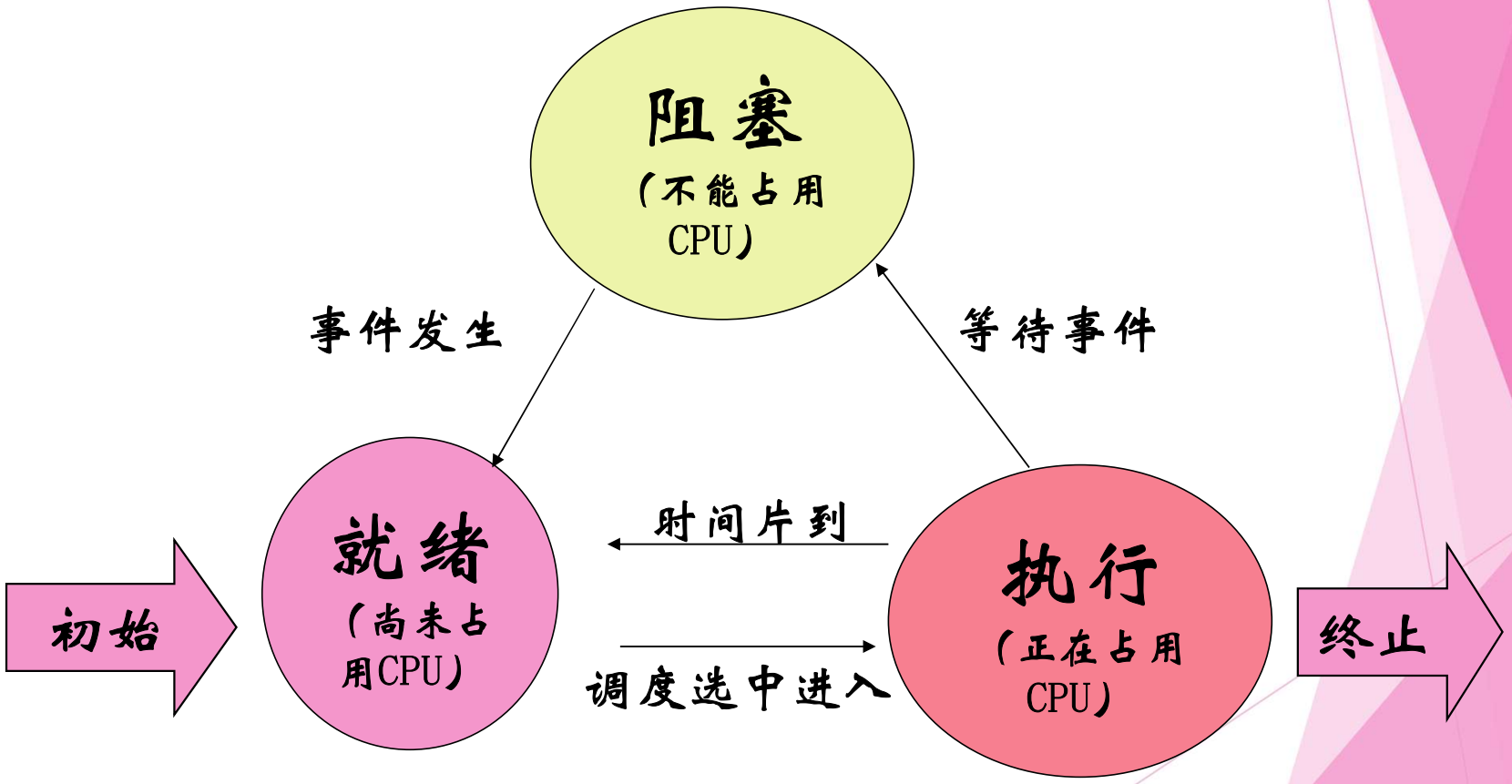


图 2-5 进程的三种基本状态及其转换

进程的五种基本状态及其转换



【15年考研25题】

25．下列选项中会导致进程从执行态变为就绪态的事件是（D）

A．执行P(wait)操作

B．申请内存失败

C．启动I/O 设备

D．被高优先级进程抢占

3. 挂起状态

1) 引入挂起状态的原因

(1) 终端用户的请求。

(2) 父进程请求。

(3) 负荷调节的需要。

(4) 操作系统的需要。

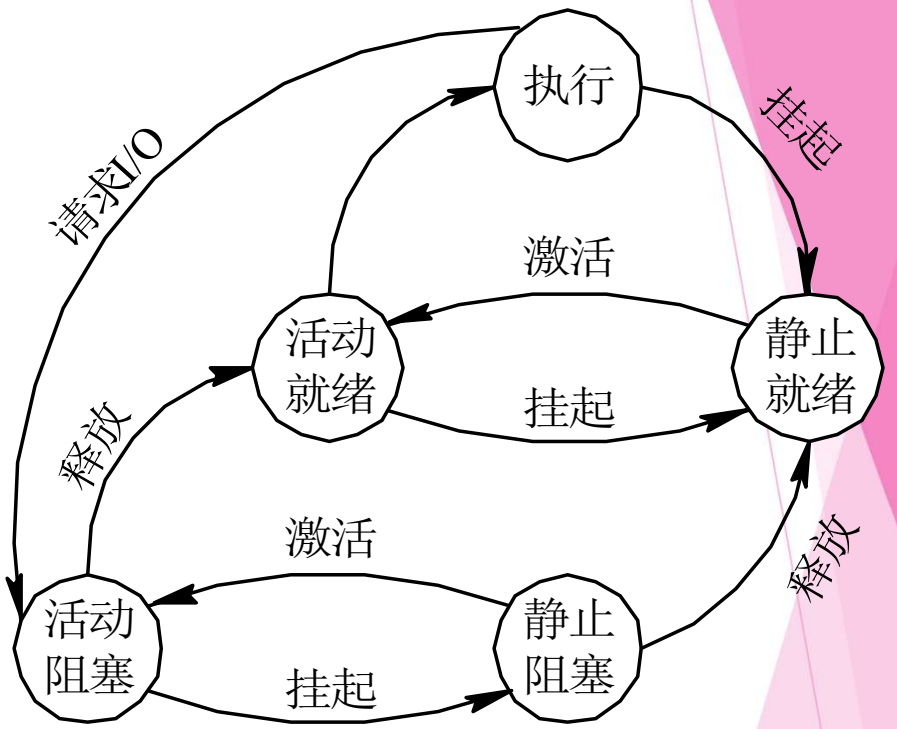


图 2-6 具有挂起状态的进程状态图

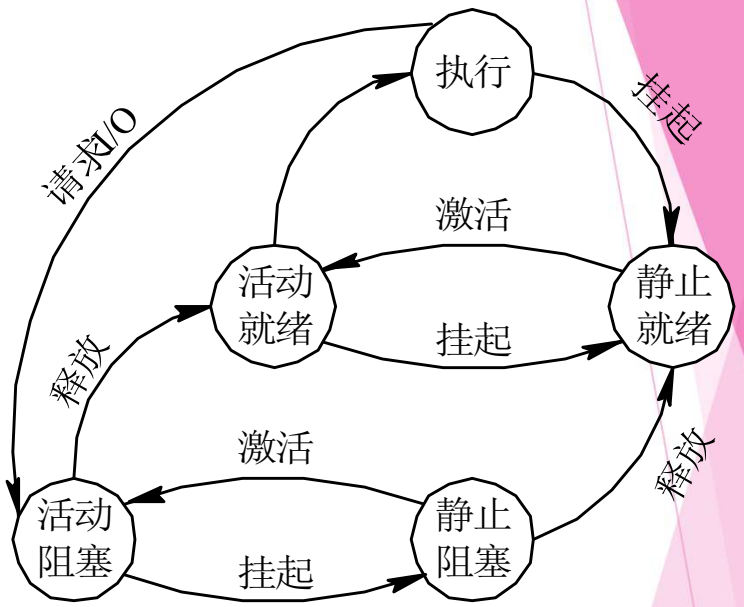
2) 挂起与激活的状态转换

(1) 活动就绪→静止就绪。

(2) 活动阻塞→静止阻塞。

(3) 静止就绪→活动就绪。

(4) 静止阻塞→活动阻塞。



例题

【10年考研24题】下列选项中，导致创建新进程的操作是 C

Ⅰ用户成功登陆 Ⅱ设备分配 Ⅲ启动程序执行

A.仅Ⅰ和Ⅱ B.仅Ⅱ和Ⅲ C.仅Ⅰ和Ⅲ D.Ⅰ，Ⅱ，Ⅲ

分析：设备分配可能引起进程状态的改变，不会创建新进程（对应的设备驱动进程一般处于阻塞状态），而用户登录成功和启动程序执行都会创建新的进程。

★临界资源：系统中一些资源一次只允许一个进程使用，这类资源称为临界资源(critical resources)。

例如：公共变量X

★临界区：进程中访问临界资源的那一段程序，称为临界区(critical region)。例如：程序T1 T2

★互斥：不允许两个以上的共享该资源的并发进程同时进入临界区称为互斥。

例如：进程T1和T2之间的关系，多个进程在竞争使用一台打印机。

★互斥进程使用临界区应遵循的准则：

(1) 空闲让进。当无进程处于临界区时，表明临界资源处于空闲状态，应允许一个请求进入临界区的进程立即进入自己的临界区，以有效地利用临界资源。

(2) 忙则等待。当已有进程进入临界区时，表明临界资源正在被访问，因而其它试图进入临界区的进程必须等待，以保证对临界资源的互斥访问。

(3) 有限等待。对要求访问临界资源的进程，应保证在有限时间内能进入自己的临界区，以免陷入“死等”状态。

(4) 让权等待。当进程不能进入自己的临界区时，应立即释放处理机，以免进程陷入“忙等”状态。

2.2 进程控制

原子操作：一个操作中的所有动作要么全做，要么全不做。

原语的特点：

a、具有独立的系统功能；

b、在系统态运行；

c、不允许中断或不允许并发。

2.2 进程控制

2.2.1 进程的创建

1）进程图(Process Graph)

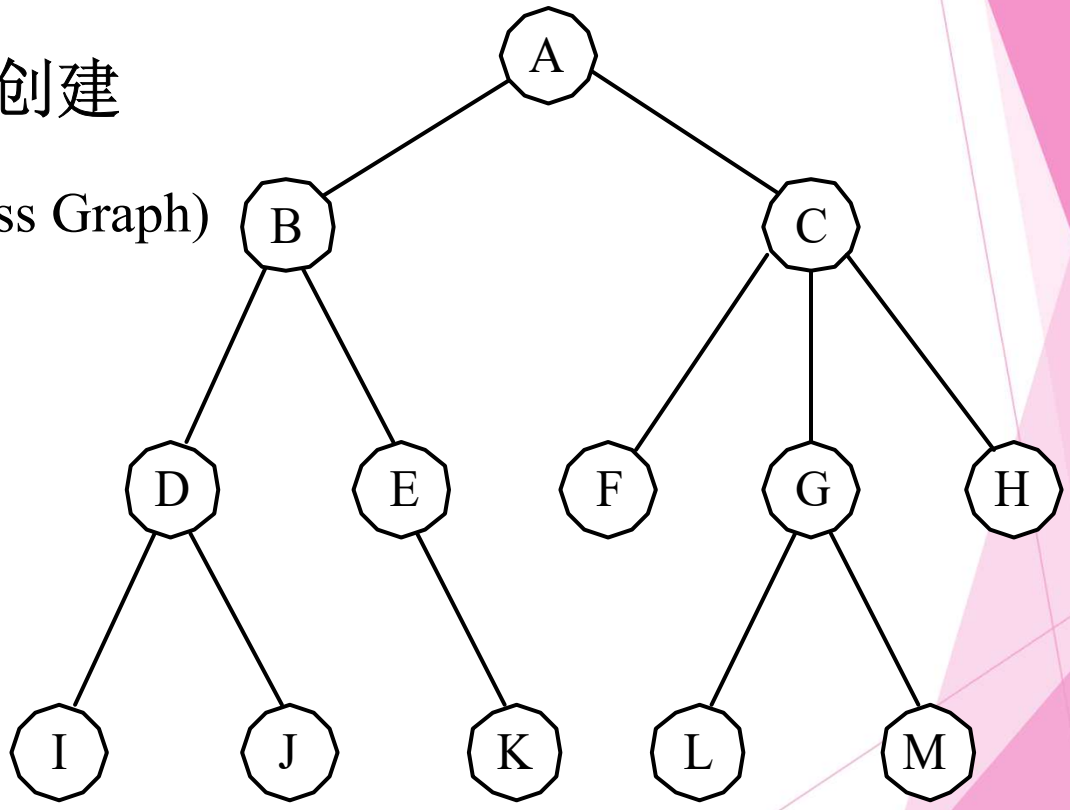


图 2-9 进程树

【16年考研27题】 使用TSL(Test and Set Lock)指令实现进程互斥的伪代码如下所示。

```
do{
……..
while(TSL(&lock));
critical section;
lock=FALSE;
……..
}while(TRUE)
```

下列与实现机制相关的叙述中，正确的是（ B ）

A. 退出临界区的进程负责唤醒阻塞态的进程

B. 等待进入临界区的进程不会主动放弃CPU

C. 上述伪代码满足“让权等待”的同步机制

D. While(TSL(&lock))语句应在关中断下执行

2.3.3 用P、V原语实现进程互斥

★信号量（semaphore）：一种特殊的变量，只能被特殊的操作（即P操作和V操作）使用。它的表面形式是：一个整型变量附加一个队列(等待该信号量的进程队列)。

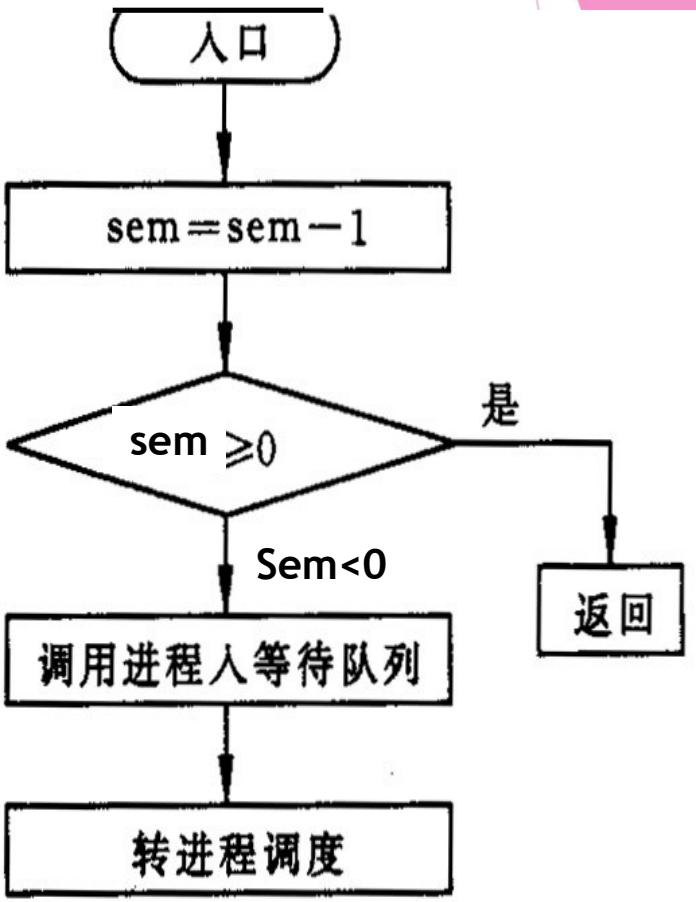
2.3.3 用P、V原语实现进程互斥

★ P原语操作的主要动作：

(1) sem减1；

(2) 若sem减1后仍大于或等于零，则进程继续执行；

(3) 若sem减l后小于零，则该进程被阻塞在与该信号相对应的队列中，然后转进程调度。



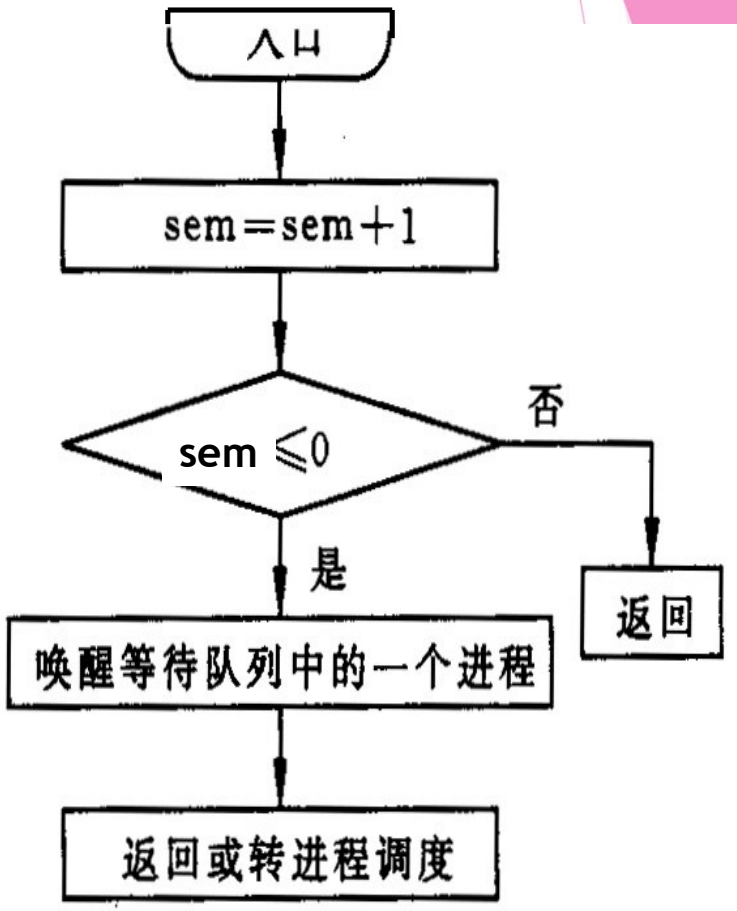
2.3.3 用P、V原语实现进程互斥

★ V原语操作的主要动作：

(1) sem加1；

(2) 若sem加1后大于零，
则进程继续执行；

(3) 若sem加l后小于等于零，
则从该信号的等待队列
中唤醒一等待进程，
然后再返回原进程继续
执行或转进程调度。



2.5.3 用P、V原语实现进程互斥

★ 信号量的取值，通常可以解释为：

1) Sem值的大小表示某类资源的数量。

2) 当Sem>0时，表示尚有资源可分配；

3) 当Sem=0时，表示该资源刚分配完；

4) 当Sem<0时，表示已无资源可分配，

此时Sem的绝对值表示Sem信号量等待队列中进程的数目。

5) 每执行一次P操作，意味着要求分配一个资源；

6) 每执行一次V操作，意味着释放一个资源。

例 题

【19考研24题】下列选项中，可能将进程唤醒的事件是（ C ）

Ⅰ. I/ O 结束

Ⅱ. 某进程退出临界区

Ⅲ . 当前进程的时间片用完

A. 仅 Ⅰ　　B. 仅 Ⅲ

C. 仅 Ⅰ、 Ⅱ　　D. Ⅰ、 Ⅱ 、 Ⅲ

例题

- 【10年考研25题】设与某资源关联的信号量(K)初值为3，当前值为1，若M表示该资源的可用个数，N表示等待资源的进程数，则M,N分别是（ B ）
- A. 0，1　　B. 1，0　　C. 1，2　　D. 2，0

2.3.3 用P、V原语实现进程互斥

★使用P、V操作应注意的事项：

(1) P与V必须成对出现

(2) P、V操作都是原语

2.3.4 同步的概念

同步的消息实现机制

如果对一个消息或事件赋以唯一的消息名，则可用过程：

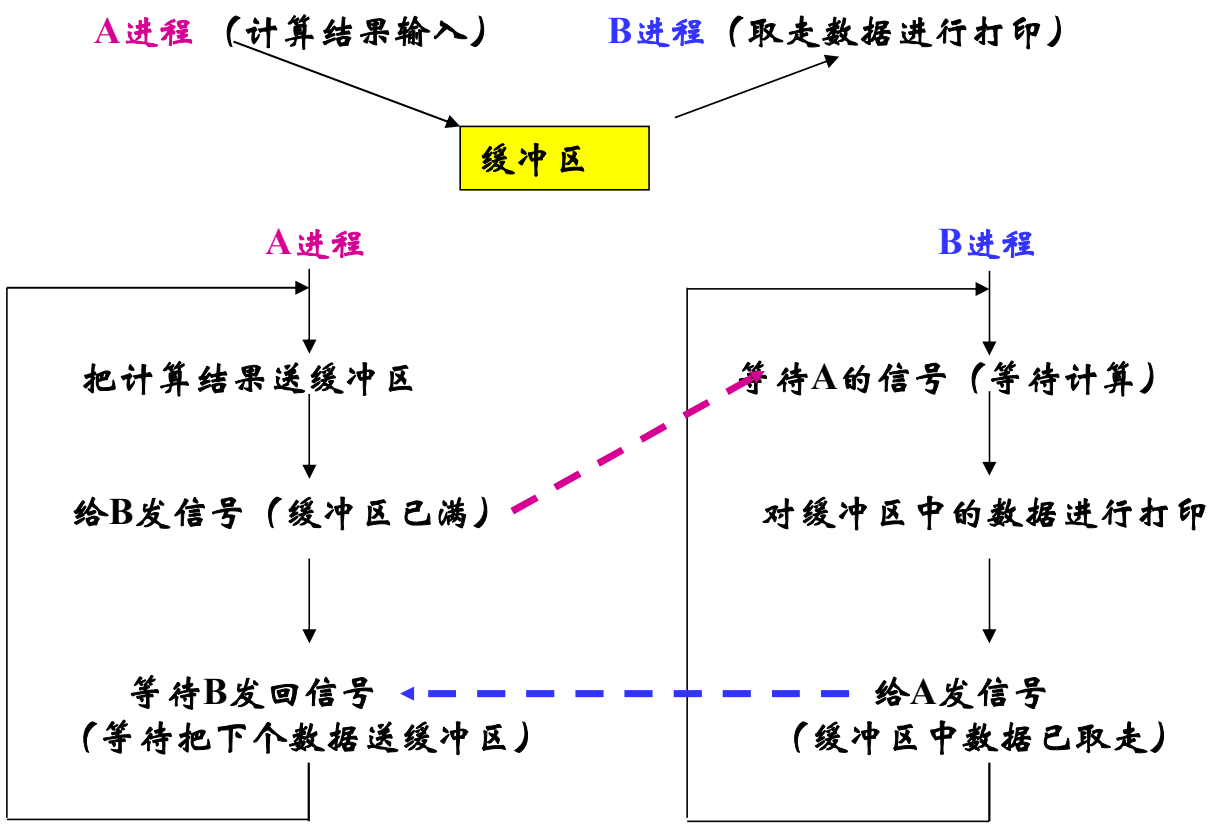
★ wait (消息名)

表示进程等待合作进程发来的消息

★ signal (消息名)

表示向合作进程发送消息。

利用过程wait和signal，可以简单地描述上面例子中的计算进程Pc和打印进程Pp的同步关系如下：


A进程（计算结果输入）
B进程（取走数据进行打印）
缓冲区
A进程
B进程
把计算结果送缓冲区
给B发信号（缓冲区已满）
等待B发回信号
（等待把下个数据送缓冲区）
等待A的信号（等待计算）
对缓冲区中的数据进行打印
给A发信号
（缓冲区中数据已取走）

进程同步示意图

Pc：

A：wait(Bufempty)

计算

Buf←计算结果

Bufempty←false

signal(Buffull)

Goto A

P_P：

B：wait(Buffull)

打印**Buf**中的数据

清除**Buf**中的数据

Buffull←false

signal(Bufempty)

Goto B

过程**wait**的功能是等待到消息名为**true**的进程后继续执行，而**signal**的功能则是向合作进程发送合作进程所需要的消息名，并将其值置为**true**。

利用信号量实现前趋关系

并发执行的进程，如何写这6个进程？

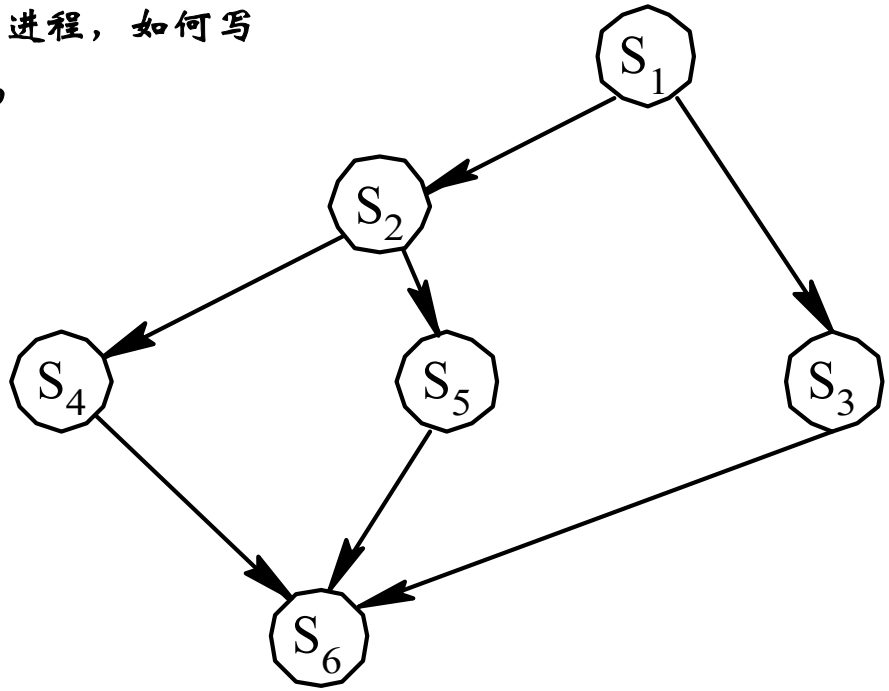


图 2-10 前趋图举例

利用信号量实现前趋关系

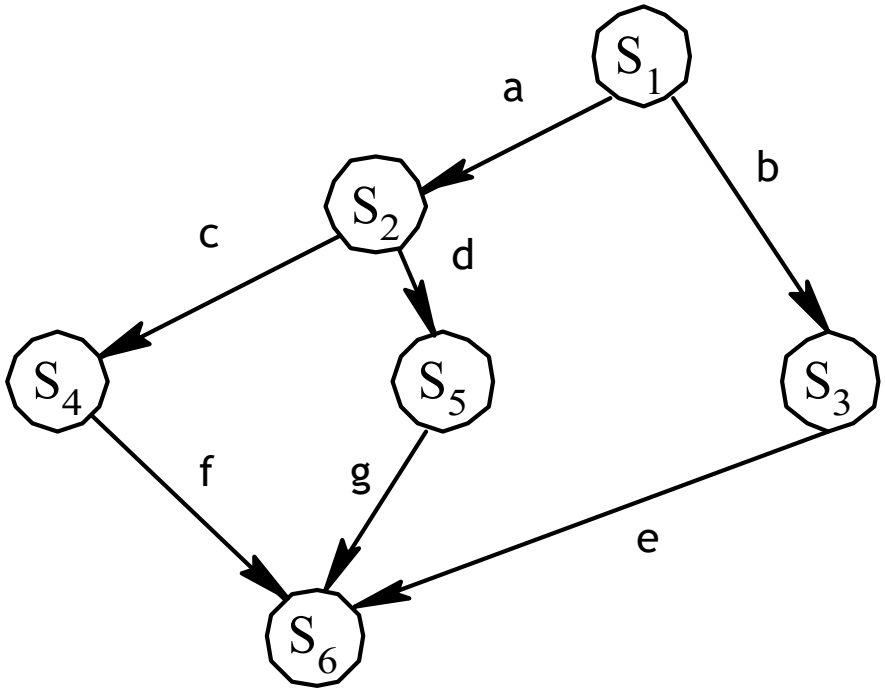


图 2-10 前趋图举例

Var a,b,c,d,e,f,g; semaphore：=0,0,0,0,0,0,0;

begin

parbegin

begin S_1; signal(a); signal(b); end;

begin wait(a); S_2; signal(c); signal(d); end;

begin wait(b); S_3; signal(e); end;

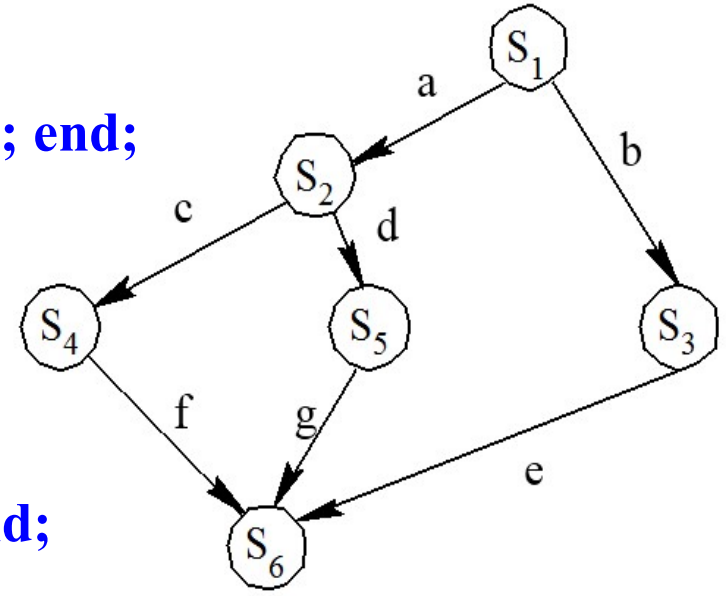
begin wait(c); S_4; signal(f); end;

begin wait(d); S_5; signal(g); end;

begin wait(e); wait(f); wait(g); S_6; end;

parend

end



2.4.1 生产者—消费者问题

★ 把并发进程的同步和互斥问题一般化，可以得到一个抽象的一般模型，即生产者—消费者问题(producer-consumer problems)。

★ 计算机系统中，每个进程都申请使用和释放各种不同类型的资源。

★ 把系统中使用某一类资源的进程称为该资源的消费者

★ 而把释放同类资源的进程称为该资源的生产者。

2.4.1 生产者—消费者问题

把一个长度为n的有界缓冲区(n＞0)与一群生产者进程P_1，P_2，…，P_m和一群消费者进程C_1，C_2，…，C_k联系起来

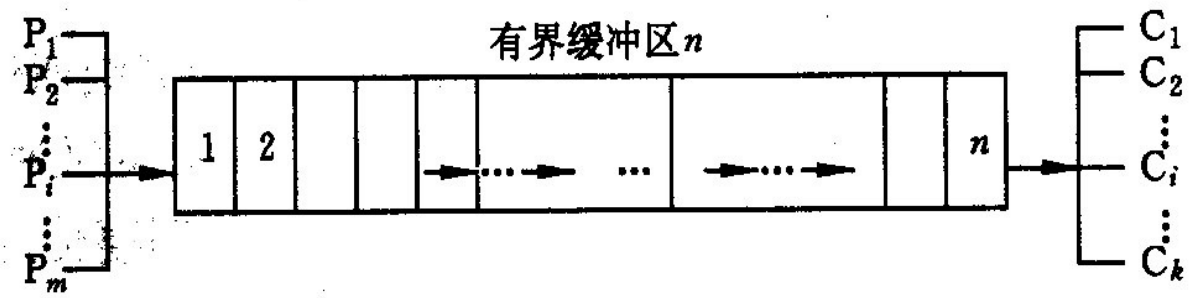


图 3.14　生产者-消费者问题

2.4.1 生产者—消费者问题

设生产者进程和消费者进程是互相等效的，其中，各生产者进程使用的过程**deposit(data)**和各消费者使用的过程**remove(data)**可描述如下：

1. 首先生产者—消费者问题是一个同步问题。即生产者和消费者之间满足如下条件：

1)消费者想接收数据时，有界缓冲区中至少有一个单元是满的

2)生产者想发送数据时，有界缓冲区中至少有一个单元是空的

2. 由于有界缓冲区是临界资源，因此，各生产者进程和各消费者进程之间必须互斥执行。

2.4.1 生产者—消费者问题

★公用信号量mutex，保证生产者进程和消费者进程之间的互斥，表示可用有界缓冲区的个数，初值为1；

★信号量avail为生产者进程的私用信号量，表示有界缓冲区中的空单元个数，初值为n；

★信号量full为消费者进程的私用信号量，表示有界缓冲区中非空单元个数，初值为0。

从而有：

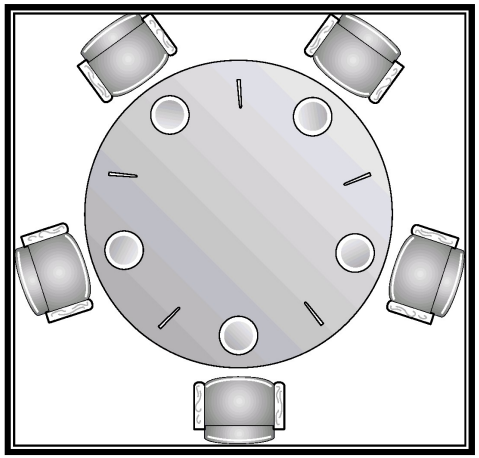
2.4.1 生产者—消费者问题

```
deposit(data)：
  begin
    P(avail)
    P(mutex)
    送数据入缓冲区某单元
    V(full)
    V(mutex)
  end
```

```
remove(data)：
  begin
    P(full)
    P(mutex)
    取缓冲区中某单元数据
    V(avail)
    V(mutex)
  end
```

2.4.2 哲学家就餐问题

- 有五个哲学家围坐在一圆桌旁，桌中央有一盘通心粉，每人面前有一只空盘子，每两人之间放一只筷子
- 每个哲学家的行为是思考，感到饥饿，然后吃通心粉
- 为了吃通心粉，每个哲学家必须拿到两只筷子，并且每个人只能直接从自己的左边或右边去取筷子



解

设fork[5]为5 个信号量，初值为均1, fork[i] 表示i号筷子被拿（i= 0， 1， 2， 3， 4）

```
Philosopheri：
while (1)
{
    思考；
    P(fork[i]);
    P(fork[(i+1) % 5]);
    进食；
    V(fork[i]);
    V(fork[(i+1) % 5]);
}
```

分 析

以上解法会出现死锁，为防止死锁发生可采取的措施：

- 最多允许4个哲学家同时坐在桌子周围
- 仅当一个哲学家左右两边的筷子都可用时，才允许他拿筷子
- 给所有哲学家编号，奇数号的哲学家必须首先拿左边的筷子，偶数号的哲学家则反之。

2.4.2 无死锁哲学家就餐问题 解1

设fork[5]为5 个信号量，初值为均1， fork[i] 表示i号筷子被拿

设信号量S ，初值为4

S用于封锁第5个哲学家

```
Philosopheri：
while (1)
{ 思考；
   P(S)
   P(fork[i]);
   P(fork[(i+1) % 5]);
        进食；
   V(fork[i]);
   V(fork[(i+1) % 5]);
   V(S)
}
```

2.4.2 无死锁哲学家就餐问题 解3

设fork[5]为5 个信号量，初值为均1， fork[i] 表示i号筷子被拿

```
Begin
if i mod 2 == 0
then
{  思考；
   P(fork[i]);
    P(fork[i+1] mod 5)；
    进食；
   V(fork[i]);
   V(fork[i+1mod 5])；
}
```

```
else
{  思考；
   P(fork[i+1 mod 5]);
    P(fork[i])；
    进食；
   V(fork[i+1 mod 5]);
   V(fork[i])；
}
```

2.4.3 读者-写者问题

1. 利用记录型信号量解决读者-写者问题

为实现Reader与Writer进程间在读或写时的互斥而设置了一个**互斥信号量Wmutex**（可写否）。另外，再设置一个整型变量**Readcount**表示**正在读的进程数目**。**由于只要有一个Reader进程在读，便不允许Writer进程去写**。又因为**Readcount**是一个可被多个Reader进程访问的**临界资源**，因此，应该为它设置一个**互斥信号量rmutex**。

读者-写者问题可描述如下：

```
Var rmutex, wmutex: semaphore：  =1,1;
   Readcount:integer：   =0;
   begin
   parbegin
    Reader:begin
      repeat
       wait(rmutex);//临界资源信号量控制读者人数的写
          if readcount=0 then wait(wmutex);   //写操作信号灯，防止写
          Readcount:  =Readcount+1;//读进程个数
       signal(rmutex);
         ...
       perform read operation;
         …
       wait(rmutex); //临界资源信号量  读完时
            readcount :  =readcount-1;//读进程个数
                  if readcount=0 then
       signal(wmutex);    //写操作信号量释放
            signal(rmutex);
           until false;
          end
```

2.4.3 读者-写者问题

```
writer:begin
      repeat
       wait(wmutex);//临界资源信号量操作
            perform write operation;
       signal(wmutex);
      until false;
     end
  parend
end
```

【13年考研45题】某博物馆最多可容纳500人同时参观，有一个出入口，该出入口一次仅允许一个人通过。参观者的活动描述如下：

```
cobegin
参观者进程i：
{
      ”
      进门；
      ”
      参观；
      ”
      出门；
      ”
}
coend
```

请添加必要的信号量和P、V（或wait（）、signal（ ））操作，以实现上述过程中的互斥与同步。要求写出完整的过程，说明信号量的含义并赋初值。

```
定义两个信号量
Semaphore empty = 500;          // 博物馆可以容纳的最多人数
Semaphore mutex = 1;            // 用于出入口资源的控制
参观者进程i;
{
    ”
    P ( empty );
    P ( mutex );
    进门;
    V( mutex );
    参观;
    P ( mutex );
    出门;
    V( mutex );
    V( empty );
    ”
}
coend
```

例题

- 【09年考研45题】三个进程P1、P2、P3互斥使用一个包含N（N>0）个单元的缓冲区。P1每次用produce()生成一个正整数并用put()送入缓冲区某一空单元中；P2每次用getodd()从该缓冲区中取出一个奇数并用countodd()统计奇数个数；P3每次用geteven()从该缓冲区中取出一个偶数并用counteven()统计偶数个数。
- 请用信号量机制实现这三个进程的同步与互斥活动，并说明所定义的信号量的含义。要求用伪代码描述。

```
semaphore  mutex=1;                    //缓冲区操作互斥信号量
semaphore  s1=0,  s2=0;                //奇数、偶数进程的同步信号量
semaphore  empty=N;                    //空缓冲区单元个数信号量

P1:begin
 x=produce();                          //生成一个数
 P(empty);                             //判断缓冲区是否有空单元
 P(mutex);                             //互斥访问缓冲区
 Put();
 If x%2==0
  V(s2);                               //如果是偶数，向P3发出信号
 else
  V(s1);                               //如果是奇数，向P2发出信号
 V(mutex);                             //使用完缓冲区，释放
 end
```

```
P2:begin
 P(s1);                          //收到P1发来的信号，已产生一个奇数
 P(mutex);                       //互斥访问缓冲区
 Getodd();
 Countodd():=countodd()+1;
 V(mutex);                       //释放缓冲区
 V(empty);                       //向P1发出信号，多一个空单元
 end

P3:begin
 P(s2)                           //收到P1发来的信号，已产生一个偶数
 P(mutex);                       //互斥访问缓冲区
 Geteven();
 Counteven():=counteven()+1;
 V(mutex);                       //释放缓冲区
 V(empty);                       //向P1发出信号，多出一个单元
 end.
```

【11年考研45题】某银行提供1个服务窗口和10个供顾客等待的座位。顾客到达银行时，若有空座位，则到取号机上领取一个号，等待叫号。取号机每次仅允许一位顾客使用。当营业员空闲时，通过叫号选取一位顾客并为其服务。请添加必要的信号量，实现下述过程中的互斥与同步。要说明所定义信号量的含义并赋初值。

顾客和营业员的活动过程描述如下：

```
cobegin
{
        process 顾客i
        {
            从取号机获取一个号码;
            等待叫号;
            获取服务;
        }

semaphore seets = 10, // 有10个坐位
mutex = 1, // 取号机互斥信号量
haveCustom = 0; // 顾客与营业员同步
```

```
    process 顾客{
        P(seets); // 等空位
        P(mutex); // 申请使用取号机
        从取号机上取号;
        V(mutex); // 取号完毕
        V(haveCustom); // 通知营业员有新顾客到来
        等待营业员叫号;
        V(seets); // 离开坐位
        接受服务; }
    process 营业员{
        while(True)
        {
            P(haveCustom); // 没有顾客则休息
            叫号;
            为顾客服务;
        }
    }
```

2.5.1 进程的通信方式

★高级通信可分为3种形式：

(1)共享存储区方式；

(2)消息或邮箱机制；

(3) 管道通信。

2.5.1 进程的通信方式

★几种通信方式都可用于大量数据传送，由于其通信方式不同，需要使用不同的控制方式来达到通信进程之间同步或互斥的目的.

2.5.1消息缓冲机制

★设公用信号量mutex为控制对缓冲区访问的互斥信号量，其初值为1。

★设SM为接收进程的私用信号量，表示等待接收的消息个数，其初值为0。

★设发送进程调用过程Send(m)将消息m送往缓冲区，

★ 接收进程调用过程Receive(m)将消息m从缓冲区读往自己的数据区，

★则Send(m)和Receive(m)可分别描述为：

Send(m)：

Begin

向系统申请一个消息缓冲区

P(mutex)

将发送区消息m送入新申请的消息缓冲区

把消息缓冲区挂入接收进程的消息队列

V(mutex)

V(SM)

end

Receive(m)：

begin

P(SM)

P(mutex)

摘下消息队列中的消息m

将消息m从缓冲区复制到接收区

释放缓冲区

V(mutex)

end

【15年考研45题】

45. 有A、B 两人通过信箱进行辩论，每人都从自己的信箱中取得对方的问题。将答案和向对方提出的新问题组成一个邮件放入对方的邮箱中，设A 的信箱最多放M 个邮件，B 的信箱最多放N 个邮件。初始时A 的信箱中有x 个邮件（0<x<m）。辩论者每取出一个邮件，邮件数减1.

A、B 两人操作过程：

Code Begin

A{

While(TRUE){

从A 的信箱中取出一个邮件；

回答问题并提出一个新问题；

将新邮件放入B 的信箱；}

}

B{

While(TRUE){

从B 的信箱中取出一个邮件；

回答问题并提出一个新问题；

将新邮件放入A 的信箱；}

}

Code End

当信箱不为空时，辩论者才能从信箱中取邮件，否则等待。
当信箱不满时，辩论者才能将新邮件放入信箱，否则等待。
请添加必要的信号量和P、V（或wait, signal）操作，以实现上述过程的同步，要求写出完整过程，并说明信号量的含义和初值。

【参考答案】

Semaphore mutexA=1;
Semaphore mutexB=1;
Semaphore emptyA=M-x;
Semaphore emptyB=N;
Semaphore fullA=x;
Semaphore fullB=0;
Code Begin

```
A{
    While(TRUE){
    P(fullA);
    P(mutexA)
    Get a mail from    A_mailbox;
    V(mutexA);
    V(emptyA);
    Answer the question and raise a question;
    P(emptyB);
    P(mutexB)
    send the mail to B;
    V(mutexB);
    V(fullB);
    }
}
```

```
B{
While(TRUE){
    P(fullB);
    P(mutexB)
    Get a mail from B_mailbox;
    V(mutexB);
    V(emptyB);
    Answer the question and raise a question;
    P(emptyA);
    P(mutexA)
    send the mail to A;
    V(mutexA);
    V(fullA);
    }
}
```

【14年考研31题】

31下列关于管道(Pipe)通信的叙述中，正确的是（ C ）

A一个管道可实现双向数据传输

B管道的容量仅受磁盘容量大小限制

C进程对管道进行读操作和写操作都可能被阻塞

D一个管道只能有一个读进程或一个写进程对其操作

2.6.1 为什么要引入线程

线程的属性

(1) 轻型实体。

(2) 独立调度和分派的基本单位。

(3) 可并发执行。

(4) 共享进程资源。

多线程OS中的进程

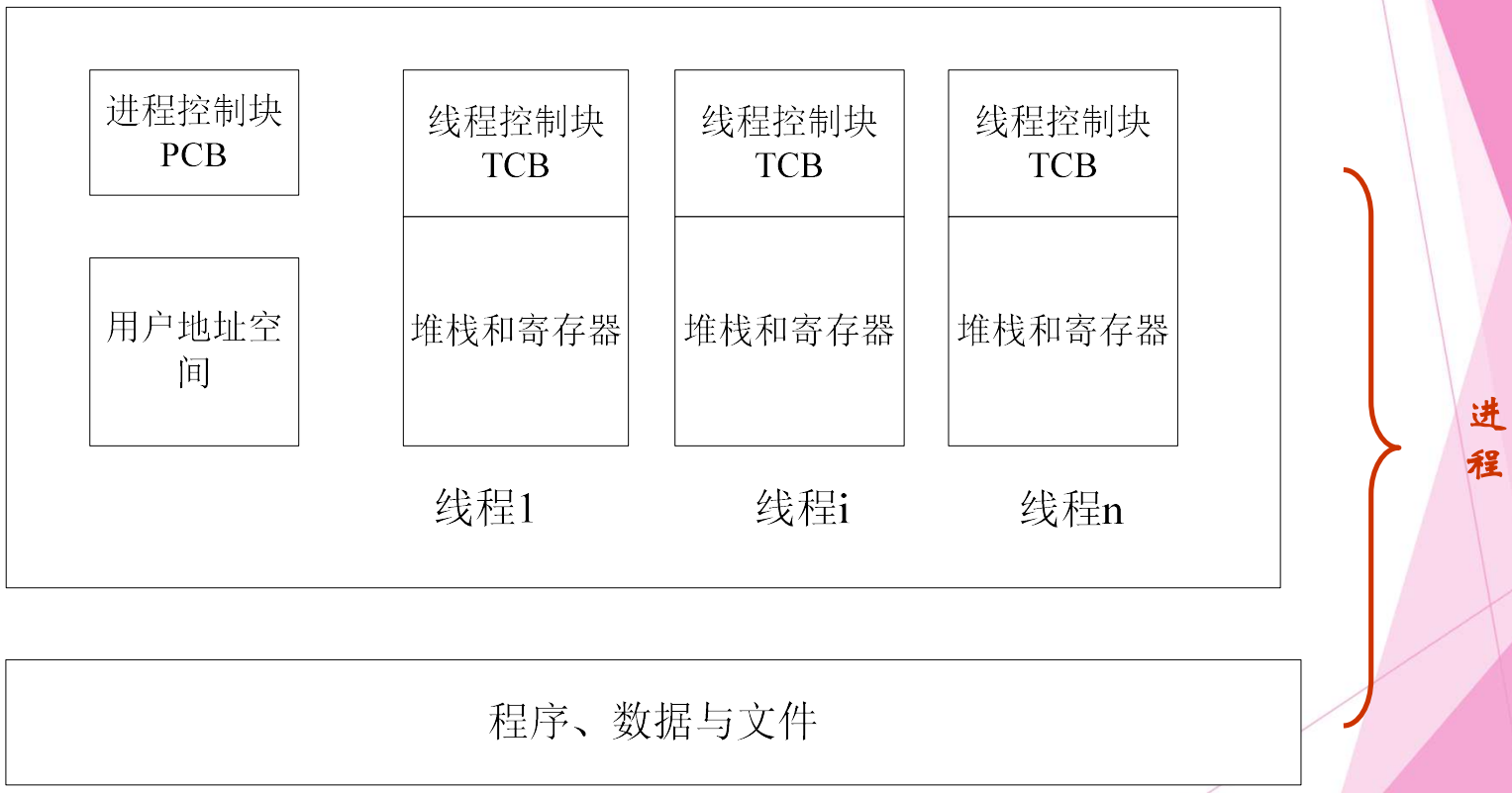
在多线程OS中，进程是作为拥有系统资源的基本单位，通常的进程都包含多个线程并为它们提供资源，但此时的进程就不再作为一个执行的实体。 多线程OS中的进程有以下属性：

(1) 作为系统资源分配的单位。

(2) 可包括多个线程。

(3) 进程*不是一个可执行的实体*。

进程和线程之间的关系


进程控制块
PCB
用户地址空间
线程控制块
TCB
堆栈和寄存器
线程控制块
TCB
堆栈和寄存器
线程控制块
TCB
堆栈和寄存器
线程1
线程i
线程n
进程
程序、数据与文件

2.6.3 进程和线程的区别

★进程是资源分配的基本单位，所有与该进程有关的资源都被记录在PCB中。

线程与资源分配无关，它属于一个进程，并与进程内的其他线程一起共享进程的资源。

★ 进程是抢占处理机的调度单位，它有完整的虚拟地址空间。

线程是进程的一部分，它没有自己的地址空间，同一个进程内的线程共享同一地址空间，线程只有自己的线程控制表TCB。

2.6.3 进程和线程的区别

★进程切换时涉及到进程上下文的保存和恢复，是一个比较庞大的场面。

线程切换时，由于共享进程的资源和地址空间，因此没有地址空间的保存和恢复，减少了系统的开销。

★进程的调度由操作系统内核完成。

线程的调度既可以由系统内核完成也可以由用户程序进行

★一个进程可以拥有一个或多个线程，但一个线程只能隶属于一个进程。

2.7.1 线程的分类

★用户级线程：线程的管理全部由用户程序完成，内核只对进程管理，系统提供一个线程库，提供线程的创建、调度、撤销、通信等功能。

★系统级线程（核心级线程）：提供了线程管理的系统调用，在系统空间执行。

2.7.2 线程的执行特性

- 3个基本状态：执行、就绪和阻塞。没有挂起状态（不会被调出内存）。
  - 挂起状态是由于进程优先级的引入，一些低优先级进程可能等待较长时间，从而被对换至外存。这样做的目的是：提高处理机效率，为运行进程提供足够内存。

2.7.2 线程的执行特性

- 5种基本操作：
  - 派生操作：创建内核态线程clonc、创建用户态线程creat-thread
  - 阻塞：因为等待事情发生阻塞。Block
  - 激活：等待的事件到来放入就绪队列unblock
  - 调度：选择一个线程进入执行状态schedule
  - 结束：结束一个线程finish

线程也存在同步问题

【16年考研30题】 进程P1和P2均包含并发执行的线程，部分伪代码描述如下所示。

```
//进程 P1
    int x=0;
    Thread1()
    {    int a;
    a=1;    x+=1;
    }
    Thread2()
    {    int a;
         a=2; x+=2;
    }
```

```
//进程 P2
    int x=0;
    Thread3()
    {    int a;
    a=x;    x+=3;
    }
    Thread4()
    {    int b;
         b=x; x+=4;
    }
```

下列选项中，需要互斥执行的操作是（ C ）

A. a=1与a=2　　B. a=x与b=x

C. x+=1与x+=2　　D. x+=1与x+=3

例 题

【19考研23题】下列关于线程的描述中，错误的是（ D ）

A.内核级线程的调度由操作系统完成

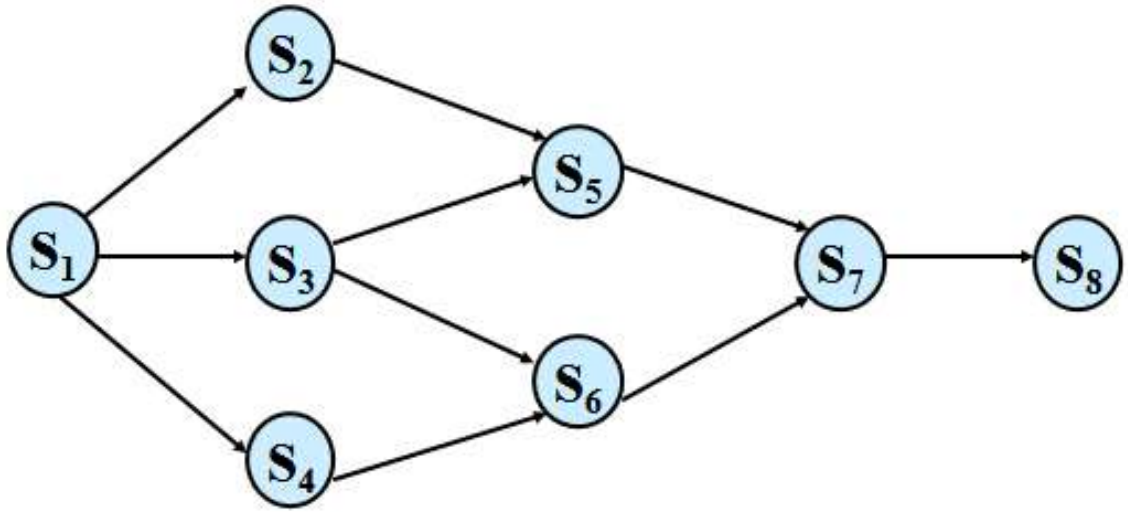
B.操作系统为每个用户级线程建立一个线程控制块

C. 用户级线程间的切换比内核级线程间的切换效率高

D. 用户级线程可以在不支持内核级线程的操作系统上实现

作 业

- 1.利用信号量机制写出相应的程序来描述如下前趋图



- 2. 试 比较。

作 业

- 3.理发店仅有一名理发师，和5个供顾客等待的座位。顾客到达时，若有空座位，则到取号机上领取一个号，等待叫号。取号机每次仅允许一位顾客使用。当理发师空闲时，通过叫号选取一位顾客，并为其服务。请添加必要的信号量和P、V（或wait()、signal()）操作，实现下述过程中的互斥与同步。要求写出完整的过程，说明信号量的含义并赋初值。

```
process 顾客{
    从取号机上取号;
    等待理发师叫号;
    接受服务;
}

  process 理发师{
    while(True)
    {
        叫号;
        为顾客服务;
    }
 }
```

Unit3

- 周转时间序列
- 会调度算法
- 死锁定义
- 解决死锁的方法
- 银行家算法（大题）

例 题

B

- 当作业进入完成状态，操作系统（ ）
  - A 将删除该作业并收回其所占资源，同时输出结果；
  - B收回其所占资源，输出结果，并将该作业的控制块从当前作业队列中删除；
  - C 将收回该作业所占资源并输出结果；
  - D 将输出结果并删除内存中的作业。

作业调度算法的目标和性能衡量

★调度目标：

1）对所有的作业应该是公平合理的。

2）应使设备有较高的利用率。

3）单位时间内执行尽可能多的作业。

4）有快的响应时间。

★由于这些目标的相互冲突，任一调度算法要想同时满足上述目标是不可能的。

作业调度算法的目标和性能衡量

★周转时间=作业完成时间 - 作业提交时间。

$$Ti = Tei - Tsi$$

★平均周转时间：

$$T=\frac{1}{N}\left(\sum_{1}^{n} Ti\right) \quad i=1..n$$

注意：

★一个作业的周转时间说明了它在系统内部停留的时间，应该包括两部分：等待时间和执行时间。

$$Ti = Twi + Tri$$

★ Twi：是作业由后备状态到执行状态的等待时间，不包括作业进入执行状态后的等待时间。

★ Tri ：是作业在执行状态的时间。

作业调度算法的目标和性能衡量

★带权周转时间= $\dfrac{\text{作业的周转时间}}{\text{作业执行时间}}$ **Wi = Ti / Tri**

★如果有多个作业同时进入系统，则平均带权周转时间：

$$W = \frac{1}{n}\left(\sum_{i=1}^{n} W_i\right)$$

★一般来说，作业的平均周转时间短，说明作业在系统的时间短，用户等待的时间短，系统的利用率高，所以，应该选择平均周转时间短的作业调度算法。

【15年考研25题】

25. 下列选项中会导致进程从执行态变为就绪态的事件是（ D ）

A. 执行P(wait)操作

B. 申请内存失败

C. 启动I/O 设备

D. 被高优先级进程抢占

【13年考研28题】

下列选项中，会导致用户进程从用户态切换到内核态的操作是（B）

I. 整数除以零　II. sin()函数调用　III. read 系统调用

A. 仅I、II　B. 仅I、III　C. 仅II、III　D. I、II和III

进程调度的时机

★抢占方式：在就绪队列中一旦有优先级高于当前执行进程的进程存在便立即发生进程调度，转让处理机。

★而非抢占方式即使在就绪队列存在有优先级高于当前执行进程时，当前进程仍将继续占有处理机，直到该进程自己因调用原语操作或等待I/O而进入阻塞状态，或时间片用完时才重新发生调度让出处理机。

进程调度性能评价

★进程调度策略的好坏直接影响作业调度的性能。

★作业调度性能评价

- 周转时间
- 平均周转时间
- 带权周转时间
- 平均带权周转时间

3.2 调度算法

- 先来先服务
- 轮转法
- 多级反馈轮转法
- 优先级法
- 最短作业优先法
- 最高响应比优先法

3.2 调度算法

★作业调度算法—FCFS（First come first serve）

- **思想**：按作业和就绪进程到来的次序进行调度。这种算法优先考虑在系统中等待时间最长的作业，而不管它要求运行时间的长短。
- **优点**：算法简单，公平，容易实现
- **缺点**：对于短作业或短进程，等待时间长

3.2 调度算法

★作业调度算法—FCFS

下面是4个作业在系统中从提交、运行的信息。

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	8	2	8	10	2	1
2	8.5	0.5	10	10.5	2	4
3	9	0.1	10.5	10.6	1.6	16
4	9.5	0.2	10.6	10.8	1.3	6.5

平均周转时间：T=1.725　平均带权周转时间W=6.875

3.2 调度算法

★短作业优先算法—SJF （shortest job first）

- **思想**：比较作业缓冲区中的作业预计的运行时间，选择**预计时间最短的作业**进入运行状态。
- **优点**：算法简单，可得到最大系统吞吐率，效率高。
- **缺点**：主要问题是对长作业不利，如果系统不断地接收短作业，就会使长作业长时间等待。

3.2 调度算法

★短作业优先算法—SJF

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	8	2	8	10	2	1
2	8.5	0.5	10.3	10.8	2.3	4.6
3	9	0.1	10	10.1	1.1	11
4	9.5	0.2	10.1	10.3	0.8	4

平均周转时间：T=1.55　平均带权周转时间W=5.15

3.2 调度算法

★最高响应比优先—HRN（highest response-ratio next）

响应比=响应时间/预计执行时间

- 响应时间=等待时间+预计执行时间
- 所以响应比为：**1+作业等待时间/预计执行时间**

- **思想**：当需要从就绪队列中选择进程投入运行时，先计算每个进程的响应比，选择响应比最高的进程运行
- **优点**：短作业响应比高，执行时间短；长作业响应比随着等待时间增加而提高，不会过长等待。既照顾了短作业、也考虑到了长作业。
- **缺点**：每次调度前计算响应比增加了系统开销。

3.2 调度算法

★最高响应比优先—HRN

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	8	2	8	10	2	1
2	8.5	0.5	10.1	10.6	2.1	4.2
3	9	0.1	10	10.1	1.1	11
4	9.5	0.20	10.6	10.8	1.3	6.5

平均周转时间：**T=1.625 W=5.675**

3.2 调度算法

★优先级法 HPF（highest priority first）

- **算法描述**：根据分配给进程的优先数来决定运行进程。
- **算法的核心**：是确定进程或作业的优先级
  - **静态法**：静态优先权是在创建进程时确定的，且在进程的整个运行期间保持不变。
  - **动态法**：动态优先权是指，在创建进程时所赋予的优先权可以随进程的推进或随其等待时间的增加而改变的，以便获得更好的调度性能。

特点：静态优先级法简单，但缺点是公平性差，可能会造成优先级低的长期等待；动态优先级法资源利用率高，公平性好，缺点是系统开销较大，实现复杂。

3.2 调度算法

★优先级法 HPF（highest priority first）

- 静态法
  - 作业调度确定优先级原则
    - 由用户根据作业的紧急程度输入一个适当的优先级；
    - 由系统或操作员根据作业的类型确定；
    - 系统根据作业要求的资源确定优先级。
  - 进程调度确定优先级原则
    - 按照进程的类型确定；
    - 将作业的优先级作为它所属进程的优先级。

3.2 调度算法

★优先级法 HPF（highest priority first）

- 动态法优先级确定原则
  - 根据进程占有CPU时间长短确定
    - 占用的时间越长，下次调度的优先级越低；
    - 占用的时间越短，下次调度的优先级越高。
  - 根据就绪进程等待CPU的时间长短确定
    - 等待时间越长，优先级越高；
    - 等待时间越短，优先级越低。

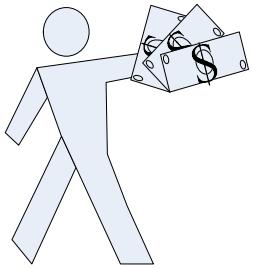
3.2 调度算法

★轮转法 RR（round robin）

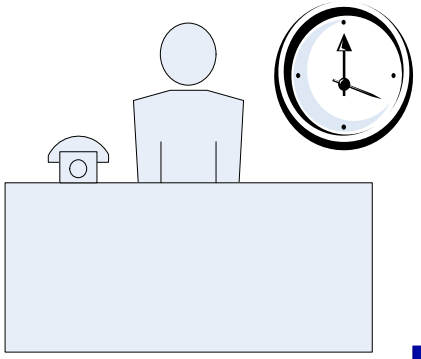
- 算法描述：将CPU的处理时间分成固定大小的时间片。如果一个进程在被调度选中之后用完了系统规定的时间片，但未完成要求的任务，则它自行释放自己所占有的CPU而排到就绪队列的末尾，等待下一次调度。

优点：具有公平性；易于实现，算法简单；

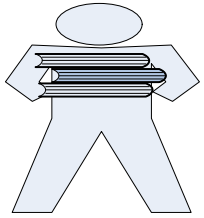
缺点：CPU存在较大额外开销，用于进程切换和调度。



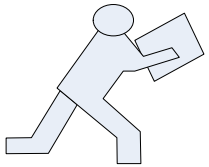
就该我了，把东西都掏出来…



老师验机



时间片用完，先把东西收起来
等下一次召见！



就绪，排好队！

就绪，排好队！

1、每人自带笔记本找老师验机
2、每人只给10分钟，验不完回去重新排队
3、验机前准备好所有的程序代码、文档

3.2 调度算法

★轮转法 RR（round robin）

- 时间片：时间片长度的选择会直接影响系统开销和响应时间。
  - 太长，则使每一个进程均能在一个时间片内完成，RR算法褪化成了FCFS；太短，导致频繁的时间片中断和调度，CPU额外开销大。

计算表达式：q＝R/Nmax

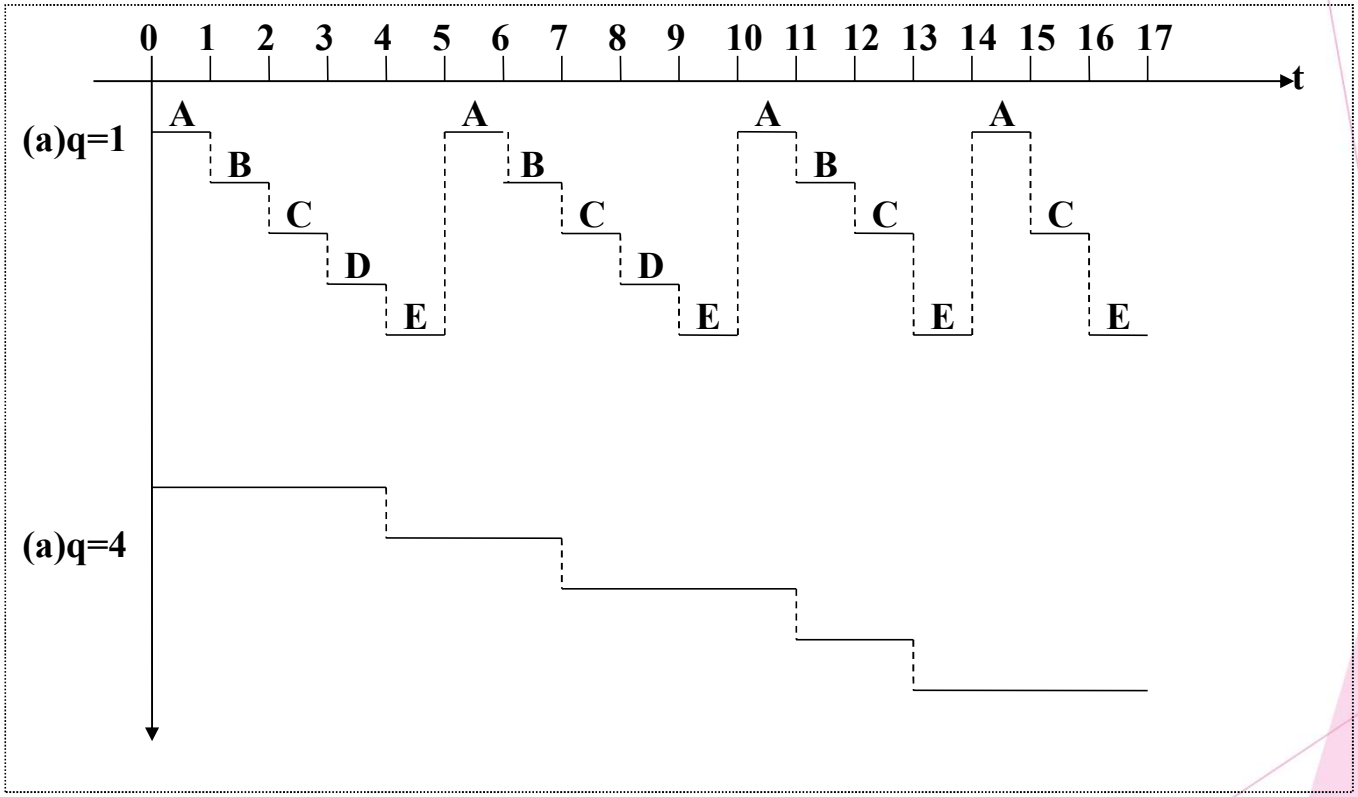
q：时间片长度；

R：系统对响应时间的要求

Nmax：就绪队列要求的最大进程数量

3.2 调度算法

★轮转法 RR（round robin）



q=1和q=4的进程运行情况

3.2 调度算法

★轮转法 RR（round robin）

作业情况 / 时间片	进程名	A	B	C	D	E	平均
	到达时间	0	1	2	3	4	
	服务时间	4	3	4	2	4	
RR q=1	完成时间	15	12	16	9	17	
	周转时间	15	11	14	6	13	11.8
	带权周转时间	3.75	3.67	3.5	3	3.33	3.26
RR q=4	完成时间	4	7	11	13	17	
	周转时间	4	6	9	10	13	8.4
	带权周转时间	1	2	2.25	5	3.33	2.5

3.2 调度算法

★多级反馈轮转法

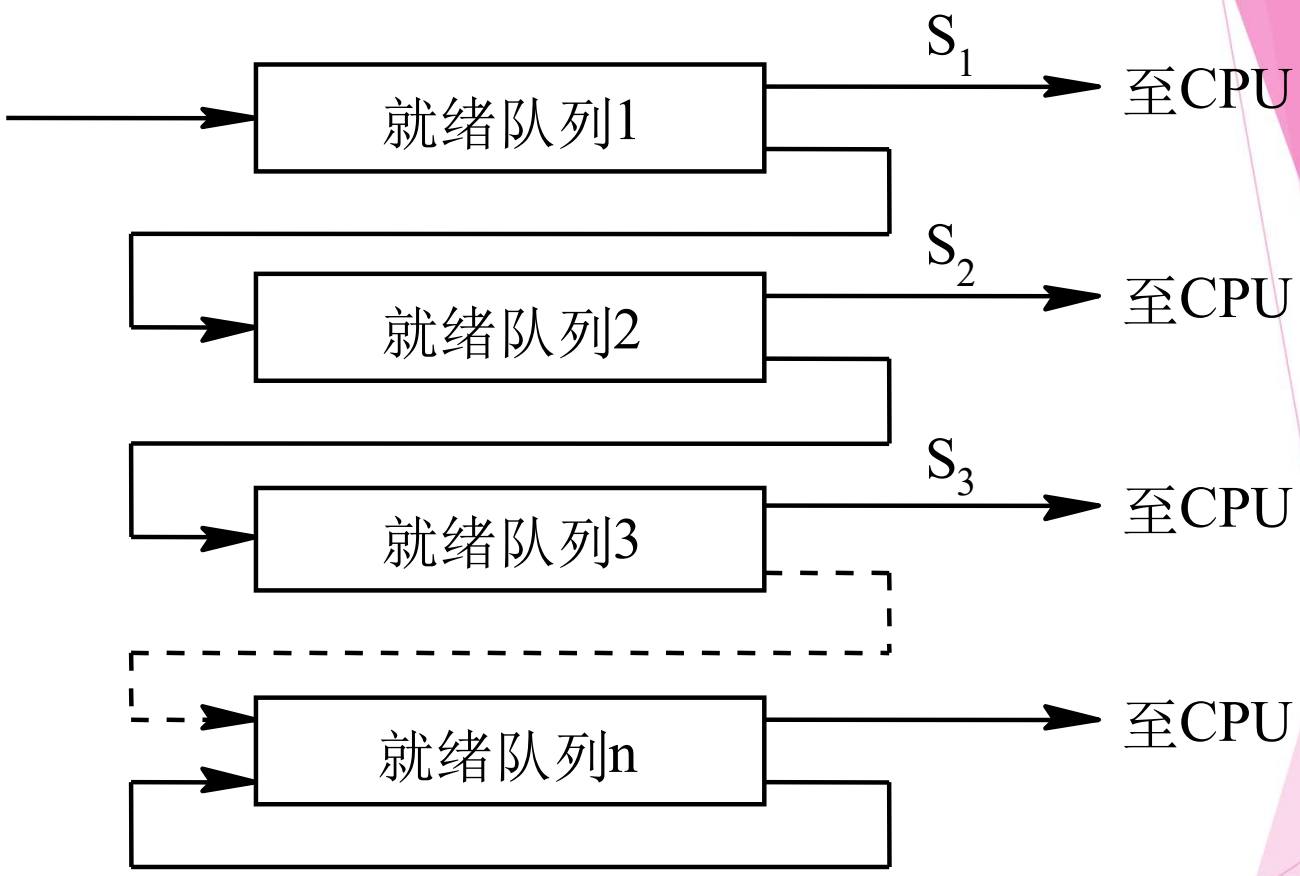
- **算法描述：**把就绪队列按照进程到达就绪队列的类型和进程被阻塞时的阻塞原因分成不同的就绪队列，每个队列按FCFS原则排列，各队列之间的进程享有不同的优先级，但同一队列内优先级相同。
- **多级反馈轮转法与优先级法在原理上的区别是**，一个进程在它执行结束之前，可能需要反复多次通过反馈循环执行，而不是优先级法中的一次执行。

特点：复杂，实现困难；是**FCFS，RR，HPF**的综合应用。

3.2 调度算法

★多级反馈轮转法

- 设置**多个就绪队列**，并为各个队列赋予不同的优先级。
  - 第一个队列的优先级最高，
  - 第二个队列次之，
  - 其余各队列的优先权逐个降低。
  - 该算法赋予各个队列中进程执行时间片的大小也各不相同，**在优先权愈高的队列中，为每个进程所规定的执行时间片就愈小**。例如，第二个队列的时间片要比第一个队列的时间片长一倍，……，第$i+1$个队列的时间片要比第i个队列的时间片长一倍。



（时间片：$S_1 < S_2 < S_3$）

多级反馈队列调度算法

3.2 调度算法

- 算法的入队原理
  - 当一个新进程进入内存后，首先将它放入第一队列的末尾，按FCFS原则排队等待调度；
  - 当轮到该进程执行时，如它能在该时间片内完成，便可准备撤离系统；
  - 如果它在一个时间片结束时尚未完成，调度程序便将该进程转入第二队列的末尾，再同样地按FCFS原则等待调度执行；
  - 如果它在第二队列中运行一个时间片后仍未完成，再依次将它放入第三队列，……，如此下去，当一个长作业(进程)从第一队列依次降到第n队列后，在第n队列中便采取按时间片轮转的方式运行。

例题

【09年考研24题】下列进程调度算法中，综合考虑进程等待时间和执行时间的是(　　)　D

A. 时间片轮转调度算法　B.短进程优先调度算法

C.先来先服务调度算法　D.高响应比优先调度算法

【11年考研23题】下列选项中，满足短任务优先且不会发生饥饿现象的是(　　)调度算法　B

A. 先来先服务　　B. 高响应比优先

C. 时间片轮转　　D. 非抢占式短作业优先

【14年考研23题】

下列调度算法中，不可能导致饥饿现象的是(A)

A时间片轮转

B静态优先数调度

C非抢占式短作业优先

D抢占式短作业优先

【17考研27题】

下列有关基于时间片的进程调度的叙述中，错误的是（ B ）

A.时间片越短，进程切换的次数越多，系统开销也越大

B.当前进程的时间片用完后，该进程状态由执行态变为阻塞态

C.时钟中断发生后，系统会修改当前进程在时间片内的剩余时间

D.影响时间片大小的主要因素包括响应时间、系统开销和进程数量等。

3.5.1 产生死锁的原因

★死锁的定义：是指各并发进程彼此互相等待对方所拥有的资源，并且这些并发进程在得到对方的资源之前不会释放自己所拥有的资源，造成并发进程都无法向前推进，称这种现象为死锁现象。

★死锁的原因：

(1) 竞争资源。

(2) 进程间推进顺序非法。

3.5.1 产生死锁的原因

1. 竞争资源引起进程死锁

1) 竞争不可抢占性资源

2) 竞争可消耗资源

2. 进程推进顺序不当引起死锁

1) 进程推进顺序合法

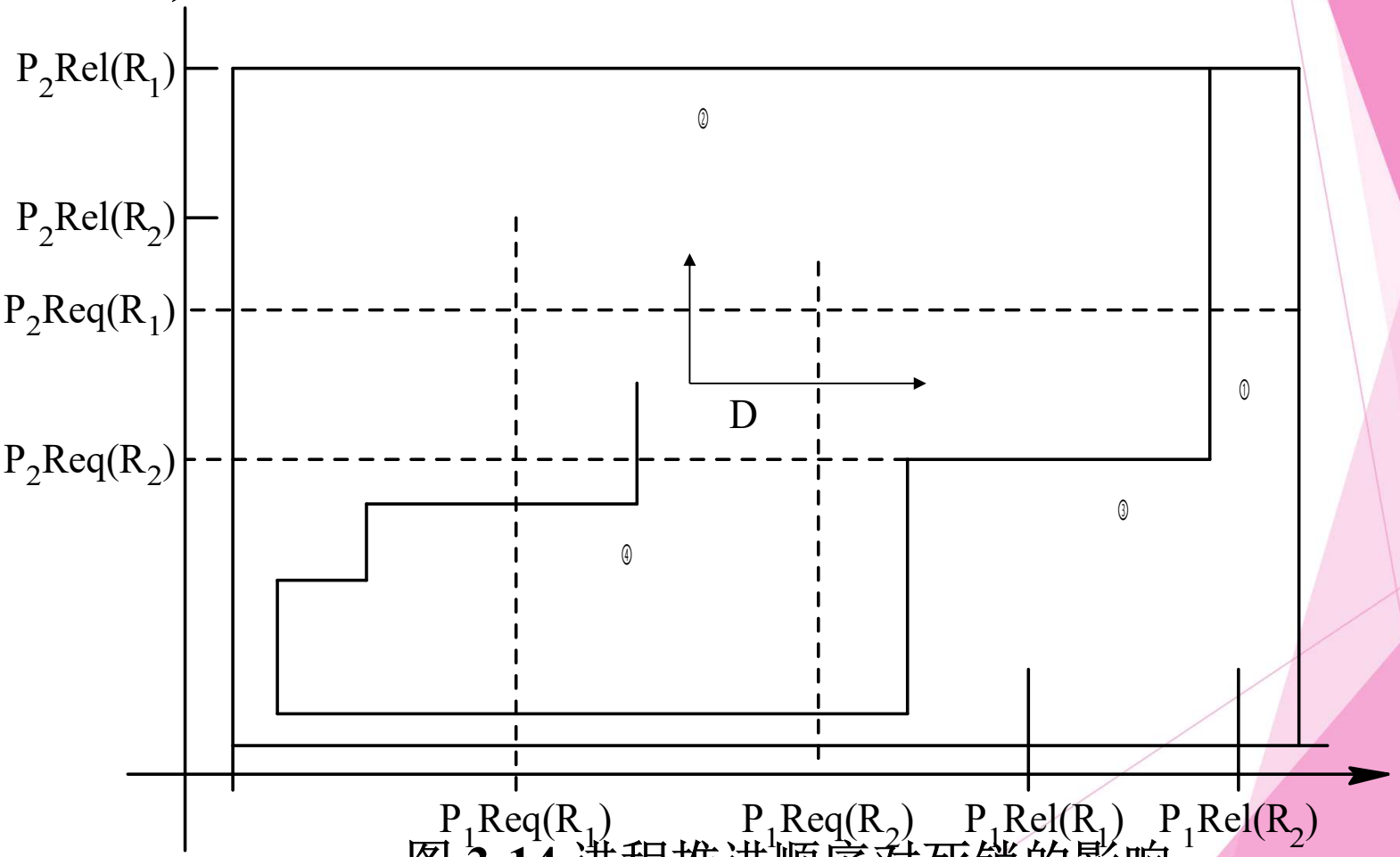


图 3-14 进程推进顺序对死锁的影响

3.5.1 产生死锁的原因

2) 进程推进顺序非法

若并发进程P_1和P_2按曲线④所示的顺序推进，它们将进入不安全区D内。此时P_1保持了资源R_1, P_2保持了资源R_2, 系统处于不安全状态。因为，这时两进程再向前推进，便可能发生死锁。例如，当P_1运行到P_1:Request(R_2)时，将因R_2已被P_2占用而阻塞；当P_2运行到P_2: Request(R_1)时，也将因R_1已被P_1占用而阻塞，于是发生了进程死锁。

3.5.2 产生死锁的必要条件

★ 产生死锁的四个必要条件

1) 互斥条件：在一段时间内，一个资源只能由一个进程独占使用，若别的进程也要求该资源，则须等待直至其占用者释放；

2) 不剥夺条件：进程所获得的资源在未使用完之前，不能被其它进程强行剥夺，而只能由其自身释放；

3) 部分分配：允许进程在不释放其已分得资源的情况下请求并等待分配新的资源；

4) 环路条件：存在一个等待进程集合，P0正在等待一个P1占用的资源，P1正在等待一个P2占用的资源，…，Pn正在等待一个由P0占用的资源。

3.5.3 处理死锁的基本方法

★ 解决死锁问题的基本方法

1)预防：破坏产生死锁的四个必要条件中的一个或多个，使系统绝不会进入死锁状态；

2)避免：产生死锁的四个必要条件有可能成立，但在资源动态分配的过程中使用某种办法防止系统进入不安全和死锁状态；

3)检测与恢复（解除）：允许系统产生死锁，然后使用检测算法及时地发现并解除它.

3.6 预防死锁的方法

★3.6.1死锁预防

1)方法1：破坏“请求和保持条件”

打破资源的部分分配这个死锁产生的必要条件。

即预先分配各并发进程需要的全部资源。

（缺点：不易满足，浪费，降低并发性）

2)方法2：破坏“不剥夺”条件：资源退还

3.6 预防死锁的方法

★3.6.1死锁预防

3)方法3：破坏死锁的环路条件。

即把资源分类按顺序排列，使进程在申请、保持资源时不形成环路。如有M种资源，则列出 1<R2<……<Rm。若进程Pi已经占了资源Ri，则它只能申请比Ri级别更高的资源Rj（Ri<Rj）。释放资源时必须是Rj先于Ri被释放，从而避免环路的产生。

3.6.3利用银行家算法避免死锁

- 数据结构：N为进程的数目，M为资源类型的数目
- 可利用资源向量available[m], available [i]=k,表示系统中现有空闲的Ri类资源有k个。
- 最大需求矩阵Max，这是一个nXm矩阵，它定义了系统中每个进程对m类资源的最大需求数目。Max[i,j]=k,表示进程Pi需要Rj类资源的最大数目为k。
- 分配矩阵Allocation，这是一个nXm矩阵，它定义了系统中每类资源当前已经分配给每一个进程的资源数目。Allocation [i,j]=k,表示进程Pi当前已分到了Rj类资源k个。

银行家算法的数据结构

- 需求矩阵Need，这是一个nXm矩阵， 定义了每个进程号需要的各类资源数目。Need[i,j]=k,表示进程Pi还需要Rj类资源k个。 $Need_i$表示进程Pi的需求向量。
- Need [i,j] = Max[i,j] - Allocation [i,j].
- 各进程当前请求资源Request矩阵，nXm矩阵。Request[i,j]=k表示进程Pi要申请Rj类资源k个。让$Request_i$作为该进程的请求向量表。

银行家算法描述

1. 如果Request[i]<=Need[i]，执行2，否则报错，因为进程当前申请的资源数量超过了它的预先声明。
2. 如果Request[j]<=Available，执行3，否则等待，因为当前系统没有足够的资源。
3. 系统试分资源给Pi，并修改：
   - Available = Available - Requesti
   - $Allocation_i = Allocation_i + Request_i$
   - $Need_i = Need_i - Request_i$
4. 如果分配之后还在安全状态，则实现分配。如果分配之后不再安全状态，则暂停分配，恢复原来状态。

银行家算法的安全检查

1. Work表示当前系统的各类资源的空闲数量。初值Work = Available，
2. Finish表示系统是否有足够的资源分配给进程，使之运行完成，初值Finish [i] = false (i:1,2,3, …, n)
3. 从进程集合中找到一个满足下面条件的进程：
   - Finish [i] = false
   - Need[i] ≤ Work ；如果没有的话，跳到5步。
4. 当进程Pi获得资源后，Work = Work + $Allocation_i$; Finish[i] = true; 转到3。
5. 若所有进程的Finish[i]都等于true，那么系统处于安全状态，否则是不安全的。

银行家算法实例

- 假定系统有四个进程P1,P2,P3,P4,三种类型的资源R1,R2,R3,数量分别为9，3，6，在T0时刻的资源分配情况如下：

Process	Max	allocated	Need	Finish	Available
P1	**3/2/2**	**1/0/0**	**2/2/2**	**false**	**1/1/2**
P2	**6/1/3**	**5/1/1**	**1/0/2**	**false**	
P3	**3/1/4**	**2/1/1**	**1/0/3**	**false**	
P4	**4/2/2**	**0/0/2**	**4/2/0**	**false**	

验证T0时刻的安全性

分析：

1. 四进程执行状态都是未完成，Finish=false
2. 找Pi，其需要的资源need<=有效资源work
3. 当前的work={1/1/2},

   need　　P1　　P2　　P3　　P4

   { (2/2/2), (1/0/2), (1/0/3), (4/2/0) }
4. 找到P2满足条件，如果让P2运行结束，

验证T0时刻的安全性

Process	Work	need	allocation	Work-new	Finish
P2	1/1/2	1/0/2	5/1/1		false
	0/1/0	0/0/0	6/1/3	6/2/3	true
P1	6/2/3	2/2/2	1/0/0		false
	4/0/1	0/0/0	3/2/2	7/2/3	true
P3	7/2/3	1/0/3	2/1/1		false
	6/2/0	0/0/0	3/1/4	9/3/4	true
P4	9/3/4	4/2/0	0/0/2		false
	5/1/4	0/0/0	4/2/2	9/3/6	true

存在运行序列：**P2,P1,P3,P4**

P2请求资源{1，0，1}

- 现在P2请求资源{1/0/1}，按照银行家算法检查：
- $Request_2\{1/0/1\} <= Need_2\{1/0/2\}$
- $Request_2\{1/0/1\} <= Available_2\{1/1/2\}$
- 假定可以分配，修改Available, Allocation, Need

Process	Max	allocated	Need	Finish	Available
P1	3/2/2	1/0/0	2/2/2	false	0/1/1
P2	6/1/3	6/1/2	0/0/1	false	
P3	3/1/4	2/1/1	1/0/3	false	
P4	4/2/2	0/0/2	4/2/0	false	

进行安全性检查

验证P2分配资源后的安全性

Process	Work	need	allocation	Work-new	Finish
P2	0/1/1	0/0/1	6/1/2		false
	0/1/0	0/0/0	6/1/3	6/2/3	true
P1	6/2/3	2/2/2	1/0/0		false
	4/0/1	0/0/0	3/2/2	7/2/3	true
P3	7/2/3	1/0/3	2/1/1		false
	6/2/0	0/0/0	3/1/4	9/3/4	true
P4	9/3/4	4/2/0	0/0/2		false
	5/1/4	0/0/0	4/2/2	9/3/6	true

存在运行序列：**P2,P1,P3,P4**

P1请求资源{1，0，1}

P1请求资源{1/0/1}，按照银行家算法检查：

$Request_1${1/0/1} <= $Need_1${2/2/2}

$Request_1${1/0/1} > $Available_1${0/1/1}

条件不满足，不能分配，让P1等待。

P3请求资源{0，0，1}

- 现在P3请求资源{0/0/1}，按照银行家算法检查：
- $Request_3\{0/0/1\} <= Need_3\{1/0/3\}$
- $Request_3\{0/0/1\} <= Available_3\{0/1/1\}$
- 假定可以分配，修改Available, Allocation, Need

Process	Max	allocated	Need	Finish	Available
P1	3/2/2	1/0/0	2/2/2	false	0/1/0
P2	6/1/3	6/1/2	0/0/1	false	
P3	3/1/4	2/1/2	1/0/2	false	
P4	4/2/2	0/0/2	4/2/0	false	

进行安全性检查

P3分配资源后的安全性

分析：四进程执行状态都是未完成，Finish=false

找Pi，其需要的资源need<=当前的work={0/1/0}

进程的need　P1　P2　P3　P4

{(2/2/2), 0/0/1), (1/0/2), (4/2/0)}

找不到满足条件的Pi,因此资源P3不能分配本次资源，回退。

Process	Max	allocated	Need	Finish	Available
P1	3/2/2	1/0/0	2/2/2	false	0/1/1
P2	6/1/3	6/1/2	0/0/1	false	
P3	3/1/4	2/1/1	1/0/3	false	
P4	4/2/2	0/0/2	4/2/0	false	

3.7 死锁的检测与解除

3.7.1 死锁的检测

1. 资源分配图(Resource Allocation Graph)

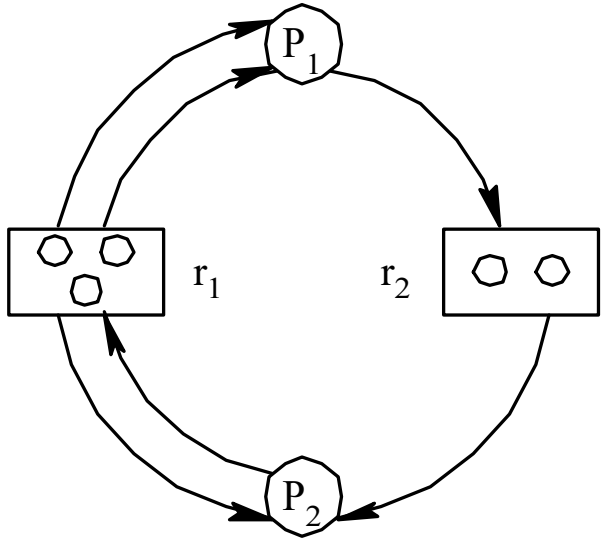


图 3-19 每类资源有多个时的情况

(2) 凡属于E中的一个边e∈E，都连接着P中的一个结点和R中的一个结点，e={p_i，r_j}是资源请求边，由进程p_i指向资源r_j， 它表示进程p_i请求一个单位的r_j资源。e={r_j，p_i}是资源分配边，由资源r_j指向进程p_i，它表示把一个单位的资源r_j分配给进程p_i。

2. 死锁定理：当且仅当**S**状态的资源分配图是不可完全简化的，说明当前状态为死锁状态

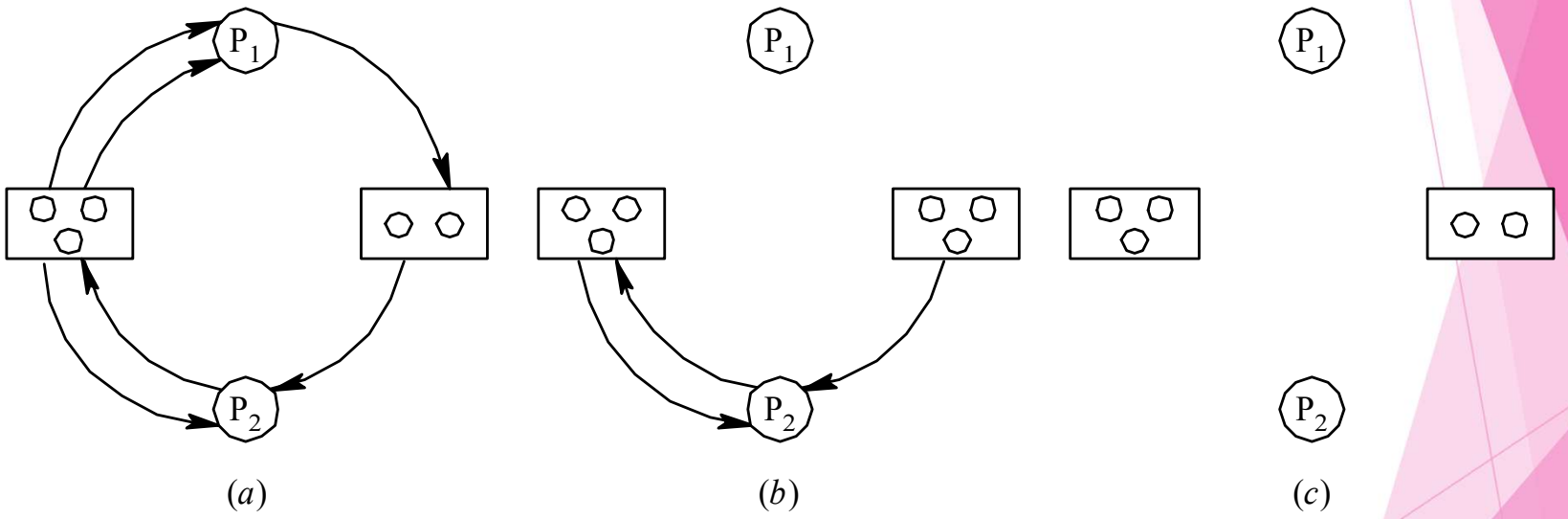
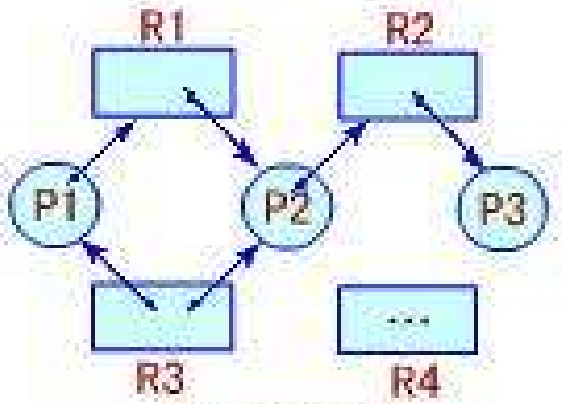
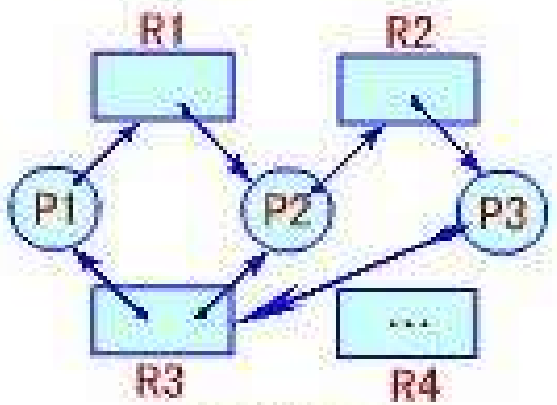


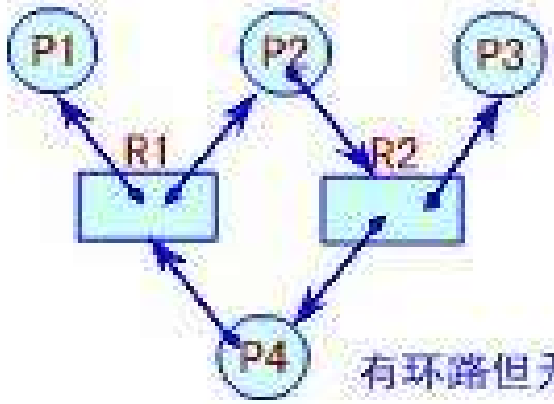
图 3-20 资源分配图的简化



无死锁发生



死锁发生



有环路但无死锁发生

3.7.2 死锁的解除

(1) 剥夺资源。抢别人的资源

(2) 撤消进程。

为把系统从死锁状态中解脱出来，所花费的代价可表示为：

$$R(S)_{min}=min\{C_{ui}\}+min\{C_{uj}\}+min\{C_{uk}\}+\cdots$$

【11年考研27题】某时刻进程的资源使用情况如下表所示。

进程	已分配资源			尚需资源			可用资源		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	2	0	0	0	0	1	0	2	1
P2	1	2	0	1	3	2			
P3	0	1	1	1	3	1			
P4	0	0	1	2	0	0			

此时的安全序列是 (D)

A. P1,P2,P3,P4　　B. P1,P3,P2,P4

C. P1,P4,P3,P2　　D. 不存在

【12年考研27题】

D

假设5个进程P0、P1、P2、P3、P4共享三类资源R1、R2、R3，这些资源总数分别为18、6、22。T0时刻的资源分配情况如下表所示，此时存在的一个安全序列是

进程	已分配资源			资源最大需求		
	R1	R2	R3	R1	R2	R3
P0	3	2	3	5	5	10
P1	4	0	3	5	3	6
P2	4	0	5	4	0	11
P3	2	0	4	4	2	5
P4	3	1	4	4	2	4

A. P0，P2，P4，P1，P3　　B. P1，P0，P3，P4，P2

C. P2，P1，P0，P3，P4　　D. P3，P4，P2，P1，P0

【13年考研32题】

32. 下列关于银行家算法的叙述中，正确的是（ B ）

A. 银行家算法可以预防死锁

B. 当系统处于安全状态时，系统中一定无死锁进程

C. 当系统处于不安全状态时，系统中一定会出现死锁进程

D. 银行家算法破坏了死锁必要条件中的“请求和保持”条件

【14年考研24题】

24某系统有n台互斥使用的同类设备，三个并发进程分别需要3、4、5台设备，可确保系统不发生死锁的设备数n最小为（B）

A 9

B 10

C 11

D 12

【15年考研26题】

26．若系统S1 采用死锁避免方法，S2 采用死锁检测方法，下列叙述中正确的是（C）

Ⅰ．S1 会限制用户申请资源的顺序

Ⅱ．S1 需要进行所需资源总量信息，而S2 不需要

Ⅲ．S1 不会给可能导致死锁的进程分配资源，S2 会

A．仅Ⅰ Ⅱ　　B．仅Ⅱ Ⅲ

C．仅Ⅰ Ⅲ　　D．Ⅰ Ⅱ Ⅲ

【17年考研23题】假设4个作业到达系统的时刻和运行时间如下表所示

作业	到达时刻t	运行时间
J1	0	3
J2	1	3
J3	1	2
J4	3	1

系统在t=2时开始作业调度。若分别采用先来先服务和短作业优先调度算法，则选中的作业分别是（D）

A.J2、J3　　B.J1、J4

C.J2、J4　　D.J1、J3

【16年考研46题】

46. 某进程调度程序采用基于优先数(priority)的调度策略，即选择优先数最小的进程运行，进程创建时由进程指定一个nice作为静态优先数。为了动态调整优先数，引入运行时间cpuTime和等待时间waitTime，初值均为零。进程处于执行态时，cpuTime定时为1，且waitTime置0；进程处于就绪态时，cpuTime置0，waitTime定时为1。请回答下列问题。

（1） 若调度程序只将nice的值作为进程优先数，即priority=nice，则可能出现解饿现象，为什么？

（2） 使用nice、cupTime和waitTime设计一种动态优先数算法，以避免产生饥饿现象，并说明waitTime的作用。

【16年考研46题】

优先数**priority**的计算公式为：

priority=nice+K1*cpuTime-K2*waitTime

其中**K1>0**，**K2>0**，用来分别调整**cpuTime**和**waitTime**在**priority**中所占的比例。

waitTime可使长时间等待的进程数减小，从而避免出现饥饿现象。

作业

1.系统中有四个资源A, B, C, D，有4个进程p1, p2, p3, p4，在当前时刻资源占用情况和进程等待情况是：

资源占用情况		进程等待情况	
资源号	进程	进程号	等待资源号
A	p1	p1	B
B	p2	p2	D
C	p4	p3	A
D	p3	p4	B

请画出资源分配图，并分析是否为死锁状态。

2. 在银行家算法中，若出现下面的资源分配情况：

Process	Allocation	Need	Available
P0	0032	0012	1622
P1	1000	1650	
P2	1354	2356	
P3	0032	0652	
P4	0014	0656	

试问：

（1）该

（2）当 给它？

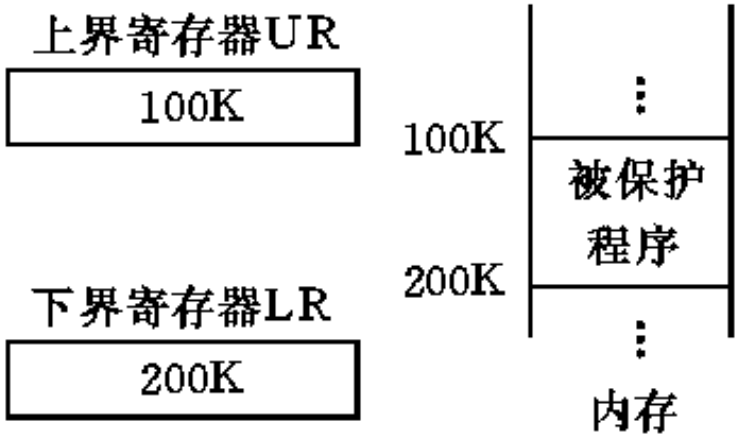
（3）如果系统立即满足P2的上述请求，则系统是否立即进入死锁状态？

Unit4

- 内存保护的方法
- 分配方式
- 动态分区的分配与回收（填空，选择，大题（灵活））
- 16进制转成二进制算，不要转成十进制算

内存共享和保护

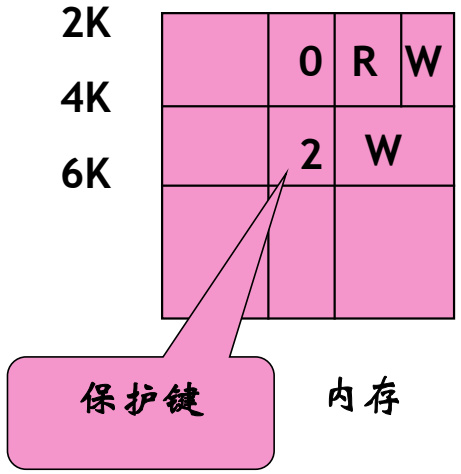
★**上下界保护法**：为每个进程设置上下界寄存器，装有被保护进程的起始地址和终止地址，在运行时先检查地址是否越界。

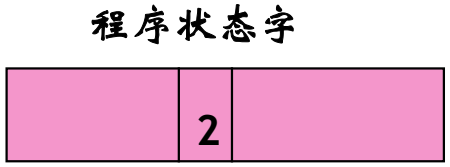


100K≤被访问地址≤200K

内存共享和保护

★保护键法：为每一个被保护存储块分配一个保护键。在程序状态字中则设置相应的保护键开关字段，对不同的进程赋予不同的开关代码和与被保护的存储块中的保护键匹配。





正确的访问：

LOAD AH 5000

STORE BL 5200

错误的访问：

LOAD BL 2500

开关字-键不匹配

4.3.3 动态分区分配

★动态分区的分配与回收

动态分区时的分配与回收主要解决三个问题：

- 对于请求表中的要求内存长度，从可用表或自由链中寻找出合适的空闲区分配程序。
- 分配空闲区之后，更新可用表或自由链。
- 进程或作业释放内存资源时，和相邻的空闲区进行链接合并，更新可用表或自由链。

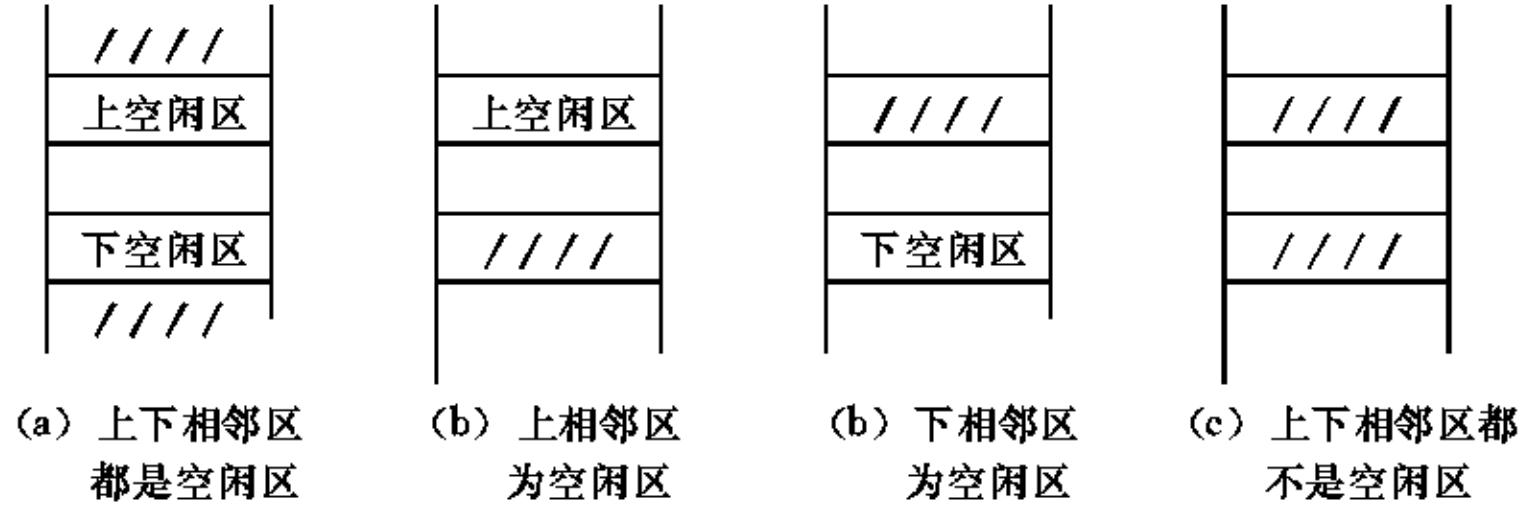
4.3.3 分区的分配与回收

★动态分区的分配与回收---常用方法

- **最先适应法**：将可用分区按起始**地址递增排列**。每次从低位开始向后找。这样经常利用的是低地址空间。后面经常是较大的空白区。
- **循环首次适应法**：每次从上次的位置开始查找。
- **最佳适应法**：按空白区大小，**从小到大**次序组成空白区可用表或自由链。
- **最坏适应法**：按空白区大小，**从大到小**次序组成空白区可用表或自由链。

4.3.3 分区的分配与回收

★动态分区时的回收与拼接



(a) 上下相邻区都是空闲区　(b) 上相邻区为空闲区　(b) 下相邻区为空闲区　(c) 上下相邻区都不是空闲区

空闲区的合并

思 考

例：用可变分区方式管理主存时,假定主存中按地址顺序依次有5个空闲区,空闲区的大小依次为32K,10K,5K,228K和100K,现有5个作业A,B,C,D,E.它们各需主存1K,10K,108K,28K和115K.若采用最先适应算法能把这5个作业按顺序全部装入主存吗?你认为怎样的次序这5个作业可全部装入?

【17考研25题】某计算机按字节编址，其动态分区内存管理采用最佳适应算法，每次分配和回收内存后都对空闲分区链重新排序。当前空闲分区信息如下表所示

分区起始地址	20K	500K	1000K	200K
分区大小	40KB	80KB	100KB	200KB

回收起始地址为60K、大小为140KB的分区后，系统中空闲分区的数量、空闲分区链第一个分区的起始地址和大小分别是（ ）B

A.3、20K、380KB　　B.3、500K、80KB

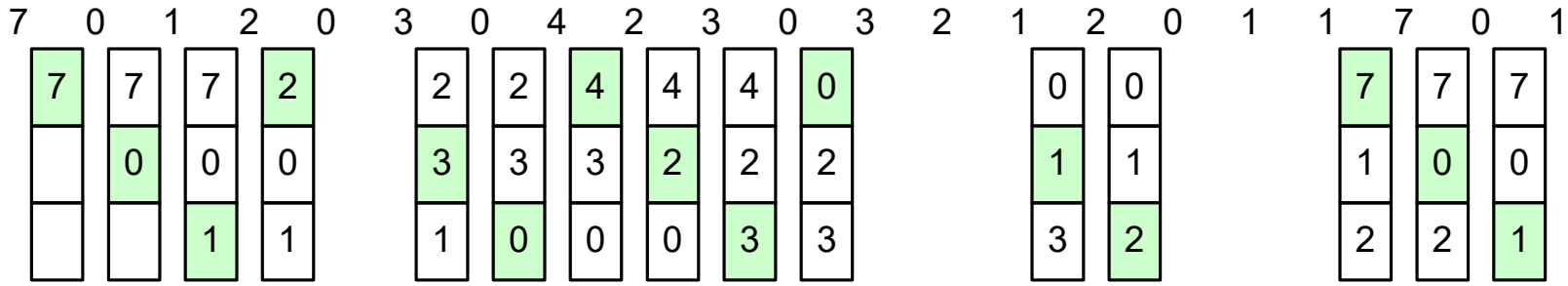
C.4、20K、180KB　　D.4、500K、80KB

4.6 请求页式管理的置换算法

★先进先出算法(FIFO- First Input First Output），

先进入内存的页面先淘汰。

- 优点：实现简单。
- 缺点：常用的页会被淘汰。



缺页率15/21=71.4%

4.6 请求页式管理的置换算法

Belady现象：分配给一个进程的页面增加，反而出现缺页增加的现象

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4

缺页率为9/12=75%

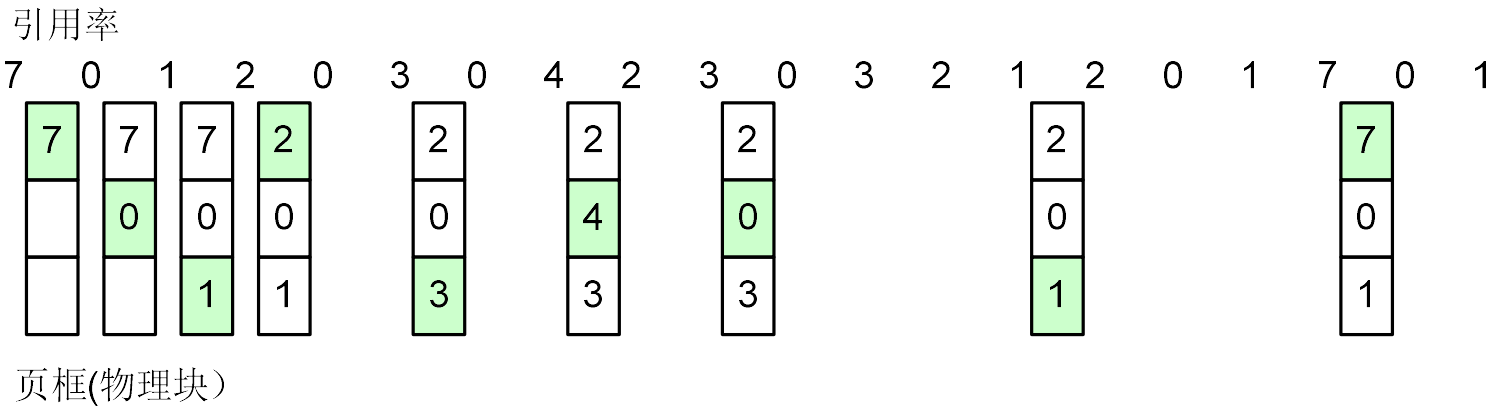
1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3

缺页率=10/12=83.3%

原因是没有考虑页的执行顺序

4.6 请求页式管理的置换算法

★最优淘汰算法（OPT-Optimal replacement algorithm）：是理想算法。系统预测作业将要访问的页面。淘汰预测不被访问或长时间后才被访问中的页面。



缺页率9/20=45%

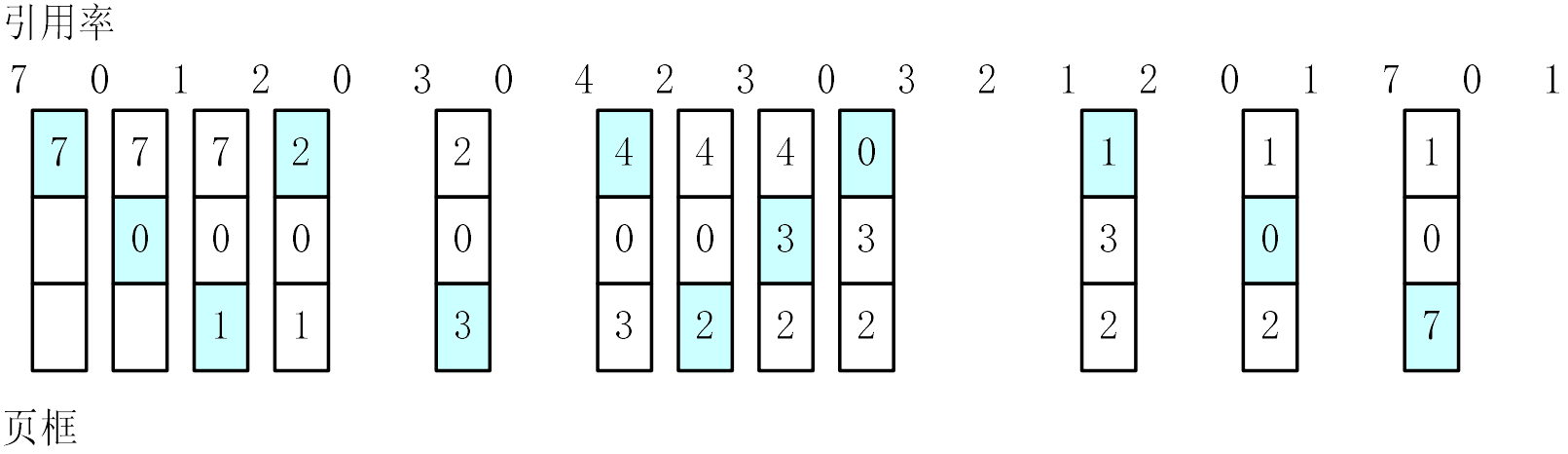
几乎无法实现!

4.6 请求页式管理的置换算法

★最近最久未使用页面淘汰法（LRU - Least Recently Used）：

淘汰最近一段时间最久没访问的页面。

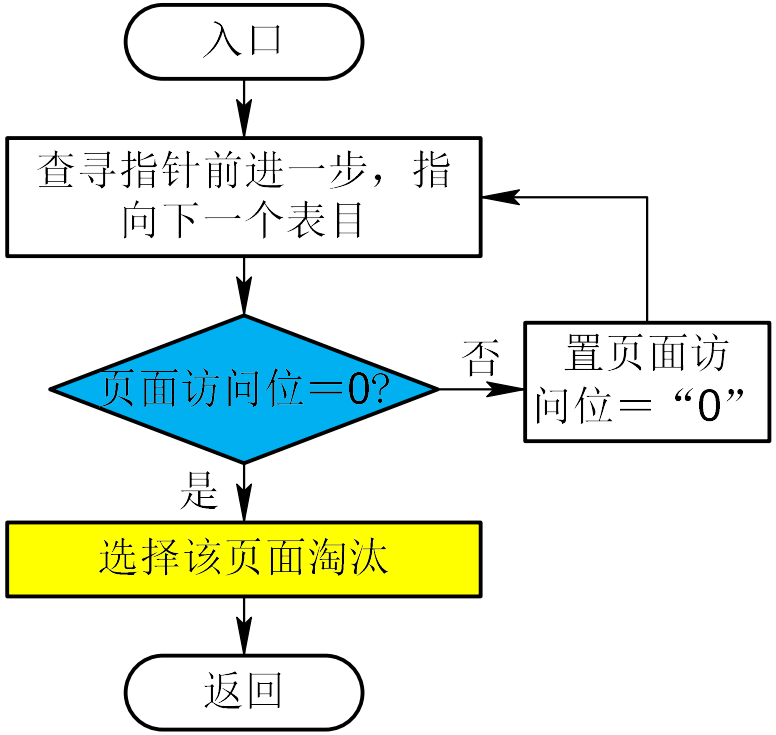
缺点：每页设访问记录，每次更新，系统开销大。



缺页率12/20=60%

★ Clock置换算法

1. 简单的Clock置换算法：　被访问页面置1，置换时：



块号	页号	访问位	指针
0			
1			
2	4	0	
3			
4	2	1	
5			
6	5	0	
7	1	1	

替换指针

图 4-30 简单Clock置换算法的流程和示例

2. 改进型Clock置换算法

由访问位A和修改位M可以组合成下面四种类型的页面：

1类(A=0, M=0)：表示该页最近既未被访问， 又未被修改，是最佳淘汰页。

2类(A=0, M=1)： 表示该页最近未被访问， 但已被修改，并不是很好的淘汰页。

3类(A=1, M=0)： 最近已被访问， 但未被修改， 该页有可能再被访问。

4类(A=1, M=1)：最近已被访问且被修改， 该页可能再被访问。

其执行过程可分成以下三步：

(1) 从指针所指示的当前位置开始， 扫描循环队列， 寻找A=0且M=0的第一类页面， 将所遇到的第一个页面作为所选中的淘汰页。 在第一次扫描期间不改变访问位A。

(2) 如果第一步失败，则开始第二轮扫描，寻找A=0且M=1的第二类页面，将所遇到的第一个这类页面作为淘汰页。在第二轮扫描期间，将所有扫描过的页面的访问位都置0。

(3) 如果第二步也失败，亦即未找到第二类页面，则将指针返回到开始的位置，并将所有的访问位复0。 然后重复第一步，如果仍失败，必要时再重复第二步，此时就一定能找到被淘汰的页。

考研真题

【16年考研26题】某系统采用改进型CLOCK置换算法，页表项中字段A为访问位，M为修改位，A=0表示页最近没有被访问过，A=1表示页最近被访问过。M=0表示页没有被修改过，M=1表示页被修改过。按（A,M）所有可能的取值，将页分为四类：（0,0）、（1,0）、（0,1）和（1,1），则该算法淘汰页的次序为（A　）

A. (0,0),(0,1),(1,0),(1,1)　　B. (0,0),(1,0),(0,1),(1,1)

C. (0,0),(0,1),(1,1),(1,0)　　D. (0,0),(1,1),(0,1),(1,0)

4.6 请求页式管理的置换算法

★最不经常使用的页面先淘汰（LFU-Least Frequent Used）：淘汰到目前为止访问次数最少的页面。对每一页设访问计数器，缺页中断时清零。

★随机数淘汰页面算法：无法确定哪些页不使用，随机淘汰一页。

思 考

例如：一个32位的系统支持的逻辑空间最大为2^{32} B，页的大小为4KB,即2^{12}，一个页表项4个字节

问：一个进程页表最多有多少个表项？

$$\frac{2^{32}}{2^{12}} = 2^{20} = 1\text{M个}$$

问：一个进程页表需要占多大的内存？

答：1M × 4B = 4MB

进程页表是要连续存放的，则4MB空间需要连续的4MB/4KB=1024页。

【16年考研28题】

28、某进程的段表内容如下所示（D ）

段号	段长	内存起始地址	权限	状态
0	100	6000	只读	在内存
1	200	—	读写	不在内存
2	300	4000	读写	在内存

当访问段号为2、段内地址为400的逻辑地址时，进行地址转换的结果是

A. 段号异常　　B. 得到内存地址4400

C. 越权异常　　D. 越界异常

考研真题

【09年考研26题】分区分配内存管理方式的主要保护措施是：A

A、界地址保护　B.程序代码保护　C.数据保护　D.栈保护

【09年考研27题】一个分段存储管理系统中，地址长度为32位，其中段号占8位，则段长最大（C ）

A、2^8字节　B.2^{16}字节　C.2^{24}字节　D.2^{32}字节

【10年考研28题】某基于动态分区存储管理的计算机，其主存容量为55Mb（初始空间），采用最佳适配（Best fit）算法，分配和释放的顺序为：分配15Mb，分配30Mb，释放15Mb，分配8Mb，分配6Mb，此时主存中最大空闲分区的大小是（B ）

A.7MB　B.9MB　C.10MB　D.15MB

考研真题

【11年考研28题】在缺页处理过程中，操作系统执行的操作可能是(D)

Ⅰ.修改页表 Ⅱ、磁盘I/O Ⅲ、分配页框

A、仅Ⅰ、Ⅱ B、仅Ⅱ C、仅Ⅲ D、Ⅰ、Ⅱ和Ⅲ

【11年考研29题】当系统发生抖动（thrashing）时，可用采取的有效措施是(A)

Ⅰ.撤销部分进程 Ⅱ、增加磁盘交换区的容量 Ⅲ、提高用户进程的优先级

A、仅Ⅰ B、仅Ⅱ C、仅Ⅲ D、仅Ⅰ、Ⅱ

【12年考研25题】下列关于虚拟存储的叙述中，正确的是(B)

A.虚拟存储只能基于连续分配技术 B.虚拟存储只能基于非连续分配技术

C.虚拟存储容量只受外存容量的限制 D.虚拟存储容量只受内存容量的限制

考研真题

【13年考研30题】若用户进程访问内存时产生缺页，则下列选项中，操作系统可能执行的操作是(B)

Ⅰ. 处理越界错　Ⅱ. 置换页　Ⅲ. 分配内存

A. 仅Ⅰ、Ⅱ　B. 仅Ⅱ、Ⅲ　C. 仅Ⅰ、Ⅲ　D. Ⅰ、Ⅱ和Ⅲ

【15年考研27题】系统为某进程分配了4个页框，该进程已访问的页号序列为2,0,2,9,3,4,2,8,2,3,8,4,5，若进程要访问的下一页的页号为7，依据LRU算法，应淘汰页的页号是（ B ）

A. 2　B. 3　C. 4　D. 8

考研真题

【14年考研28题】下列措施中，能加快虚实地址转换的是(C)

Ⅰ增大快表(TLB)容量

Ⅱ让页表常驻内存

Ⅲ增大交换区(swap)

A 仅Ⅰ　　B 仅Ⅱ　　C 仅Ⅰ、Ⅱ　　D 仅Ⅱ、Ⅲ

【14年考研32题】下列选项中，属于多级页表优点的是(D)

A.加快地址变换速度

B.减少缺页中断次数

C.减少页表项所占字节数

D.减少页表所占的连续内存空间

考研真题

【15年考研30题】在请求分页系统中，页面分配策略与页面置换策略不能组合使用的是（C）

A、可变分配，全局置换

B、可变分配，局部置换

C、固定分配，全局置换

D、固定分配，局部置换

1) 固定分配局部置换(Fixed Allocation, Local Replacement)（固定分配，置换自身进程页）

2) 可变分配全局置换(Variable Allocation, Global Replacement)（动态分配，物理块用完时可置换其他进程页）

3) 可变分配局部置换(Variable Allocation, Local Replacemen（置换自身进程页，频

【10年考研46题】设某计算机的逻辑地址空间和物理地址空间均为64KB按字节编址。若某进程最多需要6页(Page)数据存储空间，页的大小为1KB，操作系统采用固定分配局部置换策略为此进程分配4个页框(Page Fame).

页号	页框号	装入时刻	访问位
0	7	130	1
1	4	230	1
2	2	200	1
3	9	160	1

当该进程执行到时刻260时，要访问逻辑地址为17CAH的数据，请问答下列问题：

(1)、该逻辑地址对应的页号是多少?

(2)、若采用先进先出(FIFO)置换算法，该逻辑地址对应的物理地址是多少?要求给出计算过程。

考研真题

解答：17CAH=(0001 0111 1100 1010)2

（1）页大小为1K，所以页内偏移地址为10位，于是前6位是页号，所以第一问的解为：5

（2）FIFO，则被置换的页面所在页框为7，所以对应的物理地址为（0001 1111 1100 1010）2即1FCAH

【12年考研45题】某请求分页系统的局部页面置换策略如下：系统从0时刻开始扫描，每隔5个时间单位扫描一轮驻留集（扫描时间忽略不计），本轮未被访问过的页框将被系统收回，并放入到空闲页框链尾，其中内容在下一次被分配之前不被清空。当发生缺页时，如果该页曾被使用过且还在空闲页框链表中，则重新放回进程的驻留集中；否则，从空闲页框链表头部取出一个页框。

假设不考虑其他进程的影响和系统开销。初始时进程驻留集为空。目前系统空闲页框链表中页框号依次为32、15、21、41。进程P依次访问的<虚拟页号，访问时刻>为<1,1>、<3,2>、<0,4>、<0 ,6>、<1,11>、<0,13>、<2,14>。请回答以下问题：

（1）访问<0,4>时，对应的页框号是什么？说明理由。

（2）访问<1,11>时，对应的页框号是什么？说明理由。

（3）访问<2,14>时，对应的页框号是什么？说明理由。

（4）该策略是否适合于时间局部性好的程序。说明理由

【答案要点】

（1）页框号为21。因为起始驻留集为空，而0页对应的页框为空闲链表中的第三个空闲页框，其对应的页框号为21。

（2）页框号为32。因为11>10故发生第三轮扫描，页号为1的页框在第二轮已经处于空闲页框链表中，此刻该页又被重新访问，因此应被重新放回到驻留集中，其页框号为32。

（3）页框号为41。因为第2页从来没有被访问过，不在驻留集中，因此从空闲链表中取出链表头的页框，页框号为41。

（4）适合。程序的时间局部性越好，则从空闲页框链表中被重新取回的机会就越大，该策略的优势越明显。

【13年考研46题】某计算机主存按字节编址，逻辑地址和物理地址都是32位，页表项大小为4字节。请回答下列问题。

（1）若使用一级页表的分页存储管理方式，逻辑地址结构为：

页号（20位）	页内偏移量（12位）

则页的大小是多少字节？页表最大占用多少字节？

（2）若使

页目录号（10位）	页表索引（10位）	页内偏移量（12位）

设逻辑地址为LA，请分别给出其对应的页目录号和页表索引的表达式。

（3）采用（1）中的分页存储管理方式，一个代码段起始逻辑地址为0000 8000H，其长度为8KB，被装载到从物理地址0090 0000H开始的连续主存空间中。页表从主存0020 0000H开始的物理地址处连续存放，如下图所示（地址大小自下向上递增）。请计出该代码段对应的两个页表项的物理地址、这两个页表项中的页框号以及代码页面2的起始物理地址。

【13年考研46题】某计算机主存按字节编址，逻辑地址和物理地址都是32位，页表项大小为4字节。请回答下列问题。

（1）若使用一级页表的分页存储管理方式，逻辑地址结构为：

页号（20位）	页内偏移量（12位）

则页的大小是多少字节？页表最大占用多少字节？

（1）因为页内偏移量是12位，所以页大小为4 KB

页表项数为$2^{32}/4K=2^{20}$，该一级页表最大为$2^{20}\times 4\ B=4\ MB$。

（2）若

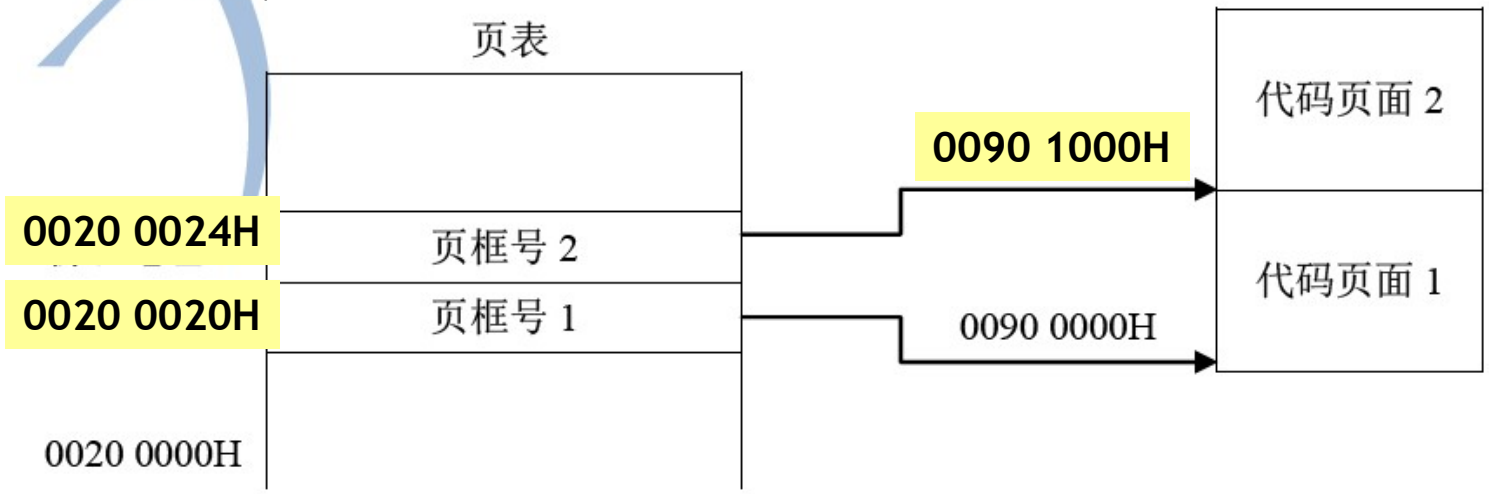
页目录号（10位）	页表索引（10位）	页内偏移量（12位）

（2）页目录号可表示为：(((unsigned int)(LA))>>22)&0x3FF。

页表索引可表示为：(((unsigned int)(LA))>>12)&0x3FF

（3）采用（1）中的分页存储管理方式，一个代码段起始逻辑地址为0000 8000H，其长度为8KB，被装载到从物理地址0090 0000H开始的连续主存空间中。页表从主存0020 0000H开始的物理地址处连续存放，如下图所示（地址大小自下向上递增）。请计出该代码段对应的两个页表项的物理地址、这两个页表项中的页框号以及代码页面2的起始物理地址。

（3）代码页面1的逻辑地址为0000 8000H，表明其位于第8个页处，对应页表中的第8个页表项，所以第8 个页表项的物理地址 = 页表起始地址+8×页表项的字节数 = 0020 0000H+8×4 = 0020 0020H。



例 题

- 例如，如果跟踪一个特定进程，那么可记录如下地址顺序：

  0722, 0050, 0121, 0212, 0012, 0310, 0019, 0421, 0289, 0302, 0076, 0365, 0196　CPU 从物理地址 0722 取数据，然后从 0050，然后从 0121，.....

- 按每页100 个字节（即页面大小为100 B），其页面引用序列结果是怎样？　7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1

- 如果分配给该进程3个页帧，按照先进先出页面置换算法，缺页率为多少？

作 业

- 1.在一个请求分页系统中，加入一个作业的页面走向为4,3,2,1,4,3,5,4,3,2,1,5，目前它还没有任何页装入内存，当分配给该作业的物理块数目M分别为3和4时，请分别计算采用OPT,LRU和FIFO页面淘汰算法时，访问过程中所发生的的缺页次数和缺页率，并比较所得的结果。

作 业

- 2.某虚拟存储器的用户空间共有32个页面，每页1KB，主存16KB。假定某时刻系统为用户的第0,1,2,3页分配的物理块号为5,10,4,7，而该用户作业的长度为6页，试将16进制的逻辑地址0A5CH,0F3CH转换成物理地址。

Unit5（只考小题）

- DMO通道
- 程序IO等，中断IO的优点
- 四种缓冲只考小题
- 磁盘访问时间
- 磁盘调度算法（大题）
- PPT选择题必考
- 作业必考
- SPOOLING不考
- 廉价磁盘冗余阵列不考

5.2.4 I/O通道控制方式

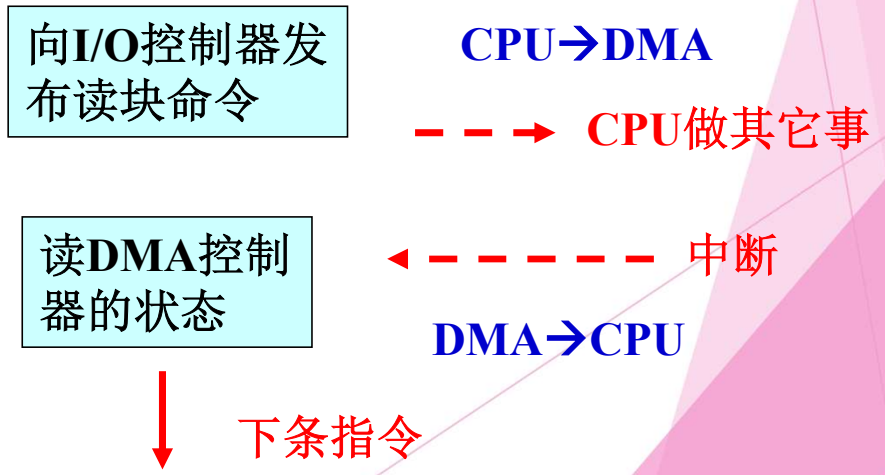
★三种通道的比较

通道类型	字节多路	数组多路	选择通道
数据宽度	单字节	定长块（字）	不定长块（字）
适用范围	大量低速设备	大量高速设备	优先级高的高速设备
工作方式	字节交叉	成组交叉	独占通道
共享性	分时共享	分时共享	独占
选择设备次数	多次	多次	一次

5.2 I/O控制方式

★ 5.2.1 直接存储器访问（DMA）I/O方式

- 数据传输基本单位是**数据块**
- 从设备**直接送入内存**或相反。
- 整块数据的传输是在**控制器的控制**下完成的。仅在开始和结束时才需CPU干预。



5.2.3 DMA I/O控制方式

★DMA控制器的组成

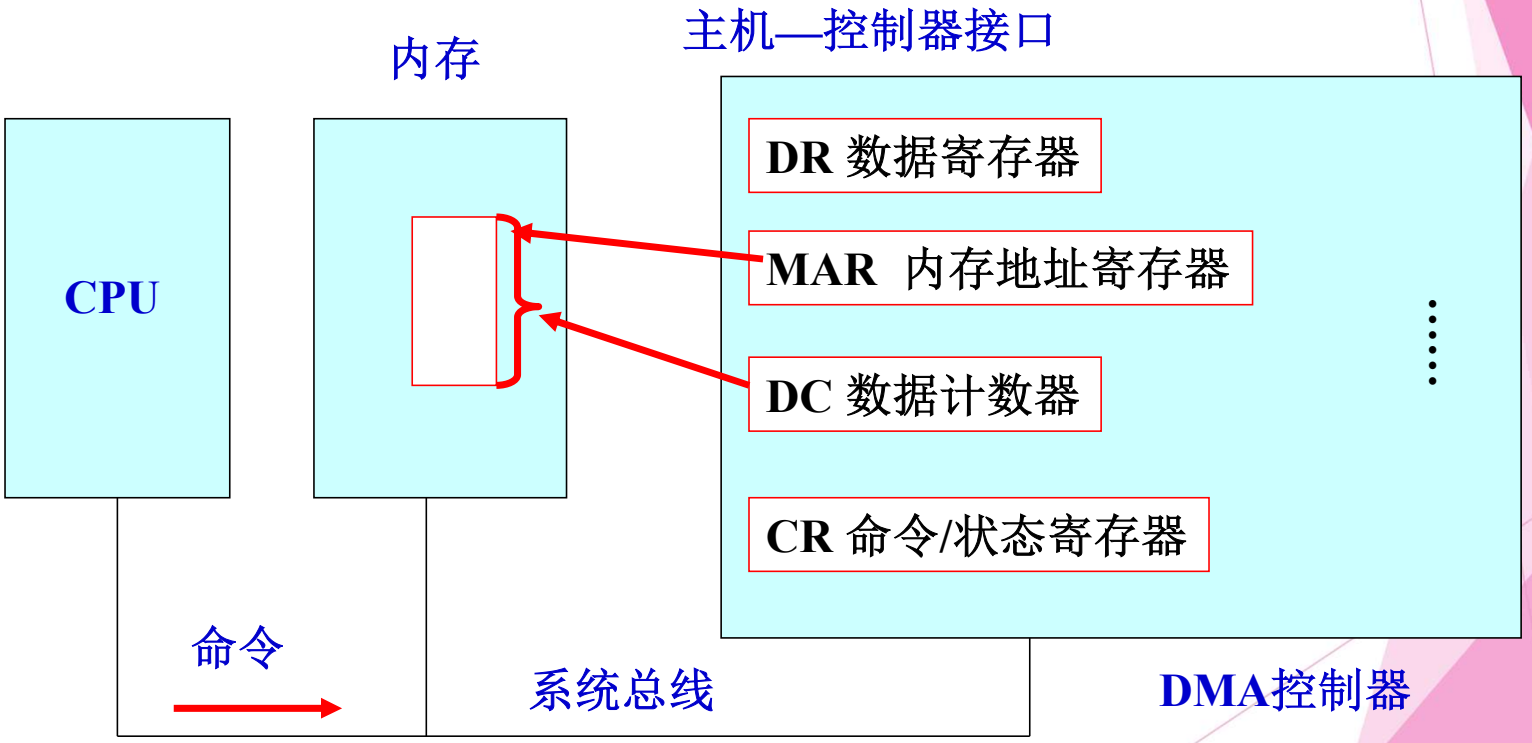
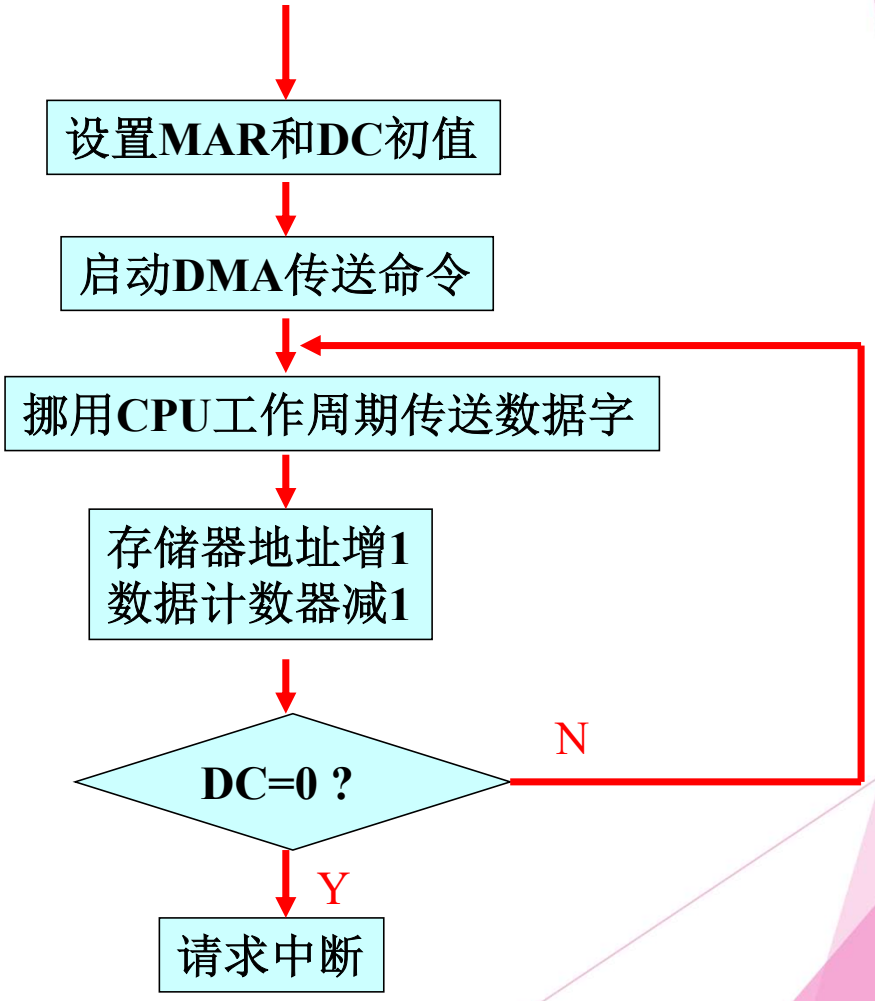


图5-4 DMA控制器的组成

5.2.3 DMA I/O控制方式

★DMA工作过程



考研真题

【17考研32题】系统将数据从磁盘读到内存的过程包括以下操作：

①DMA控制器发出中断请求

②初始化DMA控制器并启动磁盘

③从磁盘传输一块数据到内存缓冲区

④执行“DMA结束”中断服务程序

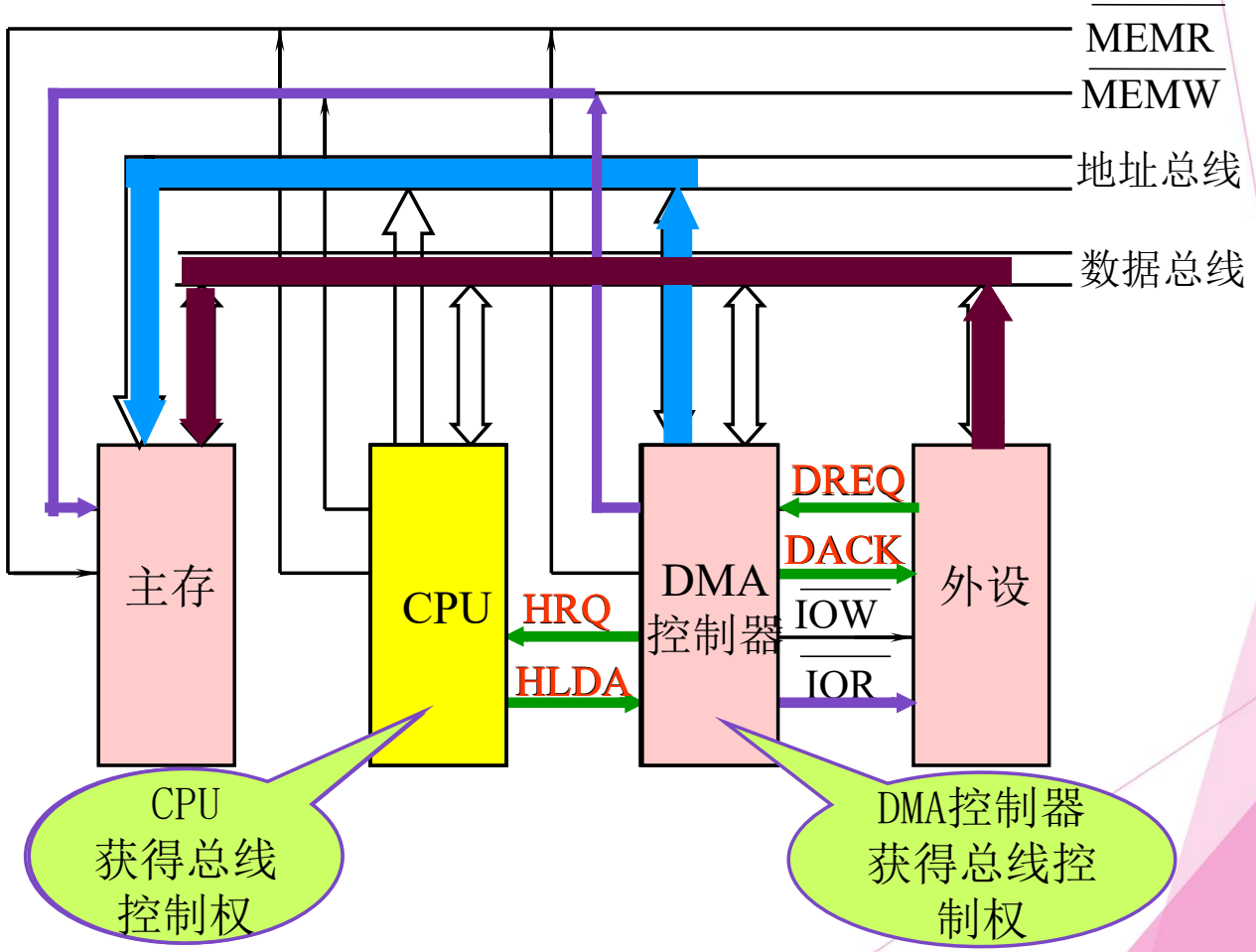
正确的执行顺序是（　　）

B

A.③→①→②→④　B.②→③→①→④

C.②→①→③→④　D.①→②→④→③

★ DMA控制器的连接和传送



5.2.3 DMA I/O控制方式

★DMA I/O方式的特点

- CPU与I/O设备在更大的程度上并行工作，效率更高。
- DMA方式适合高速批量的数据传输，如视频显示刷新、磁盘存储系统的读写，存储器到存储器的传输等。
- DMA控制器取代CPU接管地址总线的控制权。使CPU访问总线时速度会变慢。

5.2 I/O控制方式

- 对**大型系统**，设备多，数据传输频繁，DMA造成的总线冲突仍然影响CPU的效率。
- **解决办法**：采用I/O通道方式进行数据交换。
  - 执行专用指令，完成数据交换
  - **专用处理器**，受主CPU控制（启停等）

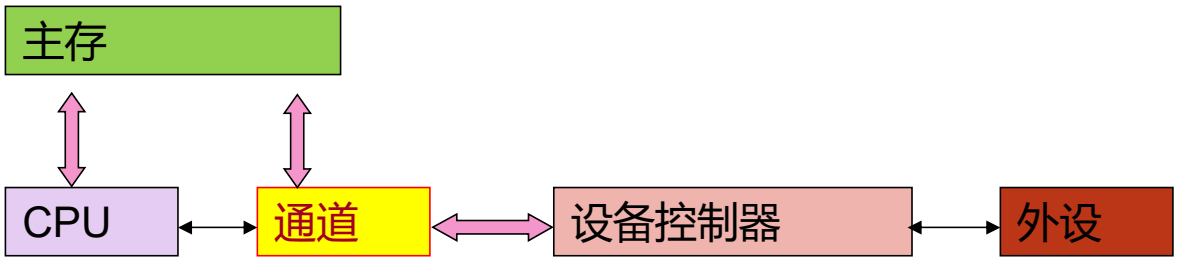


图5-5 通道的位置

考研真题

【11年考研31题】某文件占 10 个磁盘块，现要把该文件磁盘块逐个读入主存缓冲区，并送用户区进行分析，假设一个缓冲区与一个磁盘块大小相同，把一个磁盘块读入缓冲区的时间为100us，将缓冲区的数据传送到用户区的时间是50us，CPU对一块数据进行分析的时间为50us。在单缓冲区和双缓冲区结构下，读入并分析完该文件的时间分别是(　　) B

A、1500us、1000us　　B、1550us、1100us

C、1550us、1550us　　D、2000us、2000us

考研真题

【09年考研32题】程序员利用系统调用打开I/O设备时，通常使用的设备标识是（ A ）

A、逻辑设备名 B.物理设备名 C.主设备号 D.从设备号

【10年考研32题】本地用户通过键盘登录系统时，首先获得键盘输入信息的程序是（ B ）

A.命令解释程序 B.中断处理程序

C.系统调用程序 D.用户登录程序

考研真题

【11年考研26题】用户程序发出磁盘I/O请求后，系统的正确处理流程是（　）　B

A、用户程序→系统调用处理程序→中断处理程序→设备驱动程序

B、用户程序→系统调用处理程序→设备驱动程序→中断处理程序

C、用户程序→设备驱动程序→系统调用处理程序→中断处理程序

D、用户程序→设备驱动程序→中断处理程序→系统调用处理程序

考研真题

【12年考研24题】中断处理和子程序调用都需要压栈以保护现场，中断处理一定会保存而子程序调用不需要保存其内容的是(B)。

A.程序计数器　　B.程序状态字寄存器

C.通用数据寄存器　　D.通用地址寄存器

【12年考研26题】操作系统的I/O子系统通常由四个层次组成，每一层明确定义了与邻近层次的接口，其合理的层次组织排列顺序是(A)。

A.用户级I/O软件、设备无关软件、设备驱动程序、中断处理程序

B.用户级I/O软件、设备无关软件、中断处理程序、设备驱动程序

C.用户级I/O软件、设备驱动程序、设备无关软件、中断处理程序

D.用户级I/O软件、中断处理程序、设备无关软件、设备驱动程序。

5.2.1 程序I/O方式

★程序I/O方式的特点

- I/O操作由CPU直接完成。
- 外设与CPU完全串行工作。
- 外设速度慢，CPU速度快，在外设准备过程中，CPU处在不断的查询之中，CPU的效率得到了极大的浪费。

5.2.2 中断驱动I/O控制方式

★中断I/O方式的特点

- I/O操作仍然由CPU完成
- 在外设准备阶段，CPU与外设是并行工作的
- 与程序I/O相比大大提高了CPU的利用率，但是从外部设备读取一块数据到存储器时，每读一个字的数据产生一次中断。

5.2.3 DMA I/O控制方式

★DMA I/O方式的特点

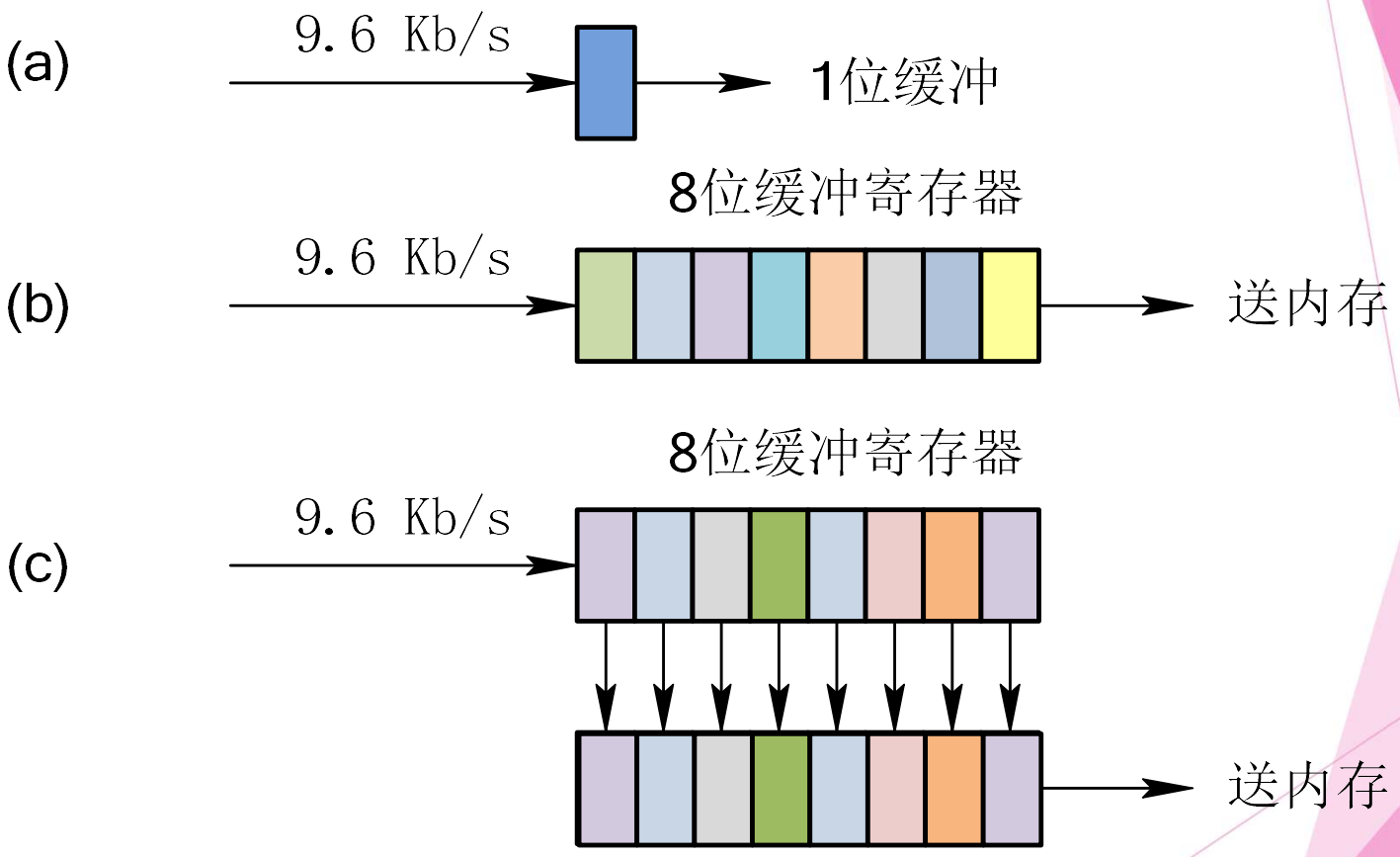
- CPU与I/O设备在更大的程度上并行工作，效率更高。
- DMA方式适合高速批量的数据传输，如视频显示刷新、磁盘存储系统的读写，存储器到存储器的传输等。
- DMA控制器取代CPU接管地址总线的控制权。使CPU访问总线时速度会变慢。

5.3缓冲管理

★缓冲区的引入

- 缓和CPU与I/O设备间速度不匹配的矛盾。
- 减少对CPU的中断频率， 放宽对CPU中断响应时间的限制。
- 提高CPU和I/O设备之间的并行性。

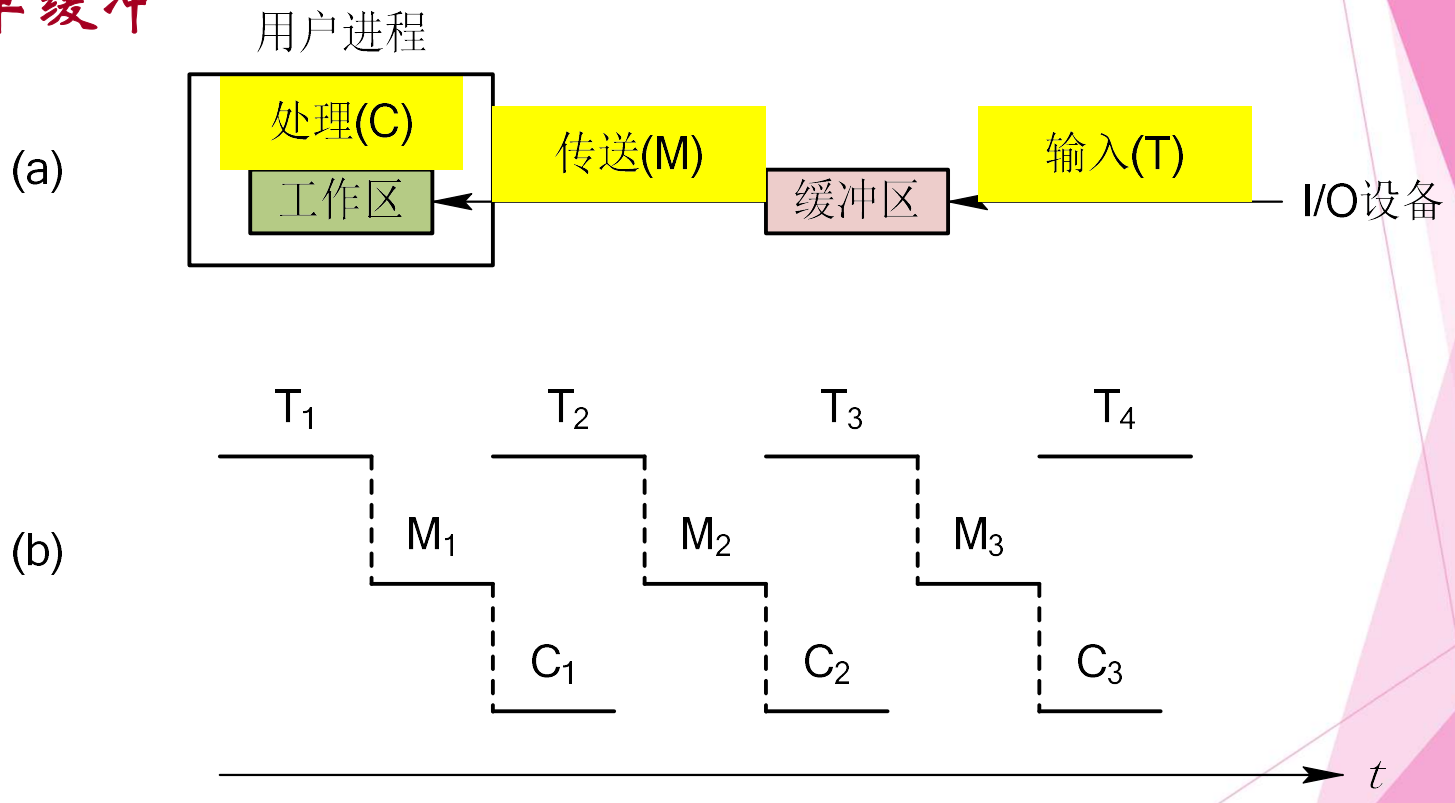
5.3.1 缓冲的引入



利用缓冲寄存器实现缓冲

5.3.2 单缓冲和双缓冲

★单缓冲



单缓冲工作示意图

系统对每一块数据的处理时间可表示为： ***MAX(C,T)+M***

5.3.2 单缓冲和双缓冲

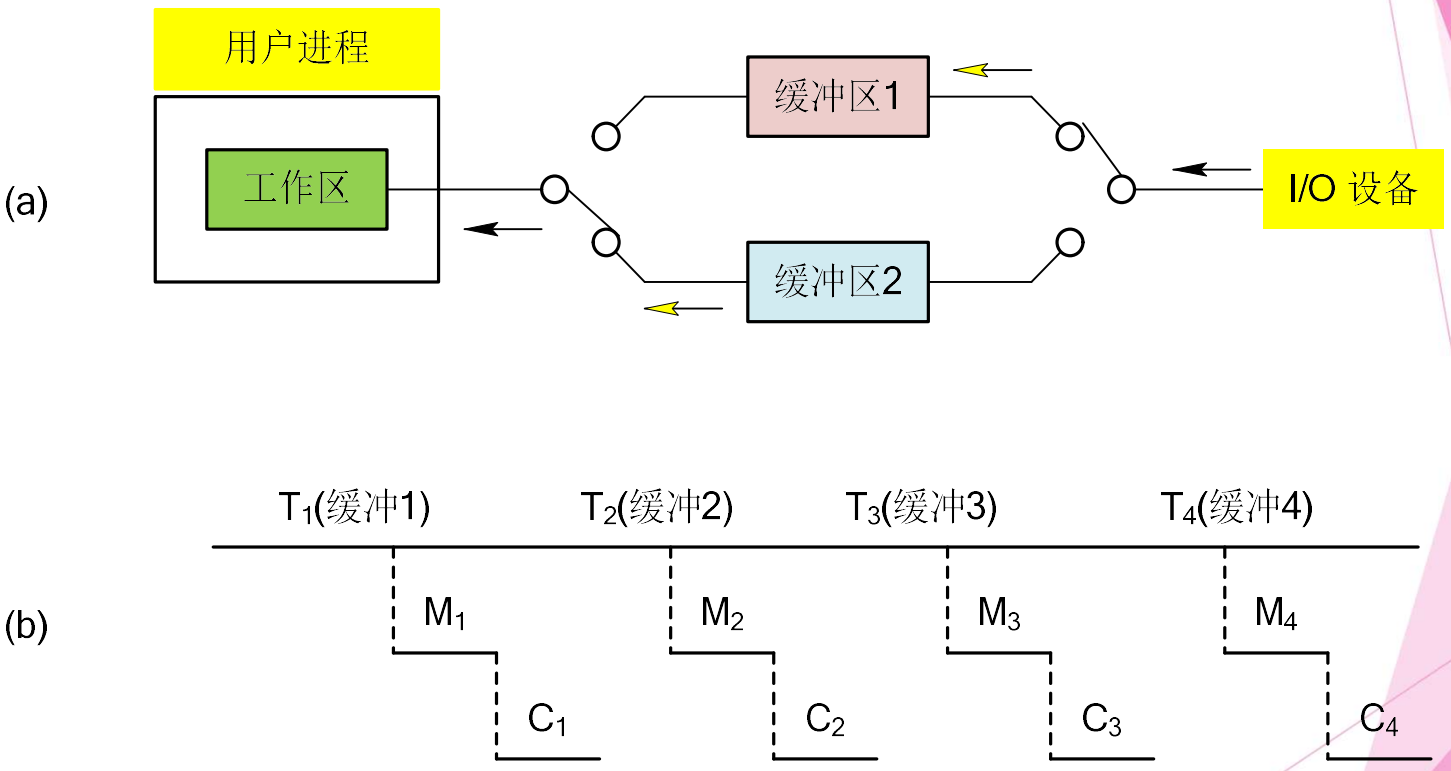
★单缓冲

字符设备输入时，缓冲区用于暂存用户输入的一行数据，**输入期间用户进程被挂起**。

当用户进程有第二行数据输出时，若第一行**数据尚未提取完毕**，则此时**用户进程应阻塞**。

5.3.2 单缓冲和双缓冲

★双缓冲



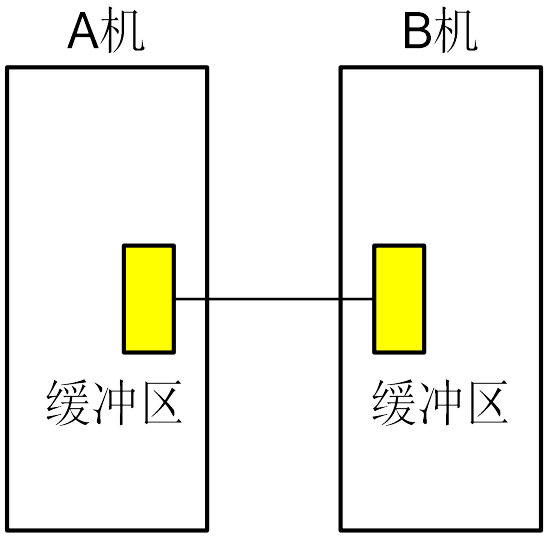
双缓冲工作示意图

5.3.2 单缓冲和双缓冲

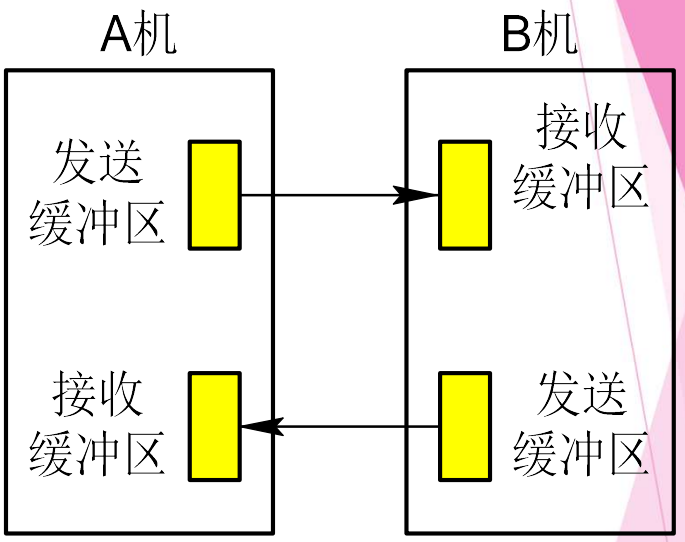
★双缓冲

- 系统处理一块数据时间可表示为：*MAX(C,T)*
- 如果C<T,则可使设备连续输入。
- 若C>T,则可使CPU不必等待设备输入。
- 字符设备输入时，当用户进程有第二行数据输出时，若第一行数据尚未提取完毕，则此时用户进程也不必阻塞。

5.3.2 单缓冲和双缓冲



(a) 单缓冲



(b) 双缓冲

双机通信时缓冲区的设置

考研真题

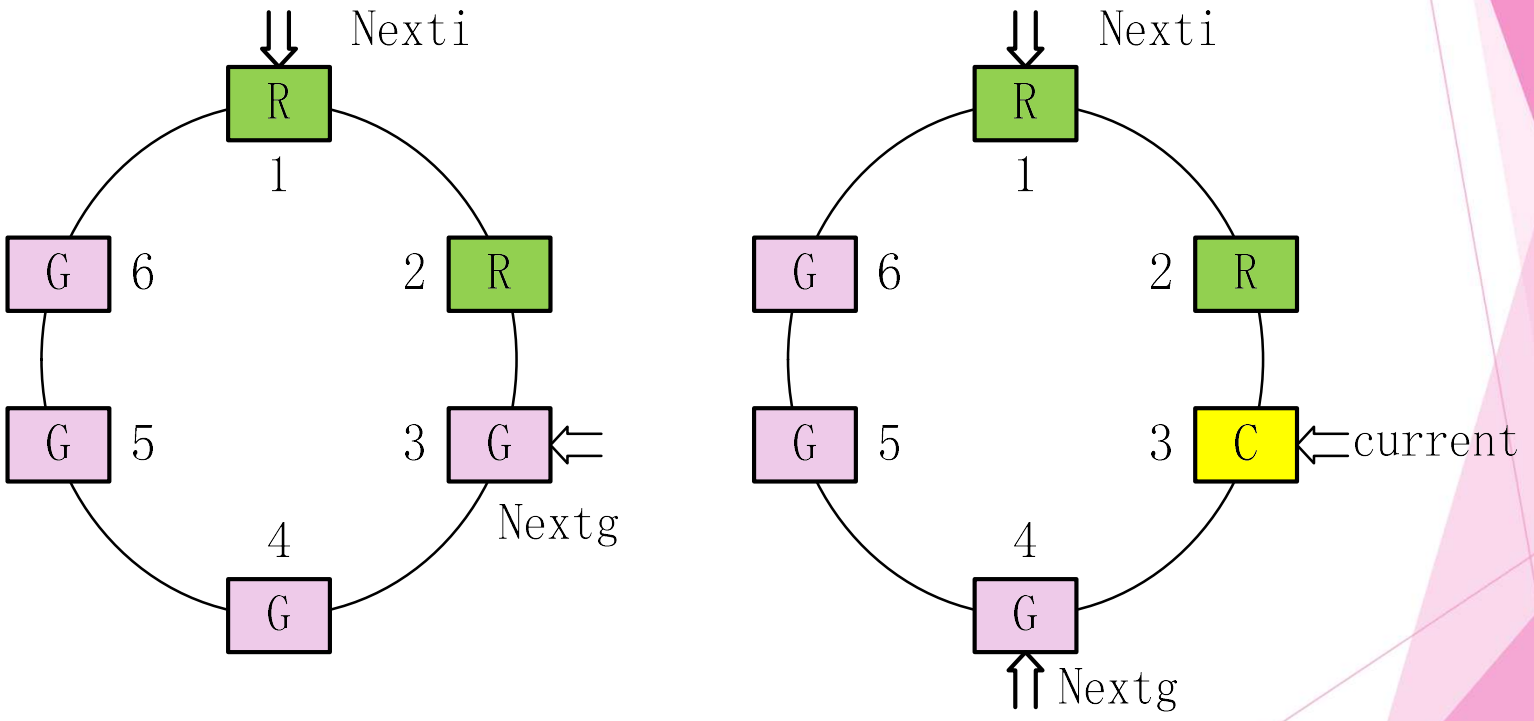
【11年考研31题】某文件占 10 个磁盘块，现要把该文件磁盘块逐个读入主存缓冲区，并送用户区进行分析，假设一个缓冲区与一个磁盘块大小相同，把一个磁盘块读入缓冲区的时间为100us，将缓冲区的数据传送到用户区的时间是50us，CPU对一块数据进行分析的时间为50us。在单缓冲区和双缓冲区结构下，读入并分析完该文件的时间分别是(　　) B

A、1500us、1000us　　B、1550us、1100us

C、1550us、1550us　　D、2000us、2000us

5.3.3 循环缓冲

★循环缓冲的组成　空缓冲区　满缓冲区

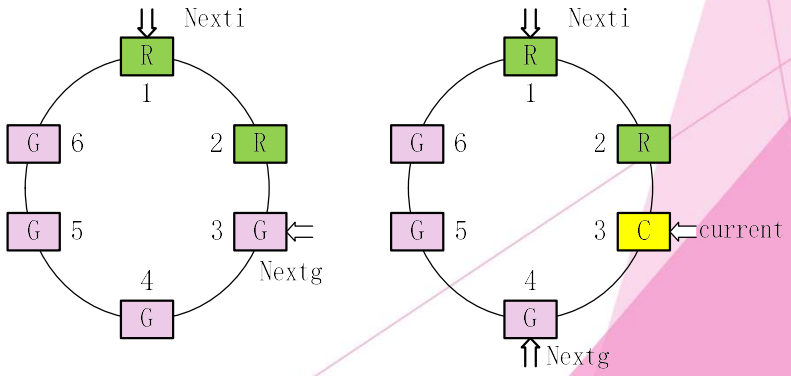


循环缓冲

5.3.3 循环缓冲

★ 循环缓冲区的使用 （以计算进程为例）

- Getbuf过程
  - 1）将Nextg所指示的工作区修改为现行工作区。
  - 2）Nextg移向下一个G缓冲区。
- Releasebuf过程
  - 1）计算进程取数完毕后， 将工作缓冲区由C改为空缓冲区R

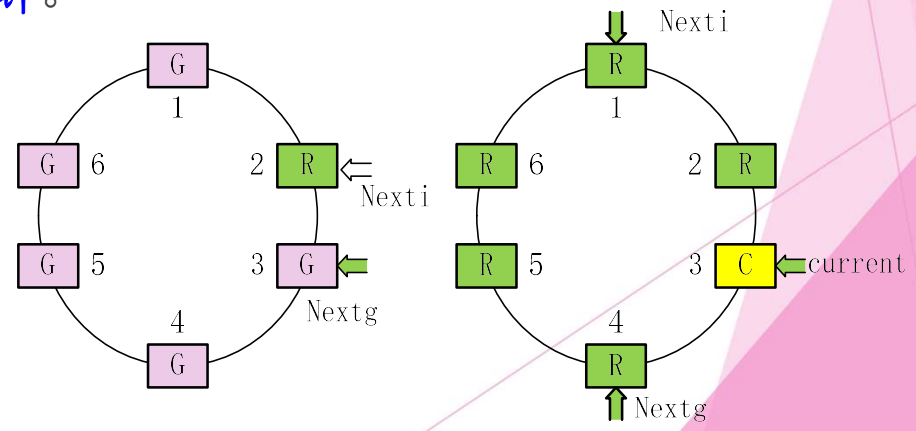


5.3.3 循环缓冲

★进程同步

使用循环缓冲可以使输入进程和计算进程并行执行，但是可能出现如下情况：

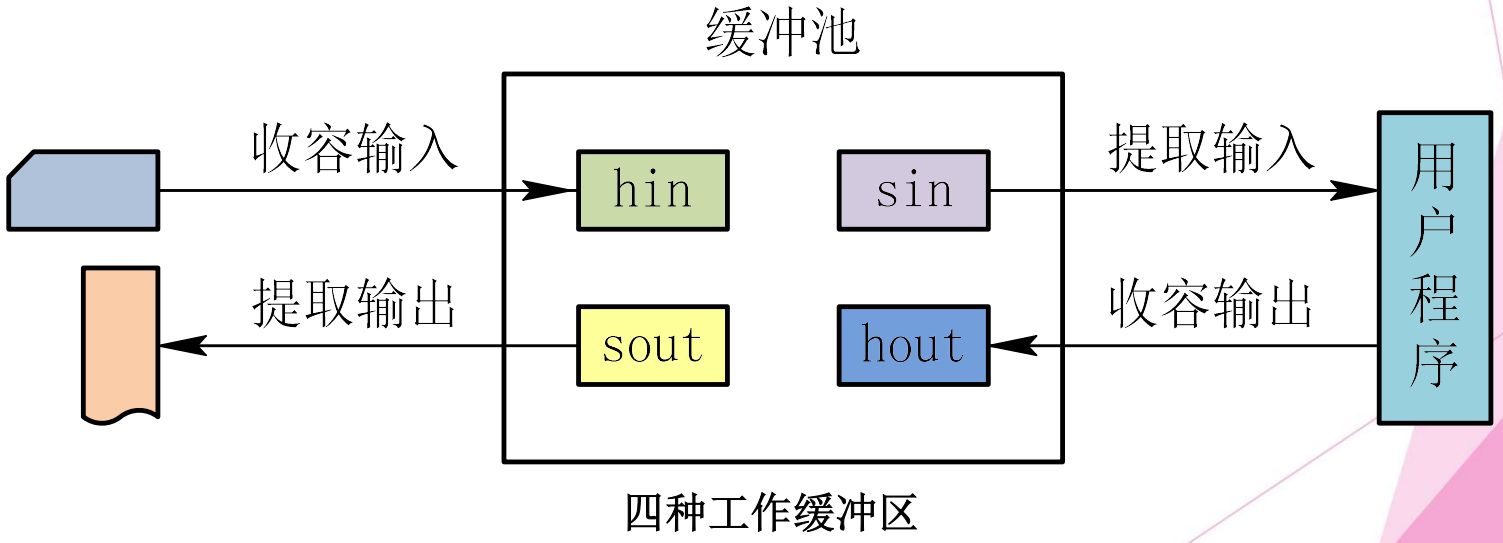
- Nexti指针追赶上Nextg指针。 全满，阻塞输入进程
- Nextg指针追赶上Nexti指针。 全空，阻塞计算进程



5.3.4 缓冲池

★缓冲池的组成

- 空缓冲队列emq
- 输入队列inq：装满输入数据的队列
- 输出队列outq：装满输出数据的队列



四种工作缓冲区

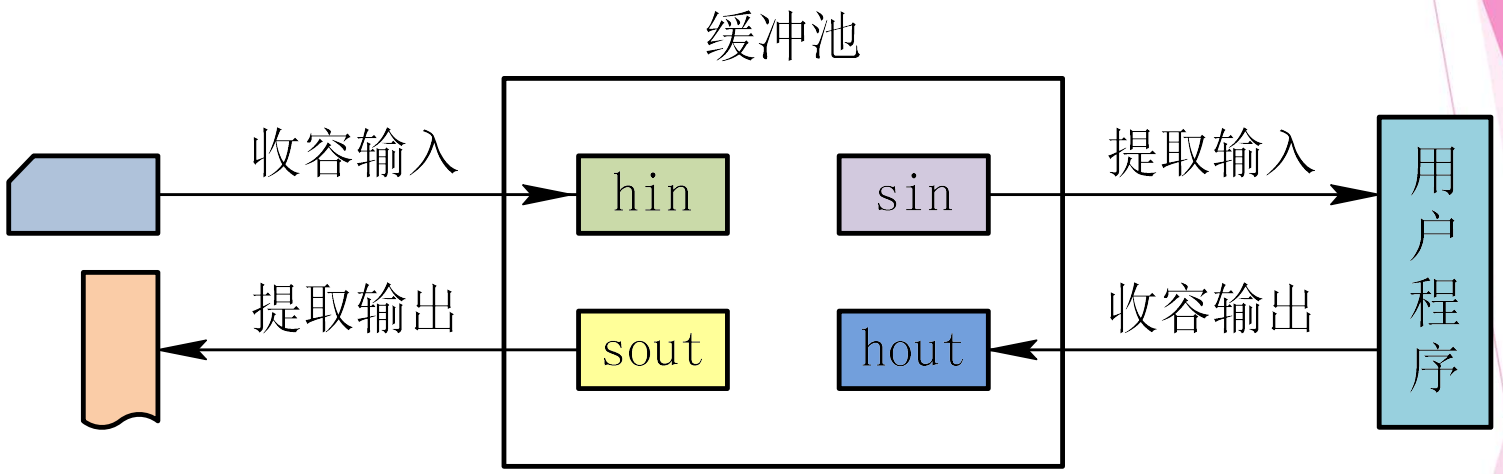
5.3.4 缓冲池

★ Getbuf过程和Putbuf过程

```
Procedure Getbuf(type)  //从队列取走缓冲区
    begin
        Wait(RS(type)); //RS(type)为资源信号量
        Wait(MS(type)); //MS(type)为互斥信号量
        B(number)：=Takebuf(type);//输出
        Signal(MS(type));
    end
Procedure Putbuf(type, number)//把缓冲区填入队列
    begin
        Wait(MS(type));
        Addbuf(type, number);//把number缓冲区输入type队列
        Signal(MS(type));
        Signal(RS(type));
    end
```

5.3.4 缓冲池

★缓冲区的工作方式



1）收容输入 Getbuff(emq), Putbuff(inq, hin)

2）提取输入 Getbuff(inq), Putbuff(emq, sin)

3）收容输出 Getbuff(emq), Putbuff(outq, hout)

4）提取输出 Getbuff(out), Putbuff(emq, sout)

5.6.1 磁盘性能简述

★磁盘访问时间

1) 寻道时间T_s

这是指把磁臂(磁头)移动到指定磁道上所经历的时间。该时间是启动磁臂的时间s与磁头移动n条磁道所花费的时间之和， 即

$$T_s=m\times n+s$$

其中，m是一常数，与磁盘驱动器的速度有关，对一般磁盘， m=0.2；对高速磁盘，m≤0.1，磁臂的启动时间约为2 ms。

5.6.1 磁盘性能简述

★磁盘访问时间

2) 旋转延迟时间T_τ

这是**指定扇区移动到磁头下面**所经历的时间。对于硬盘，典型的旋转速度大多为**5400 r/min**，每转需时**11.1 ms**，平均旋转延迟时间T_τ为**5.55 ms**；

$$T_r = \frac{1}{2r}$$

5.6.1 磁盘性能简述

★磁盘访问时间

3) 传输时间T_t

把数据从磁盘读出或向磁盘写入数据所经历的时间。T_t的大小与每次所读/写的字节数b和旋转速度有关：

$$T_t = \frac{b}{rN}$$

其中，r为磁盘每秒钟的转数；N为一条磁道上的字节数，当一次读/写的字节数相当于半条磁道上的字节数时，T_t与T_T相同。

5.6.1 磁盘性能简述

★磁盘访问时间

磁盘访问时间=寻道时间+旋转延迟时间+传输时间T_t

可将访问时间T_a表示为：（N为一条磁道上的字节数，b为读写字节数）

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

$$T_a = m \times n + s + \frac{1}{2r} + \frac{b}{rN}$$

5.6.2 磁盘调度

- 磁盘是可被多个进程共享的设备。当有多个进程都请求访问磁盘时，应采用一种适当的调度算法，以使各进程对磁盘的平均访问时间最小。

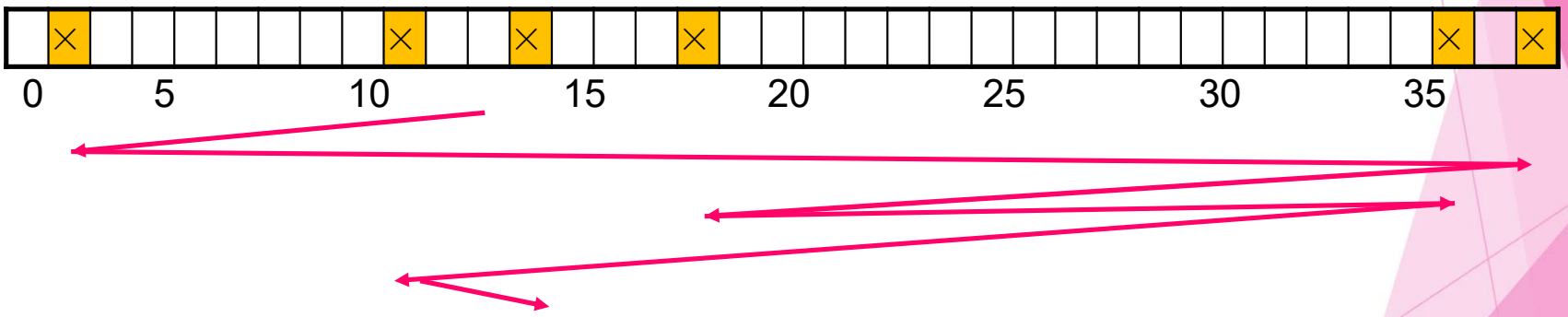
$$T_a = m \times n + s + \frac{1}{2r} + \frac{b}{rN}$$

由于在访问磁盘的时间中主要是寻道时间，因此，磁盘调度的目标应是使磁盘的平均寻道时间最少。

5.6.2 磁盘调度

★先来先服务FCFS(First Come First Served)

- 假定磁盘共有40个柱面，当前磁头正在第11道服务，等待服务的进程有6个，它们请求的磁道号分别是：1，36，16，34，9 和 12 (以请求时间先后为序)。



移动为： 11 → 1 → 36 → 16 → 34 → 9 → 12

总移动磁道数: 10+35+20+18+25+3 = 111

5.6.2 磁盘调度

★先来先服务FCFS(First Come First Served)

- 按访问请求到达的先后次序服务
- 优点：简单，公平；
- 缺点：效率不高，相邻两次请求可能会造成最内到最外的柱面寻道，使磁头反复移动，增加了服务时间，对机械也不利

5.6.2 磁盘调度

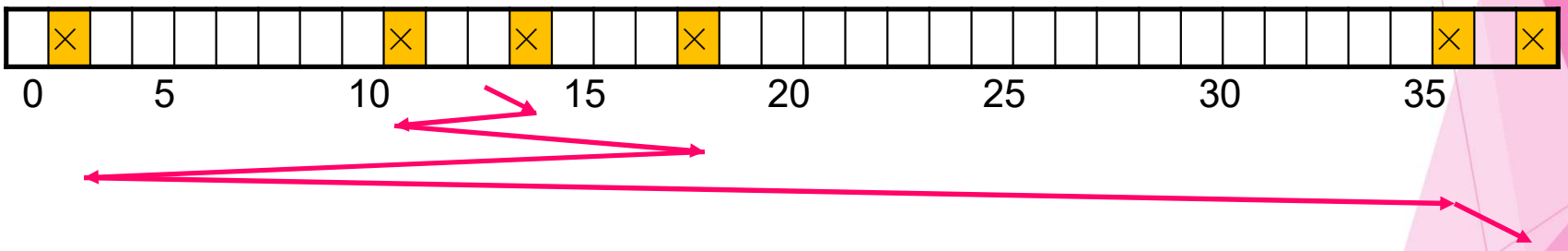
★最短寻道时间优先SSTF(Shortest Seek Time First)

- 优先选择距当前**磁头最近**的访问请求进行服务，主要考虑寻道优先。

5.6.2 磁盘调度

★最短寻道时间优先SSTF(Shortest Seek Time First)

- 假定磁盘共有40个柱面，当前磁头正在第11道服务，等待服务的进程有6个，它们请求的柱面分别是：1，36，16，34，9 和 12 (以请求时间先后为序)。



移动为：11 → 12 → 9 → 16 → 1 → 34 → 36

总移动磁道数：1+3+7+15+33+2 = 61

由此可知总的磁道移动数为61，而FCFS为111

5.6.2 磁盘调度

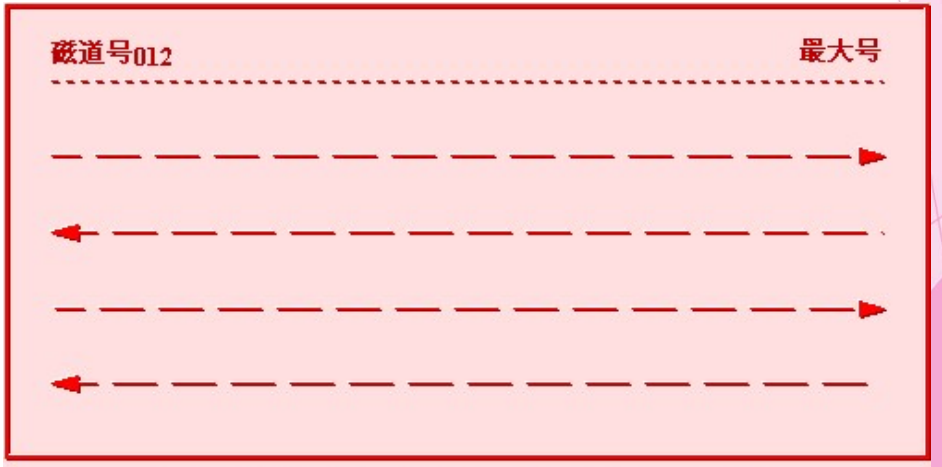
★最短寻道时间优先SSTF(Shortest Seek Time First)

- 优先选择距当前**磁头最近**的访问请求进行服务，主要考虑寻道优先。
- **优点**：改善了磁盘平均服务时间；
- **缺点**：造成某些访问请求长期等待得不到服务

5.6.2 磁盘调度

★扫描(SCAN)算法

- 又称电梯算法。
- **具体做法**：当有访问请求时，磁头**按一个方向移动**，在移动过程中对遇到的访问请求进行服务，然后判断该方向上是否还有访问请求，如果有则继续扫描；否则改变移动方向，如此反复

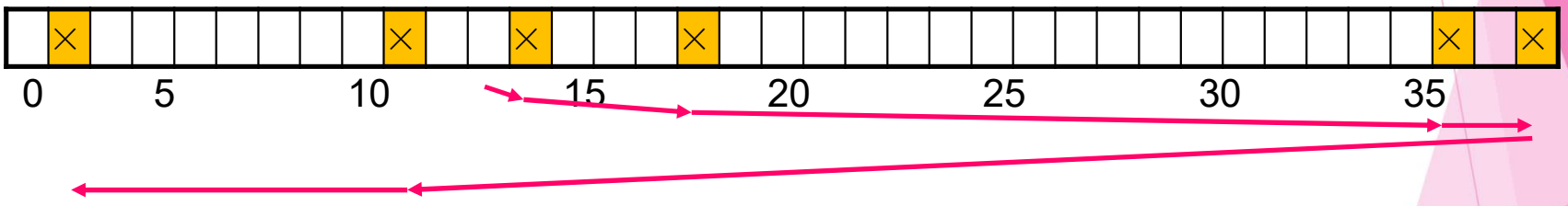


扫描算法（电梯算法）的磁头移动轨迹

5.6.2 磁盘调度

★扫描(SCAN)算法

- 假定磁盘共有40个柱面，当前磁头正在第11道自里向外服务，等待服务的进程有6个，它们请求的柱面分别是：1，36，16，34，9 和 12 (以请求时间先后为序)。



移动为：11 → 12 → 16 → 34 → 36 → 9 → 1

总移动磁道数：1+4+18+2+27+8 = 60

5.6.2 磁盘调度

★扫描(SCAN)算法

- 优点：既考虑了距离，同时又考虑了方向
- 缺点：当请求对磁道的分布是均匀时，磁头回头，近磁头端的请求很少（因为磁头刚经过），而远端请求较多，这些请求等待时间要长一些。

5.6.2 磁盘调度

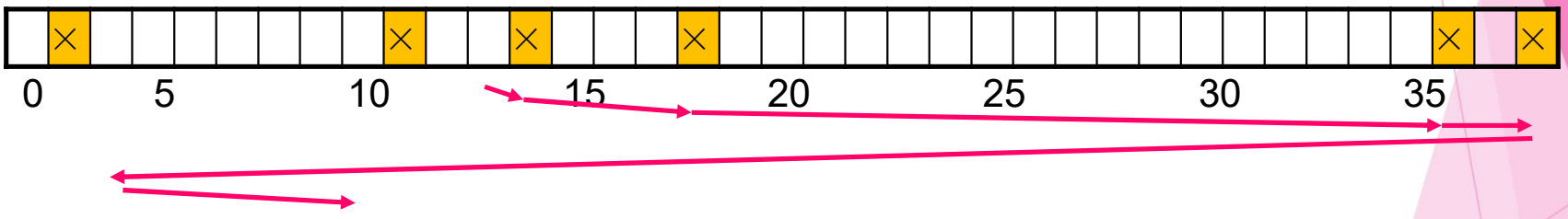
★循环扫描(CSCAN)算法

- 也称**单向扫描算法**。
- 总是从外面柱面开始向里扫描。移动臂到达最后个一个请求磁道柱面后，立即带动读写磁头快速返回。返回时不为任何的等待访问者服务。返回后可再次进行扫描。

5.6.2 磁盘调度

★循环扫描(CSCAN)算法

- 假定磁盘共有40个柱面，当前磁头正在第11道自里向外服务，等待服务的进程有6个，它们请求的柱面分别是：1，36，16，34，9 和 12 (以请求时间先后为序)。



移动为：11 → 12 → 16 → 34 → 36 → 1 → 9

总移动磁道数：1+4+18+2+35+8 = 68

调度算法的选择

- 实际系统相当普遍采用**最短寻道时间优先算法**，因为它简单有效，性价比好。
- **扫描算法**更适于磁盘负担重的系统。
- 磁盘负担很轻的系统也可以采用**先来先服务算法**
- 一般要将磁盘调度算法作为操作系统的单独模块编写，利于修改和更换。

5.6.2 磁盘调度

★ N-Step-SCAN和FSCAN调度算法

1) N-Step-SCAN算法

在SSTF、 SCAN及CSCAN几种调度算法中， 都可能出现磁臂停留在某处不动的情况， 例如，有一个或几个进程对某一磁道有较高的访问频率， 即这个(些)进程反复请求对某一磁道的I/O操作，从而垄断了整个磁盘设备。 我们把这一现象称为“磁臂粘着”(Armstickiness)。在高密度磁盘上容易出现此情况。 N步SCAN算法是将磁盘请求队列分成若干个长度为N的子队列，磁盘（外圈）调度将按FCFS算法依次处理这些子队列。 而每处理一个队列时（内圈）又是按SCAN（电梯）算法，对一个队列处理完后，再处理其他队列。 当正在处理某子队列时，如果又出现新的磁盘I/O请求，便将新请求进程放入其他队列，这样就可避免出现粘着现象。 当N值取得很大时，会使N步扫描法的性能接近于SCAN算法的性能； 当N=1时， N步SCAN算法便蜕化为FCFS算法。

5.6.2 磁盘调度

★ N-Step-SCAN和FSCAN调度算法

2) FSCAN算法

FSCAN算法实质上是N步SCAN算法的简化，即FSCAN只将磁盘请求队列分成两个子队列。一个是由当前所有请求磁盘I/O的进程形成的队列，由磁盘调度按SCAN算法进行处理。在扫描期间，将新出现的所有请求磁盘I/O的进程，放入另一个等待处理的请求队列。这样，所有的新请求都将被推迟到下一次扫描时处理。

考研真题

【09年考研29题】假设磁头当前位于第105道，正在向磁道序号增加的方向移动。现有一个磁道访问请求序列为35，45，12，68，110，180，170，195，采用SCAN调度（电梯调度）算法得到的磁道访问序列是(A)

A. 110，170，180，195，68，45，35，12

B. 110，68，45，35，12，170，180，195

C. 110，170，180，195，12，35，45，68

D. 12，35，45，68，110，170，180，195

考研真题

【15年考研28题】在系统内存中设置磁盘缓冲区的主要目的是（ A ）

A、减少磁盘I/O 次数　　B、减少平均寻道时间

C、提高磁盘数据可靠性　D、实现设备无关性

【14年考研27题】现有一个容量为10GB的磁盘分区，磁盘空间以簇(Cluster)为单位进行分配，簇的大小为4KB，若采用位图法管理该分区的空闲空间，即用一位(bit)标识一个簇是否被分配，则存放该位图所需簇的个数为(A)

A 80　　B 320　　C 80K　　D 320K

考研真题

【17考研26题】某文件系统的簇和磁盘扇区大小分别为1KB和512B。若一个文件的大小为1026B，则系统分配给该文件的磁盘空间大小是（ D）

A.1026B　　B.1536B　　C.1538B　　D.2048B

考研真题

【17考研29题】下列选项中，磁盘逻辑格式化程序所做的工作是（ B ）

Ⅰ.对磁盘进行分区

Ⅱ.建立文件系统的根目录

Ⅲ.确定磁盘扇区校验码所占位数

Ⅳ.对保存空闲磁盘块信息的数据结构进行初始化

A.仅Ⅱ　　B.仅Ⅱ、Ⅳ　　C.仅Ⅲ、Ⅳ　　D.仅Ⅰ、Ⅱ、Ⅳ

考研真题

【13年考研25题】用户程序发出磁盘I/O请求后，系统的处理流程是：用户程序→系统调用处理程序→设备驱动程序→中断处理程序。其中，计算数据所在磁盘的柱面号、磁头号、扇区号的程序是（ C ）

A. 用户程序

B. 系统调用处理程序

C. 设备驱动程序

D. 中断处理程序

考研真题

【10年考研45题】假设计算机系统采用CSCAN（循环扫描）磁盘调度策略，使用2KB的内存空间记录16384个磁盘块的空间状态。

（1）请说明在上述条件下如何进行磁盘块空闲状态管理。

（2）设某单面磁盘旋转速度为每分钟6000转。每个磁道有100个扇区，相临磁道间的平均移动时间为1ms. 若在某时刻，磁头位于100号磁道处，并沿着磁道号大的方向移动，磁道号请求队列为50，90，30，120，对请求队列中的每个磁道需读取1个随机分布的扇区，则读完这个扇区点共需要多少时间？要求给出计算过程。

（1）2KB = 2*1024*8bit = 16384bit

因此可以使用位图法进行磁盘块空闲状态管理，每1bit表示一个磁盘块是否空闲。

（2）根据CSCAN算法，被访问的磁道号顺序为100、120、30、50、90，因此，寻道用去的总时间为：

（20 + 90 + 20 + 40）* 1ms = 170ms

每分钟6000转，转一圈的时间为0.01s，通过一个扇区的时间为0.0001s，总共要随机读取四个扇区，用去的时间为：

（0.01*0.5 + 0.0001）* 4 = 0.0204s = 20.4ms

则读完这个扇区点共需要的时间为：**170+20.4=190.4ms**

作 业

- 假设磁盘有200个磁道，磁盘请求队列中是一些随机请求，它们按照到达的次序分别处于55, 58, 39, 18, 90, 160, 150, 38, 184号磁道上，当前磁头在100号磁道上，并向磁道号增加的方向上移动。请给出按FCFS, SSTF, SCAN及CSCAN算法记性磁盘调度时满足请求的次序，并计算出它们的平均寻道长度。

Unit 6

- 文件前面的部分考小题
- 逻辑结构，物理结构
- 组织形式划分（必考）
- 连续分配的主要优缺点
- 混合索引模式（必考）（小题+大题）（要写过程）
- 硬链接，软链接（符号链接）
- 作业必考

6.2 文件的逻辑结构

- 对于任何一个文件，都存在着以下两种形式的结构：

1）**文件的逻辑结构**(File Logical Structure)

- 是用户可以直接处理的数据及其结构。
- 独立于文件的物理特性。

2）**文件的物理结构**

- 又称为文件的存储结构，是指文件在外存上的存储组织形式。

6.2 .1文件逻辑结构的类型

★有结构文件

记录的组织形式有：

- 顺序文件
- 索引文件
- 索引顺序文件

6.3.1 连续分配

★连续分配的主要优缺点

优点：

- 顺序访问容易。
- 顺序访问速度快。

缺点：

- 要求有连续的存储空间。
- 必须事先知道文件的长度。

6.3.3 索引分配

★多级索引分配

- 加快了对大型文件的查找速度
- 访问文件数据时需要多次启动磁盘，随着索引级数增加而增加，不利于小文件。

6.3.3 索引分配

★混合索引分配方式

- 直接地址
  - 为了提高对文件的检索速度，在索引结点中可设置10个直接地址项，即用iaddr(0)～iaddr(9)来存放直接地址。
  - 每项中所存放的是该文件数据的盘块的盘块号。假如每个盘块的大小为 4 KB，当文件不大于40 KB时，便可直接从索引结点中读出该文件的全部盘块号。

6.3.3 索引分配

★混合索引分配方式

1）一次间接地址

- 对于大、中型文件，可利用索引结点中的地址项iaddr(10)来提供一次间接地址。
- 实质就是一级索引分配方式。系统将分配给文件的多个盘块号记入其中。如果一个索引大小为4字节，那么在一次间址块中可存放1K个盘块号， 因而允许文件长达4MB。

6.3.3 索引分配

★混合索引分配方式

2）多次间接地址

- 对于当文件长度大于4 MB+40 KB时(一次间址与10个直接地址项)，系统还须采用二次间址分配方式。用地址项iaddr(11)提供二次间接地址。
- 实质是两级索引分配方式。系统此时是在二次间址块中记入所有一次间址块的盘号。在采用二次间址方式时，文件最大长度可达4GB。同理，地址项iaddr(12)作为三次间接地址，其所允许的文件最大长度可达4TB。

例题

- UNIX的文件系统采用三级索引机制。在文件控制块(FCB)中，设置了一个索引表，共有13个索引地址。其中，前10个为直接索引地址，后3个为间接索引地址，包括1个一级索引地址、1个二级索引地址和1个三级索引地址。假定磁盘块的大小为512个字节，每个索引地址占4字节，那么UNIX系统允许一个文件最大有多大？

【17考研31题】若文件f1的硬链接为f2，两个进程分别打开f1和f2，获得对应的文件描述符为fd1和fd2，则下列叙述种，正确的是（B）

Ⅰ.f1和f2的读写指针位置保持相同

Ⅱ.f1和f2共享同一个内存索引结点

Ⅲ.fd1和fd2分别指向各自的用户打开文件表中的一项

A.仅Ⅲ　　B.仅Ⅱ、Ⅲ　C.仅Ⅰ、Ⅱ　D.仅Ⅰ、Ⅱ、Ⅲ

作 业

- 请分别解释在连续分配方式，隐式链接分配方式，显示链接分配方式和索引分配方式中如何将文件的字节3500转换为物理块号和块内位移量（设盘块大小为1KB，盘块号需占4个字节）。

作 业

有一计算机系统利用图6.9所示的位示图（行号，列号都从0开始编号）来管理空闲盘块。如果盘块从1开始编号，每个盘块的大小为1KB。

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5																
6																

图6.9 位示图