



操作系统 · 系统课

17小时精通

课程讲义



江苏博事达律师事务所

J I A N G S U B O O M S T A R L A W O F F I C E

中国 南京 奥体大街 68 号国际研发总部园 4A 栋 17 楼 邮编: 210019
17F 4ABuilding NO. 68 Aoti Street, Nanjing, China P.C: 210000
电话(Tel): (86)-25-82226685 传真(Fax): (86)-25-82226696

律师声明

江苏博事达律师事务所接受蜂考品牌的委托,发表以下律师声明:

“蜂考系列课程”(含视频、讲义、音频等)内容均为蜂考原创,蜂考品牌公司对此依法享有著作权,任何单位或个人不得侵犯。

蜂考品牌公司为维护创作作品的合法权益,已与江苏博事达律师事务所开展长期法律顾问合作,凡侵犯课程版权等知识产权的,蜂考品牌公司将授权江苏博事达律师事务所依据国家法律法规提起民事诉讼。对严重的侵权盗版分子将报送公安部门采取刑事手段予以严厉打击。

感谢大家对蜂考品牌的长期支持,愿与各位携手共同维护知识产权保护。遵守国家法律法规,从自身做起,抵制盗版!

特此声明!



课时一 操作系统引论

考点	重要程度	占分	题型
1. 操作系统的概念及特征	必考	2~3	选择、填空
2. 操作系统的功能和接口	★	1~2	选择、填空
3. 操作系统的发展过程	★★	2~5	选择、简答
4. 操作系统的运行环境	必考	2~5	选择

1. 操作系统的概念及特征

1) 操作系统的概念

操作系统是指控制和管理整个计算机系统的硬件和软件资源，并且合理地组织调度计算机的工作和资源的分配，提供给用户和其他软件比较方便的接口和环境，是计算机系统中最基本的系统软件。

2) 操作系统的基本特征：

操作系统的基本特征包括：并发、共享、虚拟、异步。

①并发：指两个或者多个事件在同一时间间隔内发生。宏观上看是同时发生的，微观是交替发生的。

并行：是指系统具有同时进行运算或操作的特性，在同一时刻能完成两种或两种以上的工作。

注意：

单核CPU同一时刻只能执行一个程序，所以各个程序只能并发的执行。

多核CPU中多个程序可以并行的执行。

②共享：系统中的资源可供内存中多个并发执行的进程共同使用。

可分为以下两种资源共享方式：

互斥共享方式：资源在特定的一段时间内只允许一个进程访问该资源。

同时共享方式：一个时间段内允许多个进程同时对某些资源进行访问。

并发和共享是操作系统两个最基本的特征，两者之间互为存在条件：①如果失去并发性，即系统中只有一个进程在运行，则共享性会失去意义。②如果失去共享性，则并发进行的进程则无法共享资源。

③虚拟：一个物理意义上的实体变为若干个逻辑上的对应物，物理实体是实际存在的，逻辑上的对应物是用户感受到的。用于实现虚拟的技术，称为虚拟技术。虚拟存储器技术是通过多道程序技术，采用让多道程序并发执行的方法，来分时使用一个处理器的。

④异步：在多道程序环境下，允许多个程序并发执行，但由于资源有限，进程的执行不是一贯到底的，而是走走停停，以不可预知的速度向前推进，这就是进程的异步性。

题 1. 现代OS的基本特征是并发性、_____、_____和异步性。

答案：共享性、虚拟性

解析：现代OS的基本特性是并发性、共享性、虚拟性和异步性，其中最基本的特征是并发性和共享性。其中，需要注意的是：并发性是指若干事件在同一时间间隔内发生，并行性是指若干事件在同一时刻发生。

2. 操作系统的功能和接口

1) 操作系统作为计算机系统资源的管理者

对资源进行管理（重点）：处理机管理、存储器管理、文件管理、设备管理。

①处理机管理

在多道程序环境下，处理机的分配和运行都以进程为基本单位，因而对处理机的管理可归纳为对进程的管理。

进程管理的主要功能包括进程控制、进程同步、进程通信、死锁处理、处理机调度等。

②存储器管理

存储器管理是为了给多道程序的运行提供良好的环境，方便用户使用、提高内存的利用率，主要包括内存分配与回收、地址映射、内存保护与共享和内存扩充等功能。

③文件管理

负责文件管理的部分称为文件系统。文件管理包括文件存储空间的管理、目录管理及文件读写管理和保护等。

④设备管理

设备管理的主要任务是完成用户的*I/O*请求，方便用户使用各种设备，并提高设备的利用率，主要包括缓冲管理、设备分配、设备处理和虚拟设备等功能。

2) 操作系统作为用户与计算机硬件系统之间的接口

操作系统提供了用户接口，主要分为两类：一类是命令接口，另一类是程序接口。

①命令接口

按作业控制方式的不同，命令接口分为联机命令接口和脱机命令接口。

联机命令接口又称为交互式命令接口，适用于分时或实时系统的接口。

脱机命令接口又称为批处理命令接口，适用于批处理系统，它由一组作业控制命令组成。

向上层提供服务（重点）：给软件或者程序员提供程序接口→系统调用

②程序接口

程序接口由一组系统调用组成。用户通过在程序中使用这些系统调用请求操作系统为其提供服务。

当前最为流行的是图形用户界面（*GUI*），即图形接口。

*GUI*最终是通过调用程序接口实现的，用户通过鼠标和键盘在图形界面上单击或使用快捷键，就能很方便地使用操作系统。

3) 操作系统用作扩充机器

没有任何软件支持的计算机称为裸机。

我们通常把覆盖了软件的机器称为扩充机器或虚拟机。

题 1. 下列选项中，OS 提供给应用程序的接口是（ ）

- A. 系统调用
- B. 中断
- C. 库函数
- D. 原语

答案：A

解析：OS 接口的主要有命令接口和程序接口（也称为系统调用）。库函数是高级语言中提供的与系统调用对应的函数（有些与库函数与系统调用无关），目的是隐藏“访管”指令中的细节，使系统调用更方便、抽象。但是，库函数属于用户程序而非系统调用，其实系统调用的上层。

题 2. 操作系统的功能包括：_____、_____、_____、_____、_____。

解析：处理机管理、存储器管理、文件管理、设备管理、用户接口

3. 操作系统的发展过程

1) 单道批处理系统

特点：单路性、独占性、自动性、封闭性、顺序性；

缺点：系统的资源得不到充分的利用。

2) 多道批处理系统

特点：多路性、共享性、自动型、封闭性、无序性、调度性；

好处：提高CPU的利用率，提高内存和I/O设备的利用率，增加系统吞吐量；

缺点：平均周转时间长，无交互能力。

3) 分时系统

分时系统是指在一台主机上连接了多个配有显示器和键盘的终端，由此所组成的系统，该系统允许多个用户同时通过自己的终端，以交互方式使用计算机，共享主机中的资源。

采用“时间片轮转”的处理机调度策略。

4) 实时系统

实时系统是指系统能及时响应外部事件的请求，在规定的时间内完成对该事件的处理，并控制所有实时任务协调一致地运行。

题 1. 与单道程序系统相比，多道程序系统的优点是()

- 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

答案：D

解析：多道程序系统通过组织作业（编码或数据）使CPU总有一个作业可执行，从而提高了CPU利用率、系统吞吐量和I/O设备的利用率 I 、 III 、 IV 是优点。但系统要付出额外的开销来组织作业和切换作业，故 II 错误，因此选D。

题 2. 引入多道程序技术的前提条件之一是系统具有()

- A. 分时功能 B. 中断功能 C. 多CPU技术 D. SPOOLing 技术

答案：B

解析：中断是多道程序技术的基础，因为多个进程之间的切换是通过中断来完成的。

题 3. 分时系统的主要目的是()

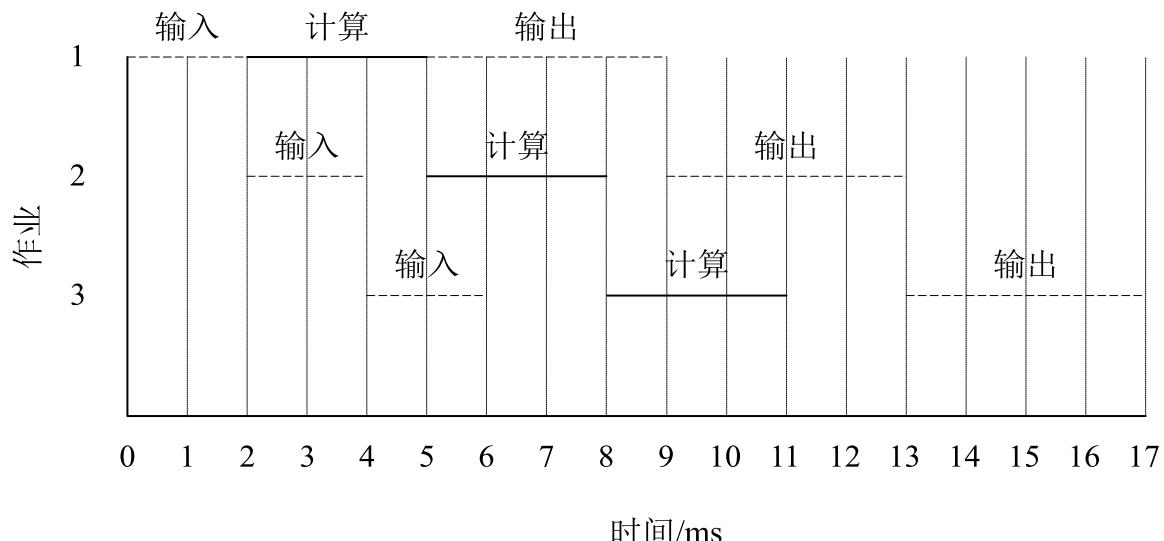
- A. 充分利用 I/O 设备 B. 比较快地响应用户
 C. 提高系统吞吐量 D. 充分利用内存

答案: B

解析: 分时系统能很好地将一台主机供给多个用户同时使用, 实现人机交互和主机共享。对每个用户而言, 他们都希望得到CPU的快速响应, 因此快速响应用户是分时系统追求的主要目标。

题 4. 某单CPU系统中有输入设备和输出设备各1台, 现有3个并发执行的作业, 每个作业的输入、计算和输出时间分别为 $2ms$ 、 $3ms$ 和 $4ms$, 且都按输入、计算和输出的顺序执行, 则执行完这3个作业需要的时间最少是多少?

解析: 因CPU、输入设备和输出设备都只有一个, 因此各操作步骤不能重叠, 作业执行时间关系图如图所示, 即执行完3个作业所需要的时间最少是 $17ms$ 。



作业执行时间关系图

4. 操作系统的运行环境

1) 用户态及内核态

CPU 执行两种不同性质的程序：一是操作系统内核程序，另外一种是应用程序。

操作系统划分为用户态和核心态，严格区分两类程序。

用户自编的程序运行在用户态，操作系统内核程序运行在核心态。

2) 内核

内核是计算机最底层的软件，是计算机功能的延伸。大多包含四个方面的内容：

①时钟管理。时钟就是一种计时器。操作系统需要通过时钟管理向用户提供准确的时间，通过时钟中断的管理，可实现进程的切换。

②中断机制。引入中断机制的初衷是提高多道程序环境中 *CPU* 的利用率。

③原语。原语是底层的一些可被调用的公用小程序，他们各自完成一个规定的操作，是不可划分的单位。

④系统控制其数据结构及处理。

3) 中断（外中断）和异常（内中断）

本质：发生中断就意味着需要操作系统介入开展管理工作。

中断可以使 *CPU* 从用户态切换为核心态，使操作系统获得计算机的控制权。

有了中断，才能实现多道程序并发执行。

“用户态→核心态”是通过中断实现的，并且中断是唯一途径。

“核心态→用户态”的切换是通过执行一个特权指令，将程序状态字(*PSW*)的标志位设置为“用户态”。

4) 中断的分类

① 内中断（异常）

信号的来源：*CPU* 内部与当前执行的指令有关。

a. 资源中断 – 指令中断

如：系统调用时使用的访管指令（又称陷入指令、*trap* 指令）

b. 强迫中断

主要有：硬件故障（如：缺页）或 软件中断（如：整数除0）。

② 外中断

信号的来源：*CPU* 外部与当前执行的指令无关。

a. 外设请求。如：*I/O* 操作完成发出的中断信号。

b. 人工干预。如：用户强制终止一个进程。

题 1. 下列选项中，在用户态执行的是（ ）

- A. 命令解释程序 B. 缺页处理程序
C. 进程调度程序 D. 时钟中断处理程序

答案：A

解析：命令解释程序属于命令接口，用户可以直接调用；缺页处理程序、时钟中断处理程序都属于中断，在核心态执行；进程调度程序在核心态执行。

题 2. 下列选项中，不可能在用户态发生的是（ ）

- A. 系统调用 B. 外部中断
C. 进程切换 D. 缺页

答案：C

解析：系统调用是提供给用户的程序接口，在用户态发生，被调用程序在核心态下执行；外部中断是用户态到核心态的“门”，也发生在用户态，在核心态

完成中断处理程序；进程切换属于系统调用执行过程中的事件，只能发生在核态；缺页发生后，在用户态发生缺页中断，然后进入核心态执行缺页中断服务程序。

题 3. 下列关于系统调用的叙述中，正确的是（ ）

- I . 在执行系统调用服务程序的过程中，*CPU*处于内核态
 - II . 操作系统通过提供系统调用避免用户程序直接访问外设
 - III. 不同的操作系统为应用程序提供了统一的系统调用接口
 - IV. 系统调用是操作系统内核为应用程序提供服务的接口
- A. 仅 I 、 IV B. 仅 II 、 III C. 仅 I 、 II 、 IV D. 仅 I 、 III 、 IV

答案：C

解析：用户可以在用户态调用操作系统的服务，但执行具体的系统调用服务程序是处于内核态的，I 正确；设备管理属于操作系统的职能之一，包括对输入/输出设备的分配、初始化、维护等，用户程序需要通过系统调用使用操作系统的设备管理服务，II 正确；操作系统不同，底层逻辑、实现方式均不相同，为应用程序提供的系统调用接口也不同，III 错误；系统调用是用户在程序中调用操作系统提供的子功能，IV 正确。

课时一 练习题

1. 计算机开机后，操作系统最终被加载到()

- A. BIOS B. ROM C. EPROM D. RAM

2. 在分时操作系统中，时间片一定，()响应时间越长。

- A. 内存越多 B. 内存越少
C. 用户数越多 D. 用户数越少

3. 下列对OS的叙述中，正确的是()

- A. OS都在内核态运行
B. 分时系统中常用的原则是时间片越小越好
C. 批处理系统的主要缺点是缺少交互性
D. DOS是一个单用户多任务的OS

4. OS的基本类型主要有()

- A. 批处理系统、分时系统和多任务系统
B. 批处理系统、分时系统和实时系统
C. 单用户系统、多用户系统和批处理系统
D. 实时系统、分时系统和多用户系统

5. 下列操作系统的各个功能组成部分中，()可不需要硬件的支持

- A. 进程调度 B. 时钟管理
C. 地址映射 D. 中断系统

6、在操作系统中，只能在核心态执行的指令是()

- A. 读时钟
- B. 取数
- C. 广义指令
- D. 寄存器清“0”

7. 下列选项中，必须在核心态下执行的指令是()

- A. 从内存中取数
- B. 将运算结果装入内存
- C. 算术运算
- D. 输入/输出

8. 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()

- 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 、 III

9. 有 A 、 B 两个程序，程序 A 按顺序使用 CPU 10s , 使用设备甲 5s , 使用 CPU 5S , 使用设备乙 5s , 最后使用 CPU 10s 。程序 B 按顺序使用设备甲 10s , 使用 CPU 10s , 使用设备乙 5s , 使用 CPU 5S , 使用设备乙 10s , 试问：

- 1) 在顺序执行程序 A 和程序 B 的情况下， CPU 的利用率是多少？
- 2) 在多道程序环境下， CPU 的利用率区是多少？请画出 A 、 B 程序的执行过程。
- 3) 在多道批处理系统中，是否并发的进程越多，资源利用率越好？为什么？

课时二 进程的描述与控制

考点	重要程度	占分	题型
1. 进程的概念	★★★	3~5	选择、填空
2. 进程的状态与转换	必考	5~10	选择、简答、应用
3. 进程通信	★	1~3	选择、简答
4. 线程的概念	★★	2~3	选择

1. 进程的概念

1) 定义：从不同的角度，进程可以有不同的定义，比较典型的定义有：

①进程是程序的一次执行。

②进程是一个程序及其数据在处理机上顺序执行时所发生的活动。

③进程是具有独立功能的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。

2) 组成：进程是由程序控制块(*PCB*)、程序段、数据段组成。

操作系统是通过*PCB*来管理进程，因此*PCB*中应该包含操作系统对其进行管理所需的各种信息，如进程描述信息、进程控制和管理信息、资源分配清单和处理机相关信息。

程序段：程序代码存放的位置。

数据段：程序运行时使用、产生的运算数据。如全局变量、局部变量、宏定义的常量就存放在数据段内。

3) 组织方式

进程的组织形式分为：链接方式和索引方式。

链接方式：按照进程状态将*PCB*分为多个队列，操作系统持有指向各个队列的指针。

索引方式：根据进程的状态不同，建立几张索引表，操作系统持有指向各个索引表的指针。

4) 进程的特征

动态性：进程的最基本的特征，进程是程序的一次执行过程，是动态的产生、变化和消亡。

并发性：内存中有多个进程实体，各进程可并发执行。

独立性：进程是能独立运行、独立获得资源、独立接受调度的基本单位。

异步性：各个进程按各自独立的、不可预知的速度向前推进，操作系统要提供进程同步机制来解决异步问题。

结构性：每个进程都会配置一个*PCB*。结构上看，进程由程序段、数据段和*PCB*组成。

题 1. 程序运行时独占系统资源，只有程序本身才能改变系统资源状态，这是指（ ）。

- A. 程序顺序执行时的再现性
- B. 并发程序失去再现性
- C. 并发程序失去封闭性
- D. 程序顺序执行时的封闭性

答案：D

解析：程序顺序执行时的封闭性是指程序在封闭的环境下运行，即程序运行时独占计算机系统的全部资源，只有本程序才能改变系统资源状态(除初始状态外)；程序一旦开始执行，其执行结果就不受外界因素影响。

题 2. 进程和程序的本质区别在于（ ）

- A. 前者分时使用CPU，或者独占CPU
- B. 前者存储在内存，后者存储在外存
- C. 前者具有异步性，后者具有可再现性
- D. 前者可以并发执行，后者不能并发执行

答案：D

解析：OS为了提高系统吞吐量和资源利用率，引入了进程来支持并发。程序是不能并发的，程序只有当被创建为进程后，才能并发执行。

题 3. 进程的状态和优先级信息存放在()中。

- A. JCB B. PCB C. 快表 D. 页表

答案：B

解析：PCB是进程存在的唯一标志，它存储进程的状态和优先级等信息。

2. 进程的状态和转换

1) 进程的状态

进程状态：一个进程的生命周期可以划分为一组状态，这些状态刻画了整个进程。进程状态体现一个进程的生命状态。

一般来说，进程有五个状态：就绪状态、运行状态、阻塞状态、创建状态、终止状态。其中前三种状态是进程的基本状态。

创建态：进程在创建时需要申请一个空白PCB，向其中填写控制和管理进程的信息完成资源分配。如果创建工作无法完成，比如资源无法满足就无法被调度运行，把此时进程所处的状态称为创建状态。

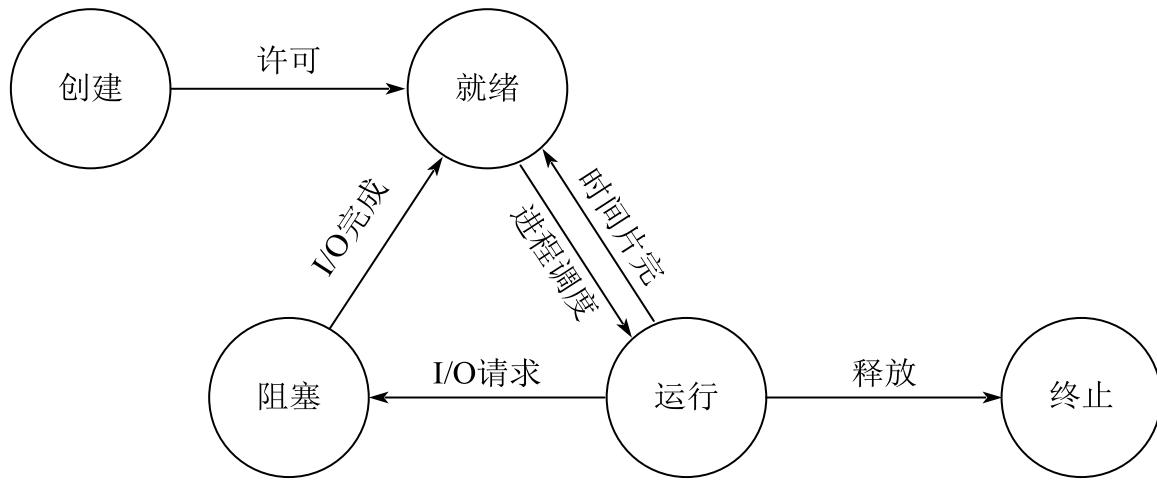
运行态：进程占用CPU，并在CPU上运行的状态。

就绪态：进程已经具备运行条件，但由于未分配CPU无法运行状态。

阻塞态：进程因等待某个事件发生而暂时不能运行的状态。

终止态：进程结束，或出现错误，或被系统终止，进入终止状态。

2) 状态的转换



上述三种基本状态之间转换可分为六种情况：

运行→就绪：①主要是进程占用CPU的时间过长，而系统分配给该进程占用CPU的时间是有限的；②在采用抢先式优先级调度算法的系统中，当有更高优先级的进程要运行时，该进程就被迫让出CPU，该进程便由运行状态转变为就绪状态。

就绪→运行：正运行进程的时间片用完，调度就转到就绪队列中选择合适的进程分配CPU。

运行→阻塞：正在运行的进程因发生某等待事件而无法执行，则进程由运行状态变为阻塞状态，如发生了I/O请求。

阻塞→就绪：进程所等待的事件已经完成，就进入就绪队列。

注意：阻塞态→运行态和就绪态→阻塞态这两种状态转换是不可能发生的。

题 1. 在单处理机系统中，关于进程的叙述，正确的是（ ）

- A. 一个处于就绪状态的进程一旦分配了CPU，即进入运行状态
- B. 只能有一个进程处于就绪状态
- C. 一个进程可以同时处于就绪状态和阻塞状态

D. 最多只有一个进程能处于运行状态

答案: D

解析: 在单处理机系统中, 任何时刻最多只能有一个进程获得CPU而处于运行状态, 但可以有多个进程处于就绪状态; 一个进程不可以同时处于就绪状态和等待状态。

题 2. 已经获得除()以外的运行所需所有资源的进程处于就绪状态。

- A. 存储器 B. 打印机 C. CPU D. 磁盘空间

答案: C

解析: 已经获得除CPU以外的运行所需所有资源的进程处于就绪状态, 其获得CPU后即会进入运行状态。

题 3. 一个进程的读磁盘操作完成后, OS针对该进程必做的是()。

- A. 修改进程状态为就绪状态 B. 降低进程优先级
C. 为进程分配用户内存空间 D. 延长进程的时间片

答案: A

解析: 进程等待读磁盘操作完成后便会从阻塞状态转换为就绪状态。

题 4. 下列选项中, 会导致进程从运行状态变为就绪状态的事件是()。

- A. 执行P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

答案: D

解析: A、B、C都会因为进程请求资源而进入阻塞状态; 进程被高优先级进程抢占, 即被剥夺了处理机资源而进入就绪状态, 其一旦重新获得处理机资源便会继续执行。

3. 进程通信

1) 进程通信是指进程之间的信息交换。

*PV*操作是最低级的通信，高级通信方式是指以较高的效率传输大量数据的通信方式。

高级通信方法主要有三类：

①共享存储。分别有基于共享数据结构的通信方式，例如生产者和消费者、基于共享存储区的通信方式。

②消息传递。是以格式化的消息为单位，利用原语传递消息，分为直接通信和间接通信。

③管道通信。是消息传递的一种特殊方式，是指用于连接一个读进程和一个写进程以实现它们之间的一个共享文件，又称 *pipe* 文件。

题 1. 计算机两个系统中两个协作进程之间不能用来进行进程间通信的时（ ）

- A. 数据库 B. 共享内存 C. 消息传递机制 D. 管道

答案： A

解析： B、C、D 是三种通信方式，数据库不是用于进程间的通信

题 2. 两个合作进程(*Cooperating Processes*)无法利用（ ）交换数据。

- A. 文件系统 B. 共享内存
C. 高级语言程序设计中的全局变量 D. 消息传递系统

答案： C

解析：不同的进程拥有不同的代码段和数据段，全局变量是对同一进程而言的，在不同的进程中是不同的变量，没有任何联系，所以不能用于交换数据。此题也可用排除法做，A、B、D 均是课本上所讲的。

4. 线程

1) 线程的引入

引入线程的目正是为了简化线程间的通信，以小的开销来提高进程内的并发程度。

线程是进程中执行运算的最小单位，是进程中的一个实体，是被系统独立调度和分派的基本单位。线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。

一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。

2) 线程——作为调度和分派的基本单位

进程是系统资源分配的单位，线程是处理器调度的单位。

线程表示进程的一个控制点，可以执行一系列的指令。通常，和应用程序的一个函数相对应。进程分解为线程还可以有效利用多处理器和多核计算机。

3) 线程与进程的比较

①进程与线程的区别：

- a. 调度。线程是调度和分配的基本单位，进程是拥有资源的基本单位。
- b. 并发性。不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行。
- c. 拥有资源。进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源。
- d. 系统开销。在创建或撤消进程时，由于系统都要为之分配和回收资源导致系统的开销明显大于创建或撤消线程时的开销。

②进程和线程的关系：

- a. 一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程。
- b. 资源分配给进程，同一进程的所有线程共享该进程的所有资源。

c. 处理机是分配给线程的，即真正在处理机上运行的是线程。

d. 线程在执行过程中，需要协作同步。不同进程的线程间要利用消息通信的办法实现同步。线程是指进程内的一个执行单元，也是进程内的可调度实体。

题 1. 在以下描述中，() 并不是多线程系统的特长。

A. 利用线程并行地执行矩阵乘法运算

B. Web 服务器利用线程响应 HTTP 请求

C. 键盘驱动程序为每个正在运行的应用配备一个线程，用以响应该应用的键盘输入

D. 基于 GUI 的调试程序用不同的线程分别处理用户输入、计算和跟踪等操作

答案：C

解析：整个系统只有一个键盘，而且键盘输入是人的操作，速度比较慢，完全可以使用一个线程来处理整个系统的键盘输入。

题 2. 下列关于进程和线程的叙述中正确的是()

A. 不管系统是否支持线程，进程都是资源分配的基本单位

B. 线程是资源分配的基本单位，进程是调度的基本单位

C. 系统级线程和用户级线程的切换都需要内核的支持

D. 同一进程中的各个线程拥有各自不同的地址空间

答案：A

解析：在引入线程后，进程依然是资源分配的基本单位，线程是调度的基本单位，同一进程中的各个线程共享进程的地址空间。在用户级线程中，有关线程管理的所有工作都由应用程序完成，无须内核的干预，内核意识不到线程的存在。

课时二 练习题

1. 系统进程所请求的一次 I/O 操作完成后，将使进程状态从()。

- A. 运行态变为就绪态
- B. 运行态变为阻塞态
- C. 就绪态变为运行态
- D. 阻塞态变为就绪态

2. 在任何时刻，一个进程的状态变化()引起另一个进程的状态变化。

- A. 必定
- B. 一定不
- C. 不一定
- D. 不可能

3. 在单处理器系统中，若同时存在10个进程，则处于就绪队列中的进程最多有()个。

- A. 1
- B. 8
- C. 9
- D. 10

4. 以下说法正确的是()

- A. 程序段和相关数据段构成一个完整的进程实体
- B. 引入线程后，进程不再是可独立分配资源的基本单位
- C. 信号量机制既可以实现进程同步，又可以实现进程互斥
- D. 同一进程中的所有线程可共享彼此所拥有的资源和变量

5. 在引入线程的OS中，把()作为调度和分配的基本单位，而把()作为拥有资源的基本单位。

- A. 进程、线程
- B. 程序、线程
- C. 程序、进程
- D. 线程、进程

6. 一个进程可以包含多个线程，各线程()。

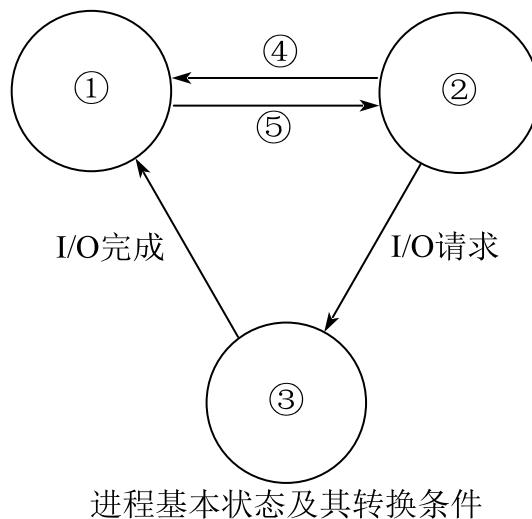
- A. 共享进程的虚拟地址空间
- B. 地址空间完全独立

- C. 是资源分配的单位 D. 共享堆栈

7. 下面的叙述中，正确的是（ ）

- A. 在一个进程中创建一个新线程比创建一个新进程所需的工作量多
- B. 同一进程中的线程间通信和不同进程中的线程间通信差不多
- C. 同一进程中的线程间切换由于有许多上下文相同而可以被简化
- D. 同一进程中的线程间通信须调用内核

8. 分析图中给出的不完整的进程基本状态及其转换条件，在5个空格内填上对应的状态名称或者相应的转换条件。



9. 在一个单处理机的多道程序设计系统中，若在某一时刻有 N 个进程同时存在，那么处于运行状态、阻塞状态和就绪状态的进程个数的最小值和最大值分别可能是多少？

10. 画出下面4条语句所对应的前驱图。

$$P_1: a = x + 2y \quad P_2: b = a + 6 \quad P_3: c = 4a - 9 \quad P_4: d = 2b + 5c$$

课时三 处理机调度（一）

考点	重要程度	占分	题型
1. 处理机调度的基本概念	★	2~3	选择、填空
2. 调度算法-先来先服务算法	必考	3~5	选择、应用
3. 调度算法-短作业优先算法	必考	3~5	选择、应用
4. 调度算法-优先级算法	必考	3~5	选择、应用

1. 处理机调度的基本概念

1) 基本准则

不同的调度算法具有不同的特性，在选择调度算法时，必须考虑算法的特性。为了比较处理机调度算法的性能，人们提出了很多评价准则，下面介绍其中主要的几种：

① *CPU* 利用率。

CPU 是计算机系统中最重要和昂贵的资源之一，所以应尽可能使 *CPU* 保持“忙”的状态，使这一资源利用率最高。

② 系统吞吐量。表示单位时间内 *CPU* 完成作业的数量。

长作业需要消耗较长的处理机时间，因此会降低系统的吞吐量。而对于短作业，它们所需要消耗的处理机时间较短，因此能提高系统的吞吐量。

调度算法和方式的不同，也会对系统的吞吐量产生较大的影响。

③ 周转时间。

周转时间是指从作业提交到作业完成所经历的时间，是作业等待、在就绪队列中排队、在处理机上运行及进行输入/输出操作所花费时间的总和。

a. 作业的周转时间可用公式表示如下：

$$\text{周转时间} = \text{作业完成时间} - \text{作业提交时间}$$

b. 平均周转时间是指多个作业周转时间的平均值：

$$\text{平均周转时间} = (\text{作业1的周转时间} + \dots + \text{作业n的周转时间}) / n$$

c. 带权周转时间是指作业周转时间与作业实际运行时间的比值：

$$\text{带权周转时间} = \text{作业周转时间} / \text{作业实际运行时间}$$

d. 平均带权周转时间是指多个作业带权周转时间的平均值：

$$\text{平均带权周转时间} = (\text{作业1的带权周转时间} + \dots + \text{作业n的带权周转时间}) / n$$

④等待时间

等待时间指进程处于等处理机状态的时间之和，等待时间越长，用户满意度越低。

⑤响应时间。

响应时间指从用户提交请求到系统首次产生响应所用的时间，在交互式系统中，周转时间不可能是最好的评价准则，一般采用响应时间作为衡量调度算法的重要准则之一。

2) 三个层次

一个作业从提交开始直到完成，往往要经历以下三级调度：

①作业调度，又称高级调度，其主要任务是按一定的原则从外存处于后备状态的作业中挑选一个(或多个)作业，给它(们)分配内存、输入/输出设备等必要的资源，并建立相应的进程，以使它(们)获得竞争处理机的权利。

②中级调度，又称内存调度，其作用是提高内存利用率和系统吞吐量。为此，应将那些暂时不能运行的进程调至外存等待，把此时的进程状态称为挂起态。当它们已具备运行条件且内存又稍有空闲时，由中级调度来决定把外存上的那些已具备运行条件的就绪进程再重新调入内存，并修改其状态为就绪态，挂在就绪队列上等待。

③进程调度，又称低级调度，其主要任务是按照某种方法和策略从就绪队列中选取一个进程，将处理机分配给它，进程调度是操作系统中最基本的一种调度，在一般的操作系统中都必须配置进程调度。进程调度的频率很高，一般几十毫秒一次。

3) 进程调度方式

所谓进程调度方式，是指当某个进程正在处理机上执行时，若有某个更为重要或紧迫的进程需要处理，即优先权更高的进程进入就绪队列，此时应如何分配处理机呢？通常有以下两种进程调度方式：

①非剥夺调度方式，又称非抢占方式。非剥夺调度方式是指当一个进程正在处理机上执行时，即使有某个更为重要或紧迫的进程进入就绪队列，仍然让正在执行的进程继续执行，直到该进程完成或发生某种事件而进入阻塞态时，才把处理机分配给更为重要或紧迫的进程。

②剥夺调度方式，又称抢占方式。采用剥夺式的调度，对提高系统吞吐率和响应效率都有明显的好处。但“剥夺”不是一种任意性行为，必须遵循一定的原则，主要有优先权、短进程优先和时间片原则等。

题 1. 若某单处理机多进程系统中有多个处于就绪状态的进程，则下列关于处理机调度的叙述中，错误的是（ ）。

- A. 在进程结束时能进行处理机调度
- B. 创建新进程后能进行处理机调度
- C. 在进程处于临界区时不能进行处理机调度
- D. 在系统调用完成并返回用户态时能进行处理机调度

答案：C

解析：选项A、B、D显然属于可以进行处理机调度的情况。对于选项C，当进程处于临界区时，说明进程正在占用处理机，只要不破坏临界资源的使用规则，就不会影响处理机的调度。比如，通常访问的临界资源可能是慢速的外设（如打印机），若进程在访问打印机时不能进行处理机调度，则系统的性能将会变得非常差。

题 2. 下面的情况中，进程调度可能发生的时机有（ ）。

- I . 正在执行的进程时间片用完
 - II . 正在执行的进程提出 I/O 请求后进入等待状态
 - III. 有新的用户登录系统
 - IV. 等待硬盘读数据的进程获得了所需的数据
- A. I B. I 、 II 、 III 、 IV
- C. I 、 II 、 IV D. I 、 III 、 IV

答案： B

解析：正在执行的进程在时间片用完后进入就绪状态，系统会调入一个新的进程来为其分配处理机并执行。正在执行的进程提出 I/O 请求后进入等待状态，系统同样会调入一个新的进程为其分配处理机并执行。有新的用户登录进入系统会创建新的进程，若处理机空闲，则分配处理机并执行。等待硬盘读数据的进程获得了所需的数据后，若处理机空闲，则可进行进程调度。

2. 调度算法

1) FCFS 算法

① 算法规则。先来先服务算法 (*first come first server , FCFS*)

按照作业/进程到达的先后顺序来进行调度。

② 适用情况。可用于作业调度也可用于进程调度。

③ 优缺点。 优点：算法实现简单。

缺点：对长作业有利，对短作业不利。

题 1. 假设 4 个作业到达系统的时刻和运行时间如表 1 所示。

表一 进程（作业）到达时刻和运行时间表

作业	到达时刻 t	运行时间
J_1	0	3
J_2	1	3
J_3	1	2
J_4	3	1

系统在 $t = 2$ 时开始调度作业，若采用 FCFS 算法则选中的作业是（ ）。

- A. J_1 B. J_2 C. J_3 D. J_4

答案： A

解析： FCFS 是作业来的越早优先级越高，因此选中 J_1 。

2) SJF 算法

① 算法规则。短作业优先调度算法 (*short job first*, SJF)

以作业的长短来计算优先级，作业越短，其优先级越高。

② 适用情况。可用于作业调度及进程调度。

③ 优缺点

优点：“最短的”平均等待时间及平均周转时间。

缺点： a. 必须先知道作业的运行时间。

 b. 对长作业不利，会出现饥饿现象。

 c. 没有考虑作业的紧迫程度。

题 1. 假设 4 个作业到达系统的时刻和运行时间如表 1 所示。

表一 进程（作业）到达时刻和运行时间表

作业	到达时刻 t	运行时间
J_1	0	3
J_2	1	3
J_3	1	2
J_4	3	1

系统在 $t = 2$ 时开始调度作业，若采用 SJF 算法则选中的作业是（ ）。

- A. J_1 B. J_2 C. J_3 D. J_4

答案：C

解析：SJF 是作业运行时间越短，优先级越高，因此选中 J_3 。

题 2. 对于相同的进程序列，下列进程调度算法中平均周转时间最短的是（ ）。

- A. FCFS 算法 B. SJF 算法 C. 优先级调度算法 D. RR 调度算法

答案：B

3) 优先级算法 (*priority-scheduling algorithm*)

① 算法规则。基于进程（作业）的紧迫程度，由外部赋予进程相应的优先级，根据优先级进行调度。

② 适用情况。可用于作业调度也可用于进程调度甚至 I/O 调度。

① 类型

抢占式优先级调度算法：

只需出现另一个优先级更高的进程，调度就会发生变化。

非抢占式优先级调度算法：主动放弃。

④优先级的类型

- a. 静态优先级：在创建进程时确定，其在进程的整个运行期间不变。
- b. 动态优先级：在创建进程之初，先赋予进程一个优先级，然后动态的调整优先级。

⑤优缺点

优点：用优先级区分紧急程度，运用于实时OS。

缺点：可能导致饥饿（低优先级进程的饥饿）。

题 1. ()优先级是创建进程时确定的，确定之后在进程的整个运行期间不再改变。

- A. 动态 B. FCFS C. 短作业 D. 静态

答案：D

解析：考察静态优先级的定义。

题 2. 有5个批处理作业A、B、C、D、E几乎同时到达，它们的估计运行时间为
分别为2min、4min、6min、8min、10min，优先级分别为1、2、3、4、5其中1为最低优先级。分别用下面的调度算法来计算上述作业的平均周转时间。

- 1) 优先级调度算法
- 2) FCFS 调度算法；作业到达顺序为CDBEA。
- 3) SJF 调度算法

解：1) 使用优先级调度算法时，调度顺序为E、D、C、B、A，各作业的周转时间如下：

作业	执行时间	优先级	开始运行时间	完成时间	周转时间
A	2	1	28	30	30
B	4	2	24	28	28
C	6	3	18	24	24

蜂考

<i>D</i>	8	4	10	18	18
<i>E</i>	10	5	0	10	10

平均周转时间: $(30 + 28 + 24 + 18 + 10)/5 = 22\text{min}$

2) 使用*FCFS*调度算法时, 调度顺序为*C*、*D*、*B*、*E*、*A*, 各作业的周转时间如下:

作业	执行时间	优先级	开始运行时间	完成时间	周转时间
<i>A</i>	2	1	28	30	30
<i>B</i>	4	2	14	18	18
<i>C</i>	6	3	0	6	6
<i>D</i>	8	4	6	14	14
<i>E</i>	10	5	18	28	28

平均周转时间: $(30 + 18 + 6 + 14 + 28)/5 = 19.2\text{min}$

3) 使用*SJF*调度算法时, 调度顺序为*A*、*B*、*C*、*D*、*E*, 各作业的周转时间如下:

作业	执行时间	优先级	开始运行时间	完成时间	周转时间
<i>A</i>	2	1	0	2	2
<i>B</i>	4	2	2	6	6
<i>C</i>	6	3	6	12	12
<i>D</i>	8	4	12	20	20
<i>E</i>	10	5	20	30	30

平均周转时间: $(2 + 6 + 12 + 20 + 30)/5 = 14\text{min}$

课时三 练习题

1. 时间片轮转调度算法是为了()。
A. 多个用户能及时干预系统 *B.* 使系统变得高效
C. 优先级较高的进程得到及时响应 *D.* 需要CPU时间最少的进程

2. 在单处理器的多进程系统中，进程什么时候占用处理器及决定占用时间的长短是由()决定的。
A. 进程相应的代码长度 *B.* 进程总共需要运行的时间
C. 进程特点和进程调度策略 *D.* 进程完成什么功能

3. ()有利于CPU繁忙型的作业，而不利于I/O繁忙型的作业。
A. 时间片轮转调度算法 *B.* 先来先服务调度算法
C. 短作业(进程)优先算法 *D.* 优先权调度算法

4. 下面有关选择进程调度算法的准则中，不正确的是().
A. 尽快响应交互式用户的请求
B. 尽量提高处理机利用率
C. 尽可能提高系统吞吐量
D. 适当增长进程就绪队列的等待时间

5. 设有4个作业同时到达，每个作业的执行时间为2h，它们在一台处理器上按单道式运行，则平均周转时间为()。
A. 1h *B. 5h* *C. 2.5h* *D. 8h*

6. 设某计算机系统有一个CPU、一台输入设备、一台打印机。现有两个进程同时进入就绪态，且进程A先得到CPU运行，进程B后运行，进程A的运行轨迹为：计算50ms，打印信息100ms，再计算50ms，打印信息100ms，结束，进程B的运行轨迹为：计算50ms，输入数据80ms，再计算100ms，结束，试画出它们的甘特图，并说明：

- 1) 开始运行后，CPU有无空闲等待？若有，在哪段时间内等待？计算CPU的利用率。
- 2) 进程A运行时有无等待现象？若有，在什么时候发生等待现象？

解析：这类实际的CPU和输入/输出设备调度的题目一定要画图，画出运行时的甘特图后就能清楚地看到不同进程间的时序关系，如下图所示。

	0	50	100	150	200	300
CPU	A	B	空闲	A	B	
输入设备	空闲		B	空闲		
打印机	空闲	A		空闲	A	

7. 假设某计算机系统有4个进程，各进程的预计运行时间和到达就绪队列的时刻见下表（相对时间，单位为“时间配额”），试用可抢占式短进程优先调度算法和时间片轮转调度算法进行调度（时间配额为2），分别计算各个进程的调度次序及平均周转时间。

进程	到达就绪队列时刻	预计运行时间
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	2

课时四 处理机调度（二）

考点	重要程度	占分	题型
1. 时间片轮转算法	★★★★	3~5	选择、应用
2. 高相应比优先算法	必考	3~5	选择、应用
3. 多级反馈队列调度算法	★★★	2~4	选择、应用

1. 调度算法

1) 时间片轮转算法(*RR*)

①算法思想：时间片轮转算法(*Round – Robin*)

公平地、轮流地为各个进程服务，让每个进程在一定时间间隔内都可以得到响应。

②算法规则：按照各进程到达就绪队列的顺序，轮流让各个进程执行一个时间片。

若进程未到一个时间片内执行完，则剥夺处理机，将进程重新放到就绪队列队尾重新排队。

③适用情况：可用于进程调度。

④是否抢占？若进程未能在时间片内运行完，将被强行剥夺处理机使用权，因此时间片轮转调度算法属于抢占式算法。由时钟装置发出时钟中断来通知CPU时间片已到。

⑤优缺点

优点：公平；响应快，适用于分时操作系统；。

缺点：不能区分任务的紧急程度，需要进程切换，消耗较大。

题 1. 假定要在一台处理机上执行表所示的作业且假定这些作业在时刻0以1, 2, 3, 4, 5的顺序到达。请说明分别采用*FCFS*、*RR* (时间片为1)、*SJF*及非抢占式优先级调度算法时，这些作业的执行情况(优先级的高低顺序依次为1到5)。针对上述每种调度算法，给出平均周转时间和平均带权周转时间。

作业执行时间表

作业	执行时间	优先级
1	10	3
2	1	1
3	2	3
4	1	4
5	5	2

解析：

1) 作业执行顺序如图

FCFS :

1	2	3	4	5

RR :

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

SJF :

2	4	3	5	1

非抢占式优先级:

2	5	1	3	4

作业执行程序图

2) 各个作业对应于各个算法的周转时间和带权周转时间见下表

算法	时间类型	P_1	P_2	P_3	P_4	P_5	平均时间
		运行时间	10	1	2	1	3.8
<i>FCFS</i>	周转时间	10	11	13	14	19	13.4
	带权周转时间	1	11	6.5	14	3.8	7.26
<i>RR</i>	周转时间	19	2	7	4	14	9.2
	带权周转时间	1.9	2	3.5	4	2.8	2.84
<i>SJF</i>	周转时间	19	1	4	2	9	7
	带权周转时间	1.9	1	2	2	1.8	1.74
非抢占式优先级	周转时间	16	1	18	19	6	12
	带权周转时间	1.6	1	9	19	1.2	6.36

2. 高响应比优先算法(HRRN)

1) 算法思想：高响应比优先调度算法 (*Highest Response Ratio Next, HRRN*)

综合考虑作业或进程的等待时间和要求服务的时间。

2) 算法规则：在每次调度前先计算各个作业或进程的响应比（优先级），选择响应比最高的作业或进程为其服务。

响应比 (R_P) = (等待时间 + 要求服务时间) / 要求服务时间

$$= \text{响应时间} / \text{要求服务时间}$$

3) 适用情况：可用于作业调度及进程调度。

4) 优缺点

优点：综合考虑了等待时间和运行时间，较好的实现了折中。

缺点：每次调度前都要计算响应比，会增加系统的开销。

注意：不会导致饥饿现象。

题 1. 请回答下列问题：

1) 处理机的调度有哪3个层次？

2) 假设某OS以单道批处理方式运行，现有4道作业，它们进入系统的时间及运行时间如表所示，试采用高响应比优先调度算法进行调度，请问这组作业的运行顺序、平均周转时间和平均带权周转时间分别是多少？并给出计算过程。

作业时间运行表

作业号	进入系统时间	运行时间(小时)
1	7:00	2.00
2	7:50	0.50
3	8:00	0.10
4	8:50	0.20

解析：

1) 处理机的调度分为高级调度、中级调度和低级调度。①高级调度：又称为长

程调度或作业调度，主要任务是根据某种算法从外存上在处于后备队列的作业中选取一个或多个作业调入内存，给它们创建进程、分配必要资源，并将它们放入就绪队列。②中级调度：又称为中程调度或内存调度，主要任务是根据某种算法将处于外存对换区中的具备运行条件的进程调入内存，或将内存中暂时不能运行的进程调至外存对换区等待。③低级调度：又称为短程调度或进程调度，主要任务是根据某种算法从就绪队列中选取一个进程，并将处理机分配给它。进程调度是OS中最基本的调度方式，在OS中必须进行配置。

2) 4道作业并非同时到达，7:00只有作业1到达，故首先运行的是作业1，其在9:00运行完毕，此时其余3道作业已全部抵达，根据高响应比优先调度算法的计算公式计算出这3道作业的响应比，响应比高的作业先运行，故运行作业3，其在9:06运行完毕，此时再计算其余两道作业的响应比，可知应先运行作业2，后运行作业4。整个作业的运行时间情况如下表所示。

作业运行时间表：

作业号	进入系统时间	运行时间(小时)	开始运行时间	完成时间	周转时间(小时)	带权周转时间
1	7:00	2.00	7:00	9:00	2.00	1.00
3	8:00	0.10	9:00	9:06	1.10	11.00
2	7:50	0.50	9:06	9:36	1.77	3.54
4	8:50	0.20	9:36	9:48	0.97	4.85

3. 多级反馈队列调度算法

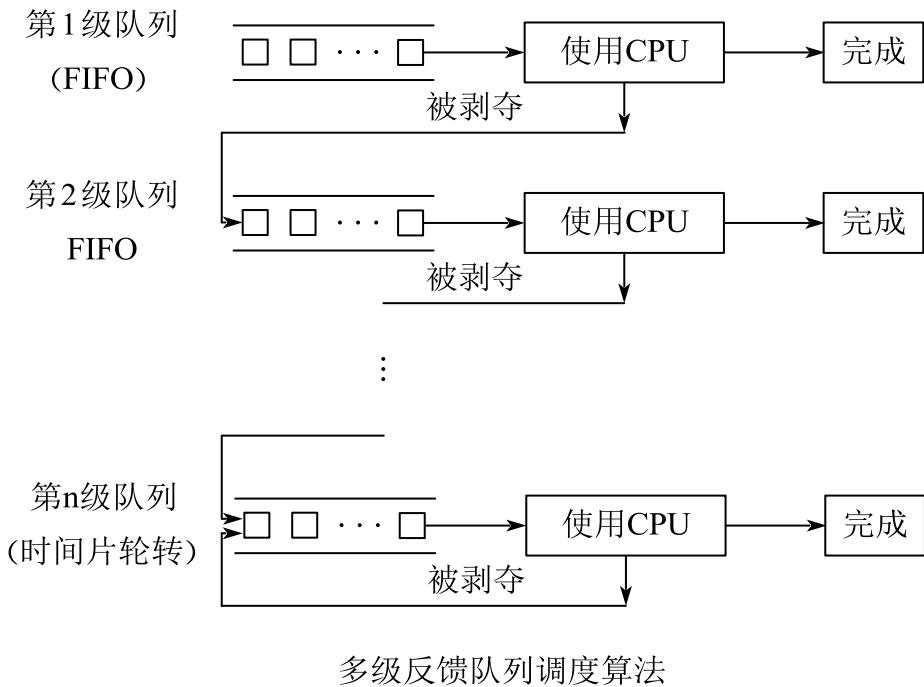
1) 算法思想： 对其他调度算法的折中权衡。

2) 算法规则：

①设置多个就绪队列。各级队列优先级从高到低，时间片从小到大。

②每个队列都采用FCFS调度算法。

③按队列优先级调度。只有当第 $1 \sim i-1$ 队列均空时，才会调度第*i*队列中的进



3) 适用情况：可用于进程调度

4) 类型：属于抢占式的算法。

5) 优缺点：

优点：用优先级区分紧急程度，运用于实时OS。

缺点：可能导致饥饿（低优先级进程的饥饿）。

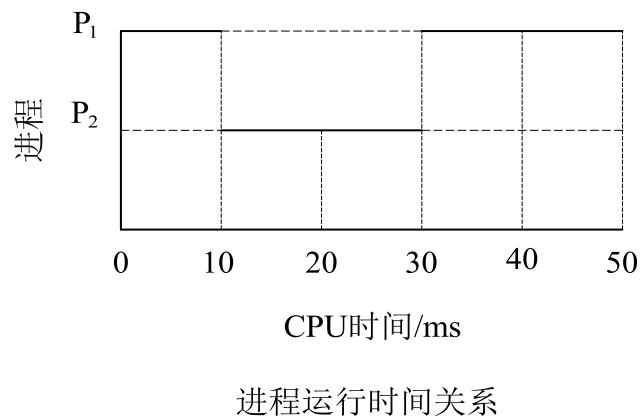
题 1. 系统采用两级反馈队列调度算法进行进程调度就绪队列 Q_1 采用 RR 调度算法，时间片为 $10ms$ ；就绪队列 Q_2 采用短进程优先调度算法。系统优先调度 Q_1 队列中的进程，当 Q_1 为空时，系统才会调度 Q_2 中的进程；新创建的进程首先进入 Q_1 ； Q_1 中的进程执行一个时间片后若未结束，则转入 Q_2 。若当前 Q_1 和 Q_2 为空，系统依次创建进程 P_1 、 P_2 后即开始调度进程， P_1 、 P_2 需要的 CPU 时间分别为 $30ms$ 和 $20ms$ ，则进程 P_1 、 P_2 在系统中的平均等待时间为（ ）。

- A. $25ms$ B. $20ms$ C. $15ms$ D. $10ms$

答案：C

解析：进程 P_1 、 P_2 依次创建后进入队列 Q_1 ，根据 RR 调度算法的规则， P_1 、 P_2 将依次被分配 $10ms$ 的 CPU 时间，分别执行完一个时间片后都会转入队列 Q_2 ；就绪队列 Q_2 采用短进程优先调度算法，此时 P_1 还需要 $20ms$ 的 CPU 时间， P_2 还需要 $10ms$ 的 CPU 时间，因此 P_2 会被优先调度执行， $10ms$ 后 P_2 执行完成， P_1 再调度执行， P_1 执行 $20ms$ 后完成。进程运行时间关系如图所示。

进程 P_1 、 P_2 的等待时间为图中的虚横线部分，平均等待时间 = (P_1 等待时间 + P_2 等待时间) / 2 = 15ms



题 2. 下列与进程调度有关的因素中，在设计多级反馈队列调度算法时要考虑的是（ ）。

- I . 就绪队列的数量
 - II . 就绪队列的优先级
 - III . 各就绪队列所采用的调度算法
 - V . 进程在就绪队列间的迁移条件
- A. I 、 II B. III、 V C. II 、 III D. I 、 II 、 III

答案：D

课时四 练习题

1. 采用时间片轮转调度算法分配CPU时，当处于运行态的进程用完一个时间片后，它的状态是()状态。

- A. 阻塞
- B. 运行
- C. 就绪
- D. 消亡

2. 一个作业8:00到达系统，估计运行时间为1h。若10:00开始执行该作业，其响应比是()

- A. 2
- B. 1
- C. 3
- D. 0.5

答案：C

3. 关于优先权大小的论述中，正确的是()

- A. 计算型作业的优先权，应高于I/O型作业优先权
- B. 用户进程的优先权，应高于系统进程的优先权
- C. 在动态优先权中，随着作业等待时间的增加，其优先权将随之下降
- D. 在动态优先权中，随着进程执行时间的增加，其优先权降低

4. 下列调度算法中，()调度算法是绝对可抢占的。

- A. 先来先服务
- B. 时间片轮转
- C. 优先级
- D. 短进程优先

5. 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()

- 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

6. 下列有关基于时间片的进程调度的叙述中，错误的是（ ）

- A. 时间片越短，进程切换的次数越多，系统开销越大
- 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

7. 分时操作系统通常采用（ ）调度算法来为用户服务

- A. 时间片轮转 B. 先来先服务
- C. 短作业优先 D. 优先级

8. 在进程调度算法中，对短进程不利的是（ ）

- A. 短进程优先调度算法 B. 先来先服务调度算法
- C. 高响应比优先调度算法 D. 多级反馈队列调度算法

9. 假设系统中所有进程同时到达，则使进程平均周转时间最短的是（ ）调度算法。

- A. 先来先服务 B. 短进程优先
- C. 时间片轮转 D. 优先级

10. 某系统采用基于优先权的非抢占式进程调度策略，完成一次进程调度和进程切换的系统时间开销为 $1\mu s$ 。在 T 时刻就绪队列中有3个进程 P_1 , P_2 和 P_3 ，其在就绪队列中的等待时间、需要的CPU时间和优先权如下表所示：

进程	等待时间	需要的CPU时间	优先权
P_1	$30\mu s$	$12\mu s$	10
P_2	$15\mu s$	$24\mu s$	30

P_3	$18\mu s$	$36\mu s$	20
-------	-----------	-----------	----

若优先权值大的进程优先获得CPU，从T时刻起系统开始进程调度，则系统的平均周转时间为()。

- A. $54\mu s$ B. $73\mu s$ C. $74\mu s$ D. $75\mu s$

课时五 死锁（一）

考点	重要程度	占分	题型
1. 死锁的概念	必考 ★★	5~10	选择、简答、应用
2. 死锁预防			

1. 死锁的概念

1) 死锁的定义

各进程互相等待对方手里的资源，导致各进程都阻塞，无法向前推进的现象。

注意区分：死锁和饥饿

	共同点	区别
死锁	都是进程无法顺利向前推进导致的现象	死锁一定是“循环等待对方手里的资源”导致的，因此如果有死锁现象，那至少有两个或两个以上的进程同时发生死锁。另外，发生死锁的进程一定处于阻塞
饥饿		可能只有一个进程发生饥饿。发生饥饿的进程既可能是阻塞态（长期得不到的I/O设备），也可能是就绪态（长期得不到处理机）

2) 产生死锁的必要条件

产生死锁必须同时满足以下四个条件，只要其中任一条件不成立，死锁就不会发生。

①互斥条件。只有对必须互斥使用的资源的争抢才会导致死锁（如哲学家的筷子、打印机设备）。

②请求和保持条件。进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源又被其他进程占有，此时请求进程被阻塞，但又对自己已有的资源保持不放。

③循环等待条件。存在一种进程资源的循环等待链，链中的每一个进程已获得的资源同时被下一个进程所请求。

注意：发生死锁时一定有循环等待，但是发生循环等待时未必死锁。

④不剥夺条件。进程所获得的资源在未使用完之前，不能由其他进程强行夺走，只能主动释放。

2. 死锁的预防

死锁的处理就是不允许死锁的发生，分为静态策略（预防死锁）和动态策略（避免死锁）。

- 1) 预防死锁。破坏死锁产生的四个必要条件中的一个或几个。
- 2) 避免死锁。用某种方法防止系统进入不安全状态，从而避免死锁（银行家算法）
- 3) 死锁的检测和解除。允许死锁的发生，不过操作系统会负责检测出死锁的发生，然后采取某种措施解除死锁。

①破坏互斥条件

如果把只能互斥使用的资源改造为允许共享使用，则系统不会进入死锁状态。比如：*SPOOLing* 技术，操作系统可以采用*SPOOLing* 技术把独占设备在逻辑上改造成共享设备。

缺点：并不是所有的资源都可以改造成可共享使用的资源，并且为了系统安全，很多地方还必须保护这种互斥性，因此，很多时候都无法破坏互斥条件。

②破坏不剥夺条件

当某个进程请求新的资源得不到满足时，它必须立即释放保持的所有资源，待以后需要时再重新申请。也就是说，即使某些资源尚未使用完，也需要主动释放，从而破坏了不可剥夺条件。

缺点：

- a. 实现起来比较复杂。

- b. 释放已获得的资源可能造成前一阶段工作的失效。因此这种方法

一般只适用于易保存和恢复状态的资源，如CPU。

- c. 反复地申请和释放资源会增加系统开销，降低系统吞吐量。

- d. 这种破坏意味着只要暂时得不到某个资源，之前获得的那些资源就都需要放弃，以后再重新申请。如果一直发生这样的情况，就会

导致进程饥饿。

③破坏请求和保持条件

可以采用静态分配的方法，即进程在运行前一次申请完它所需要的全部资源，在它的资源未满足前，不让它投入运行。一旦投入运行后，这些资源就一直归它所有，该进程就不会再请求别的任何资源了。

该策略实现起来简单，但也有明显的缺点：有些资源可能只需要用很短的时间，因此如果进程的整个运行期间都一直保持着所有资源，就会造成严重的资源浪费，资源利用率极低。另外，该策略也有可能导致某些进程饥饿。

④破坏循环等待条件

方法：对系统的所有资源类型进行线性排序（顺序资源分配法）。首先给系统中的资源编号，规定每个进程必须按编号递增的顺序请求资源，同类资源（即编号相同的资源）一次申请完。

原理分析：一个进程只有已占有小编号的资源时，才有资格申请更大编号的资源。按此规则，已持有大编号资源的进程不可能逆向回来申请小编号的资源，从而就不会产生循环等待的现象。

该策略的缺点：

a. 为系统中各类资源规定的序号必须相对稳定，这限制了新类型设备的增加。

b. 尽管在为资源的类型分配序号时已经考虑到了大多数作业在实际使用这些资源时的顺序，但也经常会发生作业使用各类资源的顺序与系统规定的顺序不同的情况，造成对资源的浪费。

c. 为了方便用户，系统对用户在编程时所施加的限制条件应尽量少，然而这种按规定次序申请资源的方法必然会限制用户进行简单、自主的编程。

题 1. 系统产生死锁的可能原因是()

- A. 独占资源分配不当 B. 系统资源不足
- C. 进程运行太快 D. CPU 内核太多

答案: A

解析: 系统死锁的可能原因主要是时间上和空间上的。时间上由于进程运行中推进顺序不当, 即调度时机不合适, 不该切换进程时进行了切换, 可能会造成死锁; 空间上的原因是对独占资源分配不当, 互斥资源部分分配又不可剥夺, 极易造成死锁。那么, 为什么系统资源不足不是造成死锁的原因? 系统资源不足只会对进程造成“饥饿”。例如, 某系统只有三台打印机, 若进程运行中要申请四台, 显然不能满足, 该进程会永远等待下去。若该进程在创建时便声明需要四台打印机, 则操作系统立即就会拒绝。这实际上是资源分配不当的一种表现。不能以系统资源不足来描述剩余资源不足的情形。

题 2. 系统中有4个进程都要使用某类资源。若每个进程最多需要3个该类资源, 则为了保证系统不发生死锁, 系统至少应提供()该类资源

- A. 3个 B. 4个 C. 9个 D. 12个

答案: C

解析: 系统中有4个进程, 每个进程最多需要3个资源, 先给每个进程分配2个资源, 共需要8个资源, 此时需要系统中还有1个空闲资源(其可被分配给任一进程), 才不会发生死锁, 故至少需要9个资源。

课时五 练习题

1. 某计算机系统中有8台打印机，由 K 个进程竞争使用它们，每个进程最多需要3台打印机。该系统可能会发生死锁的 K 的最小值是()
A. 2 B. 3 C. 4 D. 5

2. 死锁预防是保证系统不进入死锁状态的静态策略，其解决办法是破坏产生死锁的四个必要条件之一。下列方法中破坏了“循环等待”条件的是()
*A. 银行家算法
B. 一次性分配策略
C. 剥夺资源法
D. 资源有序分配策略*

3. 解除死锁通常不采用的方法是()
*A. 终止一个死锁进程
B. 终止所有死锁进程
C. 从死锁进程处抢夺资源
D. 从非死锁进程处抢夺资源*

4. 采用资源剥夺法可以解除死锁，还可以采用()方法解除死锁。
*A. 执行并行操作 B. 撤销进程
C. 拒绝分配新资源 D. 修改信号量*

5. 在下列死锁的解决方法中，属于死锁预防策略的是()
A. 银行家算法

B. 资源有序分配算法

C. 死锁检测算法

D. 资源分配图化简法

6. 三个进程共享四个同类资源，这些资源的分配与释放只能一次一个。已知每个进程最多需要两个该类资源，则该系统（ ）

A. 有些进程可能永远得不到该类资源

B. 必然有死锁

C. 进程请求该类资源必然能得到

D. 必然是死锁

7. 产生死锁的四个必要条件是：互斥、 __、循环等待和不剥夺。

A. 请求与阻塞

B. 请求与保持

C. 请求与释放

D. 释放与阻塞

8. 在__的情况下，系统出现死锁。

A. 计算机系统发生了重大故障

B. 有多个封锁的进程同时存在

C. 若干进程因竞争资源而无休止地相互等待他方释放已占有的资源

D. 资源数大大小于进程数或进程同时申请的资源数大大超过资源总数

9. 以下叙述中正确的是____。

- A. 调度原语主要是按照一定的算法，从阻塞队列中选择一个进程，将处理机分配给它。
- B. 预防死锁的发生可以通过破坏产生死锁的四个必要条件之一来实现，但破坏互斥 条件的可能性不大
- C. 进程进入临界区时要执行开锁原语。
- D. 既考虑作业等待时间，又考虑作业执行时间的调度算法是先来先服务算法

10. 死锁定理是用于处理死锁的()方法

- A. 预防死锁
- B. 避免死锁
- C. 检测死锁
- D. 解除死锁

课时六 死锁（二）

考点	重要程度	占分	题型
1. 死锁避免-银行家算法	必考	5~10	选择、应用
2. 死锁的检测与解除	★★★	3~6	选择

1. 死锁的避免

避免死锁同样属于事先预防策略，是在资源动态分配过程中，防止系统进入不安全状态，以避免发生死锁。

1) 系统安全状态

在避免死锁方法中，把系统的状态分为安全状态和不安全状态。当系统处于安全状态时可避免发生死锁，而当系统处在不安全状态时，则可能会进入死锁状态。

2) 安全序列

所谓安全序列，就是指如果系统按照这种序列分配记资源，则每个进程都能顺利完成。只要能找出一个安全序列，系统就是安全状态。当然，安全序列可能有多个。

3) 由安全状态进入不安全状态

如果分配了资源之后，系统中找不出任何一个安全序列，系统就进入了不安全状态。这就意味着之后可能所有进程都无法顺利的执行下去。

如果系统处于安全状态，就一定不会发生死锁。如果系统进入不安全状态，可能会发生死锁。（不安全状态未必就是发生了死锁，但发生死锁时一定是在不安全状态）

因此可以在资源分配之前预先判断这次分配是否会导致系统进入不安全状态，以此决定是否答应资源分配请求。这也是“银行家算法”的核心思想。

4) 银行家算法

银行家算法是荷兰学者 Dijkstra 为银行系统设计的，以确保银行在发放现金贷款时，不会发生不能满足所有客户需要的情况。后来该算法被用在操作系统

中，用于避免死锁。

核心思想：在进程提出资源申请时，先预判此次分配是否会导致系统进入不安全状态。如果会进入不安全状态，就暂时不答应这次请求，让该进程先阻塞等待。

1) 数据结构：

①可利用资源向量 $Available$ 。长度为 m 的一维数组， $Available$ 表示还有多少可用资源。

②最大需求矩阵 Max 。表示各进程对资源的最大需求数， $n \times m$ 矩阵。

③分配矩阵 $Allocation$ 。表示已经给各进程分配了多少资源， $n \times m$ 矩阵。

④需求矩阵 $Need$ 。矩阵表示各进程最多还需要多少资源，
 $Max - Allocation = Need$ 。

⑤进程 P 的请求向量。用长度为 m 的一位数组，表示进程此次申请的各种资源数。

2) 算法步骤：

设 $Request$ 是进程 P_i 的请求向量，如果 $Request[i] = K$ ，表示进程 P_i 需要 K 个 R_j 类型的资源。当 P_i 发出资源请求后，系统按下述步骤进行检查：

①如果 $Request[i] \leq Need[i]$ 便转向步骤②；否则认为出错，因为它所需要的资源数已超过它所宣布的最大值。

②如果 $Request[i] \leq Available[j]$ ，便转向步骤③；否则，表示尚无足够资源， P_i 须等待。

③系统试探着把资源分配给进程 P_i ，并修改下面数据结构中的数值

$$Available[j] = Available[j] - Request[i];$$

$$Allocation[i,j] = Allocation[i,j] + Request[i];$$

$$Need[i,j] = Need[i,j] - Request[i];$$

④系统执行安全性算法，检查此次资源分配后系统是否处于安全状态。若安

全，才正式将资源分配给进程 P_i ，以完成本次分配；否则，将本次的试探分配作废，恢复原来的资源分配状态，让进程 P_i 等待。

3) 安全性算法：

系统所执行的安全性算法可描述如下：

①设置两个向量：

- a. 工作向量 $Work$ ，它表示系统可提供给进程继续运行所需的各类资源数目，它含有 m 个元素，在执行安全算法开始时， $Work = Available$ ；
- b. $Finish$ ：它表示系统是否有足够的资源分配给进程，使之运行完成。开始时先做 $Finish[i] = false$ ；当有足够资源分配给进程时，再令 $Finish[i] = true$ 。

②从进程集合中找到一个能满足下述条件的进程

- a. $Finish[i] = false$ ；
- b. $Need[i,j] \leq Work[j]$ ；

若找到，执行步骤③，否则，执行步骤④。

- ③当进程 P_i 获得资源后，可顺利执行，直至完成，并释放出分配给它的资源，故应执行：

$$Work[j] = Work[j] + Allocation[i,j] ;$$

$$Finish[i] = true ;$$

返回步骤②。

- ④如果所有进程的 $Finish[i] = true$ 都满足，则表示系统处于安全状态；否则，系统处于不安全状态。

4) 安全算法举例

算法：银行家算法

假定系统中有 5 个进程 $\{P_0, P_1, P_2, P_3, P_4\}$ 和 3 类资源 $\{A, B, C\}$ ，各类资源的数量分别为 10、5、7，在 t_0 时刻的资源分配情况如图 1 所示

资源情况 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	7	5	3	0	1	0	7	4	3	3	3	2
										2	3	0
P_1	3	2	2	2	0	0	1	2	2			
P_2	9	0	2	3	0	2	6	0	0			
P_3	2	2	2	2	1	1	0	1	1			
P_4	4	3	3	0	0	2	4	3	1			

图1 t_0 时刻的资源分配情况

① t_0 时刻的安全性：利用安全性算法对 t_0 时刻的资源分配情况进行分析可知，在 t_0 时刻存在着一个安全序列 $\{P_0, P_1, P_2, P_3, P_4\}$ ，故系统是安全的。

资源情况 进程	Max			Need			Allocation			Work + Allocation			Finish
	A	B	C	A	B	C	A	B	C	A	B	C	
P_1	3	3	2	1	2	2	2	0	0	5	3	2	TRUE
P_3	5	3	2	0	1	1	2	1	1	7	4	3	TRUE
P_4	7	4	3	4	3	1	0	0	2	7	4	5	TRUE
P_2	7	4	5	6	0	0	3	0	2	10	4	7	TRUE
P_0	10	4	7	7	4	3	0	1	0	10	5	7	TRUE

图2 t_0 时刻的安全序列

- ② P_1 请求资源； P_1 发出请求向量 $Request_1(1, 0, 2)$ ，系统按银行家算法进行检查。
- $Request_1(1, 0, 2) \leq Need_1(1, 2, 2)$
 - $Request_1(1, 0, 2) \leq Available_1(3, 3, 2)$
 - 系统先假定可为 P_1 分配资源，并修改 $Available$ 、 $Allocation_1$ 、 $Need_1$ 向量，由此形成的资源变化情况如图1的圆括号所示。
 - 再利用安全性算法检查此时系统是否安全，如图3所示

资源情况 进程	Work			Need			Allocation			Work + Allocation			Finish
	A	B	C	A	B	C	A	B	C	A	B	C	
P_1	2	3	0	0	2	0	3	0	2	5	3	2	TRUE
P_3	5	3	2	0	1	1	2	1	1	7	4	3	TRUE
P_4	7	4	3	4	3	1	0	0	2	7	4	5	TRUE
P_0	7	4	5	7	4	3	0	1	0	7	5	5	TRUE
P_2	7	5	5	6	0	0	3	0	2	10	5	7	TRUE

图3 P_1 请求资源时的安全性检查

由所进行的安全性检查得知，可以找到一个安全序列 $\{P_1, P_3, P_4, P_2, P_0\}$ 。因此，系统是安全的，可以立即将 P_1 所申请的资源分配给它。

③ P_4 请求资源： P_4 发出请求向量 $Request_4(3, 3, 0)$ ，系统按银行家算法进行检查。

- a. $Request_4(3, 3, 0) \leq Need_4(4, 3, 1)$
- b. $Request_4(3, 3, 0) > Available(2, 3, 0)$ ，让 P_4 等待

④ P_0 请求资源： P_0 发出请求向量 $Request_0(0, 2, 0)$ ，系统按银行家算法进行检查。

- a. $Request_0(0, 2, 0) \leq Need_0(7, 4, 3)$
- b. $Request_0(0, 2, 0) \leq Available(2, 3, 0)$
- c. 系统暂时先假定可为 P_0 分配资源，并修改有关数据，如图4所示。

资源情况 进程	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	3	0	7	2	3			
P_1	3	0	2	0	2	0			
P_2	3	0	2	6	0	0	2	1	0
P_3	2	1	1	0	1	1			
P_4	0	0	2	4	3	1			

图4 为 P_0 分配资源后的有关数据

⑤进行安全检查：可用资源 $Available(2, 1, 0)$ 已不能满足任何进程的需要，故系统进入不安全状态，此时系统不分配资源。

2. 死锁的检测与解除

如果系统中既不采用预防死锁也不采用避免死锁的措施，系统就很有可能发生死锁。这种情况下，系统应当提供两个算法：

1) 死锁检测算法：用于检测系统状态，已确定系统中是否发生了死锁。

为了能对死锁进行检测，在系统中必须：

①用某种数据结构来保存资源的请求和分配信息。

②提供一种算法，利用上述信息来检测系统是否进入死锁状态。

资源分配图

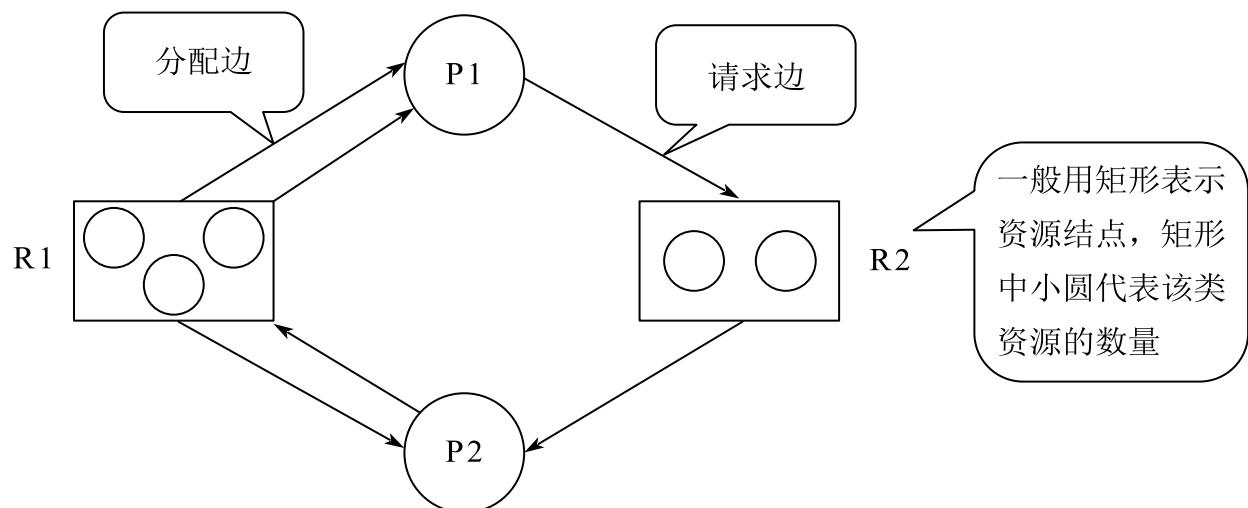
两种结点：进程结点和资源结点（资源结点可能有多个）

两种边：

进程结点→资源结点：代表一个进程结点请求一类资源。

资源结点→进程结点：代表这类资源已经分配给了指向的进程结点。

死锁定理：如果某时刻系统的资源分配图是不可简化的，那么此时系统死锁。



2) 死锁解除算法：当使用这种算法的时候该算法可将系统从死锁中解脱出来。一旦检测出死锁的发生就应该立即解除死锁。

补充：并不是系统中所有的进程都是死锁状态，用死锁检测算法分配图后，还连着边的进程就是死锁进程。

解除死锁的主要方法：

①资源剥夺法。将某些死锁进程锁占有的资源剥夺并分配给其他进程。但是应该防止被挂起的进程长时间得不到资源饿死。

②撤销进程法。强制撤销部分死锁进程或者全部死锁进程，简单粗暴但是付出的代价较大。

③进程回退法。让一个或多个死锁进程回退到足以避免死锁的地步。这个就要求记录进程的历史消息，设置还原点。

题 1. 死锁的避免是根据()采取措施实现的。

- A. 配置足够的系统资源
- B. 使进程的推进顺序合理
- C. 破坏死锁的四个必要条件之一
- D. 防止系统进入不安全状态

答案：D

解析：死锁避免是指在资源动态分配过程中用某些算法加以限制，防止系统进入不安全状态从而避免死锁的发生。选项B是避免死锁后的结果，而不是措施的原理。

题 2. 某系统中有三个并发进程都需要四个同类资源，则该系统必然不会发生死锁的最少资源是()

- A. 9
- B. 10
- C. 11
- D. 12

答案: B

解析: 资源数为9时, 存在三个进程都占有三个资源, 为死锁; 资源数为10时, 必然存在一个进程能拿到4个资源, 然后可以顺利执行完其他进程。

题 3. 某系统中共有11台磁带机, X 个进程共享此磁带机设备, 每个进程最多请求使用3台, 则系统必然不会死锁的最大 X 值是()

- A. 4 B. 5 C. 6 D. 7

答案: B

解析: 考虑一下极端情况: 每个进程已经分配了两台磁带机, 那么其中任何一个进程只要再分配一台磁带机即可满足它的最大需求, 该进程总能运行下去直到结束, 然后将磁带机归还给系统再次分配给其他进程使用。因此, 系统中只要满足 $2X + 1 = 11$ 这个条件即可认为系统不会死锁, 解得 $X = 5$, 也就是说, 系统中最多可以并发5个这样的进程是不会死锁的。

课时六 练习题

1. 死锁的避免是根据()采取措施实现的。

- A. 配置足够的系统资源
- B. 使进程的推进顺序合理
- C. 破坏死锁的四个必要条件之一
- D. 防止系统进入不安全状态

2. 以下有关资源分配图的描述中，正确的是()。

- A. 有向边包括进程指向资源类的分配边和资源类指向进程申请边两类
- B. 矩形框表示进程，其中圆点表示申请同一类资源的各个进程
- C. 圆圈结点表示资源类
- D. 资源分配图是一个有向图，用于表示某时刻系统资源与进程之间的状态

3. 死锁与安全状态的关系是()

- A. 死锁状态有可能是安全状态
- B. 安全状态有可能成为死锁状态
- C. 不安全状态就是死锁状态
- D. 死锁状态一定是不安全状态

4. 系统的资源分配图在下列情况下，无法判断是否处于死锁状态的有()

- I. 出现了环路
- II. 没有环路
- III. 每种资源只有一个，并出现环路
- IV. 每个进程结点至少有一条请求边

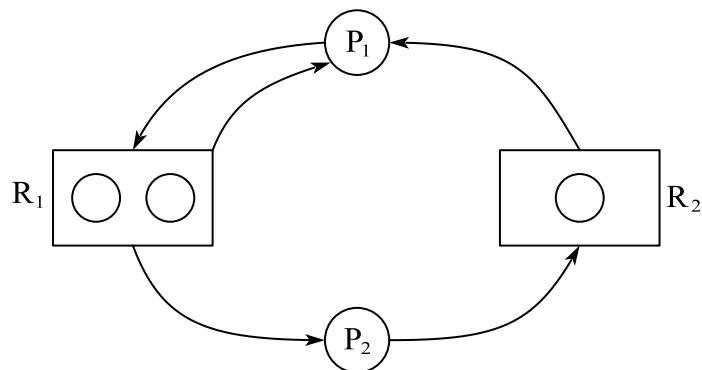
5. 假设具有 5 个进程的进程集合 $P = \{P_0, P_1, P_2, P_3, P_4\}$ ，系统中有三类资源 A, B, C ，假设在某时某刻有如下状态，见下表。

	Allocation			Max			Available		
	A	B	C	A	B	C			
P_0	0	0	3	0	0	4			
P_1	1	0	0	1	7	5	A	B	C
P_2	1	3	5	2	3	5	x	y	z
P_3	0	0	2	0	6	4			
P_4	0	0	1	0	6	5			

请问当 x, y, z 取下列哪些值时，系统是处于安全状态的？

- I . 1, 4, 0 II . 0, 6, 2 III. 1, 1, 1 IV. 0, 4, 7
 A. II、III B. I、II
 C. 仅 I D. I、III

6. 假定某计算机系统有 R_1 和 R_2 两类可使用资源（其中 R_1 有两个单位， R_2 有一个单位），它们被进程 P_1 和 P_2 所共享，且已知两个进程均以下列顺序使用两类资源：申请 $R_1 \rightarrow$ 申请 $R_2 \rightarrow$ 申请 $R_1 \rightarrow$ 释放 $R_1 \rightarrow$ 释放 $R_2 \rightarrow$ 释放 R_1 。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）。



课时七 进程同步（一）

考点	重要程度	占分	题型
1. 同步与互斥的基本概念	必考	2~5	选择、简答
2. 软件同步机制	★★	1~3	选择
3. 硬件同步机制	★★★	1~3	选择

1. 同步与互斥的基本概念

1) 临界资源

①**临界资源**: 是一次仅允许一个进程使用的共享资源。各进程采取互斥的方式，实现共享的资源称作临界资源。属于临界资源的硬件有，打印机，磁带机等；软件有消息队列，变量，数组，缓冲区等。

②**临界区**: 每个进程中访问临界资源的那段代码称为临界区，每次只允许一个进程进入临界区，进入后，不允许其他进程进入。

多个进程涉及到同一个临界资源的的临界区称为相关临界区。

临界资源的访问过程分为4个部分：

- a. 进入区。负责检查是否可以进入临界区，设置正在访问临界资源的标志。
- b. 临界区。访问临界资源的那段代码。
- c. 退出区。负责接触正在访问临界资源的标志。
- d. 剩余区。做其他处理。

题 1. 下列对临界区的论述中，正确的是（ ）

- A. 临界区是指进程中用于实现进程互斥的那段代码
- B. 临界区是指进程中用于实现进程同步的那段代码
- C. 临界区是指进程中用于实现进程通信的那段代码
- D. 临界区是指进程中用于访问临界资源的那段代码

答案：D

2) 进程同步

同步也称直接制约关系，它指的是多个进程一起完成某个任务，这些进程因为合作、因为需要在某些位置上协调他们的工作次序而产生了某些制约关系。

注意：不要把进程同步和进程调度搞混了。

①进程调度是为了最大程度的利用CPU资源，选用合适的算法调度就绪队列中的进程。

②进程同步是为了协调一些进程以完成某个任务，比如读和写，肯定先写后读，不能先读后写，这就是进程同步做的事情了，指定这些进程的先后执行次序使得某个任务能够顺利完成。

3) 进程互斥：

进程互斥也称间接制约关系。当某个进程A在访问临界区使用临界资源时，另一个B进程必须等待，直到A进程访问结束并释放打印机资源后，B进程才能去访问。

为禁止两个进程同时进入临界区，同步机制应遵循以下准则：

- ①空闲让进。临界区空闲时，可以允许一个请求进临界区的进程立即进入临界区。
- ②忙则等待。当已有进程进入临界区时，其他试图进入临界区的进程必须等待。
- ③有限等待。对请求访问的进程，应保证能在有限内时间内进入临界区。
- ④让权等待。当进程不能进入临界区时，应立即释放处理器，防止进程忙等待。

题 2. 进程之间存在着哪几种制约关系?它们各由什么原因引起?下列活动分别属于哪种制约关系?

- 1) 若干位同学去图书馆借书;
- 2) 两队举行篮球比赛;
- 3) 流水线生产的各道工序;
- 4) 商品生产和社会消费。

解析：进程之间存在着直接相互制约和间接相互制约这两种制约关系，其中直接相互制约(同步)是由进程间的相互合作引起的；而间接相互制约(互斥)则是由

进程间共享临界资源引起的。

- 1) 若干位同学去图书馆借书是间接相互制约，其中书是临界资源。
- 2) 两队举行篮球比赛是间接相互制约 其中篮球是临界资源
- 3) 流水线生产的各道工序是直接相互制约，各道工序间须相互合作，每道工序的开始都依赖于前一道工序的完成。
- 4) 商品生产和社会消费是直接相互制约的，两者须相互合作。商品生产出来后才可以被消费，商品被消费后才须再生产。

2. 进程同步机制

- 1) 软件同步机制

一个经典的解决方案：*Peterson 算法*。

算法思想：为防止两个进程为进入临界区而无限等待，设置了变量 $turn$ ，每个进程先设置自己的标志后再设置 $turn$ 标志。这时，再同时检测另一个进程状态标志和不允许进入标志，以便保证两个进程同时要求进入临界区时，只允许一个进程进入临界区。

算法如下：

```
bool flag[2];
int turn=0;
P0 进程:
flag[0]=true;
true=1;
while (flag[1]&&turn==1);
critical section;
```

```
flag[0]=false;
remainder section;
```

P1 进程:

```
flag[1]=true;
```

```
turn=0;  
while (flag[0]&&turn==0);  
critical section;  
flag[1]=false;  
remainder section;
```

Peterson 算法用软件方法解决了进程互斥问题，遵循了空闲让进、忙则等待、有限等待三个原则，但是依然未遵循让权等待的原则。但相较于其他算法，这种解决方案时最好的。

题 2. 进程 P0 和 P1 的共享变量定义及其初值为： boolean flag[2];int turn=0;
flag[0]=FALSE; flag[1]=FALSE; 若进程 P0 和 P1 访问临界资源的类 C 代码实现
如下：

```
void P0()  
{  
    while(TRUE){  
        flag[0]= TRUE;  
        turn=1;  
        while(flag[1]&&(turn==1));  
        临界区;  
        flag[0]= FALSE;  
    }  
}
```

```
void P1()  
{  
    while(TRUE){  
        flag[1]= TRUE;
```

```
turn=0;  
while(flag[0]&&(turn==0));  
临界区;  
flag[1]= FALSE;  
}  
}
```

则并发执行进程P0 和P1 时发生的情况是()

- A. 不能保证进程互斥地进入临界区，会出现“饥饿”现象
- B. 不能保证进程互斥地进入临界区，不会出现“饥饿”现象
- C. 能保证进程互斥地进入临界区，会出现“饥饿”现象
- D. 能保证进程互斥地进入临界区，不会出现“饥饿”现象

答案：D

解析：算法实现互斥的主要思想在于设置一个 *turn* 变量，用于进程间的互相谦让。如果进程P0 试图访问临界资源，flag[0]=TRUE，则表示希望访问。此时如果进程P1 还未试图访问临界资源，则 flag[1]在进程上一次访问完临界资源并退出临界区后已设置为 FALSE。因此进程P0 在执行循环判断条件时，第一个条件不满足，进程P0 可以正常进入临界区，且满足互斥条件。请注意两个进程同时试图访问临界资源的情况。*turn* 变量的含义：进程在试图访问临界资源时，首先设置自己的 flag 变量为 TRUE，表示希望访问；但又会设置 *turn* 变量为对方的进程编号，表示谦让，若在 *turn* 变量不是进程编号时就循环等待，则两个进程就会互相谦让，由于 *turn* 变量会有一个最终值，因此先做出谦让的进程先进入临界区，后做出谦让的进程则需要循环等待，不会造成“饥饿”现象。

2) 硬件同步机制

① 关中断

关中断是实现互斥最简单的方法之一，在进入锁测试之前，关闭中断，直到完成锁测试并上锁之后才能打开中断。

优点：简单、高效

缺点：不适合多CPU系统；只适合操作系统内核进程，不适合用户进程。

② 利用 *Test – and – Set* 指令实现互斥

简称TS指令或TSL指令。TSL指令是由硬件实现的，执行过程不允许被中断，即一条原语。

```
bool TestAndSet (bool*lock){
```

```
    bool old;
```

```
    old=*lock;
```

```
    *lock=true;
```

```
    return old;
```

```
}
```

```
while(TestAndSet(&lock));
```

临界区代码段...

```
lock=false;
```

剩余区代码段...

优点：实现简单，适用于多CPU系统；

缺点：不满足“让权等待”原则，暂时无法进入临界区的进程会占用CPU并循环执行TSL指令，从而导致“忙等”。

③ 利用 swap 指令直线进程互斥

swap指令被称为对换指令，是由硬件实现，执行过程也是不允许被中断的。

④实时系统

实时系统是指系统能及时响应外部事件的请求，在规定的时间内完成对该事件的处理，并控制所有实时任务协调一致地运行。

```
estAndSet(  
    swap(bool*a,bool*b){  
        bool temp;  
        temp=*a;  
        *a=*b;  
        *b=temp;  
    }
```

```
bool old=ture;  
while(old==ture)  
    swap(&lock,&old);  
临界区代码段...  
lock=false;  
剩余区代码段...
```

值得注意的是：在逻辑上 *swap* 指令和 *TSL* 并无太大区别。

优点：实现简单，适用于多 *CPU* 系统。

缺点：不满足“让权等待”原则，暂时无法进入临界区的进程会占用 *CPU* 并循环执行 *TSL* 指令，从而导致“忙等”。

课时七 练习题

1. 一个正在访问临界资源的进程由于申请等待 I/O 操作而被中断时，它是（ ）
 - A. 允许其他进程进入与该进程相关的临界区
 - B. 不允许其他进程进入任何临界区
 - C. 允许其他进程抢占处理器，但不得进入该进程的临界区
 - D. 不允许任何进程抢占处理器

2. 两个旅行社甲和乙为旅客到某航空公司订飞机票，形成互斥资源的是（ ）
 - A. 旅行社
 - B. 航空公司
 - C. 飞机票
 - D. 旅行社与航空公司

3. 临界区是指并发进程访问共享变量段的（ ）
 - A. 管理信息
 - B. 信息存储
 - C. 数据
 - D. 代码程序

4. 以下（ ）属于临界资源。
 - A. 磁盘存储介质
 - B. 公用队列
 - C. 私用数据
 - D. 可重入的程序代码

5. 在操作系统中，要对并发进程进行同步的原因是()。

- A. 进程必须在有限的时间内完成
- B. 进程具有动态性
- C. 并发进程是异步的
- D. 进程具有结构性

6. 进程A和进程B通过共享缓冲区协作完成数据处理，进程A负责产生数据并放入缓冲区，进程B从缓冲区读数据并输出。进程A和进程B之间的制约关系是()

- A. 互斥关系
- B. 同步关系
- C. 互斥和同步关系
- D. 无制约关系

7. 下列关于临界区和临界资源的说法中，正确的是()

I. 银行家算法可以用来解决临界区(*Critical Section*)问题

II. 临界区是指进程中用于实现进程互斥的那段代码

III. 公用队列属于临界资源

IV. 私用数据属于临界资源

- A. I、II
- B. I、IV
- C. 仅III
- D. 以上答案都错误

8、多个进程并发执行时，各个进程应互斥进入其临界区，所谓临界区是指

- A. 一段程序
- B. 一段数据区
- C. 一个缓冲区
- D. 一种同步机制

课时八 进程同步（二）

考点	重要程度	占分	题型
1. 信号量	必考	2~5	选择、应用
2. 信号量的基本应用	必考	3~6	选择、应用
3. 管程	★	1~2	选择、填空

1. 信号量

信号量机制是一种功能较强的机制，可用来解决互斥和同步问题，它只能被两个标准的原语 $wait(S)$ 和 $signal(S)$ 访问，也可记为“P 操作”和“V 操作”。

注意：原语是一种特殊的程序段，其执行只能一气呵成，不可被中断，原语由关中断/开中断指令实现。

1) 整形信号量

整形信号量被定义为一个用于表示资源数目的整型量 S ， $wait$ 和 $signal$ 操作可描述为：

```

wait(S){
    while(S<=0);
    S=S-1;
}
signal(S){
    S=S+1;
}

```

该机制并未遵循“让权等待”的准则，而是使进程处于“忙等”的状态。

2) 记录型信号量

记录型信号量是不存在“忙等”现象的进程同步机制。除需要一个用于代表资源数目的整型变量 $value$ 外，再增加一个进程链表 L ，用于链接所有等待该资源的进程。记录型信号量得名于采用了记录型的数据结构。记录型信号量可描述为：

```
typedef struct{
```

```

int value;
struct process*L;
}semaphore;
void wait(semaphore S){
S.value--;
if(S.value<0){
add this process to S.L;
block(S.L);
}
}

void signal(semaphore S){
S.value++;
if(S.value<=0){
remove a process to S.L;
wakeup(P);
}
}

```

① $S \rightarrow value$ 的初值表示系统中某类资源的数目，因而又被称为资源信号量，

② 对它进行的每次 $wait(S)$ 操作 (P 操作)，意味着进程请求一个单位的该类资源，这会使系统中可供分配的该类资源数减少一个，因此描述为 $S \rightarrow value--$ ；当 $S \rightarrow value < 0$ 时，表示该类资源已分配完毕，因此进程应调用 $block$ 原语，进行自我阻塞，放弃处理机，并将该进程插入信号量链表 $S \rightarrow list$ 中。可见，该机制遵循了“让权等待”准则。

③ 对信号量的每次 $signal(S)$ 操作 (V 操作)，表示执行进程释放一个单位的资源，这会使系统中可供分配的该类资源数增加一个，故 $S \rightarrow value++$ 操作表示

资源数目加1。若加1后仍是 $S \rightarrow value <= 0$ ，则表示在该信号量链表中仍有等待该资源的进程被阻塞，故还应调用 *wakeup* 原语唤醒第一个等待进程。

④如果 $S \rightarrow value$ 的初值为1，则表示只允许一个进程访问临界资源，此时的信号量会转化为互斥信号量，用于进程互斥。

2. 信号量的基本应用

1) 实现多个进程的互斥

①互斥信号量 *mutex* 初值为1；

②每个进程中将临界区代码置于 *P(mutex)* 和 *V(mutex)* 原语之间；

③必须成对使用 *P* 和 *V* 原语（在同一进程中），不能次序错误、重复或遗漏。

遗漏 *P* 原语则不能保证互斥访问，遗漏 *V* 原语则不能在使用临界资源之后将其释放（给其他等待的进程）。

```
semaphore mutex=1;
```

```
P1(){
```

```
...
```

```
P(mutex);
```

临界区代码段...

```
V(mutex);
```

```
...
```

```
}
```

```
P2(){
```

```
...
```

```
P(mutex);
```

临界区代码段...

```
V(mutex);
```

```
...
```

}

2) 实现同步

- ①分析什么地方需要实现“同步关系”，即必须保证“一前一后”执行的两个操作(或两句代码)；
- ②设置同步信号量 S ，初始为0；
- ③在“前操作”之后执行 $V(S)$ ；
- ④在“后操作”之前执行 $P(S)$ 。

/*信号机制实现同步*/

semaphore S = 0; //初始化同步信号量，初始值为0

P1(){

代码 1;

代码 2;

V(S);

代码 3;

}

P2(){

P(S);

代码 4;

代码 5;

代码 6;

}

保证了代码4一定是在代码2之后执行。

3) 实现前驱关系

其实每一对前驱关系都是一个进程同步问题(需要保证一前一后的操作)因

- ①要为每一对前驱关系各设置一个同步变量；
- ②在“前操作”之后对相应的同步变量执行 V 操作；
- ③在“后操作”之前对相应的同步变量执行 P 操作。

信号量值为0的点是限制的关键所在；

成对使用 P 和 V 原语（在有先后关系的两个进程中），不能次序错误、重复或遗漏，否则同步顺序出错。

信号量题目做题一般方法：

- ①分析问题，找出同步、互斥关系；
- ②根据资源设置信号量变量；
- ③写出代码过程，并注意 P 、 V 操作的位置；
- ④检查代码，模拟机器运行，体验信号量的变化和程序运行过程是否正确。

题 1. 设与某资源关联的信号量初值为3，当前值为1。若 M 表示该资源的可用个数， N 表示等待该资源的进程数，则 M, N 分别是（ ）

- A. 0, 1 B. 1, 0 C. 1, 2 D. 2, 0

答案：B

解析：信号量表示相关资源的当前可用数量。当信号量 $K > 0$ 时，表示还有 K 个相关资源可用，所以该资源的可用个数是1。而当信号量 $K < 0$ 时，表示有 $|K|$ 个进程在等待该资源。由于资源有剩余，可见没有其他进程等待使用该资源，因此进程数为0。

题 2. 对于两个并发进程，设互斥信号量为 $mutex$ (初值为1)，若 $mutex = -1$ ，则

()

- A. 表示没有进程进入临界区
- B. 表示有一个进程进入临界区
- C. 表示有一个进程进入临界区，另一个进程等待进入
- D. 表示有两个进程进入临界区

答案：C

解析：当有一个进程进入临界区且有另一个进程等待进入临界区时， $mutex = -1$ 。

$mutex$ 小于0时，其绝对值等于等待进入临界区的进程数。

题 3. 下面是两个并发执行的进程，它们能正确运行吗？若不能请举例说明并改正。

```
int x;  
process_P1 {  
    int y,z;  
    x=1;  
    y=0;  
    if(x>=1)  
        y=y+1;  
    z=y;  
}
```

```
process_P2 {  
    int t,u;  
    x=0;  
    t=0;  
    if(x<=1)
```

蜂考

```
t=t+2;  
u=t;  
}
```

解析：P1 和 P2 两个并发进程的执行结果是不确定的，它们都是对同一变量 x 进行操作， x 是一个临界资源，而没有进行保护。例如：

- 1) 若先执行完 P1 再执行 P2，结果是 $x = 0, y = 1, z = 1, t = 2, u = 2$ 。
- 2) 若先执行 P1 到 “ $x = 1$ ”，然后一个中断去执行完 P2，再一个中断回来执行完 P1，结果是 $x = 0, y = 0, z = 0, t = 2, u = 2$ 。

显然，两次执行结果不同，所以这两个并发进程不能正确运行。可将这个程序改为：

```
int x;  
semaphore S=1;  
process_P1 {  
    int y,z;  
    P(S);  
    x=1;  
    y=0;  
    if(x>=1)  
        y=y+1;  
    V(S);  
    z=y;  
}
```

```
process_P2 {  
    int t,u;  
    P(S);  
    x=0;
```

```
t=0;  
if(x<=1)  
t=t+2;  
V(S);  
u=t;  
}
```

3. 管程

1) 为什么引入管程?

信号量机制存在：编写程序困难、易出错等问题，而管程——一种高级的同步机制，可以让程序员学程序是不需要再关注复杂的PV操作，让写程序更加轻松。

2) 管程的定义

管程是一种特殊的软件模块，

管程包含：①多个彼此可以交互并共用资源的线程

②多个与资源使用有关的变量

③一个互斥锁

④一个用来避免竞态条件的不变量

3) 管程的基本特征

①模块化。

管程是一个基本的软件模块，可以被单独编译。

②抽象数据类型。

管程中封装了数据及对于数据的操作，这点有点像面向对象编程语言中的类。

③信息隐藏。

管程外的进程或其他软件模块只能通过管程对外的接口来访问管程提供的操作，管程内部的实现细节对外界是透明的。

④使用的互斥性。

任何一个时刻，管程只能由一个进程使用。进入管程时的互斥由编译器负责完成。

题 1. 下述()选项不是管程的组成部分。

- A. 局限于管程的共享数据结构
- B. 对管程内数据结构进行操作的一组过程
- C. 管程外过程调用管程内数据结构的说明
- D. 对局限于管程的数据结构设置初始值的语句

答案： C

解析：管程由局限于管程的共享变量说明、对管程内的数据结构进行操作的一组过程及对局限于管程的数据设置初始值的语句组成。

课时八 练习题

1. 一个正在访问临界资源的进程由于申请等待 I/O 操作而被中断时，它是_____。

- A. 可以允许其他进程进入与该进程相关的临界区
- B. 不允许其他进程进入任何临界区
- C. 可以允许其他进程抢占处理机，但不得进入该进程的临界区
- D. 不允许任何进程抢占处理机

2. 在9个生产者、6个消费者共享容量为8的缓冲器的生产者-消费者问题中，互斥使用缓冲器的信号量初始值为()。

- A. 1
- B. 6
- C. 8
- D. 9

3. 下列关于管程的叙述中，错误的是()。

- A. 管程只能用于实现进程的互斥
- B. 管程是由编程语言支持的进程同步机制
- C. 任何时候只能有一个进程在管程中执行
- D. 管程中定义的变量只能被管程内的过程访问

4. 对信号量 S 执行 P 操作后，使该进程进入资源等待队列的条件是()。

- A. $S.value < 0$
- B. $S.value <= 0$
- C. $S.value > 0$
- D. $S.value >= 0$

5. 有三个进程共享同一程序段，而每次只允许两个进程进入该程序段，若用 PV 操作同步机制，则信号量 S 的取值范围是()。

- A. 2, 1, 0, -1
- B. 3, 2, 1, 0

C. 2, 1, 0, -1, -2

D. 1, 0, -1, -2

6. 有两个并发进程 P_1 和 P_2 ，其程序代码如下：

```
P1(){  
    x=1;      //A1  
    y=2;  
    z=x+y;  
    print z;  //A2  
}  
  
P2(){  
    x=-3;     //B1  
    c=x*x;  
    print c;  //B2  
}
```

可能打印出的 z 值有()，可能打印出的 c 值有()，(其中 x 为 P_1, P_2 的共享变量)。

A. $z = 1, -3; c = -1, 9$

B. $z = -1, 3; c = 1, 9$

C. $z = -1, 3, 1; c = 9$

D. $z = 3; c = 1, 9$

7. 有两个并发进程 P_1 和 P_2 ，其程序代码如下：

```
P1(){  
    x=1;  
    y=2;  
    if(x>0)  
        z=x+y;  
    else  
        z=x*y;  
    print z;
```

```
P2() {  
    x=-1;  
    a=x+3;  
    x=a+x;  
    b=a+x;  
    c=b*b;  
    print c;  
}
```

1) 可能打印出的 z 值有多少? (假设每条赋值语句是一个原子操作)

2) 可能打印出的 c 值有多少? (其中 x 为 P_1 和 P_2 的共享变量)

8. 某寺庙有小和尚、老和尚若干, 有一水缸, 由小和尚提水入缸供老和尚饮用, 水缸可容10桶水, 水取自同一井中, 水井径窄, 每次只能容一个桶取水。水桶总数为3个。每次入缸取水仅为1桶水, 且不可同时进行。试给出有关从缸取水、入水的算法描述。

课时九 进程同步（三）

考点	重要程度	占分	题型
1. 生产者消费者问题	★★	3~9	选择、填空、简答、应用
2. 读者写者问题	★★		
3. 哲学家进餐问题	必考		

1. 生产者消费者问题

1) 问题描述

系统中有一组生产者进程和一组消费者进程，生产者进程每一次生产一个产品放入缓冲区，消费者进程每次从缓冲区中取出一个产品并使用。（注：这里的“产品”理解为某种数据）。

生产者、消费者共享一个初始为空、大小为 n 的缓冲区。

只有缓冲区没满时，生产者才能把产品放入缓冲区，否则必须等待。

只有缓冲区不空时，消费者才能从中取出产品，否则必须等待。

缓冲区是临界资源，各进程必须互斥地访问。

2) 问题分析

该问题中出现的主要的两种关系：

①生产者—消费者之间的同步关系表现为：一旦缓冲池中所有缓冲区均装满产品时，生产者必须等待消费者提供空缓冲区；一旦缓冲池中所有缓冲区全为空时，消费者必须等待生产者提供满缓冲区。

②生产者—消费者之间还有互斥关系：由于缓冲池是临界资源，所以任何进程在对缓冲区进行存取操作时都必须和其他进程互斥进行。

PV操作题目分析的步骤：

①关系分析。找出题目中描述的各个进程，分析它们之间的同步、互斥关系。

②整理思路。根据各进程的操作流程确定PV操作的大致顺序。

③设置信号量。设置需要的信号量，并根据题目条件确定信号量的初值。（互斥信号量初值一般为1，同步信号量的初值需要看对应资源的初始值是多少）

3) 如何实现

互斥的实现是在同一个进程中进行的一对 *PV* 操作。

同步的实现是在两个进程中进行的，在一个进程中执行 *P* 操作，在另一个进程中执行 *V* 操作。

```
semaphore mutex = 1; //互斥信号量
semaphore empty = n; //同步信号量。空闲缓冲区的数量
semaphore full = 0; //同步信号量。产品的数量，非空缓冲区的数量
producer(){
    while(1){
        生成一个产品;
        P(empty); //消耗一个空闲缓冲区
        P(mutex);
        把产品放入缓冲区;
        V(mutex);
        V(full); //增加一个产品
    }
}
consumer(){
    while(1){
        P(full); //消耗一个产品
        P(mutex);
        从缓冲区取出一个产品;
        V(mutex);
        V(empty); //增加一个空闲缓冲区
        使用产品;
    }
}
```

注意：实现互斥的 *P* 操作一定要放在实现同步的 *P* 操作之后。

易错点：实现互斥和实现同步的两个P操作的先后顺序。

题 1. 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专配车间的货架 F_1, F_2 上。 F_1 存放零件A， F_2 存放零件B， F_1 和 F_2 的容量均可存放10个零件。装配工人每次从货架上取一个零件A和一个零件B后组装成产品。请用P,V操作进行正确管理。

解析：本题是生产者-消费者问题的变体，生产者“车间A”和消费者“装配车间”共享缓冲区“货架 F_1 ”；生产者“车间B”和消费者“装配车间”共享缓冲区“货架 F_2 ”。因此，可为它们设置6个信号量： $empty1$ 对应货架 F_1 上的空闲空间，初值为10； $full1$ 对应货架 F_1 上面的A产品，初值为0； $empty2$ 对应货架 F_2 上的空闲空间，初值为10； $full2$ 对应货架 F_2 上面的B产品，初值为0； $mutex1$ 用于互斥地访问货架 F_1 ，初值为1； $mutex2$ 用于互斥地访问货架 F_2 ，初值为1。

A车间的工作过程可描述为：

```

while(1){
    生产一个产品 A;
    P(empty1);           //判断货架 F1 是否有空
    P(mutex1);           //互斥访问货架 F1
    将产品 A 存放到货架 F1 上;
    V(mutex1);           //释放货架 F1
    V(full1);           //货架 F1 上的零件 A 的个数加 1
}

```

B 车间的工作过程可描述为：

```
while(1){  
    生产一个产品 B;  
    P(empty2);          //判断货架 F2 是否有空  
    P(mutex2);          //互斥访问货架 F2  
    将产品 B 存放到货架 F2 上;  
    V(mutex2);          //释放货架 F2  
    V(full2);          //货架 F2 上的零件 B 的个数加 1  
}
```

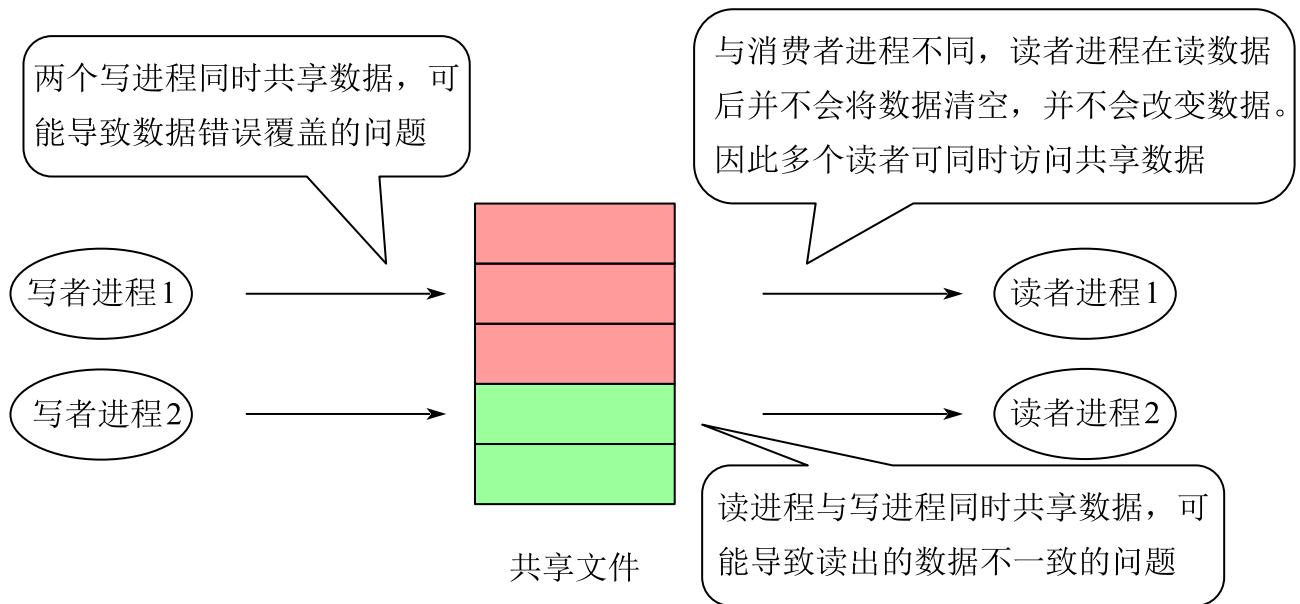
装配车间的工作过程可描述为：

```
while(1){  
    P(full1);          //判断货架 F1 是否有产品 A  
    P(mutex1);          //互斥访问货架 F1  
    从货架 F1 上取一个 A 产品;  
    V(mutex1);          //释放货架 F1  
    V(empty1);          //货架 F1 上的空闲空间数加 1  
    P(full2);          //判断货架 F2 是否有产品 B  
    P(mutex2);          //互斥访问货架 F2  
    从货架 F2 上取一个 B 产品;  
    V(mutex2);          //释放货架 F2  
    V(empty2);          //货架 F2 上的空闲空间数加 1  
    将取得的 A 产品和 B 产品组装成成品;  
}
```

2. 读者写者问题

1) 问题描述

有读者和写者两组并发进程，共享一个文件，当两个或两个以上的读进程同时访问共享数据时不会产生副作用，但若某个写进程和其他进程(读进程或写进程)同时访问共享数据时则可能导致数据不一致的错误。因此要求：①允许多个读者同时对文件执行读操作；②只允许一个写者往文件中写信息；③任一写者在



完成写操作之前不允许其他读者或写者工作；④写者执行写操作前，应让已有的读者和写者全部退出。

2) 问题分析

两类进程：写进程、读进程

互斥关系：写进程-写进程、写进程-读进程。读进程与读进程不存在互斥问题。

写者进程和任何进程都互斥，设置一个互斥信号量 rw ，在写者访问共享文件前后分别执行 P, V 操作。

读者进程和写者进程也要互斥，因此读者访问共享文件前后也要对 rw 执行 P, V 操作。

如果所有读者进程在访问共享文件之前都执行 $P(rw)$ 操作，那么会导致各个读进程之间也无法同时访问文件。

这是读者写者问题的核心思想：

$P(rw)$ 和 $V(rw)$ 其实就是对共享文件的“加锁”和“解锁”。既然各个读进程需要同时访问，而读进程与写进程又必须互斥访问，那么我们可以让第一个访问文件的读进程“加锁”，让最后一个访问完文件的读进程“解锁”。可以设置一个整数变量 $count$ 来记录当前有几个读进程在访问文件。

读者写者问题最核心的问题是如何处理多个读者可以同时对文件的读操作。

3) 如何实现

```
semaphore rw = 1;    //实现对文件的互斥访问
int count = 0;
semaphore mutex = 1;//实现对 count 变量的互斥访问
int i = 0;
writer(){
    while(1){
        P(rw); //写之前 “加锁”
        写文件;
        V(rw); //写之后 “解锁”
    }
}
reader (){
    while(1){
        P(mutex);      //各读进程互斥访问 count
        if(count==0) //第一个读进程负责 “加锁”
        {
            P(rw);
        }
    }
}
```

```

        count++;      //访问文件的进程数+1

        V(mutex);

        读文件;

        P(mutex);      //各读进程互斥访问 count

        count--;      //访问文件的进程数-1

        if(count==0)  //最后一个读进程负责“解锁”

        {

            V(rw);

        }

        V(mutex);

    }

}

```

问题：只要有源源不断的读进程存在，写进程就要一直阻塞等待，可能会造成“饿死”，在上述的算法中，读进程是优先的，那么应该怎么样来改造呢？

解决方法：新加入一个锁变量 w，用于实现“写优先”！

```

semaphore rw = 1;      //用于实现对文件的互斥访问

int count = 0;          //记录当前有几个读进程在访问文件

semaphore mutex = 1;    //用于保证对 count 变量的互斥访问

semaphore w=1;          //用于实现“写优先”

writer(){

while(1){

P(w);

P(rw);

写文件...;

V(rw);

V(w);

}

}

```

```
reader() {
    while(1) {
        P(w);
        P(mutex);
        if(count==0)
            P(rw);
        count++;
        V(mutex);
        V(w);
        读文件...;
        P(mutex);
        count--;
        if(count==0)
            V(rw);
        V(mutex);
    }
}
```

分析以下并发执行 $P(w)$ 的情况：

读者1 → 读者2

写者1 → 写者2

写者1 → 读者1

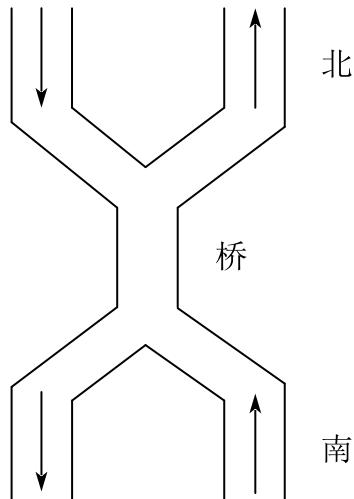
读者1 → 写者1 → 读者2

写者1 → 读者1 → 写者2

结论：在这种算法中，连续进入的多个读者可以同时读文件；写者和其他进程不能同时访问文件；写者不会饥饿，但也并不是真正的“写优先”，而是相对公平的先来先服务原则。有的书上把这种算法称为“读写公平法”。

这里我们来分析一下读者1→写者1→读者2这种情况。第一个读者1在进行到读文件操作的时候，有一个写者1操作，由于第一个读者1执行了 $V(w)$ ，所以写者1不会阻塞在 $P(w)$ ，但由于第一个读者1执行了 $P(rw)$ 但没有执行 $V(rw)$ ，写者1将被阻塞在 $P(rw)$ 上，这时候再有一个读者2，由于前面的写者1进程执行了 $P(w)$ 但没有执行 $V(w)$ ，所以读者2将被阻塞在 $P(w)$ 上，这样写者1和读者2都将被阻塞，只有当读者1结束时执行 $V(rw)$ ，此时写者1才能够继续执行直到执行 $V(w)$ ，读者2也将能够执行下去。

题 2. 有桥如下图所示，车流如箭头所示，桥上不允许两车交汇，但允许同方向多辆车依次通过(即桥上可以有多个同方向的车)。用 P 、 V 操作实现交通管理以防止桥上堵塞。



解析：这个题目要解决：南、北互斥（桥上不允许两车交汇，相当于“读、写互斥”），需要设置一个互斥信号量 $mutex$ ，初值为1；南、南共享（相当于“读、读共享”），套用实现“读、读共享”的模式，需要设置一个共享变量 $southcount$ ，用于记录当前桥上向南行驶过桥的车辆数目，初值为0，再设置一个互斥信号量 $smutex$ ，实现对 $southcount$ 的互斥访问；北、北共享（也相当于“读、读共享”），

套用实现“读、读共享”的模式，同理可得。

```
semaphore mutex = 1;// 作为桥的互斥访问信号量
semaphore smutex = 1;// 作为 southcount 的互斥访问信号量
semaphore nmutex = 1;// 作为 northcount 的互斥访问信号量
int southcount = 0;// 记录南方向的车辆的数量
int northcount = 0;// 记录北方向的车辆的数量

void south()
{
    while(true)
    {
        wait(smutex);
        if(southcount == 0)
            wait(mutex);
        southcount++;
        signal(smutex);
        // 南方车辆通过
        wait(smutex);
        southcount--;
        if(southcount == 0)
            signal(mutex);
        signal(smutex);
    }
}

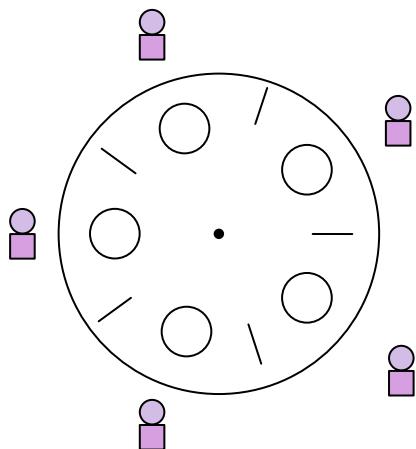
void north()
{
    while(true)
    {
        wait(nmutex);
        ... // 北方车辆通过
        signal(nmutex);
    }
}
```

```
if(northcount == 0)  
    wait(mutex);  
    northcount++;  
    signal(nmutex);  
    // 北方车辆通过  
    wait(nmutex);  
    northcount--;  
    if(northcount == 0)  
        signal(mutex);  
        signal(nmutex);  
    }  
}
```

3. 哲学家进餐问题

1) 问题分析

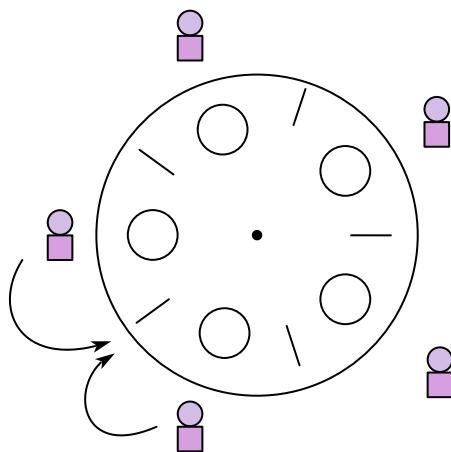
有五个哲学家，他们的生活方式是交替地进行思考和进餐。他们共用一张圆桌，分别坐在五张椅子上。在圆桌上有五个碗和五支筷子，平时一个哲学家进行思考，饥饿时便试图取用其左、右最靠近他的筷子，只有在他拿到两支筷子时才能进餐。进餐完毕，放下筷子又继续思考。



我们可以从上面的题目中得出，筷子是临界资源，同一根筷子同一时刻只能有一个哲学家可以拿到。

2) 问题分析

由问题描述我们可以知道，一共有五个哲学家，也就是五个进程；五只筷子，也就是五个临界资源；因为哲学家想要进餐，必须要同时获得左边和右边的筷子，这就是要同时进入两个临界区（使用临界资源），才可以进餐。



哲学家进餐问题可看作是并发进程并发执行时处理共享资源的一个有代表性的问题。

```
Var chopstick: array[0,...,4] of semaphore= 1;  
/*5 支筷子分别设置为初始值为 1 的互斥信号量*/
```

第 i 个哲学家的活动

```
repeat  
wait(chopstick[i]);  
wait(chopstick[(i+1) mod 5]);  
eat;  
signal(chopstick[i]);
```

```
signal(chopstick[(i+1) mod 5]);  
think;  
until false;
```

此算法可以保证不会有相邻的两位哲学家同时进餐。

若五位哲学家同时饥饿而各自拿起了左边的筷子，这使五个信号量 *chopstick* 均为0，当他们试图去拿起右边的筷子时，都将因无筷子而无限期地等待下去，即可能会引起死锁。

哲学家进餐问题的改进解法

方法 1：至多只允许四位哲学家同时去拿左筷子，最终能保证至少有一位哲学家能进餐，并在用完后释放两只筷子供他人使用。

设置一个初值为4的信号量 *r*，只允许4个哲学家同时去拿左筷子，这样就能保证至少有一个哲学家可以就餐，不会出现饿死和死锁的现象。

原理：至多只允许四个哲学家同时进餐，以保证至少有一个哲学家能够进餐，最终总会释放出他所使用过的两支筷子，从而可使更多的哲学家进餐。

```
semaphore chopstick[5]={1,1,1,1,1};  
semaphore r=4;  
void philosopher(int i)  
{  
    while(true){  
        think();  
        wait(r);  
        wait(chopstick[i]);  
        wait(chopstick[(i+1) mod 5]);  
        eat();  
        signal(chopstick[(i+1) mod 5]);  
        signal(chopstick[i]);  
    }  
}
```

```
signal(r);  
think();  
}  
}
```

方法 2：仅当哲学家的左右手筷子都拿起时才允许进餐。

解法：利用信号量的保护机制实现。

原理：通过互斥信号量 *mutex* 对 *eat()* 之前取左侧和右侧筷子的操作进行保护，可以防止死锁的出现。

```
semaphore mutex=1;  
semaphore chopstick[5]={1,1,1,1,1};  
void philosopher(int i)  
{  
    while(true){  
        think();  
        wait(mutex);  
        wait(chopstick[i]);  
        wait(chopstick[(i+1) mod 5]);  
        signal(mutex);  
        eat();  
        signal(chopstick[(i+1) mod 5]);  
        signal(chopstick[i]);  
        think();  
    }  
}
```

方法 3：规定奇数号哲学家先拿左筷子再拿右筷子，而偶数号哲学家相反。

原理：按照下图，将是 2, 3 号哲学家竞争 3 号筷子，4, 5 号哲学家竞争 5 号筷子。1 号哲学家不需要竞争。最后总会有一个哲学家能获得两支筷子而进餐。

```
semaphore chopstick[5]={1,1,1,1,1};  
void philosopher(int i)
```

```
{  
while(true){  
if(i mod 2==0)  
{  
wait(chopstick[(i+1) mod 5]);  
wait(chopstick[i]);  
eat();  
signal(chopstick[i]);  
signal(chopstick[(i+1) mod 5]);  
}  
else  
{  
wait(chopstick[i]);  
wait(chopstick[(i+1) mod 5]);  
eat();  
signal(chopstick[(i+1) mod 5]);  
signal(chopstick[i]);  
}  
}  
}
```

题 1. 有 $n(n \geq 3)$ 名哲学家围坐在一张圆桌边，每名哲学家交替地就餐和思考。在圆桌中心有 $m(m \geq 1)$ 个碗，每两名哲学家之间有一根筷子。每名哲学家必须取到一个碗和两侧地筷子后，才能就餐，进餐完毕，将碗和筷子放回原位，并继续思考。为使尽可能多的哲学家同时就餐，且防止出现死锁现象，请使用信号量的 P, V 操作 [$\text{wait}()$, $\text{signal}()$ 操作] 描述上述过程中的互斥与同步，并说明所用信号量及初值的含义。

解析：回顾传统的哲学家问题，假设餐桌上有 n 名哲学家、 n 根筷子，那么可以用这种方法避免死锁：限制至多允许 $n - 1$ 名哲学家同时“抢”筷子，那么至少

会有1名哲学家可以获得两根筷子并顺利进餐，于是不可能发生死锁情况。

本题可以用碗这个限制资源来避免死锁；当碗的数量 m 小于哲学家的数量 n 时，可以直接让碗的资源量等于 m ，确保不会出现所有哲学家都拿一侧筷子而无限等待另一侧筷子进而造成死锁的情况；当碗的数量 m 大于等于哲学家的数量 n 时，为了让碗起到同样的限制效果，我们让碗的资源量等于 $n - 1$ ，这样就能保证最多只有 $n - 1$ 名哲学家同时进餐，所以得到碗的资源量为 $\min\{n - 1, m\}$ 。在进行 PV 操作时，碗的资源量起限制哲学家取筷子的作用，所以需要先对碗的资源量进行 P 操作。具体过程如下：

```
semaphore bowl;
semaphore chopstick[n];
for(int i=0;i<n;i++)
chopstick[i]=1;
bow1=min(n-1,m);

cobegin
while(TRUE){
思考;
P(bowl);
P(chopstick[i]);
P(chopstick[(1+i)%n]);
就餐;
V(chopstick[i]);
V(chopstick[(1+i)%n]);
V(bowl);
}
coend
```

课时九 练习题

1. 有一个计数信号量 S :

- 1) 假如若干进程对 S 进行 28 次 P 操作和 18 次 V 操作后，信号量 S 的值为 0。
 - 2) 假如若干进程对信号量 S 进行了 15 次 P 操作和 2 次 V 操作。请问此时有多少个进程等待在信号量 S 的队列中？()
- A. 2 B. 3*
- C. 5 D. 7*

2. 有两个并发进程 P_1 和 P_2 ，其程序代码如下：

```

P1(){
x=1; //A1
y=2;
z=x+y;
print z; //A2
}

P2(){
x=-3; //B1
c=x*x;
print c; //B2
}

```

可能打印出的 z 值有()，可能打印出的 c 值有()，(其中 x 为 P_1, P_2 的共享变量)。

- | | |
|---|--|
| <i>A. $z = 1, -3; c = -1, 9$</i> | <i>B. $z = -1, 3; c = 1, 9$</i> |
| <i>C. $z = -1, 3, 1; c = 9$</i> | <i>D. $z = 3; c = 1, 9$</i> |

3. 在一个仓库中可以存放 A 和 B 两种产品，要求：

①每次只能存入一种产品

② A 产品数量- B 产品数量 $< M$

③ B 产品数量- A 产品数量 $< N$

其中， M, N 是正整数，试用 P 操作、 V 操作描述产品 A 与产品 B 的入库过程。

4. 设自行车生产线上有一个箱子，其中有 N 个位置($N \geq 3$)，每个位置可存放一个车架或一个车轮；又设有3名工人，其活动分别为：

工人1活动：

do{

 加工一个车架；

 车架放入箱中；

}while(1)

工人2活动：

do{

 加工一个车轮；

 车轮放入箱中；

}while(1)

工人3活动

do{

 箱中取一个车架；

 箱中取两个车轮；

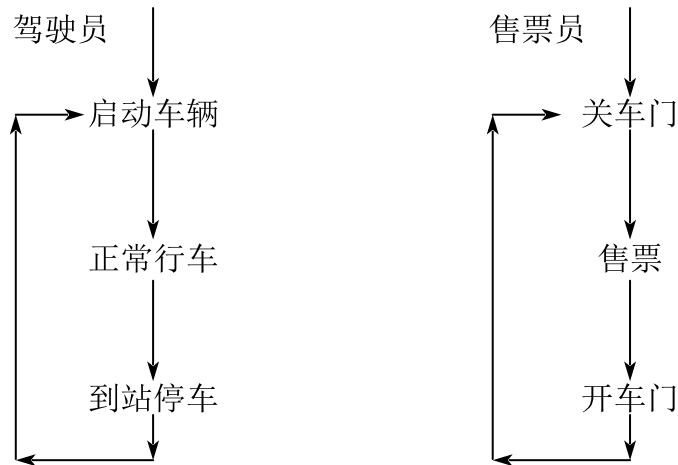
 组装为一台车；

}while(1)

试分别用信号量与 PV 操作实现三名工人的合作，要求解中不含死锁。

5. 设公共汽车上驾驶员和售票员的活动分别如下图所示。驾驶员的活动：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、行驶的过程中，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。



课时十 内存管理（一）

考点	重要程度	占分	题型
1. 内存管理的基本原理和要求	★	1~2	选择
2. 覆盖与交换	★★	1~2	选择

1. 内存管理的基本原理和要求

1) 内存管理的定义

操作系统对内存的划分和动态分配就是内存管理的概念。

2) 内存管理的功能

内存空间的分配和回收：由操作系统完成对主存的分配和回收，对编程人员透明。

地址转换：使逻辑地址转换为真实的物理地址。

内存空间的扩充：利用虚拟存储技术或自动覆盖技术，从逻辑上扩充内存。

存储保护：保证各道作业在各自的存储空间内运行，互不干扰。

3) 程序的装入和链接

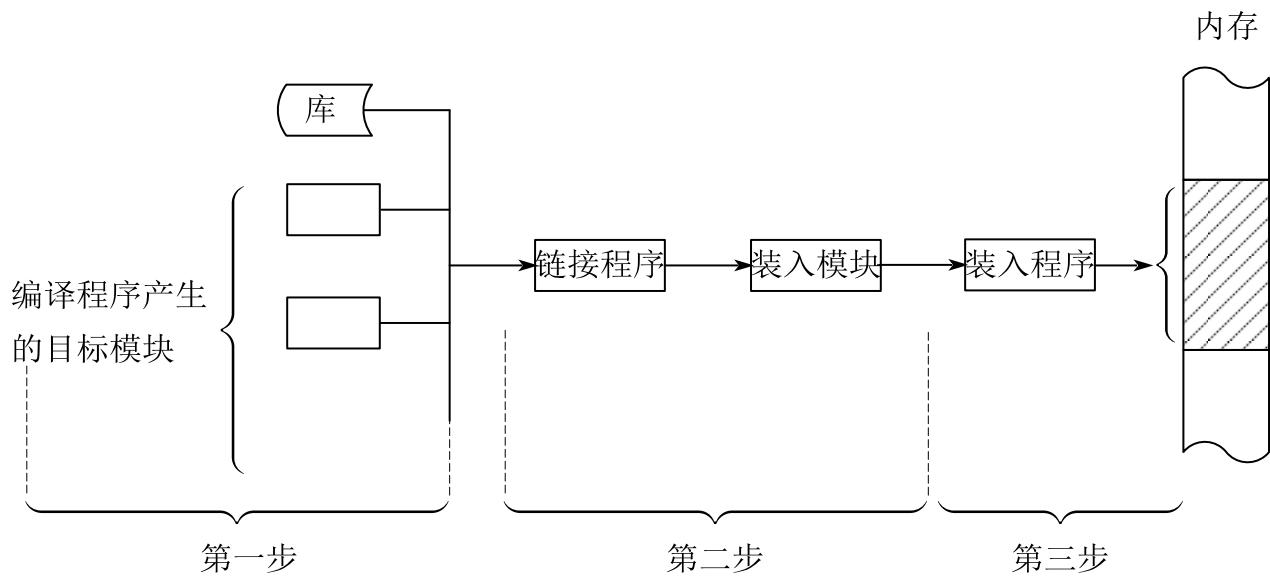
创建进程首先要将程序和数据装入内存，将用户源程序变为可在内存中执行的程序。

①步骤描述：

编译：由编译程序将用户源代码编译为若干目标模块。

链接：由链接程序将编译后的一组目标模块和所需的库函数链接在一起，形成一个完整的装入模块。

装入：由装入程序将装入模块装入内存运行。



②程序链接的方式

静态链接：在程序运行之前，先将各目标模块及他们所需的库函数链接成一个可执行程序，此后不再拆开。

装入时动态链接： 编译后所得的一组目标模块在装入时，边装入边链接。

运行时动态链接： 对某些目标模块的链接，是在程序执行中需要该目标模块时才进行的。便于修改和更新以及实现对目标模块的共享。

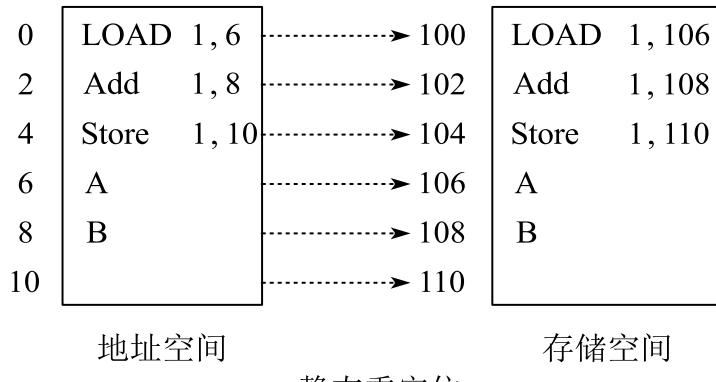
③装入内存的方式

绝对装入： 在编译时即知道程序将装入的内存具体地址，则编译程序将产生绝对地址的目标代码。而后将程序和数据装入内存，只适用于单道程序环境。

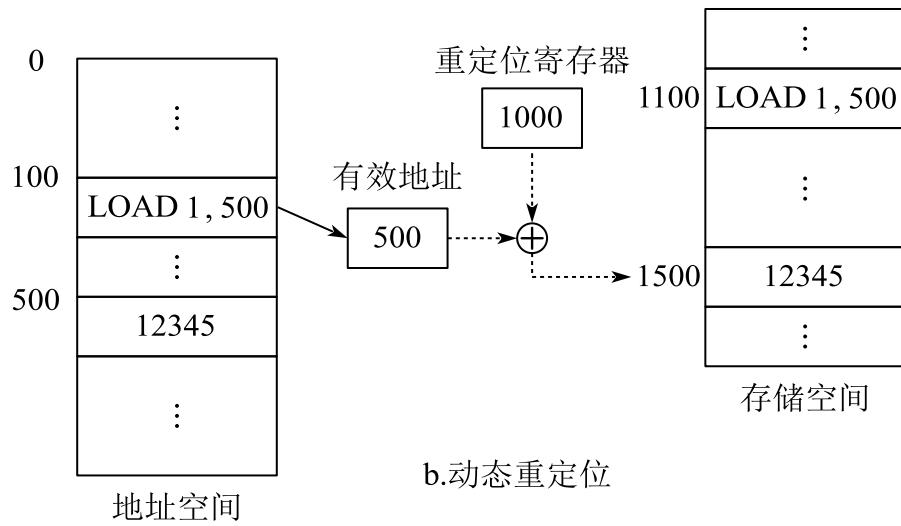
可重定位装入： 在多道程序环境中，多个目标模块的起始地址均为0开始。装入内存时，通过所分配的内存起始地址加上程序内的相对地址进行地址的动态变换，一次完成，又称为静态重定位。（必须一次性全部装入，运行期间不能动态扩充和移动）。

动态运行时装入： 装入模块装入内存后并不立即进行地址转换，而是等到程序执行时才进行。需要重定位寄存器的支持（可以将程序分到不连续的存储区中，程序运行前只需要装入部分代码，运行期间根据需要动态分配内存，便于程序段的

共享，可以向用户提供一个比存储空间大得多的地址空间)。



a.静态重定位



b.动态重定位

题 1. 在虚拟内存管理中，地址变换机构将逻辑地址变换为物理地址，形成逻辑地址的阶段是（ ）

- A. 编辑 B. 编译 C. 链接 D. 装载

答案：C

④逻辑地址空间与物理地址空间

逻辑地址空间：程序编译后的每个模块都是以0开始编址，称为目标模块的逻辑地址。当链接为一个完整的可执行目标程序时，链接程序将按顺序以0开始编址，构造统一的逻辑地址空间。

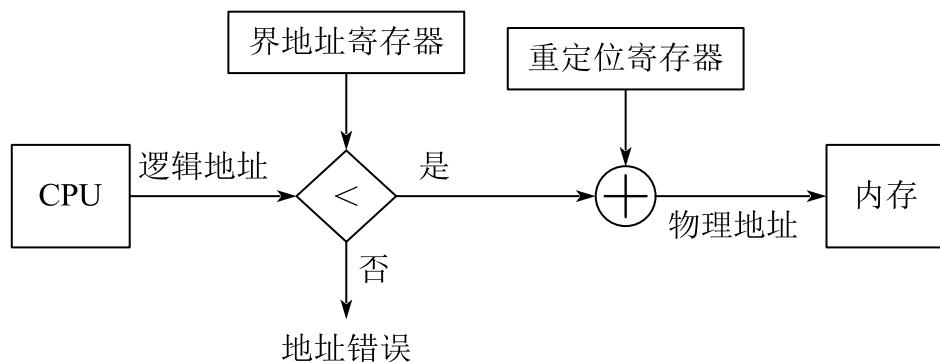
物理地址空间：内存中物理单元的集合，是地址转换的最终地址，当装入程序将可执行代码装入内存中时，必须将逻辑地址转化为物理地址。

⑤内存保护

内存分配前，需要保护操作系统不受用户进程的影响，同时保护用户进程不受其他用户进程的影响。

内存保护的两种方式：

- 在CPU中设立一对上、下限寄存器，存放用户作业在主存中的上下限地址，每当CPU要访问一个地址时，先根据这对上下限寄存器判断是否越界访问。
- 采用重定位寄存器和限长寄存器来实现这种保护。重定位寄存器保存最小的物理地址，限长寄存器保存逻辑地址的最大值。进行内存访问时，先判断逻辑地址是否大于限长寄存器值，若未越界，则加上重定位寄存器的值映射成物理地址，再进行内存访问。



题 2. 下面关于存储管理的叙述中，正确的是（ ）

- A. 存储保护的目的是限制内存的分配
- B. 在内存为M、有N个用户的分时系统中，每个用户占用 M/N 的内存空间
- C. 在虚拟内存系统中，只要磁盘空间无限大，作业就能拥有任意大的编址空间
- D. 实现虚拟内存管理必须有相应硬件的支持

答案：D

解析：选项A、B显然错误，选项C中编址空间的大小取决于硬件的访存能力，一般由地址总线宽度决定。选项D中虚拟内存的管理需要由相关的硬件和软件

支持，有请求分页页表机制、缺页中断机构、地址变换机构等。

题 3. 在使用交换技术时，若一个进程正在()，则不能交换出主存。

- A. 创建
- B. I/O 操作
- C. 处于临界段
- D. 死锁

答案：B

解析：进程正在进行 *I/O* 操作时不能换出主存，否则其 *I/O* 数据区将被新换入的进程占用，导致错误。不过可以在操作系统中开辟 *I/O* 缓冲区，将数据从外设输入或将数据输出到外设的 *I/O* 活动在系统缓冲区中进行，这时系统缓冲区与外设 *I/O* 时，进程交换不受限制。

2. 覆盖与交换

覆盖与交换技术是在多道程序环境下扩充内存的两种方法。

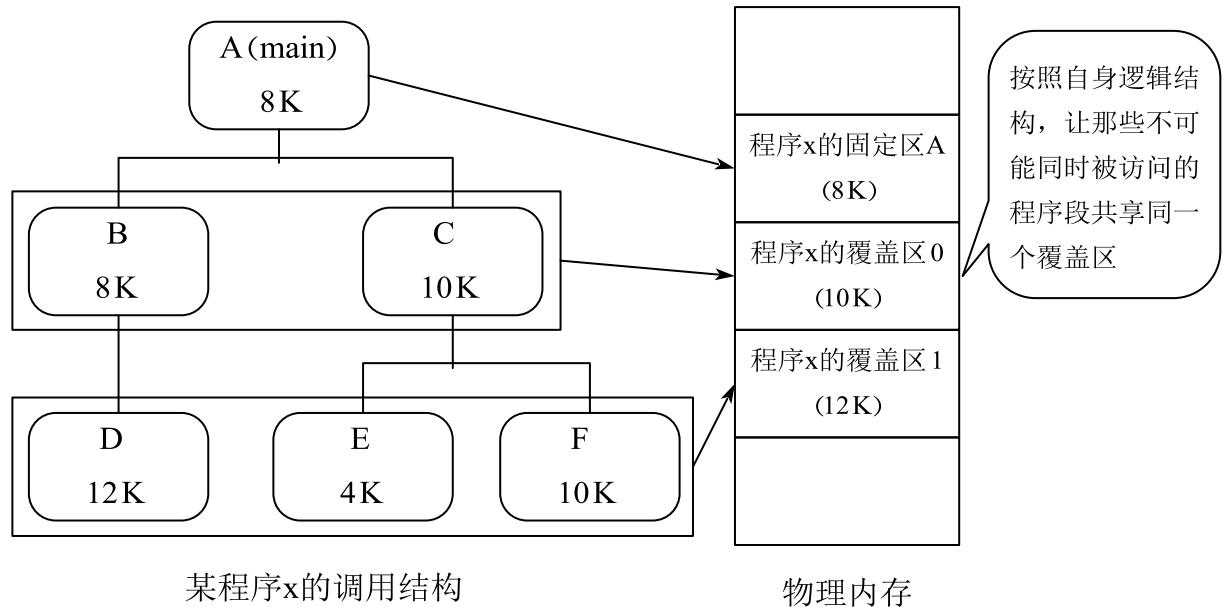
1) 覆盖

早期系统主存小，仅存放一道用户程序，但存储空间放不下用户进程也时有发生，这一问题可以通过覆盖技术解决。

覆盖技术的思想：将程序分为多个段（多个模块）。常用的段常驻内存，不常用的段在需要时调入内存。

把内存划分为一个固定区和若干个覆盖区，固定区存放用户程序经常活跃的部分，调入后就不再调出（除非运行结束），其余部分按调用关系分段，将即将访问的段放在覆盖区，需要用到时调入内存，用不到时调出内存。其他放在外存，在需要调用前，系统将其调入覆盖区，替换原有的段。

必须由程序员声明覆盖结构，操作系统完成自动覆盖。



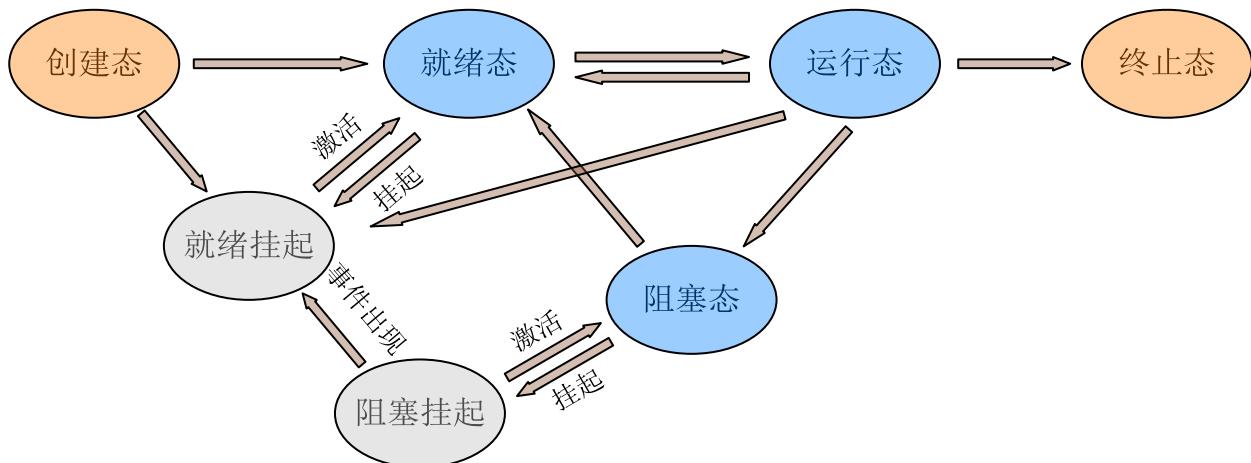
缺点：对用户不透明，增加了用户编程负担，覆盖技术只用于早期的操作系统中。

2) 交换

基本思想：把处于等待状态的进程或者被CPU剥夺运行权限的进程从内存移出到辅存，这一过程称为换出；把准备好竞争CPU的进程从辅存移到内存，这一过程称为换入。

暂时换出外存等待的进程状态为挂起状态（挂起态）。

挂起态又可以进一步细分为就绪挂起、阻塞挂起两种状态。



①具有对换功能的操作系统中，通常把磁盘空间分为文件区和对换区两部分，文件区主要用于存放文件，主要追求存储空间的利用率，因此对文件区空间的管理采用离散分配方式：对换区空间只占磁盘空间的小部分，被换出的进程数据就存放在对换区，由于对换的速度直接影响到系统的整体速度，因此对换区空间的管理主要是求换入换出速度，因此通常对换区采用连续分配方式。总之，对接 I/O 速度比文件区的更快。

②交换通常在许多进程运行且内存吃紧时运行，而系统负有降低就暂停。例如：在发现许多进程运行时经常发生缺页，就说明内存紧张，此时可以换出一些进程，如果缺页明显下降，就可以暂停换出。

③可优先换出阻塞进程：可换出优先级低的进程：为了防止优先级低的进程在被调入内存后很快又被换出，有的系统还会考虑进程在内存的驻留时间。

(注意， PCB 会常驻内存，不会被换出外存)

课时十 练习题

1. 在存储管理中，采用覆盖与交换技术的目的是()

- A. 节省主存空间
- B. 物理上扩充主存容量
- C. 提高CPU效率
- D. 实现共享主存

2. 分区分配内存管理方式的主要保护措施是()

- A. 界地址保护
- B. 程序代码保护
- C. 数据保护
- D. 栈保护

3. 内存保护需要由()完成，以保证进程空间不被非法访问。

- A. 操作系统
- B. 硬件机构
- C. 操作系统和硬件机构合作
- D. 操作系统或者硬件机构独立完成

4. 当编程人员编写好的程序经过编译转换成目标文件后，各条指令的地址编号起始一般定位()，称为()地址。

- 1) A. 1 B. 0 C. IP D. CS
- 2) A. 绝对 B. 名义 C. 逻辑 D. 实

5. 在虚拟内存管理中，地址变换机构将会把逻辑地址转变为物理地址，而形成该逻辑地址的阶段是（ ）
- A. 程序装载时
 - B. 程序链接时
 - C. 程序编译时
 - D. 源程序编辑时
6. 动态重定位是在程序的（ ）过程中进行的。
- A. 链接
 - B. 装入
 - C. 执行
 - D. 编译
7. 在使用对换技术时，如下最适于将对应进程换出到外存的情况是当该进程正（ ）时。
- A. 处于临界区
 - B. 进行复杂计算
 - C. 创建
 - D. 进行 I/O 操作
8. 在内存管理中，采用覆盖与对换技术的根本目的在于（ ）。
- A. 实现主存共享
 - B. 物理上扩充主存容量
 - C. 提高 CPU 效率
 - D. 节省主存空间

课时十一 内存管理（二）

考点	重要程度	占分	题型
1. 连续分配管理方式	★	2~3	选择
2. 动态分区分配算法	★★	3~5	选择、应用

1. 连续分配管理

1) 单一连续分配

在单一连续分配方式中，内存被分为：系统区和用户区。

系统区通常位于内存的低地址部分，用于存放操作系统系统区，用户区用于存放用户进程相关数据。

特点：内存中只能有一道用户程序，用户程序独占整个用户区空间。

优点：实现简单；无外部碎片；可以采用覆盖技术扩充内存；不一定需要采取内存保护(如：早期的PC操作系统MS-DOS)。

缺点：只能用于单用户、单任务的操作系统中；有内部碎片；存储器利用率极低。

题 1. 连续存储分配时，存储单元的地址（ ）。

- A. 一定连续
- B. 一定不连续
- C. 不一定连续
- D. 部分连续，部分不连续

答案：A

2) 固定分区分配

固定分区分配是最简单的一种多道程序存储管理方式，它将用户内存空间划分为若干个固定大小的区域，每个分区只装入一道作业。

两种方法：

①分区大小相等：缺乏灵活性，只适合用于用一台计算机控制多个相同对象的场

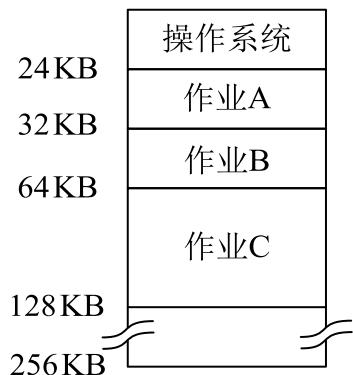
合。

②分区大小不等：增加了灵活性，可以满足不同大小的进程需求。

为便于内存分配，通常将分区按大小排队，建立一张分区说明表。

分区号	大小/KB	起址/KB	状态
1	12	20	已分配
2	32	32	已分配
3	64	64	已分配
4	128	128	已分配

a. 分区说明表



b. 存储空间分配情况

优点：实现简单，无外部碎片。

缺点：

- a. 当用户程序太大时，可能所有的分区都不能满足需求，此时不得不采用覆盖技术来解决，但又会降低性能；
- b. 会产生内部碎片，内存利用率低。

题 1. 分区分配内存管理方式的主要保护措施是()

- A. 界地址保护
- B. 程序代码保护
- C. 数据保护
- D. 栈保护

答案: A

解析: 分区分配存储管理方式的保护措施是设置界地址寄存器。每个进程都有自己独立的进程空间,如果一个进程在运行时所产生的地址在其地址空间之外,则发生地址越界。当程序要访问某个内存单元时,由硬件检查是否允许,如果允许则执行,否则产生地址越界中断,由OS进行相应处理

题 2. 分区管理要求对每个作业都分配()的内存单元。

- A. 地址连续
- B. 若干地址不连续
- C. 若干连续的帧
- D. 若干不连续的帧

答案: A

解析: 分区分配方式属于连续分配管理方式,分为固定分区分配和动态分配,这两个分区分配方式都会为每个作业分配地址连续的内存单元。

2. 动态分区分配

动态分区分配又称可变分区分配,是一种动态划分内存的分区方法。这种分配方式不会预先划分内存分区,而是在进程装入内存时,根据进程的大小动态地建立分区。系统分区的大小和数目是可变的。

两种常用的数据结构: 空闲分区表和空闲分区链

①空闲分区表: 每个空闲分区对应一个表项。表项中包含分区号、分区大小、分区起始地址等信息。

②空闲分区链: 每个分区的起始部分和末尾部分分别设置前向指针和后向指针。起始部分处还可记录分区大小等信息。

下面介绍四种动态分区分配算法

- 1) 首次适应 (*First Fit*) 算法。空闲分区以地址递增的次序链接。分配内存时顺序查找，找到大小能满足要求的第一个空闲分区。
- 2) 最佳适应 (*Best Fit*) 算法。空闲分区按容量递增的方式形成功区链，找到第一个能满足要求的空闲分区。
缺点：会产生很多的外部碎片。
- 3) 最坏适应 (*Worst Fit*) 算法。又称最大适应 (*Largest Fit*) 算法，空闲分区以容量递减的次序链接，找到第一个能满足要求的空闲分区，即挑选出最大的分区。
缺点：不利于后续大进程使用，会造成后续大进程到达时无内存分区可用的问题。
- 4) 邻近适应 (*Next Fit*) 算法。又称循环首次适应算法，由首次适应算法演变而成。不同之处是，分配内存时从上次查找结束的位置开始继续查找。

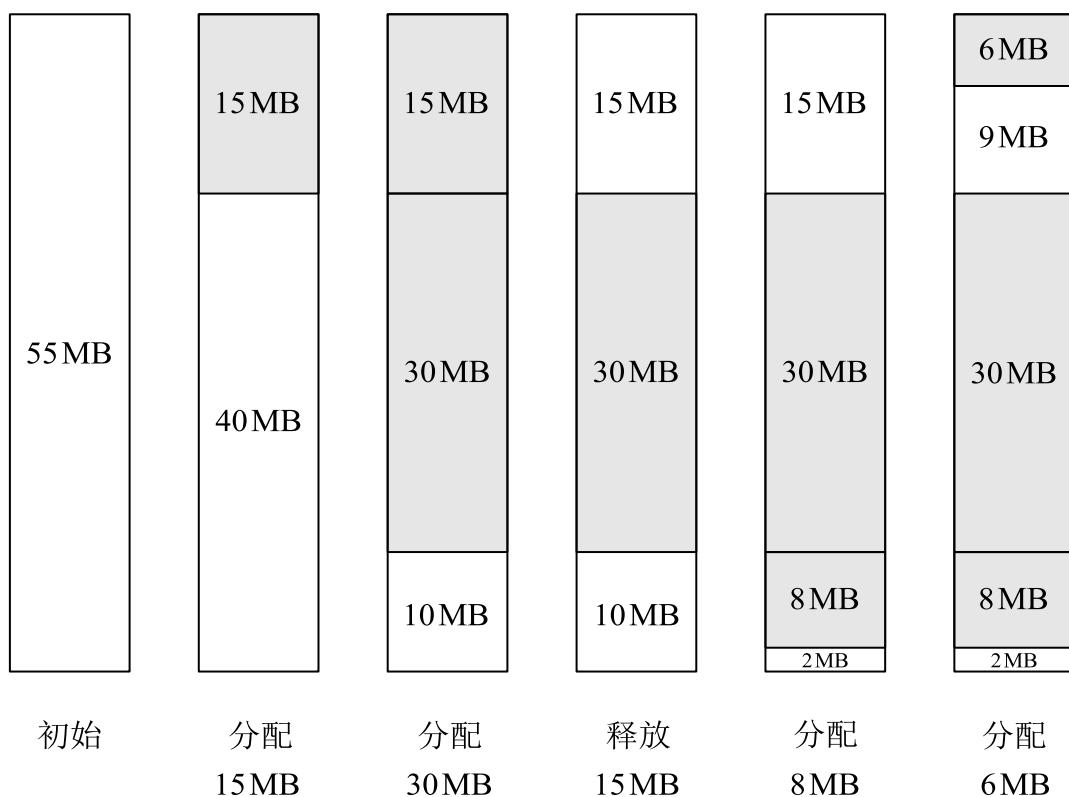
算法	算法思想	分区排列顺序	优点	缺点
首次适应	从头到尾找适合的分区	空闲分区以地址递增次序排列	综合看性能最好。算法开销小，回收分区后一般不需要对空闲分区队列重新排序	
最佳适应	优先使用更小的分区，以保留更多大分区	空闲分区以容量递增次序排列	会有更多的大分区被保留下来，更能满足大进程需求	会产生很多太小的、难以利用的碎片；算法开销大，回收分区后可能需要对空闲分区队列重新排序
最坏适应	优先使用更大的分区，以防止产生太小的不可用的碎片	空闲分区以容量递减次序排列	可以减少难以利用的小碎片	大分区容易被用完，不利于大进程；算法开销大(原因同上)
临近适应	由首次适应演变而来，每次从上次查找结束位置开始查找	空闲分区以地址递增次序排列(可排列成循环链表)	不用每次都从低地址小分区开始检索。算法开销小(原因同首次适应算法)	会使高地址的大分区也被用完

题 1. 某基于动态分区存储管理的计算机，其主存容量为55MB（初始为空），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（ ）。

- A. 7MB
- B. 9MB
- C. 10MB
- D. 15MB

答案：B

解析：最佳适配算法是指每次为作业分配内存空间时，总是找到能满足空间大小需要的最小空闲分区给作业，可以产生最小的内存空间分区。下图显示了这个过程的主存空间变化。



图中，灰色部分为分配出去的空间，白色部分为空闲区。这样，容易发现，此时主存中最大空闲分区大小为9MB。

课时十一 练习题

1. 设内存的分配情况如右图所示。若要申请一块 $40KB$ 的内存空间，采用最佳适应算法，则所得到的分区首址为()。

- A. $100K$
- B. $190K$
- C. $330K$
- D. $410K$



2. 分区管理中采用最佳适应分配算法时，把空闲区按()次序登记在空闲区表中。

- A. 长度递增
- B. 长度递减
- C. 地址递增
- D. 地址递减

3. 首次适应算法的空闲分区()

- A. 按大小递减顺序连在一起
- B. 按大小递增顺序连在一起
- C. 按地址由小到大排列
- D. 按地址由大到小排列

4. 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。

分区始址	$20K$	$500K$	$1000K$	$200K$
分区大小	$40KB$	$80KB$	$100KB$	$200KB$

回收始址为 $60K$ 、大小为 $140K$ 的分区时，系统中空闲分区的数量、空闲分区链第一个分区的始址和大小分别是()。

A. 320K, 380KB

B. 3500K, 80KB

C. 420K, 180KB

D. 4500K, 80KB

5. 采用动态分区算法回收内存时，如果回收分区仅与空闲分区链插入点的前一个分区相邻接，那么需要在空闲分区表中（ ）。

A. 增加一个新表项

B. 修改前一个分区表项的大小

C. 修改前一个分区表项的起始地址

D. 修改前一个分区表项的大小和起始地址

6. 可重定位内存的分区分配目的是（ ）。

A. 解决碎片问题

B. 便于多作业共享内存

C. 便于用户干预

D. 便于回收空白分区

7. 采用分页存储管理方式进行存储分配时产生的存储碎片，被称为（ ）。

A. 外部碎片

B. 内部碎片

C. 外部碎片或内部碎片

D. A,B,C都正确

8. 某系统的空闲分区见下表，采用可变式分区管理策略，现有如下作业序列：

96KB, 20KB, 200KB。若用首次适应算法和最佳适应算法来处理这些作业序列，则哪种算法能满足该作业序列请求？为什么？

分区号	大小	始址
1	32KB	100K
2	10KB	150K
3	5KB	200K
4	218KB	316K
5	96KB	530K

课时十二 内存管理（三）

考点	重要程度	占分	题型
1. 分页存储管理方式	必考	3~5	选择、应用
2. 分段存储管理方式	★★	2~4	选择、应用
3. 段页式管理方式	必考	3~6	选择、应用

非连续分配管理方式根据分区的大小是否固定，分为分页存储管理方式、分段存储管理方式和段页式存储管理方式。

1. 分页存储管理方式

分页存储管理方式又分为基本分页存储管理方式和请求分页存储管理方式。

1) 基本分页存储管理方式

算法思想：把内存分为一个个相等的小分区，再按照分区大小把进程拆分成一个个小部分，分页管理不会产生外部碎片。

分页存储管理的基本概念：

①页面和页面大小

进程中的块称为页 (*Page*)，内存中的块称为页框 (*Page Frame*，或页帧)。外存也以同样的单位进行划分，直接称为块 (*Block*)。进程在执行时需要申请主存空间，即要为每个页面分配主存中的可用页框，这就产生了页和页框的一一对应。页面大小应该适中，应为2的整数幂。

②地址结构

31...12	11...0
页号 P	页内偏移量 W

分页存储管理的逻辑地址结构

前一部分为页号 P ，后一部分为页内偏移量 W 。地址长度为32位，其中0~11位为页内地址，即每页大小为4KB；12~31位为页号，地址空间最多允许 2^{20} 页。

③页表

为了便于在内存中找到进程每个页面所对应的物理块，系统为每个进程建立一张页表，它记录页面在内存中对应的物理块号，页表一般存放在内存中。

页表是由页表项组成的，页表项由页号和物理内存中的块号组成。

题 1. 假设某系统物理内存大小为 $4GB$ ，页面大小为 $4KB$ ，则每个页表项至少应该为多少字节？

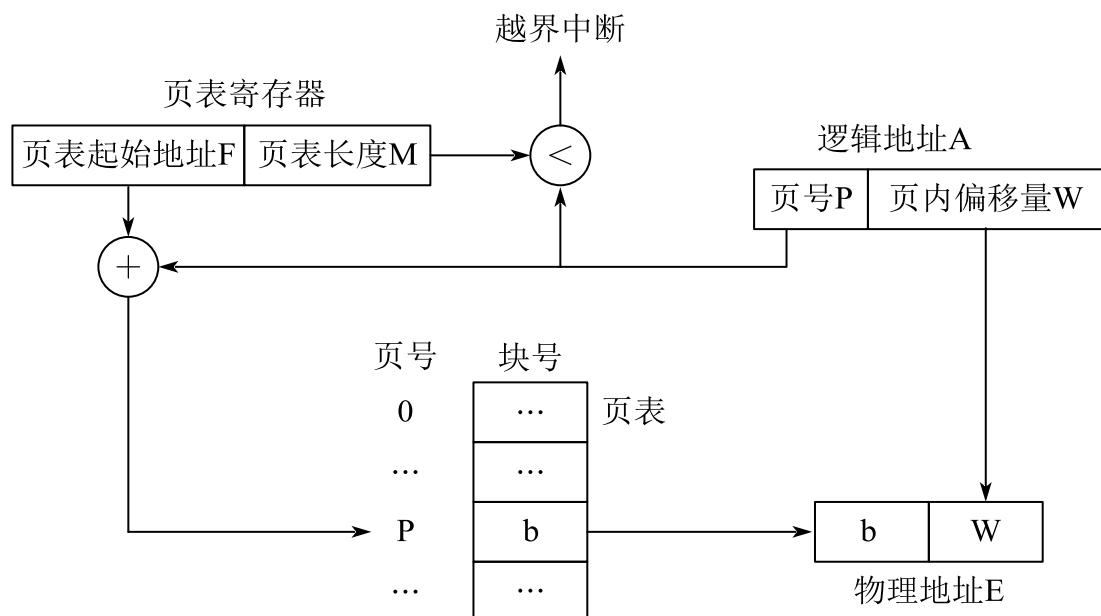
解析： $4GB = 2^{32} B$ ， $4KB = 2^{12} B$

因此 $4GB$ 的内存总共会被分为 $2^{32}/2^{12} = 2^{20}$ 个内存块，因此内存块号的范围应该是 $0 \sim 2^{20} - 1$ ，因此至少要 20 个二进制位才能表示这么多的内存块号，至少要 3 个字节才够(每个字节 8 个二进制位，3 个字节共 24 个二进制位)。

④地址变换机构

a. 基本的地址变换结构

地址变换机构的任务是将逻辑地址转换为内存中的物理地址。地址变换是借助于页表实现的。



分页存储管理系统中的地址变换机构

变换过程计算：

设页面大小为 L ，逻辑地址 A 到物理地址 E 的变换过程如下(逻辑地址、页号、每页的长度都是十进制数)：

- I . 计算页号 $P(P = A/L)$ 和页内偏移量 $W(W = A \% L)$ 。
- II. 比较页号 P 和页表长度 M ，若 $P > M$ ，则产生越界中断，否则继续执行。
- III. 页表中页号 P 对应的页表项地址 = 页表始址 $F + \text{页号 } P \times \text{页表项长度}$ ，取出该页表项内容 b ，即为物理块号。要注意区分页表长度和页表项长度。页表长度的值是指一共有多少页，页表项长度是指页地址占多大的存储空间。
- IV. 计算 $E = b \times L + W$ ，用得到的物理地址上 E 去访问内存。

题 1. 若页面大小 L 为 $1KB$ ，页号 2 对应的物理块为 $b = 8$ ，计算逻辑地址 $A = 2500$

的物理地址 E

解析：过程如下： $P = 2500/1K = 2$ ， $W = 2500 \% 1K = 452$ ，查找得到页号 2 对应的物理块的块号为 8 ， $E = 8 \times 1024 + 452 = 8644$ 。

注意区分题目中给出十进制和二进制的计算区别。

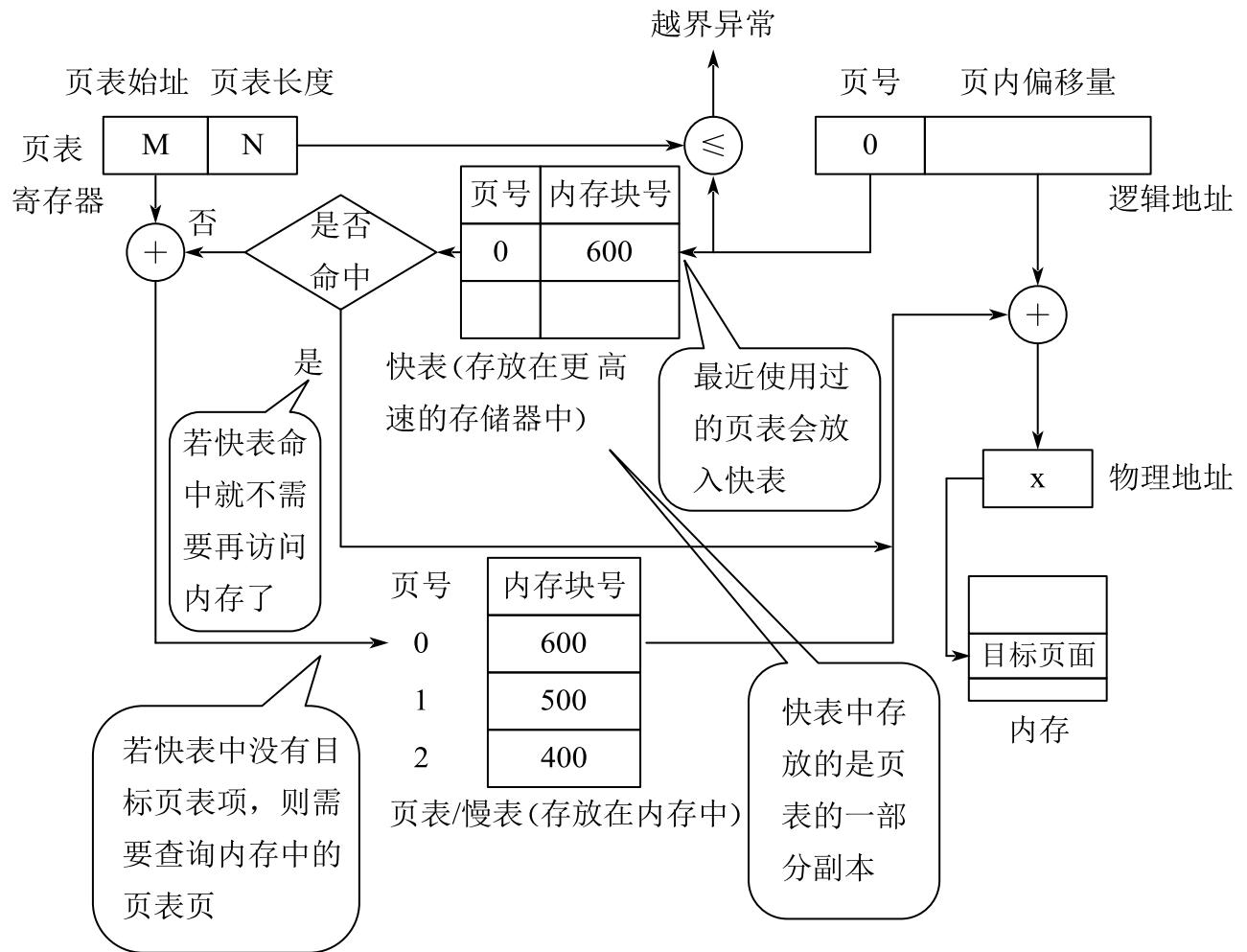
分页管理方式的两个主要问题：

- I . 每次访存操作都需要进行逻辑地址到物理地址的转换，地址转换过程必须足够快，否则访存速度会降低；
- II. 每个进程引入页表，用于存储映射机制，页表不能太大，否则内存利用率会降低。

b. 具有快表的地址变换机构

快表，又称联想寄存器(TLB)，是一种访问速度比内存快很多的高速缓冲存储器，用来存放当前访问的若干页表项，以加速地址变换的过程。与此对应，内存中的页表常称为慢表。

引入快表后，地址的变换过程：



由于查询快表的速度比查询页表的速度快很多，因此只要快表命中，就可以节省很多时间。因为局部性原理，一般来说快表的命中率可以达到99%以上。

题 1. 某系统使用基本分页存储管理，并采用了具有快表的地址变换机构。访问一次快表耗时 $1\mu s$ ，访问一次内存耗时 $100\mu s$ 。若快表的命中率为90%，那么访问一个逻辑地址的平均耗时是多少？

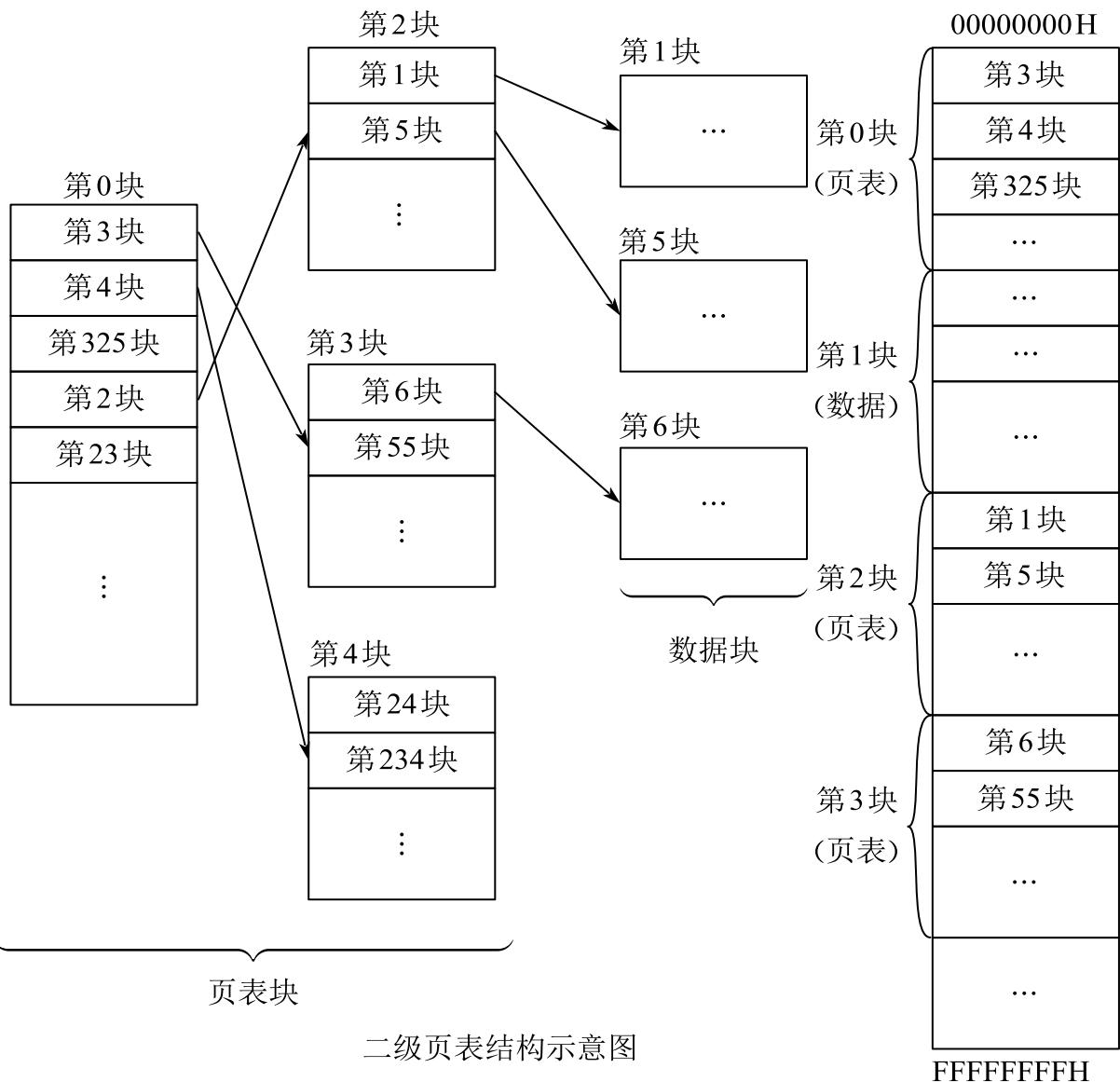
解析: $(1+100) \times 0.9 + (1+100+100) \times 0.1 = 111 \mu s$

	地址变换过程	访问一个逻辑地址的访问次数
基本地址变换机构	①算页号、页内偏移量 ②检查页号合法性 ③查页表，找到页面存放的内存块号 ④根据内存块号与页内偏移量得到物理地址 ⑤访问目标内存单元	两次访存
具有快表的地 址变换机构	①算页号、页内偏移量 ②检查页号合法性 ③查快表，若命中，即可知道页面存放的内存块号，可直接进行⑤ ④查页表，找到页面存放的内存块号，并且将页表项复制到快表中 ⑤根据内存块号与页内偏移量得到物理地址 ⑥访问目标内存单元	快表命中，只需一次访存 快表未命中，需要两次访存

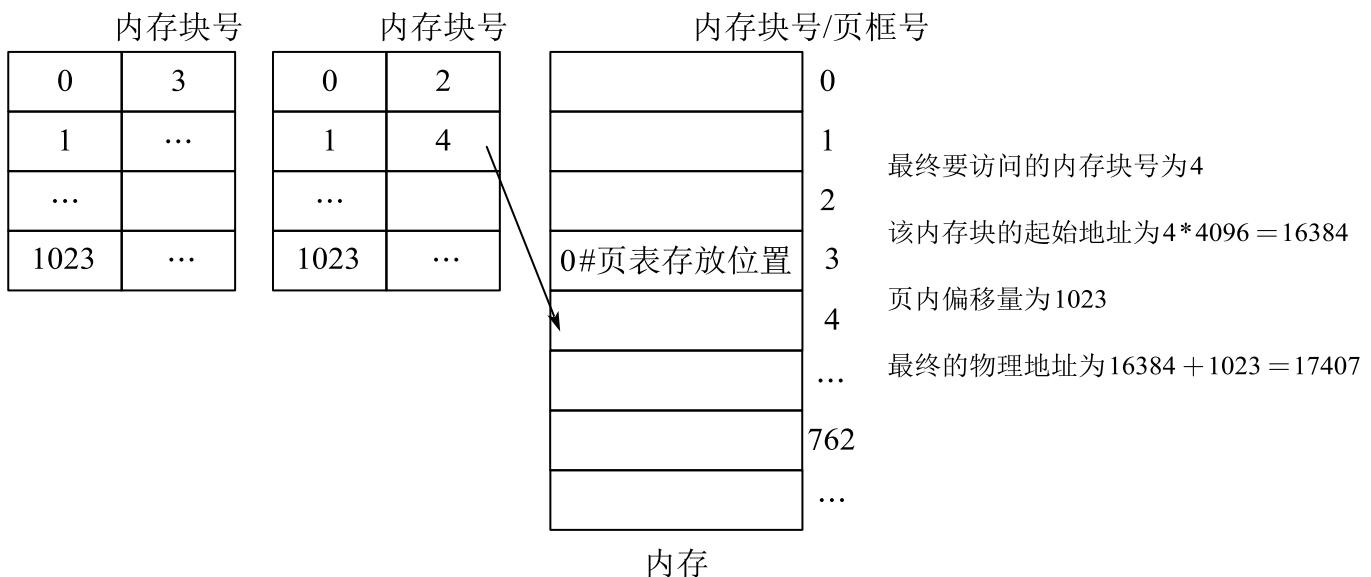
④两级页表



逻辑地址的空间格式



题1. 将逻辑地址(0000000000, 0000000001, 111111111111)转换为物理地址(用十进制表示)。



- ① 按照地址结构将逻辑地址拆分成三部分；
- ② 从 *PCB* 中读出页目录表始址，再根据一级页号查页目录表，找到下一级页表在内存中的存放位置；
- ③ 根据二级页号查表，找到最终想访问的内存块号；
- ④ 结合页内偏移量得到物理地址。

题 2. 某系统按字节编址，采用 40 位逻辑地址，页面大小为 $4KB$ ，页表项大小为 $4B$ ，假设采用纯页式存储，则要采用（ ）级页表，页内偏移量为（ ）位？

解析：页面大小 = $4KB = 2^{12}B$ ，按字节编址，因此页内偏移量为 12 位

页号 = $40 - 12 = 28$ 位

页面大小： $2^{12}B$ ，页表项大小 = $4B$ ，则每个页面可存放 $2^{12}/4 = 2^{10}$ 个页表项

因此各级页表最多包含 2^{10} 个页表项，需要 10 位二进制位才能映射到 2^{10} 个页表项，因此每一级页表对应页号应为 10 位，总共 28 位的页号至少要分为三级。

逻辑地址:	页号 28 位	页内偏移量 12 位
-------	---------	------------

逻辑地址:	一级页号 8 位	二级页号 10 位	三级页号 10 位	页内偏移量 12 位
-------	----------	-----------	-----------	------------

2. 分段存储管理方式

1) 分段: 其逻辑地址由段号 S 与段内偏移量 W 两部分组成。

31 ... 16	15 ... 0
段号 S	段内偏移量 W

分段系统中的逻辑地址结构

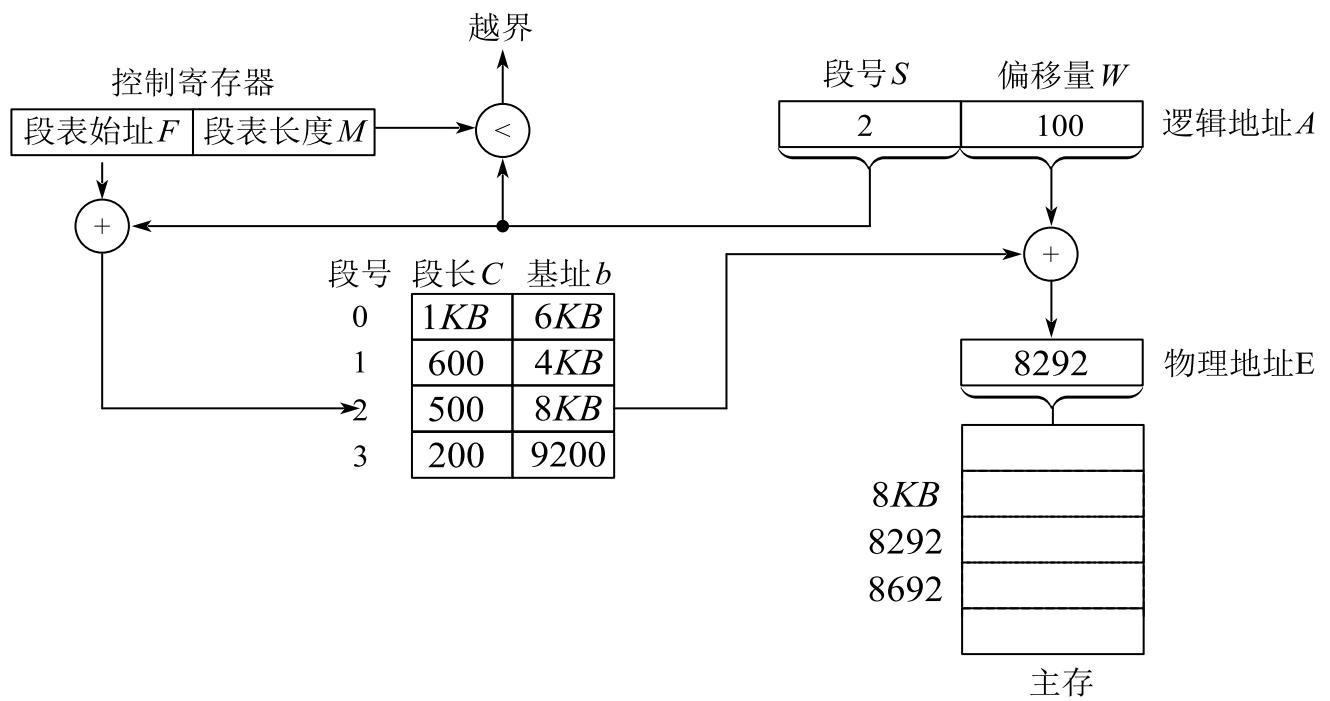
2) 段表: 每个段表项对应进程的一段, 段表项记录该段在内存中的始址和长度。

段表的内容如下图所示:

段号	段长	本段在主存的始址
----	----	----------

3) 地址变换机构:

在系统中设置了段表寄存器, 用于存放段表始址 F 和段表长度 M 。从逻辑地址 A 到物理地址 E 之间的地址变换过程如下:



分段系统的地址变换过程

题 1. 一个OS采用分段存储管理方式, 支持的最大段长位64KB, 一个进程的段

表如表所示(十进制)。请问: 逻辑地址0x47FD5H对应的物理地址是多少? 逻辑

地址0x003FFH对应的物理地址是多少?

段表

段号	段长	段起始地址
0	512	80K
1	20K	50K
2	12K	81K
3	3K	96K
4	32K	10K

解析：1) 在逻辑地址 $0x47FD5H$ 中，因为最大段长度为 $64KB = 2^{16}B$ ，所以低4个16进制数 $7FD5$ 表示段内地址，高位 $0x4$ 为段号，可知该地址对应的段号为4。该段的最大长度是 $32K$ ，即 $8000H$ ，该地址的段内地址是 $7FD5H$ ，因此逻辑地址 $0x47FD5H$ 对应的物理地址为 $10K + 7FD5H = 2800H + 7FD5H = A7D5H$ 。
 2) 在逻辑地址 $0x003FFH$ 中，段号为0，段内地址为 $03FFH$ 。段号0的段长只有 $512 = 200H$ ， $03FFH$ 大于 $200H$ ，因此会产生地址越界。

分段和分页管理的对比

- ① 页是信息的物理单位，段是信息的逻辑单位；
- ② 页的大小固定且由系统决定，段的长度不固定 ‘
- ③ 分页的用户程序地址空间是一维的，分段的用户程序地址空间是二维的。

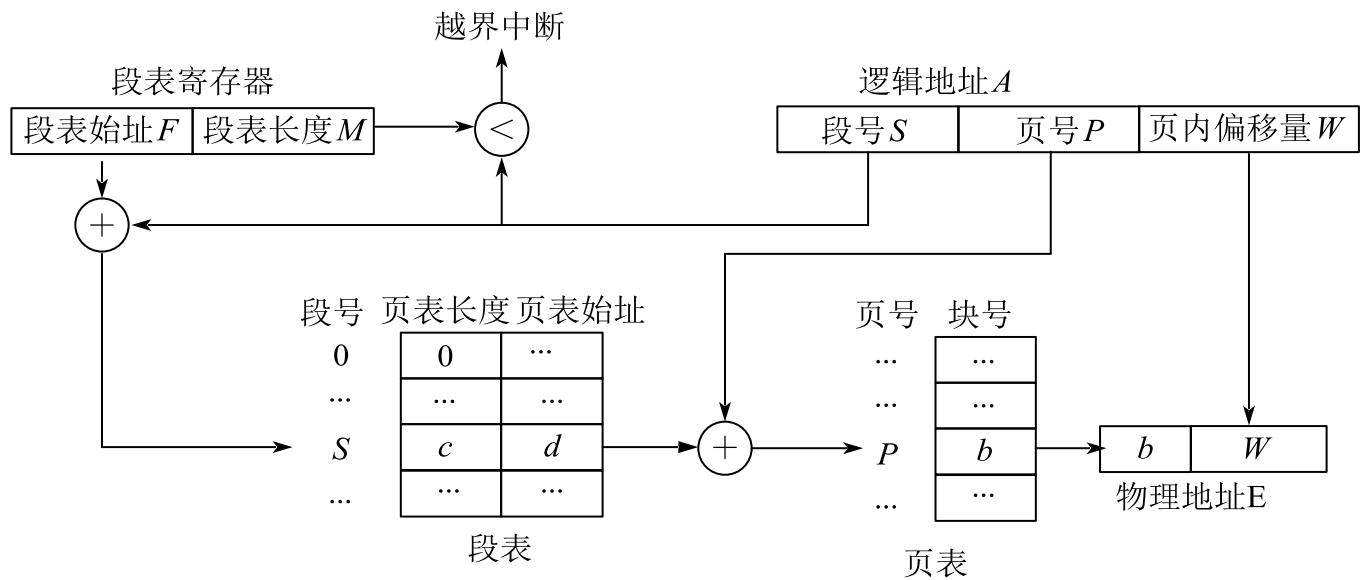
3. 段页式存储管理方式

在段页式系统中，作业的逻辑地址分为三部分：段号、页号和页内偏移量，如图所示。



段页式系统的地址变换机构

地址变换过程：



课时十二 练习题

1. 某计算机采用二级页表的分页存储管理方式，按字节编址，页大小为 $2^{10} B$ ，页表项大小为 $2B$ ，逻辑地址结构为

页目录号	页号	页内偏移量
------	----	-------

逻辑地址空间大小为 2^{16} 页，则表示整个逻辑地址空间的页目录表中包含表项的个数至少是（ ）

- A. 64 B. 128 C. 256 D. 512

2. 某进程的段表内容如下所示。

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

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. 段缺失异常 B. 得到内存地址4400
 C. 越权异常 D. 越界异常

3. 在分段存储管理系统中，用共享段表描述所有被共享的段。若进程 P_1 和 P_2 共享段 S ，则下列叙述中，错误的是（ ）

- A. 在物理内存中仅保存一份段 S 的内容
 B. 段 S 在 P_1 和 P_2 中应该具有相同的段号
 C. P_1 和 P_2 共享段 S 在共享段表中的段表项
 D. P_1 和 P_2 都不再使用段 S 时才回收段 S 所占的内存空间

4. 某计算机主存按字节编址，采用二级分页存储管理，地址结构如下：

页目录号（10位）	页号（10位）	页内偏移量（12位）
-----------	---------	------------

虚拟地址 $2050\ 1225H$ 对应的页目录号、页号分别是（ ）。

- A. $081H, 101H$
- B. $081H, 401H$
- C. $201H, 101H$
- D. $201H, 401H$

5. 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 $4B$ 。

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）
---------	------------

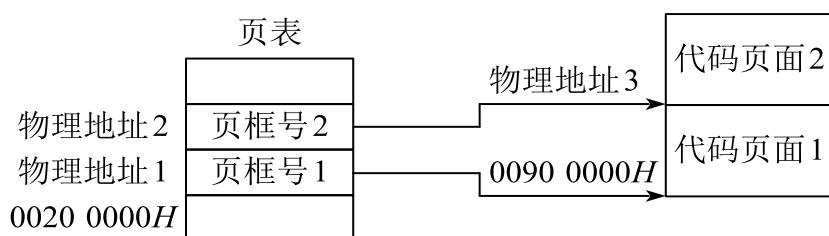
则页的大小是多少字节？页表最大占用多少字节？

2) 若使用二级页表的分页存储管理方式，逻辑地址结构为

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，一个代码段的起始逻辑地址为 $0000\ 8000H$ ，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）。请计算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。



6. 在一个段式存储管理系统中，其段表见下表A。试求表B中的逻辑地址所对应的物理地址。

表A 段表

段号	内存始址	段长
0	210	500
1	2350	20
2	100	90
3	1350	590
4	1938	95

表B 逻辑地址

段号	段内位移
0	430
1	10
2	500
3	400
4	112
5	32

7. 在某页式管理系统中，假定主存为64KB，分成16块，块号为0, 1, 2, ..., 15，设某进程有4页，共页号为0, 1, 2, 3，被分别装入主存的第9, 0, 1, 14块。

- 1) 该进程的总长度是多大？
- 2) 写出该进程每页在主存中的始址。
- 3) 若给出逻辑地址(0, 0) (1, 72) (2, 1023) (3, 99)，请计算出相应的内存地址(括号内的第一个数为十进制页号，第二个数为十进制页内地址)。

课时十三 虚拟内存管理

考点	重要程度	占分	题型
1. 虚拟内存的基本概念	★★	2~3	选择、简答
2. 页面置换算法	必考	4~6	选择、应用
3. 页面分配策略	★★★	2~3	选择

1. 虚拟内存的基本概念

传统存储管理方式具有一次性和驻留性的特征。一次性是指作业必须一次性全部嵌入内存后才能开始运行。这会导致大作业无法运行，多道程序并发下降。驻留性是指一旦作业被装入内存，就会一直驻留在内存中，直至作业运行结束。

1) 局部性原理

局部性原理既适用于程序结构，又适用于数据结构。

局部性原理表现在两个方面：

- ①时间局部性：通过将近来使用的指令和数据保存到高速缓存存储器中，并使用高速缓存的层次结构实现。
- ②空间局部性：通常使用较大的高速缓存，并将预取机制集成到高速缓存控制逻辑中实现。

2) 虚拟存储器的定义和特征

在操作系统的管理下，在用户看来似乎由一个比实际内存大得多得内存，这就是虚拟内存。

虚拟存储器有以下三个主要特征：

- ①多次性。多次性是指无须在作业运行时一次性地全部装入内存，而允许被分成多次调入内存运行。
- ②对换性。对换性是指无须在作业运行时一直常驻内存，而允许在作业的运行过程中，进行换进和换出。
- ③虚拟性。虚拟性是指从逻辑上扩充内存的容量，使用户所看到的内存容量远大于实际的内存容量。

3) 虚拟内存技术的实现

虚拟内存的实现有以下三种方式：

- ① 请求分页存储管理
- ② 请求分段存储管理
- ③ 请求段页式存储管理

不管哪种方式，都需要有一定的硬件支持。

请求分页管理方式

请求分页系统建立在基本分页系统基础之上，为了支持虚拟存储器功能而增加了请求调页功能和页面置换功能。请求分页是目前最常用的一种实现虚拟存储器的方法。

a. 页表机制

请求分页系统的页表机制不同于基本分页系统，请求分页系统是一个作业运行之前不要求全部一次性调入内存，因此在作业的运行过程中，必然会出现要访问的页面不在内存中的情况，如何发现和处理这种情况是请求分页系统必须解决的两个基本问题。为此，在请求页表项中增加了4个字段，如图所示。

页号	物理块号	状态位 P	访问字段 A	修改位 M	外存地址
----	------	---------	----------	---------	------

请求分页系统中的页表项

增加的4个字段说明如下：

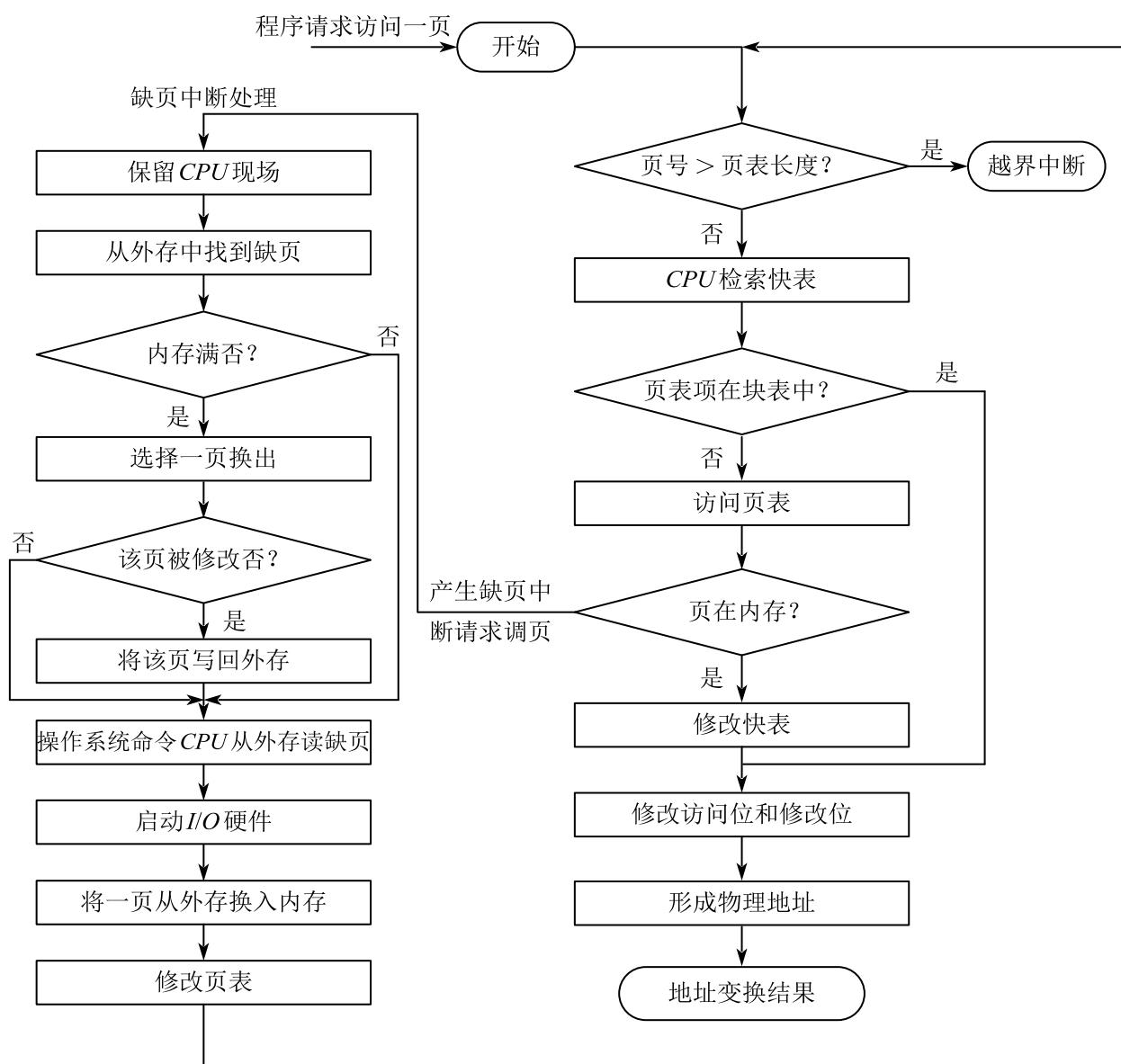
- I . 状态位 P 。用于指示该页是否已调入内存，供程序访问时参考。
- II . 访问字段 A 。用于记录本页在一段时间内被访问的次数，或记录本页最近已有多长时间未被访问，供置换算法换出页面时参考。
- III. 修改位 M 。标识该页在调入内存后是否被修改过。
- IV. 外存地址。用于指出该页在外存上的地址，通常是物理块号，供调入该页时参考。

b. 缺页中断机构

在请求分页系统中，每当所要访问的页面不在内存中时，便产生一个缺页中断，请求操作系统将所缺的页调入内存。此时应将缺页的进程阻塞（调页完成唤醒），若内存中有空闲块，则分配一个块，将要调入的页装入该块，并修改了页表中的相应页表项，若此时内存中没有空闲块，则要淘汰某页（若被淘汰页在内存期间被修改过，则要将其写回外存）。

c. 地址变换机构

请求分页系统中的地址变换机构，是在分页系统地址变换机构的基础上，为实现虚拟内存增加了某些功能而形成的。



请求分页中的地址变换过程

题 1. 下列关于虚拟存储器的叙述中，正确的是（ ）

- 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限制
- D. 虚拟存储容量只受内存容量限制

答案： B

解析：装入程序时，只将程序的一部分装入内存，而将其余部分留在外存，就可以启动程序执行。采用连续分配方式时，会使相当一部分内存空间都处于暂时或“永久”的空闲状态，造成内存资源的严重浪费，也无法从逻辑上扩大内存容量，因此虚拟内存的实现只能建立在离散分配的内存管理基础上。有以下三种实现方式：①请求分页存储管理；②请求分段存储管理；③请求段页式存储管理。虚拟存储器容量既不受外存容量限制，又不受内存容量限制，而是由CPU的寻址范围决定的。

题 2. 虚拟存储技术是（ ）

- A. 补充内存物理空间的技术
- B. 补充内存逻辑空间的技术
- C. 补充外存空间的技术
- D. 扩充输入/输出缓冲区的技术

答案： B

解析：虚拟存储技术并未实际扩充内存、外存，而是采用相关技术相对地扩充主存。

题 3. 覆盖技术与虚拟存储技术有何本质上的不同？交换技术与虚拟存储技术中使用的调入/调出技术有何相同与不同之处？

解析：1) 覆盖技术与虚拟存储技术最本质的不同在于，覆盖程序段的最大长度要受内存容量大小的限制，而虚拟存储器中程序的最大长度不受内存容量的限制，只受计算机地址结构的限制。另外，覆盖技术中的覆盖段由程序员设计，且要求覆盖段中的各个覆盖具有相对独立性，不存在直接联系或相互交叉访问；而虚拟存储技术对用户的程序段没有这种要求。

2) 交换技术就是把暂时不用的某个程序及数据从内存移到外存中，以便腾出必要的内存空间，或把指定的程序或数据从外存读到内存中的一种内存扩充技术。交换技术与虚存中使用的调入/调出技术的主要区别是：交换技术调入/调出整个进程，因此一个进程的大小要受内存容量大小的限制；而虚存中使用的调入/调出技术在内存和外存之间来回传递的是页面或分段，而不是整个进程，从而使进程的地址映射具有更大的灵活性，且允许进程的大小比可用的内存空间大。

2. 页面置换算法

常见的置换算法有以下四种：

1) 最佳(OPT)置换算法

算法思想：选择以后永不使用的页面淘汰或者在最长时间内不再被访问的页面，以保证获得最低的缺页率。

题 1. 假定系统为某进程分配了三个物理块，并考虑有页面号引用串 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1。采用最佳置换算法，产生几次中断以及页面置换的次数式多少？

访问页面	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
物理块1	7	7	7	2		2		2		2		2		2				7		
物理块2		0	0	0		0		4		0		0		0				0		
物理块3			1	1		3		3		3		3		1				1		
缺页否	√	√	√	√		√		√		√		√		√				√		

利用最佳置换算法时的置换图

发生缺页中断的次数为9， 页面置换的次数为6。

2) 先进先出 (FIFO) 页面置换算法

算法思想：优先淘汰最早进入内存的页面，即在内存中驻留时间最久的页面。

题 1. 假定系统为某进程分配了三个物理块，并考虑有页面号引用串 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1。采用先进先出 (FIFO) 置换算法，产生几次中断以及页面置换的次数式多少？

访问页面	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
物理块1	7	7	7	2		2	2	4	4	4	0			0	0			7	7	7
物理块2		0	0	0		3	3	3	2	2	2			1	1			1	0	0
物理块3			1	1		1	0	0	0	3	3			3	2			2	2	1
缺页否	√	√	√	√		√	√	√	√	√	√			√	√			√	√	√

利用 FIFO 置换算法时的置换图

发生缺页中断的次数为15， 页面置换的次数为12。

题 2. 若页面访问顺序为 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4。若采用 FIFO 置换算法，当分配的物理块为3个时，缺页次数为9次；当分配的物理块为4个时，缺页次数为10次。分配给进程的物理块增多，但缺页次数不减反增。

访问页面	3	2	1	0	3	2	4	3	2	1	0	4
物理块1	3	3	3	0	0	0	4			4	4	
物理块1		2	2	2	3	3	3			1	1	
物理块1			1	1	1	2	2			2	0	
缺页否	√	√	√	√	√	√	√			√	√	
物理块1	3	3	3	3			4	4	4	4	0	0
物理块1		2	2	2			2	3	3	3	3	4
物理块1			1	1			1	1	2	2	2	2
物理块1				0			0	0	0	1	1	1
缺页否	√	√	√	√			√	√	√	√	√	√

Belady 异常

这种现象称为 *Belady* 异常。*FIFO* 算法会出现 *Belady* 异常。

3) 最近最久未使用 (*LRU*) 置换算法

算法思想：选择最近最久时间未访问过的页面予以淘汰。

题 1. 假定系统为某进程分配了三个物理块，并考虑有页面号引用串 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1。采用最近最久未使用 (*LRU*) 置换算法，产生几次中断以及页面置换的次数式多少？

访问页面	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
物理块1	7	7	7	2		2		4	4	4	0			1		1		1		
物理块2		0	0	0		0		0	0	3	3			3		0		0		
物理块3			1	1		3		3	2	2	2			2		2		7		
缺页否	√	√	√	√		√		√	√	√	√			√		√		√		

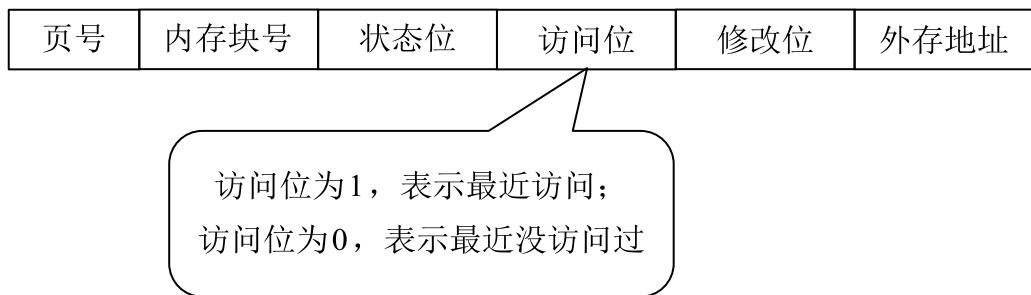
LRU 页面置换算法时的置换图

发生缺页中断的次数为 12，页面置换的次数为 9。

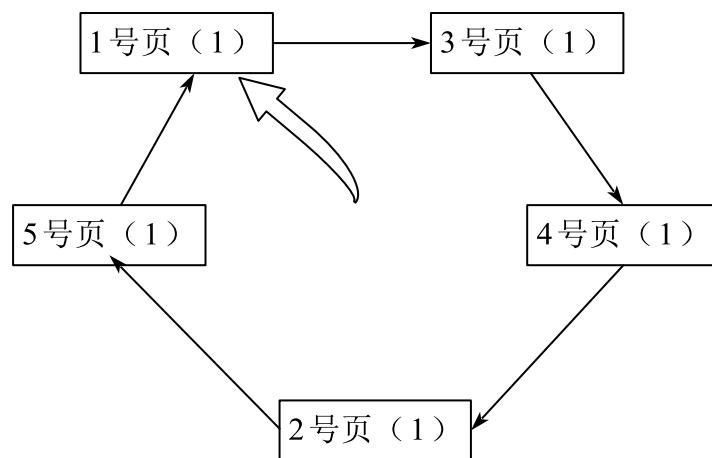
4) 时钟(CLOCK)置换算法

LRU 算法的性能接近于 *OPT* 算法，但实现起来比较困难，且开销大：*FIFO* 算法实现简单，但性能差。因此，操作系统的设计师尝试了很多算法，试图用比较小的开销接近 *LRU* 算法的性能，这类算法都是 *CLOCK* 算法的变体。因为算法要循环扫描缓冲区，像时钟的指针一样转动，所以称为 *CLOCK* 算法，又称最近未用 (*NRU*) 算法。

简单 *CLOCK* 算法实现方法：

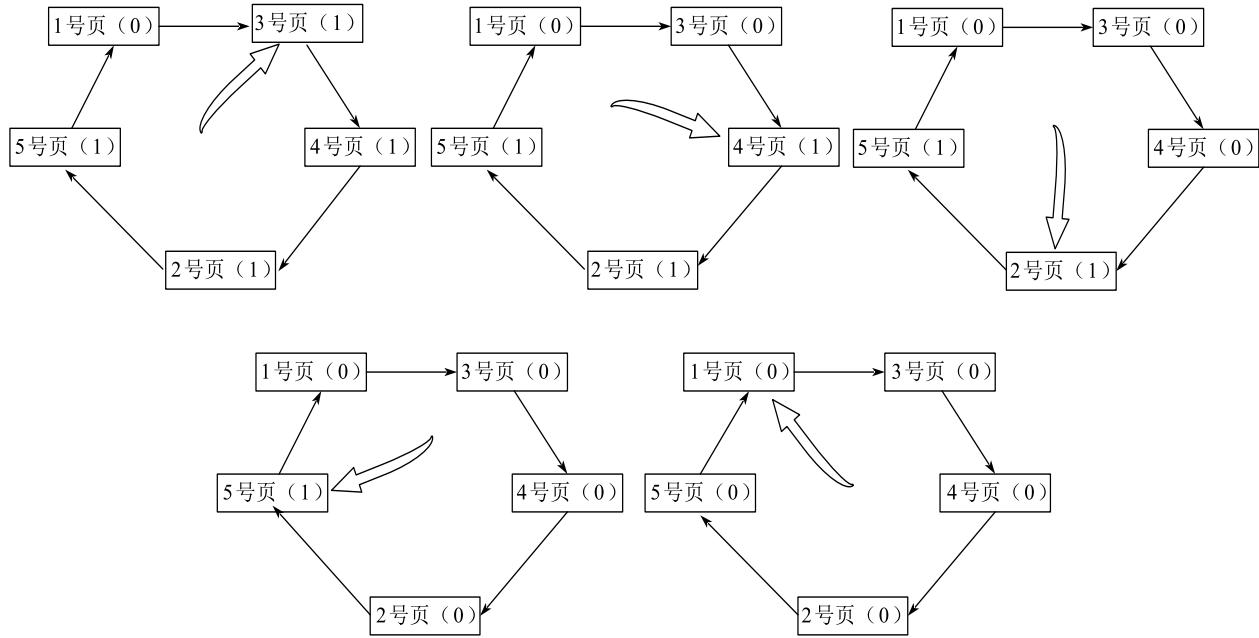


题 1. 假设系统为进程分配了 5 个内存块，并考虑到有以下页面号引用串：1, 3, 4, 2, 5, 6, 3, 4, 7。



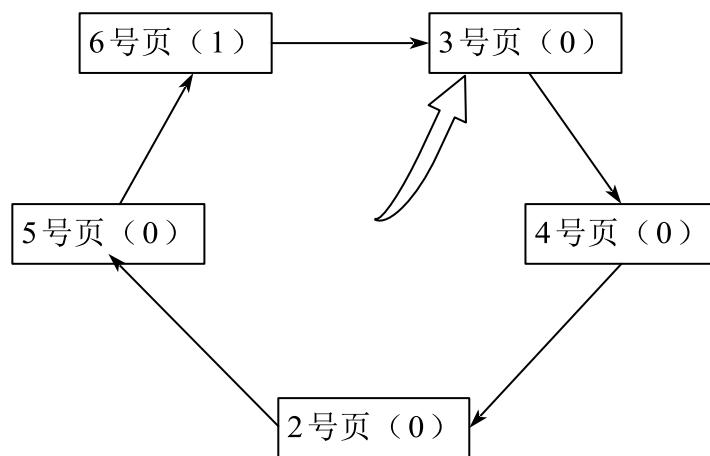
初识状态

按照CLOCK置换算法的规则，扫描过程为：



第一轮扫描过程

第一轮扫描中，未找到使用位为0的页面，因此需要进行第二轮扫描。第二轮扫描中，1号页面的使用位为0，因此将1号页面换出，将6号页面换入，将6号页的访问位设置为1，并将扫描指针后移（若下次需要换出页面，则从3号页面开始扫描），如图所示：



第二轮扫描后指针指向的页面

改进型CLOCK算法：

利用(访问位, 修改位)的形式表示各页面状态。如(1, 1)表示一个页面近期被访问过, 且被修改过。

置换算法的对比:

	算法规则	有缺点
OPT	优先淘汰最长时间内不会被访问的页面	缺页率最小, 性能最好; 但无法实现
$FIFO$	优先淘汰最先进入内存的页面	实现简单; 但性能很差, 可能出现 <i>Belady</i> 异常
LRU	优先淘汰最近最久没访问的页面	性能很好; 但需要硬件支持, 算法开销大
$CLOCK(NRU)$	循环扫描各页面 第一轮淘汰访问位 = 0 的, 并将扫描过的页面访问位改为1。若第一轮没选中, 则进行第二轮扫描	实现简单, 算法开销小; 但未考虑页面是否被修改过
改进型 $CLOCK$ (改进型 NRU)	若用(访问位, 修改位)的形式表述, 则 第一轮: 淘汰(0, 0) 第二轮: 淘汰(0, 1), 并将扫描过的页面访问位都置为0 第三轮: 淘汰(0, 0) 第四轮: 淘汰(0, 1)	算法开销较小, 性能也不错

题 2. 在一个请求分页存储管理系统中,一个作业的页面走向为 $4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5$, 当分配给作业的物理块数分别为3和4时, 试计算采用下述页面淘汰算法时的缺页率(假设开始执行时主存中没有页面), 并比较结果。

- 1) 最佳置换算法
- 2) 先进先出置换算法
- 3) 最近最久未使用算

解析:

1) 根据页面走向, 使用最佳置换算法时, 页面置换情况见下表。

物理块数为3时:

走向	4	3	2	1	4	3	5	4	3	2	1	5
块1	4	4	4	4	4	4	4	4	4	2	2	2
块2		3	3	3	3	3	3	3	3	3	1	1
块3			2	1	1	1	5	5	5	5	5	5
缺页	✓	✓	✓	✓			✓			✓	✓	

缺页率为 $7/12$ 。

物理块数为4时:

走向	4	3	2	1	4	3	5	4	3	2	1	5
块1	4	4	4	4	4	4	4	4	4	4	1	1
块2		3	3	3	3	3	3	3	3	3	3	3
块3			2	2	2	2	2	2	2	2	2	2
块4				1	1	1	5	5	5	5	5	5
缺页	✓	✓	✓	✓			✓				✓	

缺页率为 $6/12$ 。

由上述结果可以看出, 增加分配作业的内存块数可以降低缺页率。

2) 根据页面走向, 使用先进先出页面淘汰算法时, 页面置换情况见下表。

物理块数为3时:

蜂考

走向	4	3	2	1	4	3	5	4	3	2	1	5
块1	4	4	4	1	1	1	5	5	5	5	5	
块2		3	3	3	4	4	4	4	4	2	2	
块3			2	2	2	3	3	3	3	3	1	
缺页	√	√	√	√	√	√	√			√	√	

缺页率为9/12。

物理块数为4时：

走向	4	3	2	1	4	3	5	4	3	2	1	5
块1	4	4	4	4	4	4	5	5	5	5	1	1
块2		3	3	3	3	3	3	4	4	4	4	5
块3			2	2	2	2	2	2	3	3	3	3
块4				1	1	1	1	1	1	2	2	2
缺页	√	√	√	√			√	√	√	√	√	√

缺页率为10/12。

由上述结果可以看出，对先进先出算法而言，增加分配作业的内存块数反而使缺页率提升，即出现 *Belady* 现象。

3) 根据页面走向，使用最近最久未使用页面淘汰算法时，页面置换情况见下表。

物理块数为3时：

走向	4	3	2	1	4	3	5	4	3	2	1	5
块1	4	4	4	1	1	1	5	5	5	2	2	2
块2		3	3	3	4	4	4	4	4	4	1	1
块3			2	2	2	3	3	3	3	3	3	5
缺页	√	√	√	√	√	√	√			√	√	

缺页率为10/12。

物理块数为4时：

走向	4	3	2	1	4	3	5	4	3	2	1	5
块1	4	4	4	4	4	4	4	4	4	4	4	5
块2		3	3	3	3	3	3	3	3	3	3	3
块3			2	2	2	2	5	5	5	5	1	1
块4				1	1	1	1	1	1	2	2	2
缺页	√	√	√	√			√			√	√	√

缺页率为8/12。

由上述结果可以看出，增加分配作业的内存块数可以降低缺页率。

题 3. 某系统有4个页框，某个进程的页面使用情况见下表，问采用 *FIFO*、*LRU*、

简单 *CLOCK* 和改进型 *CLOCK* 置换算法，将会替换哪一页？

页号	装入时间	上次引用时间	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

其中，*R* 是读标志位，*M* 是修改标志位。

解析：

- 1) *FIFO* 置换算法选择最先进入内存的页面进行替换。由表中装入时间可知，第2页最先进入内存，因此 *FIFO* 置换算法将选择第2页替换。
- 2) *LRU* 置换算法选择最近最长时间未使用的页面进行替换。由表中的上次引用时间可知，第1页是最长时间未使用的页面，因此 *LRU* 置换算法将选择第1页替换。
- 3) 简单 *CLOCK* 置换算法从上一次位置开始扫描，选择第一个访问位为0的页面进行替换。由表中的*R* (读) 标志位可知，依次扫描2, 3, 0 (按装入顺序)，页面0未被访问，扫描结束，因此简单 *CLOCK* 置换算法将选择第0页替换。

3. 页面分配策略

1) 驻留集大小

驻留集：指请求分页存储管理中给进程分配的物理块的集合。

若驻留集太小，会导致缺页策略，系统要花大量的时间来处理缺页；若驻留集太大，又会导致多道程序并发度下降，资源利用率降低。所以应采用一定的策略：

①固定分配局部置换：它为每个进程分配一定数目的物理块，在整个运行期间都不改变。

②可变分配全局置换：它为每个进程分配一定数目的物理块，操作系统自身也保持一个空闲物理块队列。

③可变分配局部置换：它为每个进程分配一定数目的物理块，当某个进程发生缺页时，只允许从该进程在内存的页面中选出一页换出，因此不会影响其他进程的运行。

2) 调入页面的时机

①预调页策略：根据局部性原理，一次调入若干相邻的页可能会比一次调入一页更高效。

②请求调页策略：进程在运行中需要访问的页面不在内存而提出请求，由系统将所需页面调入内存。

3) 从何处调入页面

①系统拥有足够的对换区空间：可以全部从对换区调入所需页面，以提高调页速度。

②系统缺少足够的对换区空间：凡不会被修改的文件都直接从文件区调入；而当换出这些页面时，不必再换出。对于那些可能被修改的部分，在将它们换出时须调到对换区。

③UNIX方式：与进程有关的文件都放在文件区，因此未运行过的页面都应从文件区调入。

4) 抖动

在页面置换过程中，一种最糟糕的情形是，刚刚换出的页面马上又要换入主存，刚刚换入的页面马上又要换出主存，这种频繁的页面调度行为称为抖动或颠簸。频繁发生缺页中断(抖动)的主要原因是：某个进程频繁访问的页面数目高于可用的物理页帧数目。

5) 工作集

工作集是指在某段时间间隔内，进程要访问的页面集合。

一般来说，驻留集大小不能小于工作集大小，否则进程运行过程中将频繁缺页。

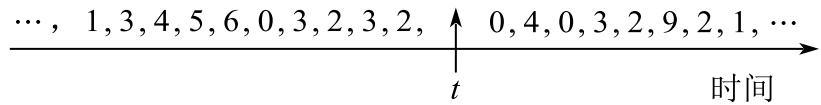
题 1. 当系统发生抖动时，可以采取的有效措施是（ ）。

- 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

答案： A

解析：在具有对换功能的操作系统中，通常把外存分为文件区和对换区。前者用于存放文件，后者用于存放从内存换出的进程。抖动现象是指刚刚被换出的页很快又要被访问，为此又要换出其他页，而该页又很快被访问，如此频繁地置换页面，以致大部分时间都花在页面置换上，导致系统性能下降。撤销部分进程可以减少所要用到的页面数，防止抖动。对换区大小和进程优先级都与抖动无关。

题 2. 某进程访问页面的序列如下所示。



若工作集的窗口大小为6，则在 t 时刻的工作集为()。

- A. {6, 0, 3, 2}
- B. {2, 3, 0, 4}
- C. {0, 4, 3, 2, 9}
- D. {4, 5, 6, 0, 3, 2}

答案：A

解析：在一时刻 t ，都存在一个集合，它包含所有最近 k 次（该窗口大小为6）内存访问所访问过的页面。这个集合 $w(k, t)$ 就是工作集。题中最近6次访问的页面分别为6, 0, 3, 2, 3, 2，去除重复的页面，形成的工作集为{6, 0, 3, 2}。

课时十三 练习题

1. 在缺页处理过程中，操作系统执行的操作可能是（ ）。

- 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

2. 以下不属于虚拟内存特征的是（ ）。

- A. 一次性
- B. 多次性
- C. 对换性
- D. 离散性

3. 为使虚存系统有效地发挥其预期地作用，所运行的程序应具有的特性是（ ）。

- A. 该程序不应含有过多的 I/O 操作
- B. 该程序的大小不应超过实际的内存容量
- C. 该程序应具有较好的局部性
- D. 该程序的指令相关性不应过多

4. 考虑页面置换算法，系统有 m 个物理块供调度，初始时全空，页面引用串长度为 p ，包含了 n 个不同的页号，无论用什么算法，缺页次数不会少于（ ）。

- A. m B. p C. n D. $\min(m, n)$

5. 在页面置换策略中，()策略可能引起抖动。

- A. FIFO B. LRU C. 没有一种 D. 所有

6. 已知系统为32位实地址，采用48位虚拟地址，页面大小为 $4KB$ ，页表项大小为 $8B$ 。假设系统使用纯页式存储，则要采用()级页表，页内偏移()位。

A. 3, 12

B. 3, 14

C. 4, 12

D. 4, 14

7. 在页式虚拟存储管理系统中，采用某些页面置换算法会出现*Belady*异常现象，即进程的缺页次数会随着分配给该进程的页框个数的增加而增加。下列算法中，可能出现*Belady*异常现象的是()。

I. LRU算法

II. FIFO算法

III. OPT算法

A. 仅 II B. 仅 I、II

C. 仅 I、III D. 仅 II、III

8. 在一个请求分页系统中，采用LRU页面置换算法时候，假如一个作业的页面走向为 $1, 3, 2, 1, 1, 3, 5, 1, 3, 2, 1, 5$ ，当分配给该作业的物理块数分别为3和4时，试计算在访问过程中发生的缺页次数和缺页率。

9. 某一进程已分配到4个页帧，见下表(编号为十进制，从0开始)。当进程访问第4页时，产生缺页中断，请分别用 *FIFO* (先进先出)、*LRU* (最近最少使用)、*NRU* (最近不用) 算法，决定缺页中断服务程序选择换出的页面。

虚拟页号	页帧	装入时间	最近访问时间	访问位	修改位
2	0	60	161	0	1
1	1	130	160	0	0
0	2	26	162	1	0
3	3	20	163	1	1

课时十四 文件管理（一）

考点	重要程度	占分	题型
1. 文件逻辑结构	★★	1~2	选择
2. 文件目录	★★	2~3	选择、简答
3. 文件共享	★	1~2	选择
4. 文件保护	★	1~2	选择

1. 文件逻辑结构

1) 文件系统基础

①文件的定义

文件是以计算机硬盘为载体的存储在计算机上的信息集合，文件可以是文本文档、图片、程序等。在系统运行时，计算机以进程为基本单位进行资源的调度和分配；而在用户进行的输入输出中，则以文件为基本单位。

②文件的属性

文件具有一定的属性，系统不同，属性也会有所不同，但通常都包括如下属性：

名称、标识符、类型、位置、大小、保护、时间、日期和用户标识。

③文件的基本操作

操作系统提供系统调用，它对文件进行创建、写、读、重定位、删除和截断等操作。

④文件的打开与关闭

题 1. 从用户的观点看，操作系统中引入文件系统的目的是（ ）。

- A. 保护用户数据
- B. 实现对文件的按名存取
- C. 实现虚拟存储
- D. 保存用户和系统文档及数据

答案： B

解析：从系统角度看，文件系统负责对文件的存储空间进行组织、分配，负责文件的存储并对存入文件进行保护、检索。从用户角度看，文件系统根据一定的格式将用户的文件存放到文件存储器中适当的地方，当用户需要使用文件时，系统根据用户所给的文件名能够从文件存储器中找到所需要的文件。

题 2. 下列说法中，（ ）属于文件的逻辑结构的范畴。

- A. 连续文件
- B. 系统文件
- C. 链接文件
- D. 流式文件

答案： D

2) 文件的逻辑结构

文件的逻辑结构是从用户观点出发看到的文件的组织方式。文件的物理结构是从实现观点出发看到的文件在外存上的存储组织形式。

按逻辑结构，文件可划分为无结构文件和有结构文件。

①无结构文件(流式文件)

无结构文件是最简单的文件组织形式。无结构文件将数据按顺序组织成记录并积累、保存，它是有序相关信息项的集合，以字节(*Byte*)为单位。

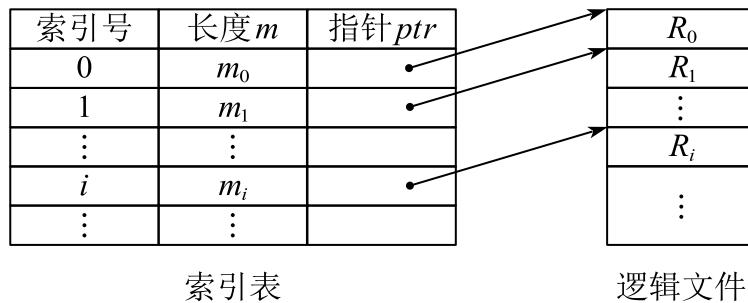
②有结构文件(记录式文件)

有结构文件按记录的组织形式可以分为如下几种：

a. 顺序文件。文件中的记录一个接一个地顺序排列，记录通常是定长的，可以顺序存储或以链表形式存储，在访问时需要顺序搜索文件。可变长记录的顺序文件无法实现随机存取，定长记录可以，定长记录、顺序结构的顺序文件可以快速检索。最大缺点是不方便增加和删除记录。

b. 索引文件。建立一张索引表以加快检索速度，索引表本身是定长记录的顺序文

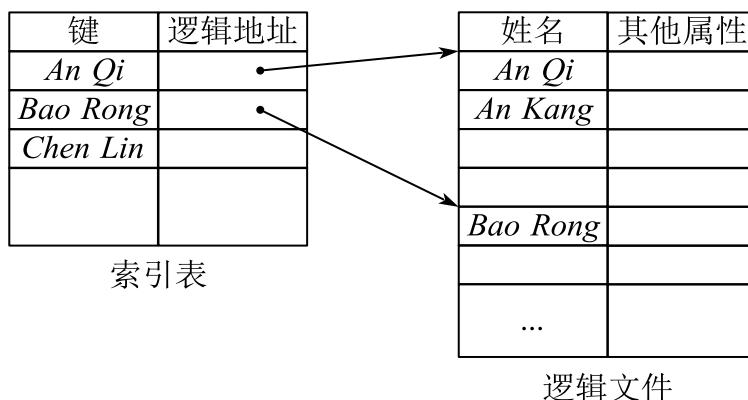
件。若索引表按关键字顺序排列，则可支持快速检索。



索引文件示意图

c. 索引顺序文件

索引顺序文件是顺序和索引两种组织形式的结合。



索引顺序文件示意图

d. 直接文件或散列文件 (Hash File)

给定记录的键值或通过散列函数转换的键值直接决定记录的物理地址。

题 1. 有一个顺序文件含有10000 条记录，平均查找的记录数为5000 个，采用索引顺序文件结构，则最好情况下平均只需查找()次记录。

-
- A. 1000 B. 10000 C. 100 D. 500

答案: C

解析: 最好的情况是有 $\sqrt{10000} = 100$ 组, 每组有 100 条记录, 因此顺序查找时平均查找记录个数 $= 50 + 50 = 100$ 。

2. 文件目录

与文件管理系统和文件集合相关联的是文件目录。

1) 文件控制块

与进程管理一样, 为实现目录管理, 操作系统中引入了文件控制块的数据结构。文件控制块 (*FCB*) 是用来存放控制文件需要的各种信息的数据结构, 以实现“按名存取”。*FCB* 的有序集合称为文件目录, 一个 *FCB* 就是一个文件目录项。为了创建一个新文件, 系统将分配一个 *FCB* 并存放在文件目录中, 成为目录项。

FCB 主要包含以下信息:

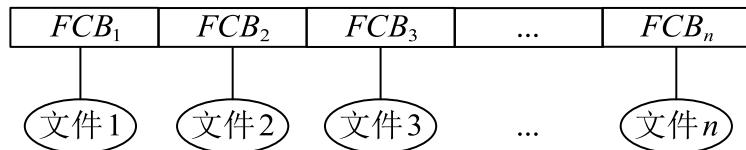
- ①基本信息, 如文件名、文件的物理位置、文件的逻辑结构、文件的物理结构等。
- ②存取控制信息, 如文件存取权限等。

最重要的是文件名、文件存放的物理地址。

2) 目录结构

①单级目录结构:

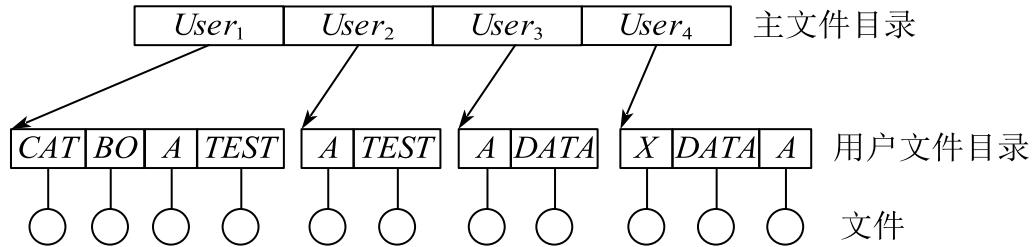
在整个文件系统中只建立一张目录表, 每个文件占一个目录项。



单级目录结构

②两级目录结构：

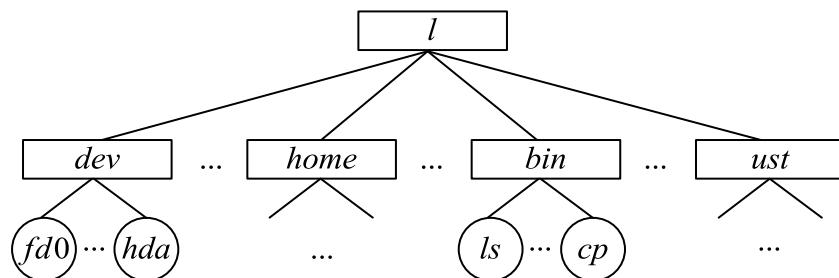
将文件目录分成文件目录和用户文件目录两级。



两级目录结构

③多级目录结构：

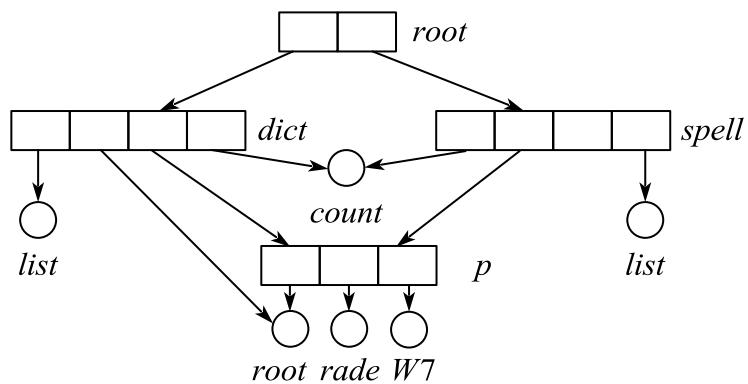
将两级目录结构层次关系加以推广，就形成了多级目录结构，即树形目录结构。



树形目录结构

④无环图目录结构

引入无环图目录结构是为了实现文件共享。



无环图目录结构

3) 索引结点

除了文件名之外的所有信息都放到索引结点中，每个文件对应一个索引结点。

目录项中只包含文件、索引结点指针，因此每个目录项的长度大幅减少。

由于目录项长度减少，因此每个磁盘块可以存放更多个目录项，因此检索文件时磁盘 I/O 的次数就少了很多。

题 1. 用户在删除某文件的过程中，操作系统不可能执行的操作是（ ）。

- A. 删除此文件所在的目录
- B. 删除与此文件关联的目录项
- C. 删除与此文件对应的文件控制块
- D. 释放与此文件关联的内存缓冲区

答案： A

解析：此文件所在目录下可能还存在其他文件，因此删除文件时不能（也不需要）删除文件所在的目录，而与此文件关联的目录项和文件控制块需要随着文件一同删除，同时释放文件关联的内存缓冲区。

题 2. 目录文件存放的信息是（ ）。

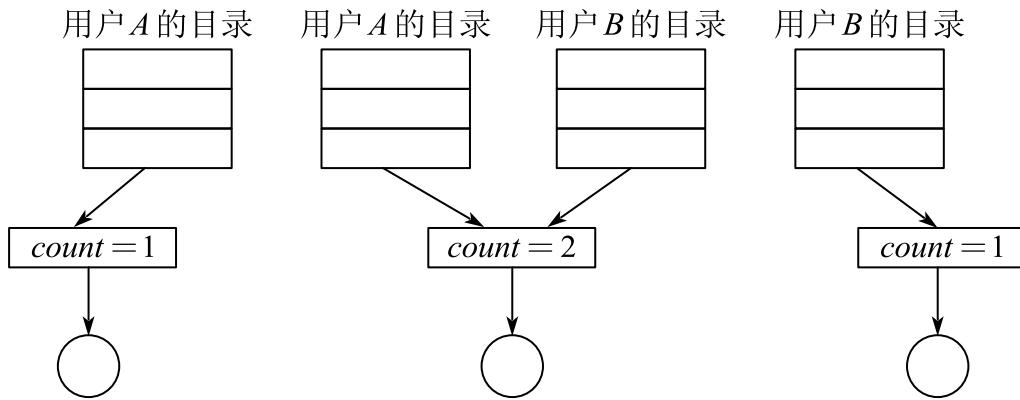
- A. 某一文件存放的数据信息
- B. 某一文件的文件目录
- C. 该目录中所有数据文件目录
- D. 该目录中所有子目录文件和数据文件的目录

答案： D

解析：目录文件是 FCB 的集合，一个目录中既可能有子目录，又可能有数据文件，因此目录文件中存放的是子目录和数据文件的信息。

3. 文件共享

1) 基于索引结点的共享方式(硬链接)



文件共享中的链接计数

索引结点中设置一个链接计数变量 $count$ ，用于表示链接到本索引结点上的用户目录项数。若 $count = 2$ ，说明此时有两个用户目录项链接到该索引结点上，或者说是有两个用户在共享此文件。若某个用户决定“删除”该文件，则只是要把用户目录中与该文件对应的目录项删除，且索引结点的 $count$ 值减1。若 $count > 0$ ，说明还有别的用户要使用该文件 暂时不能把文件数据删除，否则会导致指针悬空。

2) 基于符号链的共享方式(软链接)

为使用户 B 能共享用户 A 的一个文件 F ，可以由系统创建一个 $LINK$ 类型的新文件，也取名为 F ，并将文件 F 写入用户 B 的目录中，以实现用户 B 的目录与文件 F 的链接。在新文件中只包含被链接文件 F 的路径名。这样的链接方法被称为符号链接。新文件中的路径名只被视为符号链，当用户 B 要访问被链接的文件 F 且正要读 $LINK$ 类新文件时，操作系统根据新文件中的路径名去读该文件，从而实现用户 B 对文件 F 的共享。

题 1. 设文件 F_1 的当前引用计数值为 1，先建立文件 F_1 的符号链接(软链接)文件 F_2 ，再建立文件 F_1 的硬链接文件 F_3 ，然后删除文件 F_1 。此时，文件 F_2 和文件 F_3 的引用计数值分别是()。

- A. 0, 1 B. 1, 1 C. 1, 2 D. 2, 1

答案：B

解析：建立符号链接时，引用计数值直接复制；建立硬链接时，引用计数值加1。删除文件时，删除操作对于符号链接是不可见的，这并不影响文件系统，当以后再通过符号链接访问时，发现文件不存在，直接删除符号链接；但对于硬链接则不可直接删除，引用计数值减1，若值不为0，则不能删除此文件，因为还有其他硬链接指向此文件。

当建立 F_2 时， F_1 和 F_2 的引用计数值都为 1。当再建立 F_3 时， F_1 和 F_3 的引用计数值就都变成了 2。当后来删除 F_1 时， F_3 的引用计数值为 $2 - 1 = 1$ ， F_2 的引用计数值不变。

题 2. 若文件 F_1 的硬链接为 F_2 ，两个进程分别打开 F_1 和 F_2 ，获得对应的文件描述符为 Fd_1 和 Fd_2 ，则下列叙述中正确的是()。

- I. F_1 和 F_2 的读写指针位置保持相同
II. F_1 和 F_2 共享同一个内存索引结点
III. Fd_1 和 Fd_2 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. I、II 和 III

答案：B

解析：硬链接指通过索引结点进行连接。一个文件在物理存储器上有一个索引结点号。存在多个文件名指向同一个索引结点的情况，II 正确。两个进程各自维护自己的文件描述符，III 正确，I 错误。所以选 B。

4. 文件保护

1) 口令保护

为文件设置一个“口令”，用户想要访问文件时需要提供口令，由系统验证口令是否正确。

实现开销小，但“口令”一般存放在*FCB*或索引结点中(也就是存放在系统中)因此不太安全。

2) 加密保护

用一个“密码”对文件加密，用户想要访问文件时，需要提供相同的“密码”才能正确的解密。

安全性高，但加密/解密需要耗费一定的时间(如：异或加密)。

3) 访问控制

用一个访问控制表(*ACL*)记录各个用户(或各组用户)对文件的访问权限。

对文件的访问类型可以分为：读/写/执行/删除等。

实现灵活，可以实现复杂的文件保护功能。

题 1. 对一个文件的访问，常由（ ）共同限制。

- A. 用户访问权限和文件属性
- B. 用户访问权限和用户优先级
- C. 优先级和文件属性
- D. 文件属性和口令

答案： A

解析：对于这道题，只要能区分用户的访问权限和用户优先级，就能得到正确的答案。用户访问权限是指用户有没有权限访问该文件，而用户优先级是指在多个用户同时请求该文件时应该先满足谁。比如，图书馆的用户排队借一本书，某用户可能有更高的优先级，即他排在队伍的前面，但有可能轮到他时被告知他没有借阅那本书的权限。文件的属性包括保存在 *PCB* 中对文件访问的控制信息。

题 2. 加密保护和访问控制两种机制相比，（ ）。

- A. 加密保护机制的灵活性更好
- B. 访问控制机制的安全性更高
- C. 加密保护机制必须由系统实现
- D. 访问控制机制必须由系统实现

答案： D

解析：相对于加密保护机制，访问控制机制的安全性较差。因为访问控制的级别和保护力度较小，因此它的灵活性相对较高。若访问控制不由系统实现，则系统本身的安全性就无法保证。加密机制若由系统实现，则加密方法将无法扩展。

课时十四 练习题

1. 设置当前工作目录的主要目的是()。

- A. 节省外存空间
- B. 节省内存空间
- C. 加快文件的检索速度
- D. 加快文件的读/写速度

2. 文件系统中，文件访问控制信息存储的合理位置是()。

- A. 文件控制块
- B. 文件分配表
- C. 用户口令表
- D. 系统注册表

3. 文件的逻辑结构是为了方便()而设计的。

- A. 存储介质特性
- B. 操作系统的管理方式
- C. 主存容量
- D. 用户

4. 索引文件由逻辑文件和()组成。

- A. 符号表
- B. 索引表
- C. 交叉访问表
- D. 链接表

5. 若一个用户进程通过 *read* 系统调用读取一个磁盘文件中的数据，则下列关于此过程的叙述中，正确的是（ ）。

- I. 若该文件的数据不在内存，则该进程进入睡眠等待状态
 - II. 请求 *read* 系统调用会导致 CPU 从用户态切换到核心态
 - III. *read* 系统调用的参数应包含文件的名称
- A. 仅 I 、 II
 - B. 仅 I 、 III
 - C. 仅 II 、 III
 - D. I 、 II 和 III

6. 若文件系统中有两个文件重名，则不应采用（ ）。

- A. 单级目录结构
- B. 两级目录结构
- C. 树形目录结构
- D. 多级目录结构

7. UNIX 操作系统中，文件的索引结构放在（ ）。

- A. 超级块
- B. 索引结点
- C. 目录项
- D. 空闲块

8. 加密保护和访问控制两种机制相比，（ ）。

- A. 加密保护机制的灵活性更好

- B. 访问控制机制的安全性更高
- C. 加密保护机制必须由系统实现
- D. 访问控制机制必须由系统实现

9. 有文件系统如下图所示，图中的框表示目录，圆圈表示普通文件。

- 1) 可否建立 F 与 R 的链接？试加以说明。
- 2) 能否删除 R ？为什么？
- 3) 能否删除 N ？为什么？

课时十五 文件管理（二）

考点	重要程度	占分	题型
1. 文件物理结构(文件分配方式)	★★	1~2	选择
2. 文件存储空间管理	必考	3~5	选择、应用

1. 文件的物理结构

文件分配对应于文件的物理结构，是指如何为文件分配磁盘块。常用的磁盘空间方法有三种：

1) 连续分配

连续分配方法要求每个文件在磁盘上占有一组连续的块。文件的连续分配可以用第一块的磁盘地址和连续块的数量来定义。

若文件长 n 块并从位置 b 开始，则该文件将占有块 $b, b+1, b+2, \dots, b+n-1$ 。一个文件的目录条目包括开始块的地址和该文件所分配区域的长度。

优点：实现简单、存取速度快。

缺点：文件长度不宜动态增加。

2) 链接分配

链接分配采取离散分配的方式，消除了外部碎片，提高了磁盘空间的利用率。分为隐式链接和显式链接两种。

采用链式分配方式(隐式链接)方式的文件，除文件的最后一个盘块之外，每个盘块中都存有指向下一个盘块的指针。文件目录包括文件第一块的指针和最后一块的指针。但只支持顺序访问，不支持随机访问，查找效率低。

优点：方便文件拓展。

缺点：不会产生碎片问题，外存利用率高。

采用链式分配方式(显式链接)方式的文件，把用于链接文件各物理块的指针显式的放在一张表中，即文件分配表(*FAT*)。一个磁盘只会建立一张文件分配表。

开机时文件分配表放入内存，并常驻内存。

优点：支持顺序访问，也支持随机访问，查找效率高。

缺点：文件分配表只需要占用一定的存储空间。

3) 索引分配

索引分配允许文件离散地分配在各个磁盘块中，系统会为每个文件建立一张索引表，索引表中记录了文件的各个逻辑块对应的物理块(索引表的功能类似于内存管理中的页表——建立逻辑页面到物理页之间的映射关系)。

索引表存放的磁盘块称为索引块。文件数据存放的磁盘块称为数据块。

优点：索引分配方式可以支持随机访问。文件拓展也很容易实现(只需要给文件分配一个空闲块，并增加一个索引表项即可)。

缺点：索引表需要占用一定的存储空间。

若磁盘块装不下文件的整张索引表，需要采用以下机制来解决处理这个问题：

①链接方案：如果索引表太小，一个索引块装不下，那么可以将多个索引块链接起来存放。

②多层次索引：建立多层次索引，使第一层索引块指向第二层的索引块，还可以根据文件大小的要求再建立第三层、第四层索引块。

③混合索引：多种索引分配方式的结合。例如，一个文件的顶级索引表中既包含直接地址索引(直接指向数据块)，又包含一级间接索引(指向单层索引表)、还包含两级间接索引(指向两层索引表)。

分配方式的对比：

		实现方式	目录项内容	优点	缺点
顺序分配		为文件分配的必须是连续的磁盘块	起始块号、文件长度	顺序存取速度快，支持随机访问	会产生碎片，不利于文件拓展
链接分配	隐式链接	除文件的最后一个盘块之外，每个盘块中都存有指向下一个盘块的指针	起始块号、结束块号	可解决碎片问题，外存利用率高，文件拓展实现方便	只能顺序访问，不能随机访问
	显式链接	建立一张文件分配表(<i>FAT</i>)，显示记录盘块的先后关系(开机后 <i>FAT</i> 常驻内存)	起始块号	除了拥有隐式链接的优点之外，还可通过查询内存中的 <i>FAT</i> 实现随机访问	<i>FAT</i> 需要占用一定的存储空间
索引分配		为文件数据块建立索引表。若文件太大，可采用链接方案、多层次索引、混合索引	链接方案记录的是第一个索引块的块号，多层次/混合索引记录的是顶级索引块的块号	支持随机访问，易于实现文件的拓展	索引表需占用一定的存储空间。访问数据块前需要先读入索引块。若采用链接方案，查尔斯索引块时可能需要很多次读磁盘操作

题 1. 假定磁盘块大小为 $1KB$ ，对于 $540MB$ 的硬盘，其文件分配表(FAT)最少需

要占用多少存储空间？

解析：对于 $540MB$ 的硬盘，硬盘总块数为 $540MB/1KB = 540K$ 个。

因为 $540K$ 刚好小于 2^{20} ，所以文件分配表的每个表目可用20位，即 $20B/8 = 2.5B$ ，这样 FAT 占用的存储空间大小为 $2.5B \times 540K = 1350KB$ 。

题 2. 某文件系统采用多级索引的方式组织文件的数据存放，假定在文件的

i_node 中设有13个地址项，其中直接索引10项，一次间接索引1项，二次间接索引1项，三次间接索引1项。数据块大小为 $4KB$ ，磁盘地址用 $4B$ 表示，试问：

1) 这个文件系统允许的最大文件长度是多少？

2) 一个 $2GB$ 大小的文件，在这个文件系统中实际占用多少空间？

解析：第一问要计算混合索引结构的寻址空间大小；第二问只要计算出存储该文件索引块的大小，然后加上该文件本身的大小即可。

1) 物理块大小为 $4KB$ ，数据大小为 $4B$ ，则每个物理块可存储的地址数为 $4KB/4B = 1024$ 。最大文件的物理块数可达 $10 + 1024 + 1024^2 + 1024^3$ ，每个物理块大小为 $4KB$ ，因此总长度为

$$(10 + 1024 + 1024^2 + 1024^3) \times 4KB = 40KB + 4MB + 4GB + 4TB$$

这个文件系统允许的最大文件长度是 $4TB + 4GB + 4MB + 40KB$ ，约为 $4TB$ 。

2) 占用空间分为文件实际大小和索引项大小，文件大小为 $2GB$ ，从1)中计算知，需要使用到二次间接索引项。该文件占用 $2GB/4KB = 512 \times 1024$ 个数据块。

一次简介索引项使用1个间接索引块，二次间接索引项使用

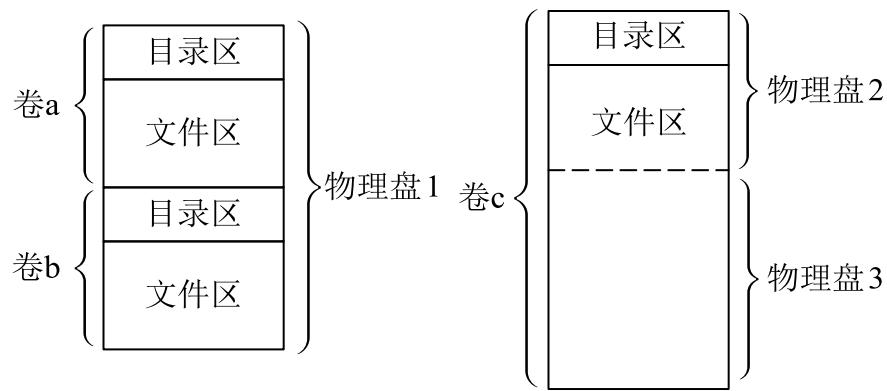
$1 + [(512 \times 1024 - 10 - 1024)/1024] \approx 512$ 个间接索引块(最左的1表示二次间址块)，所以间接索引块所占空间大小为 $(1 + 512) \times 4KB = 2MB + 4KB$ 另外每个文件使用的 i_node 数据结构占 $13 \times 4B = 52B$ ，因此该文件实际占用磁盘空间大

小为 $2GB + 2MB + 4KB + 52B$ 。

2. 文件存储空间管理

1) 文件存储器空间的划分与初始化。

一般来说，一个文件存储在一个文件卷中。文件卷可以是物理盘的一部分，也可以是整个物理盘，支持超大型文件的文件卷也可由多个物理盘组成。



逻辑卷与物理盘的关系

2) 存储空间管理

① 空闲表法

空闲表法属于连续分配方式，它与内存动态分配方式类似，为每个文件分配一块连续的存储空间。

空闲盘块表

序号	第一个空闲盘块号	空闲盘块数
1	2	4
2	9	3
3	15	5
4	-	-

如何分配磁盘块：与内存管理中的动态分区分配很类似，为一个文件分配连续的存储空间。同样可采用首次适应、最佳适应、最坏适应等算法来决定要为文件分

配哪个区间。

如何回收磁盘块:与内存管理中的动态分区分配很类似,当回收某个存储区时需要有四种情况。

- a. 回收区的前后都没有相邻空闲区;
- b. 回收区的前后都是空闲区;
- c. 回收区前面是空闲区;
- d. 回收区后面是空闲区。

总之,回收时需要注意表项的合并问题。

②空闲链表法

- a. 空闲盘块链:以盘块为单位组成一条空闲链。

操作系统保存着链头、链尾指针。

如何分配:若某文件申请 K 个盘块,则从链头开始依次摘下 K 盘块分配,并修改空闲链的链头指针。

如何回收:回收的盘块依次挂到链尾,并修改空闲链的链尾指针。

适用于离散分配的物理站构,为文件分配多个盘块时可能更主文多次操作。

- b. 空闲盘区链:以盘区为单位组成一条空闲链

操作系统保存着链头、链尾指针。

如何分配:若某文件申请 K 个盘块,则可以采用首次适应、最佳适应等算法,从链头开始检索按照算法规则找到一个大小符合要求的空闲盘区,分配给文件。若没有合适的连续空闲块,也可以将不同盘区的盘块同时分配给一个文件,注意分配后可能要修改相应的链指针、盘区大小等数据

如何回收:若回收区和某个空闲盘区相邻,则需要将回收区合并到空闲盘区中。

若回收区没有和任何空闲区相邻,将回收区作为单独的一个空闲盘区挂到链尾。

离散分配、连续分配都适用,为个文件分配多个盘块时效率更高

③位示图法

位示图,每个二进制位对应一个盘块。在本例中“0”代表盘块空闲,“1”代

表盘块已分配。

如何分配：若文件需要 K 个块，先进行顺序扫描位示图，找到 K 个相邻或不相邻的“0”，再根据字号、位号算出对应的盘块号，将相应盘块分配给文件。最后将相应位设置为“1”。

若 n 表示字长，则 (字号, 位号) = (i, j) 的二进制位对应的盘块号 $b = ni + j$ 。

b 号盘块对应的字号 $i = b/n$ ；位号 $j = b \% n$

如何回收：先根据回收的盘块号计算出对应的字号、位号，再将相应二进制位设为“0”

④成组链接法

空闲表法、空闲链表法不适用于大型文件系统，因为空闲表或空闲链表可能过大。
UNIX 系统中采用了成组链接法对磁盘空闲块进行管理。

文件卷的目录区中专门用一个磁盘块作为“超级块”，当系统启动时需要将超级块读入内存。并且要保证内存与外存中的“超级块”数据一致。

题 1. 一计算机系统利用位示图来管理磁盘文件空间，假定该磁盘组共有 100 个柱面，每个柱面有 20 个磁道，每个磁道分成 8 个盘块(扇区)，每个盘块 1KB，位示图如下图所示。

$i \backslash j$	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	
2	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	
3	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
...																

1) 试给出位示图中位置 (i, j) 与对应盘块所在的物理位置(柱面号, 磁头号, 扇区

号)之间的计算公式。假定柱面号、磁头号、扇区号都从0开始编号。

2) 试说明分配和回收一个盘块的过程。

解析:

1) 根据位示图的位置 (i,j) , 得出盘块的序号 $b = i \times 16 + j$; 用 C 表示柱面号, H 表示磁头号, S 表示扇区号, 则有

$$C = b / (20 \times 8), \quad H = (b \% (20 \times 8)) / 8, \quad S = b \% 8$$

2) 分配: 顺序扫描位示图, 找出1个其值为“0”的二进制位(“0”表示空闲), 利用上述公式将其转换成相应的序号 b , 并修改位示图, 置 $(i,j)=1$ 。

回收: 将回收盘块的盘块号换算成位示图中的 i 和 j , 转换公式为

$$b = C \times 20 \times 8 + H \times 8 + S, \quad i = b / 16, \quad j = b / 16$$

最后将计算出的 (i,j) 在位示图中置“0”。

题2. 假定一个盘组共有100个柱面, 每个柱面上有16个磁道, 每个磁道分成4个

扇区。

1) 整个磁盘空间共有多少个存储块?

2) 若用字长32位的单元来构造位示图, 共需要多少个字?

3) 位示图中第18个字的第16位对应的块号是多少?

解析:

1) 整个磁盘空间的存储块数目为 $4 \times 16 \times 100 = 6400$ 个。

2) 位示图应为6400个位, 如果用字长为32位(即 $n=32$)的单元来构造位示图, 那么需要 $6400 / 32 = 200$ 个字。

3) 位示图中第18个字的第16位(即 $i=18, j=16$)对应的块号为
 $32 \times (18 - 1) + 16 = 560$

课时十五 练习题

1. 下列文件物理结构中，适合随机访问且易于文件扩展的是（ ）。
 - A. 连续结构
 - B. 索引结构
 - C. 链式结构且磁盘块定长
 - D. 链式结构且磁盘块变长

2. 设文件索引结点中有7个地址项，其中4个地址项是直接地址索引，2个地址项是一级间接地址索引，1个地址项是二级间接地址索引，每个地址项大小为 $4B$ ，若磁盘索引块和磁盘数据块大小均为 $256B$ ，则可表示的单个文件最大长度是（ ）
 - A. $33KB$
 - B. $519KB$
 - C. $1057KB$
 - D. $16516KB$

3. 在文件的索引结点中存放直接索引指针10个，一级和二级索引指针各1个。磁盘块大小为 $1KB$ ，每个索引指针占 $4B$ 。若某文件的索引结点已在内存中，则把该文件偏移量(按字节编址)为1234 和307400 处所在的磁盘块读入内存，需访问的磁盘个数分别是（ ）。
 - A. 1, 2
 - B. 1, 3
 - C. 2, 3
 - D. 2, 4

4. 文件系统用位图法表示磁盘空间的分配情况，位图存于磁盘的32 ~ 127号块中，每个盘块占 $1024B$ ，盘块和块内字节均从0开始编号。假设要释放的盘块号为

409612，则位图中要修改的位所在的盘块号和块内字节序号分别是（ ）。

- A. 81, 1 B. 81, 2 C. 82, 1 D. 82, 2

5. 下列选项中，可用于文件系统管理空闲磁盘块的数据结构是（ ）。

- I . 位图
 - II. 索引结点
 - III. 空闲磁盘块链
 - IV. 文件分配表(*FAT*)
- A. 仅 I 、 II
B. 仅 I 、 III、 IV
C. 仅 I 、 III
D. 仅 II 、 III、 IV

6. 文件采用多重索引结构搜索文件内容。设块长为 $512B$ ，每个块号长 $2B$ ，若不考虑逻辑块号在物理块中所占的位置，分别计算二级索引和三级索引时可寻址的文件最大长度。

7. 文件 *F* 由 200 条记录组成，记录从 1 开始编号，用户打开文件后，欲将内存中的一条记录插入文件 *F* 中，作为其第 30 条记录。请回答下列问题，并说明理由。

- 1) 若文件系统采用连续分配方式，每个磁盘块存放一条记录，文件 *F* 存储区域前后均有足够的空闲磁盘空间，则完成上述插入操作最少需要访问多少次磁盘块？*F* 的文件控制块内容会发生哪些改变？
- 2) 若文件系统采用链接分配方式，每个磁盘块存放一条记录和一个链接指针，则完成上述插入操作需要访问多少次磁盘块？若每个存储块大小为 $1KB$ ，其中 $4B$ 存放链接指针，则该文件系统支持的文件最大长度是多少？

课时十六 文件管理（三）

考点	重要程度	占分	题型
1. 文件的基本操作	★★	1~3	选择、填空
2. 文件系统的层次结构	★	1~3	选择、填空

1. 文件的基本操作

文件属于抽象数据类型，为了恰当地定义文件，需要考虑有关文件的操作。操作系统提供系统调用，它对文件进行创建、写、读、重定位、删除和截断等操作。

1) 创建文件。

创建文件有两个必要步骤：一是在文件系统中为文件找到空间；二是在目录中为新文件创建条目，该条目记录文件名称、在文件系统中的位置及其他可能的信息。

2) 写文件。

为了写文件，执行一个系统调用，指明文件名称和要写入文件的内容。对于给定文件名称，系统搜索目录以查找文件位置。系统必须为该文件维护一个写位置的指针。

3) 读文件。

为了读文件，执行一个系统调用，指明文件名称和要读入文件块的内存位置。同样，需要搜索目录以找到相关目录项，系统维护一个读位置的指针。每当发生读操作时，更新读指针。一个进程通常只对一个文件读或写，因此当前操作位置可作为每个进程当前文件位置的指针。由于读和写操作都使用同一指针，因此节省了空间，也降低了系统复杂度。

4) 文件重定位(文件寻址)。

按某条件搜索目录，将当前文件位置设为给定值，并且不会读、写文件。

5) 删除文件。

先从目录中找到要删除文件的目录项，使之成为空项，然后回收该文件所占用的存储空间。

6) 截断文件。

允许文件所有属性不变，并删除文件内容，即将其长度设为0并释放其空间。这6个基本操作可以组合起来执行其他文件操作。例如，一个文件的复制，可以创建新文件、从旧文件读出并写入新文件。

7) 打开文件。

在很多操作系统中，在对文件进行操作之前，要求用户先使用*open*系统调用打开文件，需要提供的几个主要参数：

- ①文件存放路径(“*D:/Demo*”)
- ②文件名(“*text.txt*”)
- ③要对文件的操作类型(如:*r*只读：*rw*读写等)

操作系统在处理*open*系统调用时，主要做了几件事：

- ①根据文件存放路径找到相应的目录文件，从目录中找到文件名对应的的目录项，并检查该用户是否有指定的操作权限。
- ②将目录项复制到内存中的“打开文件表”中。并将对应表目的编号返回给用户。之后用户使用打开文件表的编号来指明要操作的文件。

8) 关闭文件

进程使用完文件后，要“关闭文件”。操作系统在处理*Close*系统调用时，主要做了几件事：

- ①将进程的打开文件表相应表项删除；
- ③回收分配给该文件的内存空间等资源；
- ④系统打开文件表的打开计数器*count*减1，若*count = 0*，则删除对应表项。

2. 文件系统的层次结构

用户/应用程序		文件系统需要向上层的用户提供一些简单易用的功能接口。这层就是用于处理用户发出的系统调用请求(<i>Read</i> 、 <i>Write</i> 、 <i>Open</i> 、 <i>Close</i> 等系统调用)		
文件目录系统		用户是通过文件路径来访问文件的，因此这一层需要根据用户给出的文件路径找到相应的 <i>FCB</i> 或索引结点，所有和目录、目录项相关的管理工作都在本层完成，如：管理活跃的文件目录表、管理打开文件表等。		
存取控制模块		为了保证文件数据的安全，还需要验证用户是否有访问权限。这一层主要完成了文件保护相关功能		
逻辑文件系统与文件信息缓冲区		用户指明想要访问文件记录号，这一层需要将记录号转换为对应的逻辑地址		
物理文件系统		这一层需要把上一层提供的文件逻辑地址转换为实际的物理地址		
辅助分配模块	负责文件存储空间的管理即负责分配和回收存储空间	设备管理模块	直接与硬件交互，负责和赢家直接相关的管理工作，如：分配设备、分配设备缓冲区、磁盘调度、启动设备、释放设备等	

举例：

假设某用户请求删除文件“D:/工作目录/学生信息.xlsx”的最后100条记录。

- ① 用户需要通过操作系统提供的接口发出上述请求—用户接口；
- ② 由于用户提供的是文件的存放路径，因此需要操作系统一层一层地查找目录，找到对应的目录项—文件目录系统；
- ③ 不同的用户对文件有不同的操作权限，因此为了保证安全，需要检查用户是否有访问权限—存取控制模块(存取控制验证层)；
- ④ 验证了用户的访问权限之后，需要把用户提供的“记录号”转变为对应的逻

辑地址--逻辑文件系统与文件信息缓冲区；

- ⑤ 知道了目标记录对应的逻辑地址后，还需要转换成实际的物理地址--物理文件系统；
- ⑥ 要删除这条记录，必定要对磁盘设备发出请求--设备管理程序模块；
- ⑦ 删 除这些记录后，会有一些盘块空闲，因此要将这些空闲盘块回收--辅助分配模块。

题 1. 打开文件操作的主要工作是()

- A. 把指定文件的目录复制到内存指定的区域
- B. 把指定文件复制到内存指定的区域
- C. 在指定文件所在的存储介质上找到指定文件的目录
- D. 在内存寻找指定的文件

答案： A

解析：打开文件操作是将该文件的FCB存入内存的活跃文件目录表，而不是将文件内容复制到主存，找到指定文件目录是打开文件之前的操作。

题 2. 用户在删除某文件的过程中，操作系统不可能执行的操作是()。

- A. 删除此文件所在的目录
- B. 删除此文件关联的目录项
- C. 删除与此文件对应的文件控制块
- D. 释放与此文件关联的内存缓冲区

答案： A

解析：此文件所在目录下可能还存在其他文件，因此删除文件时不能（也不需要）删除文件所在的目录，而与此文件关联的目录项和文件控制块需要随着文件一同删除，同时释放文件关联的内存缓冲区。

课时十六 练习题

1. 打开的文件使用完毕后，应该进行（ ）操作。
A. 备份 B. 重命名 C. 关闭 D. 删除

2. 判断题：文件打开操作的目的是建立用户与文件之间的联系。（ ）

3. 目录上的主要操作有（ ），（ ），（ ），（ ）和（ ）。

4. 文件系统在创建一个文件时，为文件建立一个（ ）。
A. 文件目录 B. 目录文件 C. 逻辑结构 D. 逻辑空间

5. 简述“建立文件”操作的系统处理过程。

6. 下面5种独立的操作：
 - 1) 将文件的目录信息读入活动文件表
 - 2) 向设备管理程序发出 *I/O* 请求，完成数据读入操作
 - 3) 指出文件在外存上的存储位置，并进行文件逻辑块号到物理块的转换
 - 4) 按存取控制说明检查访问的合法性
 - 5) 按文件名查找活动文件表关于读文件次序的正确描述是（ ）。
*A. 51432
B. 15432
C. 41532
D. 53241*

· 课时十七 磁盘的组织与管理

考点	重要程度	占分	题型
1. 磁盘	★★	1~2	选择、应用
2. 磁盘调度算法	必考	3~5	选择、应用
3. 磁盘初始化	★	1~2	选择、应用

1. 磁盘

磁盘 (*Disk*) 是由表面涂有磁性物质的金属或塑料构成的圆形盘片，通过一个称为磁头的导体线圈从磁盘存取数据。在读/写操作期间，磁头固定，磁盘在下面高速旋转。

如图1所示，磁盘盘面上的数据存储在一组同心圆中，称为磁道。每个磁道与磁头一样宽，一个盘面有上千个磁道。磁道又划分为几百个扇区，每个扇区固定存储大小(通常为512B)，一个扇区称为一个盘块。相邻磁道及相邻扇区间通过一定的间隙分隔开，以避免精度错误。注意，由于扇区按固定圆心角度划分，所以密度从最外道向里道增加，磁盘的存储能力受限于最内道的最大记录密度。

磁盘安装在一个磁盘驱动器中，它由有磁头臂、用于旋转磁盘的主轴和用于数据输入/输出的电子设备组成。如图所示，多个盘片垂直堆叠，组成磁盘组，每个盘面对应一个磁头所有磁头固定在一起，与磁盘中心的距离相同且一起移动。所有盘片上相对位置相同的磁道组成柱面。按照这种物理结构组织，扇区就是磁盘可寻址的最小存储单位，磁盘地址用“柱面号 · 盘面号 · 扇区号(或块号)”表示。

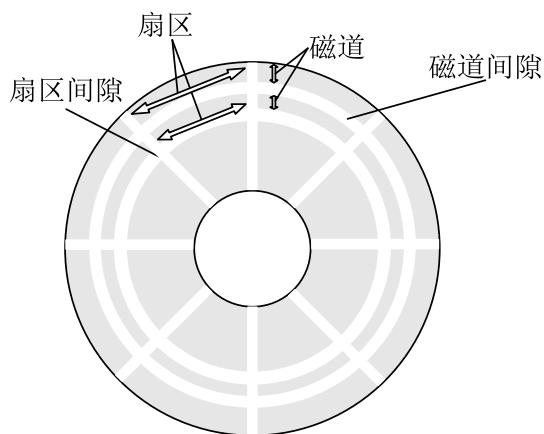


图1 磁盘盘片

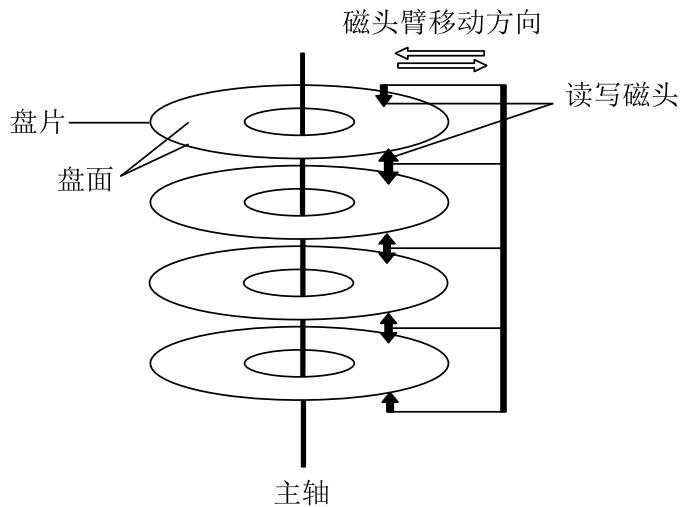


图2 磁盘驱动器

磁盘按不同的方式可分为若干类型：磁头相对于盘片的径向方向固定的，称为固定头磁盘，每个磁道一个磁头。磁头可移动的，称为活动头磁盘，磁头臂可来回伸缩定位磁道。磁盘永久固定在磁盘驱动器内的，称为固定盘磁盘，可移动和替换的，称为可换盘磁盘。

2. 磁盘调度算法

当有多个请求同时到达时，操作系统就要决定先为哪个请求服务，这就是磁盘调度算法要解决的问题。

一次磁盘读写操作的时间由寻找(寻道)时间、延迟时间和传输时间决定。

1) 寻找时间 T_s 。

活动头磁盘在读写信息前，将磁头移动到指定磁道所需要的时间。这个时间除跨越 n 条磁道的时间外，还包括启动磁臂的时间 s ，即

$$T_s = m * n + s$$

式中， m 是与磁盘驱动器速度有关的常数，约为 $0.2ms$ ，磁臂的启动时间为 $2ms$ 。

2) 延迟时间 T_r 。

磁头定位到某一磁道的扇区(块号)所需要的时间，设磁盘的旋转速度为 r ，则

$$T_r = 1/2r$$

对于硬盘，典型的旋转速度为5400转/分，相当于一周11.1ms，则 T_r 为5.55ms；

对于软盘，其旋转速度为300~600转/分，则 T_r 为50~100ms。

3) 传输时间 T_t 。

从磁盘读出或向磁盘写入数据所经历的时间，这个时间取决于每次所读/写的字节数 b 和磁盘的旋转速度：

$$T_t = b/rN$$

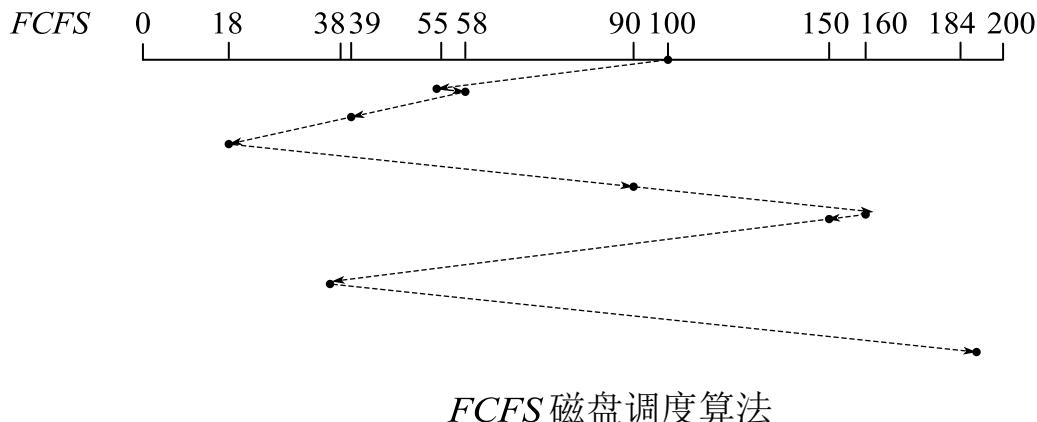
式中， r 为磁盘每秒的转数， N 为一个磁盘上的字节数。

在磁盘存取时间的计算中，寻找时间与磁盘调度算法相关，下面介绍几种算法：

目前常用的磁盘调度算法有以下几种：

a. 先来先服务(FCFS)算法

FCFS 算法根据进程请求访问磁盘的先后顺序进行调度。这是一种最简单的调度算法。



磁盘请求队列中的请求顺序分别为55, 58, 39, 18, 90, 160, 150, 38, 184，磁头的初始位置是磁道100，采用FCFS算法时磁头的运动过程如上图所示。

磁头共移动了 $(45 + 3 + 19 + 21 + 72 + 70 + 10 + 112 + 146) = 498$ 个磁道，平均寻找长度 $498/9 = 55.3$ 。

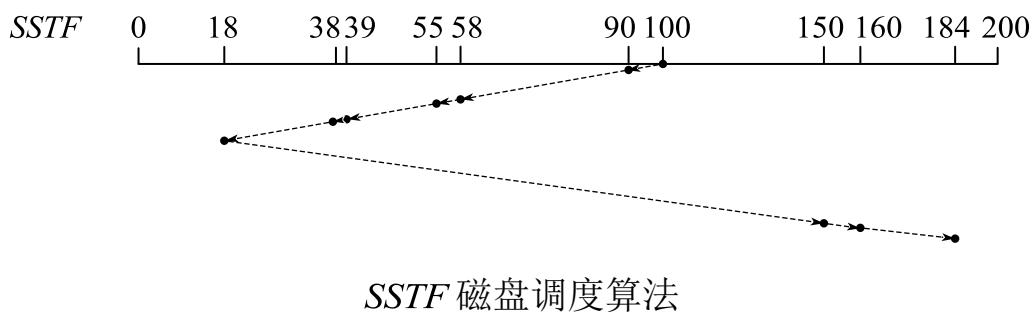
b. 最短寻找时间优先(*Shortest Seek Time First, SSTF*)算法

SSTF 算法选择调度处理的磁道是与当前磁头所在磁道距离最近的磁道，以便使每次的寻找时间最短。当然，总是选择最小寻找时间并不能保证平均寻找时间最小，但能提供比*FCFS* 算法更好的性能。

但这种算法会产生“饥饿”现象。

如图所示，若某时刻磁头正在18号磁道，而在18号磁道附近频繁地增加新的请求，则*SSTF* 算法使得磁头长时间在18号磁道附近工作，将使184号磁道附近的访问被无限期地延迟，即被“饿死”。

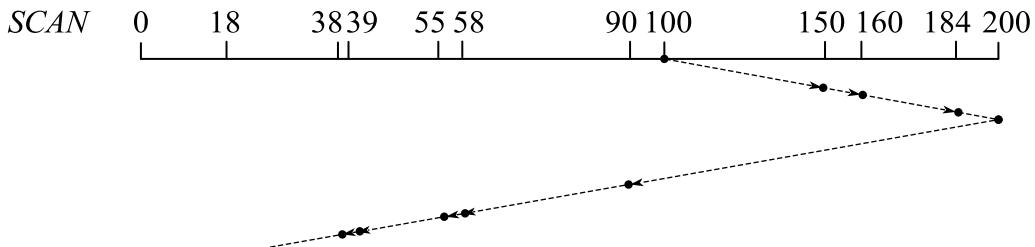
例如磁盘请求队列中的请求顺序分别为55, 58, 39, 18, 90, 160, 150, 38, 184，磁头的初始位置是磁道100，采用*SSTF* 算法时磁头的运动过程如下图所示。磁头共移动了 $10 + 32 + 3 + 16 + 1 + 20 + 132 + 10 + 24 = 248$ 个磁道，平均寻找长度 $= 248/9 = 27.5$ 。



c. 扫描(*SCAN*)算法（又称电梯调度算法）

SCAN 算法在磁头当前移动方向上选择与当前磁头所在磁道距离最近的请求作为下一次服务的对象，实际上就是在最短寻找时间优先算法的基础上规定了磁头运动的方向，如下图所示。由于磁头移动规律与电梯运行相似，因此又称电梯调度算法。*SCAN* 算法对最近扫描过的区域不公平，因此它在访问局部

方面不如FCFS算法和SSTF算法好。

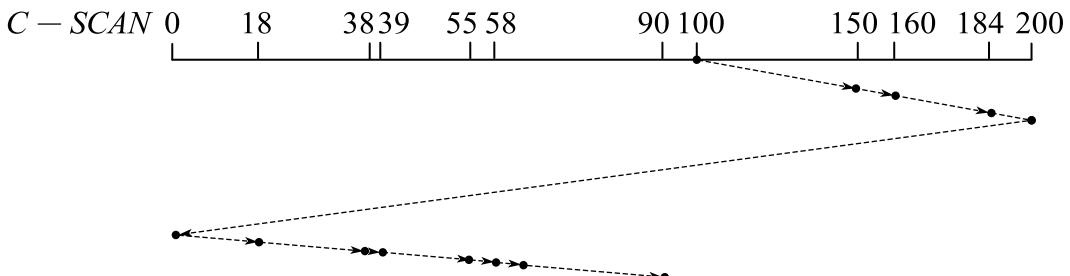


SCAN磁盘调度算法

例如：磁盘请求队列中的请求顺序分别 $55, 58, 39, 18, 90, 160, 150, 38, 184$ ，磁头的初始位置是磁道100。采用SCAN算法时，不但要知道磁头的当前位置，而且要知道磁头的移动方向，假设磁头沿磁道号增大的顺序移动，则磁头的运动过程如上图所示。移动磁道的顺序为 $100, 150, 160, 184, 200, 90, 58, 55, 39, 38, 18$ 磁头共移动了 $(50 + 10 + 24 + 16 + 110 + 32 + 3 + 16 + 1 + 20) = 282$ 个磁道，平均寻道长度 $= 282/9 = 31.33$ 。

d. 循环扫描(Circular SCAN, C-SCAN)算法

在扫描算法的基础上规定磁头单向移动来提供服务，回返时直接快速移动至起始端而不服务任何请求。由于SCAN算法偏向于处理那些接近最里或最外的磁道的访问请求，所以使用改进型的C-SCAN算法来避免这个问题，如图所示。

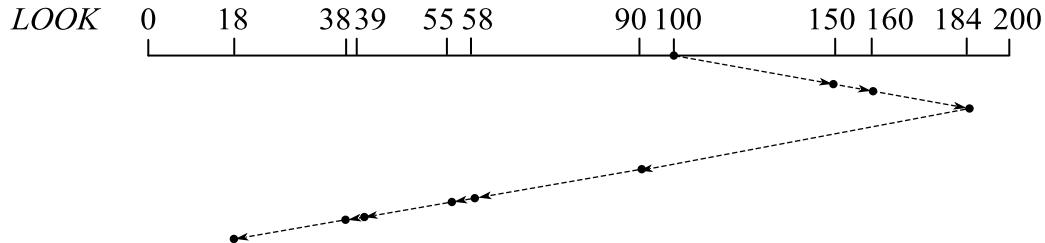


C-SCAN磁盘调度算法

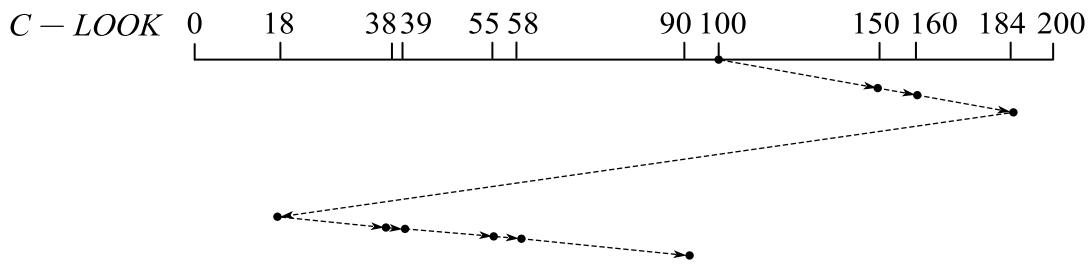
采用SCAN算法和C-SCAN算法时，磁头总是严格地遵循从盘面的一端到另一端，显然，在实际使用时还可以改进，即磁头移动只需要到达最远端的

一个请求即可返回，不需要到达磁盘端点。

这种形式 $SCAN$ 算法和 $C - SCAN$ 算法称为 $LOOK$ 调度和 $C - LOOK$ 调度，因为它们在朝一个给定方向移动前会查看是否有请求。



$LOOK$ 磁盘调度算法



$C - LOOK$ 磁盘调度算法

注意，若无特别说明，也可以默认 $SCAN$ 算法和 $C - SCAN$ 算法为 $LOOK$ 调度和 $C - LOOK$ 调度算法。

例如磁盘请求队列中的请求顺序为 $55, 58, 39, 18, 90, 160, 150, 38, 184$ ，

磁头初始位置是磁道 100 。采用 $C - SCAN$ 算法时，假设磁头沿磁道号增大的顺序移动，则磁头的运动过程如上图所示。移动磁道的顺序为 $100, 150, 160, 184, 200, 0, 18, 38, 39, 55, 58, 90$ 。磁头共移动 $50 + 10 + 24 + 16 + 200 + 18 + 20 + 1 + 16 + 3 + 32 = 390$ 个磁道，平均寻道长度 $- 390/9 = 43.33$ 。

e. 磁盘调度算法比较

	优点	缺点
FCFS 算法	公平、简单	平均寻道距离大，仅应用在磁盘 I/O 较少的场合
SSTF 算法	性能比“先来先服务”好	不能保证平均寻道时间最短，可能出现“饥饿”现象
SCAN 算法	寻道性能好，可避免“饥饿”现象	不利于远离磁头一端的访问请求
C-SCAN 算法	消除了对两端磁道请求的不公平	—

题 1. 在一个磁盘上，有1000个柱面，编号为0~999，用下面的算法计算为满足磁盘队列中的所有请求，磁盘臂必须移过的磁道的数目。假设最后服务的请求是在磁道345上，并且读写头正在朝磁道0移动。在按FIFO顺序排列的队列中包含了如下磁道上的请求：123, 874, 692, 475, 105, 376。

- 1) FIFO； 2) SSTF； 3) SCAN； 4) LOOK； 5) C-SCAN；
 6) C-LOOK。

解答：

1) FIFO：移动磁道的顺序为345, 123, 874, 692, 475, 105, 376。磁盘臂必须移过的磁道的数目为 $222 + 751 + 182 + 217 + 370 + 271 = 2013$ 。

2) SSTF：移动磁道的顺序为345, 376, 475, 692, 874, 123, 105。磁盘臂必须移过的磁道的数目为 $31 + 99 + 217 + 182 + 751 + 18 = 1298$ 。

注意：磁盘臂必须移过的磁道的数目之和的计算没有必要像上面一样对31, 99, 217, 182, 751, 18求和，从345到874是一路递增的，接着从874到105是一路递减的。所以仅需计算 $(874 - 345) + (874 - 105) = 1298$ 。这种方法是

不是要比上面得出 6 个数后再计算它们的和快捷一些？若之前未注意到此法，相信聪明的读者会马上回顾刚做完的 1)，并会仔细观察以下几问的“规律”，进而总结出自己的思路。

3) *SCAN*: 移动磁道的顺序为 $345, 123, 105, 0, 376, 475, 692, 874$ 。磁盘臂必须移过的磁道的数目为 $222 + 18 + 105 + 376 + 99 + 217 + 182 = 1219$ 。

4) *LOOK*: 移动磁道的顺序为 $345, 123, 105, 376, 475, 692, 874$ 。磁盘臂必须移过的磁道的数目为 $222 + 18 + 271 + 99 + 217 + 182 = 1109$ 。

5) *C-SCAN*: 移动磁道的顺序为 $345, 123, 105, 999, 874, 692, 475, 376$ 。磁盘臂必须移过的磁道的数目为

$$222 + 18 + 105 + 999 + 125 + 182 + 217 + 99 = 1967。$$

6) *C-LOOK*: 移动磁道的顺序为 $345, 123, 105, 874, 692, 475, 376$ 。磁盘臂必须移过的磁道的数目为 $222 + 18 + 769 + 182 + 217 + 99 = 1507$ 。

3. 磁盘的管理

1) 磁盘初始化

Step 1: 进行低级格式化（物理格式化），将磁盘的各个磁道划分为扇区，一个扇区通常可分为头、数据区域（如 $512B$ 大小）、尾三个部分组成，管理扇区所需要的各种数据结构一般存放在头、尾两个部分，包括扇区校验码（如奇偶校验、*CRC* 循环冗余校验码等，校验码用于校验扇区中的数据是否发生错误。）

Step 2: 将磁盘分区，每个分区由若干柱面组成（即分为我们熟悉的 C 盘、D 盘、E 盘）。

Step 3: 进行逻辑格式化，创建文件系统，包括创建文件系统的根目录、初始

化存储空间管理所用的数据结构（如位示图、空闲分区表）。

2) 引导块

计算机启动时需要运行一个初始化程序（自举程序），它初始化*CPU*、寄存器、设备控制器和内存等，接着启动操作系统。自举程序通常保存在*ROM*中，为了避免改变自举代码而需要改变*ROM*硬件的问题，因此只在*ROM*中保留很小的自举装入程序，保存在磁盘的启动块上，启动块位于磁盘的固定位。

3) 坏块

无法正常使用的扇区就是“坏块”。这属于硬件故障，操作系统是无法修复的，应该将坏块标记出来，以免错误地使用。

保留一些“备用扇区”，用于替换坏块，称为扇区备用，坏块对操作系统透明。

课时十七 练习题

1. 磁盘是可共享设备，但在每个时刻（ ）作业启动它。
A. 可以由任意多个 *B.* 能限定多个 *C.* 至少能由一个 *D.* 至多能由一个

2. 下列算法中，用于磁盘调度的是（ ）。
A. 时间片轮转调度算法 *B.* LRU 算法
C. 最短寻找时间优先算法 *D.* 优先级高者优先算法

3. 以下算法中，（ ）可能出现“饥饿”现象。
A. 电梯调度 *B.* 最短寻找时间优先 *C.* 循环扫描算法 *D.* 先来先服务

4. 在以下算法中，（ ）可能会随时改变磁头的运动方向。
A. 电梯调度 *B.* 先来先服务 *C.* 循环扫描算法 *D.* 以上答案都不会

5. 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. 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

6. 某硬盘有200个磁道（最外侧磁道号为0），磁道访问请求序列为
130, 42, 180, 15, 199，当前磁头位于第58号磁道并从外侧向内侧移动。按照
*SCAN*调度方法处理完上述请求后，磁头移过的磁道数是（ ）。

- A. 208 B. 287 C. 325 D. 382

7. 某系统采用循环扫描磁盘调度策略，某时刻磁头位于100号磁道，并沿磁道
号增大方向移动，磁道号的请求队列为：50, 90, 30, 120，请列出磁道访问的
顺序，每次磁头移动的磁道数，并计算平均寻道长度。

课时十八 I/O 设备管理（一）

考点	重要程度	占分	题型
1. I/O 设备的基本概念和分类	★★	1~2	选择
2. I/O 控制方式	必考	3~5	选择、简答

1. I/O 设备的基本概念和分类

计算机系统中的 I/O 设备按使用特性可分为以下类型：

- 1) 人机交互类外部设备，用于与计算机用户之间交互的设备，如打印机、显示器、鼠标、键盘等。这类设备的数据交换速度相对较慢，通常是以字节为单位进行数据交换的。
- 2) 存储设备，用于存储程序和数据的设备，如磁盘、磁带、光盘等。这类设备用于数据交换，速度较快，通常以多字节组成的块为单位进行数据交换。
- 3) 网络通信设备，用于与远程设备通信的设备，如各种网络接口、调制解调器等。其速度介于前两类设备之间。网络通信设备在使用和管理上与前两类设备也有很大不同。

除了上面最常见的分类方法，I/O 设备还可以按以下方法分类。

1) 按传输速率分类

- ①低速设备，传输速率仅为每秒几字节到数百字节的一类设备，如键盘、鼠标等。
- ②中速设备，传输速率为每秒数千字节至数万字节的一类设备，如行式打印机、激光打印机等。
- ③高速设备，传输速率在数百千字节至千兆字节的一类设备，如磁带机、磁盘机、光盘机等。

2) 按信息交换的单位分类

- ①块设备，由于信息的存取总是以数据块为单位的，所以存储信息的设备称为块设备。它属于有结构设备，如磁盘等。

磁盘设备的基本特征是传输速率较高、可寻址，即对它可随机地读/写任一块。

- ②字符设备，用于数据输入/输出的设备为字符设备，因为其传输的基本单位是

字符。它属于无结构类型，如交互式终端机、打印机等。

它们的基本特征是传输速率低、不可寻址，并且在输入/输出时常采用中断驱动方式。

题 1. 以下关于设备属性的叙述中，正确的是（ ）。

- A. 字符设备的基本特征是可寻址到字节，即能指定输入的源地址或输出的目标地址。
- B. 共享设备必须是可寻址的和可随机访问的设备。
- C. 共享设备是指同一时间内允许多个进程同时访问的设备。
- D. 在分配共享设备和独占设备时都可能引起进程死锁。

答案：B

解析：可寻址是块设备的基本特征，A 不正确，共享设备是指一段时间内允许多个进程同时访问的设备，因此C 选项不正确，分配共享设备时不会引起进程死锁的。D 选项不正确。

2. I/O 控制方式

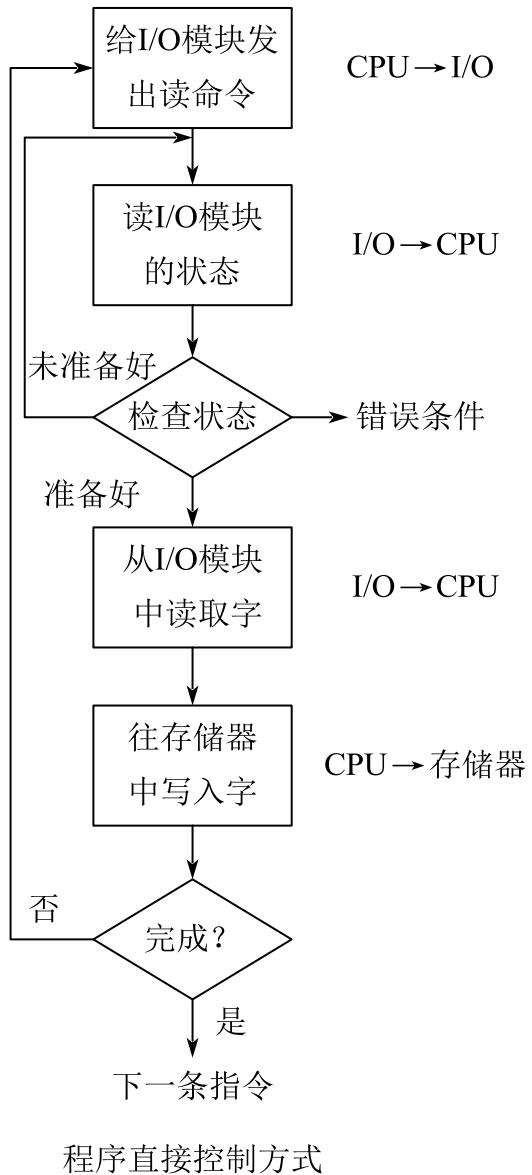
设备管理的主要任务之一是控制设备和内存或处理机之间的数据传送。外围设备和内存之间的输入/输出控制方式有4种，下面分别加以介绍：

1) 程序直接控制方式

如图所示，计算机从外部设备读取数据到存储器，每次读一个字的数据。对读入的每个字，CPU 需要对外设状态进行循环检查，直到确定该字已经在 I/O 控制器的数据寄存器中。在程序直接控制方式中，由于 CPU 的高速性和 I/O 设备的低速性，致使 CPU 的绝大部分时间都处于等待 I/O 设备完成数据 I/O 的循环测试中，造成了 CPU 资源的极大浪费。

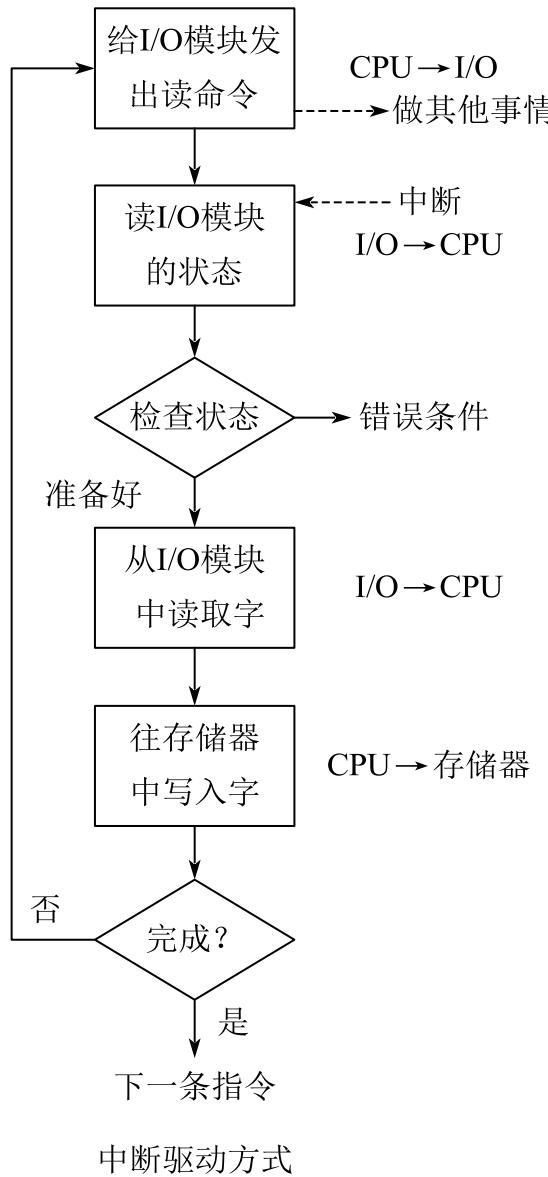
程序直接控制方式虽然简单且易于实现 但其缺点也显而易见，由于 CPU 和

*I/O*设备只能串行工作，导致*CPU*的利用率相当低。



2) 中断驱动方式

中断驱动方式的思想是，允许*I/O*设备主动打断*CPU*的运行并请求服务，从而“解放”*CPU*，使得其向*I/O*控制器发送读命令后可以继续做其他有用的工作。如图所示，我们从*I/O*控制器和*CPU*两个角度分别来看中断驱动方式的工作过程。



中断驱动方式比程序直接控制方式有效，但由于数据中的每个字在存储器与 *I/O* 控制器之间的传输都必须经过 *CPU*，这就导致了中断驱动方式仍然会消耗较多的 *CPU* 时间。

3) DMA 方式

在中断驱动方式中，*I/O* 设备与内存之间的数据交换必须经过 *CPU* 中的寄存器，所以速度还是受限，而 *DMA* (直接存储器存取) 方式的基本思想是在 *I/O* 设备和内存之间开辟直接的数据交换通路，彻底“解放” *CPU*。*DMA* 方式的特点如下：

- ② 基本单位是数据块；
- ② 所传送的数据，是从设备直接送入内存的，或者相反；

③仅在传送一个或多个数据块的开始和结束时，才需 *CPU* 干预，整块数据的传送是在 *DMA* 控制器的控制下完成的。

要在主机与控制器之间实现成块数据的直接交换，须在 *DMA* 控制器中设置如下 4 类寄存器。

①命令/状态寄存器 (*CR*)。

用于接收从 *CPU* 发来的 *I/O* 命令或有关控制信息，或设备的状态。

②内存地址寄存器 (*MAR*)。

在输入时，它存放把数据从设备传送到内存的起始目标地址：

在输出时，它存放由内存到设备的内存源地址。

③数据寄存器 (*DR*)。

用于暂存从设备到内存或从内存到设备的数据。

④数据计数器 (*DC*)。存放本次要传送的字(节)数。

DMA 方式的工作过程如下：

CPU 接收到 *I/O* 设备的 *DMA* 请求时，它给 *I/O* 控制器发出一条命令，启动 *DMA* 控制器，然后继续其他工作。之后 *CPU* 就把控制操作委托给 *DMA* 控制器，由该控制器负责处理。*DMA* 控制器直接与存储器交互，传送整个数据块，每次传送一个字，这个过程不需要 *CPU* 参与。传送完成后，*DMA* 控制器发送一个中断信号给处理器。

DMA 控制方式与中断驱动方式的主要区别：

中断驱动方式在每个数据需要传输时中断 *CPU*，而 *DMA* 控制方式则是在所要求传送的一批数据全部传送结束时才中断 *CPU*；此外，中断驱动方式数据传送是在中断处理时由 *CPU* 控制完成的，而 *DMA* 控制方式则是在 *DMA* 控制器的控制下完成的。

4) 通道控制方式

*I/O*通道是指专门负责输入/输出的处理机。

*I/O*通道方式是*DMA*方式的发展，它可以进一步减少*CPU*的干预，即把对一个数据块的读或写)为单位的干预，减少为对一组数据块的读(或写)及有关控制和管理为单位的干预。同时又可以实现*CPU*、通道和*I/O*设备三者的并行操作，从而更有效地提高整个系统的资源利用率。

*I/O*通道与*DMA*方式的区别：*DMA*方式需要*CPU*来控制传输的数据块大小、传输的内存位置，而通道方式中这些信息是由通道控制的。另外，每个*DMA*控制器对应一台设备与内存传递数据，而一个通道可以控制多台设备与内存的数据交换。

题 1. 系统将数据从磁盘读到内存的过程包括以下操作：

- ① *DMA*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()。

- A. ③→①→②→④
- B. ②→③→①→④
- C. ②→①→③→④
- D. ①→②→④→③

答案：B

解析：在开始*DMA*传输时，主机向内存写入*DMA*命令块，向*DMA*控制器写入该命令块的地址，启动*I/O*设备。然后，*CPU*继续其他工作，*DMA*控制器则继续直接操作内存总线，将地址放到总线上开始传输。整个传输完成后，*DMA*控制器中断*CPU*。因此执行顺序是2, 3, 1, 4，选B。

题 2. *I/O* 中断是 *CPU* 与通道协调工作的一种手段，所以在（ ）时，便要产生中断。

- A. *CPU* 执行“启动 *I/O*”指令而被通道拒绝接收
- B. 通道接收了 *CPU* 的启动请求
- C. 通道完成了通道程序的执行
- D. 通道在执行通道程序的过程中

答案：C

解析：*CPU* 启动通道时不管启动成功与否，通道都要回答 *CPU*，通道在执行通道程序的过程中，*CPU* 与通道并行，通道完成通道程序的执行后，便发 *I/O* 中断向 *CPU* 报告。

课时十八 练习题

1. 磁盘设备的 I/O 控制主要采取()方式。

- A. 位 B. 字节 C. 帧 D. DMA

2. 在设备控制器中用于实现设备控制功能的是()。

- A. CPU B. 设备控制器与处理器的接口
C. I/O 逻辑 D. 设备控制器与设备的接口

3. DMA 方式是在()之间建立一条直接数据通路。

- A. I/O 设备和主存 B. 两个 I/O 设备
C. I/O 设备和 CPU D. CPU 和主存

4. 通道又称 I/O 处理机，它用于实现()之间的信息传输。

- A. 内存与外设 B. CPU 与外设
C. 内存与外存 D. CPU 与外存

5. 在操作系统中，()指的是一种硬件机制。

- A. 通道技术 B. 缓冲池
C. SPOOLing 技术 D. 内存覆盖技术

6. 若 I/O 设备与存储设备进行数据交换不经过 CPU 来完成，则这种数据交换方式是()。

- A. 程序查询 B. 中断方式
C. DMA 方式 D. 无条件存取方式

7. 计算机系统中，不属于DMA控制器的是()。

- A. 命令/状态寄存器
- B. 内存地址寄存器
- C. 数据寄存器
- D. 堆栈指针寄存器

8. DMA方式与中断控制方式的主要区别是什么？DMA方式与通道方式的主要区别是什么？

课时十九 I/O 设备管理（二）

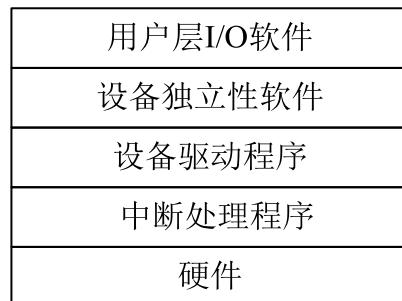
考点	重要程度	占分	题型
1. I/O 软件层次结构	★	1~2	选择
2. I/O 核心子系统	★	1~2	选择

1. I/O 软件层次结构

为了使复杂的 I/O 软件具有清晰的结构、良好的可移植性和适应性，在 I/O 软件中普遍采用了层次式结构，将系统输入/输出功能组织成一系列的层次，每层都利用其下层提供的服务，完成输入/输出功能中的某些子功能，并屏蔽这些功能实现的细节，向高层提供服务。

一个比较合理的层次划分如图所示。整个 I/O 系统可以视为具有 4 个层次的系统结构，各层次及其功能如下：

- 1) 用户层 I/O 软件。实现与用户交互的接口，用户可直接调用在用户层提供的、与 I/O 操作有关的库函数，对设备进行操作。
- 2) 设备独立性软件。用于实现用户程序与设备驱动器的统一接口、设备命令、设备保护及设备分配与释放等，同时为设备管理和数据传送提供必要的存储空间。
设备独立性也称设备无关性，使得应用程序独立于具体使用的物理设备。
使用逻辑设备名的好处是：①增加设备分配的灵活性；②易于实现 I/O 重定向。
- 3) 设备驱动程序。与硬件直接相关，负责具体实现系统对设备发出的操作指令，驱动 I/O 设备工作的驱动程序。
- 4) 中断处理程序。用于保存被中断进程的 CPU 环境，转入相应的中断处理程序进行处理，处理完并恢复被中断进程的现场后，返回到被中断进程。
- 5) 硬件设备。I/O 设备通常包括一个机械部件和一个电子部件。



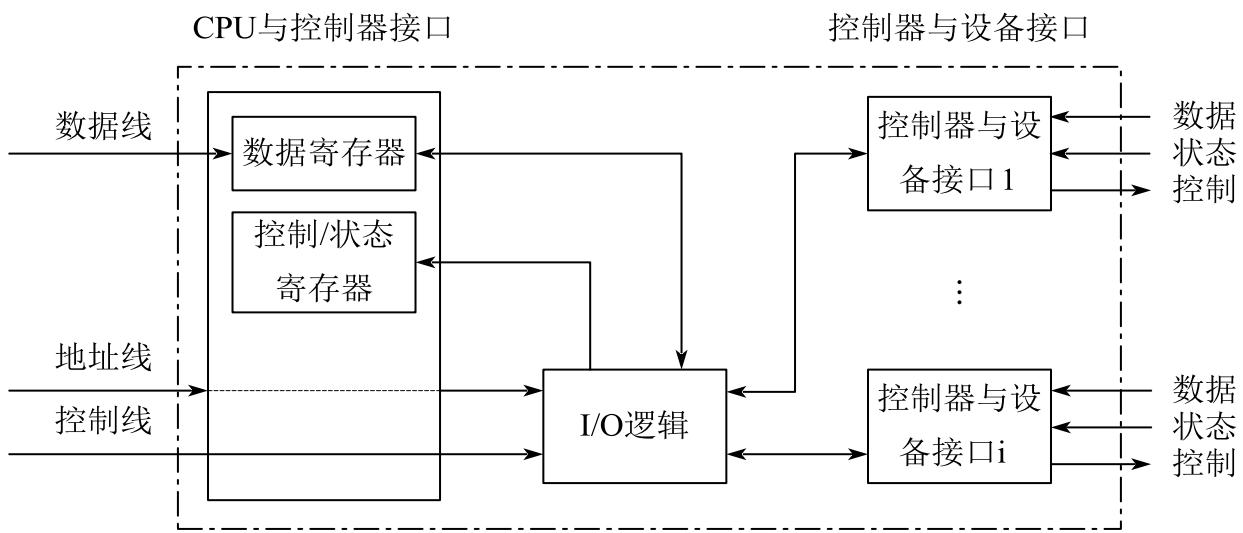
I/O层次结构

设备控制器的主要功能如下：

- 1) 接收和识别CPU或通道发来的命令，如磁盘控制器能接收读、写、查找等命令。
- 2) 实现数据交换，包括设备和控制器之间的数据传输；通过数据总线或通道，控制器和主存之间的数据传输。
- 3) 发现和记录设备及自身的状态信息，供CPU处理使用。
- 4) 设备地址识别。

为实现上述功能，设备控制器必须包含以下组成部分：

- ①设备控制器与CPU的接口。该接口有三类信号线：数据线、地址线和控制线。
- ②设备控制器与设备的接口。设备控制器连接设备需要相应数量的接口，一个接口连接一台设备。每个接口中都存在数据、控制和状态三种类型的信号。
- ③I/O控制逻辑。用于实现对设备的控制。



设备控制器的组成

蜂考

类似于文件系统的层次结构，例如，当用户要读取某设备的内容时，通过操作系统提供的*read*命令接口，这就经过了用户层。

操作系统提供给用户使用的接口，一般是统一的通用接口，也就是几乎每个设备都可以响应的统一命令，如*read*命令，用户发出的*read*命令，首先经过设备独立层进行解析，然后再交往下层。

接下来，不同类型的设备对*read*命令的行为会有所不同，如磁盘接收*read*命令后的行为与打印机接收*read*命令后的行为是不同的。因此，需要针对不同的设备，把*read*命令解析成不同的指令，这就经过了设备驱动层。

命令解析完毕后，需要中断正在运行的进程，转而执行*read*命令，这就需要中断处理程序。最后，命令真正抵达硬件设备，硬件设备的控制器按照上层传达的命令操控硬件设备，完成相应的功能。

题 1. 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（ ）。

- A. 命令解释程序
- B. 中断处理程序
- C. 系统调用服务程序
- D. 用户登录程序

答案： B

解析：键盘是典型的通过中断I/O方式工作的外设，当用户输入信息时，计算机响应中断并通过中断处理程序获得输入信息。

题 2. 一个计算机系统配置了2台绘图机和3台打印机，为了正确驱动这些设备，

系统应该提供（ ）个设备驱动程序。

- A. 5
- B. 3
- C. 2
- D. 1

答案： C

解析：因为绘图机和打印机属于两种不同类型的设备，系统只要按设备类型配置设备驱动程序即可，即每类设备只需一个设备驱动程序。

2. I/O 子系统概述

1) I/O 调度概念

I/O 调度就是确定一个好的顺序来执行这些 I/O 请求。磁盘调度算法就是 I/O 调度的一种。

2) 磁盘高速缓存 (*Disk Cache*)

操作系统中使用磁盘高速缓存技术来提高磁盘的 I/O 速度，对高速缓存复制的访问要比原始数据访问更为高效。

不过，磁盘高速缓存技术不同于通常意义上的介于 CPU 与内存之间的小容量高速存储器，而是指利用内存中的存储空间来暂存从磁盘中读出的一系列盘块中的信息。因此，磁盘高速缓存逻辑上属于磁盘，物理上则是驻留在内存中的盘块。高速缓存在内存中分为两种形式：

- ① 在内存中开辟一个单独的存储空间作为磁盘高速缓存，大小固定；
- ② 把未利用的内存空间作为一个缓冲池，供请求分页系统和磁盘 I/O 时共享。

3) 缓冲区 (*Buffer*)

在设备管理子系统中，引入缓冲区的主要目的如下：

- ① 缓和 CPU 与 I/O 设备间速度不匹配的矛盾；
- ② 减少对 CPU 的中断频率，放宽对 CPU 中断响应时间的限制；
- ③ 解决基本数据单元大小（即数据粒度）不匹配的问题；
- ④ 提高 CPU 和 I/O 设备之间的并行性；

其实现方法如下：

- ① 采用硬件缓冲器，但由于成本太高，除一些关键部位外，一般不采用硬件缓冲器；

② 采用缓冲区(位于内存区域)。

缓冲区有一个特点，即当缓冲区的数据非空时，不能往缓冲区冲入数据，只能从缓冲区把数据传出；当缓冲区为空时，可以往缓冲区冲入数据，但必须把缓冲区冲满后，才能从缓冲区把数据传出。

高速缓存和缓冲区的对比：

高速缓存和缓冲区的对比

		高速缓存	缓冲区
相同点		都介于高速设备和低速设备之间	
区别	存放数据	存放的是低速设备上的某些数据的复制数据，即高速缓存上有，低速设备上面必然有	存放的是低速设备传递给高速设备的数据(或相反)，而这些数据在低速设备(或高速设备)上却不一定有备份，这些数据再从缓冲区传送到高速设备(或低速设备)
	目的	高速缓存存放的是高速设备经常要访问的数据，若高速设备要访问的数据不在高速缓存中，则高速设备就需要访问低速设备。	高速设备和低速设备的通信都要经过缓冲区，高速设备永远不会直接去访问低速设备。

课时十九 练习题

1. 计算机系统中，不属于DMA控制器的是()。

- A. 命令/状态寄存器
- B. 内存地址寄存器
- C. 数据寄存器
- D. 堆栈指针寄存器

2. 在下列问题中，()不是设备分配中应考虑的问题。

- A. 及时性
- B. 设备的固有属性
- C. 设备独立性
- D. 安全性

3. 有关设备管理的叙述中，不正确的是()。

- A. 通道是处理输入/输出的软件
- B. 所有设备的启动工作都由系统统一来做
- C. 来自通道的I/O中断事件由设备管理负责处理
- D. 编制好的通道程序是存放在主存中的

4. 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()。

- A. 命令解释程序
- B. 中断处理程序
- C. 系统调用服务程序
- D. 用户登录程序

5. 一个计算机系统配置了2台绘图机和3台打印机，为了正确驱动这些设备，系统应该提供()个设备驱动程序。

- A. 5
- B. 3
- C. 2
- D. 1

6. 用户程序发出磁盘 *I/O* 请求后，系统的正确处理流程是（ ）。
- 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

课时二十 I/O 设备管理（三）

考点	重要程度	占分	题型
1. 设备分配与回收	★	1~2	选择
2. SPOOLing 技术	必考	3~5	选择、应用

1. 设备分配与回收

1) 设备分配概述

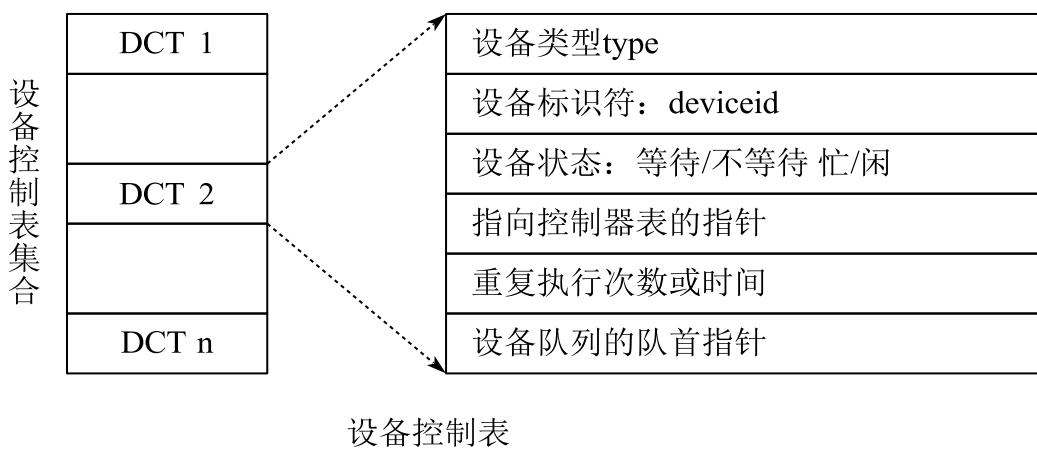
设备分配是指根据用户的 *I/O* 请求分配所需的设备。分配的总原则是充分发挥设备的使用效率，尽可能地让设备忙碌，又要避免由于不合理的分配方法造成进程死锁。从设备的特性看，分为独占设备、共享设备和虚拟设备。

- ① 独占式使用设备。是指在申请设备时，若设备空闲，则将其独占，不再允许其他进程申请使用，一直等到该设备被释放才允许其他进程申请使用。
- ② 分时式共享使用设备。独占设备时通过分时共享使用提高利用率。
- ③ 以 *SPOOLing* 方式使用外部设备。*SPOOLing* 技术（假脱机 *I/O* 技术），是指对 *I/O* 操作进行批处理。*SPOOLing* 技术实质上是一种用空间换时间的技术。

2) 设备分配的数据结构

设备分配依据的主要数据结构有设备控制表 (*DCT*)、控制器控制表 (*COCT*)、通道控制表 (*CHCT*) 和系统设备表 (*SDT*)，各数据结构功能如下。

设备控制表 (*DCT*)：我们可以认为，一个设备控制表就表征一个设备，而这个控制表中的表项就是设备的各个属性，如图所示。



每个设备都分为机械部件和电子部件两部分，其中负责解析上层传达的命令并控制机械部件运作的是电子部件(控制器)，所以每个DCT者需要一个表项来表示控制器，即需要一个指向控制器控制表(COCT)的指针，因此DCT与COCT有一一对应的关系。

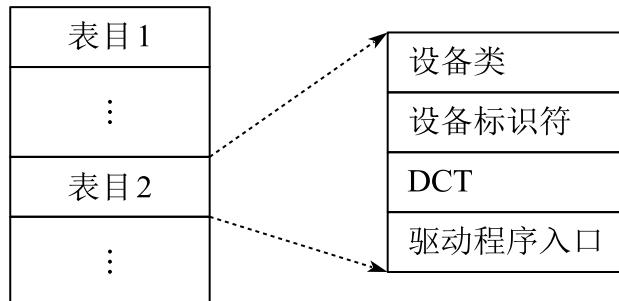
现代操作系统的I/O控制采用的都是通道控制。设备控制器控制设备与内存交换数据，而设备控制器又需要请求通道为它服务，因此每个COCT必定有一个表项存放指向相应通道控制表(CHCT)的指针，而一个通道可为多个设备控制器服务，因此CHCT中必定有一个指针，指向一个表，这个表上的信息表达的是CHCT提供服务的那几个设备控制器。CHCT与COCT的关系是一对多的关系。

控制器标识符: controllerid
控制器状态: 忙/闲
与控制器连接的通道表指针
控制器队列的队首指针
控制器队列的队尾指针

a.COCT

通道标识符: channelid
通道状态: 忙/闲
与通道连接的控制器表首址
通道队列的队首指针
通道队列的队尾指针

b.CHCT



c.SDT

系统设备表(SDT): 整个系统只有一张SDT，如图所示。它记录已连接到系统中的所有物理设备的情况，每个物理设备占一个表目。

3) 设备分配的策略

①设备分配原则。设备分配应根据设备特性、用户要求和系统配置情况。分配的

总原则是：既要充分发挥设备的使用效率，又要避免造成进程死锁，还要将用户程序和具体设备隔离开。

②设备分配方式。设备分配方式有静态分配和动态分配两种。

静态分配主要用于对独占设备的分配，它在用户作业开始执行前，由系统一次性分配该作业所要求的全部设备、控制器（如通道等）。

动态分配在进程执行过程中根据执行需要进行。当进程需要设备时，通过系统调用命令向系统提出设备请求，由系统按照事先规定的策略给进程分配所需要的设备、*I/O*控制器，一旦用完，便立即释放。

③设备分配算法。

常用的动态设备分配算法有先请求先分配、优先级高者优先等。

对于独占设备，既可以采用动态分配方式，又可以采用静态分配方式，但往往采用静态分配方式，即在作业执行前，将作业所要用的这一类设备分配给它。

共享设备可被多个进程所共享，一般采用动态分配方式，但在每个*I/O*传输的单位时间内只被一个进程所占有，通常采用先请求先分配和优先级高者优先的分配算法。

4) 设备分配的安全性

设备分配的安全性是指设备分配中应防止发生进程死锁。

①安全分配方式。

每当进程发出*I/O*请求后便进入阻塞态，直到其*I/O*操作完成时才被唤醒这样，一旦进程已经获得某种设备后便阻塞，不能再请求任何资源，而且在它阻塞时也不保持任何资源。

②不安全分配方式。

进程在发出*I/O*请求后继续运行，需要时又发出第二个、第三个*I/O*请求等。仅当进程所请求的设备已被另一进程占用时，才进入阻塞态。

题 1. 以下()不属于设备管理数据结构。

- A. PCB B. DCT C. COCT D. CHCT

答案: A

解析: *DCT*是设备控制表; *COCT*是控制器控制表; *CHCT*是通道控制表; *PCB*是进程控制块, 不属于设备管理的数据结构。

题 2. 设备的独立性是指()。

- A. 设备独立于计算机系统
B. 系统对设备的管理是独立的
C. 用户编程时使用的设备与实际使用的设备无关
D. 每台设备都有一个唯一的编号

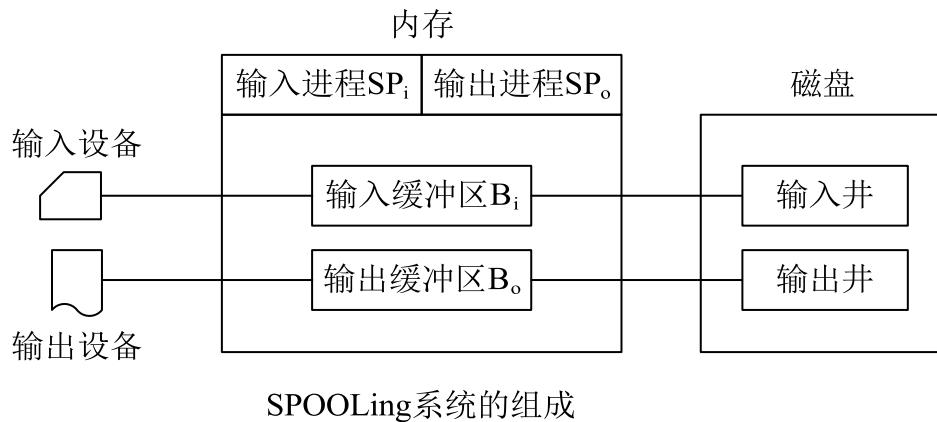
答案: C

设备的独立性主要是指用户使用设备的透明性, 即使用户程序和实际使用的物理设备无关。

2. *SPOOLing* 技术

为了缓和*CPU*的高速性与*I/O*设备低速性之间的矛盾, 引入了脱机输入/输出技术。该技术利用专门的外围控制机, 将低速*I/O*设备上的数据传送到高速磁盘上, 或者相反。*SPOOLing*的意思是外部设备同时联机操作, 又称假脱机输入/输出操作, 是操作系统中采用的一项将独占设备改造成共享设备的技术。

SPOOLing 系统的组成如图所示。



1) 输入井和输出井

输入井和输出井是指在磁盘上开辟出的两个存储区域。

输入井模拟脱机输入时的磁盘，用于收容 I/O 设备输入的数据。

输出井模拟脱机输出时的磁盘，用于收容用户程序的输出数据。

2) 输入缓冲区和输出缓冲区

输入缓冲区和输出缓冲区是在内存中开辟的两个缓冲区。

输入缓冲区用于暂存由输入设备送来的数据，以后再传送到输入井。

输出缓冲区用于暂存从输出井送来的数据，以后再传送到输出设备。

3) 输入进程和输出进程

输入进程模拟脱机输入时的外围控制机，将用户要求的数据从输入机通过输入缓冲区再送到输入井。当 CPU 需要输入数据时，直接将数据从输入井读入内存。

输出进程模拟脱机输出时的外围控制机，把用户要求输出的数据先从内存送到输出井，待输出设备空闲时，再将输出井中的数据经过输出缓冲区送到输出设备。共享打印机是使用 *SPOOLing* 技术的一个实例，当用户进程请求打印输出时，

SPOOLing 系统同意为它打印输出，但并不真正立即把打印机分配给该用户进程，而只为它做两件事：

- ①由输出进程在输出井中为之申请一个空闲磁盘块区，并将要打印的数据送入其中；
- ②输出进程再为用户进程申请一张空白的用户请求打印表，并将用户的打印要求填入其中再将该表挂到请求打印队列上。

SPOOLing 系统的主要特点有：提高了 *I/O* 的速度；将独占设备改造为共享设备；实现了虚拟设备功能。

题 1. 在采用 *SPOOLing* 技术的系统中，用户的打印数据首先被送到（ ）。

- A. 磁盘固定区域
- B. 内存固定区域
- C. 终端
- D. 打印机

答案： A

解析：用户的打印数据首先被送到输出井，输出井在磁盘中。

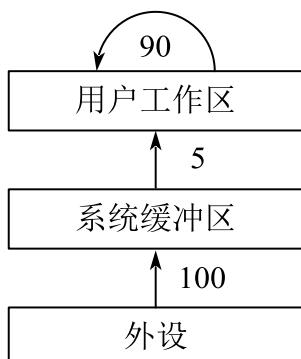
题 2. 下列关于 *SPOOLing* 技术的叙述中，错误的是（ ）。

- A. 需要外存的支持
- B. 需要多道程序设计技术的支持
- C. 可以让多个作业共享一台独占式设备
- D. 由用户作业控制设备与输入/输出井之间的数据传送

答案： D

解析： *SPOOLing* 利用专门的外围控制机，将低速 *I/O* 设备上的数据传送到高速磁盘上，或者相反。*SPOOLing* 的意思是外部设备同时联机操作，又称假脱机输入/输出操作，是操作系统中采用的一项将独占设备改造成共享设备的技术。高速磁盘即外存， A 正确。*SPOOLing* 技术需要进行输入/输出操作，单道批处理系统无法满足， B 正确。*SPOOLing* 技术实现了将独占设备改造成共享设备的技术， C 正确。设备与输入井/输出井之间数据的传送是由系统实现的， D 错误。

课时二十 练习题

1. 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. 逻辑设备名 B. 物理设备名
 C. 主设备号 D. 从设备号
2. 下列选项中，不能改善磁盘设备 *I/O* 性能的是（ ）。
- A. 重排 *I/O* 请求次序 B. 在一个磁盘上设置多个分区
 C. 预读和滞后写 D. 优化文件物理块的分布
3. 为了使并发进程能有效地进入输入和输出，最好采用（ ）结构的缓冲技术。
- A. 缓冲池 B. 循环缓冲
 C. 单缓冲 D. 双缓冲
4. 设系统缓冲区和用户工作区均采用单缓冲，从外设读入一个数据块到系统缓冲区的时间为100，从系统缓冲区读入一个数据块到用户工作区的时间为5，对用户工作区中的一个数据块进行分析的时间为90（如下图所示）。进程从外设读入并分析2个数据块的最短时间是（ ）
- A. 200 B. 295
 C. 300 D. 390
- 
5. 下面关于 *SPOOLing* 系统的说法中，正确的是（ ）。
- A. 构成 *SPOOLing* 系统的基本条件是有外围输入机与外围输出机
 B. 构成 *SPOOLing* 系统的基本条件是要有大容量、高速度的硬盘作为输入井和

输出井

- C. 当输入设备忙时, *SPOOLing* 系统中的用户程序暂停执行, 待 *I/O* 空闲时再被唤醒执行输出操作
- D. *SPOOLing* 系统中的用户程序可以随时将输出数据送到输出井中, 待输出设备空闲时再由 *SPOOLing* 系统完成数据的输出操作。
6. 在系统内存中设置磁盘缓冲区的主要目的是()。
- 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

恭喜你完成本课程学习！

丰富校园资讯

精彩大学生活

更多课程和学习资料

请关注公众号【蜂考】



一起学习，答疑解惑
请加蜂考学习微信群

