希赛

操作系统

课程内容提要

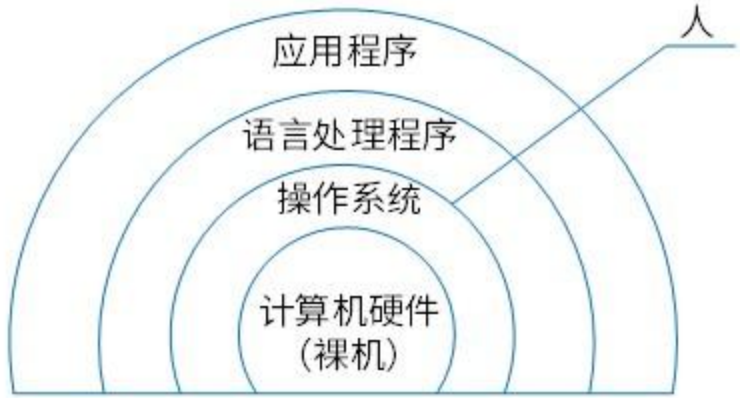
希赛

- **操作系统概述**（★）
- **进程管理**
  - 进程的状态（★）
  - 前趋图（★★★★）
  - 信号量与PV操作（★★★★★）
  - 死锁及银行家算法（★★★）
- **存储管理**
  - 段页式存储（★★★）
  - 页面置换算法（★）
- **文件管理**
  - 绝对路径与相对路径（★）
  - 索引文件（★）
  - 位示图（★★★）
- **设备管理**
  - 数据传输控制方式（★★★）
- **嵌入式操作系统**
  - 实时操作系统（★★）
  - 微内核操作系统（★★）

操作系统概述

希赛

- 管理系统的硬件、软件、数据资源
- 控制程序运行
- 人机之间的接口
- 应用软件与硬件之间的接口



- 进程管理
- 存储管理
- 文件管理
- 作业管理
- 设备管理

操作系统（OS，Operating System）

操作系统概述

希赛

计算机系统中硬件层之上的软件通常按照三层来划分，如下图所示，图中①②③分别表示（ ）。

A 操作系统、应用软件和其他系统软件

B 操作系统、其他系统软件和应用软件

C 其他系统软件、操作系统和应用软件

D 应用软件、其他系统软件和操作系统

进程管理 - 进程的概念

希赛

进程是程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位。它由程序块、进程控制块（PCB）和数据块三部分组成。

PCB，PCB是进程存在的唯一标志。内容包含进程标识符、状态、位置信息、控制信息、队列指针（链接同一状态的进程）、优先级、现场保护区等。

进程管理 - 进程与程序

希赛

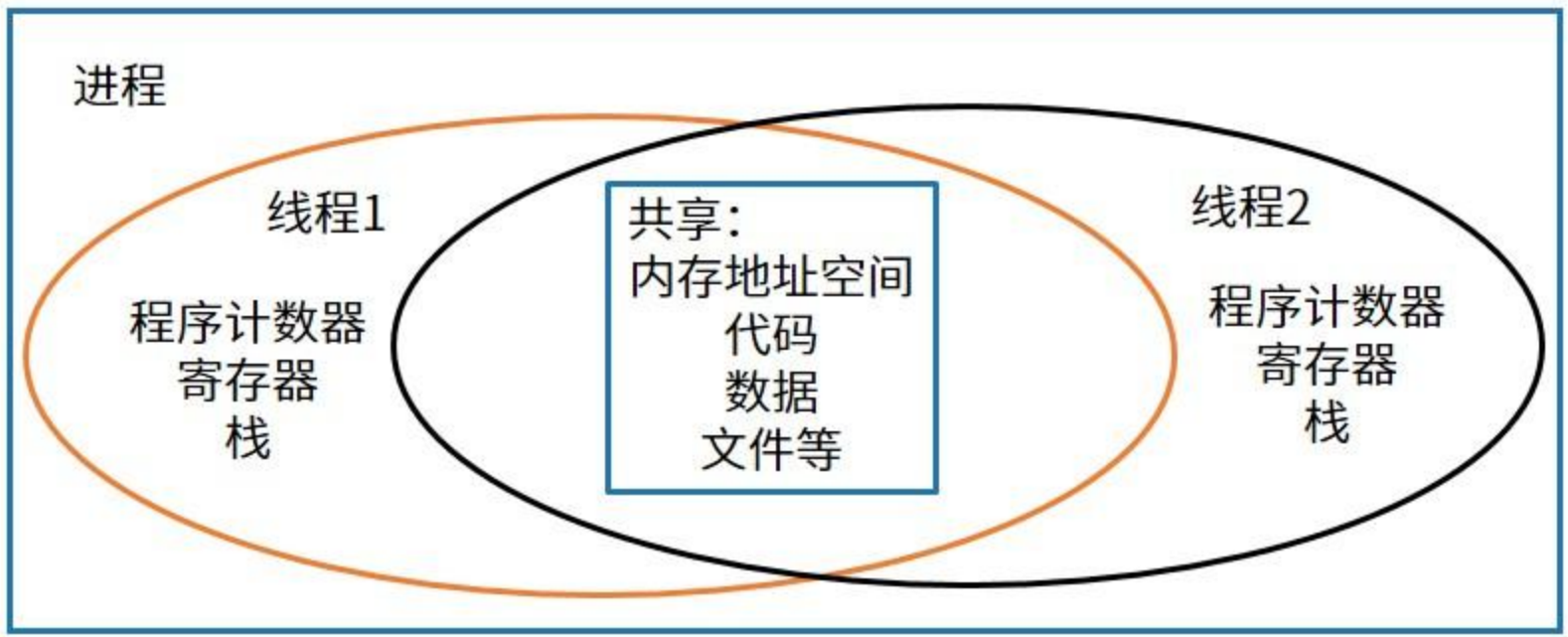
- 进程与程序的区别：进程是程序的一次执行过程，没有程序就没有进程。

- 程序是一个静态的概念，而进程是一个动态的概念，它由创建而产生，完成任务后因撤销而消亡；进程是系统进行资源分配和调度的独立单位，而程序不是。

进程管理 - 进程与线程

希赛

进程的2个基本属性：可拥有资源的独立单位；可独立调度和分配资源的基本单位。



进程管理

希赛

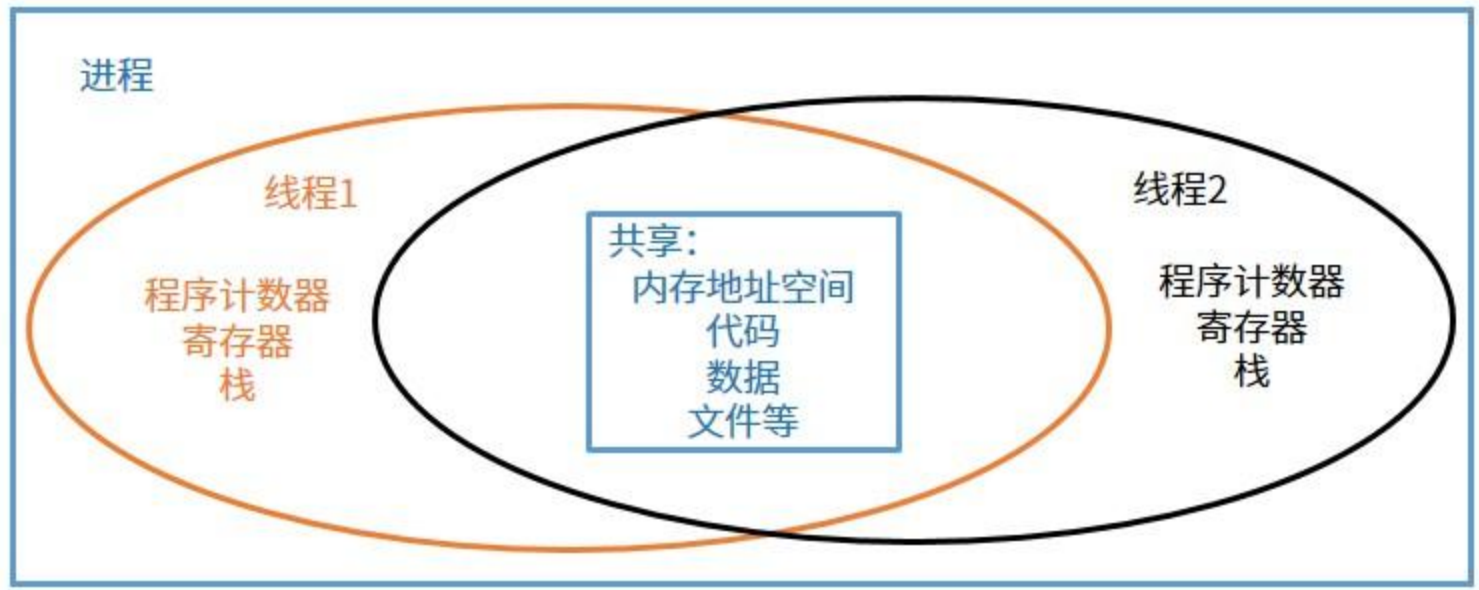
在支持多线程的操作系统中，假设进程P创建了若干个线程，那么（ ）是不能被这些线程共享的。

A 该进程中打开的文件

B 该进程的代码段

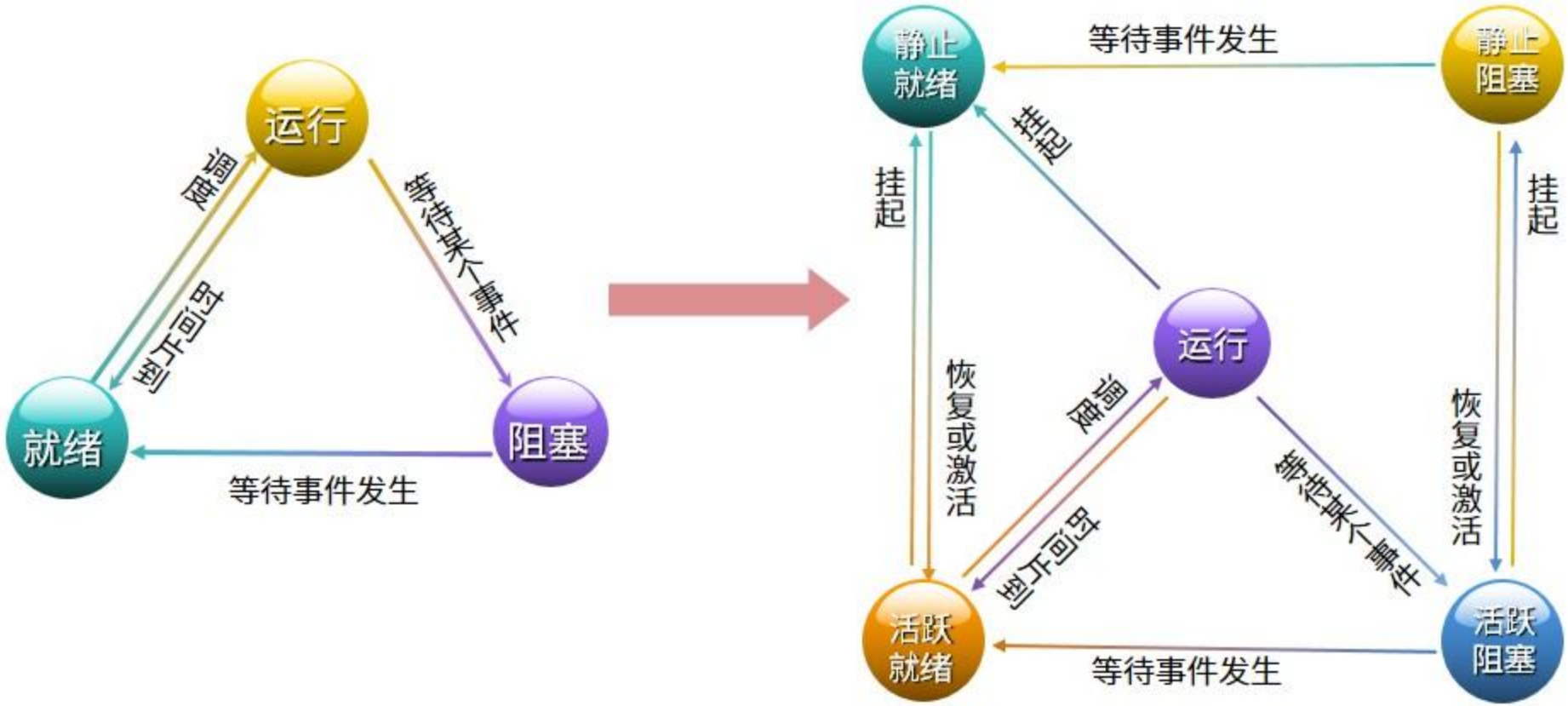
C 该进程中某线程的栈指针

D 该进程的全局变量



进程管理-进程的状态

希赛



进程管理-进程的状态

希赛

某计算机系统中的进程管理采用三态模型，那么下图所示的PCB（进程控制块）的组织方式采用（ ），图中（ ）。

A 顺序方式

B 链接方式

C 索引方式

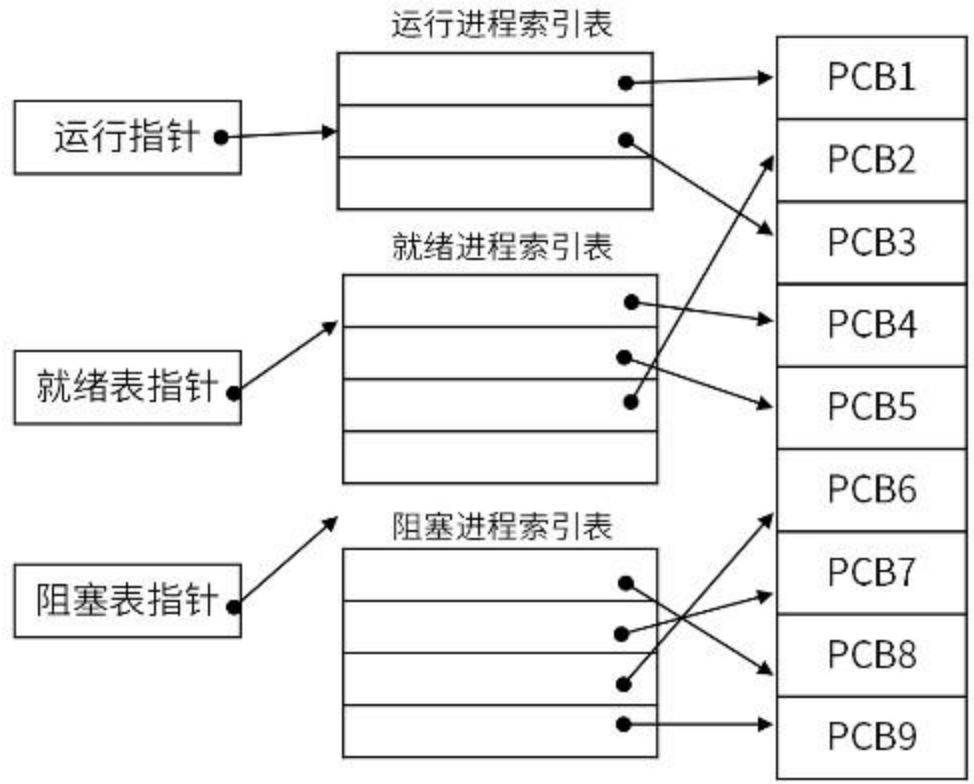
D Hash

A 有1个运行进程，2个就绪进程，4个阻塞进程

B 有2个运行进程，3个就绪进程，3个阻塞进程

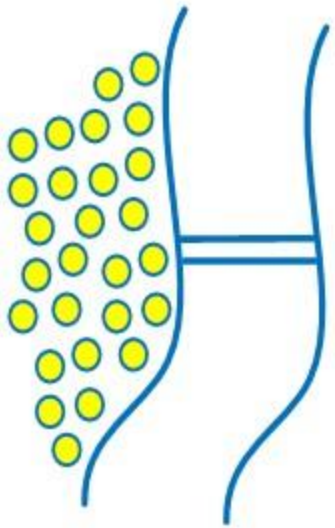
C 有2个运行进程，3个就绪进程，4个阻塞进程

D 有3个运行进程，2个就绪进程，4个阻塞进程

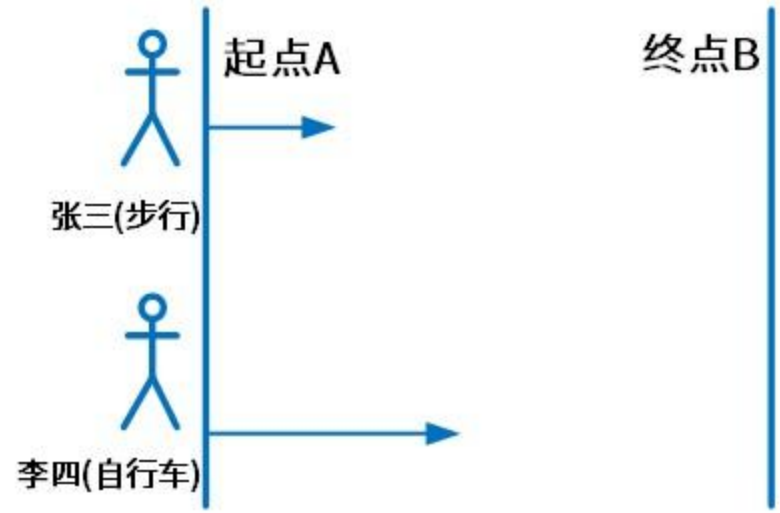


进程管理-进程的同步与互斥

希赛



互斥：如千军万马过独木桥



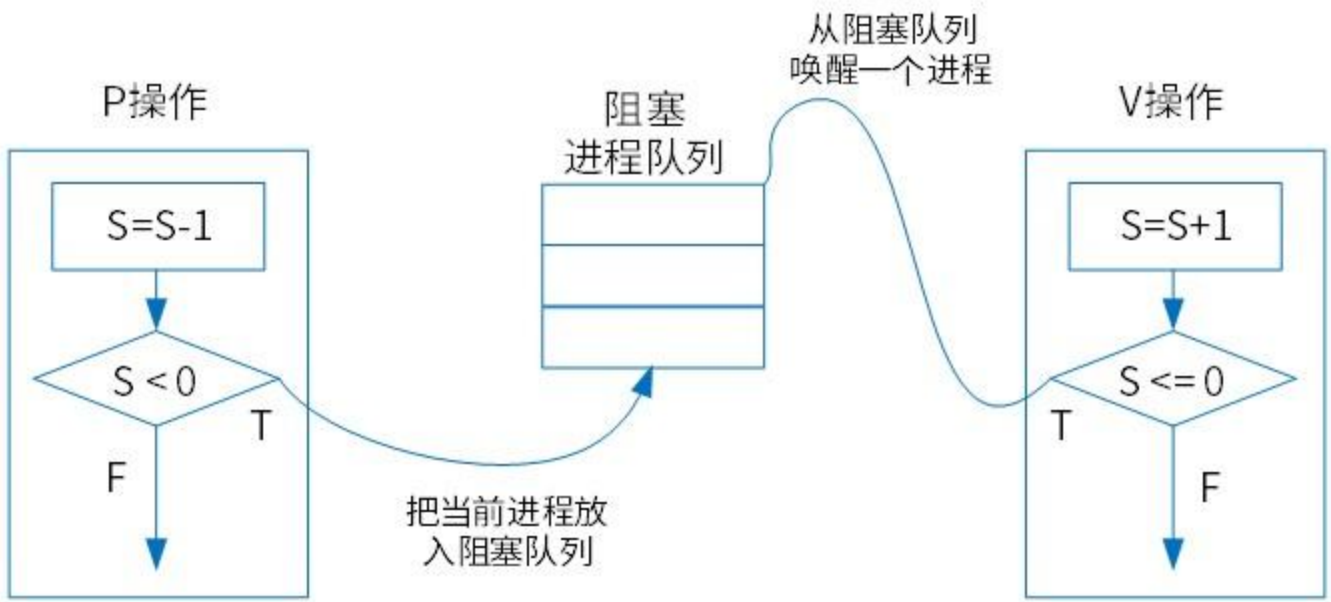
同步：速度有差异，在一定情况停下等待

进程管理-进程的同步与互斥

希赛

- 临界资源：诸进程间需要互斥方式对其进行共享的资源，如打印机、磁带机等
- 临界区：每个进程中访问临界资源的那段代码称为临界区
- 信号量：是一种特殊的变量

注：P是荷兰语的Passeren，V是荷兰语的Verhoog。



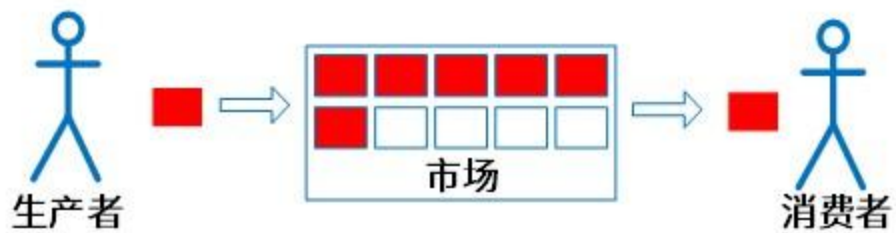
进程管理-进程的同步与互斥

希赛

生产者消费者问题

单缓冲区情况

多缓冲区情况



进程管理-PV操作

希赛

未使用PV操作：

生产者：	消费者：
生产一个产品；	从缓冲区取产品；
送产品到缓冲区；	消费产品；

使用了PV操作：

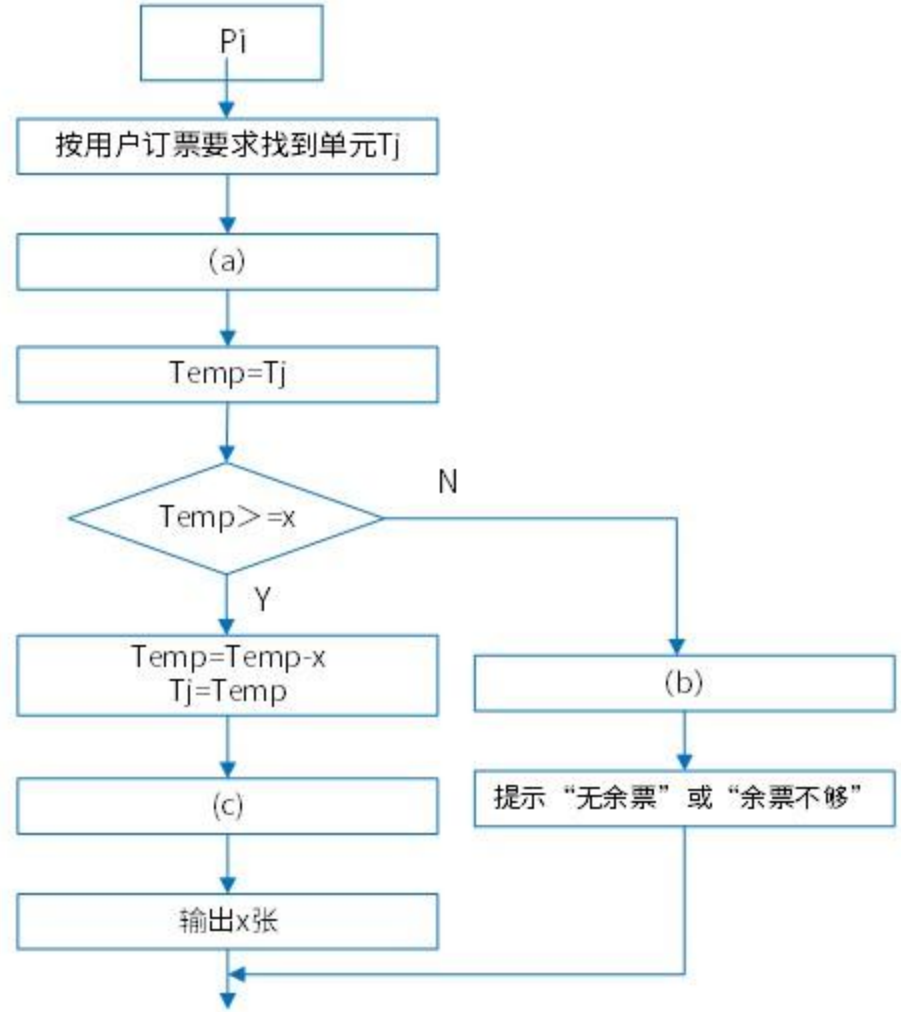
生产者：	消费者：
生产一个产品；	P(S2)；
P(S1)；	从缓冲区取产品；
送产品到缓冲区；	V(S1)；
V(S2)；	消费产品；

S1初值为1，S2初值为0

进程管理-PV操作

希赛

某航空公司机票销售系统有n个售票点，该系统为每个售票点创建一个进程Pi（i=1，2，…，n）管理机票销售。假设Tj（j=1，2，…，m）单元存放某日某航班的机票剩余票数，Temp为Pi进程的临时工作单元，x为某用户的订票张数。初始化时系统应将信号量S赋值为（ ）。Pi进程的工作流程如下图所示，若用P操作和V操作实现进程间的同步与互斥，则图中空（a），空（b）和空（c）处应分别填入（ ）。



A 0

B 1

C 2

D 3

A P(S)，V(S)和V(S)

B P(S)，P(S)和V(S)

C V(S)，P(S)和P(S)

D V(S)，V(S)和P(S)

进程管理-PV操作

希赛

假设某系统采用非抢占式优先级调度算法，若该系统有两个优先级相同的进程P1和P2，各进程的程序段如下所示，若信号量S1和S2的初值都为0。进程P1和P2并发执行后a、b和c的结果分别为：a= （4），b= （5），c= （6）。

P1程序段

```
begin{
   A =1;
   A =a+1;
   V(S1);
   C =a+5;
   P(S2);
   A =a+c;
  }
end
```

P2程序段

```
begin{
   B =2;
   B =b+1;
   P(S1);
   B =a+b;
   V(S2);
   C =b+c;
  }
end
```

(4)	A 9	B 12	C 13	D 14
(5)	A 5	B 6	C 9	D 10
(6)	A 4	B 6	C 12	D 13

进程管理-PV操作

希赛

P1程序段

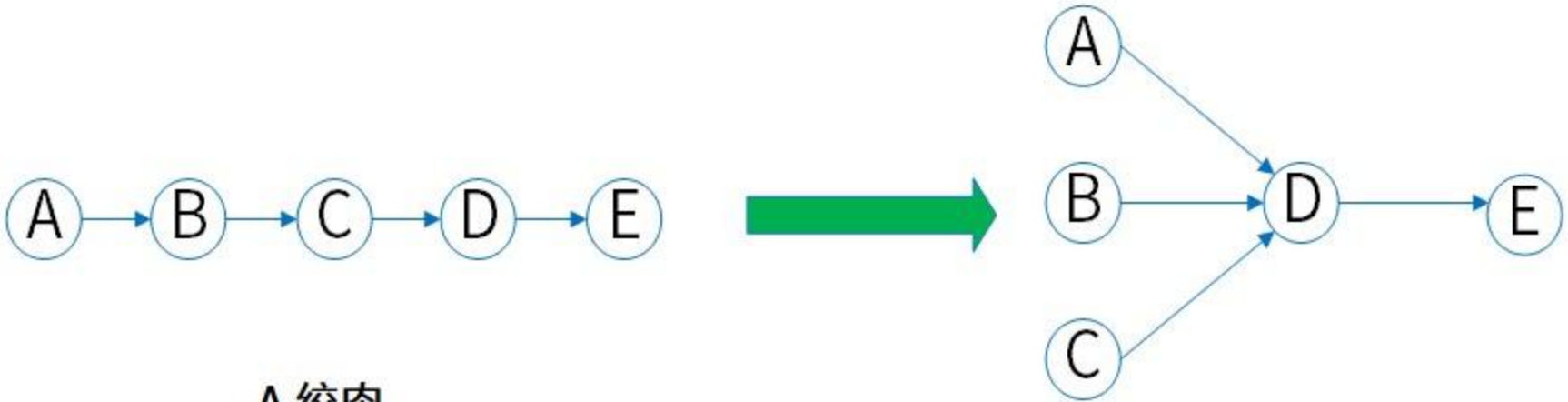
```
begin{
   A =1;
   A =a+1; // a=2
   V（S1）；
   C =a+5; // c=7
   P（S2）；//唤醒
   A =a+c; // a=14
  }
end
```

P2程序段

```
begin{
   B =2;
   B =b+1; // b=3
   P（S1）；//唤醒
   B =a+b; // b=5
   V（S2）；
   C =b+c; // c=12
  }
end
```

进程管理-前趋图

希赛



A 绞肉

B 切葱末

C 切姜末

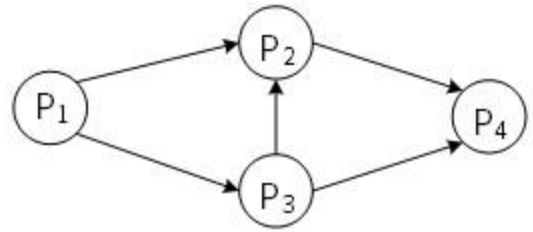
D 搅拌

E 包饺子

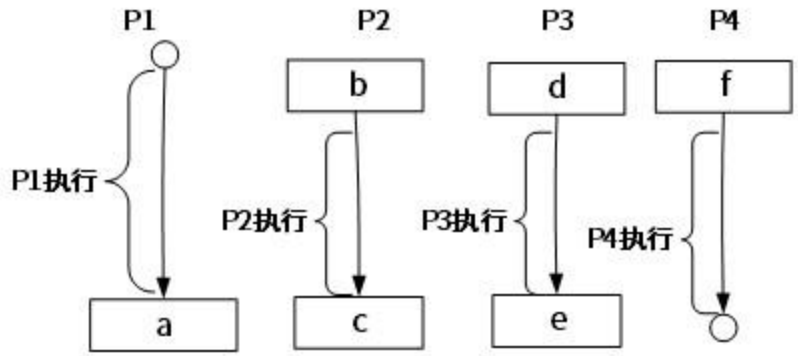
进程管理-PV操作

希赛

进程P1、P2、P3和P4的前趋图如下所示：



若用PV操作控制进程P1~P4并发执行的过程，则需要设置5个信号量S1、S2、S3、S4和S5，且信号量S1-S5的初值都等于0。下图中a、b和c处应分别填写（ ）；d、e和f处应分别填写（ ）。



A VS1)V(S2)、P(S1)V(S3)和V(S4)

B P(S1)V(S2)、P(S1)P(S2)和V(S1)

C V(S1)V(S2)、P(S1)P(S3)和V(S4)

D P(S1)P(S2)、V(S1)P(S3)和V(S2)

A P(S2)、V(S3)V(S5)和P(S4)P(S5)

B V(S2)、P(S3)V(S5)和V(S4)P(S5)

C P(S2)、V(S3)P(S5)和P(S4)V(S5)

D V(S2)、V(S3)P(S5)和P(S4)V(S5)

进程管理-死锁

希赛

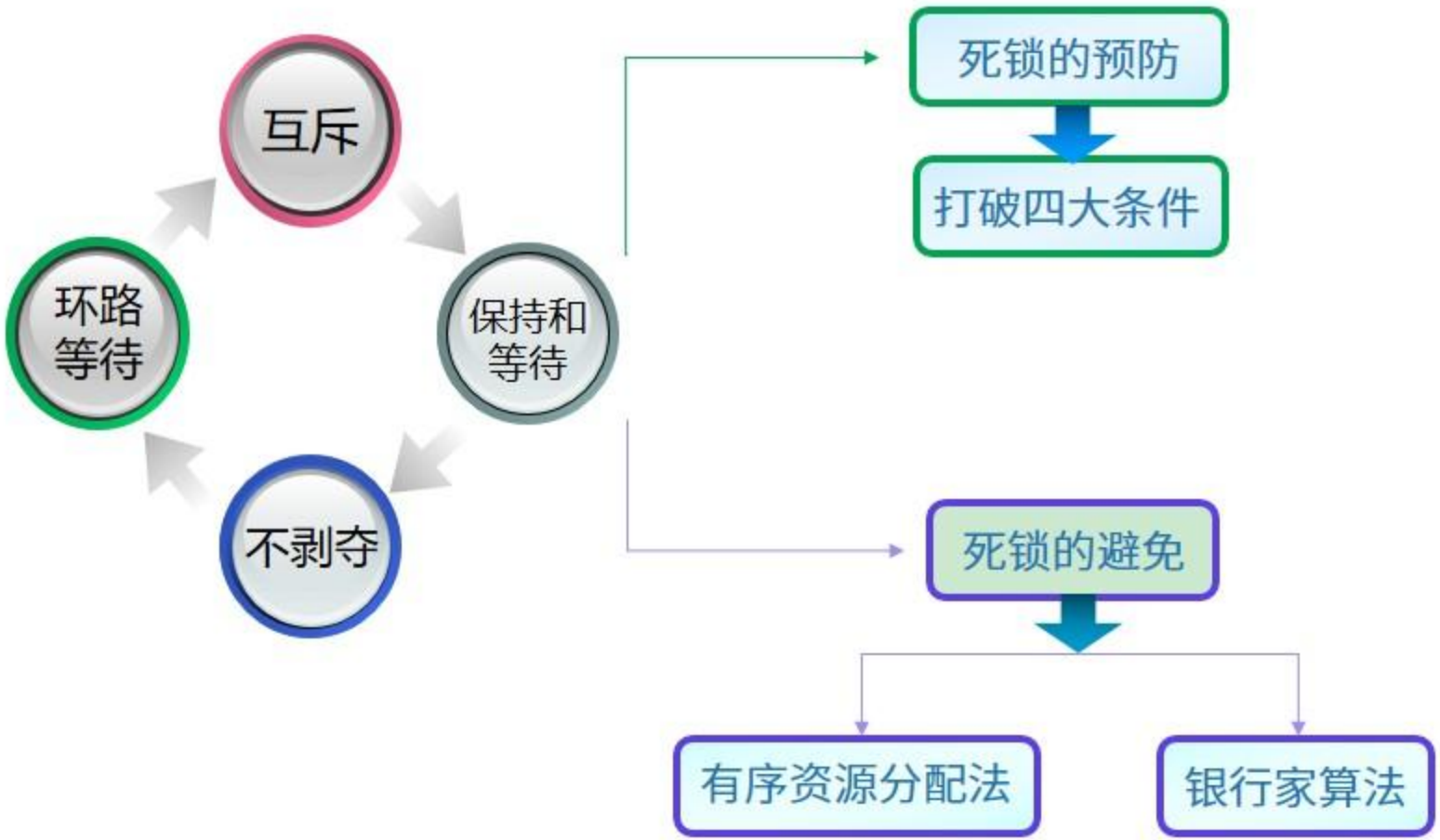
- 进程管理是操作系统的核心，但如果设计不当，就会出现死锁的问题。如果一个进程在等待一件不可能发生的事，则进程就死锁了。而如果一个或多个进程产生死锁，就会造成系统死锁。

例：系统有3个进程：A、B、C。这3个进程都需要5个系统资源。如果系统至少有多少个资源，则不可能发生死锁。

进程A	进程B	进程C

进程管理-死锁

希赛



进程管理 – 银行家算法

希赛

银行家算法：分配资源的原则

- ★ 当一个进程对资源的最大需求量不超过系统中的资源数时可以接纳该进程
- ★ 进程可以分期请求资源，但请求的总数不能超过最大需求量
- ★ 当系统现有的资源不能满足进程尚需资源数时，对进程的请求可以推迟分配，但总能使进程在有限的时间里得到资源

进程管理 – 银行家算法

希赛

银行家算法例子：

假设系统中有三类互斥资源R1、R2、R3，可用资源分别是9、8、5。在T0时刻系统中有P1、P2、P3、P4和P5五个进程，这些进程对资源的最大需求量和已分配资源数如下所示，如果进程按_____序列执行，那么系统状态是安全的。

进程资源表

进程 \ 资源	最大需求量			已分配资源数		
	R1	R2	R3	R1	R2	R3
P1	6	5	2	1	2	1
P2	2	2	1	2	1	1
P3	8	1	1	2	1	0
P4	1	2	1	1	2	0
P5	3	4	4	1	1	3

供选择的答案：

A P1→P2→P4→P5→P3　　B P2→P4→P5→P1→P3

C P2→P1→P4→P5→P3　　D P4→P2→P5→P1→P3

进程管理 – 银行家算法

希赛

首先求剩下的资源数：

R1 = 9-(1+2+2+1+1) = 2

R2 = 8-(2+1+1+2+1) = 1

R3 = 5-(1+1+3) = 0

还需资源数表

进程 \ 资源	最大需求量			已分配资源数			还需资源数		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P 1	6	5	2	1	2	1	5	3	1
P 2	2	2	1	2	1	1	0	1	0
P 3	8	1	1	2	1	0	6	0	1
P 4	1	2	1	1	2	0	0	0	1
P 5	3	4	4	1	1	3	2	3	1

求序列P 2 →P 4 →P 5 →P 1 →P3是否安全：

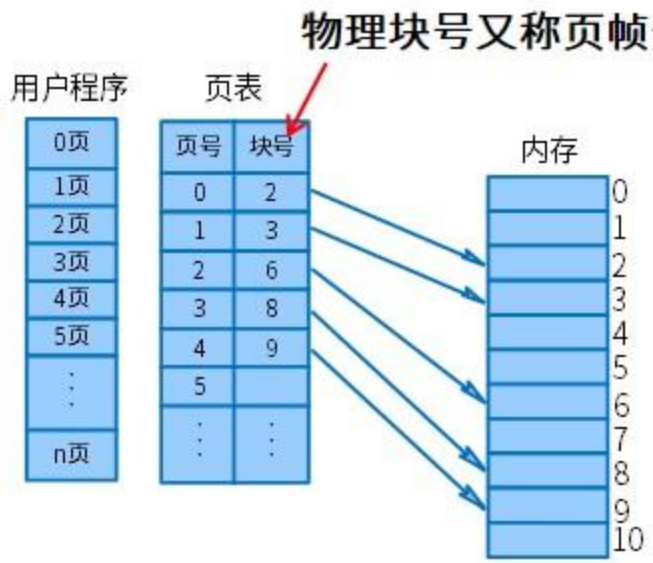
进程序列P 2 →P 4 →P 5 →P 1 →P3运行分析表

资源 \ 进程	现有资源			需要资源			已经分配			现有+已经分配			完成
	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	
P2	2	1	0	0	1	0	2	1	1	4	2	1	True
P4	4	2	1	0	0	1	1	2	0	5	4	1	True
P5	5	4	1	2	3	1	1	1	3	6	5	4	True
P1	6	5	4	5	3	1	1	2	1	7	7	5	True
P3	7	7	5	6	0	1	2	1	0	9	8	5	True

存储管理-页式存储

希赛

页式存储：将程序与内存均划分为同样大小的块，以页为单位将程序调入内存。



高级程序语言使用逻辑地址；
运行状态，内存中使用物理地址。

逻辑地址=页号+页内地址
物理地址=页帧号+页内地址

例如，页式存储系统中，每个页的大小为4KB。
逻辑地址：
10 1100 1101 1110
对应的物理地址为：
110 1100 1101 1110

★ 优点：利用率高，碎片小，分配及管理简单

★ 缺点：增加了系统开销；可能产生抖动现象

存储管理-页式存储

希赛

高级程序语言中使用

1：在内存中
0：不在内存中

1：内容被修改过
0：内容未被修改

内存中使用

1：最近访问过
0：最近未被访问

页号 (逻辑)	页帧号 (物理)	状态位	访问位	修改位
0	2	1	1	0
1	3	1	0	1
2	5	1	1	0
3	—	0	0	0
4	—	0	0	0
5	6	1	1	1

存储管理-页式存储

希赛

进程P有8个页面，页号分别为0~7，页面大小为4K ，假设系统给进程P分配了4个存储块，进程P的页面变换表如下所示。表中状态位等于1和0分别表示页面在内存和不在内存。若进程P要访问的逻辑地址为十六进制 5148H，则该地址经过变换后， 其物理地址应为十六进制（ ）；如果进程P要访问的页面6不在内存，那么应该淘汰页号为（ ）的页面。

页号	页帧号	状态位	访问位	修改位
0	-	0	0	0
1	7	1	1	0
2	5	1	0	1
3	-	0	0	0
4	-	0	0	0
5	3	1	1	1
6	-	0	0	0
7	9	1	1	0

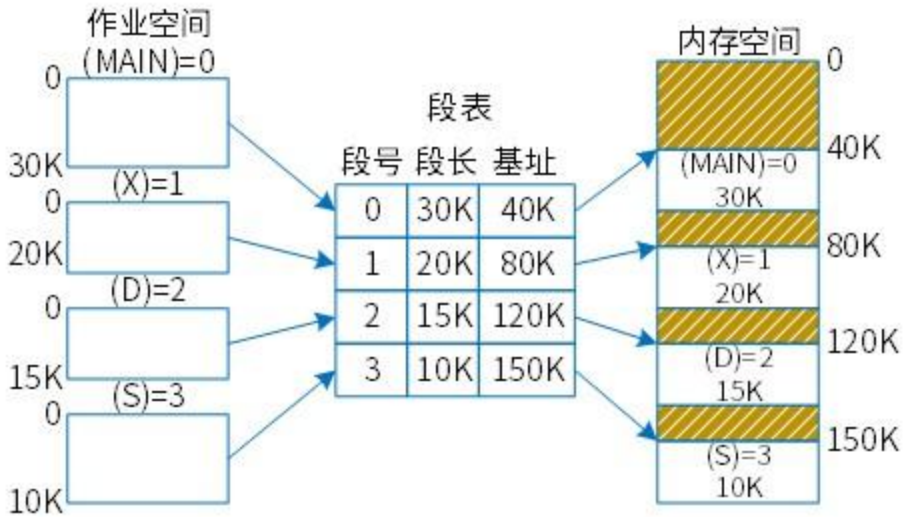
A 3148H　　B 5148H　　C 7148H　　D 9148H

A 1　　B 2　　C 5　　D 9

存储管理-段式存储

希赛

段式存储：按用户作业中的自然段来划分逻辑空间，然后调入内存，段的长度可以不一样。



逻辑段地址：（段号，段内偏移量）
合法段地址：（0，25K）
非法段地址：（0，35K）

★ 优点：多道程序共享内存，各段程序修改互不影响

★ 缺点：内存利用率低，内存碎片浪费大

存储管理-段式存储

希赛

假设系统采用段式存储管理方法，进程P的段表如下所示。逻辑地址（ ）不能转换为对应的物理地址；不能转换为对应的物理地址的原因是进行（ ）。

段号	基地址	段长
0	1100	800
1	3310	50
2	5000	200
3	4100	580
4	2000	100

A （0，790）和（2，88）

B （1，30）和（3，290）

C （2，88）和（4，98）

D （0，810）和（4，120）

A 除法运算时除数为零

B 算术运算时有溢出

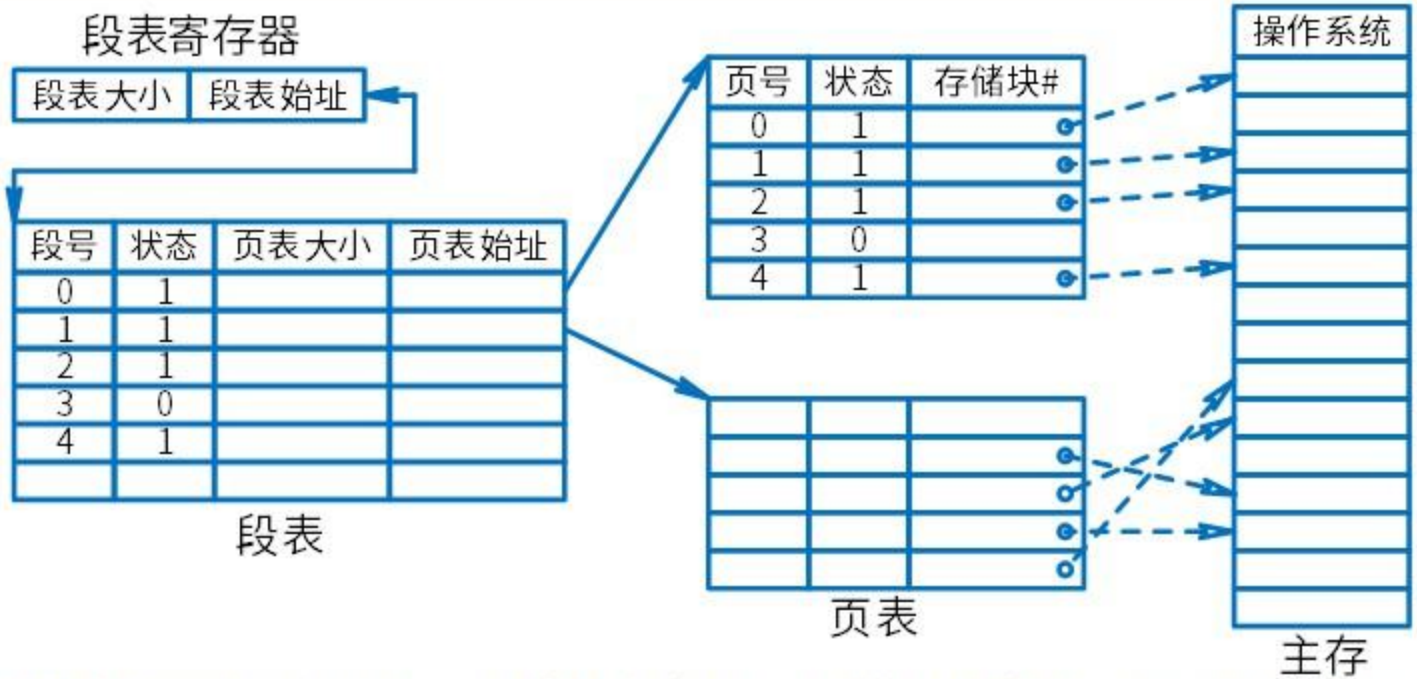
C 逻辑地址到物理地址转换时地址越界

D 物理地址到逻辑地址转换时地址越界

存储管理-段页式存储

16 14	13 9	8 0
段号	页号	页内地址

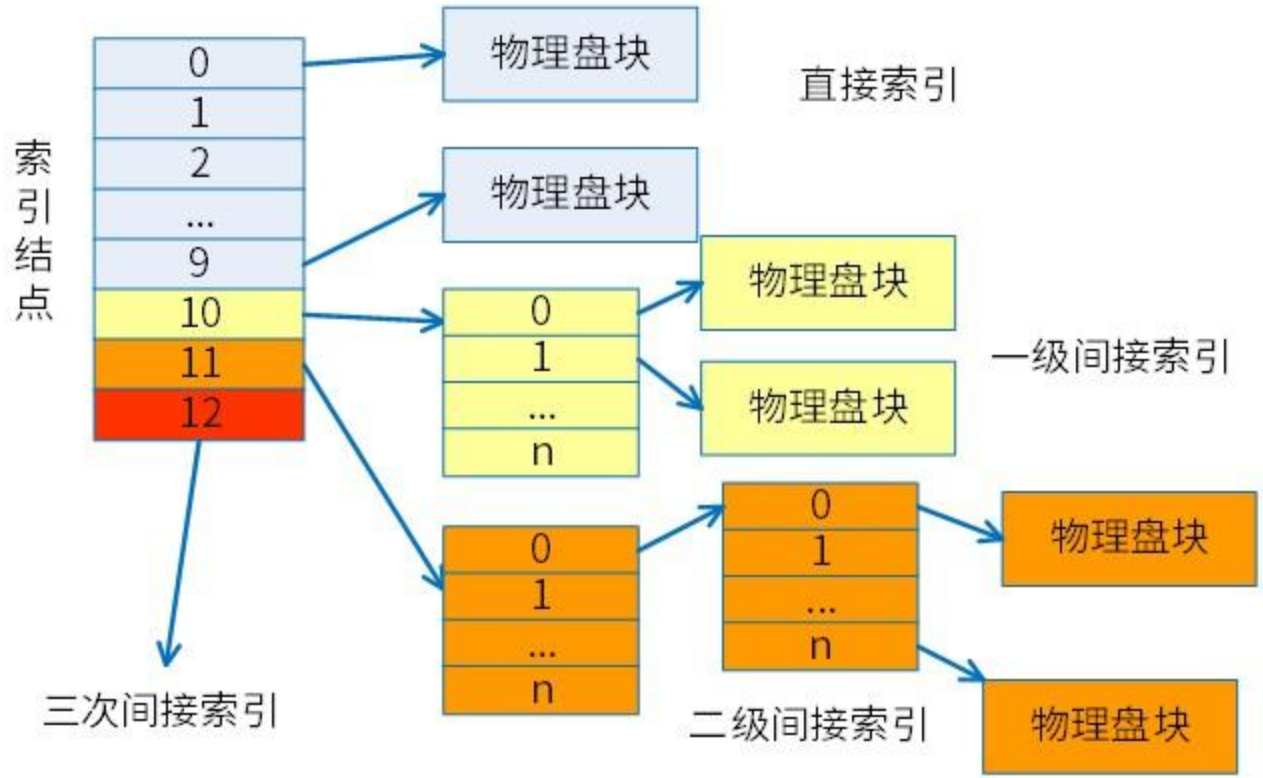
段页式存储：段式与页式的综合体。先分段，再分页。1个程序有若干个段，每个段中可以有若干页，每个页的大小相同，但每个段的大小不同。



★ 优点：空间浪费小、存储共享容易、存储保护容易、能动态连接

★ 缺点：由于管理软件的增加，复杂性和开销也随之增加，需要的硬件以及占用的内容也有所增加，使得执行速度大大下降

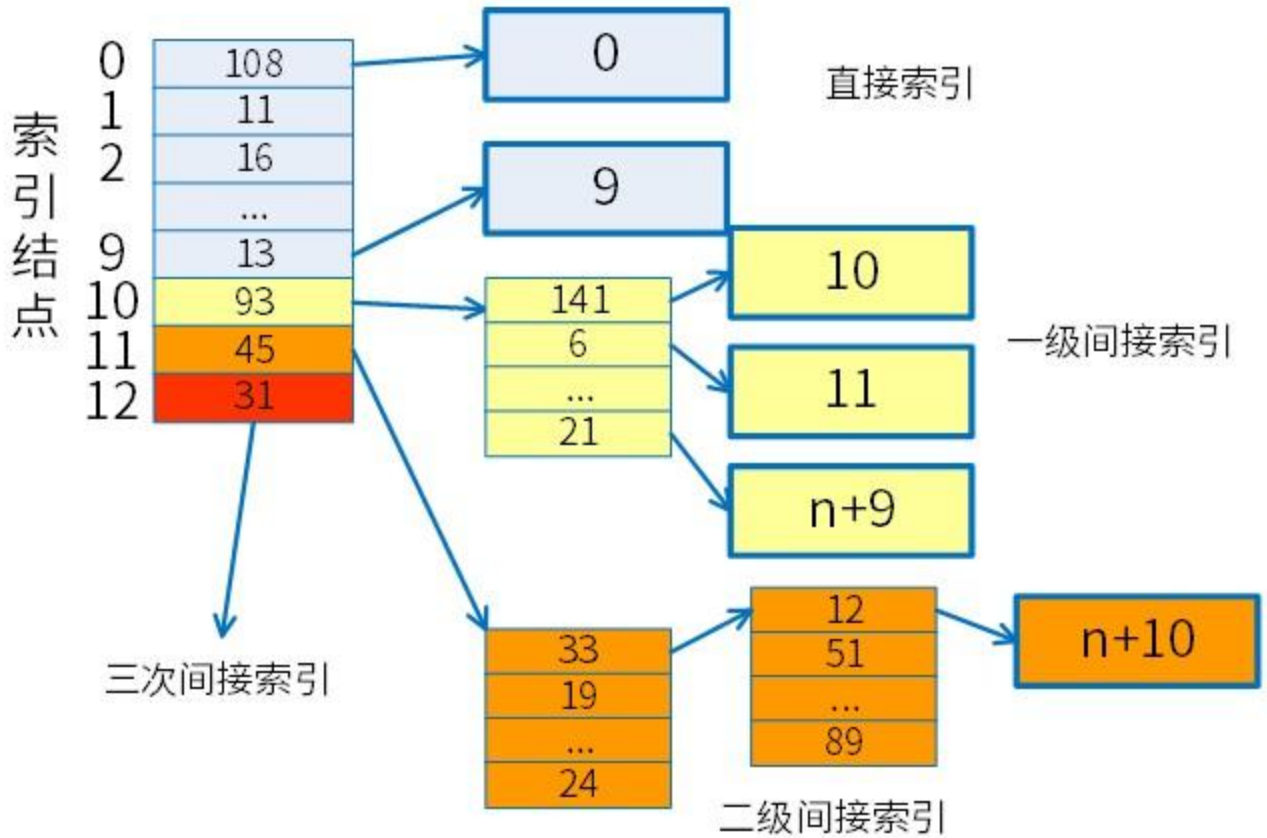
文件管理-索引文件结构


索引结点
0
1
2
...
9
10
11
12
物理盘块
物理盘块
直接索引
0
1
...
n
物理盘块
物理盘块
一级间接索引
0
1
...
n
0
1
...
n
物理盘块
物理盘块
二级间接索引
三次间接索引

文件管理-索引文件结构

希赛

文件在逻辑上一定是连续的，在物理上可以是分散的。



- ✓ 逻辑位置（字节或页号）对应的索引方式
- ✓ 不同索引方式指向的对象（数据块&索引表）
- ✓ 不同索引方式访问磁盘的次数
- ✓ 能够表示的文件长度

文件管理-索引文件结构

希赛

假设文件系统采用索引节点管理，且索引节点有8个地址项iaddr[0]～iaddr[7]，每个地址项大小为4字节，iaddr[0]～iaddr[4]采用直接地址索引，iaddr[5]和iaddr[6]采用一级间接地址索引，iaddr[7]采用二级间接地址索引。假设磁盘索引块和磁盘数据块大小均为1KB字节，文件File1的索引节点如图所示。若用户访问文件File1中逻辑块号为5和261的信息，则对应的物理块号分别为（ ）；101号物理块存放的是（ ）。

A 89和90　　B 89和136

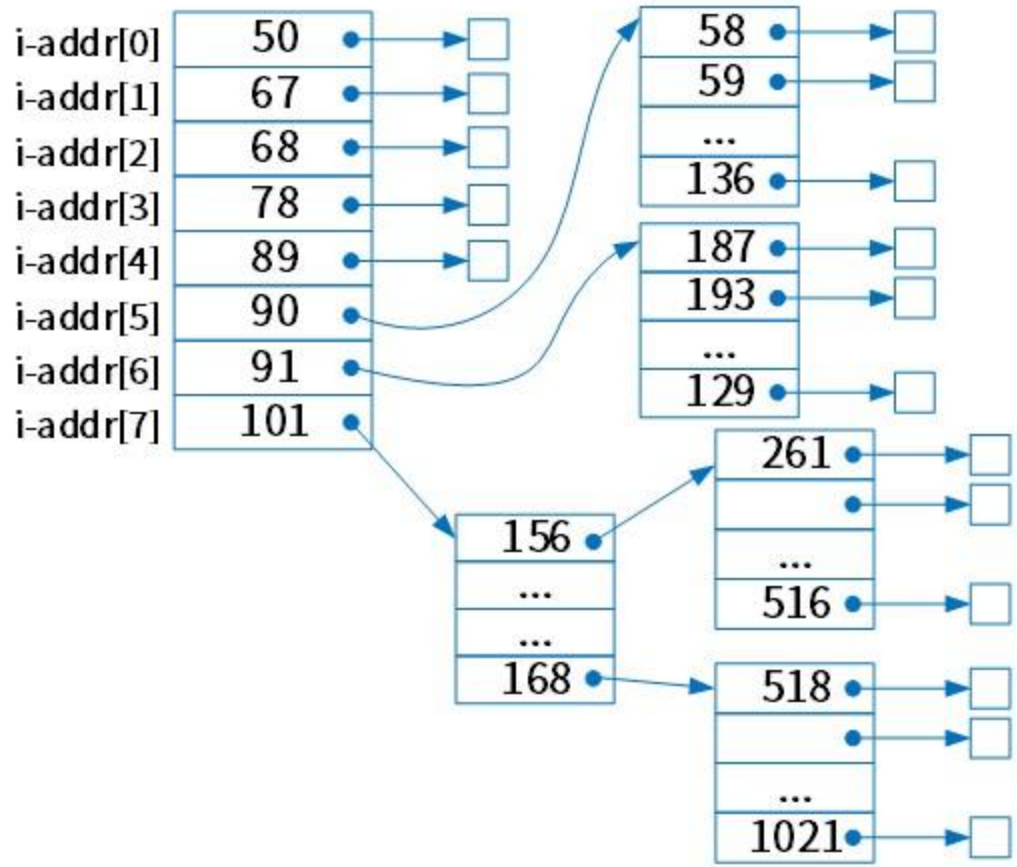
C 58和187　　D 90和136

A File1的信息

B 直接地址索引表

C 一级地址索引表

D 二级地址索引表



文件管理-位示图

希赛

- ✓ 空闲区表法（空闲文件目录）
- ✓ 空闲链表法
- ✓ 位示图法
- ✓ 成组链接法

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

位示图

文件管理-位示图

希赛

某文件管理系统在磁盘上建立了位示图（bitmap），记录磁盘的使用情况。若磁盘上物理块的编号依次为0、1、2、…；系统中的字长为32位，字的编号依次为0、1、2、…，字中的一位对应文件存储器上的一个物理块，取值0和1分别表示空闲和占用，如下图所示。

字号

位号

	31	30	…		3	2	1	0
0	0	1	…	1	0	0	0	1
1	1	1	…	1	0	1	1	0
2	0	1	…	0	1	1	0	1
3	0	1	…	1	1	1	0	1
…			…					
n	1	1	…	0	1	0	0	1

假设操作系统将2053号物理块分配给某文件，那么该物理块的使用情况在位示图中编号为（ ）的字中描述；系统应该将（ ）。

A 32　　B 33　　C 64　　D 65

A 该字的位号5的位置“0”　　B 该字的位号5的位置“1”

C 该字的位号6的位置“0”　　D 该字的位号6的位置“1”

文件管理-位示图

希赛

字号

位号

	31	30	…		3	2	1	0
0	0	1	…	1	0	0	0	1
1	1	1	…	1	0	1	1	0
2	0	1	…	0	1	1	0	1
3	0	1	…	1	1	1	0	1
…			…					
n	1	1	…	0	1	0	0	1

(2053+1)/32 = 64.1875 -> 第65字（编号为64）。

64*32=2048 -> 0 – 2048。

第65字（64号字）中：

位号0-> 2048　　位号1 -> 2049　位号2 -> 2050　位号3 -> 2051　　位号4 -> 2052　　位号5 -> 2053

文件管理-树形目录结构

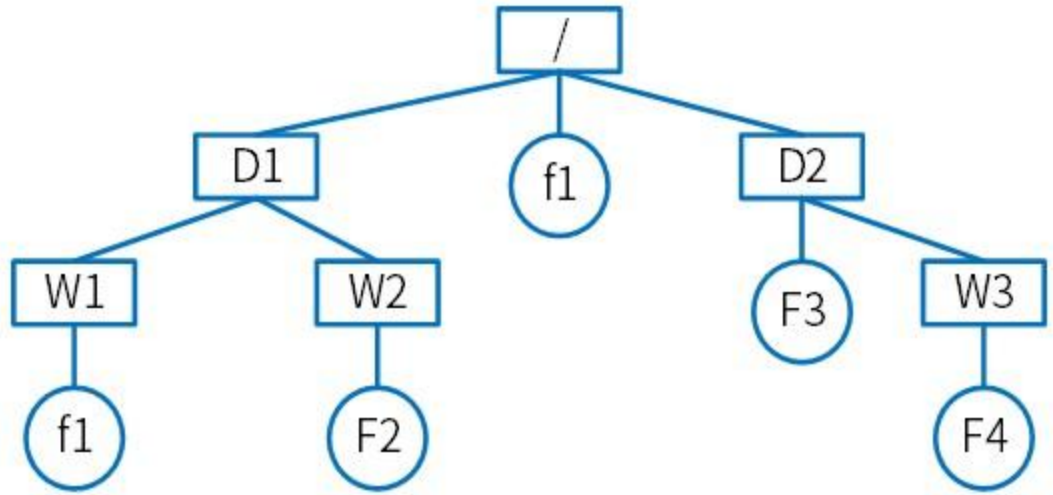
希赛

◆文件属性

- R 只读文件属性
- A 存档属性
- S 系统文件
- H 隐藏文件

◆文件名的组成

- 驱动器号
- 路径
- 主文件名
- 扩展名



★绝对路径：是从盘符开始的路径。

★相对路径：是从当前目录开始的路径。

★若当前目录为：D1，要求F2路径，则：绝对路径：/D1/W2/F2，相对路径：W2/F2

数据传输控制方式

希赛

效率越来越高

- ✓ 程序控制（查询）方式：分为无条件传送和程序查询方式两种。方法简单，硬件开销小，但I/O能力不高，严重影响CPU的利用率。
- ✓ 程序中断方式：与程序控制方式相比，中断方式因为CPU无需等待而提高了传输请求的响应速度。
- ✓ DMA方式：DMA方式是为了在主存与外设之间实现高速、批量数据交换而设置的。DMA方式比程序控制方式与中断方式都高效。

（DMAC向总线裁决逻辑提出总线请求；CPU执行完当前总线周期即可释放总线控制权。此时DMA响应，通过DMAC通知I/O接口开始DMA传输。）

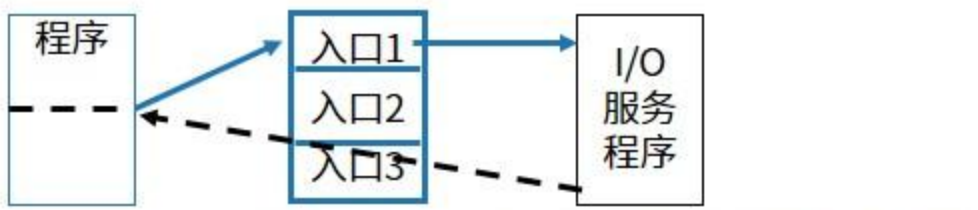
- ✓ 通道方式
- ✓ I/O处理机

数据传输控制方式

希赛

中断处理过程：

- ✓ CPU无需等待也不必查询I/O状态。
- ✓ 当I/O系统准备好以后，发出中断请求信号通知CPU；
- ✓ CPU接到中断请求后，保存正在执行程序的现场（保存现场），打断的程序当前位置即为断点；
- ✓ （通过中断向量表）转入I/O中的服务程序的执行，完成I/O系统的数据交换；
- ✓ 返回被打断的程序继续执行（恢复现场）。



中断向量表（保存中断服务程序的入口地址）

数据传输控制方式

希赛

计算机系统中常用的输入/输出控制方式有无条件传送、中断、程序查询和DMA 方式等。当采用（ ）方式时，不需要CPU执行程序指令来传送数据。

A 中断　　B 程序查询　　C 无条件传送　　D DMA

数据传输控制方式

希赛

计算机运行过程中，遇到突发事件，要求CPU暂时停止正在运行的程序，转去为突发事件服务，服务完毕，再自动返回原程序继续执行，这个过程称为（ ），其处理过程中保存现场的目的是（ ）。

A 阻塞　　B 中断　　C 动态绑定　　D 静态绑定

A 防止丢失数据　　B 防止对其他部件造成影响

C 返回去继续执行原程序　　D 为中断处理程序提供数据

I/O管理软件

希赛

- 硬件：完成具体的I/O操作。
- 中断处理程序：I/O完成后唤醒设备驱动程序
- 设备驱动程序：设置寄存器，检查设备状态
- 设备无关I/O层：设备名解析、阻塞进程、分配缓冲区
- 用户级I/O层：发出I/O调用。

I/O管理软件

希赛

I/O设备管理软件一般分为4个层次，如下图所示。图中①②③分别对应（ ）。

A 设备驱动程序、虚设备管理、与设备无关的系统软件

B 设备驱动程序、与设备无关的系统软件、虚设备管理

C 与设备无关的系统软件、中断处理程序、设备驱动程序

D 与设备无关的系统软件、设备驱动程序、中断处理程序

嵌入式操作系统特点

希赛

1、嵌入式操作系统特点：

（1）微型化、（2）代码质量高、（3）专业化、（4）实时性强、（5）可裁减、可配置。

针对不同的硬件平台，操作系统通常建立在一个硬件抽象层（HAL）上，该层位于底层硬件和内核之间，为内核提供各种方便移植的宏定义接口，在不同的平台间移植时，只需要修改宏定义即可。

与硬件相关，与操作系统相关。

嵌入式操作系统

希赛

以下描述中，（ ）不是嵌入式操作系统的特点。

A 面向应用，可以进行裁剪和移植

B 用于特定领域，不需要支持多任务

C 可靠性高，无需人工干预独立运行，并处理各类事件和故障

D 要求编码体积小，能够在嵌入式系统的有效存储空间内运行

实时操作系统-实时性能指标

希赛

- 任务切换时间
- 中断处理相关的时间指标
  - 中断延迟时间
  - 中断响应时间
- 系统响应时间（对用户的输入或请求作出反应的时间）
- 信号量混洗时间（指从一个任务释放信号量到另一个等待该信号量的任务被激活的时间延迟）

实时操作系统-多任务调度算法

希赛

实时系统存在多种调度算法。

优先级调度算法：分配一个相对固定的优先顺序，然后调度程序根据优先级的高低排序，按时间顺序进行高优先级任务优先调度。（非抢占式）

抢占式优先级调度算法：是在优先级调度算法基础上，允许高优先级任务抢占低优先级任务而运行。

时间轮转调度算法：调度程序会依次调度每个任务运行一个小的时间片，然后再调度另一个任务。每个任务运行完一个时间片，无论是否结束都会释放CPU让下一个任务运行。（纯粹的时间轮转调度无法满足实时系统的要求，取而代之的是基于优先级的抢占式时间轮转调度）

最晚截止期调度算法：指调度程序按每个任务的最接近其截止期末端的时间进行调度。

最早截止期调度算法：指调度程序按每个任务的截止期时间，选择最早到截止期头端时间的任务进行调度。

在RTOS中，大多数的RTOS调度算法都是抢占式的。

实时操作系统

希赛

常见的嵌入式RTOS（实时操作系统，Real-Time Operating System）VxWorks、 RT-Linux、QNX、pSOS。

比较类型	VxWorks	RT-Linux
工作方式	操作系统与应用程序处于同一存储空间	操作系统与应用程序处于不同存储空间
多任务支持	支持多任务（线程）操作	支持多进程、多线程操作
实时性	实时系统	实时系统
安全性	任务间无隔离保护	支持进程间隔离保护
标准API	支持	支持

实时操作系统

希赛

以下关于RTOS（实时操作系统）的叙述中，不正确的是（ ）。

A RTOS不能针对硬件变化进行结构与功能上的配置及裁剪

B RTOS可以根据应用环境的要求对内核进行裁剪和重配

C RTOS的首要任务是调度一切可利用的资源来完成实时控制任务

D RTOS实质上就是一个计算机资源管理程序，需要及时响应实时事件和中断

实时操作系统

希赛

以下关于实时操作系统（RTOS）任务调度器的叙述中，正确的是（ ）。

A 任务之间的公平性是最重要的调度目标

B 大多数RTOS调度算法都是抢占方式（可剥夺方式）

C RTOS调度器都采用了基于时间片轮转的调度算法

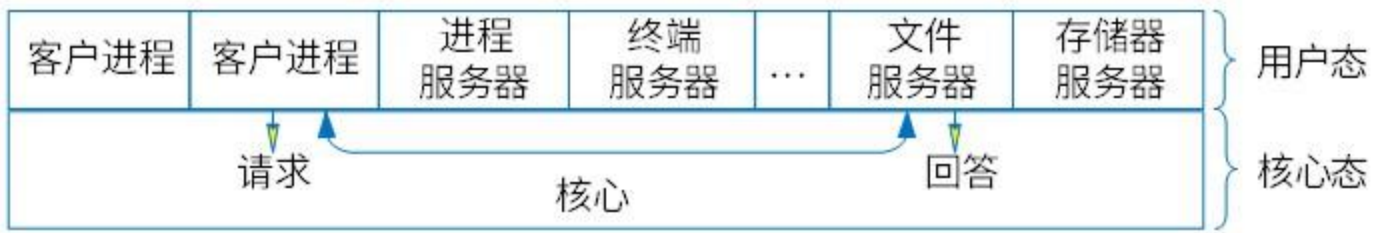
D 大多数RTOS调度算法只采用一种静态优先级调度算法

微内核操作系统

希赛

现代操作系统大多拥有两种工作状态，分别是核心态和用户态。一般应用程序工作在用户态，而内核模块和最基本的操作系统核心工作在核心态。

将传统的操作系统代码放置到更高层，从操作系统中去掉尽可能多的东西，而只留下最小的核心，称之为微内核。（C/S结构）



操作系统的内核服务：异常和中断、计时器、I/O管理等。

微内核操作系统

希赛

	实质	优点	缺点
单体内核	将图形、设备驱动及文件系统等功能全部在内核中实现，运行在内核状态和同一地址空间。	减少进程间通信和状态切换的系统开销，获得较高的运行效率。	内核庞大，占用资源较多且不易剪裁。 系统的稳定性和安全性不好。
微内核	只实现基本功能，将图形系统、文件系统、设备驱动及通信功能放在内核之外。	内核精练，便于剪裁和移植。 系统服务程序运行在用户地址空间，系统的可靠性、稳定性和安全性较高。 可用于分布式系统	用户状态和内核状态需要频繁切换，从而导致系统效率不如单体内核。

微内核操作系统

希赛

采用微内核结构的操作系统提高了系统的灵活性和可扩展性，（ ）。

A 并增强了系统的可靠性和可移植性，可运行于分布式系统中

B 并增强了系统的可靠性和可移植性，但不适用于分布式系统

C 但降低了系统的可靠性和可移植性，可运行于分布式系统中

D 但降低了系统的可靠性和可移植性，不适用于分布式系统