

存在的；而程序则是一组代码的集合，是永久存在的，可长期保存。

- 3) 一个进程可以执行一个或几个程序，一个程序也可构成多个进程。进程可创建进程，而程序不可能形成新的程序。
- 4) 进程与程序的组成不同。进程的组成包括程序、数据和 PCB。

## 2. 死锁与饥饿

一组进程处于死锁状态是指组内的每个进程都在等待一个事件，而该事件只可能由组内的另一个进程产生。这里所关心的主要是事件是资源的获取和释放。与死锁相关的另一个问题是无限期阻塞（Indefinite Blocking）或饥饿（Starvation），即进程在信号量内无穷等待的情况。

产生饥饿的主要原因是：在一个动态系统中，对于每类系统资源，操作系统需要确定一个分配策略，当多个进程同时申请某类资源时，由分配策略确定资源分配给进程的次序。有时资源分配策略可能是不公平的，即不能保证等待时间上界的存在。在这种情况下，即使系统没有发生死锁，某些进程也可能会长时间等待。当等待时间给进程推进和响应带来明显影响时，称发生了进程“饥饿”，当“饥饿”到一定程度的进程所赋予的任务即使完成也不再具有实际意义时，称该进程被“饿死”。例如，当有多个进程需要打印文件时，若系统分配打印机的策略是最短文件优先，则长文件的打印任务将由于短文件的源源不断到来而被无限期推迟，导致最终“饥饿”甚至“饿死”。“饥饿”并不表示系统一定会死锁，但至少有一个进程的执行被无限期推迟。

“饥饿”与死锁的主要差别如下：

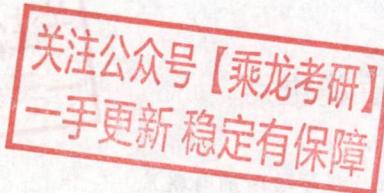
- 1) 进入“饥饿”状态的进程可以只有一个，而因循环等待条件而进入死锁状态的进程却必须大于或等于两个。
- 2) 发生“饥饿”的进程的状态可能是就绪态（长期得不到处理机），也可能是阻塞态（如长期得不到所需的 I/O 设备），而发生死锁的进程的状态则必定是阻塞态。

## 3. 银行家算法的工作原理

银行家算法的主要思想是避免系统进入不安全状态。在每次进行资源分配时，它首先检查系统是否有足够的资源满足要求，若有则先进行试分配，并对分配后的新状态进行安全性检查。若新状态安全，则正式分配上述资源，否则拒绝分配上述资源。这样，它保证系统始终处于安全状态，从而避免了死锁现象的发生。

## 4. 进程同步、互斥的区别和联系

并发进程的执行会产生相互制约的关系：一种是进程之间竞争使用临界资源，只能让它们逐个使用，这种现象称为互斥，是一种竞争关系；另一种是进程之间协同完成任务，在关键点上等待另一个进程发来的消息，以便协同一致，是一种协作关系。



# 第3章 内存管理

关注公众号【乘龙考研】  
一手更新 稳定有保障

## 【考纲内容】

### (一) 内存管理基础

内存管理概念：逻辑地址与物理地址空间，地址变换，内存共享，内存保护，

内存分配与回收

连续分配管理方式；页式管理；段式管理；段页式管理

扫一扫



视频讲解

### (二) 虚拟内存管理

虚拟内存基本概念；请求页式管理；页框分配；页置换算法

内存映射文件（Memory-Mapped Files）；虚拟存储器性能的影响因素及改进方式

## 【复习提示】

内存管理和进程管理是操作系统的核心内容，需要重点复习。本章围绕分页机制展开：通过分页管理方式在物理内存大小的基础上提高内存的利用率，再进一步引入请求分页管理方式，实现虚拟内存，使内存脱离物理大小的限制，从而提高处理器的利用率。

## 3.1 内存管理概念

在学习本节时，请读者思考以下问题：

- 1) 为什么要进行内存管理？
- 2) 页式管理中每个页表项大小的下限如何决定？
- 3) 多级页表解决了什么问题？又会带来什么问题？

在学习经典的管理方法前，同样希望读者先思考，自己给出一些内存管理的想法，并在学习过程中和经典方案进行比较。注意本节给出的内存管理是循序渐进的，后一种方法通常会解决前一种方法的不足。希望读者多多思考，比较每种方法的异同，着重掌握页式管理。

### 3.1.1 内存管理的基本原理和要求

内存管理（Memory Management）是操作系统设计中最重要和最复杂的内容之一。虽然计算机硬件技术一直在飞速发展，内存容量也在不断增大，但仍然不可能将所有用户进程和系统所需要的全部程序与数据放入主存，因此操作系统必须对内存空间进行合理的划分和有效的动态分配。操作系统对内存的划分和动态分配，就是内存管理的概念。

有效的内存管理在多道程序设计中非常重要，它不仅可以方便用户使用存储器、提高内存利用率，还可以通过虚拟技术从逻辑上扩充存储器。

内存管理的主要功能有：

- 内存空间的分配与回收。由操作系统完成主存储器空间的分配和管理，使程序员摆脱存储

分配的麻烦，提高编程效率。

- 地址转换。在多道程序环境下，程序中的逻辑地址与内存中的物理地址不可能一致，因此存储管理必须提供地址变换功能，把逻辑地址转换成相应的物理地址。
- 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。
- 内存共享。允许多个进程访问内存的同一部分。例如，多个合作进程可能需要访问同一块数据，因此必须支持对内存共享区域进行受控访问。
- 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。

在进行具体的内存管理之前，需要了解进程运行的基本原理和要求。

### 1. 程序的链接与装入

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，通常需要以下几个步骤：

- 编译。由编译程序将用户源代码编译成若干目标模块。
- 链接。由链接程序将编译后形成的一组目标模块及它们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。

这三步过程如图 3.1 所示。

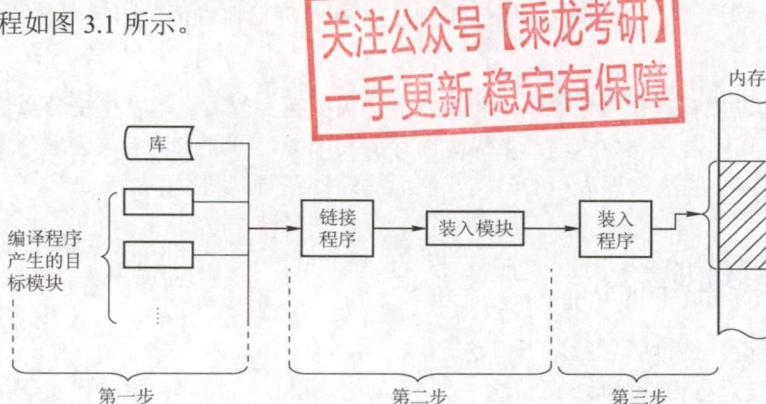


图 3.1 将用户程序变为可在内存中执行的程序的步骤

程序的链接有以下三种方式。

#### (1) 静态链接

在程序运行之前，先将各目标模块及它们所需的库函数链接成一个完整的装配模块，以后不再拆开。将几个目标模块装配成一个装入模块时，需要解决两个问题：①修改相对地址，编译后的所有目标模块都是从 0 开始的相对地址，当链接成一个装入模块时要修改相对地址。②变换外部调用符号，将每个模块中所用的外部调用符号也都变换为相对地址。

#### (2) 装入时动态链接

将用户源程序编译后所得到的一组目标模块，在装入内存时，采用边装入边链接的方式。其优点是便于修改和更新，便于实现对目标模块的共享。

#### (3) 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。凡在执行过程中未被用到的目标模块，都不会被调入内存和被链接到装入模块上。其优点是能加快程序的装入过程，还可节省大量的内存空间。

内存的装入模块在装入内存时，同样有以下三种方式：

### (1) 绝对装入

绝对装入方式只适用于单道程序环境。在编译时，若知道程序将驻留在内存的某个位置，则编译程序将产生绝对地址的目标代码。绝对装入程序按照装入模块中的地址，将程序和数据装入内存。由于程序中的逻辑地址与实际内存地址完全相同，因此不需对程序和数据的地址进行修改。

另外，程序中所用的绝对地址，可在编译或汇编时给出，也可由程序员直接赋予。而通常情况下在程序中采用的是符号地址，编译或汇编时再转换为绝对地址。

### (2) 可重定位装入

在多道程序环境下，多个目标模块的起始地址通常都从 0 开始，程序中的其他地址都是相对于起始地址的，此时应采用可重定位装入方式。根据内存的当前情况，将装入模块装入内存的适当位置。在装入时对目标程序中指令和数据地址的修改过程称为重定位，又因为地址变换通常是在进程装入时一次完成的，故称为静态重定位，如图 3.2(a)所示。

当一个作业装入内存时，必须给它分配要求的全部内存空间，若没有足够的内存，则无法装入。此外，作业一旦进入内存，整个运行期间就不能在内存中移动，也不能再申请内存空间。

### (3) 动态运行时装入

也称动态重定位。程序在内存中若发生移动，则需要采用动态的装入方式。装入程序把装入模块装入内存后，并不立即把装入模块中的相对地址转换为绝对地址，而是把这种地址转换推迟到程序真正要执行时才进行。因此，装入内存后的所有地址均为相对地址。这种方式需要一个重定位寄存器的支持，如图 3.2(b)所示。

动态重定位的优点：可以将程序分配到不连续的存储区；在程序运行之前可以只装入部分代码即可投入运行，然后在程序运行期间，根据需要动态申请分配内存；便于程序段的共享。

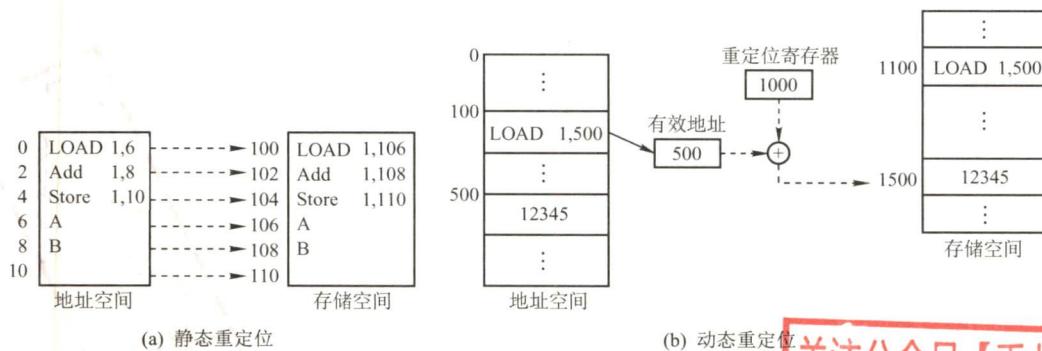


图 3.2 重定位类型

## 2. 逻辑地址与物理地址

编译后，每个目标模块都从 0 号单元开始编址，这称为该目标模块的相对地址（或逻辑地址）。当链接程序将各个模块链接成一个完整的可执行目标程序时，链接程序顺序依次按各个模块的相对地址构成统一的从 0 号单元开始编址的逻辑地址空间（或虚拟地址空间），对于 32 位系统，逻辑地址空间的范围为  $0 \sim 2^{32} - 1$ 。进程在运行时，看到和使用的地址都是逻辑地址。用户程序和程序员只需知道逻辑地址，而内存管理的具体机制则是完全透明的。不同进程可以有相同的逻辑地址，因为这些相同的逻辑地址可以映射到主存的不同位置。

物理地址空间是指内存中物理单元的集合，它是地址转换的最终地址，进程在运行时执行指

令和访问数据，最后都要通过物理地址从主存中存取。当装入程序将可执行代码装入内存时，必须通过地址转换将逻辑地址转换成物理地址，这个过程称为地址重定位。

操作系统通过内存管理部件（MMU）将进程使用的逻辑地址转换为物理地址。进程使用虚拟内存空间中的地址，操作系统在相关硬件的协助下，将它“转换”成真正的物理地址。逻辑地址通过页表映射到物理内存，页表由操作系统维护并被处理器引用。

### 3. 进程的内存映像

不同于存放在硬盘上的可执行程序文件，当一个程序调入内存运行时，就构成了进程的内存映像。一个进程的内存映像一般有几个要素：

- 代码段：即程序的二进制代码，代码段是只读的，可以被多个进程共享。
- 数据段：即程序运行时加工处理的对象，包括全局变量和静态变量。
- 进程控制块（PCB）：存放在系统区。操作系统通过 PCB 来控制和管理进程。
- 堆：用来存放动态分配的变量。通过调用 malloc 函数动态地向高地址分配空间。
- 栈：用来实现函数调用。从用户空间的最大地址往低地址方向增长。

代码段和数据段在程序调入内存时就指定了大小，而堆和栈不一样。当调用像 malloc 和 free 这样的 C 标准库函数时，堆可以在运行时动态地扩展和收缩。用户栈在程序运行期间也可以动态地扩展和收缩，每次调用一个函数，栈就会增长；从一个函数返回时，栈就会收缩。

图 3.3 是一个进程在内存中的映像。其中，共享库用来存放进程用到的共享函数库代码，如 printf() 函数等。在只读代码段中，.init 是程序初始化时调用的\_init 函数；.text 是用户程序的机器代码；.rodata 是只读数据。在读/写数据段中，.data 是已初始化的全局变量和静态变量；.bss 是未初始化及所有初始化为 0 的全局变量和静态变量。

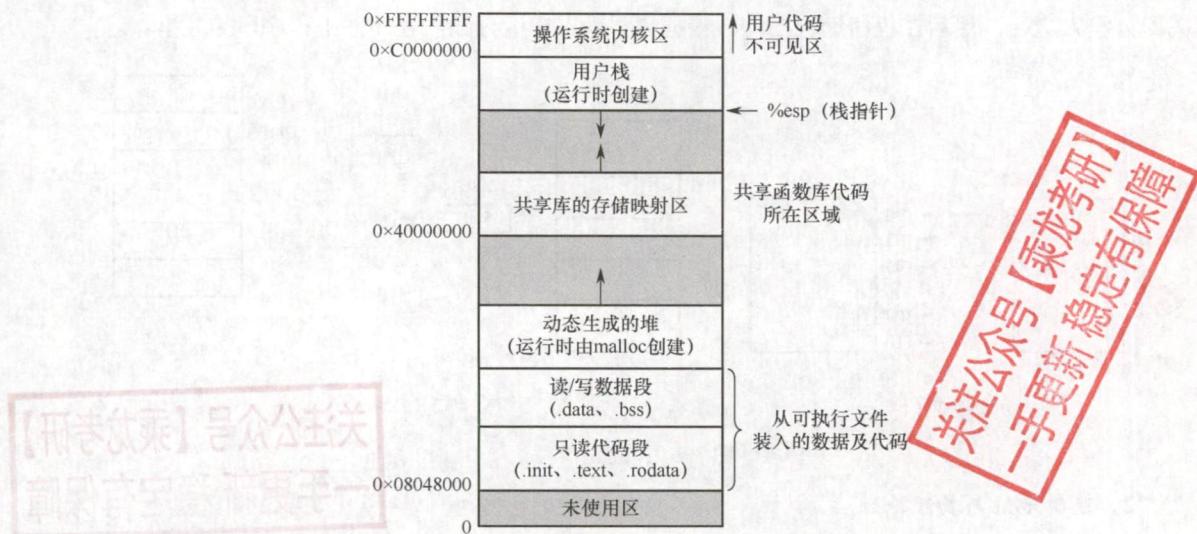


图 3.3 内存中的一个进程

### 4. 内存保护

确保每个进程都有一个单独的内存空间。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。内存保护可采取两种方法：

- 1) 在 CPU 中设置一对上、下限寄存器，存放用户作业在主存中的下限和上限地址，每当 CPU 要访问一个地址时，分别和两个寄存器的值相比，判断有无越界。

- 2) 采用重定位寄存器(又称基地址寄存器)和界地址寄存器(又称限长寄存器)来实现这种保护。重定位寄存器含最小的物理地址值, 界地址寄存器含逻辑地址的最大值。内存管理机构动态地将逻辑地址与界地址寄存器进行比较, 若未发生地址越界, 则加上重定位寄存器的值后映射成物理地址, 再送交内存单元, 如图 3.4 所示。

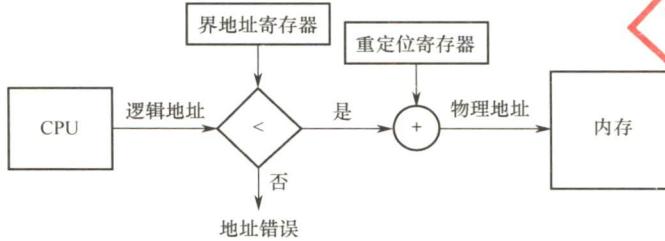


图 3.4 重定位寄存器和界地址寄存器的硬件支持

实现内存保护需要重定位寄存器和界地址寄存器, 因此要注意两者的区别。重定位寄存器是用来“加”的, 逻辑地址加上重定位寄存器中的值就能得到物理地址; 界地址寄存器是用来“比”的, 通过比较界地址寄存器中的值与逻辑地址的值来判断是否越界。

加载重定位寄存器和界地址寄存器时必须使用特权指令, 只有操作系统内核才可以加载这两个存储器。这种方案允许操作系统内核修改这两个寄存器的值, 而不允许用户程序修改。

## 5. 内存共享

并不是所有的进程内存空间都适合共享, 只有那些只读的区域才可以共享。可重入代码又称纯代码, 是一种允许多个进程同时访问但不允许被任何进程修改的代码。但在实际执行时, 也可以为每个进程配以局部数据区, 把在执行中可能改变的部分复制到该数据区, 这样, 程序在执行时只需对该私有数据区中的内存进行修改, 并不去改变共享的代码。

下面通过一个例子来说明内存共享的实现方式。考虑一个可以同时容纳 40 个用户的多用户系统, 他们同时执行一个文本编辑程序, 若该程序有 160KB 代码区和 40KB 数据区, 则共需 8000KB 的内存空间来支持 40 个用户。如果 160KB 代码是可分享的纯代码, 则不论是在分页系统中还是在分段系统中, 整个系统只需保留一份副本即可, 此时所需的内存空间仅为  $40KB \times 40 + 160KB = 1760KB$ 。对于分页系统, 假设页面大小为 4KB, 则代码区占用 40 个页面、数据区占用 10 个页面。为实现代码共享, 应在每个进程的页表中都建立 40 个页表项, 它们都指向共享代码区的物理页号。此外, 每个进程还要为自己的数据区建立 10 个页表项, 指向私有数据区的物理页号。对于分段系统, 由于是以段为分配单位的, 不管该段有多大, 都只需为该段设置一个段表项(指向共享代码段始址, 以及段长 160KB)。由此可见, 段的共享非常简单易行。

此外, 在第 2 章中我们介绍过基于共享内存的进程通信, 由操作系统提供同步互斥工具。在本章的后面, 还将介绍一种内存共享的实现——内存映射文件。

## 6. 内存分配与回收

存储管理方式随着操作系统的发展而发展。在操作系统由单道向多道发展时, 存储管理方式便由单一连续分配发展为固定分区分配。为了能更好地适应不同大小的程序要求, 又从固定分区分配发展到动态分区分配。为了更好地提高内存的利用率, 进而从连续分配方式发展到离散分配方式——页式存储管理。引入分段存储管理的目的, 主要是为了满足用户在编程和使用方面的要求, 其中某些要求是其他几种存储管理方式难以满足的。

### \*3.1.2 覆盖与交换<sup>①</sup>

覆盖与交换技术是在多道程序环境下用来扩充内存的两种方法。

#### (1) 覆盖

早期的计算机系统中，主存容量很小，虽然主存中仅存放一道用户程序，但存储空间放不下用户进程的现象也经常发生，这一矛盾可以用覆盖技术来解决。

覆盖的基本思想如下：由于程序运行时并非任何时候都要访问程序及数据的各个部分（尤其是大程序），因此可把用户空间分成一个固定区和若干覆盖区。将经常活跃的部分放在固定区，其余部分按调用关系分段。首先将那些即将要访问的段放入覆盖区，其他段放在外存中，在需要调用前，系统再将其调入覆盖区，替换覆盖区中原有的段。

覆盖技术的特点是，打破了必须将一个进程的全部信息装入主存后才能运行的限制，但当同时运行程序的代码量大于主存时仍不能运行，此外，内存中能够更新的地方只有覆盖区的段，不在覆盖区中的段会常驻内存。覆盖技术对用户和程序员不透明。

#### (2) 交换

交换（对换）的基本思想是，把处于等待状态（或在 CPU 调度原则下被剥夺运行权利）的程序从内存移到辅存，把内存空间腾出来，这一过程又称换出；把准备好竞争 CPU 运行的程序从辅存移到内存，这一过程又称换入。第 2 章介绍的中级调度采用的就是交换技术。

例如，有一个 CPU 采用时间片轮转调度算法的多道程序环境。时间片到，内存管理器将刚刚执行过的进程换出，将另一进程换入刚刚释放的内存空间。同时，CPU 调度器可以将时间片分配给其他已在内存中的进程。每个进程用完时间片都与另一进程交换。在理想情况下，内存管理器的交换过程速度足够快，总有进程在内存中可以执行。

有关交换，需要注意以下几个问题：

- 交换需要备份存储，通常是磁盘。它必须足够大，并提供对这些内存映像的直接访问。
- 为了有效使用 CPU，需要使每个进程的执行时间比交换时间长。
- 若换出进程，则必须确保该进程完全处于空闲状态。
- 交换空间通常作为磁盘的一整块，且独立于文件系统，因此使用起来可能很快。
- 交换通常在有许多进程运行且内存空间吃紧时开始启动，而在系统负荷降低时就暂停。
- 普通的交换使用不多，但交换策略的某些变体在许多系统（如 UNIX）中仍发挥作用。

交换技术主要在不同进程（或作业）之间进行，而覆盖则用于同一个程序或进程中。对于主存无法存放用户程序的矛盾，现代操作系统是通过虚拟内存技术来解决的，覆盖技术则已成为历史；而交换技术在现代操作系统中仍具有较强的生命力。

### 3.1.3 连续分配管理方式

连续分配方式是指为一个用户程序分配一个连续的内存空间，譬如某用户需要 100MB 的内存空间，连续分配方式就在内存空间中为用户分配一块连续的 100MB 空间。连续分配方式主要包括单一连续分配、固定分区分配和动态分区分配。

<sup>①</sup> 加 “\*” 号表示新大纲中已删除，仅供学习参考。

### 1. 单一连续分配

内存此方式下分为系统区和用户区，系统区仅供操作系统使用，通常在低地址部分；在用户区内存中，仅有一道用户程序，即整个内存的用户空间由该程序独占。

这种方式的优点是简单、无外部碎片，无须进行内存保护，因为内存中永远只有一道程序。缺点是只能用于单用户、单任务的操作系统中，有内部碎片，存储器的利用率极低。

### 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固定大小的区域，每个分区只装入一道作业。当有空闲分区时，便可再从外存的后备作业队列中选择适当大小的作业装入该分区，如此循环。在划分分区时有两种不同的方法。

- 分区大小相等。程序太小会造成浪费，程序太大又无法装入，缺乏灵活性。
- 分区大小不等。划分为多个较小的分区、适量的中等分区和少量大分区。

为了便于分配，建立一张分区使用表，通常按分区大小排队，各表项包括每个分区的起始地址、大小及状态（是否已分配），如图 3.5 所示。分配内存时，便检索该表，以找到一个能满足要求且尚未分配的分区分配给装入程序，并将对应表项的状态置为“已分配”；若找不到这样的分区，则拒绝分配。回收内存时，只需将对应表项的状态置为“未分配”即可。

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	未分配

(a) 分区使用表

20KB	操作系统
32KB	作业A
64KB	作业B
128KB	作业C
256KB	

(b) 存储空间分配情况

图 3.5 固定分区说明表和内存分配情况

这种方式存在两个问题：一是程序可能太大而放不进任何一个分区，这时就需要采用覆盖技术来使用内存空间；二是当程序小于固定分区大小时，也要占用一个完整的内存分区，这样分区内部就存在空间浪费，这种现象称为内部碎片。固定分区是可用于多道程序设计的最简单的存储分配，无外部碎片，但不能实现多进程共享一个主存区，所以存储空间利用率低。

### 3. 动态分区分配

又称可变分区分配，它是在进程装入内存时，根据进程的实际需要，动态地为之分配内存，并使分区的大小正好适合进程的需要。因此，系统中分区的大小和数目是可变的。

如图 3.6 所示，系统有 64MB 内存空间，其中低 8MB 固定分配给操作系统，其余为用户可用内存。开始时装入前三个进程，它们分别分配到所需的空间后，内存仅剩 4MB，进程 4 无法装入。在某个时刻，内存中没有一个就绪进程，CPU 出现空闲，操作系统就换出进程 2，换入进程 4。由于进程 4 比进程 2 小，这样在主存中就产生了一个 6MB 的内存块。之后 CPU 又出现空闲，需要换入进程 2，而主存无法容纳进程 2，操作系统就换出进程 1，换入进程 2。

动态分区在开始时是很好的，但随着时间的推移，内存中会产生越来越多小的内存块，内存的利用率也随之下降。这些小的内存块称为外部碎片，它存在于所有分区的外部，这与固定分区中的内部碎片正好相对。克服外部碎片可以通过紧凑技术来解决，即操作系统不时地对进程进行

移动和整理。但这需要动态重定位寄存器的支持，且相对费时。紧凑的过程实际上类似于 Windows 系统中的磁盘碎片整理程序，只不过后者是对外存空间的紧凑。



图 3.6 动态分区分配

在进程装入或换入主存时，若内存中有多个足够大的空闲块，则操作系统必须确定分配哪个内存块给进程使用，这就是动态分区的分配策略。考虑以下几种算法：

- 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，从链首开始顺序查找，找到大小能满足要求的第一个空闲分区分配给作业。
- 2) 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。
- 3) 最佳适应 (Best Fit) 算法。空闲分区按容量递增的次序形成空闲分区链，找到第一个能满足要求且最小的空闲分区分配给作业，避免“大材小用”。
- 4) 最坏适应 (Worst 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，即最大的分区，从中分割一部分存储空间给作业。

首次适应算法最简单，通常也是最好和最快的。不过，首次适应算法会使得内存的低地址部分出现很多小的空闲分区，而每次分分配查找时都要经过这些分区，因此增加了开销。

邻近适应算法试图解决这个问题。但它常常导致在内存空间的尾部（因为在一遍扫描中，内存前面部分使用后再释放时，不会参与分配）分裂成小碎片。通常比首次适应算法要差。

最佳适应算法虽然称为“最佳”，但是性能通常很差，因为每次最佳的分配会留下很小的难以利用的内存块，会产生最多的外部碎片。

最坏适应算法与最佳适应算法相反，它选择最大的可用块，这看起来最不容易产生碎片，但是却把最大的连续内存划分开，会很快导致没有可用的大内存块，因此性能也非常差。

在动态分区分配中，与固定分区分配类似，设置一张空闲分区链（表），并按始址排序。分配内存时，检索空闲分区链，找到所需的分区，若其大小大于请求大小，便从该分区中按请求大小分割一块空间分配给装入进程（若剩余部分小到不足以划分，则无须分割），余下部分仍留在空闲分区链中。回收内存时，系统根据回收分区的始址，从空闲分区链中找到相应的插入点，此时可能出现四种情况：①回收区与插入点的前一空闲分区相邻，将这两个分区合并，并修改前一分区表项的大小为两者之和；②回收区与插入点的后一空闲分区相邻，将这两个分区合并，并修改后一分区表项的始址和大小；③回收区同时与插入点的前、后两个分区相邻，此时将这三个分区合并，修改前一分区表项的大小为三者之和，取消后一分区表项；④回收区没有相邻的空闲分区，此时应为回收区新建一个表项，填写始址和大小，并插入空闲分区链。

以上三种内存分区管理方法有一个共同特点，即用户程序在主存中都是连续存放的。

在连续分配方式中，我们发现，即使内存有超过 1GB 的空闲空间，但若没有连续的 1GB 空

间，则需要 1GB 空间的作业仍然是无法运行的；但若采用非连续分配方式，则作业所要求的 1GB 内存空间可以分散地分配在内存的各个区域，当然，这也需要额外的空间去存储它们（分散区域）的索引，使得非连续分配方式的存储密度低于连续分配方式。非连续分配方式根据分区的大小是否固定，分为分页存储管理和分段存储管理。在分页存储管理中，又根据运行作业时是否要把作业的所有页面都装入内存才能运行，分为基本分页存储管理和请求分页存储管理。

### 3.1.4 基本分页存储管理

固定分区会产生内部碎片，动态分区会产生外部碎片，这两种技术对内存的利用率都比较低。我们希望内存的使用能尽量避免碎片的产生，这就引入了分页的思想：把主存空间划分为大小相等且固定的块，块相对较小，作为主存的基本单位。每个进程也以块为单位进行划分，进程在执行时，以块为单位逐个申请主存中的块空间。

分页的方法从形式上看，像分区相等的固定分区技术，分页管理不会产生外部碎片。但它又有本质的不同点：块的大小相对分区要小很多，而且进程也按照块进行划分，进程运行时按块申请主存可用空间并执行。这样，进程只会在为最后一个不完整的块申请一个主存块空间时，才产生主存碎片，所以尽管会产生内部碎片，但这种碎片相对于进程来说也是很小的，每个进程平均只产生半个块大小的内部碎片（也称页内碎片）。

#### 1. 分页存储的几个基本概念

##### (1) 页面和页面大小

进程中的块称为页或页面（Page），内存中的块称为页框或页帧（Page Frame）。外存也以同样的单位进行划分，直接称为块或盘块（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。

为方便地址转换，页面大小应是 2 的整数幂。同时页面大小应该适中，页面太小会使进程的页面数过多，这样页表就会过长，占用大量内存，而且也会增加硬件地址转换的开销，降低页面换入/换出的效率；页面过大又会使页内碎片增多，降低内存的利用率。

##### (2) 地址结构

分页存储管理的逻辑地址结构如图 3.7 所示。

31	...	12	11	...	0
页号 $P$			页内偏移量 $W$		

图 3.7 分页存储管理的逻辑地址结构

地址结构包含两部分：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，即最多允许  $2^{20}$  页。

注意，地址结构决定了虚拟内存的寻址空间有多大。在实际问题中，页号、页内偏移、逻辑地址可能是用十进制数给出的，若题目用二进制地址的形式给出时，读者要学会转换。

##### (3) 页表

为了便于在内存中找到进程的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。

在配置页表后，进程执行时，通过查找该表，即可找到每页在内存中的物理块号。可见，页表的作用是实现从页号到物理块号的地址映射，如图 3.8 所示。

页表是由页表项组成的，初学者容易混淆页表项与地址结构，页表项与地址都由两部分构成，

而且第一部分都是页号，但页表项的第二部分是物理内存中的块号，而地址的第二部分是页内偏移；页表项的第二部分与地址的第二部分共同组成物理地址。

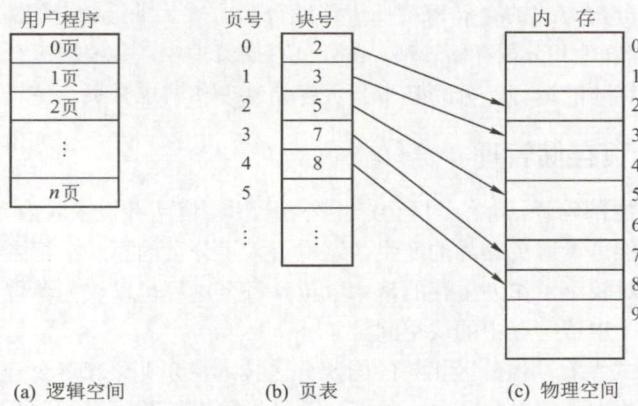


图 3.8 页表的作用

## 2. 基本地址变换机构

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。图 3.9 给出了分页存储管理系统中的地址变换机构。

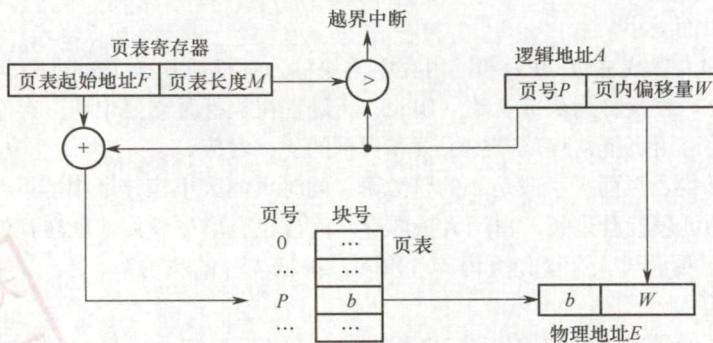


图 3.9 分页存储管理系统中的地址变换机构

在系统中通常设置一个页表寄存器 (PTR)，存放页表在内存的起始地址  $F$  和页表长度  $M$ 。平时，进程未执行时，页表的始址和页表长度存放在本进程的 PCB 中，当进程被调度执行时，才将页表始址和页表长度装入页表寄存器中。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（假设逻辑地址、页号、每页的长度都是十进制数）：

- ① 计算页号  $P$  ( $P = A/L$ ) 和页内偏移量  $W$  ( $W = A \% L$ )。
- ② 比较页号  $P$  和页表长度  $M$ ，若  $P \geq M$ ，则产生越界中断，否则继续执行。
- ③ 页表中页号  $P$  对应的页表项地址 = 页表始址  $F +$  页号  $P \times$  页表项长度，取出该页表项内容  $b$ ，即为物理块号。注意区分页表长度和页表项长度。页表长度是指一共有多少页，页表项长度是指页地址占多大的存储空间。
- ④ 计算  $E = b \times L + W$ ，用得到的物理地址  $E$  去访问内存。

以上整个地址变换过程均是由硬件自动完成的。例如，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  $E$  的过程如下： $P = 2500/1K = 2$ ， $W = 2500 \% 1K = 452$ ，查找得到页号 2 对应的物理块的块号为 8， $E = 8 \times 1024 + 452 = 8644$ 。

计算条件用十进制数和用二进制数给出，过程会稍有不同。页式管理只需给出一个整数就能确定对应的物理地址，因为页面大小  $L$  是固定的。因此，页式管理中地址空间是一维的。

页表项的大小不是随意规定的，而是有所约束的。如何确定页表项的大小？

页表项的作用是找到该页在内存中的位置。以 32 位逻辑地址空间、字节编址单位、一页 4KB 为例，地址空间内一共有  $2^{32}B/4KB = 1M$  页，因此需要  $\log_2 1M = 20$  位才能保证表示范围能容纳所有页面，又因为以字节作为编址单位，即页表项的大小  $\geq \lceil 20/8 \rceil = 3B$ 。所以在这个条件下，为了保证页表项能够指向所有页面，页表项的大小应该大于或等于 3B，当然，也可选择更大的页表项让一个页面能够正好容下整数个页表项，进而方便存储（如取成 4B，这样一页正好可以装下 1K 个页表项），或增加一些其他信息。

下面讨论分页管理方式存在的两个主要问题：①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；②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。

### 3. 具有快表的地址变换机构

由上面介绍的地址变换过程可知，若页表全部放在内存中，则存取一个数据或一条指令至少要访问两次内存：第一次是访问页表，确定所存取的数据或指令的物理地址；第二次是根据该地址存取数据或指令。显然，这种方法比通常执行指令的速度慢了一半。

为此，在地址变换机构中增设一个具有并行查找能力的高速缓冲存储器——快表，又称相联存储器（TLB），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，主存中的页表常称为慢表。具有快表的地址变换机构如图 3.10 所示。

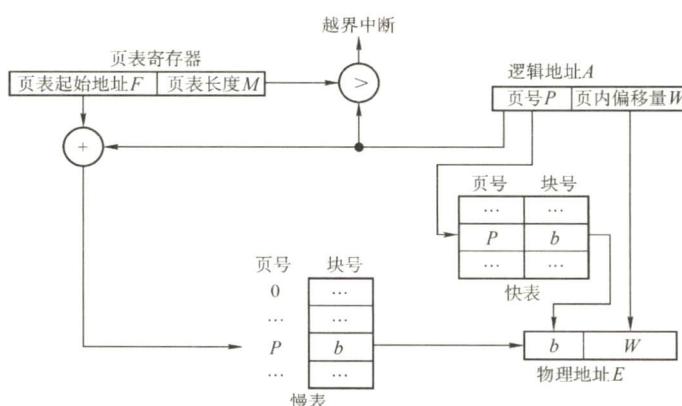


图 3.10 具有快表的地址变换机构

在具有快表的分页机制中，地址的变换过程如下：

- ① CPU 给出逻辑地址后，由硬件进行地址转换，将页号送入高速缓存寄存器，并将此页号与快表中的所有页号进行比较。
- ② 若找到匹配的页号，说明所要访问的页表项在快表中，则直接从中取出该页对应的页框号，与页内偏移量拼接形成物理地址。这样，存取数据仅一次访存便可实现。
- ③ 若未找到匹配的页号，则需要访问主存中的页表，读出页表项后，应同时将其存入快表，以便后面可能的再次访问。若快表已满，则须按特定的算法淘汰一个旧页表项。

**注意：**有些处理机设计为快表和慢表同时查找，若在快表中查找成功则终止慢表的查找。

关注公众号【乘龙考研】  
一手更新 稳定有保障

一般快表的命中率可达 90%以上，这样分页带来的速度损失就可降低至 10%以下。快表的有效性基于著名的局部性原理，后面讲解虚拟内存时将会具体讨论它。

#### 4. 两级页表

由于引入了分页管理，进程在执行时不需要将所有页调入内存页框，而只需将保存有映射关系的页表调入内存。但是，我们仍然需要考虑页表的大小。以 32 位逻辑地址空间、页面大小 4KB、页表项大小 4B 为例，若要实现进程对全部逻辑地址空间的映射，则每个进程需要  $2^{20}$  即约 100 万个页表项。也就是说，每个进程仅页表这一项就需要 4MB 主存空间，而且还要求是连续的，显然这是不切实际的。即便不考虑对全部逻辑地址空间进行映射的情况，一个逻辑地址空间稍大的进程，其页表大小也可能是过大的。以一个 40MB 的进程为例，页表项共 40KB ( $40\text{MB}/4\text{KB} \times 4\text{B}$ )，若将所有页表项内容保存在内存中，则需要 10 个内存页框来保存整个页表。整个进程大小约为 1 万个页面，而实际执行时只需要几十个页面进入内存页框就可运行，但若要求 10 个页面大小的页表必须全部进入内存，则相对实际执行时的几十个进程页面的大小来说，肯定降低了内存利用率；从另一方面来说，这 10 页的页表项也并不需要同时保存在内存中，因为在大多数情况下，映射所需要的页表项都在页表的同一个页面中。

为了压缩页表，我们进一步延伸页表映射的思想，就可得到二级分页，即使用层次结构的页表：将页表的 10 页空间也进行地址映射，建立上一级页表，用于存储页表的映射关系。这里对页表的 10 个页面进行映射只需要 10 个页表项，所以上一级页表只需要 1 页就已足够（可以存储  $2^{10} = 1024$  个页表项）。在进程执行时，只需要将这一页的上一级页表调入内存即可，进程的页表和进程本身的页面可在后面的执行中再调入内存。根据上面提到的条件（32 位逻辑地址空间、页面大小 4KB、页表项大小 4B，以字节为编址单位），我们来构造一个适合的页表结构。页面大小为 4KB，页内偏移地址为  $\log_2 4K = 12$  位，页号部分为 20 位，若不采用分级页表，则仅页表就要占用  $2^{20} \times 4B / 4KB = 1024$  页，这大大超过了许多进程自身需要的页面，对于内存来说是非常浪费资源的，而且查询页表工作也会变得十分不便、试想若把这些页表放在连续的空间内，查询对应页的物理页号时可以通过页表首地址 + 页号  $\times 4B$  的形式得到，而这种方法查询起来虽然相对方便，但连续的 1024 页对于内存的要求实在太高，并且上面也说到了其中大多数页面都是不会用到的，所以这种方法并不具有可行性。若不把这些页表放在连续的空间里，则需要一张索引表来告诉我们第几张页表该上哪里去找，这能解决页表的查询问题，且不用把所有的页表都调入内存，只在需要它时才调入（下节介绍的虚拟存储器思想），因此能解决占用内存空间过大的问题。读者也许发现这个方案就和当初引进页表机制的方式一模一样，实际上就是构造一个页表的页表，也就是二级页表。为查询方便，顶级页表最多只能有 1 个页面（一定要记住这个规定），因此顶级页表总共可以容纳  $4KB / 4B = 1K$  个页表项，它占用的地址位数为  $\log_2 1K = 10$  位，而之前已经计算出页内偏移地址占用了 12 位，因此一个 32 位的逻辑地址空间就剩下了 10 位，正好使得二级页表的大小在一页之内，这样就得到了逻辑地址空间的格式，如图 3.11 所示。

一级页号	二级页号	页内偏移
------	------	------

图 3.11 逻辑地址空间的格式

二级页表实际上是在原有页表结构上再加上一层页表，示意结构如图 3.12 所示。

建立多级页表的目的在于建立索引，以便不用浪费主存空间去存储无用的页表项，也不用盲目地顺序式查找页表项。

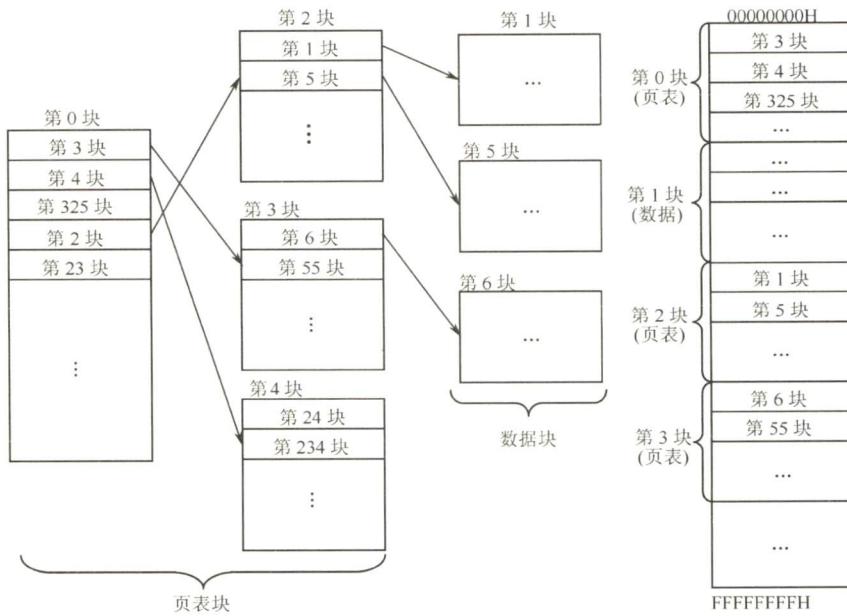


图 3.12 二级页表结构示意图

关注公众号【乘龙考研】  
一手更新 稳定有保障

### 3.1.5 基本分段存储管理

分页管理方式是从计算机的角度考虑设计的，目的是提高内存的利用率，提升计算机的性能。分页通过硬件机制实现，对用户完全透明。分段管理方式的提出则考虑了用户和程序员，以满足方便编程、信息保护和共享、动态增长及动态链接等多方面的需要。

#### 1. 分段

段式管理方式按照用户进程中的自然段划分逻辑空间。例如，用户进程由主程序段、两个子程序段、栈段和数据段组成，于是可以把这个用户进程划分为 5 段，每段从 0 开始编址，并分配一段连续的地址空间（段内要求连续，段间不要求连续，因此整个作业的地址空间是二维的），其逻辑地址由段号  $S$  与段内偏移量  $W$  两部分组成。

在图 3.13 中，段号为 16 位，段内偏移量为 16 位，因此一个作业最多有  $2^{16} = 65536$  段，最大段长为 64KB。

31	...	16	15	...	0
段号 $S$			段内偏移量 $W$		

图 3.13 分段系统中的逻辑地址结构

在页式系统中，逻辑地址的页号和页内偏移量对用户是透明的，但在段式系统中，段号和段内偏移量必须由用户显式提供，在高级程序设计语言中，这个工作由编译程序完成。

#### 2. 段表

每个进程都有一张逻辑空间与内存空间映射的段表，其中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。段表的内容如图 3.14 所示。

段号	段长	本段在主存的始址
----	----	----------

图 3.14 段表的内容

配置段表后，执行中的进程可通过查找段表，找到每段所对应的内存区。可见，段表用于实现从逻辑段到物理内存区的映射，如图 3.15 所示。

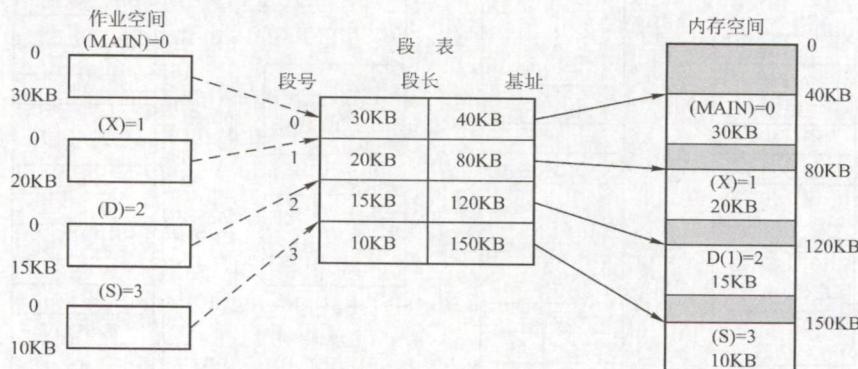


图 3.15 利用段表实现物理内存区映射

### 3. 地址变换机构

分段系统的地址变换过程如图 3.16 所示。为了实现进程从逻辑地址到物理地址的变换功能，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：

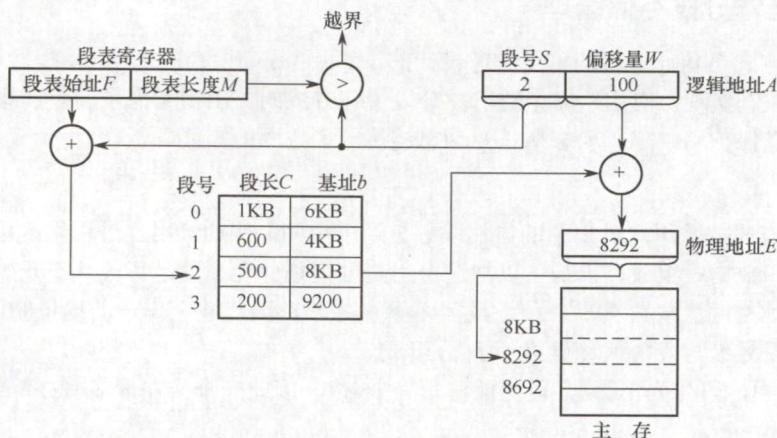


图 3.16 分段系统的地址变换过程

- ① 从逻辑地址  $A$  中取出前几位为段号  $S$ ，后几位为段内偏移量  $W$ 。注意，在地址变换的题目中，要注意逻辑地址是用二进制数还是用十进制数给出的。
- ② 比较段号  $S$  和段表长度  $M$ ，若  $S \geq M$ ，则产生越界中断，否则继续执行。
- ③ 段表中段号  $S$  对应的段表项地址 = 段表始址  $F$  + 段号  $S \times$  段表项长度，取出该段表项的前几位得到段长  $C$ 。若段内偏移量  $\geq C$ ，则产生越界中断，否则继续执行。从这句话我们可以看出，段表项实际上只有两部分，前几位是段长，后几位是始址。
- ④ 取出段表项中该段的始址  $b$ ，计算  $E = b + W$ ，用得到的物理地址  $E$  去访问内存。

### 4. 段的共享与保护

在分段系统中，段的共享是通过两个作业的段表中相应表项指向被共享的段的同一个物理副本实现的。当一个作业正从共享段中读取数据时，必须防止另一个作业修改此共享段中的

数据。不能修改的代码称为纯代码或可重入代码（它不属于临界资源），这样的代码和不能修改的数据可以共享，而可修改的代码和数据不能共享。

与分页管理类似，分段管理的保护方法主要有两种：一种是存取控制保护，另一种是地址越界保护。地址越界保护将段表寄存器中的段表长度与逻辑地址中的段号比较，若段号大于段表长度，则产生越界中断；再将段表项中的段长和逻辑地址中的段内偏移进行比较，若段内偏移大于段长，也会产生越界中断。分页管理只需要判断页号是否越界，页内偏移是不可能越界的。

与页式管理不同，段式管理不能通过给出一个整数便确定对应的物理地址，因为每段的长度是不固定的，无法通过整数除法得出段号，无法通过求余得出段内偏移，所以段号和段内偏移一定要显式给出（段号，段内偏移），因此分段管理的地址空间是二维的。

### 3.1.6 段页式管理

分页存储管理能有效地提高内存利用率，而分段存储管理能反映程序的逻辑结构并有利于段的共享和保护。将这两种存储管理方法结合起来，便形成了段页式存储管理方式。

在段页式系统中，作业的地址空间首先被分成若干逻辑段，每段都有自己的段号，然后将每段分成若干大小固定的页。对内存空间的管理仍然和分页存储管理一样，将其分成若干和页面大小相同的存储块，对内存的分配以存储块为单位，如图 3.17 所示。

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 3.18 所示。

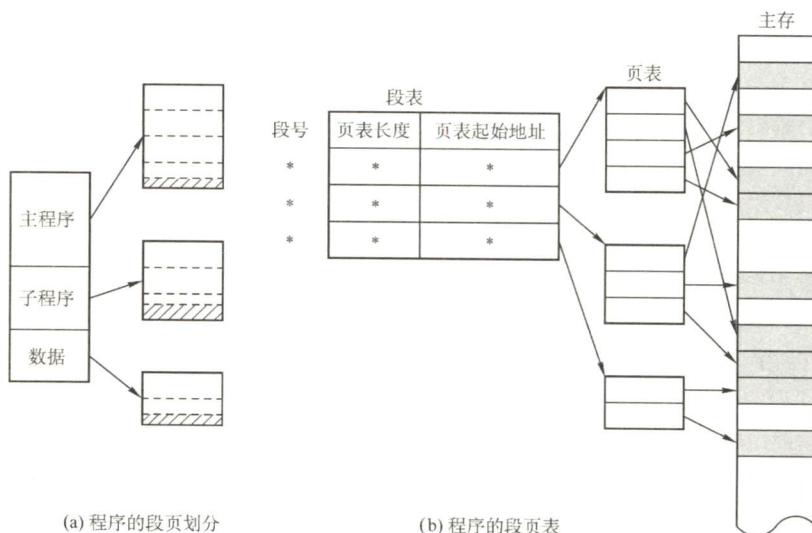


图 3.17 段页式管理方式



图 3.18 段页式系统的逻辑地址结构

为了实现地址变换，系统为每个进程建立一张段表，每个分段有一张页表。段表表项中至少包括段号、页表长度和页表始址，页表表项中至少包括页号和块号。此外，系统中还应有一个段表寄存器，指出作业的段表始址和段表长度（段表寄存器和页表寄存器的作用都有两个，一是在段表或页表中寻址，二是判断是否越界）。

**注意：**在一个进程中，段表只有一个，而页表可能有多个。

在进行地址变换时，首先通过段表查到页表始址，然后通过页表找到页帧号，最后形成物理地址。如图 3.19 所示，进行一次访问实际需要三次访问主存，这里同样可以使用快表来加快查找速度，其关键字由段号、页号组成，值是对应的页帧号和保护码。

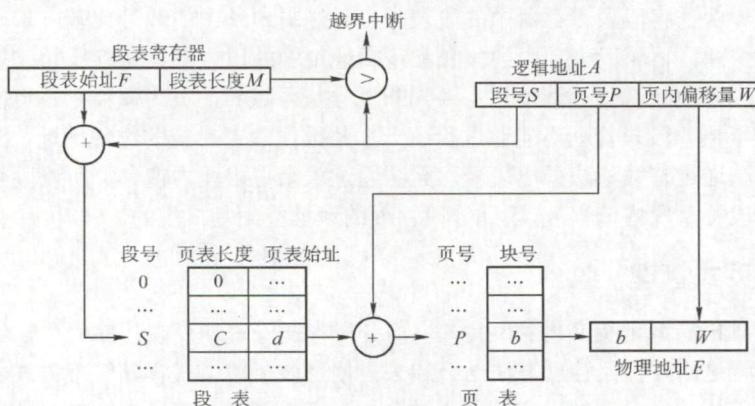


图 3.19 段页式系统的地址变换机构

结合上面对段式和页式管理地址空间的分析，得出结论：段页式管理的地址空间是二维的。

### 3.1.7 本节小结

本节开头提出的问题的参考答案如下。

#### 1) 为什么要进行内存管理？

在单道系统阶段，一个系统在一个时间段内只执行一个程序，内存的分配极其简单，即仅分配给当前运行的进程。引入多道程序后，进程之间共享的不仅仅是处理机，还有主存储器。然而，共享主存会形成一些特殊的挑战。若不对内存进行管理，则容易导致内存数据的混乱，以至于影响进程的并发执行。因此，为了更好地支持多道程序并发执行，必须进行内存管理。

#### 2) 页式管理中每个页表项大小的下限如何决定？

页表项的作用是找到该页在内存中的位置。以 32 位逻辑地址空间、字节编址单位、一页 4KB 为例，地址空间内共含有  $2^{32}B / 4KB = 1M$  页，需要  $\log_2 1M = 20$  位才能保证表示范围能容纳所有页面，又因为以字节作为编址单位，即页表项的大小  $\geq \lceil 20/8 \rceil = 3B$ 。当然，也可选择更大的页表项大小，让一个页面能够正好容下整数个页表项，以方便存储（例如取成 4B，一页正好可以装下 1K 个页表项），或增加一些其他信息。

#### 3) 多级页表解决了什么问题？又会带来什么问题？

多级页表解决了当逻辑地址空间过大时，页表的长度会大大增加的问题。而采用多级页表时，一次访存需要多次访问内存甚至磁盘，会大大增加一次访存的时间。

无论是段式管理、页式管理还是段页式管理，读者都只需要掌握下面三个关键问题：①逻辑地址结构，②表项结构，③寻址过程。搞清楚这三个问题，就相当于搞清楚了上面几种存储管理方式。再次提醒读者区分逻辑地址结构和表项结构。

### 3.1.8 本节习题精选

#### 一、单项选择题

01. 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

02. 段页式存储管理中，地址映射表是（）。  
 A. 每个进程一张段表，两张页表  
 B. 每个进程的每个段一张段表，一张页表  
 C. 每个进程一张段表，每个段一张页表  
 D. 每个进程一张页表，每个段一张段表

03. 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

04. 存储管理方案中，（）可采用覆盖技术。  
 A. 单一连续存储管理  
 B. 可变分区存储管理  
 C. 段式存储管理  
 D. 段页式存储管理

05. 在可变分区分配方案中，某一进程完成后，系统回收其主存空间并与相邻空闲区合并为此需修改空闲区表，造成空闲区数减 1 的情况是（）。  
 A. 无上邻空闲区也无下邻空闲区  
 B. 有上邻空闲区但无下邻空闲区  
 C. 有下邻空闲区但无上邻空闲区  
 D. 有上邻空闲区也有下邻空闲区

06. 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

07. 某段表的内容见右表，一逻辑地址为(2, 154)，它对应的物理地址为（）。  
 A.  $120K + 2$   
 B.  $480K + 154$   
 C.  $30K + 154$   
 D.  $480K + 2$

08. 动态重定位是在作业的（）中进行的。  
 A. 编译过程  
 B. 装入过程  
 C. 链接过程  
 D. 执行过程

09. 下面的存储管理方案中，（）方式可以采用静态重定位。  
 A. 固定分区  
 B. 可变分区  
 C. 页式  
 D. 段式

10. 在可变分区管理中，采用拼接技术的目的是（）。  
 A. 合并空闲区  
 B. 合并分配区  
 C. 增加主存容量  
 D. 便于地址转换

11. 在一页式存储管理系统中，页表内容见右表。若页的大小为 4KB，则地址转换机构将逻辑地址 0 转换成的物理地址为（块号从 0 开始计算）（）。  
 A. 8192  
 B. 4096  
 C. 2048  
 D. 1024

12. 不会产生内部碎片的存储管理是（）。  
 A. 分页式存储管理  
 B. 分段式存储管理

关注公众号【乘龙考研】  
 一手更新 稳定有保障

0		
操作系統		
100K		
180K		
190K		
280K		
330K		
390K		
410K		
512K		

段号	段首址	段长度
0	120K	40K
1	760K	30K
2	480K	20K
3	370K	20K

页号	块号
0	2
1	1
3	3
4	7



0	操作系统
100K	
180K	占用
190K	
280K	占用
330K	
390K	
410K	占用
512K	

段号	段首址	段长度
0	120K	40K
1	760K	30K
2	480K	20K
3	370K	20K

页号	块号
0	2
1	1
3	3
4	7

- C. 固定分区式存储管理                    D. 段页式存储管理
13. 多进程在主存中彼此互不干扰的环境下运行，操作系统是通过（ ）来实现的。  
 A. 内存分配                    B. 内存保护                    C. 内存扩充                    D. 地址映射
14. 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
15. 首次适应算法的空间分区（ ）。  
 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
16. 采用分页或分段管理后，提供给用户的物理地址空间（ ）。  
 A. 分页支持更大的物理地址空间                    B. 分段支持更大的物理地址空间  
 C. 不能确定                    D. 一样大
17. 分页系统中的页面是为（ ）。  
 A. 用户所感知的                    B. 操作系统所感知的  
 C. 编译系统所感知的                    D. 连接装配程序所感知的
18. 页式存储管理中，页表的始地址存放在（ ）中。  
 A. 内存                    B. 存储页表                    C. 快表                    D. 寄存器
19. 对重定位存储管理方式，应（ ）。  
 A. 在整个系统中设置一个重定位寄存器                    B. 为每道程序设置一个重定位寄存器  
 C. 为每道程序设置两个重定位寄存器                    D. 为每道程序和数据都设置一个重定位寄存器
20. 采用段式存储管理时，一个程序如何分段是在（ ）时决定的。  
 A. 分配主存                    B. 用户编程                    C. 装作业                    D. 程序执行
21. 下面的（ ）方法有利于程序的动态链接。  
 A. 分段存储管理                    B. 分页存储管理  
 C. 可变式分区管理                    D. 固定式分区管理
22. 当前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为（①），称为（②）地址。  
 ① A. 1                    B. 0                    C. IP                    D. CS  
 ② A. 绝对                    B. 名义                    C. 逻辑                    D. 实
23. 可重入程序是通过（ ）方法来改善系统性能的。  
 A. 改变时间片长度                    B. 改变用户数  
 C. 提高对换速度                    D. 减少对换数量
24. 操作系统实现（ ）存储管理的代价最小。  
 A. 分区                    B. 分页                    C. 分段                    D. 段页式
25. 动态分区又称可变式分区，它是系统运行过程中（ ）动态建立的。  
 A. 在作业装入时                    B. 在作业创建时  
 C. 在作业完成时                    D. 在作业未装入时
26. 对外存对换区的管理应以（ ）为主要目标。  
 A. 提高系统吞吐量                    B. 提高存储空间的利用率

关注公众号【乘龙考研】  
一手更新 稳定有保障

- C. 降低存储费用 D. 提高换入、换出速度

27. 下列关于虚拟存储器的论述中，正确的是（ ）。

  - A. 作业在运行前，必须全部装入内存，且在运行过程中也一直驻留内存
  - B. 作业在运行前，不必全部装入内存，且在运行过程中也不必一直驻留内存
  - C. 作业在运行前，不必全部装入内存，但在运行过程中必须一直驻留内存
  - D. 作业在运行前，必须全部装入内存，但在运行过程中不必一直驻留内存

28. 在页式存储管理中选择页面的大小，需要考虑下列（ ）因素。

  - I. 页面大的好处是页表比较少
  - II. 页面小的好处是可以减少由内碎片引起的内存浪费
  - III. 影响磁盘访问时间的主要因素通常不是页面大小，所以使用时优先考虑较大的页面

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

29. 某个操作系统对内存的管理采用页式存储管理方法，所划分的页面大小（ ）。

  - A. 要根据内存大小确定
  - B. 必须相同
  - C. 要根据 CPU 的地址结构确定
  - D. 要依据外存和内存的大小确定

30. 引入段式存储管理方式，主要是为了更好地满足用户的一系列要求。下面选项中不属于这一系列要求的是（ ）。

  - A. 方便操作
  - B. 方便编程
  - C. 共享和保护
  - D. 动态链接和增长

31. 存储管理的目的是（ ）。

  - A. 方便用户
  - B. 提高内存利用率
  - C. 方便用户和提高内存利用率
  - D. 增加内存实际容量

32. 对主存储器的访问，（ ）。

  - A. 以块（即页）或段为单位
  - B. 以字节或字为单位
  - C. 随存储器的管理方案不同而异
  - D. 以用户的逻辑记录为单位

33. 把作业空间中使用的逻辑地址变为内存中的物理地址称为（ ）。

  - A. 加载
  - B. 重定位
  - C. 物理化
  - D. 逻辑化

34. 以下存储管理方式中，不适合多道程序设计系统的是（ ）。

  - A. 单用户连续分配
  - B. 固定式分区分配
  - C. 可变式分区分配
  - D. 分页式存储管理方式

35. 在分页存储管理中，主存的分配（ ）。

  - A. 以物理块为单位进行
  - B. 以作业的大小进行
  - C. 以物理段进行
  - D. 以逻辑记录大小进行

36. 在段式分配中，CPU 每次从内存中取一次数据需要（ ）次访问内存。

  - A. 1
  - B. 3
  - C. 2
  - D. 4

37. 在段页式分配中，CPU 每次从内存中取一次数据需要（ ）次访问内存。

  - A. 1
  - B. 3
  - C. 2
  - D. 4

38. （ ）存储管理方式提供一维地址结构。

  - A. 分段
  - B. 分页
  - C. 分段和段页式
  - D. 以上答案都不正确

39. 操作系统采用分页存储管理方式，要求（ ）。

  - A. 每个进程拥有一张页表，且进程的页表驻留在内存中

- B. 每个进程拥有一张页表，但只有执行进程的页表驻留在内存中  
 C. 所有进程共享一张页表，以节约有限的内存空间，但页表必须驻留在内存中  
 D. 所有进程共享一张页表，只有页表中当前使用的页面必须驻留在内存中，以最大限度地节省有限的内存空间
40. 在分段存储管理方式中，( )。  
 A. 以段为单位，每段是一个连续存储区      B. 段与段之间必定不连续  
 C. 段与段之间必定连续                          D. 每段是等长的
41. 段页式存储管理汲取了页式管理和段式管理的长处，其实现原理结合了页式和段式管理的基本思想，即( )。  
 A. 用分段方法来分配和管理物理存储空间，用分页方法来管理用户地址空间  
 B. 用分段方法来分配和管理用户地址空间，用分页方法来管理物理存储空间  
 C. 用分段方法来分配和管理主存空间，用分页方法来管理辅存空间  
 D. 用分段方法来分配和管理辅存空间，用分页方法来管理主存空间
42. 以下存储管理方式中，会产生内部碎片的是( )。  
 I. 分段虚拟存储管理                              II. 分页虚拟存储管理  
 III. 段页式分区管理                              IV. 固定式分区管理  
 A. I、II、III                                      B. III、IV                              C. 仅 II                                      D. II、III、IV
43. 下列关于页式存储的论述中，正确的是( )。  
 I. 在页式存储管理中，若关闭 TLB，则每当访问一条指令或存取一个操作数时都要访问 2 次内存  
 II. 页式存储管理不会产生内部碎片  
 III. 页式存储管理中的页面是为用户所感知的  
 IV. 页式存储方式可以采用静态重定位  
 A. I、II、IV                                      B. I、IV                                    C. 仅 I                                      D. 全都正确
44. 【2009 统考真题】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
45. 【2009 统考真题】一个分段存储管理系统中，地址长度为 32 位，其中段号占 8 位，则最大段长是( )。  
 A.  $2^8$ B    B.  $2^{16}$ B    C.  $2^{24}$ B    D.  $2^{32}$ B
46. 【2010 统考真题】某基于动态分区存储管理的计算机，其主存容量为 55MB（初始为空），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
47. 【2010 统考真题】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
48. 【2011 统考真题】在虚拟内存管理中，地址变换机构将逻辑地址转换为物理地址，形成该逻辑地址的阶段是( )。

- A. 编辑      B. 编译      C. 链接      D. 装载

49. 【2014 统考真题】现有一个容量为 10GB 的磁盘分区，磁盘空间以簇为单位进行分配，簇的大小为 4KB，若采用位图法管理该分区的空闲空间，即用一位标识一个簇是否被分配，则存放该位图所需的簇为（ ）个。

- A. 80      B. 320      C. 80K      D. 320K

50. 【2014 统考真题】下列选项中，属于多级页表优点的是（ ）。

- A. 加快地址变换速度      B. 减少缺页中断次数  
C. 减少页表项所占字节数      D. 减少页表所占的连续内存空间

51. 【2016 统考真题】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

52. 【2017 统考真题】某计算机按字节编址，其动态分区内存管理采用最佳适应算法，每次分配和回收内存后都对空闲分区链重新排序。当前空闲分区信息如下表所示。

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

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

- A. 3, 20K, 380KB      B. 3, 500K, 80KB      C. 4, 20K, 180KB      D. 4, 500K, 80KB

53. 【2019 统考真题】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<sub>1</sub> 和 P<sub>2</sub> 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<sub>1</sub> 和 P<sub>2</sub> 中应该具有相同的段号  
C. P<sub>1</sub> 和 P<sub>2</sub> 共享段 S 在共享段表中的段表项  
D. P<sub>1</sub> 和 P<sub>2</sub> 都不再使用段 S 时才回收段 S 所占的内存空间

关注公众号【乘龙考研】  
一手更新 稳定有保障

54. 【2019 统考真题】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 (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

55. 【2019 统考真题】在下列动态分区分配算法中，最容易产生内存碎片的是（ ）。

- A. 首次适应算法      B. 最坏适应算法  
C. 最佳适应算法      D. 循环首次适应算法

56. 【2021 统考真题】在采用二级页表的分页系统中，CPU 页表基址寄存器中的内容是（ ）。

- A. 当前进程的一级页表的起始虚拟地址      B. 当前进程的一级页表的起始物理地址  
C. 当前进程的二级页表的起始虚拟地址      D. 当前进程的二级页表的起始物理地址

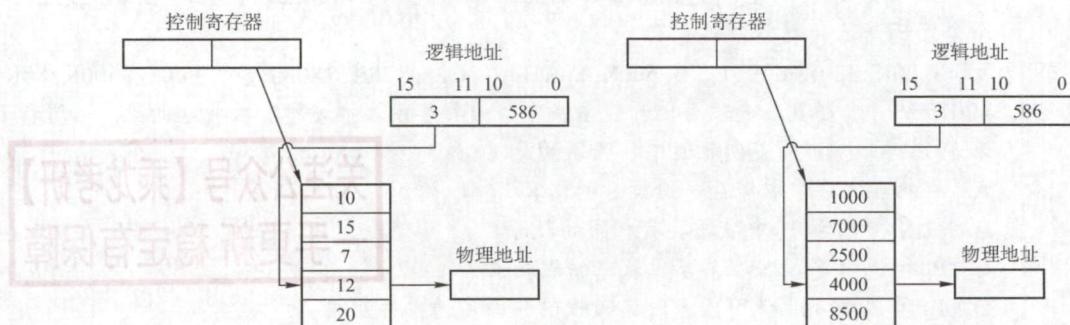
## 二、综合应用题

01. 动态分区与固定分区分配方式相比，是否解决了碎片问题？
02. 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	220K
5	96KB	530K

关注公众号【乘龙考研】  
一手更新 稳定有保障

03. 某操作系统采用段式管理，用户区主存为 512KB，空闲块链入空块表，分配时截取空块的前半部分（小地址部分）。初始时全部空闲。执行申请、释放操作序列 reg(300KB), reg(100KB), release(300KB), reg(150KB), reg(50KB), reg(90KB) 后：
- 1) 采用最先适配，空块表中有哪些空块？（指出大小及始址）
  - 2) 采用最佳适配，空块表中有哪些空块？（指出大小及始址）
  - 3) 若随后又要申请 80KB，针对上述两种情况会产生什么后果？这说明了什么问题？
04. 下图给出了页式和段式两种地址变换示意（假定段式变换对每段不进行段长越界检查，即段表中无段长信息）。
- 1) 指出这两种变换各属于何种存储管理。
  - 2) 计算出这两种变换所对应的物理地址。



05. 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

06. 页式存储管理允许用户的编程空间为 32 个页面（每页 1KB），主存为 16KB。如有一用户程序为 10 页长，且某时刻该用户程序页表见右表。

若分别遇到三个逻辑地址 0AC5H, 1AC5H, 3AC5H 处的操作，计算并说明存储管理系统将如何处理。

07. 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，其页号为 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）。

08. 某操作系统存储器采用页式存储管理，页面大小为 64B，假定一进程的代码段的长度为 702B，页表见表 A，该进程在快表中的页表见表 B。现进程有如下访问序列：其逻辑地址为八进制的 0105, 0217, 0567, 01120, 02500。试问给定的这些地址能否进行转换？

表 A 进程页表

页号	页帧号	页号	页帧号
0	F0	6	F6
1	F1	7	F7
2	F2	8	F8
3	F3	9	F9
4	F4	10	F10
5	F5	6	F6

逻辑页号	物理块号
0	8
1	7
2	4
3	10

表 B 快表

页号	页帧号
0	F0
1	F1
2	F2
3	F3
4	F4

09. 某一页式系统，其页表存放在主存中：

- 1) 若对主存的一次存取需  $1.5\mu s$ ，问实现一次页面访问时存取时间是多少？
- 2) 若系统有快表且其平均命中率为 85%，而页表项在快表中的查找时间为  $0.2\mu s$ ，若快表的命中率是 85%，则有效存取时间是多少？若快表的命中率为 50%，则有效存取时间是多少？

10. 在页式、段式和段页式存储管理中，当访问一条指令或数据时，各需要访问内存几次？其过程如何？假设一个页式存储系统具有快表，多数活动页表项都可以存在其中。若页表存放在内存中，内存访问时间是  $1\mu s$ ，检索快表的时间为  $0.2\mu s$ ，若快表的命中率是 85%，则有效存取时间是多少？若快表的命中率为 50%，则有效存取时间是多少？

11. 在一个分页存储管理系统中，地址空间分页（每页 1KB），物理空间分块，设主存总容量是 256KB，描述主存分配情况的位示图如下图所示（0 表示未分配，1 表示已分配），此时作业调度程序选中一个长为 5.2KB 的作业投入内存。试问：

- 1) 为该作业分配内存后（分配内存时，首先分配低地址的内存空间），请填写该作业的页表内容。
- 2) 页式存储管理有无内存碎片存在？若有，会存在哪种内存碎片？为该作业分配内存后，会产生内存碎片吗？如果产生，那么大小为多少？
- 3) 假设一个 64MB 内存容量的计算机，采用页式存储管理（页面大小为 4KB），内存分配采用位示图方式管理，请问位示图将占用多大的内存？

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	1	0	0	0	1	0	1	1
0	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....

页号	块号（从 0 开始编址）

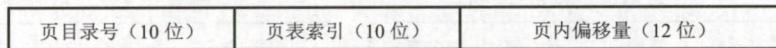
12. 【2013 统考真题】某计算机主存按字节编址，逻辑地址和物理地址都是 32 位，页表项大小为 4B。请回答下列问题：

- 1) 若使用一级页表的分页存储管理方式，逻辑地址结构为



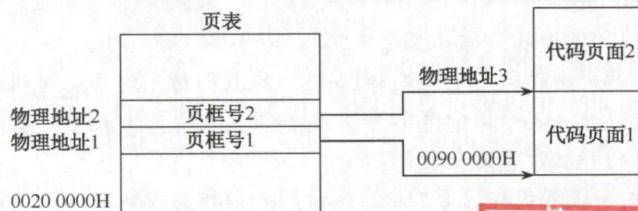
则页的大小是多少字节？页表最大占用多少字节？

- 2) 若使用二级页表的分页存储管理方式，逻辑地址结构为



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 8 KB，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。



### 3.1.9 答案与解析

#### 一、单项选择题

01. D

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。

02. C

段页式系统中，进程首先划分为段，每段再进一步划分为页。

03. C

内存保护是内存管理的一部分，是操作系统的任务，但是出于安全性和效率考虑，必须由硬件实现，所以需要操作系统和硬件机构的合作来完成。

关注公众号【乘龙考研】  
一手更新 稳定有保障

**04. A**

覆盖技术是早期在单一连续存储管理中使用的扩大存储容量的一种技术，它同样可用于固定分区分配的存储管理。

**05. D**

将上邻空闲区、下邻空闲区和回收区合并为一个空闲区，因此空闲区数反而减少了一个。而仅有上邻空闲区或下邻空闲区时，空闲区数并不减少。

**06. C**

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闲分区。从图中可以看出应选择大小为 60KB 的空闲分区，其首地址为 330K。

**07. B**

段号为 2，其对应的首地址为 480K，段长度为 20K，大于 154，所以逻辑地址(2, 154)对应的物理地址为  $480K + 154$ 。

**08. D**

静态装入是指在编程阶段就把物理地址计算好。

可重定位是指在装入时把逻辑地址转换成物理地址，但装入后不能改变。

动态重定位是指在执行时再决定装入的地址并装入，装入后有可能会换出，所以同一个模块在内存中的物理地址是可能改变的。

动态重定位是指在作业运行过程中执行到一条访存指令时，再把逻辑地址转换为主存中的物理地址，实际中是通过硬件地址转换机制实现的。

**09. A**

固定分区方式中，作业装入后位置不再改变，可以采用静态重定位。其余三种管理方案均可能在运行过程中改变程序位置，静态重定位不能满足其要求。

**10. A**

在可变分区管理中，回收空闲区时采用拼接技术对空闲区进行合并。

**11. A**

按页表内容可知，逻辑地址 0 对应块号 2，页大小为 4KB，因此转换成的物理地址为  $2 \times 4K = 8K = 8192$ 。

**12. B**

分页式存储管理有内部碎片，分段式存储管理有外部碎片，固定分区存储管理方式有内部碎片，段页式存储管理方式有内部碎片。

**13. B**

多进程的执行通过内存保护实现互不干扰，如页式管理中有页地址越界保护，段式管理中有段地址越界保护。

**14. A**

最佳适应算法要求从剩余的空闲分区中选出最小且满足存储要求的分区，空闲区应按长度递增登记在空闲区表中。

**15. C**

首次适应算法的空闲分区按地址递增的次序排列。

**16. C**

页表和段表同样存储在内存中，系统提供给用户的物理地址空间为总空间大小减去页表或段

关注公众号【乘龙考研】  
一手更新 稳定有保障

表的长度。由于页表和段表的长度不能确定，所以提供给用户的物理地址空间大小也不能确定。

**17. B**

内存分页管理是在硬件和操作系统层面实现的，对用户、编译系统、连接装配程序等上层是不可见的。

**18. D**

页表的功能由一组专门的存储器实现，其始址放在页表基址寄存器（PTBR）中。这样才能满足在地址变换时能够较快地完成逻辑地址和物理地址之间的转换。

**19. A**

为使地址转换不影响到指令的执行速度，必须有硬件地址变换结构的支持，即需在系统中增设一个重定位寄存器，用它来存放程序（数据）在内存中的始址。在执行程序或访问数据时，真正访问的内存地址由相对地址与重定位寄存器中的地址相加而成，这时将始址存入重定位寄存器，之后的地址访问即可通过硬件变换实现。因为系统处理器在同一时刻只能执行一条指令或访问数据，所以为每道程序（数据）设置一个寄存器没有必要（同时也不现实，因为寄存器是很昂贵的硬件，而且程序的道数是无法预估的），而只需在切换程序执行时重置寄存器内容。

**20. B**

分段是指在用户编程时，将程序按照逻辑划分为几个逻辑段。

**21. A**

程序的动态链接与程序的逻辑结构相关，分段存储管理将程序按照逻辑段进行划分，因此有利于其动态链接。其他的内存管理方式与程序的逻辑结构无关。

**22. B、C**

编译后一个目标程序所限定的地址范围称为该作业的逻辑地址空间。换句话说，地址空间仅指程序用来访问信息所用的一系列地址单元的集合。这些单元的编号称为逻辑地址。通常，编译地址都是相对始址“0”的，因而也称逻辑地址为相对地址。

**注意：**区分编译后形成的逻辑地址和链接后形成的最终逻辑地址。

**23. D**

可重入程序主要是通过共享来使用同一块存储空间的，或通过动态链接的方式将所需的程序段映射到相关进程中去，其最大的优点是减少了对程序段的调入/调出，因此减少了对换数量。

**24. A**

实现分页、分段和段页式存储管理需要特定的数据结构支持，如页表、段表等。为了提高性能，还需要硬件提供快存和地址加法器等，代价高。分区存储管理是满足多道程序设计的最简单的存储管理方案，特别适合嵌入式等微型设备。

**25. A**

动态分区时，在系统启动后，除操作系统占据一部分内存外，其余所有内存空间是一个大空闲区，称为自由空间。若作业申请内存，则从空闲区中划出一个与作业需求量相适应的分区分配给该作业，将作业创建为进程，在作业运行完毕后，再收回释放的分区。

**26. D**

内存管理是为了提高内存的利用率，引入覆盖和交换技术，即在较小的内存空间中采用重复使用的方法来节省存储空间，但它付出的代价是需要消耗更多的处理器时间，因此实际上是一种以时间换空间的技术。为此，从节省处理器时间上来讲，换入、换出速度越快，付出的时间代价就越小，反之就越大，当时间代价大到一定程度时，覆盖和交换技术就没有了意义。

**27. B**

在非虚拟存储器中，作业必须全部装入内存且在运行过程中也一直驻留内存；在虚拟存储器中，作业不必全部装入内存且在运行过程中也不用一直驻留内存，这是虚拟存储器和非虚拟存储器的主要区别之一。

**28. C**

页面大，用于管理页面的页表就少，但是页内碎片会比较大；页面小，用于管理页面的页表就大，但是页内碎片小。通过适当的计算可以获得较佳的页面大小和较小的系统开销。

**29. B**

页式管理中很重要的一个问题就是页面大小如何确定。确定页面大小有很多因素，如进程的平均大小、页表占用的长度等。而一旦确定，所有的页面就是等长的（一般取2的整数幂倍），以便易于系统管理。

**30. A**

引入段式存储管理方式，主要是为了满足用户的下列要求：方便编程、分段共享、分段保护、动态链接和动态增长。

**31. C**

存储管理的目的有两个：一个是方便用户，二是提高内存利用率。

**32. B**

这里是指主存的访问，不是主存的分配。对主存的访问是以字节或字为单位的。例如，在页式管理中，不仅要知道块号，而且要知道页内偏移。

**33. B**

在一般情况下，一个作业在装入时分配到的内存空间和它的地址空间是不一致的，因此，作业在CPU上运行时，其所要访问的指令、数据的物理地址和逻辑地址是不同的。显然，若在作业装入或执行时，不对有关的地址部分加以相应的修改，则会导致错误的结果。这种将作业的逻辑地址变为物理地址的过程称为地址重定位。

**34. A**

单用户连续分配管理方式只适用于单用户、单任务的操作系统，不适用于多道程序设计。

**35. A**

在分页存储管理中，逻辑地址分配是按页为单位进行分配的，而主存的分配即物理地址分配是以内存块为单位分配的。

**36. C**

在段式分配中，取一次数据时先从内存查找段表，再拼成物理地址后访问内存，共需要2次内存访问。

**37. B**

在段页式分配中，取一次数据时先从内存查找段表，再访问内存查找相应的页表，最后拼成物理地址后访问内存，共需要3次内存访问。

**38. B**

分页存储管理中，作业地址空间是一维的，即单一的线性地址空间，程序员只需要一个记忆符来表示地址。在分段存储分配管理中，段之间是独立的，而且段长不定长，而页长是固定的，因此作业地址空间是二维的，程序员在标识一个地址时，既需给出段名，又需给出段内地址。简言之，确定一个地址需要几个参数，作业地址空间就是几维的。

**39. A**

在多个进程并发执行时，所有进程的页表大多数驻留在内存中，在系统中只设置一个页表寄

存器 (PTR)，它存放页表在内存中的始址和长度。平时，进程未执行时，页表的始址和页表长度存放在本进程的 PCB 中，当调度到某进程时，才将这两个数据装入页表寄存器中。每个进程都有一个单独的逻辑地址，有一张属于自己的页表。

#### 40. A

在分段存储管理方式中，以段为单位进行分配，每段是一个连续存储区，每段不一定等长，段与段之间可连续，也可不连续。

#### 41. B

段页式存储管理兼有页式管理和段式管理的优点，采用分段方法来分配和管理用户地址空间，采用分页方法来管理物理存储空间。但它的开销要比段式和页式管理的开销大。

#### 42. D

只要是固定的分配就会产生内部碎片，其余的都会产生外部碎片。若固定和不固定同时存在（例如段页式），则仍视为固定。分段虚拟存储管理：每段的长度都不一样（对应不固定），所以会产生外部碎片。分页虚拟存储管理：每页的长度都一样（对应固定），所以会产生内部碎片。段页式分区管理：既有固定，又有不固定，以固定为主，所以会有内部碎片。固定式分区管理：很明显固定，会产生内部碎片。综上分析，II、III、IV 选项会产生内部碎片。

#### 43. C

I 正确：关闭 TLB 后，每当访问一条指令或存取一个操作数时都要先访问页表（内存中），得到物理地址后，再访问一次内存进行相应操作。II 错误：记住，凡是分区固定的都会产生内部碎片，而无外部碎片。III 错误：页式存储管理对于用户是透明的。IV 错误：静态重定位是在程序运行之前由装配程序完成的，必须分配其要求的全部连续内存空间。而页式存储管理方案是将程序离散地分成若干页（块），从而可以将程序装入不连续的内存空间，显然静态重定位不能满足其要求。

#### 44. A

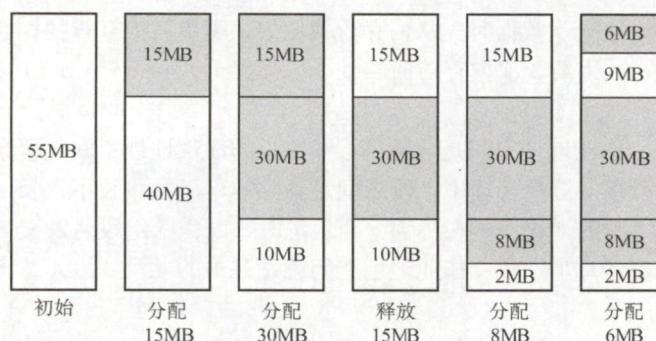
每个进程都拥有自己独立的进程空间，若一个进程在运行时所产生的地址在其地址空间之外，则发生地址越界，因此需要进行界地址保护，即当程序要访问某个内存单元时，由硬件检查是否允许，若允许则执行，否则产生地址越界中断。

#### 45. C

分段存储管理的逻辑地址分为段号和位移量两部分，段内位移的最大值就是最大段长。地址长度为 32 位，段号占 8 位，因此位移量占  $32 - 8 = 24$  位，因此最大段长为  $2^{24}$ B。

#### 46. B

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闲分区。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的大小为 9MB。

#### 47. B

页大小为  $2^{10}$ B，页表项大小为 2B，因此一页可以存放  $2^9$  个页表项，逻辑地址空间大小为  $2^{16}$  页，即共需  $2^{16}/2^9 = 2^7 = 128$  个页面保存页表项，即页目录表中包含表项的个数至少是 128。

#### 48. C

编译后的程序需要经过链接才能装载，而链接后形成的目标程序中的地址也就是逻辑地址。以 C 语言为例：C 程序经过预处理→编译→汇编→链接产生了可执行文件，其中链接的前一步是产生可重定位的二进制目标文件。C 语言采用源文件独立编译的方法，如程序 main.c, file1.c, file2.c, file1.h, file2.h 在链接的前一步生成了 main.o, file1.o, file2.o，这些目标模块的逻辑地址都从 0 开始，但只是相对于该模块的逻辑地址。链接器将这三个文件、libc 和其库文件链接成一个可执行文件，从而形成整个程序的完整逻辑地址空间。

例如，file1.o 的逻辑地址为 0~1023，main.o 的逻辑地址为 0~1023，假设链接时将 file1.o 链接在 main.o 之后，则链接之后 file1.o 对应的逻辑地址应为 1024~2047。

#### 49. A

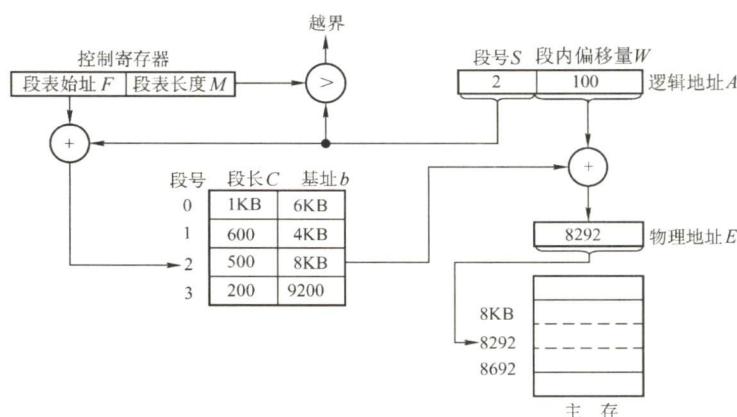
簇的总数为  $10GB/4KB = 2.5M$ ，用一位标识一簇是否被分配，整个磁盘共需要 2.5Mb，即需要  $2.5M/8 = 320KB$ ，因此共需要  $320KB/4KB = 80$  簇，选 A。

#### 50. D

多级页表不仅不会加快地址的变换速度，还会因为增加更多的查表过程，使地址变换速度减慢；也不会减少缺页中断的次数，反而如果访问过程中多级的页表都不在内存中，会大大增加缺页的次数，并不会减少页表项所占的字节数，而多级页表能够减少页表所占的连续内存空间，即当页表太大时，将页表再分级，把每张页表控制在一页之内，减少页表所占的连续内存空间，因此选 D。

#### 51. D

分段系统的逻辑地址 A 到物理地址 E 之间的地址变换过程如下：



关注公众号【乘龙考研】  
一手更新 稳定有保障

- ① 从逻辑地址 A 中取出前几位为段号 S，后几位为段内偏移量 W，注意段式存储管理的题目中，逻辑地址一般以二进制数给出，而在页式存储管理中，逻辑地址一般以十进制数给出，读者要具体问题具体分析。
- ② 比较段号 S 和段表长度 M，若  $S \geq M$ ，则产生越界异常，否则继续执行。

③ 段表中段号  $S$  对应的段表项地址 = 段表始址  $F$  + 段号  $S \times$  段表项长度  $M$ ，取出该段表项的前几位得到段长  $C$ 。若段内偏移量  $\geq C$ ，则产生越界异常，否则继续执行。从这句话可以看出，段表项实际上只有两部分，前几位是段长，后几位是始址。

④ 取出段表项中该段的基址  $b$ ，计算  $E = b + W$ ，用得到的物理地址  $E$  去访问内存。

题目中段号为 2 的段长为 300，小于段内地址 400，因此发生越界异常，D 正确。

### 52. B

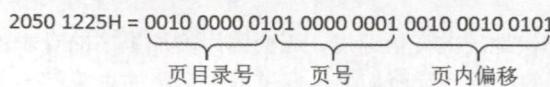
回收始址为 60K、大小为 140KB 的分区时，它与表中第一个分区和第四个分区合并，成为始址为 20K、大小为 380KB 的分区，剩余 3 个空闲分区。在回收内存后，算法会对空闲分区链按分区大小由小到大进行排序，表中的第二个分区排第一，所以选择 B。

### 53. B

段的共享是通过两个作业的段表中相应表项指向被共享的段的同一个物理副本实现的，因此在内存中仅保存一份段  $S$  的内容，A 正确。段  $S$  对于进程  $P_1, P_2$  来说，使用位置可能不同，所以在不同进程中的逻辑段号可能不同，B 错误。段表项存放的是段的物理地址（包括段始址和段长度），对于共享段  $S$  来说物理地址唯一，C 正确。为了保证进程可以顺利使用段  $S$ ，段  $S$  必须确保在没有任何进程使用它（可在段表项中设置共享进程计数）后才能被删除，D 正确。

### 54. A

题中给出的是十六进制地址，首先将它转化为二进制地址，然后用二进制地址去匹配题中对应的地址结构。转换为二进制地址和地址结构的对应关系如下图所示。



前 10 位、11~20 位、21~32 位分别对应页目录号、页号和页内偏移。把页目录号、页号单独拿出，转换为十六进制时缺少的位数在高位补零，0000 1000 0001，0001 0000 0001 分别对应 081H，101H，选项 A 正确。

### 55. C

最佳适应算法总是匹配与当前大小要求最接近的空闲分区，但是大多数情况下空闲分区的大小不可能完全和当前要求的大小相等，几乎每次分配内存都会产生很小的难以利用的内存块，所以最佳适应算法最容易产生最多的内存碎片，C 正确。

### 56. B

在多级页表中，页表基址寄存器存放的是顶级页表的起始物理地址，故存放的是一级页表的起始物理地址。

## 二、综合应用题

### 01. 【解答】

动态分区和固定分区分配方式相比，内存空间的利用率要高一些。但是，总会存在一些分散的较小空闲分区，即外部碎片，它们存在于已分配的分区之间，不能充分利用。可以采用拼接技术加以解决。固定分区分配方式存在内部碎片，而无外部碎片；动态分区分配方式存在外部碎片，无内部碎片。

### 02. 【解答】

采用首次适应算法时，96KB 大小的作业进入 4 号空闲分区，20KB 大小的作业进入 1 号空闲分区，这时空闲分区如下表所示。

分区号	大 小	始 址
1	12KB	120K
2	10KB	150K
3	5KB	200K
4	122KB	316K
5	96KB	530K

此时再无空闲分区可以满足 200KB 大小的作业，所以该作业序列请求无法满足。

采用最佳适应算法时，作业序列分别进入 5, 1, 4 号空闲分区，可以满足其请求。分配处理之后的空闲分区表见下表：

分区号	大 小	始 址
1	12KB	120K
2	10KB	150K
3	5KB	200K
4	18KB	420K



### 03. 【解答】

- 1) 最先适配的内存分配情况如下图中的(a)所示。

内存中的空块为：

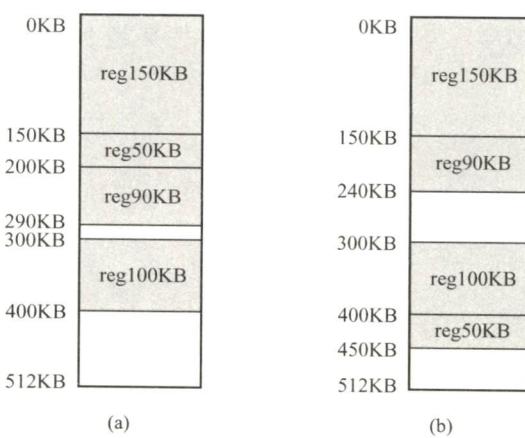
第一块：始址 290K，大小 10KB；第二块：始址 400K，大小 112KB。

- 2) 最佳适配的内存分配情况如下图中的(b)所示。

内存中的空块为：

第一块：始址 240K，大小 60KB；第二块：超始地址 450K，大小 62KB。

- 3) 若随后又要申请 80KB，则最先适配算法可以分配成功，而最佳适配算法则没有足够大的空闲区分配。这说明最先适配算法尽可能地使用了低地址部分的空闲区域，留下了高地址部分的大的空闲区，更有可能满足进程的申请。



### 04. 【解答】

- 1) 由题图所示的逻辑地址结构可知：页或段的最大个数为  $2^5 = 32$ 。如果左图是段式管理，那么段始址 12 加上偏移量 586，远超第 1 段的段始址 15，超过第 4 段的段始址 20，所以左图是页式变换，而右图满足段式变换。对于页式管理，由逻辑地址的位移量位数可知，

一页的大小为 2KB。

- 2) 对图中的页式地址变换，其物理地址为  $12 \times 2048 + 586 = 25162$ ；对图中的段式地址变换，其物理地址为  $4000 + 586 = 4586$ 。

#### 05. 【解答】

- 1) 由段表知，第 0 段内存始址为 210，段长为 500，因此逻辑地址(0, 430)是合法地址，对应的物理地址为  $210 + 430 = 640$ 。
- 2) 由段表知，第 1 段内存始址为 2350，段长为 20，因此逻辑地址(1, 10)是合法地址，对应的物理地址为  $2350 + 10 = 2360$ 。
- 3) 由段表知，第 2 段内存始址为 100，段长为 90，逻辑地址(2, 500)的段内位移 500 超过了段长，因此为非法地址。
- 4) 由段表知，第 3 段内存始址为 1350，段长为 590，因此逻辑地址(3, 400)是合法地址，对应的物理地址为  $1350 + 400 = 1750$ 。
- 5) 由段表知，第 4 段内存始址为 1938，段长为 95，逻辑地址(4, 112)的段内位移 112 超过了段长，因此为非法地址。
- 6) 由段表知，不存在第 5 段，因此逻辑地址(5, 32)为非法地址。

#### 06. 【解答】

页面大小为 1KB，所以低 10 位为页内偏移地址；用户编程空间为 32 个页面，即逻辑地址高 5 位为虚页号；主存为 16 个页面，即物理地址高 4 位为主存块号。

逻辑地址 0AC5H 转换为二进制是 000 1010 1100 0101B，虚页号为 2(00010B)，映射至物理块号 4，因此系统访问物理地址 12C5H(01 0010 1100 0101B)。

逻辑地址 1AC5H 转换为二进制是 001 1010 1100 0101B，虚页号为 6(00110B)，不在页面映射表中，会产生缺页中断，系统进行缺页中断处理。

逻辑地址 3AC5H 转换为二进制是 011 1010 1100 0101B，页号为 14，而该用户程序只有 10 页，因此系统产生越界中断。

**注意：**在将十六进制地址转换为二进制地址时，我们可能会习惯性地写为 16 位，这是容易犯错的细节。例如，题中的逻辑地址为 15 位，物理地址为 14 位。逻辑地址 0AC5H 的二进制表示为 000 1010 1100 0101B，对应物理地址 12C5H 的二进制表示为 01 0010 1100 0101B。这一点应该引起注意。

#### 07. 【解答】

- 1) 页面的大小为  $(64/16)KB = 4KB$ ，该进程共有 4 页，所以该进程的总长度为  $4 \times 4KB = 16KB$ 。
- 2) 页面大小为 4KB，因此低 12 位为页内偏移地址；主存分为 16 块，因此内存物理地址高 4 位为主存块号。  
页号为 0 的页面被装入主存的第 9 块，因此该地址在内存中的始址为 1001 0000 0000 0000B，即 9000H。  
页号为 1 的页面被装入主存的第 0 块，因此该地址在内存中的始址为 0000 0000 0000 0000B，即 0000H。  
页号为 2 的页面被装入主存的第 1 块，因此该地址在内存中的始址为 0001 0000 0000 0000B，即 1000H。  
页号为 3 的页面被装入主存的第 14 块，因此该地址在内存中的始址为 1110 0000 0000 0000B，即 E000H。

- 3) 逻辑地址为(0, 0), 因此内存地址为 $(0, 0) = 1001\ 0000\ 0000\ 0000B$ , 即 9000H。  
 逻辑地址为(1, 72), 因此内存地址为 $(0, 72) = 0000\ 0000\ 0100\ 1000B$ , 即 0048H。  
 逻辑地址为(2, 1023), 因此内存地址为 $(1, 1023) = 0001\ 0011\ 1111\ 1111$ , 即 13FFH。  
 逻辑地址为(3, 99), 因此内存地址为 $(14, 99) = 1110\ 0000\ 0110\ 0011$ , 即 E063H。

### 08. 【解答】

要注意题目中的逻辑地址使用哪种进制的数给出, 若是十进制, 则一般通过整数除法和求余得到页号和页内偏移, 若用其他进制给出, 则一般转换成二进制, 然后按照地址结构划分为页号部分和页内偏移部分, 再把页号和页内偏移计算出来。

页面大小为 64B, 因此页内位移为 6 位, 进程代码段长度为 702B, 因此需要 11 个页面, 编号为 0~10。

- 1) 八进制逻辑地址 0105 的二进制表示为 0 0100 0101B。逻辑页号为 1, 此页号可在快表中查找到, 得页帧号为 F1; 页内位移为 5, 因此物理地址为(F1, 5)。
- 2) 八进制逻辑地址 0217 的二进制表示为 0 1000 1111B。逻辑页号为 2, 此页号可在快表中查找到, 得页帧号为 F2; 页内位移为 15, 因此物理地址为(F2, 15)。
- 3) 八进制逻辑地址 0567 的二进制表示为 1 0111 0111B。逻辑页号为 5, 此页号不在快表中, 在内存页表中可以查找到, 得页帧号为 F5; 页内位移为 55, 因此物理地址为(F5, 55)。
- 4) 八进制逻辑地址 01120 的二进制表示为 0010 0101 0000B。逻辑页号为 9, 此页号不在快表中, 在内存页表中可以查找到, 得页帧号为 F9; 页内位移为 16, 因此物理地址为(F9, 16)。
- 5) 八进制逻辑地址 02500 的二进制表示为 0101 0100 0000B。逻辑页号为 21, 此页号已超过页表的最大页号 10, 因此产生越界中断。

**注意:** 根据题中条件无法得知逻辑地址位数, 所以其二进制表示中, 其位数并不一致, 只是根据八进制表示进行转换。若已知逻辑地址空间大小或位数, 则二进制表示必须保持一致。

### 09. 【解答】

页表在主存时, 实现一次存取需要访问主存两次: 第一次是访问页表, 获得所需访问数据所在页面的物理地址; 第二次才是根据这个物理地址存取数据。

- 1) 因为页表在主存, 所以 CPU 必须访问主存两次, 即实现一次页面访问的存取时间是

$$1.5 \times 2 = 3\mu s$$

- 2) 系统增加快表后, 在快表中找到页表项的概率为 85%, 所以实现一次页面访问的存取时间为

$$0.85 \times (0 + 1.5) + (1 - 0.85) \times 2 \times 1.5 = 1.725\mu s$$

### 10. 【解答】

- 1) 在页式存储管理中, 访问指令或数据时, 首先要访问内存中的页表, 查找到指令或数据所在页面对应的页表项, 然后根据页表项查找访问指令或数据所在的内存页面。需要访问内存 2 次。

段式存储管理同理, 需要访问内存 2 次。

段页式存储管理, 首先要访问内存中的段表, 然后访问内存中的页表, 最后访问指令或数据所在的内存页面, 需要访问内存 3 次。

对于比较复杂的情况, 如多级页表, 若页表划分为 N 级, 则需要访问内存  $N + 1$  次。若系统中有快表, 则在快表命中时, 只需要访问内存 1 次。

- 2) 按 1) 中的访问过程分析, 有效存取时间为

$$(0.2 + 1) \times 85\% + (0.2 + 1 + 1) \times (1 - 85\%) = 1.35\mu s$$

3) 同理可计算得

$$(0.2 + 1) \times 50\% + (0.2 + 1 + 1) \times (1 - 50\%) = 1.7 \mu s$$

从结果可以看出，快表的命中率对访存时间影响非常大。当命中率从 85% 降低到 50% 时，有效存取时间增加  $0.35 \mu s$ 。因此在页式存储系统中，应尽可能地提高快表的命中率，从而提高系统效率。

**注意：**在有快表的分页存储系统中，计算有效存取时间时，需注意访问快表与访问内存的时间关系。通常的系统中，先访问快表，未命中时再访问内存；在有些系统中，快表与内存的访问同时进行，当快表命中时就停止对内存的访问。这里题中未具体指明，我们按照前者进行计算。但若题中有具体的说明，计算时就应注意区别。

### 11. 【解答】

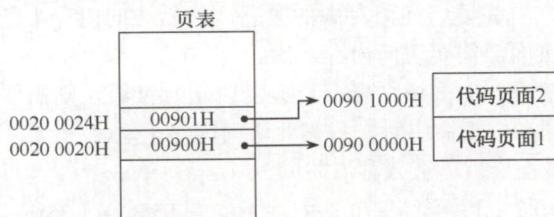
- 1) 位示图是利用二进制的一位来表示磁盘中一个盘块的使用情况，其值为“0”时表示对应盘块空闲，为“1”时表示已分配，地址空间分页，每页为 1KB，则对应的盘块大小也为 1KB，主存总容量为 256KB，可分成 256 个盘块，长 5.2KB 的作业需要占用 6 页空间，假设页号与物理块号都从 0 开始，则根据位示图可得到页表内容如下：

页 号	块 号
0	21
1	27
2	28
3	29
4	34
5	35

- 2) 页式存储管理中有内存碎片的存在，会存在内部碎片。为该作业分配内存后，会产生内存碎片，因为此作业大小为 5.2KB，占 6 页，前 5 页满，最后一页只占 0.2KB 的空间，因此内存碎片得大小为  $1KB - 0.2KB = 0.8KB$ 。
- 3) 64MB 内存，一页大小为 4KB，共可分成  $64K \times 1K / 4K = 2^{14}$  个物理盘块，在位示图中每个盘块占 1 位，共占  $2^{14}$  位空间，因为  $1B = 8$  位，所以此位示图共占 2KB 空间的内存。

### 12. 【解答】

- 1) 因为主存按字节编址，页内偏移量是 12 位，所以页大小为  $2^{12}B = 4KB$ 。  
页表项数为  $2^{20}$ ，因此该一级页表最大为  $2^{20} \times 4B = 4MB$ 。
- 2) 页目录号可表示为  $((unsigned int)(LA)) >> 22 \& 0x3FF$ 。  
页表索引可表示为  $((unsigned int)(LA)) >> 12 \& 0x3FF$ 。
- 3) 代码页面 1 的逻辑地址为 0000 8000H，表明其位于第 8 个页处，对应页表中的第 8 个页表项，所以第 8 个页表项的物理地址 = 页表始址 +  $8 \times$ 页表项的字节数 = 0020 0000H +  $8 \times 4 = 0020 0020H$ 。由此可得如下图所示的答案。



## 3.2 虚拟内存管理

在学习本节时,请读者思考以下问题:

- 1) 为什么要引入虚拟内存?
- 2) 虚拟内存空间的大小由什么因素决定?
- 3) 虚拟内存是怎么解决问题的?会带来什么问题?

读者要掌握虚拟内存解决问题的思想,了解各种替换算法的优劣,掌握虚实地址的变换方法。



### 3.2.1 虚拟内存的基本概念

#### 1. 传统存储管理方式的特征

3.1 节讨论的各种内存管理策略都是为了同时将多个进程保存在内存中,以便允许进行多道程序设计。它们都具有以下两个共同的特征:

- 1) 一次性。作业必须一次性全部装入内存后,才能开始运行。这会导致两种情况:①当作业很大而不能全部被装入内存时,将使该作业无法运行;②当大量作业要求运行时,由于内存不足以容纳所有作业,只能使少数作业先运行,导致多道程序度的下降。
- 2) 驻留性。作业被装入内存后,就一直驻留在内存中,其任何部分都不会被换出,直至作业运行结束。运行中的进程会因等待 I/O 而被阻塞,可能处于长期等待状态。

由以上分析可知,许多在程序运行中不用或暂时不用的程序(数据)占据了大量的内存空间,而一些需要运行的作业又无法装入运行,显然浪费了宝贵的内存资源。

#### 2. 局部性原理

要真正理解虚拟内存技术的思想,首先须了解著名的局部性原理。从广义上讲,快表、页高速缓存及虚拟内存技术都属于高速缓存技术,这个技术所依赖的原理就是局部性原理。局部性原理既适用于程序结构,又适用于数据结构(更远地讲,Dijkstra 关于“goto 语句有害”的著名论文也出于对程序局部性原理的深刻认识和理解)。

局部性原理表现在以下两个方面:

- 1) 时间局部性。程序中的某条指令一旦执行,不久后该指令可能再次执行;某数据被访问过,不久后该数据可能再次被访问。产生的原因是程序中存在着大量的循环操作。
- 2) 空间局部性。一旦程序访问了某个存储单元,在不久后,其附近的存储单元也将被访问,即程序在一段时间内所访问的地址,可能集中在一定的范围之内,因为指令通常是顺序存放、顺序执行的,数据也一般是以向量、数组、表等形式簇聚存储的。

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中,并使用高速缓存的层次结构实现。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。虚拟内存技术实际上建立了“内存-外存”的两级存储器结构,利用局部性原理实现高速缓存。

#### 3. 虚拟存储器的定义和特征

基于局部性原理,在程序装入时,仅须将程序当前要运行的少数页面或段先装入内存,而将其余部分暂留在外存,便可启动程序执行。在程序执行过程中,当所访问的信息不在内存时,由操作系统将所需要的部分调入内存,然后继续执行程序。另一方面,操作系统将内存中暂时不使用的内容换出到外存上,从而腾出空间存放将要调入内存的信息。这样,系统好像为用户提供了一个比实际内存容量大得多的存储器,称为虚拟存储器。

之所以将其称为虚拟存储器,是因为这种存储器实际上并不存在,只是由于系统提供了部分

装入、请求调入和置换功能后（对用户透明），给用户的感觉是好像存在一个比实际物理内存大得多的存储器。但容量大只是一种错觉，是虚的。虚拟存储器有以下三个主要特征：

- 1) 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，即只需将当前要运行的那部分程序和数据装入内存即可开始运行。以后每当要运行到尚未调入的那部分程序时，再将它调入。多次性是虚拟存储器最重要的特征。
- 2) 对换性。是指无须在作业运行时一直常驻内存，在进程运行期间，允许将那些暂不使用的程序和数据从内存调至外存的对换区（换出），待以后需要时再将它们从外存调至内存（换进）。正是由于对换性，才使得虚拟存储器得以正常运行。
- 3) 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。这是虚拟存储器所表现出的最重要特征，也是实现虚拟存储器的最重要目标。

#### 4. 虚拟内存技术的实现

虚拟内存技术允许将一个作业分多次调入内存。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，而且也无法从逻辑上扩大内存容量。因此，虚拟内存的实现需要建立在离散分配的内存管理方式的基础上。

虚拟内存的实现有以下三种方式：

- 请求分页存储管理。
- 请求分段存储管理。
- 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。一般需要的支持有以下几个方面：

- 一定容量的内存和外存。
- 页表机制（或段表机制），作为主要的数据结构。
- 中断机构，当用户程序要访问的部分尚未调入内存时，则产生中断。
- 地址变换机构，逻辑地址到物理地址的变换。

#### 3.2.2 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。

在请求分页系统中，只要求将当前需要的一部分页面装入内存，便可以启动作业运行。在作业执行过程中，当所要访问的页面不在内存中时，再通过调页功能将其调入，同时还可通过置换功能将暂时不用的页面换出到外存上，以便腾出内存空间。

为了实现请求分页，系统必须提供一定的硬件支持。除了需要一定容量的内存及外存的计算机系统，还需要有页表机制、缺页中断机构和地址变换机构。

##### 1. 页表机制

请求分页系统的页表机制不同于基本分页系统，请求分页系统在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3.20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
----	------	-------	--------	-------	------

图 3.20 请求分页系统中的页表项

增加的4个字段说明如下：

- 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。
- 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。
- 修改位  $M$ 。标识该页在调入内存后是否被修改过，以确定页面置换时是否写回外存。
- 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。

## 2. 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）。

缺页中断作为中断，同样要经历诸如保护 CPU 环境、分析中断原因、转入缺页中断处理程序、恢复 CPU 环境等几个步骤。但与一般的中断相比，它有以下两个明显的区别：

- 在指令执行期间而非一条指令执行完后产生和处理中断信号，属于内部异常。
- 一条指令在执行期间，可能产生多次缺页中断。

## 3. 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，又增加了某些功能而形成的，如产生和处理缺页中断，及从内存中换出一页的功能等等。

如图 3.21 所示，在进行地址变换时，先检索快表：

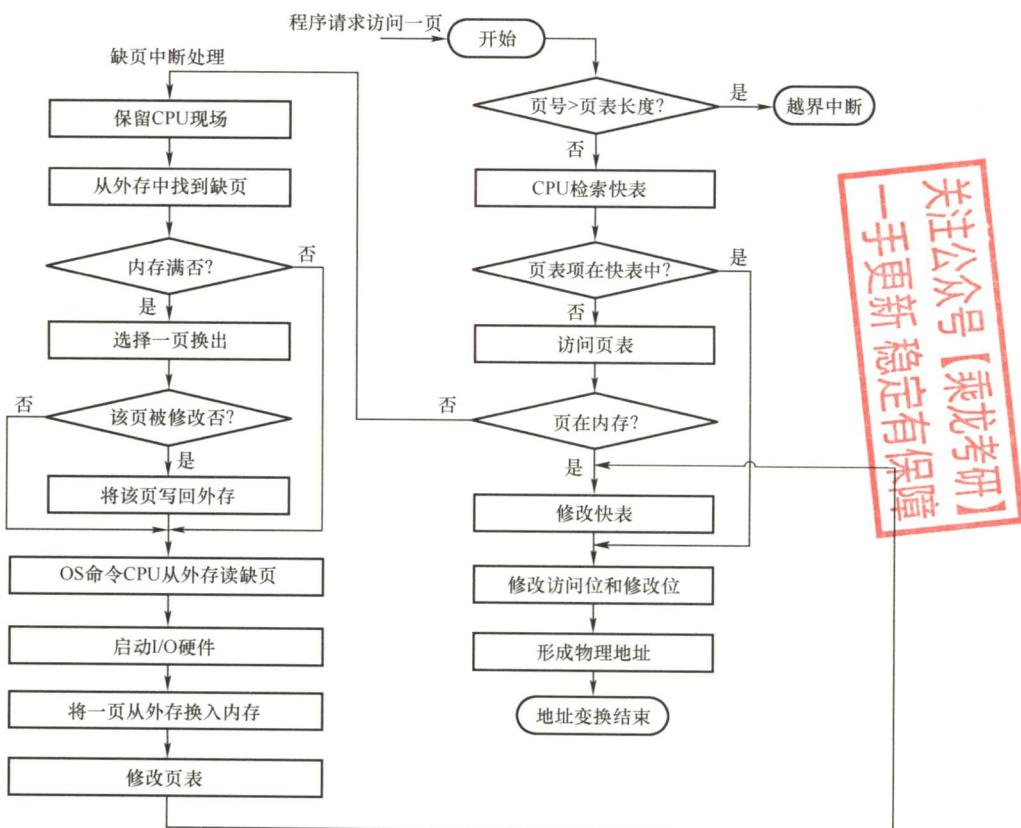


图 3.21 请求分页中的地址变换过程

- 若找到要访问的页，则修改页表项中的访问位（写指令还需要重置修改位），然后利用页表项中给出的物理块号和页内地址形成物理地址。
- 若未找到该页的页表项，则应到内存中去查找页表，再对比页表项中的状态位  $P$ ，看该页是否已调入内存，若页面已调入，则将该页的页表写入快表，若快表已满，则需采用某种算法替换。若页面未调入，则产生缺页中断，请求从外存把该页调入内存。

### 3.2.3 页框分配

#### 1. 驻留集大小

对于分页式的虚拟内存，在进程准备执行时，不需要也不可能把一个进程的所有页都读入主存。因此，操作系统必须决定读取多少页，即决定给特定的进程分配几个页框。给一个进程分配的物理页框的集合就是这个进程的驻留集。需要考虑以下几点：

- 1) 分配给一个进程的页框越少，驻留在主存中的进程就越多，从而可提高 CPU 的利用率。
- 2) 若一个进程在主存中的页面过少，则尽管有局部性原理，缺页率仍相对较高。
- 3) 若分配的页框过多，则由于局部性原理，对该进程的缺页率没有太明显的影响。

#### 2. 内存分配策略

在请求分页系统中，可采取两种内存分配策略，即固定和可变分配策略。在进行置换时，也可采取两种策略，即全局置换和局部置换。于是可组合出下面三种适用的策略。

##### (1) 固定分配局部置换

为每个进程分配一定数目的物理块，在进程运行期间都不改变。所谓局部置换，是指如果进程在运行中发生缺页，则只能从分配给该进程在内存的页面中选出一页换出，然后再调入一页，以保证分配给该进程的内存空间不变。实现这种策略时，难以确定应为每个进程分配的物理块数目：太少会频繁出现缺页中断，太多又会降低 CPU 和其他资源的利用率。

##### (2) 可变分配全局置换

先为每个进程分配一定数目的物理块，在进程运行期间可根据情况适当地增加或减少。所谓全局置换，是指如果进程在运行中发生缺页，系统从空闲物理块队列中取出一块分配给该进程，并将所缺页调入。这种方法比固定分配局部置换更加灵活，可以动态增加进程的物理块，但也存在弊端，如它会盲目地给进程增加物理块，从而导致系统多道程序的并发能力下降。

##### (3) 可变分配局部置换

为每个进程分配一定数目的物理块，当某进程发生缺页时，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。若进程在运行中频繁地发生缺页中断，则系统再为该进程分配若干物理块，直至该进程的缺页率趋于适当程度；反之，若进程在运行中的缺页率特别低，则可适当减少分配给该进程的物理块，但不能引起其缺页率的明显增加。这种方法在保证进程不会过多地调页的同时，也保持了系统的多道程序并发能力。当然它需要更复杂的实现，也需要更大的开销，但对比频繁地换入/换出所浪费的计算机资源，这种牺牲是值得的。

页面分配策略在 2015 年的统考选择题中出现过，考查的是这三种策略的名称。往年很多读者看到这里时，由于认为不是重点，复习时便一带而过，最后在考试中失分。在这种基础题上失分是十分可惜的。再次提醒读者，考研成功的秘诀在于“反复多次”和“全面”。

### 3. 物理块调入算法

采用固定分配策略时，将系统中的空闲物理块分配给各个进程，可采用下述几种算法。

- 1) 平均分配算法，将系统中所有可供分配的物理块平均分配给各个进程。

- 2) 按比例分配算法，根据进程的大小按比例分配物理块。
- 3) 优先权分配算法，为重要和紧迫的进程分配较多的物理块。通常采取的方法是把所有可分配的物理块分成两部分：一部分按比例分配给各个进程；一部分则根据优先权分配。

#### 4. 调入页面的时机

为确定系统将进程运行时所缺的页面调入内存的时机，可采取以下两种调页策略：

- 1) 预调页策略。根据局部性原理，一次调入若干相邻的页会比一次调入一页更高效。但若调入的一批页面中的大多数都未被访问，则又是低效的。因此，需要采用以预测为基础的预调页策略，将那些预计在不久之后便会被访问的页面预先调入内存。但目前预调页的成功率仅约 50%。因此这种策略主要用于进程的首次调入，由程序员指出应先调入哪些页。
- 2) 请求调页策略。进程在运行中需要访问的页面不在内存，便提出请求，由系统将其所需页面调入内存。由这种策略调入的页一定会被访问，且这种策略比较易于实现，因此目前的虚拟存储器大多采用此策略。其缺点是每次仅调入一页，增加了磁盘 I/O 开销。

预调入实际上就是运行前的调入，请求调页实际上就是运行期间调入。

#### 5. 从何处调入页面

请求分页系统中的外存分为两部分：用于存放文件的文件区和用于存放对换页面的对换区。对换区采用连续分配方式，而文件区采用离散分配方式，因此对换区的磁盘 I/O 速度比文件区的更快。这样，当发生缺页请求时，系统从何处将缺页调入内存就分为三种情况：

- 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。为此，在进程运行前，需将与该进程有关的文件从文件区复制到对换区。
- 2) 系统缺少足够的对换区空间。凡是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，由于它们未被修改而不必再将它们换出。但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，以后需要时再从对换区调入（因为读比写的速度快）。
- 3) UNIX 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。曾经运行过但又被换出的页面，由于是放在对换区，因此在下次调入时应从对换区调入。进程请求的共享页面若被其他进程调入内存，则无须再从对换区调入。

#### 6. 如何调入页面

当进程所访问的页面不在内存中时（存在位为 0），便向 CPU 发出缺页中断，中断响应后便转入缺页中断处理程序。该程序通过查找页表得到该页的物理块，此时如果内存未满，则启动磁盘 I/O，将所缺页调入内存，并修改页表。如果内存已满，则先按某种置换算法从内存中选出一页准备换出；如果该页未被修改过（修改位为 0），则无须将该页写回磁盘；但是，如果该页已被修改（修改位为 1），则必须将该页写回磁盘，然后将所缺页调入内存，并修改页表中的相应表项，置其存在位为 1。调入完成后，进程就可利用修改后的页表形成所要访问数据的内存地址。

#### 3.2.4 页面置换算法

进程运行时，若其访问的页面不在内存中而需将其调入，但内存已无空闲空间时，就需要从内存中调出一页程序或数据，送入磁盘的对换区。

选择调出页面的算法就称为页面置换算法。好的页面置换算法应有较低的页面更换频率，也就是说，应将以后不会再访问或以后较长时间内不会再访问的页面先调出。

常见的置换算法有以下 4 种。

### 1. 最佳 (OPT) 置换算法

最佳置换算法选择的被淘汰页面是以后永不使用的页面，或是在最长时间内不再被访问的页面，以便保证获得最低的缺页率。然而，由于人们目前无法预知进程在内存下的若干页面中哪个是未来最长时间内不再被访问的，因而该算法无法实现。但可利用该算法去评价其他算法。

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

进程运行时，先将 7, 0, 1 三个页面依次装入内存。当进程要访问页面 2 时，产生缺页中断，根据最佳置换算法，选择将第 18 次访问才需调入的页面 7 淘汰。然后，访问页面 0 时，因为它已在内存中，所以不必产生缺页中断。访问页面 3 时，又会根据最佳置换算法将页面 1 淘汰……以此类推，如图 3.22 所示，从图中可以看出采用最佳置换算法时的情况。

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		1					1			
缺页否	√	√	√	√		√		√		√		√		√			√			

图 3.22 利用最佳置换算法时的置换图

可以看到，发生缺页中断的次数为 9，页面置换的次数为 6。

最长时间不被访问和以后被访问次数最小是不同的概念，在理解 OPT 算法时千万不要混淆。

### 2. 先进先出 (FIFO) 页面置换算法

优先淘汰最早进入内存的页面，即淘汰在内存中驻留时间最久的页面。该算法实现简单，只需把已调入内存的页面根据先后次序链接成队列，设置一个指针总是指向最老的页面。但该算法与进程实际运行时的规律不适应，因为在进程中，有的页面经常被访问。

这里仍用上面的例子采用 FIFO 算法进行页面置换。当进程访问页面 2 时，把最早进入内存的页面 7 换出。然后访问页面 3 时，把 2, 0, 1 中最先进入内存的页面 0 换出。由图 3.23 可以看出，利用 FIFO 算法时进行了 12 次页面置换，比最佳置换算法正好多一倍。

FIFO 算法还会产生所分配的物理块数增大而页故障数不减反增的异常现象，称为 Belady 异常。只有 FIFO 算法可能出现 Belady 异常，LRU 和 OPT 算法永远不会出现 Belady 异常。

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	√	√	√	√		√	√	√	√	√	√			√	√			√	√	√

图 3.23 FIFO 置换算法时的置换图

如图 3.24 所示，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 2		2	2	2	3	3	3			1	1	
物理块 3			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 2*		2	2	2			2	3	3	3	3	4
物理块 3*			1	1			1	1	2	2	2	2
物理块 4*				0			0	0	0	1	1	1
缺页否	√	√	√	√			√	√	√	√	√	√

图 3.24 Belady 异常



### 3. 最近最久未使用 (LRU) 置换算法

选择最近最长时间未访问过的页面予以淘汰，它认为过去一段时间内未访问过的页面，在最近的将来可能也不会被访问。该算法为每个页面设置一个访问字段，用来记录页面自上次被访问以来所经历的时间，淘汰页面时选择现有页面中值最大的予以淘汰。

再对上面的例子采用 LRU 算法进行页面置换，如图 3.25 所示。进程第一次对页面 2 访问时，将最近最久未被访问的页面 7 置换出去。然后在访问页面 3 时，将最近最久未使用的页面 1 换出。

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	√	√	√	√		√		√	√	√	√			√		√		√		

图 3.25 LRU 页面置换算法时的置换图

在图 3.25 中，前 5 次置换的情况与最佳置换算法相同，但两种算法并无必然联系。实际上，LRU 算法根据各页以前的使用情况来判断，是“向前看”的，而最佳置换算法则根据各页以后的使用情况来判断，是“向后看”的。而页面过去和未来的走向之间并无必然联系。

LRU 算法的性能较好，但需要寄存器和栈的硬件支持。LRU 是堆栈类的算法。理论上可以证明，堆栈类算法不可能出现 Belady 异常。FIFO 算法基于队列实现，不是堆栈类算法。

### 4. 时钟 (CLOCK) 置换算法

LRU 算法的性能接近 OPT 算法，但实现起来的开销大。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。

#### (1) 简单的 CLOCK 置换算法

为每帧设置一位访问位，当某页首次被装入或被访问时，其访问位被置为 1。对于替换算法，将内存中的所有页面视为一个循环队列，并有一个替换指针与之相关联，当某一页被替换时，该指针被设置指向被替换页面的下一页。在选择一页淘汰时，只需检查页的访问位。若为 0，就选择该页换出；若为 1，则将它置为 0，暂不换出，给予该页第二次驻留内存的机会，再依次顺序检查下一个页面。当检查到队列中的最后一个页面时，若其访问位仍为 1，则返回到队首去循环检查。由于该算法是循环地检查各个页面的使用情况，故称 CLOCK 算法。但是，因为该算法只有一位访问位，而置换时将未使用过的页面换出，故又称最近未用 (NRU) 算法。

假设页面访问顺序为 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 3, 2，采用简单 CLOCK 置换算法，分配 4 个页帧，每个页对应的结构为 (页面号, 访问位)，置换过程如图 3.26 所示。

访问顺序	7	0	1	2	0	3	0	4	2	3	0	3	2	1	3	2
帧 1	7	1	7	1	7	1	7	1	3	1	3	1	3	1	3	1
帧 2		0	1	0	1	0	1	0	0	0	1	0	0	1	0	1
帧 3			1	1	1	1	1	1	0	1	0	4	1	4	1	4
帧 4				2	1	2	1	2	0	2	0	2	1	2	1	2
缺页否	√	√	√	√		√		√						√		√

图 3.26 CLOCK 算法时的置换图

首次访问 7, 0, 1, 2 时, 产生缺页中断, 依次调入主存, 访问位都置为 1。访问 0 时, 已存在, 访问位置为 1。访问 3 时, 产生第 5 次缺页中断, 替换指针初始指向帧 1, 此时所有帧的访问位均为 1, 则替换指针完整地扫描一周, 把所有帧的访问位都置为 0, 然后回到最初的位置(帧 1), 替换帧 1 中的页(包括置换页面和置访问位为 1), 如图 3.27(a)所示。访问 0 时, 已存在, 访问位置为 1。访问 4 时, 产生第 6 次缺页中断, 替换指针指向帧 2(上次替换位置的下一帧), 帧 2 的访问位为 1, 将其修改为 0, 继续扫描, 帧 3 的访问位为 0, 替换帧 3 中的页, 如图 3.27(b)所示。然后依次访问 2, 3, 0, 3, 2, 均已存在, 每次访问都将其访问位置为 1。访问 1 时, 产生缺页中断, 替换指针指向帧 4, 此时所有帧的访问位均为 1, 又完整扫描一周并置访问位为 0, 回到帧 4, 替换之。访问 3 时, 已存在, 访问位置为 1。访问 2 时, 产生缺页中断, 替换指针指向帧 1, 帧 1 的访问位为 1, 将其修改为 0, 继续扫描, 帧 2 的访问位为 0, 替换帧 2 中的页。

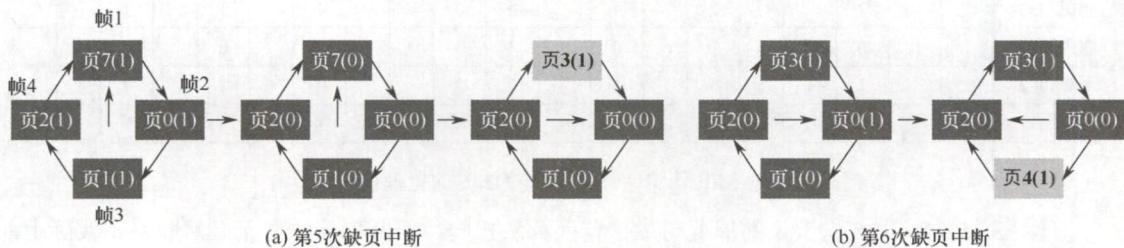


图 3.27 缺页中断时替换指针扫描示意图

## (2) 改进型 CLOCK 置换算法

将一个页面换出时, 若该页已被修改过, 则要将该页写回磁盘, 若该页未被修改过, 则不必将它写回磁盘。可见, 对于修改过的页面, 替换代价更大。在改进型 CLOCK 算法中, 除考虑页面使用情况外, 还增加了置换代价——修改位。在选择页面换出时, 优先考虑既未使用过又未修改过的页面。由访问位  $A$  和修改位  $M$  可以组合成下面四种类型的页面:

- 1 类  $A = 0, M = 0$ : 最近未被访问且未被修改, 是最佳淘汰页。
- 2 类  $A = 0, M = 1$ : 最近未被访问, 但已被修改, 不是很好的淘汰页。
- 3 类  $A = 1, M = 0$ : 最近已被访问, 但未被修改, 可能再被访问。
- 4 类  $A = 1, M = 1$ : 最近已被访问且已被修改, 可能再被访问。

内存中的每页必定都是这四类页面之一。在进行页面置换时, 可采用与简单 CLOCK 算法类似的算法, 差别在于该算法要同时检查访问位和修改位。算法执行过程如下:

- 1) 从指针的当前位置开始, 扫描循环队列, 寻找  $A = 0$  且  $M = 0$  的 1 类页面, 将遇到的第一个 1 类页面作为选中的淘汰页。在第一次扫描期间不改变访问位  $A$ 。
- 2) 若第 1) 步失败, 则进行第二轮扫描, 寻找  $A = 0$  且  $M = 1$  的 2 类页面。将遇到的第一个 2 类页面作为淘汰页。在第二轮扫描期间, 将所有扫描过的页面的访问位都置 0。

- 3) 若第2)步也失败，则将指针返回到开始的位置，并将所有帧的访问位复0。重复第1)步，并且若有必要，重复第2)步，此时一定能找到被淘汰的页。

改进型CLOCK算法优于简单CLOCK算法的地方在于，可减少磁盘的I/O操作次数。但为了找到一个可置换的页，可能要经过几轮扫描，即实现算法本身的开销将有所增加。

操作系统中的页面置换算法都有一个原则，即尽可能保留访问过的页面，而淘汰未访问的页面。简单的CLOCK算法只考虑页面是否被访问过；改进型CLOCK算法对这两类页面做了细分，分为修改过的页面和未修改的页面。因此，若有未使用过的页面，则当然优先将其中未修改过的页面换出。若全部页面都用过，还是优先将其中未修改过的页面换出。

### 3.2.5 抖动和工作集

#### 1. 抖动

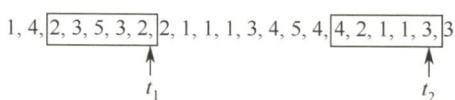
在页面置换过程中，一种最糟糕的情形是，刚刚换出的页面马上又要换入主存，刚刚换入的页面马上又要换出主存，这种频繁的页面调度行为称为抖动或颠簸。

系统发生抖动的根本原因是，系统中同时运行的进程太多，由此分配给每个进程的物理块太少，不能满足进程正常运行的基本要求，致使每个进程在运行时频繁地出现缺页，必须请求系统将所缺页面调入内存。这会使得在系统中排队等待页面调入/调出的进程数目增加。显然，对磁盘的有效访问时间也随之急剧增加，造成每个进程的大部分时间都用于页面的换入/换出，而几乎不能再去做任何有效的工作，进而导致发生处理机的利用率急剧下降并趋于零的情况。

抖动是进程运行时出现的严重问题，必须采取相应的措施解决它。由于抖动的发生与系统为进程分配物理块的多少有关，于是又提出了关于进程工作集的概念。

#### 2. 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。基于局部性原理，可以用最近访问过的页面来确定工作集。一般来说，工作集 $W$ 可由时间 $t$ 和工作集窗口大小 $\Delta$ 来确定。例如，某进程对页面的访问次序如下：



假设系统为该进程设定的工作集窗口大小 $\Delta$ 为5，则在 $t_1$ 时刻，进程的工作集为{2, 3, 5}，在 $t_2$ 时刻，进程的工作集为{1, 2, 3, 4}。

实际应用中，工作集窗口会设置得很大，即对于局部性好的程序，工作集大小一般会比工作集窗口 $\Delta$ 小很多。工作集反映了进程在接下来的一段时间内很有可能会频繁访问的页面集合，因此，若分配给进程的物理块小于工作集大小，则该进程就很有可能频繁缺页，所以为了防止这种抖动现象，一般来说分配给进程的物理块数（即驻留集大小）要大于工作集大小。

工作集模型的原理是，让操作系统跟踪每个进程的工作集，并为进程分配大于其工作集的物理块。落在工作集内的页面需要调入驻留集中，而落在工作集外的页面可从驻留集中换出。若还有空闲物理块，则可再调一个进程到内存。若所有进程的工作集之和超过了可用物理块总数，则操作系统会暂停一个进程，将其页面调出并将物理块分配给其他进程，防止出现抖动现象。

关注公众号【乘龙考研】  
一手更新 稳定有保障

### 3.2.6 内存映射文件

内存映射文件（Memory-Mapped Files）与虚拟内存有些相似，将磁盘文件的全部或部分内容

与进程虚拟地址空间的某个区域建立映射关系，便可以直接访问被映射的文件，而不必执行文件 I/O 操作，也无须对文件内容进行缓存处理。这种特性非常适合用来管理大尺寸文件。

使用内存映射文件所进行的任何实际交互都是在内存中进行的，并且是以标准的内存地址形式来访问的。磁盘的周期性分页是由操作系统在后台隐蔽实现的，对应用程序而言是完全透明的。系统内存中的所有页面都由虚拟存储器负责管理，虚拟存储器以统一的方式处理所有磁盘 I/O。当进程退出或显式地解除文件映射时，所有被改动的页面会被写回磁盘文件。

多个进程允许并发地内存映射同一文件，以便允许数据共享。实际上，很多时候，共享内存是通过内存映射来实现的。进程可以通过共享内存来通信，而共享内存是通过映射相同文件到通信进程的虚拟地址空间实现的。内存映射文件充当通信进程之间的共享内存区域，如图 3.28 所示。一个进程在共享内存上完成了写操作，此刻当另一个进程在映射到这个文件的虚拟地址空间上执行读操作时，就能立刻看到上一个进程写操作的结果。

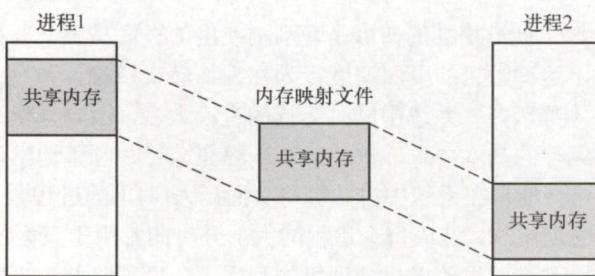


图 3.28 采用内存映射 I/O 的共享内存

### 3.2.7 虚拟存储器性能影响因素

“抖动和工作集”一节原本是 2017 年统考大纲中已删除的考点，但是 2022 年的统考大纲中新增了考点“虚拟存储器性能影响因素及改进方式”。缺页率（缺页率高即为抖动）是影响虚拟存储器性能的主要因素，且缺页率又受到页面大小、分配给进程的物理块数（取决于工作集）、页面置换算法以及程序的编制方法的影响。因此，该考点也与本节的内容相关。

根据局部性原理，页面较大则缺页率较低，页面较小则缺页率较高。页面较小时，一方面减少了内存碎片，有利于提高内存利用率；另一方面，也会使每个进程要求较多的页面，导致页表过长，占用大量内存。页面较大时，虽然可以减少页表长度，但会使页内碎片增大。

分配给进程的物理块数越多，缺页率就越低，但是当物理块超过某个数目时，再为进程增加一个物理块对缺页率的改善是不明显的。可见，此时已没有必要再为它分配更多的物理块，否则也只能是浪费内存空间。只要保证活跃页面在内存中，保持缺页率在一个很低的范围即可。

好的页面置换算法可使进程在运行过程中具有较低的缺页率。选择 LRU、CLOCK 等置换算法，将未来有可能访问的页面尽量保留在内存中，从而提高页面的访问速度。

写回磁盘（见《计算机组成原理考研复习指导》）的频率。换出已修改过的页面时，应当写回磁盘，如果每当一个页面被换出时就将它写回磁盘，那么每换出一个页面就需要启动一次磁盘，效率极低。为此在系统中建立一个已修改换出页面的链表，对每个要被换出的页面（已修改），可以暂不将它们写回磁盘，而将它们挂在该链表上，仅当被换出页面数达到给定值时，才将它们一起写回磁盘，这样就可显著减少磁盘 I/O 的次数，即减少已修改页面换出的开销。此外，如果有进程在这批数据还未写回磁盘时需要再次访问这些页面，就不需从外存调入，而直接从已修改换出页面链表上获取，这样也可以减少页面从磁盘读入内存的频率，减少页面换进的开销。

编写程序的局部化程度越高，执行时的缺页率就越低。如果存储采用的是按行存储，访问时就要尽量采用相同的访问方式，避免按列访问造成缺页率过高的现象。

### 3.2.8 地址翻译

考虑到统考真题越来越喜欢考查学科综合的趋势，这里结合“计算机组成原理”的Cache部分，来说明虚实地址的变换过程。对于不参加统考的读者，可视情况跳过本节；对于参加统考却尚未复习计算机组成原理的读者，可在复习之后，再回过头来学习本节内容。

设某系统满足以下条件：

- 有一个 TLB 与一个 data Cache
- 存储器以字节为编址单位
- 虚拟地址 14 位
- 物理地址 12 位
- 页面大小为 64B
- TLB 为四路组相联，共有 16 个条目
- data Cache 是物理寻址、直接映射的，行大小为 4B，共有 16 组

写出访问地址为 0x03d4, 0x00f1 和 0x0229 的过程。

因为本系统以字节编址，页面大小为 64B，则页内偏移地址为  $\log_2(64B/1B) = 6$  位，所以虚拟页号为  $14 - 6 = 8$  位，物理页号为  $12 - 6 = 6$  位。因为 TLB 为四路组相联，共有 16 个条目，则 TLB 共有  $16/4 = 4$  组，因此虚拟页号中低  $\log_2 4 = 2$  位就为组索引，高 6 位就为 TLB 标记。又因为 Cache 行大小为 4B，因此物理地址中低  $\log_2 4 = 2$  位为块偏移，Cache 共有 16 组，可知接下来  $\log_2 16 = 4$  位为组索引，剩下高 6 位作为标记。地址结构如图 3.29 所示。

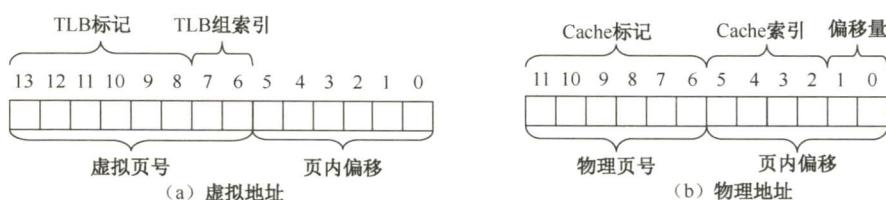


图 3.29 地址结构

TLB、页表、data Cache 内容如表 3.1、表 3.2 及表 3.3 所示。

表 3.1 TLB

索引	标记位	物理页号	有效位	标记位	物理页号	有效位
0	03	-	0	09	0D	1
	00	-	0	07	02	1
1	03	2D	1	02	-	0
	04	-	0	0A	-	0
2	02	-	0	08	-	0
	06	-	0	03	-	0
3	07	-	0	03	0D	1
	0A	34	1	02	-	0

表 3.2 部分页表

虚拟页号	物理页号	有效位	虚拟页号	物理页号	有效位
00	28	1	08	-	0
01	-	0	09	17	1
02	33	1	0A	09	1
03	02	1	0B	-	0
04	-	0	0C	-	0
05	16	1	0D	2D	1
06	-	0	0E	11	1
07	-	0	0F	0D	1

表 3.3 data Cache 内容

索引	标记位	有效位	块 0	块 1	块 2	块 3	索引	标记位	有效位	块 0	块 1	块 2	块 3
0	19	1	99	11	23	11	8	24	1	3A	00	51	89
1	15	0	-	-	-	-	9	2D	0	-	-	-	-
2	1B	1	00	02	04	08	A	2D	1	93	15	DA	3B
3	36	0	-	-	-	-	B	0B	0	-	-	-	-
4	32	1	43	6D	8F	09	C	02	0	-	-	-	-
5	0D	1	36	72	F0	1D	D	16	1	04	96	34	15
6	31	0	-	-	-	-	E	13	1	83	77	1B	D3
7	16	1	11	C2	DF	03	F	14	0	-	-	-	-

先把十六进制的虚拟地址 0x03d4, 0x00f1 和 0x0229 转化为二进制形式，如表 3.4 所示。

表 3.4 虚拟地址结构

虚拟地址	TLB 标记							组索引							
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x03d4	0	0	0	0	1	1	1	1	0	1	0	1	0	0	0
0x00f1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1
0x0229	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1
	虚拟页号							页内偏移							

得到每个地址的组索引和 TLB 标记，接下来就要找出每个地址的页面在不在主存中，若在主存中，则还要找出物理地址。

对于 0x03d4，组索引为 3，TLB 标记为 0x03，查 TLB，第 3 组中正好有标记为 03 的项，有效位为 1，可知页面在主存中，对应的物理页号为 0d (001101)，再拼接页内地址 010100，可得物理地址为 0x354 (001101010100)。

对于 0x00f1，组索引为 3，TLB 标记为 0x00，查 TLB，第 3 组中没有标记为 00 的项，再去找页表，虚拟页号为 0x03，页表第 3 行的有效位为 1，可知页面在主存中，物理页号为 02(000010)，再拼接页内地址 110001，可得物理地址为 0x0b1 (000010110001)。

对于 0x0229，组索引为 0，TLB 标记为 0x02，查 TLB，第 0 组中没有标记为 02 的项，再去找页表，虚拟页号为 0x08，页表第 8 行的有效位为 0，页面不在主存中，产生缺页中断。

找出在主存中的页面的物理地址后，就要通过物理地址访问数据，接下来要找该物理地址的内容在不在 Cache 中，物理地址结构如表 3.5 所示。

表 3.5 物理地址结构

物理地址	Cache 标记						Cache 索引				偏移	
	11	10	9	8	7	6	5	4	3	2	1	0
0x354	0	0	1	1	0	1	0	1	0	1	0	0
0x0b1	0	0	0	0	1	0	1	1	0	0	0	1
	物理页号						页内偏移					

对于 0x354，Cache 索引为 5，Cache 标记为 0x0d，对照 Cache 中索引为 5 的行，标记正好为

0d，有效位为 1，可知该块在 Cache 中，偏移 0，即块 0，可得虚拟地址 0x03d4 的内容为 36H。

对于 0x0b1，Cache 索引为 c，Cache 标记为 0x02，对照 Cache 中索引为 c 的行，有效位为 0，可知该块不在 Cache 中，要去主存中查找物理页号为 2、偏移为 0x31 的内容。

以上例子基本覆盖了从虚拟地址到 Cache 查找内容的所有可能出现的情况，读者务必要掌握此节的内容，查找顺序是从 TLB 到页表（TLB 不命中），再到 Cache 和主存，最后到外存。

### 3.2.9 本节小结

本节开头提出的问题的参考答案如下。

1) 为什么要引入虚拟内存？

上一节提到过，多道程序并发执行不仅使进程之间共享了处理器，而且同时共享了主存。然而，随着对处理器需求的增长，进程的执行速度会以某种合理平滑的方式慢下来。但是，若同时运行的进程太多，则需要很多的内存，当一个程序没有内存空间可用时，那么它甚至无法运行。所以，在物理上扩展内存相对有限的条件下，应尝试以一些其他可行的方式在逻辑上扩充内存。

2) 虚拟内存（虚存）空间的大小由什么因素决定？

虚存的容量要满足以下两个条件：

- ① 虚存的实际容量≤内存容量和外存容量之和，这是硬件的硬性条件规定的，若虚存的实际容量超过了这个容量，则没有相应的空间来供虚存使用。
- ② 虚存的最大容量≤计算机的地址位数能容纳的最大容量。假设地址是 32 位的，按字节编址，一个地址代表 1B 存储空间，则虚存的最大容量≤4GB ( $2^{32}B$ )。这是因为若虚存的最大容量超过 4GB，则 32 位的地址将无法访问全部虚存，也就是说 4GB 以后的空间被浪费了，相当于没有一样，没有任何意义。

实际虚存的容量是取条件①和②的交集，即两个条件都要满足，仅满足一个条件是不行的。

3) 虚拟内存是怎么解决问题的？会带来什么问题？

虚拟内存使用外存上的空间来扩充内存空间，通过一定的换入/换出，使得整个系统在逻辑上能够使用一个远远超出其物理内存大小的内存容量。因为虚拟内存技术调换页面时需要访问外存，会导致平均访存时间增加，若使用了不合适的替换算法，则会大大降低系统性能。

本节学习了 4 种页面置换算法，要把它们与处理机调度算法区分开。当然，这些调度算法之间也是有联系的，它们都有一个共同点，即通过一定的准则决定资源的分配对象。在处理机调度算法中这些准则比较多，有优先级、响应比、时间片等，而在页面调度算法中就比较简单，即是否被用到过或近段时间内是否经常使用。在操作系统中，几乎每类资源都会有相关的调度算法，读者通过将这些调度算法作为线索，可把整个操作系统的课程连成一个整体。

### 3.2.10 本节习题精选

#### 一、单项选择题

01. 请求分页存储管理中，若把页面尺寸增大一倍而且可容纳的最大页数不变，则在程序顺序执行时缺页中断次数会（ ）。
 

A. 增加	B. 减少
C. 不变	D. 可能增加也可能减少
02. 进程在执行中发生了缺页中断，经操作系统处理后，应让其执行（ ）指令。
 

A. 被中断的前一条	B. 被中断的那一条
C. 被中断的后一条	D. 启动时的第一条

关注公众号【乘龙考研】  
一手更新 稳定有保障

03. 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
04. 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
05. 为使虚存系统有效地发挥其预期的作用，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
06. ( ) 是请求分页存储管理方式和基本分页存储管理方式的区别。  
A. 地址重定向      B. 不必将作业全部装入内存  
C. 采用快表技术      D. 不必将作业装入连续区域
07. 下面关于请求页式系统的页面调度算法中，说法错误的是( )。  
A. 一个好的页面调度算法应减少和避免抖动现象  
B. FIFO 算法实现简单，选择最先进入主存储器的页面调出  
C. LRU 算法基于局部性原理，首先调出最近一段时间内最长时间未被访问过的页面  
D. CLOCK 算法首先调出一段时间内被访问次数多的页面
08. 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.  $\min(m, n)$
09. 在请求分页存储管理中，若采用 FIFO 页面淘汰算法，则当可供分配的页帧数增加时，缺页中断的次数( )。  
A. 减少      B. 增加  
C. 无影响      D. 可能增加也可能减少
10. 设主存容量为 1MB，外存容量为 400MB，计算机系统的地址寄存器有 32 位，那么虚拟存储器的最大容量是( )。  
A. 1MB      B. 401MB      C.  $1MB + 2^{32}MB$       D.  $2^{32}B$
11. 虚拟存储器的最大容量( )。  
A. 为内外存容量之和      B. 由计算机的地址结构决定  
C. 是任意的      D. 由作业的地址空间决定
12. 某虚拟存储器系统采用页式内存管理，使用 LRU 页面替换算法，考虑页面访问地址序列为 18178272183821317137。假定内存容量为 4 个页面，开始时是空的，则页面失效次数是( )。  
A. 4      B. 5      C. 6      D. 7
13. 导致 LRU 算法实现起来耗费高的原因是( )。  
A. 需要硬件的特殊支持      B. 需要特殊的中断处理程序  
C. 需要在页表中标明特殊的页类型      D. 需要对所有的页进行排序
14. 在虚拟存储器系统的页表项中，决定是否会发生页故障的是( )。  
A. 合法位      B. 修改位      C. 页类型      D. 保护码
15. 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
16. 虚拟存储管理系统的基础是程序的( )理论。  
 A. 动态性      B. 虚拟性      C. 局部性      D. 全局性
17. 请求分页存储管理的主要特点是( )。  
 A. 消除了页内零头      B. 扩充了内存  
 C. 便于动态链接      D. 便于信息共享
18. 在请求分页存储管理的页表中增加了若干项信息，其中修改位和访问位供( )参考。  
 A. 分配页面      B. 调入页面      C. 置换算法      D. 程序访问
19. 产生内存抖动的主要原因是( )。  
 A. 内存空间太小      B. CPU运行速度太慢  
 C. CPU调度算法不合理      D. 页面置换算法不合理
20. 在页面置换算法中，存在Belady现象的算法是( )。  
 A. 最佳页面置换算法(OPT)      B. 先进先出置换算法(FIFO)  
 C. 最近最久未使用算法(LRU)      D. 最近未使用算法(NRU)
21. 页式虚拟存储管理的主要特点是( )。  
 A. 不要求将作业装入主存的连续区域  
 B. 不要求将作业同时全部装入主存的连续区域  
 C. 不要求进行缺页中断处理  
 D. 不要求进行页面置换
22. 提供虚拟存储技术的存储管理方法有( )。  
 A. 动态分区存储管理      B. 页式存储管理  
 C. 请求段式存储管理      D. 存储覆盖技术
23. 在计算机系统中，快表用于( )。  
 A. 存储文件信息      B. 与主存交换信息  
 C. 地址变换      D. 存储通道程序
24. 在虚拟分页存储管理系统中，若进程访问的页面不在主存中，且主存中没有可用的空闲帧时，系统正确的处理顺序为( )。  
 A. 决定淘汰页→页面调出→缺页中断→页面调入  
 B. 决定淘汰页→页面调入→缺页中断→页面调出  
 C. 缺页中断→决定淘汰页→页面调出→页面调入  
 D. 缺页中断→决定淘汰页→页面调入→页面调出
25. 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
26. 下列说法中，正确的是( )。  
 I. 先进先出(FIFO)页面置换算法会产生Belady现象  
 II. 最近最少使用(LRU)页面置换算法会产生Belady现象  
 III. 在进程运行时，若其工作集页面都在虚拟存储器内，则能够使该进程有效地运行，否则会出现频繁的页面调入/调出现象  
 IV. 在进程运行时，若其工作集页面都在主存储器内，则能够使该进程有效地运行，否则会出现频繁的页面调入/调出现象

关注公众号【乘龙考研】  
一手更新 稳定有保障

- A. I、III      B. I、IV      C. II、III      D. II、IV
27. 测得某个采用按需调页策略的计算机系统的部分状态数据为：CPU 利用率为 20%，用于交换空间的磁盘利用率为 97.7%，其他设备的利用率为 5%。由此判断系统出现异常，这种情况下（）能提高系统性能。
- A. 安装一个更快的硬盘      B. 通过扩大硬盘容量增加交换空间  
C. 增加运行进程数      D. 加内存条来增加物理空间容量
28. 假定有一个请求分页存储管理系统，测得系统各相关设备的利用率为：CPU 的利用率为 10%，磁盘交换区的利用率为 99.7%，其他 I/O 设备的利用率为 5%。下面（）措施将可能改进 CPU 的利用率。
- I. 增大内存的容量      II. 增大磁盘交换区的容量  
III. 减少多道程序的度数      IV. 增加多道程序的度数  
V. 使用更快速的磁盘交换区      VI. 使用更快速的 CPU  
A. I、II、III、IV      B. I、III      C. II、III、V      D. II、VI
29. 【2011 统考真题】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
30. 【2011 统考真题】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
31. 【2012 统考真题】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的限制      D. 虚拟存储容量只受内存容量的限制
32. 【2013 统考真题】若用户进程访问内存时产生缺页，则下列选项中，操作系统可能执行的操作是（）。
- I. 处理越界错      II. 置换页      III. 分配内存  
A. 仅 I、II      B. 仅 II、III      C. 仅 I、III      D. I、II 和 III
33. 【2014 统考真题】下列措施中，能加快虚实地址转换的是（）。
- I. 增大快表（TLB）容量      II. 让页表常驻内存  
III. 增大交换区（swap）      A. 仅 I      B. 仅 II      C. 仅 I、II      D. 仅 II、III
34. 【2014 统考真题】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
35. 【2016 统考真题】某系统采用改进型 CLOCK 置换算法，页表项中字段  $A$  为访问位， $M$  为修改位。 $A=0$  表示页最近没有被访问， $A=1$  表示页最近被访问过。 $M=0$  表示页未被修改过， $M=1$  表示页被修改过。按 $(A, M)$ 所有可能的取值，将页分为  $(0, 0), (1, 0), (0, 1)$  和  $(1, 1)$  四类，则该算法淘汰页的次序为（）。
- A.  $(0, 0), (0, 1), (1, 0), (1, 1)$       B.  $(0, 0), (1, 0), (0, 1), (1, 1)$

- C.  $(0, 0), (0, 1), (1, 1), (1, 0)$       D.  $(0, 0), (1, 1), (0, 1), (1, 0)$
36. 【2015 统考真题】在请求分页系统中，页面分配策略与页面置换策略不能组合使用的是（ ）。
- A. 可变分配，全局置换      B. 可变分配，局部置换  
C. 固定分配，全局置换      D. 固定分配，局部置换
37. 【2015 统考真题】系统为某进程分配了 4 个页框，该进程已访问的页号序列为 2, 0, 2, 9, 3, 4, 2, 8, 2, 4, 8, 4, 5。若进程要访问的下一页的页号为 7，依据 LRU 算法，应淘汰页的页号是（ ）。
- A. 2      B. 3      C. 4      D. 8
38. 【2016 统考真题】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}
39. 【2019 统考真题】某系统采用 LRU 页面置换算法和局部置换策略，若系统为进程 P 预分配了 4 个页框，进程 P 访问页号的序列为 0, 1, 2, 7, 0, 5, 3, 5, 0, 2, 7, 6，则进程访问上述页的过程中，产生页置换的总次数是（ ）。
- A. 3      B. 4      C. 5      D. 6
40. 【2020 统考真题】下列因素中，影响请求分页系统有效（平均）访存时间的是（ ）。
- I. 缺页率      II. 磁盘读写时间      III. 内存访问时间  
IV. 执行缺页处理程序的 CPU 时间
- A. 仅 II、III      B. 仅 I、IV      C. 仅 I、III、IV      D. I、II、III 和 IV
41. 【2021 统考真题】某请求分页存储系统的页大小为 4KB，按字节编址。系统给进程 P 分配 2 个固定的页框，并采用改进型 Clock 置换算法，进程 P 页表的部分内容见下表。

页号	页框号	存在位	访问位	修改位
		1: 存在, 0: 不存在	1: 访问, 0: 未访问	1: 修改, 0: 未修改
...	...	...	...	...
2	20 H	0	0	0
3	60 H	1	1	0
4	80 H	1	1	1
...	...	...	...	...

- 若 P 访问虚拟地址为 02A01H 的存储单元，则经地址变换后得到的物理地址是（ ）。
- A. 00A01H      B. 20A01H      C. 60A01H      D. 80A01H
42. 【2021 统考真题】下列选项中，通过系统调用完成的操作是（ ）。
- A. 页置换      B. 进程调度      C. 创建新进程      D. 生成随机整数
43. 【2022 统考真题】某进程访问的页 b 不在内存中，导致产生缺页异常，该缺页异常处理过程中不一定包含的操作是（ ）。
- A. 淘汰内存中的页      B. 建立页号与页框号的对应关系



- C. 将页 b 从外存读入内存                    D. 修改页表中页 b 对应的存在位
44. 【2022 统考真题】下列选项中，不会影响系统缺页率的是（ ）。
- A. 页置换算法                                  B. 工作集的大小  
C. 进程的数量                                  D. 页缓冲队列的长度

## 二、综合应用题

01. 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？
02. 假定某操作系统存储器采用页式存储管理，一个进程在相联存储器中的页表项见表 A，不在相联存储器的页表项见表 B。

表 A 相联存储器中的页表

页号	页帧号
0	f1
1	f2
2	f3
3	f4

表 B 内存中的页表

页号	页帧号
4	f5
5	f6
6	f7
7	f8
8	f9
9	f10

注：只列出不在相联存储器中的页表项。

- 假定该进程长度为 320B，每页 32B。现有逻辑地址（八进制）为 101, 204, 576，若上述逻辑地址能转换成物理地址，说明转换的过程，并指出具体的物理地址；若不能转换，说明其原因。
03. 某分页式虚拟存储系统，用于页面交换的磁盘的平均访问及传输时间是 20ms。页表保存在主存中，访问时间为 1μs，即每引用一次指令或数据，需要访问内存两次。为改善性能，可以增设一个关联寄存器，若页表项在关联寄存器中，则只需访问一次内存。假设 80% 的访问的页表项在关联寄存器中，剩下的 20% 中，10% 的访问（即总数的 2%）会产生缺页。请计算有效访问时间。
04. 在页式虚存管理系统中，假定驻留集为 m 个页帧（初始所有页帧均为空），在长为 p 的引用串中具有 n 个不同页号（n > m），对于 FIFO、LRU 两种页面置换算法，试给出页故障数的上限和下限，说明理由并举例说明。
05. 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) 最近最久未使用算法。
06. 一个页式虚拟存储系统，其并发进程数固定为 4 个。最近测试了它的 CPU 利用率和用于页面交换的磁盘的利用率，得到的结果就是下列 3 组数据中的一组。针对每组数据，说明系统发生了什么事情。增加并发进程数能提升 CPU 的利用率吗？页式虚拟存储系统有用吗？
- 1) CPU 利用率为 13%；磁盘利用率为 97%。

- 2) CPU 利用率为 87%; 磁盘利用率为 3%。  
 3) CPU 利用率为 13%; 磁盘利用率为 3%。
07. 现有一请求页式系统，页表保存在寄存器中。若有一个可用的空页或被置换的页未被修改，则它处理一个缺页中断需要  $8\mu s$ ; 若被置换的页已被修改，则处理一缺页中断因增加写回外存时间而需要  $20\mu s$ ，内存的存取时间为  $1\mu s$ 。假定 70% 被置换的页被修改过，为保证有效存取时间不超过  $2\mu s$ ，可接受的最大缺页中断率是多少？
08. 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，每段最大为 4GB。  
 1) 假设系统使用纯页式存储，则要采用多少级页表？页内偏移多少位？  
 2) 假设系统采用一级页表，TLB 命中率为 98%，TLB 访问时间为 10ns，内存访问时间为 100ns，并假设当 TLB 访问失败时才开始访问内存，问平均页面访问时间是多少？  
 3) 若是二级页表，页面平均访问时间是多少？  
 4) 上题中，若要满足访问时间小于 120ns，则命中率至少需要为多少？  
 5) 若系统采用段页式存储，则每用户最多可以有多少个段？段内采用几级页表？
09. 在一个请求分页系统中，采用 LRU 页面置换算法时，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。
10. 一个进程分配给 4 个页帧，见下表（所有数字均为十进制，均从 0 开始计数）。时间均为从进程开始到该事件之前的时钟值，而不是从事件发生到当前的时钟值。请回答：

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

- 1) 当进程访问虚页 4 时，产生缺页中断，请分别用 FIFO（先进先出）、LRU（最近最少使用）、改进型 CLOCK 算法，决定缺页中断服务程序选择换出的页面。  
 2) 在缺页之前给定上述的存储器状态，考虑虚页访问串 4, 0, 0, 0, 2, 4, 2, 1, 0, 3, 2，如果使用 LRU 页面置换算法，分给 4 个页帧，那么会发生多少缺页？
11. 在页式虚拟管理的页面替换算法中，对于任何给定的驻留集大小，在什么样的访问情况下，FIFO 与 LRU 替换算法一样（即被替换的页面和缺页情况完全一样）？
12. 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。

13. 有一个矩阵 int A[100, 100]以行优先方式进行存储。计算机采用虚拟存储系统，物理内

存共有三页，其中一页用来存放程序，其余两页用于存放数据。假设程序已在内存中占一页，其余两页空闲。若每页可存放 200 个整数，程序 1、程序 2 执行的过程中各会发生多少次缺页？每页只能存放 100 个整数时，会发生多少次缺页？以上结果说明了什么问题？

程序 1：

```
for(i=0; i<100; i++)
    for(j=0; j<100; j++)
        A[i,j]=0;
```

程序 2：

```
for(j=0; j<100; j++)
    for(i=0; i<100; i++)
        A[i,j]=0;
```

14. Gribble 公司正在开发一款 64 位的计算机体系结构，也就是说，在访问内存时，最多可以使用 64 位的地址。假设采用的是虚拟页式存储管理，现在要为这款机器设计相应的地址映射机制。

- 1) 假设页面的大小是 4KB，每个页表项的长度是 4B，而且必须采用三级页表结构，每级页表结构中的每个页表都必须正好存放在一个物理页面中，请问在这种情形下，如何实现地址的映射？具体来说，对于给定的一个虚拟地址，应该把它划分为几部分，每部分的长度分别是多少，功能是什么？另外，采用这种地址映射机制后，可以访问的虚拟地址空间有多大？（提示：64 位地址并不一定全部用上。）
- 2) 假设每个页表项的长度变成了 8B，而且必须采用四级页表结构，每级页表结构中的页表都必须正好存放在一个物理页面中，请问在这种情形下，系统能够支持的最大的页面大小是多少？此时，虚拟地址应该如何划分？

15. 【2009 统考真题】请求分页管理系统中，假设某进程的页表内容如下表所示。

页面大小为 4KB，一次内存的访问时间是 100ns，一次快表（TLB）的访问时间是 10ns，处理一次缺页的平均时间为  $10^8$ ns（已含更新 TLB 和页表的时间），进程的驻留集大小固定为 2，采用最近最少使用（LRU）置换算法和局部淘汰策略。假设：①TLB 初始为空；②地址转换时先访问 TLB，若 TLB 未命中，再访问页表（忽略访问页表后的 TLB 更新时间）；③有效位为 0 表示页面不在内存，产生缺页中断，缺页中断处理后，返回到产生缺页中断的指令处重新执行。设有虚地址访问序列 2362H, 1565H, 25A5H，请问：

页号	页框（Page Frame）号	有效位（存在位）
0	101H	1
1	—	0
2	254H	1

- 1) 依次访问上述三个虚拟地址，各需多少时间？给出计算过程。
- 2) 基于上述访问序列，虚地址 1565H 的物理地址是多少？请说明理由。

16. 【2010 统考真题】设某计算机的逻辑地址空间和物理地址空间均为 64KB，按字节编址。若某进程最多需要 6 页（Page）数据存储空间，页的大小为 1KB，操作系统采用固定分配局部置换策略为此进程分配 4 个页框（Page Frame），见下表。在装入时刻 260 前，该进程的访问情况也见下表（访问位即使用位）。

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

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

- 1) 该逻辑地址对应的页号是多少?
- 2) 若采用先进先出 (FIFO) 置换算法, 则该逻辑地址对应的物理地址是多少? 要求给出计算过程。若采用时钟 (Clock) 置换算法, 则该逻辑地址对应的物理地址是多少? 要求给出计算过程。设搜索下一页的指针沿顺时针方向移动, 且当前指向 2 号页框, 如下图所示。



17. 【2012 统考真题】某请求分页系统的页面置换策略如下: 从 0 时刻开始扫描, 每隔 5 个时间单位扫描一轮驻留集(扫描时间忽略不计)且本轮未被访问过的页框将被系统回收, 并放入空闲页框链尾, 其中内容在下一次分配之前不清空。当发生缺页时, 若该页曾被使用过且还在空闲页链表中, 则重新放回进程的驻留集中; 否则, 从空闲页框链表头部取出一个页框。

忽略其他进程的影响和系统开销。初始时进程驻留集为空。目前系统空闲页的页框号依次为 32, 15, 21, 41。进程 P 依次访问的<虚拟页号, 访问时刻>为<1, 1>, <3, 2>, <0, 4>, <0, 6>, <1, 11>, <0, 13>, <2, 14>。请回答下列问题:

- 1) 当虚拟页为<0, 4>时, 对应的页框号是什么?
- 2) 当虚拟页为<1, 11>时, 对应的页框号是什么? 说明理由。
- 3) 当虚拟页为<2, 14>时, 对应的页框号是什么? 说明理由。
- 4) 这种方法是否适合于时间局部性好的程序? 说明理由。

18. 【2015 统考真题】某计算机系统按字节编址, 采用二级页表的分页存储管理方式, 虚拟地址格式如下所示:

10 位 页目录号	10 位 页表索引	12 位 页内偏移量
--------------	--------------	---------------

请回答下列问题:

- 1) 页和页框的大小各为多少字节? 进程的虚拟地址空间大小为多少页?
- 2) 若页目录项和页表项均占 4B, 则进程的页目录和页表共占多少页? 写出计算过程。
- 3) 若某指令周期内访问的虚拟地址为 0100 0000H 和 0111 2048H, 则进行地址转换时共访问多少个二级页表? 说明理由。

19. 【2017 统考真题】假定 2017 年题 44<sup>②</sup>给出的计算机 M 采用二级分页虚拟存储管理方式, 虚拟地址格式如下:

页目录号 (10 位)	页表索引 (10 位)	页内偏移量 (12 位)
-------------	-------------	--------------

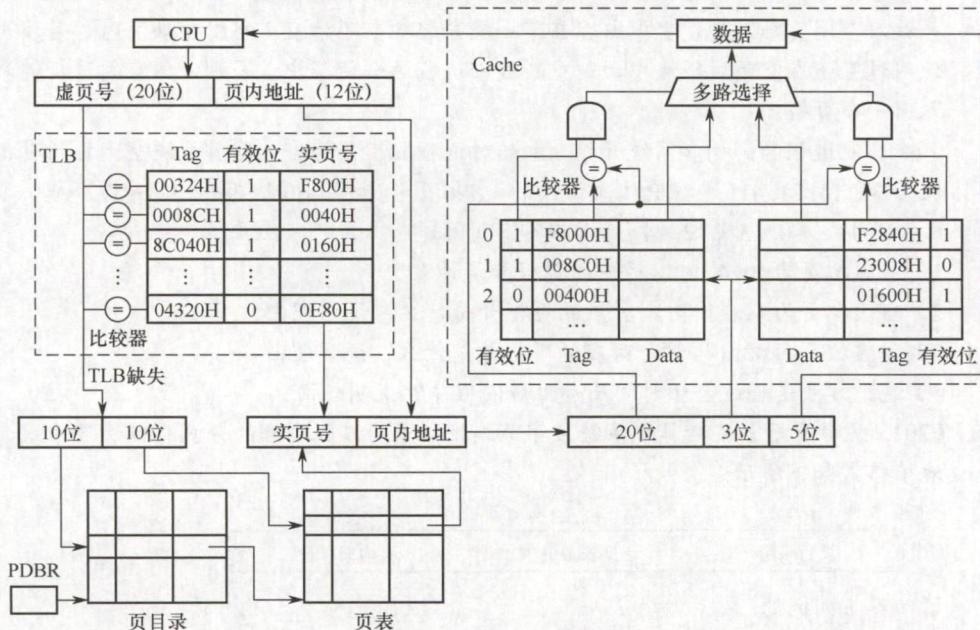
② 本题是 2017 年统考真题, 关联的题干信息太多, 请在王道论坛下载 2017 年的电子版真题。

请针对 2017 年题 43 的函数 f1 和题 44 中的机器指令代码，回答下列问题。

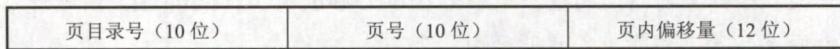
- 1) 函数 f1 的机器指令代码占多少页？
- 2) 取第一条指令（push ebp）时，若在进行地址变换的过程中需要访问内存中的页目录和页表，则会分别访问它们各自的第几个表项（编号从 0 开始）？
- 3) M 的 I/O 采用中断控制方式。若进程 P 在调用 f1 前通过 scanf() 获取 n 的值，则在执行 scanf() 的过程中，进程 P 的状态会如何变化？CPU 是否会进入内核态？

**20. 【2018 统考真题】**某计算机采用页式虚拟存储管理方式，按字节编址，CPU 进行存储访问的过程如下图所示，回答下列问题。

- 1) 某虚拟地址对应的页目录号为 6，在相应的页表中对应的页号为 6，页内偏移量为 8，该虚拟地址的十六进制表示是什么？
- 2) 寄存器 PDBR 用于保存当前进程的页目录起始地址，该地址是物理地址还是虚拟地址？进程切换时，PDBR 的内容是否会变化？说明理由。同一进程的线程切换时，PDBR 的内容是否会变化？说明理由。
- 3) 为了支持改进型 CLOCK 置换算法，需要在页表项中设置哪些字段？



**21. 【2020 统考真题】**某 32 位系统采用基于二级页表的请求分页存储管理方式，按字节编址，页目录项和页表项长度均为 4 字节，虚拟地址结构如下所示。



某 C 程序中数组 a[1024][1024]的起始虚拟地址为 1080 0000H，数组元素占 4 字节，该程序运行时，其进程的页目录起始物理地址为 0020 1000H，请回答下列问题。

- 1) 数组元素 a[1][2]的虚拟地址是什么？对应的页目录号和页号分别是什么？对应的页目录项的物理地址是什么？若该目录项中存放的页框号为 00301H，则 a[1][2]所在页对应的页表项的物理地址是什么？
- 2) 数组 a 在虚拟地址空间中所占的区域是否必须连续？在物理地址空间中所占的区域

是否必须连续？

- 3) 已知数组 a 按行优先方式存放，若对数组 a 分别按行遍历和按列遍历，则哪种遍历方式的局部性更好？

### 3.2.11 答案与解析

#### 一、单项选择题

01. B

对于顺序执行程序，缺页中断的次数等于其访问的页帧数。由于页面尺寸增大，存放程序需要的页帧数就会减少，因此缺页中断的次数也会减少。

02. B

缺页中断是访存指令引起的，说明所要访问的页面不在内存中，进行缺页中断处理并调入所要访问的页后，访存指令显然应该重新执行。

03. B

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。

04. A

多次性、对换性和离散性是虚拟内存的特征，一次性则是传统存储系统的特征。

05. C

虚拟存储技术基于程序的局部性原理。局部性越好，虚拟存储系统越能更好地发挥作用。

06. B

请求分页存储管理方式和基本分页存储管理方式的区别是，前者采用虚拟技术，因此开始运行时，不必将作业全部一次性装入内存，而后者不是。

07. D

CLOCK 算法选择将最近未使用的页面置换出去，因此又称 NRU 算法。

08. C

无论采用什么页面置换算法，每种页面第一次访问时不可能在内存中，必然发生缺页，所以缺页次数大于或等于 n。

09. D

请求分页存储管理中，若采用 FIFO 页面淘汰算法，可能会产生当驻留集增大时页故障数不减反增的 Belady 异常。然而，还有另外一种情况。例如，页面序列为 1, 2, 3, 1, 2, 3，当页帧数为 2 时产生 6 次缺页中断，当页帧数为 3 时产生 3 次缺页中断。所以在请求分页存储管理中，若采用 FIFO 页面淘汰算法，则当可供分配的页帧数增加时，缺页中断的次数可能增加，也可能减少。

10. D

虚拟存储器的最大容量是由计算机的地址结构决定的，与主存容量和外存容量没有必然的联系，其虚拟地址空间为  $2^{32}$ B。

11. B

虽然从实际使用来说，虚拟存储器能使得进程的可用内存扩大到内外存容量之和，但进程的内存寻址仍由计算机的地址结构决定，这就决定了虚拟存储器理论上的最大容量。比如，64 位系统环境下，虚拟内存技术使得进程可用内存空间达  $2^{64}$ B，但外存显然是达不到这个大小的。

关注公众号【乘龙考研】  
一手更新 稳定有保障

**12. C**

利用 LRU 置换算法时的置换如下图所示。

访问页面	1	8	1	7	8	2	7	2	1	8	3	8	2	1	3	1	7	1	3	7
物理块 1	1	1		1	1					1						1				
物理块 2		8		8	8					8						7				
物理块 3				7	7					3						3				
物理块 4					2					2						2				
缺页否	√	√		√	√					√						√				

分别在访问第 1 个、第 2 个、第 4 个、第 6 个、第 11 个、第 17 个页面时产生中断，共产生 6 次中断。

**13. D**

LRU 算法需要对所有页最近一次被访问的时间进行记录，查找时间最久的进行替换，这涉及排序，对置换算法而言，开销太大。为此需要在页表项中增加 LRU 位，选项 A 可视为“耗费高”这一结果，选项 D 才是造成选项 A 的原因。

**14. A**

页表项中的合法位信息显示本页面是否在内存中，即决定了是否会发生页面故障。

**15. D**

抖动是进程的页面置换过程中，频繁的页面调度（缺页中断）行为，所有的页面调度策略都不可能完全避免抖动。

**16. C**

基于局部性原理：在程序装入时，不必将其全部读入内存，而只需将当前需要执行的部分页或段读入内存，就可让程序开始执行。在程序执行过程中，若需执行的指令或访问的数据尚未在内存（称为缺页或缺段）中，则由处理器通知操作系统将相应的页或段调入内存，然后继续执行程序。由于程序具有局部性，虚拟存储管理在扩充逻辑地址空间的同时，对程序执行时内存调换的代价很小。

**17. B**

请求分页存储管理就是为了解决内存容量不足而使用的方法，它基于局部性原理实现了以时间换取空间的目的。它的主要特点自然是间接扩充了内存。

**18. C**

当需要置换页面时，置换算法根据修改位和访问位选择调出内存的页面。

**19. D**

内存抖动是指频繁地引起主存页面淘汰后又立即调入，调入后又很快淘汰的现象。这是由页面置换算法不合理引起的一种现象，是页面置换算法应当尽量避免的。

**20. B**

FIFO 是队列类算法，有 Belady 现象；选项 C、D 均为堆栈类算法，理论上可以证明不会出现 Belady 现象。

**21. B**

页式虚拟存储管理的主要特点是，不要求将作业同时全部装入主存的连续区域，一般只装入 10%~30%。不要求将作业装入主存连续区域是所有离散式存储管理（包括页式存储管理）的特点；页式虚拟存储管理需要进行缺页中断处理和页面置换。

**22. C**

虚拟存储技术是基于页或段从内存的调入/调出实现的，需要有请求机制的支持。

**23. C**

计算机系统中，为了提高系统的存取速度，在地址映射机制中增加一个小容量的硬件部件——快表（又称相联存储器），用来存放当前访问最频繁的少数活动页面的页号。快表查找内存块的物理地址消耗的时间大大降低，使得系统效率得到很大提高。

**24. C**

根据缺页中断的处理流程，产生缺页中断后，首先去内存寻找空闲物理块，若内存没有空闲物理块，则使用页面置换算法决定淘汰页面，然后调出该淘汰页面，最后再调入该进程欲访问的页面。整个流程可归纳为缺页中断→决定淘汰页→页面调出→页面调入。

**25. C**

页面大小为 4KB，因此页内偏移为 12 位。系统采用 48 位虚拟地址，因此虚页号  $48 - 12 = 36$  位。采用多级页表时，最高级页表项不能超出一页大小；每页能容纳的页表项数为  $4KB/8B = 512 = 2^9$ ， $36/9 = 4$ ，因此应采用 4 级页表，最高级页表项正好占据一页空间，所以本题选择选项 C。

**26. B**

FIFO 算法可能产生 Belady 现象，例如页面走向为 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 时，当分配 3 帧时产生 9 次缺页中断，分配 4 帧时产生 10 次缺页中断，I 正确。最近最少使用法不会产生 Belady 现象，II 错误。若页面在内存中，则不会产生缺页中断，即不会出现页面的调入/调出，而不是虚拟存储器（包括作为虚拟内存那部分硬盘），故 III 错误、IV 正确。

**27. D**

用于交换空间的磁盘利用率已达 97.7%，其他设备的利用率为 5%，CPU 的利用率为 20%，说明在任务作业不多的情况下交换操作非常频繁，因此判断物理内存严重短缺。

**28. B**

I 正确：增大内存的容量。增大内存可使每个程序得到更多的页框，能减少缺页率，进而减少换入/换出过程，可提高 CPU 的利用率。II 错误：增大磁盘交换区的容量。因为系统实际已处于频繁的换入/换出过程中，不是因为磁盘交换区容量不够，因此增大磁盘交换区的容量无用。III 正确：减少多道程序的度数。可以提高 CPU 的利用率，因为从给定的条件知道磁盘交换区的利用率为 99.7%，说明系统现在已经处于频繁的换入/换出过程中，可减少主存中的程序。IV 错误：增加多道程序的度数。系统处于频繁的换入/换出过程中，再增加主存中的用户进程数，只能导致系统的换入/换出更频繁，使性能更差。V 错误：使用更快速的磁盘交换区。因为系统现在处于频繁的换入/换出过程中，即使采用更快的磁盘交换区，其换入/换出频率也不会改变，因此没用。VI 错误：使用更快速的 CPU。系统处于频繁的换入/换出过程中，CPU 处于空闲状态，利用率不高，提高 CPU 的速度无济于事。综上分析：I、III 可以改进 CPU 的利用率。

**29. D**

缺页中断调入新页面，肯定要修改页表项和分配页框，所以 I、III 可能发生，同时内存没有页面，需要从外存读入，会发生磁盘 I/O。

**30. A**

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。

### 31. B

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的基础上。有以下三种实现方式：请求分页；请求分段；请求段页式。虚存的实际容量受外存和内存容量之和限制，虚存的最大容量是由计算机的地址位数决定的。

### 32. B

用户进程访问内存时缺页，会发生缺页中断。发生缺页中断时，系统执行的操作可能是置换页面或分配内存。系统内没有越界错误，不会进行越界出错处理。

### 33. C

虚实地址转换是指逻辑地址和物理地址的转换。增大快表容量能把更多的表项装入快表，会加快虚实地址转换的平均速率；让页表常驻内存可以省去一些不在内存中的页表从磁盘上调入的过程，也能加快虚实地址转换；增大交换区对虚实地址转换速度无影响，因此 I、II 正确，选择选项 C。

### 34. A

只有 FIFO 算法会导致 Belady 异常，选择选项 A。

### 35. A

改进型 CLOCK 置换算法执行的步骤如下：

- 1) 从指针的当前位置开始，扫描帧缓冲区。在这次扫描过程中，对使用位不做任何修改。选择遇到的第一个帧 ( $A = 0, M = 0$ ) 用于替换。
- 2) 若第 1) 步失败，则重新扫描，查找 ( $A = 0, M = 1$ ) 的帧。选择遇到的第一个这样的帧用于替换。在这个扫描过程中，对每个跳过的帧，将其使用位设置成 0。
- 3) 若第 2) 步失败，则指针将回到它的最初位置，并且集合中所有帧的使用位均为 0。重复第 1) 步，并在有必要时重复第 2) 步，这样将可以找到供替换的帧。

因此，该算法淘汰页的次序为  $(0, 0), (0, 1), (1, 0), (1, 1)$ ，即选项 A 正确。

### 36. C

对各进程进行固定分配时页面数不变，不可能出现全局置换。而选项 A、B、D 是现代操作系统中常见的 3 种策略。

### 37. A

可以采用书中常规的解法思路，也可以采用便捷法。对页号序列从后往前计数，直到数到 4 (页框数) 个不同的数字为止，这个停止的数字就是要淘汰的页号（最近最久未使用的页），题中为页号 2。

### 38. A

在任意时刻  $t$ ，都存在一个集合，它包含所有最近  $k$  次（该题窗口大小为 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  $\{6, 0, 3, 2\}$ 。

### 39. C

最近最久未使用 (LRU) 算法每次执行页面置换时会换出最近最久未使用过的页面。第一次访问 5 页面时，会把最久未被使用的 1 页面换出，第一次访问 3 页面时，会把最久未访问的 2 页面换出。具体的页面置换情况如下图所示。

关注公众号【乘龙考研】  
一手更新 稳定有保障

访问页面	0	1	2	7	0	5	3	5	0	2	7	6
物理块 1	0	0	0	0	0	0	0	0	0	0	0	0
物理块 2		1	1	1	1	5	5	5	5	5	5	6
物理块 3			2	2	2	2	3	3	3	3	7	7
物理块 4				7	7	7	7	7	7	2	2	2
缺页否	√	√	√	√		√	√			√	√	√

需要注意的是，题中问的是页置换次数，而不是缺页次数，所以前 4 次缺页未换页的情况不考虑在内，答案为 5 次，因此选择选项 C。

#### 40. D

I 影响缺页中断的频率，缺页率越高，平均访存时间越长；II 和 IV 影响缺页中断的处理时间，中断处理时间越长，平均访存时间越长；III 影响访问页表和访问目标物理地址的时间，故 I、II、III 和 IV 均正确。

#### 41. C

页面大小为 4KB，低 12 位是页内偏移。虚拟地址为 02A01H，页号为 02H，02H 页对应的页表项中存在位为 0，进程 P 分配的页框固定为 2，且内存中已有两个页面存在。根据 Clock 算法，选择将 3 号页换出，将 2 号页放入 60H 页框，经过地址变换后得到的物理地址是 60A01H。

#### 42. C

系统调用是由用户进程发起的，请求操作系统的服务。对于选项 A，当内存中的空闲页框不够时，操作系统会将某些页面调出，并将要访问的页面调入，这个过程完全由操作系统完成，不涉及系统调用。对于选项 B，进程调度完全由操作系统完成，无法通过系统调用完成。对于选项 C，创建新进程可以通过系统调用来完成，如 Linux 中通过 fork 系统调用来创建子进程。对于选项 D，生成随机数只需要普通的函数调用，不涉及请求操作系统的服务，如 C 语言中 random() 函数。

#### 43. A

缺页异常需要从磁盘调页到内存中，将新调入的页与页框建立对应关系，并修改该页的存在位，选项 B、C、D 正确；如果内存中有空闲页框，就不需要淘汰其他页，选项 A 错误。

#### 44. D

页置换算法会影响缺页率，例如，LRU 算法的缺页率通常要比 FIFO 算法的缺页率低，排除选项 A。工作集的大小决定了分配给进程的物理块数，分配给进程的物理块数越多，缺页率就越低，排除选项 B。进程的数量越多，对内存资源的竞争越激烈，每个进程被分配的物理块数越少，缺页率也就越高，排除选项 C。页缓冲队列是将被淘汰的页面缓存下来，暂时不写回磁盘，队列长度会影响页面置换的速度，但不会影响缺页率，答案选择选项 D。

## 二、综合应用题

### 01. 【解答】

- 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/调出技术的主要相同点是，都要在内存与外存之间交换信息。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，从而使得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。

## 02. 【解答】

一页的大小为 32B，逻辑地址结构为：低 5 位为页内位移，其余高位为页号。

$101$ （八进制）=  $00100001$ （二进制），则页号为 2，在相联存储器中，对应的页帧号为  $f_3$ ，即物理地址为( $f_3, 1$ )。

$204$ （八进制）=  $010000100$ （二进制），则页号为 4，不在相联存储器中，查内存的页表得页帧号为  $f_5$ ，即物理地址为( $f_5, 4$ )，并用其更新相联存储器中的一项。

$576$ （八进制）=  $101111110$ （二进制），则页号为 11，已超出页表范围，即产生越界中断。

## 03. 【解答】

1) 80%的访问的页表项在关联寄存器中，访问耗时  $1\mu s$ 。

2) 18%的访问的页表项不在关联寄存器中，但在内存中，耗时  $(1 + 1)\mu s$ 。

3) 2%的访问产生缺页中断，访问耗时  $(1\mu s + 1\mu s + 20ms)$ 。

从而有效访问时间为  $80\% \times 1 + 18\% \times 2 + 2\% \times (1 \times 2 + 20 \times 1000) = 401.2\mu s$ 。

**注意：**针对 3) 中的耗时情况，有些读者可能对缺页中断的页面调度过程有些模糊，导致此情况下的访问耗时计算出现偏差。题目中已经明确说明“页表保存在主存”，意味着物理页号一定可以通过查找内存获得并计算出相应的物理地址。即若关联寄存器命中（可以取得页框号，再结合逻辑地址中的偏移量，便可算得对应的物理地址。至此，已经获得物理地址），则仅访问一次内存（根据物理地址取得相应的页面，耗时  $1\mu s$ ）即可；若未命中，则还要从主存中 [\*注：若地址变换是通过查找内存中的页表完成的，则还应将这次所查到的页表项存入关联寄存器中。] 取出相应的页表项（耗时  $1\mu s$ ），再根据得到的物理地址访问内存取得对应的页面（耗时  $1\mu s$ ）。但是，无论关联寄存器是否命中，欲访问的页面 [ 非页表 (Page) ] 不一定在主存中，即若页面不在内存中，则产生缺页中断，操作系统需要启动磁盘将缺页调入内存。题目中明确说明“剩下的 20% 中，10% 的访问（即总数的 2%）会产生缺页”，其中“剩下的 20%”意味着是通过查找内存（关联寄存器未命中）得到物理地址的（耗时  $1\mu s$ ），缺页中断（耗时  $20ms$ ）后，将缺页调入主存，此时“系统恢复缺页中断发生前的状态，将程序指令器重新指向引起缺页中断的指令，重新执行该指令”，这时页表项已在关联寄存器中（对此不解的读者请回看上文的“\*注”处），则根据取得的物理地址仅访存一次即可取得对应的页面（耗时  $1\mu s$ ）。

**注意：**建议读者结合《计算机组成原理考研复习指导》中的“虚拟存储器”小节进行复习。会总结的读者在完成本节习题后应会注意到：该题、第 15 题（2009 真题）中的 1)、第 7 题、第 8 题中的 2) 都是同一类题，望读者在完成本题的基础上总结出求解该类题型的方法，以便在以后遇到类似的题目时得心应手。读者应做到基础扎实、注意细节、触类旁通。

## 04. 【解答】

发生页故障的原因是，当前访问的页不在主存，需要将该页调入主存。此时不管主存中是否已满（已满则先调出一页），都要发生一次页故障，即无论怎样安排， $n$  个不同的页号在首次进入主存时必须要发生一次页故障，总共发生  $n$  次，这是页故障数的下限。虽然不同的页号数为  $n$  小于或等于总长度  $p$ （访问串可能会有一些页重复出现），但驻留集  $m < n$ ，所以可能会有某些页进入主存后又被调出主存，当再次访问时又发生一次页故障的现象，即有些页可能会出现多次页故

障。最差的情况是每访问一个页号时，该页都不在主存中，这样共发生  $p$  次故障。

因此，对于 FIFO、LRU 置换算法，页故障数的上限均为  $p$ ，下限均为  $n$ 。例如，当  $m=3, p=12, n=4$  时，有访问串 1 1 1 2 2 3 3 3 4 4 4 4，则页故障数为 4，这是下限  $n$  的情况。又如，有访问串 1 2 3 4 1 2 3 4 1 2 3 4，则页故障数为 12，这是上限  $p$  的情况。

### 05. 【解答】

- 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。

- 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：

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	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上升，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。

## 06. 【解答】

- 1) 系统出现“抖动”现象。这时若再增加并发进程数，反而会恶化系统性能。页式虚拟存储系统因“抖动”现象而未能充分发挥功用。
- 2) 系统正常。不需要采取什么措施。
- 3) CPU 没有充分利用。应该增加并发进程数。

## 07. 【解答】

在缺页中断处理完成，调入请求页面后，还需  $1\mu s$  的存取访问，即

- 1) 当未缺页时，直接访问内存，用时  $1\mu s$ 。
- 2) 当缺页时，若未修改，则用时  $8\mu s + 1\mu s$ 。
- 3) 当缺页时，而且修改了，则用时  $20\mu s + 1\mu s$ 。

因此，设最大缺页中断率为  $p$ ，有  $(1-p) \times 1\mu s + (1 - 70\%) \times p \times (1\mu s + 8\mu s) + 70\% \times p \times (1\mu s + 20\mu s) = 2\mu s$ ，即  $1\mu s + (1 - 70\%) \times p \times 8\mu s + 70\% \times p \times 20\mu s = 2\mu s$ ，解得  $p \approx 0.061 = 6.1\%$ 。

## 08. 【解答】

- 1) 页面大小为 4KB，因此页内偏移为 12 位。系统采用 48 位虚拟地址，因此虚页号为  $48 - 12 = 36$  位。采用多级页表时，最高级页表项不能超出一页大小；每页能容纳的页表项数为  $4KB/8B = 512 = 2^9$ ， $36/9 = 4$ ，因此应采用 4 级页表，最高级页表项正好占据一页空间。
- 2) 系统进行页面访问操作时，首先读取页面对应的页表项，有 98% 的概率可以在 TLB 中直接读取到，然后进行地址转换，访问内存读取页面；若 TLB 未命中，则要通过一次内存访问来读取页表项。页面平均访问时间为

$$98\% \times (10 + 100) + (1 - 98\%) \times (10 + 100 + 100) = 112ns$$

- 3) 二级页表的平均访问时间计算同理：

$$98\% \times (10 + 100) + (1 - 98\%) \times (10 + 100 + 100 + 100) = 114ns$$

4) 设快表命中率为  $p$ , 则应满足

$$p \times (10 + 100) + (1-p) \times (10 + 100 + 100 + 100) \leq 120\text{ns}$$

解得  $p \geq 95\%$ 。

5) 系统采用 48 位虚拟地址, 每段最大为 4GB, 因此段内地址为 32 位, 段号为  $48 - 32 = 16$  位。每个用户最多可以有  $2^{16}$  段。段内采用页式地址, 与 1) 中计算同理,  $(32 - 12)/9$ , 取上整为 3, 因此段内应采用 3 级页表。

**注意:** 在采用多级页表的页式存储管理中, 若快表命中, 则只需要一次访问内存操作即可存取指令或数据, 这一点需要注意和理解。以本题 1) 中假设的条件为例, 不考虑分段时, 需要 4 级页表。若快表未命中, 则需要从虚拟地址的高位起, 每 9 位逐级访问各级页表, 第 5 次才能访问到指令或数据所在的内存页面。

若快表命中, 则首先考虑快表中的实际内容: 快表存放经常被访问的页面对应的页表项, 页表项中是完整的  $48 - 12 = 36$  位页面号, 所以根据快表可以直接对虚拟地址进行转换。因此多级页表中, 快表命中时同样只需要一次访问内存操作。根本原因在于, 快表提供了进行地址转换的完整的页面号, 而不是某一级的页面号。

#### 09. 【解答】

1) 物理块数为 3 时, 缺页情况见下表:

访问串	1	3	2	1	1	3	5	1	3	2	1	5
内存	1	1	1	1	1	1	1	1	1	1	1	1
	3	3	3	3	3	3	3	3	3	3	3	5
		2	2	2	2	5	5	5	2	2	2	
	缺页	√	√	√			√		√		√	

缺页次数为 6, 缺页率为  $6/12 = 50\%$ 。

2) 物理块数为 4 时, 缺页情况见下表:

访问串	1	3	2	1	1	3	5	1	3	2	1	5
内存	1	1	1	1	1	1	1	1	1	1	1	1
	3	3	3	3	3	3	3	3	3	3	3	3
		2	2	2	2	2	2	2	2	2	2	2
	缺页	√	√	√			√					

缺页次数为 4, 缺页率为  $4/12 = 33\%$ 。

**注意:** 当分配给作业的物理块数为 4 时, 注意到作业请求页面序列中只有 4 个页面, 可以直接得出缺页次数为 4, 而不需要按表列出缺页情况。

#### 10. 【解答】

1) FIFO 算法: 最先进入的页帧号应最先替换, 因此访问虚页 4 发生缺页时, 应置换 3 号页帧中的 3 号虚页, 因为它是最先进入存储器的。

LRU 算法: 应置换 1 号页帧中的 1 号虚页, 因为它是最久未被访问和修改过的, 又是最先进入存储器的。

改进型 CLOCK 算法: 第一轮扫描淘汰访问位和修改位都为 0 的页面, 因此淘汰 1 号页面。



2) 采用 LRU 算法时缺页情况如下表, 缺页次数为 3 次。

页访问串	当前状态	4	0	0	0	2	4	2	1	0	3	2
标记		*							*		*	
M <sub>1</sub>	2	2	2	2	2	2	2	2	2	2	2	2
M <sub>2</sub>	1	4	4	4	4	4	4	4	4	4	3	3
M <sub>3</sub>	0	0	0	0	0	0	0	0	0	0	0	0
M <sub>4</sub>	3	3	3	3	3	3	3	3	1	1	1	1

### 11. 【解答】

由于驻留集大小任意, 现要求两种算法的替换页面和缺页情况完全一样, 就意味着要求 FIFO 与 LRU 的置换选择一致。FIFO 替换最早进入主存的页面, LRU 替换上次访问以来最久未被访问的页面, 这两个页面一致。就是说, 最先进入主存的页面在此次缺页之前不能再被访问, 这样该页面也就同时是最久未被访问的页面。

例如, 合法驻留集大小为 4 时, 对访问串 1, 2, 3, 4, 1, 2, 5, 当 5 号页面调入主存时, 应在 1, 2, 3, 4 页中选择一个替换, FIFO 选择 1, LRU 选择 3。原因在于 1 号页面虽然最先进入主存, 但由于其进入主存后又被再次访问, 所以它不是最久未被访问的页面。若去掉对 1 号页面的第二次访问, 则 FIFO 与 LRU 的替换选择就会相同。同理, 当 5 号页面调入主存后, 若再访问新的 6 号页面, 则 2 号页面会遇到同样的问题。因此, 以此类推, 访问串中的所有页面号都应不同, 但要注意到, 连续访问相同页面时不影响后面的替换选择, 所以对访问串的要求是: 不连续的页面号均不相同。

### 12. 【解答】

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, 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。

4) 改进型 CLOCK 置换算法从上一次的位置开始扫描, 首先寻找未被访问和修改的页面。由表中的 R (读) 标志位和 M (修改) 标志位可知, 只有页面 0 满足  $R = 0$  和  $M = 0$ , 因此改进型 CLOCK 置换算法将选择第 0 页替换。

### 13. 【解答】

程序 1 按行优先的顺序访问数组元素, 与数组在内存中存放的顺序一致, 每个内存页面可存放 200 个数组元素。这样, 程序 1 每访问两行数组元素就产生一次缺页中断, 所以程序 1 的执行过程会发生 50 次缺页。

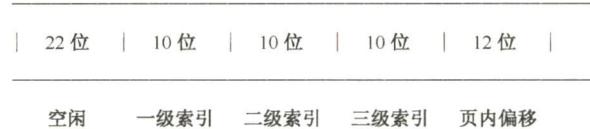
程序 2 按列优先的顺序访问数组元素, 由于每个内存页面存放两行数组元素, 因此程序 2 每访问两个数组元素就产生一次缺页中断, 整个执行过程会发生 5000 次缺页。

若每页只能存放 100 个整数, 则每页仅能存放一行数组元素, 同理可以计算出: 程序 1 的执行过程产生 100 次缺页; 程序 2 的执行过程产生 10000 次缺页。

以上说明缺页的次数与内存中数据存放的方式及程序执行的顺序有很大关系; 同时说明, 当缺页中断次数不多时, 减小页面大小影响并不大, 但缺页中断次数很多时, 减小页面大小会带来很严重的影响。

**14. 【解答】**

- 1) 页面大小为 4KB，每个页表项大小为 4B，因此在每个页表当中，共有 1024 个页表项，对于每个层次的页表来说，都满足这一点，这样每级页表的索引均为 10 位，由于页面大小为 4KB，所以页内偏移地址为 12 位。逻辑地址被划分为 5 个部分：



可访问的虚拟地址空间大小为  $2^{42}B = 4TB$ 。

- 2) 假定一个页面的大小为  $2^Y$ ，即页内偏移地址为  $Y$  位，每个页面可以包含  $2^Y/8 = 2^{(Y-3)}$  个页表项，因此每级页表的索引位为  $Y-3$  位，共有 4 级页表，所以  $4(Y-3) + Y \leq 64$ ， $Y \leq 15.2$ ，因此  $Y = 15$ 。所以最大的页面大小为  $2^{15}B = 32KB$ 。

总结：求解这类题目的关键是清楚地划分逻辑地址，清楚地划分了逻辑地址的每个部分，这类题目就很容易求解。

**15. 【解答】**

- 1) 根据页式管理的工作原理，应先考虑页面大小，以便将页号和页内位移分解出来。页面大小为 4KB，即  $2^{12}$ ，得到页内位移占虚地址的低 12 位，页号占剩余高位。可得三个虚地址的页号 P 如下（十六进制的一位数字转换成二进制的 4 位数字，因此十六进制的低三位正好为页内位移，最高位为页号）：

2362H:  $P = 2$ ，访问快表 10ns，因初始为空，访问页表 100ns 得到页框号，合成物理地址后访问主存 100ns，共计  $10ns + 100ns + 100ns = 210ns$ 。

1565H:  $P = 1$ ，访问快表 10ns，落空，访问页表 100ns 落空，进行缺页中断处理  $10^8ns$ ，访问快表 10ns，合成物理地址后访问主存 100ns，共计  $10ns + 100ns + 10^8ns + 10ns + 100ns = 100000220ns$ 。

25A5H:  $P = 2$ ，访问快表，因第一次访问已将该页号放入快表，因此花费 10ns 便可合成物理地址，访问主存 100ns，共计  $10ns + 100ns = 110ns$ 。

- 2) 当访问虚地址 1565H 时，产生缺页中断，合法驻留集为 2，必须从页表中淘汰一个页面，根据题目的置换算法，应淘汰 0 号页面，因此 1565H 的对应页框号为 101H。由此可得 1565H 的物理地址为 101565H。

**16. 【解答】**

- 1) 由于该计算机的逻辑地址空间和物理地址空间均为  $64KB = 2^{16}B$ ，按字节编址，且页的大小为  $1K = 2^{10}$ ，因此逻辑地址和物理地址的地址格式均为

页号/页框号（6 位）	页内偏移量（10 位）
-------------	-------------

$17CAH = 0001\ 0111\ 1100\ 1010B$ ，可知该逻辑地址的页号为  $000101B = 5$ 。

- 2) 采用 FIFO 置换算法，与最早调入的页面即 0 号页面置换，其所在的页框号为 7，于是对应的物理地址为  $0001\ 1111\ 1100\ 1010B = 1FCAH$ 。
- 3) 采用 CLOCK 置换算法，首先从当前位置（2 号页框）开始顺时针寻找访问位为 0 的页面，当指针指向的页面的访问位为 1 时，就把该访问位清“0”，指针遍历一周后，回到 2 号页框，此时 2 号页框的访问位为 0，置换该页框的页面，于是对应的物理地址为  $0000\ 1011\ 1100\ 1010B = 0BCAH$ 。

**17. 【解答】**

- 1) 页框号为 21。因为起始驻留集为空，而 0 页对应的页框为空闲链表中的第三个空闲页框 (21)，其对应的页框号为 21。
- 2) 页框号为 32。理由：因  $11 > 10$ ，因此发生第三轮扫描，页号为 1 的页框在第二轮已处于空闲页框链表中，此刻该页又被重新访问，因此应被重新放回驻留集中，其页框号为 32。
- 3) 页框号为 41。理由：因为第 2 页从来没有被访问过，它不在驻留集中，因此从空闲页框链表中取出链表头的页框 41，页框号为 41。
- 4) 合适。理由：程序的时间局部性越好，从空闲页框链表中重新取回的机会越大，该策略的优势越明显。

**18. 【解答】**

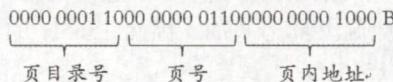
- 1) 页和页框大小均为 4KB。进程的虚拟地址空间大小为  $2^{32}/2^{12} = 2^{20}$  页。
- 2)  $(2^{10} \times 4)/2^{12}$  (页目录所占页数) +  $(2^{20} \times 4)/2^{12}$  (页表所占页数) = 1025 页。
- 3) 需要访问一个二级页表。因为虚拟地址 0100 0000H 和 0111 2048H 的最高 10 位的值都是 4，访问的是同一个二级页表。

**19. 【解答】**

- 1) 函数 f1 的代码段中，所有指令的虚拟地址的高 20 位相同，因此 f1 的机器指令代码在同一页中，仅占用 1 页。页目录号用于寻找页目录的表项，该表项包含页表的位置。页表索引用于寻找页表的表项，该表项包含页的位置。
- 2) push ebp 指令的虚拟地址的最高 10 位（页目录号）为 00 0000 0001，中间 10 位（页表索引）为 00 0000 0001，所以取该指令时访问了页目录的 1 号表项，在对应的页表中访问了 1 号表项。
- 3) 在执行 scanf() 的过程中，进程 P 因等待输入而从执行态变为阻塞态。输入结束时，P 被中断处理程序唤醒，变为就绪态。P 被调度程序调度，变为运行态。CPU 状态会从用户态变为内核态。

**20. 【解答】**

- 1) 由图可知，地址总长度为 32 位，高 20 位为虚页号，低 12 位为页内地址，且虚页号高 10 位为页目录号，低 10 位为页号。展开成二进制表示为



因此十六进制表示为 0180 6008H。

- 2) PDBR 为页目录基址地址寄存器 (Page-Directory Base Register)，其存储页目录表物理内存基地址。进程切换时，PDBR 的内容会变化；同一进程的线程切换时，PDBR 的内容不会变化。每个进程的地址空间、页目录和 PDBR 的内容存在一一对应的关系。进程切换时，地址空间发生了变化，对应的页目录及其起始地址也相应变化，因此需要用进程切换后当前进程的页目录起始地址刷新 PDBR。同一进程中的线程共享该进程的地址空间，其线程发生切换时，地址空间不变，线程使用的页目录不变，因此 PDBR 的内容也不变。
- 3) 改进型 CLOCK 置换算法需要用到使用位和修改位，所以需要设置访问字段（使用位）和修改字段（脏位）。

**21. 【解答】**

- 1)
- ① 页面大小 =  $2^{12}B = 4096B = 4KB$ 。每个数组元素 4B，每个页面可以存放  $4KB/4B = 1024$

个数组元素，正好是数组的一行，数组 a 按行优先方式存放。1080 0000H 的虚页号为 10800H，因此 a[0]行存放在虚页号为 10800H 的页面中，a[1]行存放在页号为 10801H 的页面中。a[1][2]的虚拟地址为  $10801\ 000H + 4 \times 2 = 10801\ 008H$ 。

- ② 转换为二进制 0001000010 0000000001 000000001000，根据虚拟地址结构可知，对应的页目录号为 042H，页号为 001H。
- ③ 进程的页目录表起始地址为 0020 1000H，每个页目录项长 4B，因此 042H 号页目录项的物理地址是  $0020\ 1000H + 4 \times 42H = 0020\ 1108H$ 。
- ④ 页目录项存放的页框号为 00301H，二级页表的起始地址为 00301 000H，因此 a[1][2]所在页的页号为 001H，每个页表项 4B，因此对应的页表项物理地址是  $00301\ 000H + 001H \times 4 = 00301\ 004H$ 。
- 2) 根据数组的随机存取特点，数组 a 在虚拟地址空间中所占的区域必须连续，由于数组 a 不止占用一页，相邻逻辑页在物理上不一定相邻，因此数组 a 在物理地址空间中所占的区域可以不连续。
- 3) 由 1) 可知每个页面正好可以存放一整行的数组元素，“按行优先方式存放”意味着数组的同一行的所有元素都存放在同一个页面中，同一列的各个元素都存放在不同的页面中，因此数组 a 按行遍历的局部性较好。

### 3.3 本章疑难点

分页管理方式和分段管理方式在很多地方是相似的，比如在内存中都是不连续的、都有地址变换机构来进行地址映射等。但两者也存在许多区别，表 3.6 列出了两种方式的对比。

表 3.6 分页管理方式和分段管理方式的比较

	分 页	分 段
目的	分页仅是系统管理上的需要，是为实现离散分配方式，以提高内存的利用率。而不是用户的需要	段是信息的逻辑单位，它含有一组意义相对完整的信息。分段的目的是能更好地满足用户的需要
长度	页的大小固定且由系统决定，由系统把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的	段的长度不固定，决定于用户所编写的程序，通常由编译程序在编译时根据信息的性质来划分
地址空间	分页的程序地址空间是一维的，即单一的线性地址空间，程序员利用一个记号即可表示一个地址	分段的程序地址空间是二维的，程序员在标识一个地址时，既需给出段名，又需给出段内地址
碎片	有内部碎片，无外部碎片	有外部碎片，无内部碎片



# 第 4 章 文件管理

关注公众号【乘龙考研】  
一手更新 稳定有保障

## 【考纲内容】

### (一) 文件

文件的基本概念；文件元数据和索引结点（inode）

文件的操作：建立，删除，打开，关闭，读，写

文件的保护；文件的逻辑结构；文件的物理结构

扫一扫



视频讲解

### (二) 目录

目录的基本概念；树形目录；目录的操作；硬链接和软链接

### (三) 文件系统

文件系统的全局结构（layout）：文件系统在外存中的结构，文件系统在内存中的结构

外存空间管理办法；虚拟文件系统；文件系统挂载（mounting）

## 【复习提示】

本章内容较为具体，要注意对概念的理解。重点掌握文件系统的结构及其实现、文件分配和空闲空间管理等。要掌握文件系统的文件控制块、物理分配方法、索引结构、树形目录结构、文件共享原理、文件系统的布局、虚拟文件系统原理等。这些都是统考真题易考查的内容。

## 4.1 文件系统基础

在学习本节时，请读者思考以下问题：

1) 什么是文件？

2) 单个文件的逻辑结构和物理结构之间是否存在某些制约关系？

本节内容较为抽象，要注意区分文件的逻辑结构和物理结构。在学习过程中，可尝试以上的两个问题为线索，构建整个文件系统的概念。在前面的学习中，曾经提醒过读者不要忽略对基本概念的理解。从历年的情况来看，大部分同学对进程管理、内存管理有较好的掌握，但对于文件管理及后面的 I/O 管理，往往理解不太深入。在考试中，即使面对一些基本问题也容易失分，这十分可惜。主要原因还是对概念的理解不够全面和透彻，希望读者能够关注这个问题。

### 4.1.1 文件的基本概念

文件（File）是以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。大多数应用程序的输入都是通过文件来实现的，其输出也都保存在文件中，以便信息的长期存储及将来的访问。当用户将文件用于程序的输入、输出时，还希望

可以访问、修改和保存文件等，实现对文件的维护管理，这就需要系统提供一个文件管理系统，操作系统中的文件系统（File System）就是用于实现用户的这些管理要求的。

要清晰地理解文件的概念，就要了解文件究竟由哪些东西组成。

首先，文件中肯定包括一块存储空间，更准确地说，是存储空间中的数据；其次，由于操作系统要管理成千上万的数据，因此必定需要对这些数据进行划分，然后贴上“标签”，以便于分类和索引，所以文件必定包含分类和索引的信息；最后，不同的用户拥有对数据的不同访问权限，因此文件中一定包含一些关于访问权限的信息。

再举一个直观的例子“图书馆管理图书”来类比文件。可以认为，计算机中的一个文件相当于图书馆中的一本书，操作系统管理文件，相当于图书管理员管理图书馆中的书。

首先，一本书的主体一定是书中的内容，相当于文件中的数据；其次，不同类别的书需要放在不同的书库，然后加上编号，再把编号登记在图书管理系统中，方便读者查阅，相当于文件的分类和查找；最后，有些已经绝版或价格比较高的外文书籍，只能借给VIP会员或权限比较高的其他读者，而有些普通的书籍可供任何人借阅，这就是文件中的访问权限。所举例子与实际操作系统中的情形并不等价。但对于某些关键的属性，图书馆管理图书和操作系统管理文件的思想却有相一致的地方，因此通过这种类比可使初学者快速认识陌生的概念。

从用户的角度看，文件系统是操作系统的重要部分之一。用户关心的是如何命名、分类和查找文件，如何保证文件数据的安全性及对文件可以进行哪些操作等。而对于其中的细节，如文件如何存储在辅存上、如何管理文件辅存区域等方面关心甚少。

文件系统提供了与二级存储相关的资源的抽象，让用户能在不了解文件的各种属性、文件存储介质的特征及文件在存储介质上的具体位置等情况下，方便快捷地使用文件。用户通过文件系统建立文件，用于应用程序的输入、输出，对资源进行管理。

首先了解文件的结构，我们通过自底向上的方式来定义。

1) 数据项。是文件系统中最低级的数据组织形式，可分为以下两种类型：

- 基本数据项。用于描述一个对象的某种属性的一个值，是数据中的最小逻辑单位。
- 组合数据项。由多个基本数据项组成。

2) 记录。是一组相关的数据项的集合，用于描述一个对象在某方面的属性。

3) 文件。是指由创建者所定义的、具有文件名的一组相关元素的集合，可分为有结构文件和无结构文件两种。在有结构的文件中，文件由若干个相似的记录组成，如一个班的学生记录；而无结构文件则被视为一个字符流，比如一个二进制文件或字符文件。

虽然上面给出了结构化的表述，但实际上关于文件并无严格的规定。在操作系统中，通常将程序和数据组织成文件。文件可以是数字、字符或二进制代码，基本访问单元可以是字节或记录。文件可以长期存储在硬盘中，允许可控制的进程间共享访问，能够被组织成复杂的结构。

## 4.1.2 文件控制块和索引结点

与进程管理一样，为便于文件管理，在操作系统中引入了文件控制块的数据结构。

### 1. 文件的属性

除了文件数据，操作系统还会保存与文件相关的信息，如所有者、创建时间等，这些附加信息称为文件属性或文件元数据。文件属性在不同系统中差别很大，但通常都包括如下属性。

- 1) 名称。文件名称唯一，以容易读取的形式保存。
- 2) 类型。被支持不同类型的文件系统所使用。

- 3) 创建者。文件创建者的 ID。
  - 4) 所有者。文件当前所有者的 ID。
  - 5) 位置。指向设备和设备上文件的指针。
  - 6) 大小。文件当前大小 (用字节、字或块表示), 也可包含文件允许的最大值。
  - 7) 保护。对文件进行保护的访问控制信息。
  - 8) 创建时间、最后一次修改时间和最后一次存取时间。文件创建、上次修改和上次访问的相关信息, 用于保护和跟踪文件的使用。
- 操作系统通过文件控制块 (FCB) 来维护文件元数据。

## 2. 文件控制块

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。图 4.1 是一个典型的 FCB。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, 称为目录项。

文件时间
文件权限 (创建, 访问, 写)
文件所有者, 组, ACL
文件大小
文件数据块

图 4.1 一个典型的 FCB

文件名	索引结点编号
文件名1	
文件名2	
:	
:	

图 4.2 UNIX 的文件目录结构

FCB 主要包含以下信息:

- 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。
- 存取控制信息, 包括文件主的存取权限、核准用户的存取权限以及一般用户的存取权限。
- 使用信息, 如文件建立时间、上次修改时间等。

一个文件目录也被视为一个文件, 称为目录文件。

## 3. 索引结点

文件目录通常存放在磁盘上, 当文件很多时, 文件目录会占用大量的盘块。在查找目录的过程中, 要先将存放目录文件的第一个盘块中的目录调入内存, 然后用给定的文件名逐一比较, 若未找到指定文件, 就还需要不断地将下一盘块中的目录项调入内存, 逐一比较。我们发现, 在检索目录的过程中, 只用到了文件名, 仅当找到一个目录项 (其中的文件名与要查找的文件名匹配) 时, 才需从该目录项中读出该文件的物理地址。也就是说, 在检索目录时, 文件的其他描述信息不会用到, 也不需要调入内存。因此, 有的系统 (如 UNIX, 图 4.2) 便采用了文件名和文件描述信息分开的方法, 使文件描述信息单独形成一个称为索引结点的数据结构, 简称 i 结点 (inode)。在文件目录中的每个目录项仅由文件名和指向该文件所对应的 i 结点的指针构成。

假设一个 FCB 为 64B, 盘块大小是 1KB, 则每个盘块中可以存放 16 个 FCB (FCB 必须连续存放), 若一个文件目录共有 640 个 FCB, 则查找文件平均需要启动磁盘 20 次。而在 UNIX 系统中, 一个目录项仅占 16B, 其中 14B 是文件名, 2B 是 i 结点指针。在 1KB 的盘块中可存放 64 个目录项。这样, 可使查找文件的平均启动磁盘次数减少到原来的 1/4, 大大节省了系统开销。

### (1) 磁盘索引结点

它是指存放在磁盘上的索引结点。每个文件有一个唯一的磁盘索引结点, 主要包括以下内容:

- 文件主标识符, 拥有该文件的个人或小组的标识符。

- 文件类型，包括普通文件、目录文件或特别文件。
- 文件存取权限，各类用户对该文件的存取权限。
- 文件物理地址，每个索引结点中含有 13 个地址项，即  $iaddr(0) \sim iaddr(12)$ ，它们以直接或间接方式给出数据文件所在盘块的编号。
- 文件长度，指以字节为单位的文件长度。
- 文件链接计数，在本文件系统中所有指向该文件的文件名的指针计数。
- 文件存取时间，本文件最近被进程存取的时间、最近被修改的时间及索引结点最近被修改的时间。

### (2) 内存索引结点

它是指存放在内存中的索引结点。当文件被打开时，要将磁盘索引结点复制到内存的索引结点中，便于以后使用。在内存索引结点中增加了以下内容：

- 索引结点编号，用于标识内存索引结点。
- 状态，指示  $i$  结点是否上锁或被修改。
- 访问计数，每当有一进程要访问此  $i$  结点时，计数加 1；访问结束减 1。
- 逻辑设备号，文件所属文件系统的逻辑设备号。
- 链接指针，设置分别指向空闲链表和散列队列的指针。

FCB 或索引结点相当于图书馆中图书的索书号，我们可以在图书馆网站上找到图书的索书号，然后根据索书号找到想要的书本。



## 4.1.3 文件的操作

### 1. 文件的基本操作

文件属于抽象数据类型。为了正确地定义文件，需要考虑可以对文件执行的操作。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：一是为新文件分配必要的外存空间；二是在目录中为之创建一个目录项，目录项记录了新文件名、在外存中的地址及其他可能的信息。
- 2) 写文件。为了写文件，执行一个系统调用。对于给定文件名，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。每当发生写操作时，便更新写指针。
- 3) 读文件。为了读文件，执行一个系统调用。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) 重新定位文件，也称文件定位。搜索目录以找到适当的条目，并将当前文件位置指针重新定位到给定值。重新定位文件不涉及读、写文件。
- 5) 删除文件。为了删除文件，先从目录中检索指定文件名的目录项，然后释放该文件所占的存储空间，以便可被其他文件重复使用，并删除目录条目。
- 6) 截断文件。允许文件所有属性不变，并删除文件内容，将其长度置为 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。

### 2. 文件的打开与关闭

当用户对一个文件实施操作时，每次都要从检索目录开始。为了避免多次重复地检索目录，大多数操作系统要求，在文件使用之前通过系统调用 `open` 被显式地打开。操作系统维护一个包含

所有打开文件信息的表（打开文件表）。所谓“打开”，是指调用 `open` 根据文件名搜索目录，将指明文件的属性（包括该文件在外存上的物理位置），从外存复制到内存打开文件表的一个表目中，并将该表目的编号（也称索引）返回给用户。当用户再次向系统发出文件操作请求时，可通过索引在打开文件表中查到文件信息，从而节省再次搜索目录的开销。当文件不再使用时，可利用系统调用 `close` 关闭它，操作系统将会从打开文件表中删除这一条目。

在多个不同进程可以同时打开文件的操作系统中，通常采用两级表：整个系统表和每个进程表。整个系统的打开文件表包含 FCB 的副本及其他信息。每个进程的打开文件表根据它打开的所有文件，包含指向系统表中适当条目的指针。一旦有进程打开了一个文件，系统表就包含该文件的条目。当另一个进程执行调用 `open` 时，只不过是在其文件打开表中增加一个条目，并指向系统表的相应条目。通常，系统打开文件表为每个文件关联一个 打开计数器(Open Count)，以记录多少进程打开了该文件。每个关闭操作 `close` 使 count 递减，当打开计数器为 0 时，表示该文件不再被使用，并且可从系统打开文件表中删除相应条目。图 4.3 展示了这种结构。

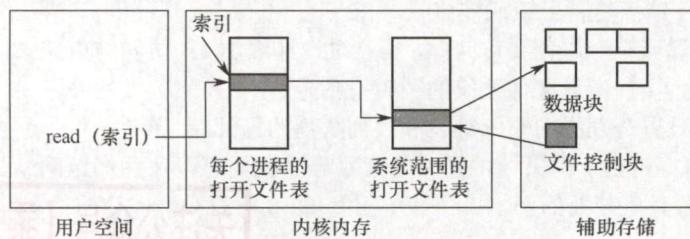


图 4.3 内存中文件的系统结构

文件名不必是打开文件表的一部分，因为一旦完成对 FCB 在磁盘上的定位，系统就不再使用文件名。对于访问打开文件表的索引，UNIX 称之为文件描述符，而 Windows 称之为文件句柄。因此，只要文件未被关闭，所有文件操作就通过打开文件表来进行。

每个打开文件都具有如下关联信息：

- 文件指针。系统跟踪上次的读写位置作为当前文件位置的指针，这种指针对打开文件的某个进程来说是唯一的，因此必须与磁盘文件属性分开保存。
- 文件打开计数。计数器跟踪当前文件打开和关闭的数量。因为多个进程可能打开同一个文件，所以系统在删除打开文件条目之前，必须等待最后一个进程关闭文件。
- 文件磁盘位置。大多数文件操作要求系统修改文件数据。查找磁盘上的文件所需的信息保存在内存中，以便系统不必为每个操作都从磁盘上读取该信息。
- 访问权限。每个进程打开文件都需要有一个访问模式（创建、只读、读写、添加等）。该信息保存在进程的打开文件表中，以便操作系统能够允许或拒绝后续的 I/O 请求。

#### 4.1.4 文件保护

为了防止文件共享可能会导致文件被破坏或未经核准的用户修改文件，文件系统必须控制用户对文件的存取，即解决对文件的读、写、执行的许可问题。为此，必须在文件系统中建立相应的文件保护机制。文件保护通过口令保护、加密保护和访问控制等方式实现。其中，口令和加密是为了防止用户文件被他人存取或窃取，而访问控制则用于控制用户对文件的访问方式。

##### 1. 访问类型

对文件的保护可从限制对文件的访问类型中出发。可加以控制的访问类型主要有以下几种。

- 读。从文件中读。
- 写。向文件中写。
- 执行。将文件装入内存并执行。
- 添加。将新信息添加到文件结尾部分。
- 删除。删除文件，释放空间。
- 列表清单。列出文件名和文件属性。

关注公众号【乘龙考研】  
一手更新 稳定有保障

此外还可以对文件的重命名、复制、编辑等加以控制。这些高层的功能可以通过系统程序调用低层系统调用来实现。保护可以只在低层提供。例如，复制文件可利用一系列的读请求来完成，这样，具有读访问权限的用户同时也就具有了复制和打印权限。

## 2. 访问控制

解决访问控制最常用的方法是根据用户身份进行控制。而实现基于身份访问的最为普通的方法是，为每个文件和目录增加一个访问控制列表（Access-Control List, ACL），以规定每个用户名及其所允许的访问类型。这种方法的优点是可以使用复杂的访问方法，缺点是长度无法预计并且可能导致复杂的空间管理，使用精简的访问列表可以解决这个问题。

精简的访问列表采用拥有者、组和其他三种用户类型。

- 1) 拥有者。创建文件的用户。
- 2) 组。一组需要共享文件且具有类似访问的用户。
- 3) 其他。系统内的所有其他用户。

这样，只需用三个域即可列出访问表中这三类用户的访问权限。文件主在创建文件时，说明创建者用户名及所在的组名，系统在创建文件时也将文件主的名字、所属组名列在该文件的FCB中。用户访问该文件时，若用户是文件主，按照文件主所拥有的权限访问文件；若用户和文件主在同一个用户组，则按照同组权限访问，否则只能按其他用户权限访问。

口令和密码是另外两种访问控制方法。

口令指用户在建立一个文件时提供一个口令，系统为其建立FCB时附上相应口令，同时告诉允许共享该文件的其他用户。用户请求访问时必须提供相应的口令。这种方法时间和空间的开销不多，缺点是口令直接存在系统内部，不够安全。

密码指用户对文件进行加密，文件被访问时需要使用密钥。这种方法保密性强，节省了存储空间，不过编码和译码要花费一定的时间。

口令和密码都是防止用户文件被他人存取或窃取，并没有控制用户对文件的访问类型。

注意两个问题：

- 1) 现代操作系统常用的文件保护方法是，将访问控制列表与用户、组和其他成员访问控制方案一起组合使用。
- 2) 对于多级目录结构而言，不仅需要保护单个文件，而且需要保护子目录内的文件，即需要提供目录保护机制。目录操作与文件操作并不相同，因此需要不同的保护机制。

### 4.1.5 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形式。文件的物理结构（又称文件的存储结构，见下一节）是从实现观点出发看到的文件在外存上的存储组织形式。文件的逻辑结构与存储介质特性无关，它实际上是指在文件的内部，数据逻辑上是如何组织起来的。

按逻辑结构，文件可划分为无结构文件和有结构文件两大类。

## 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。由于无结构文件没有结构，因而对记录的访问只能通过穷举搜索的方式，因此这种文件形式对大多数应用不适用。但字符流的无结构文件管理简单，用户可以方便地对其进行操作。所以，那些对基本信息单位操作不多的文件较适于采用字符流的无结构方式，如源程序文件、目标代码文件等。

## 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：

### (1) 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。顺序文件有以下两种结构：第一种是串结构，记录之间的顺序与关键字无关，通常是按存入时间的先后进行排列，对串结构文件进行检索必须从头开始顺序依次查找，比较费时。第二种是顺序结构，指文件中的所有记录按关键字顺序排列，可采用折半查找法，提高了检索效率。

在对记录进行批量操作，即每次要读或写一大批记录时，顺序文件的效率是所有逻辑文件中最高的。此外，对于顺序存储设备（如磁带），也只有顺序文件才能被存储并能有效地工作。在经常需要查找、修改、增加或删除单个记录的场合，顺序文件的性能也比较差。

### (2) 索引文件

对于定长记录文件，要查找第  $i$  条记录，可直接根据下式计算得到第  $i$  条记录相对于第 1 条记录的地址： $A_i = i \times L$ 。然而，对于可变长记录的文件，要查找第  $i$  条记录，必须顺序地查找前  $i - 1$  条记录，从而获得相应记录的长度  $L$ ，进而按下式计算出第  $i$  条记录的首址：

$$A_i = \sum_{i=0}^{i-1} L_i + 1$$

**注意：**假定每条记录前用一个字节指明该记录的长度。

变长记录文件只能顺序查找，效率较低。为此，可以建立一张索引表，为主文件的每个记录在索引表中分别设置一个表项，包含指向变长记录的指针（即逻辑起始地址）和记录长度，索引表按关键字排序，因此其本身也是一个定长记录的顺序文件。这样就把对变长记录顺序文件的检索转变为对定长记录索引文件的随机检索，从而加快了记录的检索速度。图 4.4 所示为索引文件示意图。

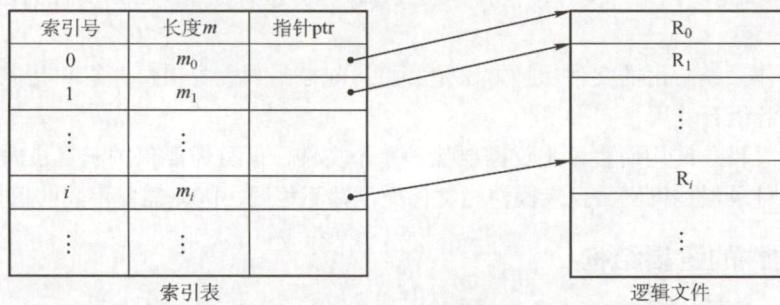


图 4.4 索引文件示意图

### (3) 索引顺序文件

索引顺序文件是顺序文件和索引文件的结合。最简单的索引顺序文件只使用了一级索引。索

关注公众号【乘龙考研】  
一手更新 稳定有保障

引顺序文件将顺序文件中的所有记录分为若干组，为顺序文件建立一张索引表，在索引表中为每组中的第一条记录建立一个索引项，其中含有该记录的关键字值和指向该记录的指针。

如图 4.5 所示，主文件名包含姓名和其他数据项。姓名为关键字，索引表中为每组的第一条记录（不是每条记录）的关键字值，用指针指向主文件中该记录的起始位置。索引表只包含关键字和指针两个数据项，所有姓名关键字递增排列。主文件中记录分组排列，同一个组中的关键字可以无序，但组与组之间的关键字必须有序。查找一条记录时，首先通过索引表找到其所在的组，然后在该组中使用顺序查找，就能很快地找到记录。

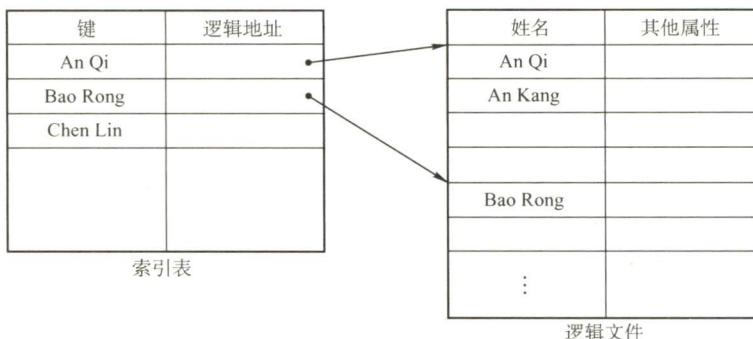


图 4.5 索引顺序文件示意图

对于含有  $N$  条记录的顺序文件，查找某关键字的记录时，平均需要查找  $N/2$  次。在索引顺序文件中，假设  $N$  条记录分为  $\sqrt{N}$  组，索引表中有  $\sqrt{N}$  个表项，每组有  $\sqrt{N}$  条记录，在查找某关键字的记录时，先顺序查找索引表，需要查找  $\sqrt{N}/2$  次，然后在主文件中对应的组中顺序查找，也需要查找  $\sqrt{N}/2$  次，因此共需查找  $\sqrt{N}/2 + \sqrt{N}/2 = \sqrt{N}$  次。显然，索引顺序文件提高了查找效率，若记录数很多，则可采用两级或多级索引。这种方式就是数据结构中的分块查找。

索引文件和索引顺序文件都提高了存取的速度，但因为配置索引表而增加了存储空间。

#### (4) 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。这种映射结构不同于顺序文件或索引文件，没有顺序的特性。

散列文件有很高的存取速度，但是会引起冲突，即不同关键字的散列函数值相同。

复习了数据结构的读者读到这里时，会有这样的感觉：有结构文件逻辑上的组织，是为在文件中查找数据服务的（顺序查找、索引查找、索引顺序查找、哈希查找）。

前面介绍了文件内部的逻辑结构，下面介绍多个文件之间在逻辑上是如何组织的，这实际上是文件“外部”的逻辑结构的问题。

### 4.1.6 文件的物理结构

前面说过，文件实际上是一种抽象数据类型，我们要研究它的逻辑结构、物理结构，以及关于它的一系列操作。文件的物理结构就是研究文件的实现，即文件数据在物理存储设备上是如何分布和组织的。同一个问题有两个方面的回答：一是文件的分配方式，讲的是对磁盘非空闲块的管理；二是文件存储空间管理，讲的是对磁盘空闲块的管理（详见 4.3 节）。

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。常用的磁盘空间分配方法有三种：连续分配、链接分配和索引分配。有的系统（如 RDOS 操作系统）对三种方法都支持，但更普遍的是一个系统只支持一种方法。对于本节的内容，读者要注意与文件的逻辑结构区分，从

历年经验来看，这是很多读者容易搞混的地方（读者复习完数据结构后，应该了解线性表、顺序表和链表之间的关系，类比到这里就不易混淆）。

### 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，如图 4.6 所示。磁盘地址定义了磁盘上的一个线性排序，这种排序使作业访问磁盘时需要的寻道数和寻道时间最小。

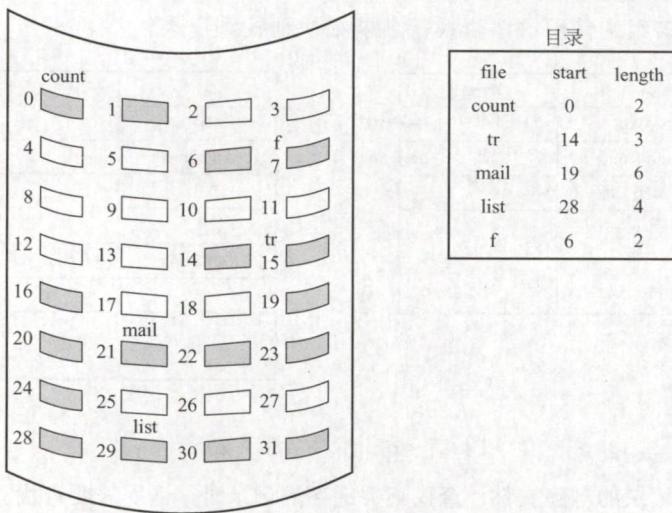


图 4.6 连续分配

采用连续分配时，逻辑文件中的记录也顺序存储在相邻接的块中。一个文件的目录项中“文件物理地址”字段应包括第一块的地址和该文件所分配区域的长度，若文件长  $n$  块并从位置  $b$  开始，则该文件将占有块  $b, b+1, b+2, \dots, b+n-1$ 。

连续分配支持顺序访问和直接访问。优点是实现简单、存取速度快。缺点是：①文件长度不宜动态增加，因为一个文件末尾后的盘块可能已分配给其他文件，一旦需要增加，就需要大量移动盘块。②为保持文件的有序性，删除和插入记录时，需要对相邻的记录做物理上的移动，还会动态改变文件的长度。③反复增删文件后会产生外部碎片（与内存管理分配方式中的碎片相似）。④很难确定一个文件需要的空间大小，因而只适用于长度固定的文件。

### 2. 链接分配

链接分配是一种采用离散分配的方式。它消除了磁盘的外部碎片，提高了磁盘的利用率。可以动态地为文件分配盘块，因此无须事先知道文件的大小。此外，对文件的插入、删除和修改也非常方便。链接分配又可分为隐式链接和显式链接两种形式。

#### (1) 隐式链接

隐式链接方式如图 4.7 所示。目录项中含有文件第一块的指针和最后一块的指针。每个文件对应一个磁盘块的链表；磁盘块分布在磁盘的任何地方，除最后一个盘块外，每个盘块都含有指向文件下一个盘块的指针，这些指针对用户是透明的。

隐式链接的缺点是只适合顺序访问，若要访问文件的第  $i$  个盘块，则只能从第 1 个盘块开始通过盘块指针顺序查找到第  $i$  块，随机访问效率很低。隐式链接的稳定性也是一个问题，系统在运行过程中由于软件或硬件错误导致链表中的指针丢失或损坏，会导致文件数据的丢失。

通常的解决方案是，将几个盘块组成簇（cluster），按簇而不按块来分配，可以成倍地减少查

找时间。比如一簇为4块，这样，指针所占的磁盘空间比例也要小得多。这种方法的代价是增加了内部碎片。簇可以改善许多算法的磁盘访问时间，因此应用于大多数操作系统。

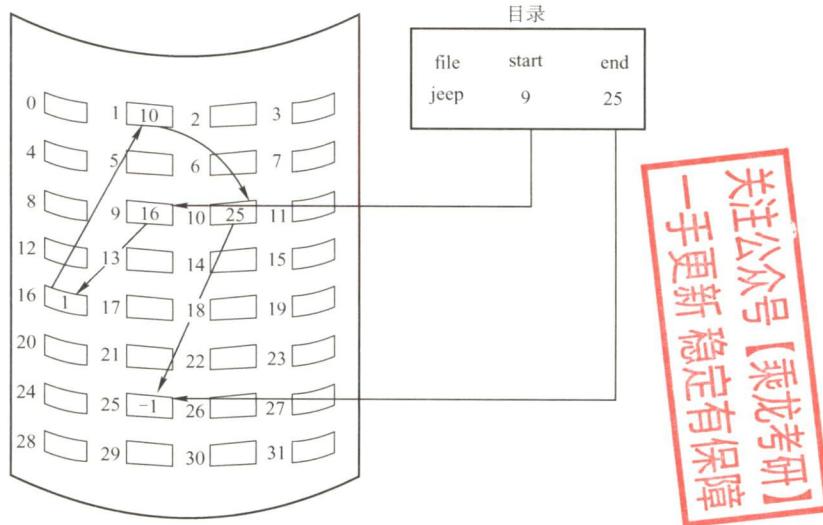


图 4.7 隐式链接分配

## (2) 显式链接

显式链接是指把用于链接文件各物理块的指针，从每个物理块的末尾中提取出来，显式地存放在内存的一张链接表中。该表在整个磁盘中仅设置一张，称为文件分配表（File Allocation Table, FAT）。每个表项中存放链接指针，即下一个盘块号。文件的第一个盘块号记录在目录项“物理地址”字段中，后续的盘块可通过查 FAT 找到。例如，

某磁盘共有 100 个磁盘块，存放了两个文件：文件“aaa”占三个盘块，依次是 2→8→5；文件“bbb”占两个盘块，依次是 7→1。其余盘块都是空闲盘块，则该磁盘的 FAT 表如图 4.8 所示。

不难看出，文件分配表 FAT 的表项与全部磁盘块一一对应，并且可以用一个特殊的数字 -1 表示文件的最后一块，可以用 -2 表示这个磁盘块是空闲的（当然也可指定为 -3、-4）。因此，FAT 不仅记录了文件各块之间的先后链接关系，同时还标记了空闲的磁盘块，操作系统也可以通过 FAT 对文件存储空间进行管理。

当某进程请求操作系统分配一个磁盘块时，操作系统只需从 FAT 中找到 -2 的表项，并将对应的磁盘块分配给进程即可。

FAT 表在系统启动时就会被读入内存，因此查找记录的过程是在内存中进行的，因而不仅显著地提高了检索速度，而且明显减少了访问磁盘的次数。

## 3. 索引分配

链接分配解决了连续分配的外部碎片和文件大小管理的问题。但依然存在问题：①链接分配不能有效支持直接访问（FAT 除外）；②FAT 需要占用较大的内存空间。事实上，在打开某个文件

盘块号	下一块
0	-2
1	-1
2	8
3	-2
4	-2
5	-1
6	-2
7	1
8	5
9	-2
...	...
98	-2
99	-2

文件目录

FAT(文件分配表)

图 4.8 文件分配表

时，只需将该文件对应盘块的编号调入内存即可，完全没有必要将整个 FAT 调入内存。为此，索引分配将每个文件所有的盘块号都集中放在一起构成索引块（表），如图 4.9 所示。

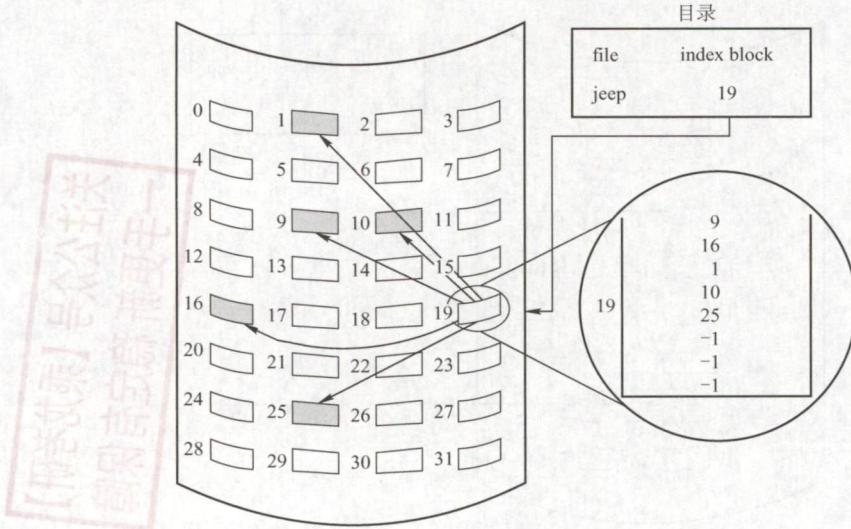


图 4.9 索引分配

每个文件都有其索引块，这是一个磁盘块地址的数组。索引块的第  $i$  个条目指向文件的第  $i$  块。要读第  $i$  块，通过索引块的第  $i$  个条目的指针来查找和读入所需的块。

索引分配的优点是支持直接访问，且没有外部碎片问题。缺点是由于索引块的分配，增加了系统存储空间的开销。索引块的大小是一个重要的问题，每个文件必须有一个索引块，因此索引块应尽可能小，但索引块太小就无法支持大文件。可以采用以下机制来处理这个问题。

- **链接方案。**一个索引块通常为一个磁盘块，因此它本身能直接读写。为了支持大文件，可以将多个索引块链接起来。
- **多层索引。**通过第一级索引块指向一组第二级的索引块，第二级索引块再指向文件块。查找时，通过第一级索引查找第二级索引，再采用这个第二级索引查找所需数据块。这种方法根据最大文件大小，可以继续到第三级或第四级。例如，4096B 的块，能在索引块中存入 1024 个 4B 的指针。两级索引支持 1048576 个数据块，即支持最大文件为 4GB。
- **混合索引。**将多种索引分配方式相结合的分配方式。例如，系统既采用直接地址，又采用单级索引分配方式或两级索引分配方式。该内容为高频考点，下面用一节专门介绍。

此外，访问文件需两次访问外存，先读取索引块的内容，然后访问具体的磁盘块，因而降低了文件的存取速度。为了解决这一问题，通常将文件的索引块读入内存，以提高访问速度。

#### 4. 混合索引分配

为了能够较全面地照顾到小型、中型、大型和特大型文件，可采用混合索引分配方式。对于小文件，为了提高对众多小文件的访问速度，最好能将它们的每个盘块地址直接放入 FCB，这样就可以直接从 FCB 中获得该文件的盘块地址，即为直接寻址。对于中型文件，可以采用单级索引方式，需要先从 FCB 中找到该文件的索引表，从中获得该文件的盘块地址，即为一次间址。对于大型或特大型文件，可以采用两级和三级索引分配方式。UNIX 系统采用的就是这种分配方式，在其索引结点中，共设有 13 个地址项，即  $i.\text{addr}(0) \sim i.\text{addr}(12)$ ，如图 4.10 所示。

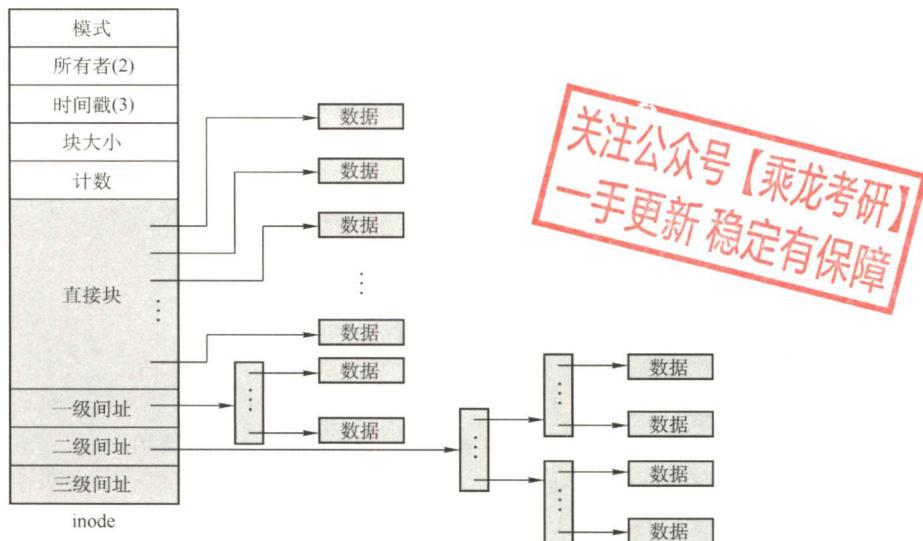


图 4.10 UNIX 系统的 inode 结构示意图

- 1) 直接地址。为了提高对文件的检索速度，在索引结点中可设置 10 个直接地址项，即用 `i.addr(0)~i.addr(9)` 来存放直接地址，即文件数据盘块的盘块号。假如每个盘块的大小为 4KB，当文件不大于 40KB 时，便可直接从索引结点中读出该文件的全部盘块号。
- 2) 一次间接地址。对于中、大型文件，只采用直接地址并不现实的。为此，可再利用索引结点中的地址项 `i.addr(10)` 来提供一次间接地址。这种方式的实质就是一级索引分配方式。图中的一次间址块也就是索引块，系统将分配给文件的多个盘块号记入其中。在一次间址块中可存放 1024 个盘块号，因而允许文件长达 4MB。
- 3) 多次间接地址。当文件长度大于  $4MB + 40KB$  (一次间接地址与 10 个直接地址项) 时，系统还需采用二次间接地址分配方式。这时，用地址项 `i.addr(11)` 提供二次间接地址。该方式的实质是两级索引分配方式。系统此时在二次间址块中记入所有一次间址块的盘号。地址项 `i.addr(11)` 作为二次间址块，允许文件最大长度可达 4GB。同理，地址项 `i.addr(12)` 作为三次间址块，其允许的文件最大长度可达 4TB。

#### 4.1.7 本节小结

本节开头提出的问题的参考答案如下。

- 1) 什么是文件？

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，它的形式多样。

- 2) 单个文件的逻辑结构和物理结构之间是否存在某些制约关系？

文件的逻辑结构是用户可见的结构，即从用户角度看到的文件的全貌。文件的物理结构是文件在存储器上的组织结构，它表示一个文件在辅存上安置、链接、编目的方法。它和文件的存取方法以及存储设备的特性等都有着密切的联系。单个文件的逻辑结构和物理结构之间虽无明显的制约或关联关系，但是如果物理结构选择不慎，也很难体现出逻辑结构的特点，比如一个逻辑结构是顺序结构，而物理结构是隐式链接结构的文件，即使理论上可以很快找出某条记录的地址，而实际找时仍然需要在磁盘上一块一块地找。

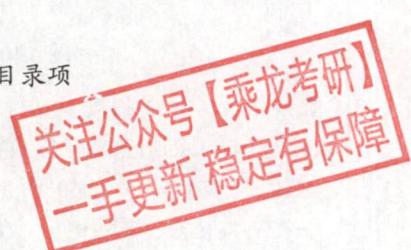
学到这里时，读者应能有这样的体会：现代操作系统的思想中，到处能见到面向对象程序设

计的影子。本节我们学习的一个新概念——文件，实质上就是一个抽象数据类型，也就是一种数据结构，若读者在复习操作系统之前已复习完数据结构，则遇到一种新的数据结构时，一定会有这样的意识：要认识它的逻辑结构、物理结构，以及对这种数据结构的操作。操作系统对文件的操作不是本课程重点关心的问题，无须做深入研究。

### 4.1.8 本节习题精选

#### 一、单项选择题

01. UNIX 操作系统中，输入/输出设备视为（ ）。
  - A. 普通文件
  - B. 目录文件
  - C. 索引文件
  - D. 特殊文件
02. 文件系统在创建一个文件时，为它建立一个（ ）。
  - A. 文件目录项
  - B. 目录文件
  - C. 逻辑结构
  - D. 逻辑空间
03. 打开文件操作的主要工作是（ ）。
  - A. 把指定文件的目录项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项
  - D. 在内存寻找指定的文件
04. 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
05. FAT32 的文件目录项不包括（ ）。
  - A. 文件名
  - B. 文件访问权限说明
  - C. 文件控制块的物理位置
  - D. 文件所在的物理位置
06. 有些操作系统中将文件描述信息从目录项中分离出来，这样做的好处是（ ）。
  - A. 减少读文件时的 I/O 信息量
  - B. 减少写文件时的 I/O 信息量
  - C. 减少查找文件时的 I/O 信息量
  - D. 减少复制文件时的 I/O 信息量
07. 操作系统为保证未经文件拥有者授权，任何其他用户不能使用该文件，所提供的解决方法是（ ）。
  - A. 文件保护
  - B. 文件保密
  - C. 文件转储
  - D. 文件共享
08. 在文件系统中，以下不属于文件保护的方法是（ ）。
  - A. 口令
  - B. 存取控制
  - C. 用户权限表
  - D. 读写之后使用关闭命令
09. 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
10. 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
11. 为了对文件系统中的文件进行安全管理，任何一个用户在进入系统时都必须进行注册，



- 这一级安全管理是( )。
- 系统级
  - 目录级
  - 用户级
  - 文件级
12. 下列说法中, ( )属于文件的逻辑结构的范畴。
- 连续文件
  - 系统文件
  - 链接文件
  - 流式文件
13. 文件的逻辑结构是为了方便( )而设计的。
- 存储介质特性
  - 操作系统的管理方式
  - 主存容量
  - 用户
14. 索引文件由逻辑文件和( )组成。
- 符号表
  - 索引表
  - 交叉访问表
  - 链接表
15. 下列关于索引表的叙述中, ( )是正确的。
- 索引表中每条记录的索引项可以有多个
  - 对索引文件存取时, 必须先查找索引表
  - 索引表中含有索引文件的数据及其物理地址
  - 建立索引的目的之一是减少存储空间
16. 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约查找( )次记录。
- 500
  - 50
  - 101
  - 100
17. 用磁带做文件存储介质时, 文件只能组织成( )。
- 顺序文件
  - 链接文件
  - 索引文件
  - 目录文件
18. 以下不适合直接存取的外存分配方式是( )。
- 连续分配
  - 链接分配
  - 索引分配
  - 以上答案都适合
19. 在以下文件的物理结构中, 不利于文件长度动态增长的是( )。
- 连续结构
  - 链接结构
  - 索引结构
  - 散列结构
20. 文件系统中若文件的物理结构采用连续结构, 则 FCB 中有关文件的物理位置的信息应包括( )。
- 首块地址
  - 文件长度
  - 索引表地址
- 仅 I
  - I、II
  - II、III
  - I、III
21. 在磁盘上, 最容易导致存储碎片发生的物理文件结构是( )。
- 隐式链接
  - 顺序存放
  - 索引存放
  - 显式链接
22. 文件系统采用两级索引分配方式。若每个磁盘块的大小为 1KB, 每个盘块号占 4B, 则该系统中, 单个文件的最大长度是( )。
- 64MB
  - 128MB
  - 32MB
  - 以上答案都不对
23. 设有一个记录文件, 采用链接分配方式, 逻辑记录的固定长度为 100B, 在磁盘上存储时采用记录成组分解技术。盘块长度为 512B。若该文件的目录项已经读入内存, 则对第 22 个逻辑记录完成修改后, 共启动了磁盘( )次。
- 3
  - 4
  - 5
  - 6
24. 物理文件的组织方式是由( )确定的。
- 应用程序
  - 主存容量
  - 外存容量
  - 操作系统
25. 文件系统为每个文件创建一张( ), 存放文件数据块的磁盘存放位置。

关注公众号【乘龙考研】  
一手更新 稳定有保障

- A. 打开文件表      B. 位图      C. 索引表      D. 空闲盘块链表
26. 下面关于索引文件的叙述中，正确的是（ ）。
- A. 索引文件中，索引表的每个表项中含有相应记录的关键字和存放该记录的物理地址  
 B. 顺序文件进行检索时，首先从 FCB 中读出文件的第一个盘块号；而对索引文件进行检索时，应先从 FCB 中读出文件索引块的开始地址  
 C. 对于一个具有三级索引的文件，存取一条记录通常要访问三次磁盘  
 D. 文件较大时，无论是进行顺序存取还是进行随机存取，通常索引文件方式都最快
27. 设某文件为链接文件，它由 5 个逻辑记录组成，每个逻辑记录的大小与磁盘块的大小相等，均为 512B，并依次存放在 50, 121, 75, 80, 63 号磁盘块上。若要存取文件的第 1569 逻辑字节处的信息，则应该访问（ ）号磁盘块。
- A. 3      B. 80      C. 75      D. 63
28. 【2009 统考真题】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
29. 【2009 统考真题】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
30. 【2010 统考真题】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
31. 【2012 统考真题】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
32. 【2013 统考真题】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
33. 【2013 统考真题】若某文件系统索引结点（inode）中有直接地址项和间接地址项，则下列选项中，与单个文件长度无关的因素是（ ）。
- A. 索引结点的总数      B. 间接地址索引的级数  
 C. 地址项的个数      D. 文件块大小
34. 【2013 统考真题】为支持 CD-ROM 中视频文件的快速随机播放，播放性能最好的文件数据块组织方式是（ ）。
- A. 连续结构      B. 链式结构      C. 直接索引结构      D. 多级索引结构
35. 【2014 统考真题】在一个文件被用户进程首次打开的过程中，操作系统需做的是（ ）。
- A. 将文件内容读到内存中  
 B. 将文件控制块读到内存中  
 C. 修改文件控制块中的读写权限  
 D. 将文件的数据缓冲区首指针返回给用户进程

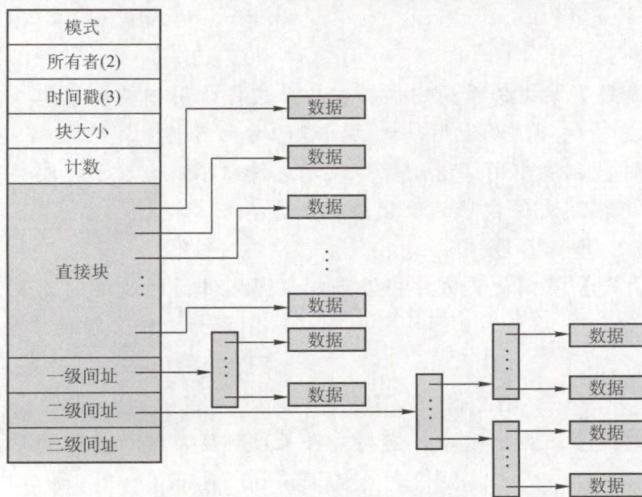
36. 【2015 统考真题】在文件的索引结点中存放直接索引指针 10 个，一级和二级索引指针各 1 个。磁盘块大小为 1KB，每个索引指针占 4B。若某文件的索引结点已在内存中，则把该文件偏移量（按字节编址）为 1234 和 307400 处所在的磁盘块读入内存，需访问的磁盘块个数分别是（ ）。
- A. 1, 2      B. 1, 3      C. 2, 3      D. 2, 4
37. 【2017 统考真题】某文件系统中，针对每个文件，用户类别分为 4 类：安全管理员、文件主、文件主的伙伴、其他用户；访问权限分为 5 种：完全控制、执行、修改、读取、写入。若文件控制块中用二进制位串表示文件权限，为表示不同类别用户对一个文件的访问权限，则描述文件权限的位数至少应为（ ）。
- A. 5      B. 9      C. 12      D. 20
38. 【2018 统考真题】下列优化方法中，可以提高文件访问速度的是（ ）。
- |          |               |
|----------|---------------|
| I. 提前读   | II. 为文件分配连续的簇 |
| III. 延迟写 | IV. 采用磁盘高速缓存  |
- A. 仅 I、II      B. 仅 II、III      C. 仅 I、III、IV      D. I、II、III、IV
39. 【2020 统考真题】下列选项中，支持文件长度可变、随机访问的磁盘存储空间分配方式是（ ）。
- A. 索引分配      B. 链接分配      C. 连续分配      D. 动态分区分配
40. 【2020 统考真题】某文件系统的目录项由文件名和索引结点号构成。若每个目录项长度为 64 字节，其中 4 字节存放索引结点号，60 字节存放文件名。文件名由小写英文字母构成，则该文件系统能创建的文件数量的上限为（ ）。
- A.  $2^{26}$       B.  $2^{32}$       C.  $2^{60}$       D.  $2^{64}$

## 二、综合应用题

01. 简述文件的外存分配中，连续分配、链接分配和索引分配各自的主要优缺点。
02. 在实现文件系统时，为加快文件目录的检索速度，可利用“FCB 分解法”。假设目录文件存放在磁盘上，每个盘块 512B。FCB 占 64B，其中文件名占 8B。通常将 FCB 分解成两部分，第一部分占 10B（包括文件名和文件内部号），第二部分占 56B（包括文件内部号和文件的其他描述信息）。
- 1) 假设某一目录文件共有 254 个 FCB，试分别给出采用分解法前和分解法后，查找该目录文件的某个 FCB 的平均访问磁盘次数（访问每个文件的概率相同）。
  - 2) 一般地，若目录文件分解前占用  $n$  个盘块，分解后改用  $m$  个盘块存放文件名和文件内部号，请给出访问磁盘次数减少的条件（假设  $m$  和  $n$  个盘块中都正好装满）。
03. 假定磁盘块的大小为 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？
04. 在 UNIX 操作系统中，给文件分配外存空间采用的是混合索引分配方式，如下图所示。UNIX 系统中的某个文件的索引结点指示出了为该文件分配的外存的物理块的寻找方法。在该索引结点中，有 10 个直接块（每个直接块都直接指向一个数据块），有 1 个一级间接块、1 个二级间接块及 1 个三级间接块，间接块指向的是一个索引块，每个索引块和数据块的大小均为 4KB，而 UNIX 系统中地址所占空间为 4B（指针大小为 4B），假设以下问题都建立在该索引结点已在内存中的前提下。  
现请回答：
- 1) 文件的大小为多大时可以只用到索引结点的直接块？

关注公众号【乘龙考研】  
一手更新 稳定有保障

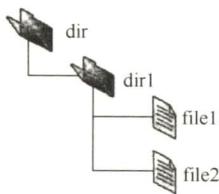
- 2) 该索引结点能访问到的地址空间大小总共为多大（小数点后保留 2 位）？  
 3) 若要读取一个文件的第 10000B 的内容，需要访问磁盘多少次？  
 4) 若要读取一个文件的第 10MB 的内容，需要访问磁盘多少次？



05. 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，一次间接索引项 1 项，二次间接索引项 1 项，三次间接索引项 1 项。数据块的大小为 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？
06. 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。
07. 【2011 统考真题】某文件系统为一级目录结构，文件的数据一次性写入磁盘，已写入的文件不可修改，但可多次创建新文件。请回答如下问题。
- 1) 在连续、链式、索引三种文件的数据块组织方式中，哪种更合适？说明理由。为定位文件数据块，需要在 FCB 中设计哪些相关描述字段？
  - 2) 为快速找到文件，对于 FCB，是集中存储好，还是与对应的文件数据块连续存储好？说明理由。
08. 【2012 统考真题】某文件系统空间的最大容量为 4TB ( $1TB = 2^{40}B$ )，以磁盘块为基本分配单位。磁盘块大小为 1KB。文件控制块 (FCB) 包含一个 512B 的索引表区。请回答下列问题：
- 1) 假设索引表区仅采用直接索引结构，索引表区存放文件占用的磁盘块号，索引表项中块号最少占多少字节？可支持的单个文件的最大长度是多少字节？
  - 2) 假设索引表区采用如下结构：第 0~7 字节采用<起始块号，块数>格式表示文件创建时预分配的连续存储空间。其中起始块号占 6B，块数占 2B，剩余 504B 采用直接索引结构，一个索引项占 6B，则可支持的单个文件的最大长度是多少字节？为使单个文件的长度达到最大，请指出起始块号和块数分别所占字节数的合理值并说明理由。
09. 【2014 统考真题】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，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

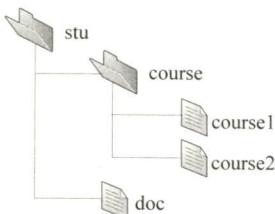
有足够的空闲磁盘空间，则完成上述插入操作最少需要访问多少次磁盘块？F 的文件控制块内容会发生哪些改变？

- 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？
10. 【2016 统考真题】某磁盘文件系统使用链接分配方式组织文件，簇大小为 4KB。目录文件的每个目录项包括文件名和文件的第一个簇号，其他簇号存放在文件分配表 FAT 中。
- 1) 假定目录树如下图所示，各文件占用的簇号及顺序如下表所示，其中 dir, dir1 是目录，file1, file2 是用户文件。请给出所有目录文件的内容。
  - 2) 若 FAT 的每个表项仅存放簇号，占 2B，则 FAT 的最大长度为多少字节？该文件系统支持的文件长度最大是多少？
  - 3) 系统通过目录文件和 FAT 实现对文件的按名存取，说明 file1 的 106, 108 两个簇号分别存放在 FAT 的哪个表项中。



文件名	簇号
dir	1
dir1	48
file1	100、106、108
file2	200、201、202

- 4) 假设仅 FAT 和 dir 目录文件已读入内存，若需将文件 dir/dir1/file1 的第 5000 个字节读入内存，则要访问哪几个簇？
11. 【2018 统考真题】某文件系统采用索引结点存放文件的属性和地址信息，簇大小为 4KB。每个文件索引结点占 64B，有 11 个地址项，其中直接地址项 8 个，一级、二级和三级间接地址项各 1 个，每个地址项长度为 4B。请回答下列问题：
- 1) 该文件系统能支持的最大文件长度是多少？（给出计算表达式即可）
  - 2) 文件系统用 1M ( $1M = 2^{20}$ ) 个簇存放文件索引结点，用 512M 个簇存放文件数据。若一个图像文件的大小为 5600B，则该文件系统最多能存放多少个这样的图像文件？
  - 3) 若文件 F1 的大小为 6KB，文件 F2 的大小为 40KB，则该文件系统获取 F1 和 F2 最后一个簇的簇号需要的时间是否相同？为什么？
12. 【2022 统考真题】某文件系统的磁盘块大小为 4 KB，目录项由文件名和索引结点号构成，每个索引结点占 256 字节，其中包含直接地址项 10 个，一级、二级和三级间接地址项各 1 个，每个地址项占 4 字节。该文件系统中子目录 stu 的结构如图(a)所示，stu 包含子目录 course 和文件 doc，course 子目录包含文件 course1 和 course2。各文件的文件名、索引结点号、占用磁盘块的块号如图(b)所示。请回答下列问题。



图(a)

文件名	索引结点号	磁盘块号
stu	1	10
course	2	20
course1	10	30
course2	100	40
doc	10	x

图(b)

- 1) 目录文件 stu 中每个目录项的内容是什么?
- 2) 文件 doc 占用的磁盘块的块号 x 的值是多少?
- 3) 若目录文件 course 的内容已在内存, 则打开文件 course1 并将其读入内存, 需要读几个磁盘块? 说明理由。
- 4) 若文件 course2 的大小增长到 6 MB, 则为了存取 course2 需要使用该文件索引结点的哪几级间接地址项? 说明理由。

#### 4.1.9 答案与解析

##### 一、单项选择题

01. D

UNIX 操作系统中, 所有设备都被视为特殊的文件, 因为 UNIX 操作系统控制和访问外部设备的方式和访问一个文件的方式是相同的。

02. A

一个文件对应一个 FCB, 而一个文件目录项就是一个 FCB。

03. A

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。

04. D

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。

05. C

文件目录项即 FCB, 通常由文件基本信息、存取控制信息和使用信息组成。基本信息包括文件物理位置。文件目录项显然不包括 FCB 的物理位置信息。

06. C

将文件描述信息从目录项中分离, 即应用了索引结点的方法, 磁盘的盘块中可以存放更多的目录项, 查找文件时可以大大减少其 I/O 信息量。

07. A

文件保护是针对文件访问权限的保护。

08. D

在文件系统中, 口令、存取控制和用户权限表都是常用的文件保护方法。

09. A

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 FCB 中对文件访问的控制信息。

10. D

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。

**一手更新 稳定有保障**

**11. A**

系统级安全管理包括注册和登录。另外，通过“进入系统时”这个关键词也可推测出正确答案。

**12. D**

逻辑文件有两种：无结构文件（流式文件）和有结构式文件。连续文件和链接文件都属于文件的物理结构，而系统文件是按文件用途分类的。

**13. D**

文件结构包括逻辑结构和物理结构。逻辑结构是用户组织数据的结构形式，数据组织形式来自需求，而物理结构是操作系统组织物理存储块的结构形式。

因此说，逻辑文件的组织形式取决于用户，物理结构的选择取决于文件系统设计者针对硬件结构（如磁带介质很难实现链接结构和索引结构）所采取的策略（即A和B选项）。

**14. B**

索引文件由逻辑文件和索引表组成。文件的逻辑结构和物理结构都有索引的概念，引入逻辑索引和物理索引的目的是截然不同的。逻辑索引的目的是加快文件数据的定位，是从用户角度出发的，而物理索引的主要目的是管理不连续的物理块，是从系统管理的角度出发的。

**15. B**

索引文件由逻辑文件和索引表组成，对索引文件存取时，必须先查找索引表。索引项只包含每条记录的长度和在逻辑文件中的起始位置。因为每条记录都要有一个索引项，因此提高了存储代价。

**16. C**

采用索引顺序文件时，最好情况是有 $\sqrt{10000} = 100$ 组，每组有100条记录，则查找100组平均需要 $(1 + 100)/2 = 50.5$ 次，组内查找平均需要 $(1 + 100)/2 = 50.5$ 次，共需要 $50.5 + 50.5 = 101$ 次。

**17. A**

磁带是一种顺序存储设备，用它存储文件时只能采用顺序存储结构。注意：若允许磁带来回倒带，也可组织为其他的文件形式，本题不做讨论。

**18. B**

直接存取即随机存取，采用连续分配和索引分配的文件都适合于直接存取方式，只有采用链接分配的文件不具有随机存取特性。

**19. A**

要求有连续的存储空间，所以必须事先知道文件的大小，然后根据其大小在存储空间中找出一块大小足够的存储区。如果文件动态地增长，那么会使文件所占的空间越来越大，即使事先知道文件的最终大小，在采用预分配的存储空间的方法时，也是很低效的，它会使大量的存储空间长期闲置。

**20. B**

在连续分配方式中，为了使系统能找到文件存放的地址，应在目录项的“文件物理地址”字段中，记录该文件第一条记录所在的盘块号和文件长度（以盘块数进行计量）。

**21. B**

顺序文件占用连续的磁盘空间，容易导致存储碎片（外部碎片）的发生。

**22. A**

每个磁盘块中最多有 $1KB/4B = 256$ 个索引项，则两级索引分配方式下单个文件的最大长度

为  $256 \times 256 \times 1\text{KB} = 64\text{MB}$ 。

**23. D**

第 22 个逻辑记录存放在第 5 个物理块中 ( $22 \times 100 / 512 = 4$ , 余 152), 由于文件采用的物理结构是链接文件, 因此需要从目录项所指的第一个物理块开始读取, 共启动磁盘 5 次。修改后还需要写回操作, 由于写回时已获得该块的物理地址, 只需启动磁盘 1 次, 因此共需启动磁盘 6 次。

**24. D**

通常用户可以根据需要来确定文件的逻辑结构, 而文件的物理结构是由操作系统的设计师根据文件存储器的特性来确定的, 一旦确定, 就由操作系统管理。

**25. C**

打开文件表仅存放已打开文件信息的表, 将指名文件的属性从外存复制到内存, 再使用该文件时直接返回索引, 选项 A 错误。位图和空闲盘块链表是磁盘管理方法, 选项 B、D 错误。只有索引表中记录每个文件所存放的盘块地址, 选项 C 正确。

**26. B**

索引表的表项中含有相应记录的关键字和存放该记录的逻辑地址; 三级索引需要访问 4 次磁盘; 无论是顺序存取还是随机存取, 顺序文件通常都是速度最快的。

**27. B**

因为  $1569 = 512 \times 3 + 33$ , 故要访问字节位于第 4 个磁盘块上, 对应的物理磁盘块号为 80。

**28. A**

为了实现“按名存取”, 在文件系统中为每个文件设置用于描述和控制文件的数据结构, 称之为文件控制块 (FCB)。在文件控制块中, 通常包含三类信息, 即基本信息、存取控制信息及使用信息。

**29. B**

文件的物理结构包括连续、链式、索引三种, 其中链式结构不能实现随机访问, 连续结构的文件不易于扩展。因此随机访问且易于扩展是索引结构的特性。

**30. C**

每个磁盘索引块和磁盘数据块大小均为 256B, 每个磁盘索引块有  $256 / 4 = 64$  个地址项。因此, 4 个直接地址索引指向的数据块大小为  $4 \times 256\text{B}$ ; 2 个一级间接索引包含的直接地址索引数为  $2 \times (256 / 4)$ , 即其指向的数据块大小为  $2 \times (256 / 4) \times 256\text{B}$ 。1 个二级间接索引所包含的直接地址索引数为  $(256 / 4) \times (256 / 4)$ , 即其所指向的数据块大小为  $(256 / 4) \times (256 / 4) \times 256\text{B}$ 。因此, 7 个地址项所指向的数据块总大小为  $4 \times 256 + 2 \times (256 / 4) \times 256 + (256 / 4) \times (256 / 4) \times 256 = 1082368\text{B} = 1057\text{KB}$ 。

**31. A**

对于 I, 当所读文件的数据不在内存时, 产生中断 (缺页中断), 原进程进入阻塞态, 直到所需数据从外存调入内存后, 才将该进程唤醒。对于 II, read 系统调用通过陷入将 CPU 从用户态切换到核心态, 从而获取操作系统提供的服务。对于 III, 要读一个文件, 首先要用 open 系统调用打开该文件。open 中的参数包含文件的路径名与文件名, 而 read 只需使用 open 返回的文件描述符, 并不使用文件名作为参数。read 要求用户提供三个输入参数: ①文件描述符 fd; ②buf 缓冲区首址; ③传送的字节数 n。read 的功能是试图从 fd 所指示的文件中读入 n 个字节的数据, 并将它们送至由指针 buf 所指示的缓冲区中。

**32. A**

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。

**33. A**

四个选项中，只有 A 选项是与单个文件长度无关的。

**34. A**

为了实现快速随机播放，要保证最短的查询时间，即不能选取链表和索引结构，因此连续结构最优。

**35. B**

一个文件被用户进程首次打开即被执行了 open 操作，会把文件的 FCB 调入内存，而不会把文件内容读到内存中，只有进程希望获取文件内容时才会读入文件内容；选项 C、D 明显错误，选择选项 B。

**36. B**

10 个直接索引指针指向的数据块大小为  $10 \times 1KB = 10KB$ 。每个索引指针占 4B，则每个磁盘块可存放  $1KB/4B = 256$  个索引指针，一级索引指针指向的数据块大小为  $256 \times 1KB = 256KB$ ，二级索引指针指向的数据块大小为  $256 \times 256 \times 1KB = 2^{16}KB = 64MB$ 。

按字节编址，偏移量为 1234 时，因  $1234B < 10KB$ ，由直接索引指针可得到其所在的磁盘块地址。文件的索引结点已在内存中，因此地址可直接得到，因此仅需 1 次访盘即可。

偏移量为 307400 时，因  $10KB + 256KB < 307400B < 64MB$ ，可知该偏移量的内容在二级索引指针所指向的某个磁盘块中，索引结点已在内存中，因此先访盘 2 次得到文件所在的磁盘块地址，再访盘 1 次即可读出内容，共需 3 次访盘。

**37. D**

可以把用户访问权限抽象为一个矩阵，行代表用户，列代表访问权限。这个矩阵有 4 行 5 列，1 代表 true，0 代表 false，所以需要 20 位，选择选项 D。

**38. D**

II 和 IV 显然均能提高文件访问速度。对于 I，提前读是指在读当前盘块时，将下一个可能要访问的盘块数据读入缓冲区，以便需要时直接从缓冲区中读取，提高了文件的访问速度。对于 III，延迟写是指先将数据写入缓冲区，并置上“延迟写”标志，以备不久之后访问，当缓冲区需要再次被分配出去时，才将缓冲区数据写入磁盘，减少了访问磁盘的次数，提高了文件的访问速度，III 也正确，答案选择选项 D。

**39. A**

索引分配支持变长的文件，同时可以随机访问文件的指定数据块，选项 A 正确。链接分配不支持随机访问，需要依靠指针依次访问，选项 B 错误。连续分配的文件长度固定，不支持可变文件长度（连续分配的文件长度虽然也可变，但是需要大量移动数据，代价较大，相比之下不太合适），选项 C 错误。动态分区分配是内存管理方式，不是磁盘空间的管理方式，选项 D 错误。

**40. B**

在总长为 64 字节的目录项中，索引结点占 4 字节，即 32 位。不同目录下的文件的文件名可以相同，所以在考虑系统创建最多文件数量时，只需考虑索引结点的个数，即创建文件数量上限 = 索引结点数量上限。整个系统中最多存储  $2^{32}$  个索引结点，因此整个系统最多可以表示  $2^{32}$  个文件，选项 B 正确。

## 二、综合应用题

### 01. 【解答】

连续分配方式的优点是可以随机访问（磁盘），访问速度快；缺点是要求有连续的存储空间，

容易产生碎片，降低磁盘空间利用率，并且不利于文件的增长扩充。

链接分配方式的优点是不要求连续的存储空间，能更有效地利用磁盘空间，并且有利于扩充文件；缺点是只适合顺序访问，不适合随机访问；另外，链接指针占用一定的空间，降低了存储效率，可靠性也差。

索引分配方式的优点是既支持顺序访问又支持随机访问，查找效率高，便于文件删除；缺点是索引表会占用一定的存储空间。

### 02. 【解答】

目录是存放在磁盘上的，检索目录时需要访问磁盘，速度很慢。利用“FCB 分解法”加快目录检索速度的原理是：将 FCB 的一部分分解出去，存放在另一个数据结构中，而在目录中仅留下文件的基本信息和指向该数据结构的指针，这样一来就有效地缩减了目录的体积，减少了目录所占磁盘的块数，检索目录时读取磁盘的次数也减少，于是就加快了检索目录的速度。

因为原本整个 FCB 都是在目录中的，而 FCB 分解法将 FCB 的部分内容放在了目录外，所以检索完目录后还需要读取一次磁盘，以找齐 FCB 的所有内容。

1) 分解法前，目录的磁盘块数为  $64 \times 254 / 512 = 31.75$ ，即 32 块。前 31 块中每块放了  $512 / 64 = 8$  个，而最后一块放了  $254 - 31 \times 8 = 6$  个。所以查找该目录文件的某个 FCB 的平均访问磁盘次数  $= [8 \times (1 + 2 + 3 + \dots + 31) + 6 \times 32] / 254 = 16.38$  次。

分解法后，目录的磁盘块数为  $10 \times 254 / 512 = 4.96$ ，即 5 块。前 4 块中每块放了  $512 / 10 = 51$  个，而最后一块放了  $254 - 4 \times 51 = 50$  个。所找的目录项在第 1, 2, 3, 4, 5 块所需的磁盘访问次数分别为 2, 3, 4, 5, 6 次。所以查找该目录文件的某个 FCB 的平均访问磁盘次数  $= [51 \times (2 + 3 + 4 + 5) + 50 \times 6] / 254 = 3.99$  次。

2) 分解法前，平均访问磁盘次数  $= (1 + 2 + 3 + \dots + n) / n = (n + 1) / 2$  次。

分解法后，平均访问磁盘次数  $= [2 + 3 + 4 + \dots + (m + 1)] / m = (m + 3) / 2$  次。

为了使访问磁盘次数减少，显然需要  $(m + 3) / 2 < (n + 1) / 2$ ，即  $m < n - 2$ 。

**注意：**第二问中的每个盘块都正好装满，相当于访问每个盘块的概率是相等的，因此计算起来比第一问方便很多。

### 03. 【解答】

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  $20 / 8 = 2.5B$ ，这样 FAT 占用的存储空间大小为  $2.5B \times 540K = 1350KB$ 。

### 04. 【解答】

1) 要想只用到索引结点的直接块，这个文件应能全部在 10 个直接块指向的数据块中放下，而数据块的大小为 4KB，所以该文件的大小应小于或等于  $4KB \times 10 = 40KB$ ，即文件的大小不超过 40KB 时可以只用到索引结点的直接块。

2) 只需要算出索引结点指向的所有数据块的块数，再乘以数据块的大小即可。直接块指向的数据块数 = 10 块。

一级间接块指向的索引块里的指针数为  $4KB / 4B = 1024$ ，所以一级间接块指向的数据块数为 1024 块。

二级间接块指向的索引块里的指针数为  $4KB / 4B = 1024$ ，指向的索引块里再拥有  $4KB / 4B = 1024$  个指针数。所以二级间接块指向的数据块数为  $(4KB / 4B)^2 = 1024^2$ 。

三级间接块指向的数据块数为 $(4KB/4B)^3 = 1024^3$ 。所以，该索引结点能访问到的地址空间大小为

$$\left[ 10 + 1 \times \frac{4KB}{4B} + 1 \times \left( \frac{4KB}{4B} \right)^2 + 1 \times \left( \frac{4KB}{4B} \right)^3 \right] \times 4KB \approx 4100.00GB = 4.00TB$$

- 3) 因为  $10000B/4KB = 2.44$ ，所以第 10000B 的内容存放在第 3 个直接块中，若要读取一个文件的第 10000B 的内容，需要访问磁盘 1 次。
- 4) 因为 10MB 的内容需要数据块数为  $10MB/4KB = 2.5 \times 1024$ ，直接块和一级间接块指向的数据块数  $= 10 + (4KB/4B) = 1034 < 2.5 \times 1024$ ，直接块和一级间接块及二级间接块的数据块数  $= 10 + (4KB/4B) + (4KB/4B)^2 > 1 \times 1024^2 > 2.5 \times 1024$ ，所以第 10MB 数据应该在二级间接块下属的某个数据块中，若要读取一个文件的第 10MB 的内容，需要访问磁盘 3 次。

#### 05. 【解答】

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) 中的计算知，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。  
一次间接索引项使用 1 个间接索引块，二次间接索引项使用  $1 + \lceil (512 \times 1024 - 10 - 1024) / 1024 \rceil \approx 512$  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  $2GB + 2MB + 4KB + 52B$ 。

#### 06. 【解答】

由于块长为 512B，每个块号长 2B，因此一个一级索引表可容纳 256 个磁盘块地址。同样，一个二级索引表可容纳 256 个磁盘块地址，一个三级索引表也可容纳 256 个磁盘块地址。

所以采用二级索引时，可寻址的文件最大长度是

$$256 \times 256 \times 512 = 32MB$$

采用三级索引时，可寻址的文件最大长度是

$$256 \times 256 \times 256 \times 512 = 8GB$$

关注公众号【乘龙考研】  
一手更新 稳定有保障

#### 07. 【解答】

- 1) 在磁盘中连续存放（采取连续结构），磁盘寻道时间更短，文件随机访问效率更高；在 FCB 中加入的字段为<起始块号，块数>或<起始块号，结束块号>。
- 2) 将所有的 FCB 集中存放，文件数据集中存放。这样在随机查找文件名时，只需访问 FCB 对应的块，可减少磁头移动和磁盘 I/O 访问次数。

#### 08. 【解答】

- 1) 文件系统中所能容纳的磁盘块总数为  $4TB/1KB = 2^{32}$ 。要完全表示所有磁盘块，索引项中的块号最少要占  $32/8 = 4B$ 。而索引表区仅采用直接索引结构，因此 512B 的索引表区能

容纳  $512B/4B = 128$  个索引项。每个索引项对应一个磁盘块，所以该系统可支持的单个文件最大长度是  $128 \times 1KB = 128KB$ 。

- 2) 这里考查的分配方式不同于我们熟悉的三种经典分配方式，但题目中给出了详细的解释。所求的单个文件最大长度一共包含两部分：预分配的连续空间和直接索引区。

连续区块数占 2B，共可表示  $2^{16}$  个磁盘块，即  $2^{26}B$ 。直接索引区共  $504B/6B = 84$  个索引项。所以该系统可支持的单个文件最大长度是  $2^{26}B + 84KB$ 。

为了使单个文件的长度达到最大，应使连续区的块数字段表示的空间大小尽可能接近系统最大容量 4TB。分别设起始块号和块数占 4B，这样起始块号可以寻址的范围是  $2^{32}$  个磁盘块，共 4TB，即整个系统空间。同样，块数字段可以表示最多  $2^{32}$  个磁盘块，共 4TB。

#### 09. 【解答】

- 1) 系统采用顺序分配方式时，插入记录需要移动其他的记录块，整个文件共有 200 条记录，要插入新记录作为第 30 条，而存储区前后均有足够的磁盘空间，且要求最少的访问存储块数，则要把文件前 29 条记录前移，若算访盘次数，移动一条记录读出和存回磁盘各是一次访盘，29 条记录共访盘 58 次，存回第 30 条记录访盘 1 次，共访盘 59 次。

F 的文件控制区的起始块号和文件长度的内容会因此改变。

- 2) 文件系统采用链接分配方式时，插入记录并不用移动其他记录，只需找到相应的记录，修改指针即可。插入的记录为其第 30 条记录，因此需要找到文件系统的第 29 块，一共需要访盘 29 次，然后把第 29 块的下块地址部分赋给新块，把新块存回磁盘会访盘 1 次，然后修改内存中第 29 块的下块地址字段，再存回磁盘，一共访盘 31 次。

4B 共 32 位，可以寻址  $2^{32} = 4G$  块存储块，每块的大小为 1KB，即 1024B，其中下块地址部分占 4B，数据部分占 1020B，因此该系统的文件最大长度是  $4G \times 1020B = 4080GB$ 。

#### 10. 【解答】

- 1) 两个目录文件 dir 和 dir1 的内容如下表所示。

dir 目录文件		dir1 目录文件	
文件名	簇号	文件名	簇号
dir1	48	file1	100
		file2	200

- 2) 由于 FAT 的簇号为 2 个字节，即 16 比特，因此在 FAT 表中最多允许  $2^{16}$  (65536) 个表项，一个 FAT 文件最多包含  $2^{16}$  (65536) 个簇。FAT 的最大长度为  $2^{16} \times 2B = 128KB$ 。文件的最大长度是  $2^{16} \times 4KB = 256MB$ 。

- 3) 在 FAT 的每个表项中存放下一个簇号。file1 的簇号 106 存放在 FAT 的 100 号表项中，簇号 108 存放在 FAT 的 106 号表项中。
- 4) 先在 dir 目录文件里找到 dir1 的簇号，然后读取 48 号簇，得到 dir1 目录文件，接着找到 file1 的第一个簇号，据此在 FAT 里查找 file1 的第 5000 个字节所在的簇号，最后访问磁盘中的该簇。因此，需要访问目录文件 dir1 所在的 48 号簇，及文件 file1 的 106 号簇。

#### 11. 【解答】

- 1) 簇大小为 4KB，每个地址项长度为 4B，因此每簇有  $4KB/4B = 1024$  个地址项。最大文件的物理块数可达  $8 + 1 \times 1024 + 1 \times 1024^2 + 1 \times 1024^3$ ，每个物理块（簇）大小为 4KB，因此最大文件长度为  $(8 + 1 \times 1024 + 1 \times 1024^2 + 1 \times 1024^3) \times 4KB = 32KB + 4MB + 4GB + 4TB$ 。

- 2) 文件索引结点总个数为  $1M \times 4KB / 64B = 64M$ , 5600B 的文件占 2 个簇, 512M 个簇可存放的文件总个数为  $512M / 2 = 256M$ 。可表示的文件总个数受限于文件索引结点总个数, 因此能存储 64M 个大小为 5600B 的图像文件
- 3) 文件 F1 的大小为  $6KB < 4KB \times 8 = 32KB$ , 因此获取文件 F1 的最后一个簇的簇号只需要访问索引结点的直接地址项。文件 F2 大小为 40KB,  $4KB \times 8 < 40KB < 4KB \times 8 + 4KB \times 1024$ , 因此获取 F2 的最后一个簇的簇号还需要读一级索引表。综上, 需要的时间不相同。

## 12. 【解答】

- 1) 在该文件系统中, 目录项由文件名和索引结点号构成。由图(a)可知, stu 目录下有两个文件, 分别是 course 和 doc。由图(b)可知, 这两个文件分别对应索引结点号 2 和 10。因此, 目录文件 stu 中两个目录项的内容是

文件名	索引结点号
course	2
doc	10

- 2) 由图(b)可知, 文件 doc 和文件 course1 对应的索引结点号都是 10, 说明 doc 和 course1 两个目录项共享同一个索引结点, 本质上对应同一个文件。而文件 course1 存储在 30 号磁盘块, 因此文件 doc 占用的磁盘块的块号 x 为 30。
- 3) 需要读 2 个磁盘块。先读 course1 的索引结点所在的磁盘块, 再读 course1 的内容所在的磁盘块。目录文件 course 的内容已在内存中, 即 course1、course2 对应的目录项已在内存中, 根据 course1 对应的目录项可以知道其索引结点号, 即可读入 course1 的索引结点所在的磁盘块; 根据 course1 的索引结点可知该文件存储在 30 号磁盘块, 因此可再读入 course1 的内容所在的磁盘块。
- 4) 存取 course2 需要使用索引结点的一级和二级间接地址项。6MB 大小的文件需要占用  $6MB / 4KB = 1536$  个磁盘块。直接地址项可以记录 10 个磁盘块号, 一级间接地址块可以记录  $4KB / 4B = 1024$  个磁盘块号, 二级间接地址块可以记录  $1024 \times 1024$  个磁盘块号, 而  $10 + 1024 < 1536 < 10 + 1024 + 1024 \times 1024$ 。因此, 6MB 大小的文件, 需要使用一级间接地址项和二级间接地址项 (拓展: 若文件的总大小超出  $10 + 1024 + 1024 \times 1024$  块, 则还需使用三级间接地址项)。

## 4.2 目录

在学习本节时, 请读者思考以下问题:

- 1) 目录管理的要求是什么?
- 2) 在目录中查找某个文件可以使用什么方法?

上节介绍了文件的逻辑结构和物理结构, 本节将介绍目录的实现。文件目录也是一种数据结构, 用于标识系统中的文件及其物理地址, 供检索时使用。通常, 一个文件目录也被视为一个文件。在学习本节的内容时, 读者可以围绕目录管理的要求来思考。

### 4.2.1 目录的基本概念

上节说过, FCB 的有序集合称为文件目录, 一个 FCB 就是一个文件目录项。与文件管理系

关注公众号【乘龙考研】  
一手更新 稳定有保障

统和文件集合相关联的是文件目录，它包含有关文件的属性、位置和所有权等。

首先来看目录管理的基本要求：从用户的角度看，目录在用户（应用程序）所需要的文件名和文件之间提供一种映射，所以目录管理要实现“按名存取”；目录存取的效率直接影响到系统的性能，所以要提高对目录的检索速度；在多用户系统中，应允许多个用户共享一个文件，因此目录还需要提供用于控制访问文件的信息。此外，应允许不同用户对不同文件采用相同的名字，以便于用户按自己的习惯给文件命名，目录管理通过树形结构来解决和实现。

## 4.2.2 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，如图 4.11 所示。



图 4.11 单级目录结构

当访问一个文件时，先按文件名在该目录中查找到相应的 FCB，经合法性检查后执行相应操作。当建立一个新文件时，必须先检索所有目录项，以确保没有“重名”的情况，然后在该目录中增设一项，把新文件的属性信息填入到该项中。当删除一个文件时，先从该目录中找到该文件的目录项，回收该文件所占用的存储空间，然后清除该目录项。

单级目录结构实现了“按名存取”，但是存在查找速度慢、文件不允许重名、不便于文件共享等缺点，而且对于多用户的操作系统显然是不适用的。

### 2. 两级目录结构

为了克服单级目录所存在的缺点，可以采用两级方案，将文件目录分成主文件目录（Master File Directory, MFD）和用户文件目录（User File Directory, UFD）两级，如图 4.12 所示。

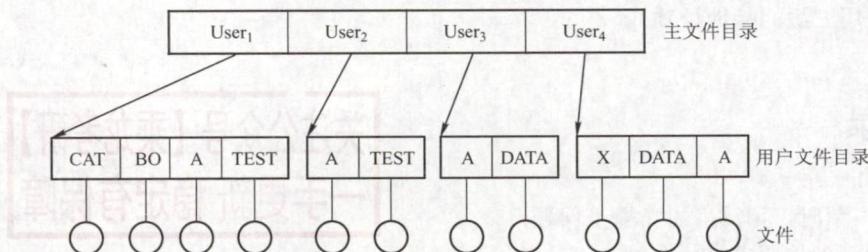


图 4.12 两级目录结构

主文件目录项记录用户名及相应用户文件目录所在的存储位置。用户文件目录项记录该用户文件的 FCB 信息。当某用户欲对其文件进行访问时，只需搜索该用户对应的 UFD，这既解决了不同用户文件的“重名”问题，又在一定程度上保证了文件的安全。

两级目录结构提高了检索的速度，解决了多用户之间的文件重名问题，文件系统可以在目录上实现访问限制。但是两级目录结构缺乏灵活性，不能对文件分类。

### 3. 树形目录结构

将两级目录结构加以推广，就形成了树形目录结构，如图 4.13 所示。它可以明显地提高对目录的检索速度和文件系统的性能。当用户要访问某个文件时，用文件的路径名标识文件，文件路径名是个字符串，由从根目录出发到所找文件通路上所有目录名与数据文件名用分隔符“/”链接而成。从根目录出发的路径称为绝对路径。当层次较多时，每次从根目录查询会浪费时间，于是加入了当前目录（又称工作目录），进程对各文件的访问都是相对于当前目录进行的。当用户要访问某个文件时，使用相对路径标识文件，相对路径由从当前目录出发到所找文件通路上所有目录名与数据文件名用分隔符“/”链接而成。

图 4.13 是 Linux 操作系统的目录结构，“/dev/hda”就是一个绝对路径。若当前目录为“/bin”，则“./ls”就是一个相对路径，其中符号“.”表示当前工作目录。

通常，每个用户都有各自的“当前目录”，登录后自动进入该用户的“当前目录”。操作系统提供一条专门的系统调用，供用户随时改变“当前目录”。例如，在 UNIX 系统中，“/etc/passwd”文件就包含有用户登录时默认的“当前目录”，可用 cd 命令改变“当前目录”。

树形目录结构可以很方便地对文件进行分类，层次结构清晰，也能够更有效地进行文件的管理和保护。在树形目录中，不同性质、不同用户的文件，可以分别呈现在系统目录树的不同层次或不同子树中，很容易地赋予不同的存取权限。但是，在树形目录中查找一个文件，需要按路径名逐级访问中间结点，增加了磁盘访问次数，这无疑会影响查询速度。目前，大多数操作系统如 UNIX、Linux 和 Windows 系统都采用了树形文件目录。

### 4. 无环图目录结构

树形目录结构能便于实现文件分类，但不便于实现文件共享，为此在树形目录结构的基础上增加了一些指向同一结点的有向边，使整个目录成为一个有向无环图，如图 4.14 所示。

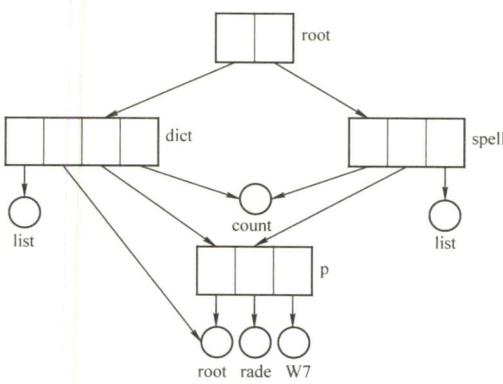


图 4.14 无环图目录结构

当某用户要求删除一个共享结点时，若系统只是简单地将它删除，则当另一共享用户需要访问时，会因无法找到这个文件而发生错误。为此，可为每个共享结点设置一个共享计数器，每当图中增加对该结点的共享链时，计数器加 1；每当某用户提出删除该结点时，计数器减 1。仅当共享计数器为 0 时，才真正删除该结点，否则仅删除请求用户的共享链。

共享文件（或目录）不同于文件拷贝（副本）。

若有两个文件拷贝，则每个程序员看到的是拷贝而不是原件；然而，若一个文件被修改，则另一个程序员的拷贝不会改变。对于共享文件，只存在一个真正的文件，任何改变都会为其他用户所见。

无环图目录结构方便地实现了文件的共享，但使得系统的管理变得更加复杂。

### 4.2.3 目录的操作

在理解一个文件系统的需求前，我们首先考虑在目录这个层次上所需要执行的操作，这有助于后面文件系统的整体理解。

- 搜索。当用户使用一个文件时，需要搜索目录，以找到该文件的对应目录项。
- 创建文件。当创建一个新文件时，需要在目录中增加一个目录项。
- 删除文件。当删除一个文件时，需要在目录中删除相应的目录项。
- 创建目录。在树形目录结构中，用户可创建自己的用户文件目录，并可再创建子目录。
- 删除目录。有两种方式：①不删除非空目录，删除时要先删除目录中的所有文件，并递归地删除子目录。②可删除非空目录，目录中的文件和子目录同时被删除。
- 移动目录。将文件或子目录在不同的父目录之间移动，文件的路径名也会随之改变。
- 显示目录。用户可以请求显示目录的内容，如显示该用户目录中的所有文件及属性。
- 修改目录。某些文件属性保存在目录中，因而这些属性的变化需要改变相应的目录项。

### \*4.2.4 目录实现

在访问一个文件时，操作系统利用路径名找到相应目录项，目录项中提供了查找文件磁盘块所需要的信息。目录实现的基本方法有线性列表和哈希表两种，要注意目录的实现就是为了查找，因此线性列表实现对应线性查找，哈希表的实现对应散列查找。

#### 1. 线性列表

最简单的目录实现方法是，采用文件名和数据块指针的线性列表。当创建新文件时，必须首先搜索目录以确定没有同名的文件存在，然后在目录中增加一个新的目录项。当删除文件时，则根据给定的文件名搜索目录，然后释放分配给它的空间。当要重用目录项时有许多种方法：可以将目录项标记为不再使用，或将它加到空闲目录项的列表上，还可以将目录的最后一个目录项复制到空闲位置，并减少目录的长度。采用链表结构可以减少删除文件的时间。

线性列表的优点在于实现简单，不过由于线性表的特殊性，查找比较费时。

#### 2. 哈希表

除了采用线性列表存储文件目录项，还可以采用哈希数据结构。哈希表根据文件名得到一个值，并返回一个指向线性列表中元素的指针。这种方法的优点是查找非常迅速，插入和删除也较简单，不过需要一些措施来避免冲突（两个文件名称哈希到同一位置）。

目录查询是通过在磁盘上反复搜索完成的，需要不断地进行 I/O 操作，开销较大。所以如前所述，为了减少 I/O 操作，把当前使用的文件目录复制到内存，以后要使用该文件时只需在内存中操作，因此降低了磁盘操作次数，提高了系统速度。

### 4.2.5 文件共享

文件共享使多个用户共享同一个文件，系统中只需保留该文件的一个副本。若系统不能提供共享功能，则每个需要该文件的用户都要有各自的副本，会造成对存储空间的极大浪费。

现代常用的两种文件共享方法如下。

#### 1. 基于索引结点的共享方式（硬链接）

在树形结构的目录中，当有两个或多个用户要共享一个子目录或文件时，必须将共享文件或

子目录链接到两个或多个用户的目录中，才能方便地找到该文件，如图 4.15 所示。

在这种共享方式中，诸如文件的物理地址及其他文件属性等信息，不再放在目录项中，而放在索引结点中。在文件目录中只设置文件名及指向相应索引结点的指针。在索引结点中还应有一个链接计数 count，用于表示链接到本索引结点（即文件）上的用户目录项的数目。当 count = 2 时，表示有两个用户目录项链接到本文件上，或者说有两个用户共享此文件。

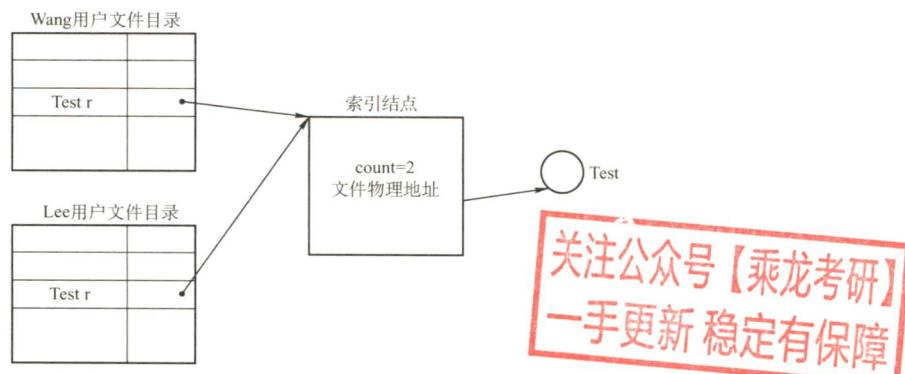


图 4.15 基于索引结点的共享方式

用户 A 创建一个新文件时，他便是该文件的所有者，此时将 count 置为 1。用户 B 要共享此文件时，在 B 的目录中增加一个目录项，并设置一个指针指向该文件的索引结点。此时，文件主仍然是用户 A，count = 2。如果用户 A 不再需要此文件，能否直接将其删除呢？答案是否定的。因为若删除了该文件，也必然删除了该文件的索引结点，这样便会使用户 B 的指针悬空，而 B 可能正在此文件上执行写操作，此时将因此半途而废。因此用户 A 不能删除此文件，只是将该文件的 count 减 1，然后删除自己目录中的相应目录项。用户 B 仍可以使用该文件。当 count = 0 时，表示没有用户使用该文件，才会删除该文件。如图 4.16 给出了用户 B 链接到文件上的前、后情况。

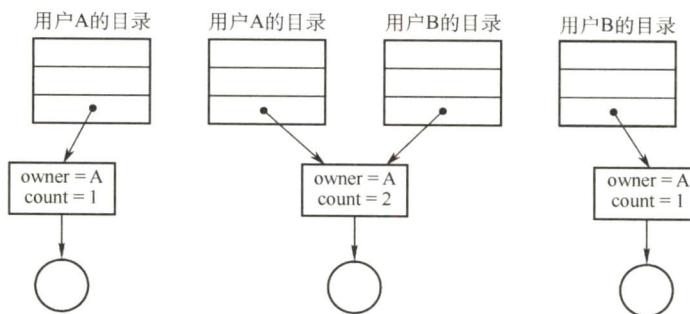


图 4.16 文件共享中的链接计数

## 2. 利用符号链实现文件共享（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，并将该文件写入用户 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。当用户 B 要访问被链接的文件 F 且正要读 LINK 类新文件时，操作系统查看到要读的文件是 LINK 类型，则根据该文件中的路径名去找到文件 F，然后对它进行读，从而实现用户 B 对文件 F 的共享。这样的链接方法被称为符号链接。

在利用符号链方式实现文件共享时，只有文件主才拥有指向其索引结点的指针。而共享该文件的其他用户只有该文件的路径名，并不拥有指向其索引结点的指针。这样，也就不会发生在文件主

删除一共享文件后留下一悬空指针的情况。当文件主把一个共享文件删除后，若其他用户又试图通过符号链去访问它时，则会访问失败，于是将符号链删除，此时不会产生任何影响。

在符号链的共享方式中，当其他用户读共享文件时，系统根据文件路径名逐个查找目录，直至找到该文件的索引结点。因此，每次访问共享文件时，都可能要多次地读盘。使得访问文件的开销甚大，且增加了启动磁盘的频率。此外，符号链的索引结点也要耗费一定的磁盘空间。

利用符号链实现网络文件共享时，只需提供该文件所在机器的网络地址及文件路径名。

硬链接和软链接都是文件系统中的静态共享方法，在文件系统中还存在着另外的共享需求，即两个进程同时对同一个文件进行操作，这样的共享称为动态共享。

可以说：文件共享，“软”“硬”兼施。硬链接就是多个指针指向一个索引结点，保证只要还有一个指针指向索引结点，索引结点就不能删除；软链接就是把到达共享文件的路径记录下来，当要访问文件时，根据路径寻找文件。可见，硬链接的查找速度要比软链接的快。

## 4.2.6 本节小结

本节开头提出的问题的参考答案如下。

### 1) 目录管理的要求是什么？

①实现“按名存取”，这是目录管理最基本的功能。②提高对目录的检索速度，从而提高对文件的存取速度。③为了方便用户共享文件，目录还需要提供用于控制访问文件的信息。④允许不同用户对不同文件采用相同的名字，以便用户按自己的习惯给文件命名。

### 2) 在目录中查找某个文件可以使用什么方法？

可以采用线性列表法或哈希表法。线性列表把文件名组织成一个线性表，查找时依次与线性表中的每个表项进行比较。若把文件名按序排列，则使用折半查找法可以降低平均的查找时间，但建立新文件时会增加维护线性表的开销。哈希表用文件名通过哈希函数得到一个指向文件的指针，这种方法非常迅速，但要注意避免冲突。

## 4.2.7 本节习题精选

### 一、单项选择题

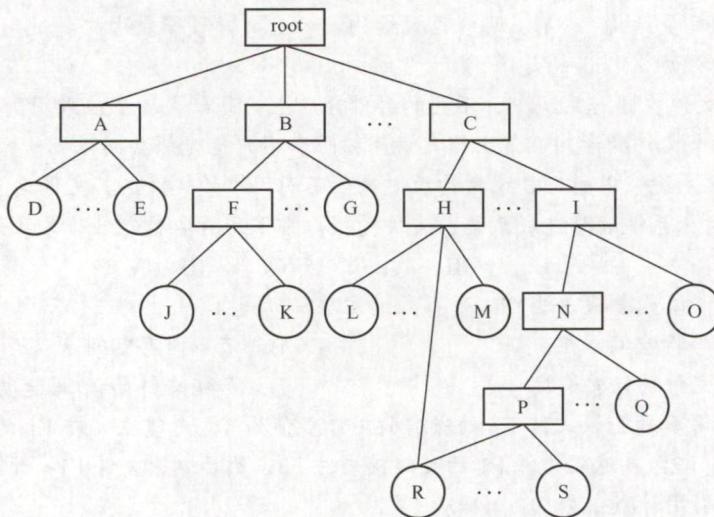
01. 一个文件系统中，其 FCB 占 64B，一个盘块大小为 1KB，采用一级目录。假定文件目录中有 3200 个目录项。则查找一个文件平均需要（ ）次访问磁盘。
  - A. 50
  - B. 54
  - C. 100
  - D. 200
02. 下列关于目录检索的论述中，正确的是（ ）。
  - A. 由于散列法具有较快的检索速度，因此现代操作系统中都用它来替代传统的顺序检索方法
  - B. 在利用顺序检索法时，对树形目录应采用文件的路径名，且应从根目录开始逐级检索
  - C. 在利用顺序检索法时，只要路径名的一个分量名未找到，就应停止查找
  - D. 利用顺序检索法查找完成后，即可得到文件的物理地址
03. 一个文件的相对路径名是从（ ）开始，逐步沿着各级子目录追溯，最后到指定文件的整个通路上所有子目录名组成的一个字符串。
  - A. 当前目录
  - B. 根目录
  - C. 多级目录
  - D. 二级目录
04. 文件系统采用多级目录结构的目的是（ ）。
  - A. 减少系统开销
  - B. 节省存储空间
  - C. 解决命名冲突
  - D. 缩短传送时间
05. 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
06. 下面的说法中，错误的是（ ）。
- 一个文件在同一系统中、不同的存储介质上的复制文件，应采用同一种物理结构
  - 对一个文件的访问，常由用户访问权限和用户优先级共同限制
  - 文件系统采用树形目录结构后，对于不同用户的文件，其文件名应该不同
  - 为防止系统故障造成系统内文件受损，常采用存取控制矩阵方法保护文件
- A. II                  B. I、III                  C. I、III、IV                  D. 全选
07. 【2010 统考真题】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
08. 【2009 统考真题】设文件 F1 的当前引用计数值为 1，先建立文件 F1 的符号链接（软链接）文件 F2，再建立文件 F1 的硬链接文件 F3，然后删除文件 F1。此时，文件 F2 和文件 F3 的引用计数值分别是（ ）。
- A. 0, 1                  B. 1, 1                  C. 1, 2                  D. 2, 1
09. 【2017 统考真题】若文件 f1 的硬链接为 f2，两个进程分别打开 f1 和 f2，获得对应的文件描述符为 fd1 和 fd2，则下列叙述中正确的是（ ）。
- I. f1 和 f2 的读写指针位置保持相同  
II. f1 和 f2 共享同一个内存索引结点  
III. fd1 和 fd2 分别指向各自的用户打开文件表中的一项
- A. 仅 III                  B. 仅 II、III                  C. 仅 I、II                  D. I、II 和 III
10. 【2020 统考真题】若多个进程共享同一个文件 F，则下列叙述中，正确的是（ ）。
- A. 各进程只能用“读”方式打开文件 F  
B. 在系统打开文件表中仅有一个表项包含 F 的属性  
C. 各进程的用户打开文件表中关于 F 的表项内容相同  
D. 进程关闭 F 时，系统删除 F 在系统打开文件表中的表项
11. 【2021 统考真题】若目录 dir 下有文件 file1，则为删除该文件内核不必完成的工作是（ ）。
- A. 删除 file1 的快捷方式                  B. 释放 file1 的文件控制块  
C. 释放 file1 占用的磁盘空间                  D. 删除目录 dir 中与 file1 对应的目录项

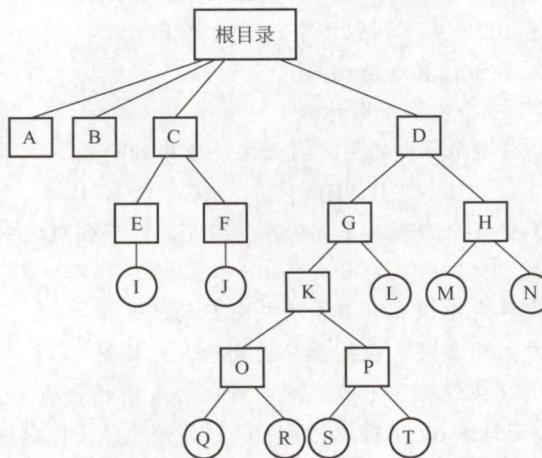
## 二、综合应用题

01. 设某文件系统采用两级目录的结构，主目录中有 10 个子目录，每个子目录中有 10 个目录项。在同样多目录的情况下，若采用单级目录结构，所需平均检索目录项数是两级目录结构平均检索目录项数的多少倍？
02. 对文件的目录结构回答以下问题：
- 若一个共享文件可以被用户随意删除或修改，会有什么问题？
  - 若允许用户随意地读、写和修改目录项，会有什么问题？
  - 如何解决上述问题？
03. 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。
- 可否建立 F 与 R 的链接？试加以说明。
  - 能否删除 R？为什么？
  - 能否删除 N？为什么？

关注公众号【乘龙考研】  
一手更新 稳定有保障



04. 某树形目录结构的文件系统如下图所示。该图中的方框表示目录，圆圈表示文件。



1) 可否进行下列操作?

①在目录 D 中建立一个文件，取名为 A。

②将目录 C 改名为 A。

2) 若 E 和 G 分别为两个用户的目录:

①用户 E 欲共享文件 Q, 应有什么条件? 如何操作?

②在一段时间内用户 G 主要使用文件 S 和 T。为简化操作和提高速度, 应如何处理?

③用户 E 欲对文件 I 加以保护, 不许别人使用, 能否实现? 如何实现?

05. 有一个文件系统如图 A 所示。图中的方框表示目录，圆圈表示普通文件。根目录常驻内存，目录文件组织成链接文件，不设 FCB，普通文件组织成索引文件。目录表指示下一级文件名及其磁盘地址(各占 2B, 共 4B)。下级文件是目录文件时，指示其第一个磁盘块地址。下级文件是普通文件时，指示其 FCB 的磁盘地址。每个目录的文件磁盘块的最后 4B 供拉链使用。下级文件在上级目录文件中的次序在图中为从左至右。每个磁盘块有 512B，与普通文件的一页等长。

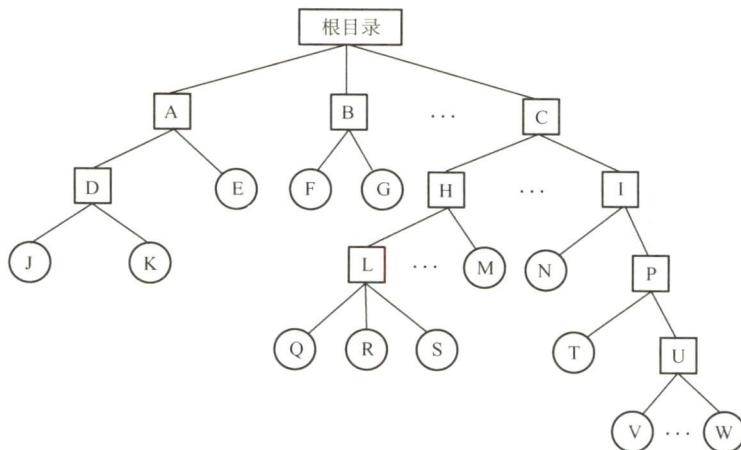


图 A 某树形结构文件系统框图

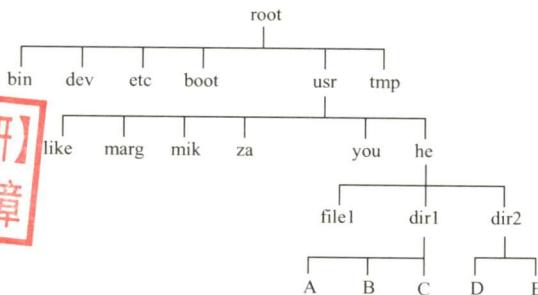
普通文件的 FCB 组织如图 B 所示。其中，每个磁盘地址占 2B，前 10 个地址直接指示该文件前 10 页的地址。第 11 个地址指示一级索引表地址，一级索引表中的每个磁盘地址指示一个文件页地址；第 12 个地址指示二级索引表地址，二级索引表中的每个地址指示一个一级索引表地址；第 13 个地址指示三级索引表地址，三级索引表中的每个地址指示一个二级索引表地址。请问：

- 1) 一个普通文件最多可有多少个文件页？
- 2) 若要读文件 J 中的某一页，最多启动磁盘多少次？
- 3) 若要读文件 W 中的某一页，最少启动磁盘多少次？
- 4) 根据 3)，为最大限度地减少启动磁盘的次数，可采用什么方法？此时，磁盘最多启动多少次？

06. 某个文件系统中，外存为硬盘。物理块大小为 512B，有文件 A 包含 598 条记录，每条记录占 255B，每个物理块放 2 条记录。文件 A 所在的目录如下图所示。文件目录采用多级树形目录结构，由根目录结点、作为目录文件的中间结点和作为信息文件的树叶组成，每个目录项占 127B，每个物理块放 4 个目录项，根目录的第一块常驻内存。试问：

该文件的有关描述信息	
1	磁盘地址
2	磁盘地址
3	磁盘地址
:	...
11	磁盘地址
12	磁盘地址
13	磁盘地址

图 B FCB 组织



- 1) 若文件的物理结构采用链式存储方式，链指针地址占 2B，则要将文件 A 读入内存，至少需要存取几次硬盘？
- 2) 若文件为连续文件，则要读文件 A 的第 487 条记录至少要存取几次硬盘？
- 3) 一般为减少读盘次数，可采取什么措施，此时可减少几次存取操作？

关注公众号【乘龙考研】  
一手更新 稳定有保障

### 4.2.8 答案与解析

#### 一、单项选择题

**01. C**

3200 个目录项占用的盘块数为  $3200 \times 64B / 1KB = 200$  个。因为一级目录平均访盘次数为  $1/2$  盘块数（顺序查找目录表中的所有目录项，每个目录项为一个 FCB），所以平均的访问磁盘次数为  $200/2 = 100$  次。

**02. C**

实现用户对文件的按名存取，系统先利用用户提供的文件名形成检索路径，对目录进行检索。在顺序检索中，路径名的一个分量未找到，说明路径名中的某个目录或文件不存在，不需要继续检索，选项 C 正确。目录的查询方式有两种：顺序检索法和 Hash 法，通常采用顺序检索法，选项 A 错误。在树形目录中，为了加快文件检索速度，可设置当前目录，于是文件路径可以从当前目录开始查找，选项 B 错误。在顺序检索法查找完成后，得到的是文件的逻辑地址，选项 D 错误。

**03. A**

相对路径是从当前目录出发到所找文件通路上所有目录名和数据文件名用分隔符连接起来而形成的，注意与绝对路径的区别。

**04. C**

在文件系统中采用多级目录结构后，符合了多层次管理的需要，提高了文件查找的速度，还允许用户建立同名文件。因此，多级目录结构的采用解决了命名冲突。

**05. A**

在单级目录文件中，每当新建一个文件时，必须先检索所有的目录项，以保证新文件名在目录中是唯一的。所以单级目录结构无法解决文件重名问题。

**06. D**

文件在磁带上通常采用连续存放方法，在硬盘上通常不采用连续存放方法，在内存上采用随机存放方法，I 错误。对文件的访问控制，常由用户访问权限和文件属性共同限制，II 错误。在树形目录结构中，对于不同用户的文件，文件名可以不同也可以相同，III 错误。防止文件受损常采用备份的方法，而存取控制矩阵方法用于多用户之间的存取权限保护，IV 错误。

**07. C**

当一个文件系统含有多级目录时，每访问一个文件，都要使用从树根开始到树叶为止、包括各中间结点名的全路径名。当前目录又称工作目录，进程对各个文件的访问都相对于当前目录进行，而不需要从根目录一层一层地检索，加快了文件的检索速度。选项 A、B 都与相对目录无关；选项 D，文件的读/写速度取决于磁盘的性能。

**08. B**

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 F2 时，F1 和 F2 的引用计数值都为 1。当再建立 F3 时，F1 和 F3 的引用计数值就都变成了 2。当后来删除 F1 时，F3 的引用计数值为  $2 - 1 = 1$ ，F2 的引用计数值不变。

**09. B**

硬链接指通过索引结点进行链接。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。所以选择选项 B。

#### 10. B

多个进程可以同时以“读”或“写”的方式打开文件，操作系统并不保证写操作的互斥性，进程可通过系统调用对文件加锁，保证互斥写（读者-写者问题），选项 A 错误。整个系统只有一个系统打开文件表，同一个文件打开多次只需改变引用计数，选项 B 正确。用户进程的打开文件表关于同一个文件不一定相同，例如读写指针位置不一定相同，选项 C 错误。进程关闭文件时，文件的引用计数减 1，引用计数变为 0 时才删除系统打开文件表中的表项，选项 D 错误。

#### 11. A

删除一个文件时，会根据文件控制块回收相应的磁盘空间，将文件控制块回收，并删除目录中对应的目录项。选项 B、C、D 正确。快捷方式属于文件共享中的软连接，本质上创建的是一个链接文件，其中存放的是访问该文件的路径，删除文件并不会导致文件的快捷方式被删除，正如在 Windows 中删除一个程序后，其快捷方式可能仍然留在桌面上，但是已经无法打开。

## 二、综合应用题

### 01. 【解答】

依题意，文件系统中共有  $10 \times 10 = 100$  个目录，若采用单级目录结构，目录表中有 100 个目录项，在检索一个文件时，平均检索的目录项数 = 目录项/2 = 50。采用两级目录结构时，主目录有 10 个目录项，每个子目录均有 10 个目录项，每级平均检索 5 个目录项，即检索一个文件时平均检索 10 个目录项，所以采用单级目录结构所需检索目录项数是两级目录结构检索目录项数的  $50/10 = 5$  倍。

### 02. 【解答】

- 1) 将有可能导致共享该文件的其他用户无文件可用，或使用了不想使用的文件，导致发生错误。
- 2) 用户可以通过修改目录项来改变对文件的存取权限，从而非法地使用系统的文件；另外，对目录项不负责任的修改会造成管理上的混乱。
- 3) 解决办法是不允许用户直接执行上述操作，而必须通过系统调用来执行这些操作。

### 03. 【解答】

- 1) 可以建立链接。因为 F 是目录而 R 是文件，所以可以建立 R 到 F 的符号链接。除了符号链接，也可以通过硬链接的方式。
- 2) 不一定能删除 R。由于 R 被多个目录共享，能否删除 R 取决于文件系统实现共享的方法。若采用基于索引结点的共享方法，则因删除后存在指针悬空问题而不能删除 R 结点。若采用基于符号共享的方法，则可以删除 R 结点。
- 3) 不一定能删除 N。由于 N 的子目录中存在共享文件 R，而 R 结点本身不一定能被删除，所以 N 也不一定能被删除。

### 04. 【解答】

- 1) ①由于目录 D 中没有已命名为 A 的文件，因此在目录 D 中，可以建立一个名为 A 的文件。②因为在文件系统的根目录下已存在一个名为 A 的目录，所以根目录下的目录 C 不能改名为 A。
- 2) ①用户 E 欲共享文件 Q，需要用户 E 有访问文件 Q 的权限。在访问权限许可的情况下，用户 E 可通过相应路径来访问文件 Q，即用户 E 通过自己的主目录 E 找到其父目录 C，再访问目录 C 的父目录（根目录），然后依次通过目录 D, G, K 和 O 访问文件 Q。若用户

E 的当前目录为 E，则访问路径为 $..../D/G/K/O/Q$ ，其中符号“..”表示父目录，符号“/”用于分隔路径中的各目录名。②用户 G 需要通过依次访问目录 K 和目录 P，才能访问文件 S 和文件 T。为了提高文件访问速度，可在目录 G 下建立两个链接文件，分别链接到文件 S 和文件 T 上。这样用户 G 就可直接访问这两个文件。③用户 E 可以修改文件 I 的存取控制表来对文件 I 加以保护，不让别的用户使用。具体实现方法是：在文件 I 的存取控制表中，只留下用户 E 的访问权限，其他用户对该文件无操作权限，从而达到不让其他用户访问的目的。

### 05. 【解答】

- 1) 因为磁盘块大小为 512B，所以索引块大小也为 512B，每个磁盘地址大小为 2B。因此，一个一级索引表可容纳 256 个磁盘地址。同样，一个二级索引表可容纳 256 个一级索引表地址，一个三级索引表可容纳 256 个二级索引表地址。这样，一个普通文件最多可有的文件页数为  $10 + 256 + 256 \times 256 + 256 \times 256 \times 256 = 16843018$ 。
- 2) 由图可知，目录文件 A 和 D 中的目录项都只有两个，因此这两个目录文件都只占用一个物理块。要读文件 J 中的某一页，先从内存的根目录中找到目录文件 A 的磁盘地址，将其读入内存（已访问磁盘 1 次）。然后从目录 A 中找出目录文件 D 的磁盘地址读入内存（已访问磁盘 2 次）。再从目录 D 中找出文件 J 的 FCB 地址读入内存（已访问磁盘 3 次）。在最坏情况下，该访问页存放在三级索引下，这时候需要一级级地读三级索引块才能得到文件 J 的地址（已访问磁盘 6 次）。最后读入文件 J 中的相应页（共访问磁盘 7 次）。所以，若要读文件 J 中的某一页，最多启动磁盘 7 次。
- 3) 由图可知，目录文件 C 和 U 的目录项较多，可能存放在多个链接在一起的磁盘块中。在最好情况下，所需的目录项都在目录文件的第一个磁盘块中。先从内存的根目录中找到目录文件 C 的磁盘地址并读入内存（已访问磁盘 1 次）。在 C 中找出目录文件 I 的磁盘地址并读入内存（已访问磁盘 2 次）。在 I 中找出目录文件 P 的磁盘地址并读入内存（已访问磁盘 3 次）。从 P 中找到目录文件 U 的磁盘地址并读入内存（已访问磁盘 4 次）。从 U 的第一个磁盘块中找出文件 W 的 FCB 地址并读入内存（已访问磁盘 5 次）。在最好情况下，要访问的页在 FCB 的前 10 个直接块中，按照直接块指示的地址读文件 W 的相应页（已访问磁盘 6 次）。所以，若要读文件 W 中的某页，最少启动磁盘 6 次。
- 4) 为了减少启动磁盘的次数，可以将需要访问的 W 文件挂在根目录的最前面的目录项中。此时，只需读内存中的根目录就可找到 W 的 FCB，将 FCB 读入内存（已访问磁盘 1 次），最差情况下，需要的 W 文件的那个页挂在 FCB 的三级索引下，因此读 3 个索引块需要访问磁盘 3 次（已访问磁盘 4 次）得到该页的物理地址，再去读这个页即可（已访问磁盘 5 次）。此时，磁盘最多启动 5 次。

### 06. 【解答】

- 1) 由于根目录的第一块常驻内存（即 root 所指的/bin, /dev, /etc, /boot 等可直接获得），根目录找到文件 A 需要 5 次读盘。由  $255 \times 2 + 2 = 512$  可知，一个物理块在链式存储结构下可放 2 条记录及下一个物理块地址，而文件 A 共有 598 条记录，因此读取 A 的所有记录所需的读盘次数为  $598/2 = 299$ ，所以将文件 A 读到内存至少需读盘  $299 + 5 = 304$  次。
- 2) 当文件为连续文件时，找到文件 A 同样需要 5 次读盘，且知道文件 A 的地址后通过计算只需一次读盘即可读出第 487 条记录，所以至少需要  $5 + 1 = 6$  次读盘。
- 3) 为减少因查找目录而读盘的次数，可采用索引结点方法。若一个目录项占 16B，则一个盘块可存放  $512/16 = 32$  个目录项，与本题一个盘块仅能存放 4 个目录相比，可使因访问

目录而读盘的次数减少 1/8。对查找文件的记录而言，可用一个或多个盘块来存放该文件的所有盘块号，即用链接索引方法；一个盘块可存放  $512/2-1 = 255$  个盘块号，留下一个地址用来指向下一个存储盘块号（索引块）的磁盘块号。这样，就本题来说，查找目录时需启动 5 次磁盘。文件 A 共有 299 个盘块，则查找文件 A 的某一记录时需两次取得所有盘块号，再需最多启动一次磁盘即可把 A 中的任意一条记录读入内存。所以，查找一条记录最多需要 8 次访盘，而原来的链接方法查找一条记录时，读盘次数为 6~304。

## 4.3 文件系统

在学习本节时，请读者思考以下问题：

- 1) 什么是文件系统？
- 2) 文件系统要完成哪些功能？

本节除了“外存空间管理”，其他都是 2022 年统考大纲的新增考点，这些内容都是比较抽象的、看不见也摸不着的原理，在常用的国内教材（如汤小丹的教材）中都鲜有涉及。如果觉得不太好理解这些内容，建议读者结合王道的最新课程进行学习。

### 4.3.1 文件系统结构

文件系统（File system）提供高效和便捷的磁盘访问，以便允许存储、定位、提取数据。文件系统有两个不同的设计问题：第一个问题是，定义文件系统的用户接口，它涉及定义文件及其属性、所允许的文件操作、如何组织文件的目录结构。第二个问题是，创建算法和数据结构，以便映射逻辑文件系统到物理外存设备。现代操作系统有多种文件系统类型，因此文件系统的层次结构也不尽相同。图 4.17 是一个合理的文件系统层次结构。

#### (1) I/O 控制

包括设备驱动程序和中断处理程序，在内存和磁盘系统之间传输信息。设备驱动程序将输入的命令翻译成底层硬件的特定指令，硬件控制器利用这些指令使 I/O 设备与系统交互。设备驱动程序告诉 I/O 控制器对设备的什么位置采取什么动作。

#### (2) 基本文件系统

向对应的设备驱动程序发送通用命令，以读取和写入磁盘的物理块。每个物理块由磁盘地址标识。该层也管理内存缓冲区，并保存各种文件系统、目录和数据块的缓存。在进行磁盘块传输前，分配合适的缓冲区，并对缓冲区进行管理。管理它们对于系统性能的优化至关重要。

#### (3) 文件组织模块

组织文件及其逻辑块和物理块。文件组织模块可以将逻辑块地址转换成物理块地址，每个文件的逻辑块从 0 到  $N$  编号，它与数据的物理块不匹配，因此需要通过转换来定位。文件组织模块还包括空闲空间管理器，以跟踪未分配的块，根据需求提供给文件组织模块。

#### (4) 逻辑文件系统

用于管理元数据信息。元数据包括文件系统的所有结构，而不包括实际数据（或文件内容）。



图 4.17 合理的文件系统层次结构

逻辑文件系统管理目录结构，以便根据给定文件名称为文件组织模块提供所需要的信息。它通过文件控制块来维护文件结构。逻辑文件系统还负责文件保护。

### 4.3.2 文件系统布局

#### 1. 文件系统在磁盘中的结构

文件系统存放在磁盘上，多数磁盘划分为一个或多个分区，每个分区中有一个独立的文件系统。文件系统可能包括如下信息：启动存储在那里的操作系统的方式、总的块数、空闲块的数量和位置、目录结构以及各个具体文件等。图 4.18 所示为一个可能的文件系统布局。

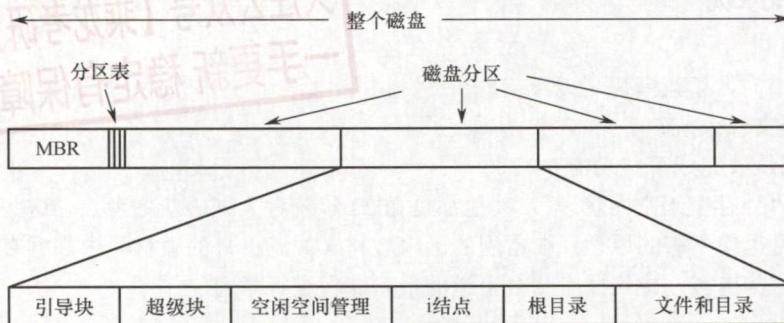


图 4.18 一个可能的文件系统布局

简单描述如下：

- 1) 主引导记录 (Master Boot Record, MBR)，位于磁盘的 0 号扇区，用来引导计算机，MBR 后面是分区表，该表给出每个分区的起始和结束地址。表中的一个分区被标记为活动分区，当计算机启动时，BIOS 读入并执行 MBR。MBR 做的第一件事是确定活动分区，读入它的第一块，即引导块。
- 2) 引导块 (boot block)，MBR 执行引导块中的程序后，该程序负责启动该分区中的操作系统。为统一起见，每个分区都从一个引导块开始，即使它不含有一个可启动的操作系统，也不排除以后会在该分区安装一个操作系统。Windows 系统称之为分区引导扇区。除了从引导块开始，磁盘分区的布局是随着文件系统的不同而变化的。文件系统经常包含有如图 4.18 所列的一些项目。
- 3) 超级块 (super block)，包含文件系统的所有关键信息，在计算机启动时，或者在该文件系统首次使用时，超级块会被读入内存。超级块中的典型信息包括分区的块的数量、块的大小、空闲块的数量和指针、空闲的 FCB 数量和 FCB 指针等。
- 4) 文件系统中空闲块的信息，可以使用位示图或指针链接的形式给出。后面也许跟的是一组 i 结点，每个文件对应一个结点，i 结点说明了文件的方方面面。接着可能是根目录，它存放文件系统目录树的根部。最后，磁盘的其他部分存放了其他所有的目录和文件。

#### 2. 文件系统在内存中的结构

内存中的信息用于管理文件系统并通过缓存来提高性能。这些数据在安装文件系统时被加载，在文件系统操作期间被更新，在卸载时被丢弃。这些结构的类型可能包括：

- 1) 内存中的安装表 (mount table)，包含每个已安装文件系统分区的有关信息。
- 2) 内存中的目录结构的缓存包含最近访问目录的信息。对安装分区的目录，它可以包括一个指向分区表的指针。

- 3) 整个系统的打开文件表，包含每个打开文件的 FCB 副本及其他信息。
- 4) 每个进程的打开文件表，包含一个指向整个系统的打开文件表中的适当条目的指针，以及其他信息。

为了创建新的文件，应用程序调用逻辑文件系统。逻辑文件系统知道目录结构的格式，它将为文件分配一个新的 FCB。然后，系统将相应的目录读入内存，使用新的文件名和 FCB 进行更新，并将它写回磁盘。

一旦文件被创建，它就能用于 I/O。不过，首先要打开文件。系统调用 open() 将文件名传递给逻辑文件系统。调用 open() 首先搜索整个系统的打开文件表，以确定这个文件是否已被其他进程使用。如果已被使用，则在单个进程的打开文件表中创建一个条目，让其指向现有整个系统的打开文件表的相应条目。该算法在文件已打开时，能节省大量开销。如果这个文件尚未打开，则根据给定文件名来搜索目录结构。部分目录结构通常缓存在内存中，以加快目录操作。找到文件后，它的 FCB 会复制到整个系统的打开文件表中。该表不但存储 FCB，而且跟踪打开该文件的进程的数量。然后，在单个进程的打开文件表中创建一个条目，并且通过指针将整个系统打开文件表的条目与其他域（如文件当前位置的指针和文件访问模式等）相连。调用 open() 返回的是一个指向单个进程的打开文件表中的适当条目的指针。以后，所有文件操作都通过该指针执行。一旦文件被打开，内核就不再使用文件名来访问文件，而使用文件描述符（Windows 称之为文件句柄）。

当进程关闭一个文件时，就会删除单个进程打开文件表中的相应条目，整个系统的打开文件表的文件打开数量也会递减。当所有打开某个文件的用户都关闭该文件后，任何更新的元数据将复制到磁盘的目录结构中，并且整个系统的打开文件表的对应条目也会被删除。

### 4.3.3 外存空闲空间管理

一个存储设备可以按整体用于文件系统，也可以细分。例如，一个磁盘可以划分为 4 个分区，每个分区都可以有单独的文件系统。包含文件系统的分区通常称为卷（volume）。卷可以是磁盘的一部分，也可以是整个磁盘，还可以是多个磁盘组成 RAID 集，如图 4.19 所示。

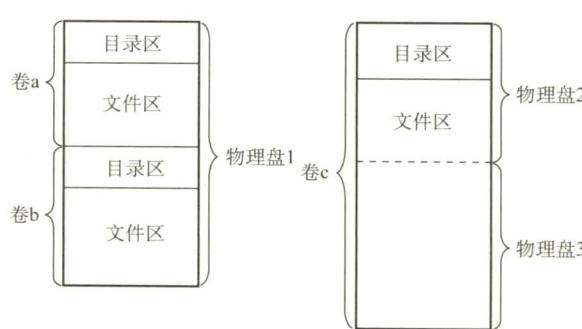


图 4.19 逻辑卷与物理盘的关系

在一个卷中，存放文件数据的空间（文件区）和 FCB 的空间（目录区）是分离的。由于存在很多种类的文件表示和存放格式，所以现代操作系统中一般都有很多不同的文件管理模块，通过它们可以访问不同格式的卷中的文件。卷在提供文件服务前，必须由对应的文件程序进行初始化，划分好目录区和文件区，建立空闲空间管理表格及存放卷信息的超级块。

文件存储设备分成许多大小相同的物理块，并以块为单位交换信息，因此，文件存储设备的管理实质上是对空闲块的组织和管理，它包括空闲块的组织、分配与回收等问题。

### 1. 空闲表法

空闲表法属于连续分配方式，它与内存的动态分配方式类似，为每个文件分配一块连续的存储空间。系统为外存上的所有空闲区建立一张空闲表，

表 4.1 空闲表

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	—	—

每个空闲区对应一个空闲表项，其中包括表项序号、该空闲区的第一个盘块号、该区的空闲盘块数等信息。再将所有空闲区按其起始盘块号递增的次序排列，如表 4.1 所示。

空闲盘区的分配与内存的动态分配类似，同样采用首次适应算法和最佳适应算法等。例如，在系统为某新创建的文件分配空闲盘块时，先顺序地检索空闲盘块表的各项表项，直至找到第一个其大小能满足要求的空闲区，再将该盘区分配给用户，同时修改空闲盘块表。

系统在对用户所释放的存储空间进行回收时，也采取类似于内存回收的方法，即要考虑回收区是否与空闲盘块表中插入点的前区和后区相邻接，对相邻接者应予以合并。

### 2. 空闲链表法

将所有空闲盘区拉成一条空闲链。根据构成链所用基本元素的不同，分为两种形式：

- 1) 空闲盘块链。将磁盘上的所有空闲空间以盘块为单位拉成一条链。当用户因创建文件而请求分配存储空间时，系统从链首开始，依次摘下适当数目的空闲盘块分配给用户。当用户因删除文件而释放存储空间时，系统将回收的盘块依次插入空闲盘块链的末尾。这种方法的优点是分配和回收一个盘块的过程非常简单，但在为一个文件分配盘块时可能要重复操作多次，效率较低。又因它是以盘块为单位的，空闲盘块链会很长。
- 2) 空闲盘区链。将磁盘上的所有空闲盘区（每个盘区可包含若干个盘块）拉成一条链。每个盘区除含有用于指示下一个空闲盘区的指针外，还应有能指明本盘区大小（盘块数）的信息。分配盘区的方法与内存的动态分区分配类似，通常采用首次适应算法。在回收盘区时，同样也要将回收区与相邻接的空闲盘区合并。这种方法的优缺点刚好与第一种方法的相反，即分配与回收的过程比较复杂，但效率通常较高，且空闲盘区链较短。

### 3. 位示图法

位示图是利用二进制的一位来表示磁盘中一个盘块的使用情况，磁盘上所有的盘块都有一个二进制位与之对应。当其值为“0”时，表示对应的盘块空闲；为“1”时，表示已分配。这样，一个  $m \times n$  位组成的位示图就可用来表示  $m \times n$  个盘块的使用情况，如图 4.20 所示。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																

图 4.20 位示图法示意图

注意：本书如无特别提示，所使用的位示图法中行和列都从 1 开始编号。特别注意，若题目中指明从 0 开始编号，则上述计算方法要进行相应调整。

盘块的分配：

- 1) 顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位。

- 2) 将找到的一个或一组二进制位，转换成与之对应的盘块号。若找到的其值为“0”的二进制位位于位示图的第*i*行、第*j*列，则其相应的盘块号应按下式计算(*n*为每行位数)：

$$b = n(i-1) + j$$

- 3) 修改位示图，令  $\text{map}[i, j] = 1$ 。

盘块的回收：

- 1) 将回收盘块的盘块号转换成位示图中的行号和列号。转换公式为：

$$i = (b - 1) \text{ DIV } n + 1$$

$$j = (b - 1) \text{ MOD } n + 1$$

- 2) 修改位示图，令  $\text{map}[i, j] = 0$ 。

空闲表法和空闲链表法都不适用于大型文件系统，因为这会使空闲表或空闲链表太大。

#### 4. 成组链接法

在 UNIX 系统中采用的是成组链接法，这种方法结合了空闲表和空闲链表两种方法，它具有上述两种方法的优点，克服了两种方法均有的表太长的缺点。

用来存放一组空闲盘块号（空闲盘块的块号）的盘块称为成组链块。成组链接法的大致思想是：把顺序的 *n* 个空闲盘块号保存在第一个成组链块中，其最后一个空闲盘块（作为成组链块）则用于保存另一组空闲盘块号，如此继续，直至所有空闲盘块均予以链接。系统只需保存指向第一个成组链块的指针。假设磁盘最初全为空闲盘块，其成组链接如图 4.21 所示。

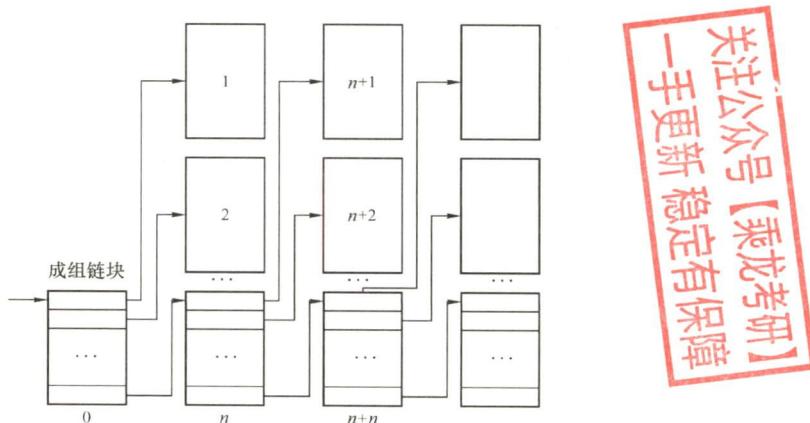


图 4.21 成组链接法示意图

盘块的分配：

根据第一个成组链块的指针，将其对应的盘块分配给用户，然后将指针下移一格。若该指针指向的是最后一个盘块（即成组链块），由于该盘块记录的是下一组空闲盘块号，因此要将该盘块读入内存，并将指针指向新的成组链块的第一条记录，然后执行上述分配操作。

盘块的回收：

成组链块的指针上移一格，再记入回收盘块号。当成组链块的链接数达到 *n* 时，表示已满，便将现有已记录 *n* 个空闲盘块号的成组链块号记入新回收的盘块（作为新的成组链块）。

表示空闲空间的位向量表或第一个成组链块，以及卷中的目录区、文件区划分信息都要存放在磁盘中，一般放在卷头位置，在 UNIX 系统中称为超级块。在对卷中的文件进行操作前，超级块需要预先读入系统空闲的主存，并且经常保持主存超级块与磁盘卷中超级块的一致性。

### 4.3.4 虚拟文件系统

虚拟文件系统（VFS）为用户程序提供了文件系统操作的统一接口，屏蔽了不同文件系统的差异和操作细节，如图 4.22 所示。用户程序可以通过 VFS 提供的统一调用函数（如 `open()` 等）来操作不同文件系统（如 ext3 等）的文件，而无须考虑具体的文件系统和实际的存储介质。

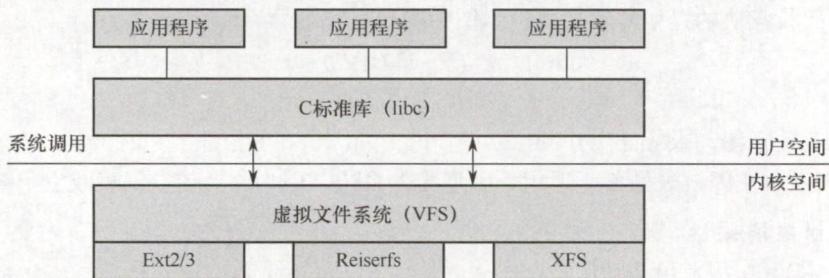


图 4.22 虚拟文件系统的示意图

虚拟文件系统采用了面向对象的思想，它抽象出一个通用的文件系统模型，定义了通用文件系统都支持的接口。新的文件系统只要支持并实现这些接口，即可安装和使用。以 Linux 中调用 `write()` 操作为例，它在 VFS 中通过 `sys_write()` 函数处理，`sys_write()` 找到具体文件系统，将控制权交给该文件系统，最后由具体文件系统与物理介质交互并写入数据，如图 4.23 所示。

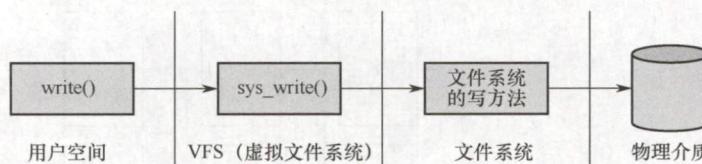


图 4.23 `write()` 系统调用操作示意图

为了实现 VFS，Linux 主要抽象了四种对象类型。每个 VFS 对象都存放在一个适当的数据结构中，其中包括对象的属性和指向对象方法（函数）表的指针。

- 超级块对象：表示一个已安装（或称挂载）的特定文件系统。
- 索引结点对象：表示一个特定的文件。
- 目录项对象：表示一个特定的目录项。
- 文件对象：表示一个与进程相关的已打开文件。

Linux 将目录当作文件对象来处理，文件操作能同时应用于文件或目录。文件系统是由层次目录组成的，一个目录项可能包含文件名和其他目录名。目录项作为单独抽象的对象，是因为目录可以层层嵌套，以便于形成文件路径，而路径中的每一部分其实就是目录项。

- 1) 超级块对象：超级块对象对应于磁盘上特定扇区的文件系统超级块，用于存储已安装文件系统的元信息，元信息中包含文件系统的基本属性信息，如文件系统类型、文件系统基本块的大小、文件系统所挂载的设备、操作方法（函数）指针等。其中操作方法（函数）指针指向该超级块的操作方法表，包含一系列可在超级块对象上调用的操作函数，主要有分配 `inode`、销毁 `inode`、读 `inode`、写 `inode`、文件同步等。
- 2) 索引结点对象。文件系统处理文件所需要的所有信息，都放在一个称为索引结点的数据结构中，索引结点对文件是唯一的。只有当文件被访问时，才在内存中创建索引结点对

象，每个索引结点对象都会复制磁盘索引结点包含的一些数据。该对象中有一个状态字段表示是否被修改，其值为“脏”时，说明对应的磁盘索引结点必须被更新。索引结点对象还提供许多操作接口，如创建新索引结点、创建硬链接、创建新目录等。

- 3) 目录项对象。由于 VFS 经常执行切换到某个目录这种操作，为了提高效率，便引入了目录项的概念。目录项对象是一个路径的组成部分，它要么是目录名，要么是文件名。例如，在查找路径名/test 时，内核为根目录 “/” 创建一个目录项对象，为根目录下的 test 创建一个第二级目录项对象。目录项对象包含指向关联索引结点的指针，还包含指向父目录和指向子目录的指针。不同于前面两个对象，目录项对象在磁盘上没有对应的数据结构，而是 VFS 在遍历路径的过程中，将它们逐个解析成目录项对象的。
- 4) 文件对象。文件对象代表进程打开的一个文件。可以通过 open() 调用打开一个文件，通过 close() 调用关闭一个文件。文件对象和物理文件的关系类似于进程和程序的关系。由于多个进程可以打开和操作同一文件，所以同一文件在内存中可能存在多个对应的文件对象，但对应的索引结点和目录项是唯一的。文件对象仅在进程观点上代表已经打开的文件，它反过来指向其索引结点。文件对象包含与该文件相关联的目录项对象，包含该文件的文件系统、文件指针等，还包含在该文件对象上调用的一系列操作函数。

图 4.24 所示是一个进程与文件进行交互的简单实例。三个不同的进程已打开了同一个文件，其中两个进程使用同一个硬链接。在这种情况下，每个进程都使用自己的文件对象，但只需要两个目录项对象，每个硬链接对应一个目录项对象。这两个目录项对象指向同一个索引结点对象，这个索引结点对象标识的是超级块对象及随后的普通磁盘文件。

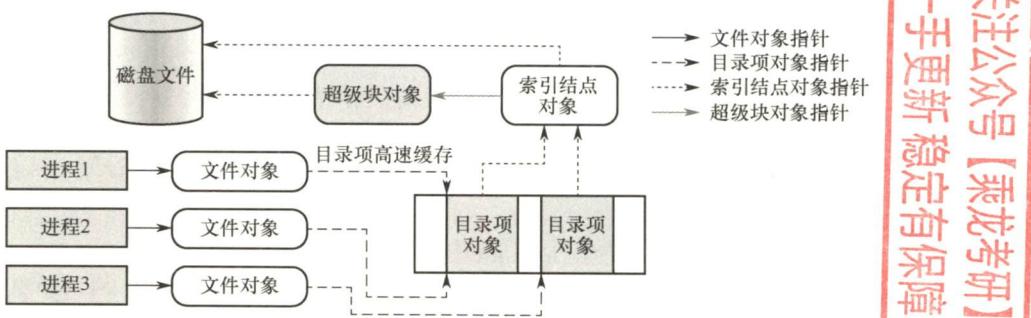


图 4.24 进程与 VFS 对象之间的交互

VFS 还有另一个重要作用，即提高系统性能。最近最常使用的目录项对象被放在目录项高速缓存的磁盘缓存中，以加速从文件路径名到最后一个路径分量的索引结点的转换过程。

对用户来说，不需要关心不同文件系统的具体实现细节，只需要对一个虚拟的文件操作界面进行操作。VFS 对每个文件系统的所有细节进行抽象，使得不同的文件系统在系统中运行的其他进程看来都是相同的。严格来说，VFS 并不是一种实际的文件系统，它只存在于内存中，不存在于任何外存空间中。VFS 在系统启动时建立，在系统关闭时消亡。

### 4.3.5 分区和安装

一个磁盘可以划分为多个分区，每个分区都可以用于创建单独的文件系统，每个分区还可以包含不同的操作系统。分区可以是原始的，没有文件系统，当没有合适的文件系统时，可以使用

关注公众号【乘龙考研】  
一手更新 稳定有保障

原始磁盘。例如，UNIX 交换空间可以使用原始磁盘格式，而不使用文件系统。

第 1 章中介绍过操作系统的引导。Linux 启动后，首先载入 MBR，随后 MBR 识别活动分区，并且加载活动分区中的引导程序。图 4.25 中显示了一个典型的 Linux 分区。

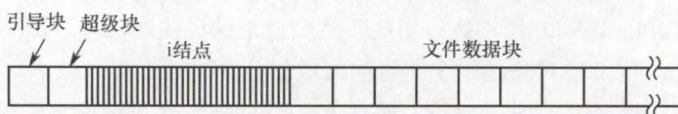


图 4.25 一个典型的 Linux 分区

分区的第一部分是引导块，里面存储着引导信息，它有自身的格式，因为在引导时系统并未加载文件系统代码，因此不能解释文件系统的格式。引导信息是一系列可以加载到内存中的连续块，加载到内存后从其第一条代码开始执行，引导程序便启动一个具体的操作系统。引导块之后是超级块，它存储文件系统的有关信息，包括文件系统的类型、*i* 结点的数目、数据块的数目。随后是多个索引结点，它们是实现文件存储的关键，每个文件对应一个索引结点，索引结点中包含多个指针，指向属于该文件的各个数据块。最后是文件数据块。

如文件在使用前必须打开一样，文件系统在进程使用前必须先安装，也称挂载。

Windows 系统维护一个扩展的两级目录结构，用驱动器字母表示设备和卷。卷具有常规树结构的目录，与驱动器号相关联，还含有指向已安装文件系统的指针。特定文件的路径形式为 *driver-letter:\path\to\file*，操作系统找到相应文件系统的指针，并且遍历该设备的目录结构，以查找指定的文件。新版本的 Windows 允许文件系统安装在目录树下的任意位置，就像 UNIX 一样。在启动时，Windows 操作系统自动发现所有设备，并且安装所有找到的文件系统。

UNIX 使用系统的根文件系统，由内核在引导阶段直接安装，其他文件系统要么由初始化脚本安装，要么由用户安装在已安装文件系统的目录下。作为一个目录树，每个文件系统都拥有自己的根目录。安装文件系统的这个目录称为安装点，安装就是将磁盘分区挂载到该安装点下，进入该目录就可以读取该分区的数据。已安装文件系统属于安装点目录的一个子文件系统。安装的实现是在目录 *inode* 的内存副本上加上一个标志，表示该目录是安装点。还有一个域指向安装表的条目，表示哪个设备安装在哪里，这个条目还包括该设备的文件系统超级块的一个指针。

假定将存放在 /dev/fd0 软盘上的 ext2 文件系统通过 mount 命令安装到 /f1p：

```
mount -t ext2 /dev/fd0 /f1p
```

如需卸载该文件系统，可以使用 umount 命令。

我们可以这么理解：UNIX 本身是一个固定的目录树，只要安装就有，但是如果不能给它分配存储空间，就不能对它进行操作，所以首先要给根目录分配空间，这样才能操作这个目录树。

贯穿本章内容有两条主线：第一条主线是介绍一种新的抽象数据类型——文件，从逻辑结构和物理结构两个方面进行；第二条主线是操作系统是如何管理“文件”的，介绍了多文件的逻辑结构的组织，即目录，还介绍了如何处理用户对文件的服务请求，即磁盘管理。只从宏观上认识是远不够的，从宏观上把握知识的目的是从微观上更加准确地掌控细微知识点，在考试中得到好成绩。读者要通过反复做题、反复思考，不断加深自己对知识点的掌握程度。

### 4.3.6 本节小结

本节开头提出的问题的参考答案如下。

1) 什么是文件系统？

操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统。文件系统由三部分组成：与文件管理有关的软件、被管理文件及实施文件管理所需的数据结构。

2) 文件系统要完成哪些功能?

对于用户而言,文件系统最主要的功能是实现对文件的基本操作,让用户可以按名存储和查找文件,组织成合适的结构,并应当具有基本的文件共享和文件保护功能。对于操作系统本身而言,文件系统还需要管理与磁盘的信息交换,完成文件逻辑结构和物理结构上的变换,组织文件在磁盘上的存放,采取好的文件排放顺序和磁盘调度方法以提升整个系统的性能。

### 4.3.7 本节习题精选

#### 一、单项选择题

关注公众号【乘龙考研】  
一手更新 稳定有保障

01. 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
02. 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
--------	---------	--------	--------
03. 位示图可用于( )。
 

A. 文件目录的查找	B. 磁盘空间的管理
C. 主存空间的管理	D. 文件的保密
04. 文件的存储空间管理实质上是对( )的组织和管理。
 

A. 文件目录	B. 外存已占用区域	C. 外存空闲区	D. 文件控制块
---------	------------	----------	----------
05. 若用 8 个字(字长 32 位)组成的位示图管理内存,假定用户归还一个块号为 100 的内存块时,它对应位示图的位置为( )。
 

A. 字号为 3, 位号为 5	B. 字号为 4, 位号为 4
C. 字号为 3, 位号为 4	D. 字号为 4, 位号为 5
06. 下列选项中,( )不是 Linux 实现虚拟文件系统 VFS 所定义的对象类型。
 

A. 超级块 (superblock) 对象	B. 目录项 (inode) 对象
C. 文件 (file) 对象	D. 数据 (data) 对象
07. 【2015 统考真题】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
----------	----------	----------	----------
08. 【2019 统考真题】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

#### 二、综合应用题

01. 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\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	1	1
2	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0
4	0	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。

**02.** 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?

### 4.3.8 答案与解析

#### 一、单项选择题

**01. B**

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。

**02. B**

UNIX采用树形目录结构, 文件信息存放在索引结点中。超级块是用来描述文件系统的。

**03. B**

位示图方法是空闲块管理方法, 用于管理磁盘空间。

**04. C**

文件存储空间管理即文件空闲空间管理。文件管理要解决的重要问题是, 如何为创建文件分配存储空间, 即如何找到空闲盘块, 并对其进行管理。

**05. B**

先求出块号100所在的行号, 1~32在行号1中, 33~64在行号2中, 65~96在行号3中, 97~128在行号4中, 所以块号100在行号4中; 然后求出块号100在行号4中的哪列, 行号4的第一列是块号97, 以此类推, 块号100在行号4中的第4列。另解, 行号row和列号col分别为

$$\text{row} = (100 - 1) \text{ DIV } 32 + 1 = 4$$

$$\text{col} = (100 - 1) \text{ MOD } 32 + 1 = 4$$

即字号为4, 位号也为4。注意, 如果注明行号和列号从0开始, 那么答案是不同的。

**06. D**

为了实现虚拟文件系统(VFS), Linux主要抽象了四种对象类型: 超级块对象、索引结点对象、目录项对象和文件对象。D错误。

**07. C**

盘块号 = 起始块号 + ⌊盘块号/(1024×8)⌋ = 32 + ⌊409612/(1024×8)⌋ = 32 + 50 = 82, 这里问的是块内字节号而不是位号, 因此还需除以8(1B=8位), 块内字节号 = ⌋(盘块号%(1024×8))/8⌋ = 1。

**08. B**

传统文件系统管理空闲磁盘的方法包括空闲表法、空闲链表法、位示图法和成组链接法, I、III正确。文件分配表(FAT)的表项与物理磁盘块一一对应, 并且可以用一个特殊的数字-1表示文件的最后一块, 用-2表示这个磁盘块是空闲的(当然也可用-3, -4来表示), 因此FAT不仅记录了文件中各个块的先后链接关系, 同时还标记了空闲的磁盘块, 操作系统可以通过FAT对文件

存储空间进行管理, IV 正确。索引结点是操作系统为了实现文件名与文件信息分开而设计的数据结构, 存储了文件描述信息, 索引结点属于文件目录管理部分的内容, II 错误。

## 二、综合应用题

### 01. 【解答】

- 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”。

### 02. 【解答】

- 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  $32 \times (18 - 1) + 16 = 560$ 。

## 4.4 本章疑难点

### 1. 文件的物理分配方式的比较

文件的三种物理分配方式的比较如表 4.2 所示。



表 4.2 文件三种分配方式的比较

	访问第 $n$ 条记录	优 点	缺 点
连续分配	需访问磁盘 1 次	顺序存取时速度快, 文件定长时可根据文件起始地址及记录长度进行随机访问	文件存储要求连续的存储空间, 会产生碎片, 不利于文件的动态扩充
链接分配	需访问磁盘 $n$ 次	可解决外存的碎片问题, 提高外存空间的利用率, 动态增长较方便	只能按照文件的指针链顺序访问, 查找效率低, 指针信息存放消耗外存空间
索引分配	$m$ 级需访问磁盘 $m + 1$ 次	可以随机访问, 文件易于增删	索引表增加存储空间的开销, 索引表的查找策略对文件系统效率影响较大

### 2. 文件打开的过程描述

①检索目录, 要求打开的文件应该是已经创建的文件, 它应登记在文件目录中, 否则会出错。在检索到指定文件后, 就将其磁盘 iNode 复制到活动 iNode 表中。

②把参数 mode 所给出的打开方式与活动 iNode 中在创建文件时所记录的文件访问权限相比, 如果合法, 则此次打开操作成功。

③当打开合法时, 为文件分配用户打开文件表表项和系统打开文件表表项, 并为后者设置初值, 通过指针建立表项与活动 iNode 之间的联系, 再把文件描述符 fd 返回给调用者。

# 第 5 章 输入/输出 (I/O) 管理

关注公众号【乘龙考研】  
一手更新 稳定有保障

【考纲内容】

扫一扫



视频讲解

## (一) I/O 管理基础

设备：设备的基本概念，设备的分类，I/O 接口

I/O 控制方式：轮询方式，中断方式，DMA 方式

I/O 软件层次结构：中断处理程序，驱动程序，设备独立性软件，用户层 I/O 软件

输入/输出应用程序接口：字符设备接口，块设备接口，网络设备接口

阻塞/非阻塞 I/O

## (二) 设备独立软件

缓冲区管理；设备分配与回收；假脱机技术 (SPOOLing)；设备驱动程序接口

## (三) 外存管理

磁盘：磁盘结构，格式化，分区，磁盘调度方法

固态硬盘：读写性能特效，磨损均衡

## 【复习提示】

本章的内容较为分散，重点掌握 I/O 接口、I/O 软件、三种 I/O 控制方式、高速缓存与缓冲区、SPOOLing 技术，磁盘特性和调度算法。本章很多知识点与硬件高度相关，建议与计算机组成原理的对应章节结合复习。已复习过计算机组成原理的读者遇到比较熟悉的内容时也可适当跳过。另外，未复习过计算机组成原理的读者可能会觉得本章的习题较难，但不需要担心。

本章内容在历年统考真题中所占的比重不大，若统考中出现本章的题目，则基本上可以断定一定非常简单，看过相关内容的读者就一定会做，而未看过的读者基本上只能靠“蒙”。考研成功的秘诀是复习要反复多次并全面，偷工减料是要吃亏的，希望读者重视本章的内容。

## 5.1 I/O 管理概述

### 5.1.1 I/O 设备

I/O 设备管理是操作系统设计中最凌乱也最具挑战性的部分。由于它包含了很多领域的不同设备及与设备相关的应用程序，因此很难有一个通用且一致的设计方案。

#### 1. 设备的分类

按信息交换的单位分类，I/O 设备可分为：

- 1) 块设备。信息交换以数据块为单位。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/写任意一块。

2) 字符设备。信息交换以字符为单位。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，并且时常采用中断 I/O 方式。

按传输速率分类，I/O 设备可分为：

- 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，如磁盘机、光盘机等。

## 2. I/O 接口

I/O 接口（设备控制器）位于 CPU 与设备之间，它既要与 CPU 通信，又要与设备通信，还要具有按 CPU 发来的命令去控制设备工作的功能，主要由三部分组成，如图 5.1 所示。

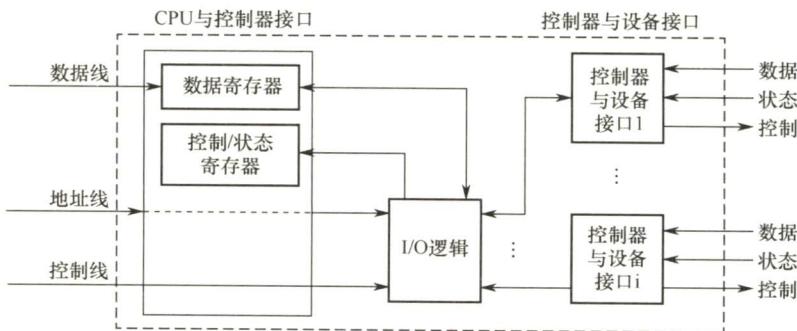


图 5.1 设备控制器的组成

- 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。数据线通常与两类寄存器相连：数据寄存器（存放从设备送来的输入数据或从 CPU 送来的输出数据）和控制/状态寄存器（存放从 CPU 送来的控制信息或设备的状态信息）。
- 2) 设备控制器与设备的接口。一个设备控制器可以连接一个或多个设备，因此控制器中有一个或多个设备接口。每个接口中都存在数据、控制和状态三种类型的信号。
- 3) I/O 逻辑。用于实现对设备的控制。它通过一组控制线与 CPU 交互，对从 CPU 收到的 I/O 命令进行译码。CPU 启动设备时，将启动命令发送给控制器，同时通过地址线把地址发送给控制器，由控制器的 I/O 逻辑对地址进行译码，并相应地对所选设备进行控制。

设备控制器的主要功能有：①接收和识别 CPU 发来的命令，如磁盘控制器能接收读、写、查找等命令；②数据交换，包括设备和控制器之间的数据传输，以及控制器和主存之间的数据传输；③标识和报告设备的状态，以供 CPU 处理；④地址识别；⑤数据缓冲；⑥差错控制。

## 3. I/O 端口

I/O 端口是指设备控制器中可被 CPU 直接访问的寄存器，主要有以下三类寄存器。

- 数据寄存器：实现 CPU 和外设之间的数据缓冲。
- 状态寄存器：获取执行结果和设备的状态信息，以让 CPU 知道是否准备好。
- 控制寄存器：由 CPU 写入，以便启动命令或更改设备模式。

为了实现 CPU 与 I/O 端口进行通信，有两种方法，如图 5.2 所示。

- 1) 独立编址。为每个端口分配一个 I/O 端口号，

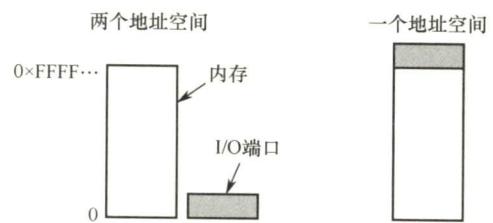


图 5.2 独立编址 I/O 和内存映射 I/O

所有 I/O 端口形成 I/O 端口空间，普通用户程序不能对其进行访问，只有操作系统使用特殊的 I/O 指令才能访问端口。

2) 统一编址。又称内存映射 I/O，每个端口被分配唯一的内存地址，且不会有内存被分配这一地址，通常分配给端口的地址靠近地址空间的顶端。

### 5.1.2 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 CPU 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 5.3(a)所示，计算机从外部设备读取的每个字，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。在该方式中，CPU 之所以要不断地测试 I/O 设备的状态，就是在 CPU 中未采用中断机构，使 I/O 设备无法向 CPU 报告它已完成了第一个字符的输入操作。

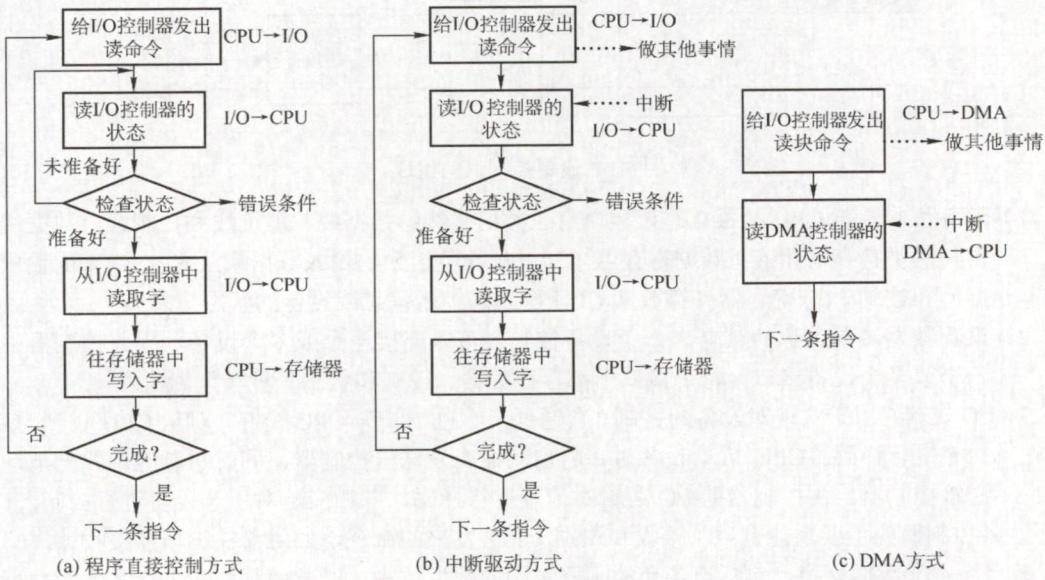


图 5.3 I/O 控制方式

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 5.3(b)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。

从 I/O 控制器的角度来看，I/O 控制器从 CPU 接收一个读命令，然后从外部设备读数据。一旦数据读入 I/O 控制器的数据寄存器，便通过控制线给 CPU 发出中断信号，表示数据已准备好，然后等待 CPU 请求该数据。I/O 控制器收到 CPU 发出的取数据请求后，将数据放到数据总线上，传到 CPU 的寄存器中。至此，本次 I/O 操作完成，I/O 控制器又可开始下一次 I/O 操作。

从 CPU 的角度来看, CPU 发出读命令, 然后保存当前运行程序的上下文(包括程序计数器及处理机寄存器), 转去执行其他程序。在每个指令周期的末尾, CPU 检查中断。当有来自 I/O 控制器的中断时, CPU 保存当前正在运行程序的上下文, 转去执行中断处理程序以处理该中断。这时, CPU 从 I/O 控制器读一个字的数据传送到寄存器, 并存入主存。接着, CPU 恢复发出 I/O 命令的程序(或其他程序)的上下文, 然后继续运行。

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要经过 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:

- 1) 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。

图 5.4 列出了 DMA 控制器的组成。

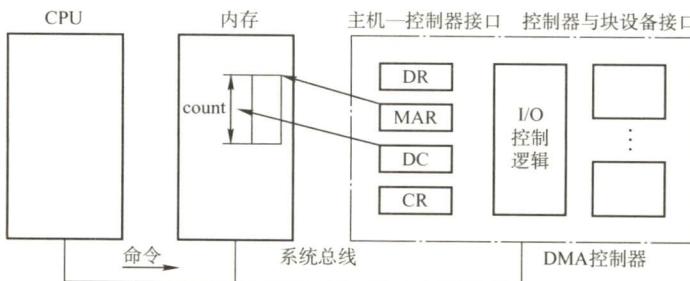


图 5.4 DMA 控制器的组成

关注公众号【乘龙考研】  
一手更新 稳定有保障

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)。接收从 CPU 发来的 I/O 命令、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)。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。

如图 5.3(c)所示, DMA 方式的工作过程是: CPU 接收到 I/O 设备的 DMA 请求时, 它给 DMA 控制器发出一条命令, 同时设置 MAR 和 DC 初值, 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。因此只有在传送开始和结束时才需要 CPU 的参与。

DMA 方式与中断方式的主要区别是, 中断方式在每个数据需要传输时中断 CPU, 而 DMA 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, 中断方式的数据传送是在中断处理时由 CPU 控制完成的, 而 DMA 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。

例如，当 CPU 要完成一组相关的读（或写）操作及有关控制时，只需向 I/O 通道发送一条 I/O 指令，以给出其所要执行的通道程序的首地址和要访问的 I/O 设备，通道接到该指令后，执行通道程序便可完成 CPU 指定的 I/O 任务，数据传送结束时向 CPU 发中断请求。

I/O 通道与一般处理机的区别是：通道指令的类型单一，没有自己的内存，通道所执行的通道程序是放在主机的内存中的，也就是说通道与 CPU 共享内存。

I/O 通道与 DMA 方式的区别是：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。

下面用一个例子来总结这 4 种 I/O 方式。想象一位客户要去裁缝店做一批衣服的情形。

采用程序控制方式时，裁缝没有客户的联系方式，客户必须每隔一段时间去裁缝店看看裁缝把衣服做好了没有，这就浪费了客户不少的时间。采用中断方式时，裁缝有客户的联系方式，每当他完成一件衣服后，给客户打一个电话，让客户去拿，与程序直接控制能省去客户不少麻烦，但每完成一件衣服就让客户去拿一次，仍然比较浪费客户的时间。采用 DMA 方式时，客户花钱雇一位单线秘书，并向秘书交代好把衣服放在哪里（存放仓库），裁缝要联系就直接联系秘书，秘书负责把衣服取回来并放在合适的位置，每处理完 100 件衣服，秘书就要给客户报告一次（大大节省了客户的时间）。采用通道方式时，秘书拥有更高的自主权，与 DMA 方式相比，他可以决定把衣服存放在哪里，而不需要客户操心。而且，何时向客户报告，是处理完 100 件衣服就报告，还是处理完 10000 件衣服才报告，秘书是可以决定的。客户有可能在多个裁缝那里订了货，一位 DMA 类的秘书只能负责与一位裁缝沟通，但通道类秘书却可以与多名裁缝进行沟通。

### 5.1.3 I/O 软件层次结构

I/O 软件涉及的面很宽，往下与硬件有着密切关系，往上又与虚拟存储器系统、文件系统和用户直接交互，它们都需要 I/O 软件来实现 I/O 操作。

为使复杂的 I/O 软件能具有清晰的结构、良好的可移植性和易适应性，目前已普遍采用层次式结构的 I/O 软件。将系统中的设备管理模块分为若干个层次，

每层都是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。在层次式结构的 I/O 软件中，只要层次间的接口不变，对某一层次中的软件的修改都不会引起其下层或高层代码的变更，仅最低层才涉及硬件的具体特性。

图 5.5 I/O 层次结构

一个比较合理的层次划分如图 5.5 所示。整个 I/O 软件可以视为具有 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。一般而言，大部分的 I/O 软件都在操作系统内部，但仍有一小部分在用户层，包括与用户程序链接在一起的库函数。用户层软件必须通过一组系统调用来获取操作系统服务。

#### (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、设备的保护及设备的分配与释放等，

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。为实现设备独立性而引入了逻辑设备和物理设备这两个概念。在应用程序中，使用逻辑设备名来请求使用某类设备；而在系统实际执行时，必须将逻辑设备名映射成物理设备名使用。

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，所谓 I/O 重定向，是指用于 I/O 操作的设备可以更换（即重定向），而不必改变应用程序。

为了实现设备独立性，必须再在驱动程序之上设置一层设备独立性软件。总体而言，设备独立性软件的主要功能可分为以下两个方面：①执行所有设备的公有操作，包括：对设备的分配与回收；将逻辑设备名映射为物理设备名；对设备进行保护，禁止用户直接访问设备；缓冲管理；差错控制；提供独立于设备的大小统一的逻辑块，屏蔽设备之间信息交换单位大小和传输速率的差异。②向用户层（或文件层）提供统一接口。无论何种设备，它们向用户所提供的接口应是相同的。例如，对各种设备的读/写操作，在应用程序中都统一使用 read/write 命令等。

### （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。通常，每类设备配置一个设备驱动程序，它是 I/O 进程与设备控制器之间的通信程序，通常以进程的形式存在。设备驱动程序向上层用户程序提供一组标准接口，设备具体的差别被设备驱动程序所封装，用于接收上层软件发来的抽象 I/O 要求，如 read 和 write 命令，转换为具体要求后，发送给设备控制器，控制 I/O 设备工作；它也将由设备控制器发来的信号传送给上层软件，从而为 I/O 内核系统隐藏设备控制器之间的差异。

### （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，处理完毕再恢复被中断进程的现场后，返回到被中断进程。

中断处理层的主要任务有：进行进程上下文的切换，对处理中断信号源进行测试，读取设备状态和修改进程状态等。由于中断处理与硬件紧密相关，对用户而言，应尽量加以屏蔽，因此应放在操作系统的底层，系统的其余部分尽可能少地与之发生联系。

类似于文件系统的层次结构，I/O 子系统的层次结构也是我们需要记忆的内容，但记忆不是死记硬背，我们以用户对设备的一次命令来总结各层次的功能，帮助各位读者记忆。

例如，①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。②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，然后交往下层。③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。④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。⑤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，完成相应功能。

## 5.1.4 应用程序 I/O 接口

在 I/O 系统与高层之间的接口中，根据设备类型的不同，又进一步分为若干接口。

### （1）字符设备接口

字符设备是指数据的存取和传输是以字符为单位的设备，如键盘、打印机等。基本特征是传输速率较低、不可寻址，并且在输入/输出时通常采用中断驱动方式。

get 和 put 操作。由于字符设备不可寻址，只能采取顺序存取方式，通常为字符设备建立一个字符缓冲区，用户程序通过 get 操作从缓冲区获取字符，通过 put 操作将字符输出到缓冲区。

in-control 指令。字符设备类型繁多，差异甚大，因此在接口中提供一种通用的 in-control 指令来处理它们（包含了许多参数，每个参数表示一个与具体设备相关的特定功能）。

字符设备都属于独占设备，为此接口中还需要提供打开和关闭操作，以实现互斥共享。

#### (2) 块设备接口

块设备是指数据的存取和传输是以数据块为单位的设备，典型的块设备是磁盘。基本特征是传输速率较高、可寻址。磁盘设备的 I/O 常采用 DMA 方式。

隐藏了磁盘的二维结构。在二维结构中，每个扇区的地址需要用磁道号和扇区号来表示。块设备接口将磁盘的所有扇区从 0 到  $n - 1$  依次编号，这样，就将二维结构变为一种线性序列。

将抽象命令映射为低层操作。块设备接口支持上层发来的对文件或设备的打开、读、写和关闭等抽象命令，该接口将上述命令映射为设备能识别的较低层的具体操作。

内存映射接口通过内存的字节数组来访问磁盘，而不提供读/写磁盘操作。映射文件到内存的系统调用返回包含文件副本的一个虚拟内存地址。只在需要访问内存映像时，才由虚拟存储器实际页。内存映射文件的访问如同内存读写一样简单，极大地方便了程序员。

#### (3) 网络设备接口

现代操作系统都提供面向网络的功能，因此还需要提供相应的网络软件和网络通信接口，使计算机能够通过网络与网络上的其他计算机进行通信或上网浏览。

许多操作系统提供的网络 I/O 接口为网络套接字接口，套接字接口的系统调用使应用程序创建的本地套接字连接到远程应用程序创建的套接字，通过此连接发送和接收数据。

#### (4) 阻塞/非阻塞 I/O

操作系统的 I/O 接口还涉及两种模式：阻塞和非阻塞。

阻塞 I/O 是指当用户进程调用 I/O 操作时，进程就被阻塞，需要等待 I/O 操作完成，进程才被唤醒继续执行。非阻塞 I/O 是指用户进程调用 I/O 操作时，不阻塞该进程，该 I/O 调用返回一个错误返回值，通常，进程需要通过轮询的方式来查询 I/O 操作是否完成。

大多数操作系统提供的 I/O 接口都是采用阻塞 I/O。

### 5.1.5 本节小结

本节开头提出的问题的参考答案如下。

I/O 管理要完成哪些功能？

I/O 管理需要完成以下 4 部分内容：

- 1) 状态跟踪。要能实时掌握外部设备的状态。
- 2) 设备存取。要实现对设备的存取操作。
- 3) 设备分配。在多用户环境下，负责设备的分配与回收。
- 4) 设备控制。包括设备的驱动、完成和故障的中断处理。

### 5.1.6 本节习题精选

#### 一、单项选择题

01. 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
02. 虚拟设备是指( )。
- 允许用户使用比系统中具有的物理设备更多的设备
  - 允许用户以标准化方式来使用物理设备
  - 把一个物理设备转换成多个对应的逻辑设备
  - 允许用户程序不必全部装入主存便可使用系统中的设备
03. 磁盘设备的I/O控制主要采取( )方式。
- 位
  - 字节
  - 帧
  - DMA
04. 为了便于上层软件的编制,设备控制器通常需要提供( )。
- 控制寄存器、状态寄存器和控制命令
  - I/O地址寄存器、工作方式状态寄存器和控制命令
  - 中断寄存器、控制寄存器和控制命令
  - 控制寄存器、编程空间和控制逻辑寄存器
05. 在设备控制器中用于实现设备控制功能的是( )。
- CPU
  - 设备控制器与处理器的接口
  - I/O逻辑
  - 设备控制器与设备的接口
06. 在设备管理中,设备映射表(DMT)的作用是( )。
- 管理物理设备
  - 管理逻辑设备
  - 实现输入/输出
  - 建立逻辑设备与物理设备的对应关系
07. DMA方式是在( )之间建立一条直接数据通路。
- I/O设备和主存
  - 两个I/O设备
  - I/O设备和CPU
  - CPU和主存
08. 通道又称I/O处理器,它用于实现( )之间的信息传输。
- 内存与外设
  - CPU与外设
  - 内存与外存
  - CPU与外存
09. 在操作系统中,( )指的是一种硬件机制。
- 通道技术
  - 缓冲池
  - SPOOLing技术
  - 内存覆盖技术
10. 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( )。
- 程序查询
  - 中断方式
  - DMA方式
  - 无条件存取方式
11. 计算机系统中,不属于DMA控制器的是( )。
- 命令/状态寄存器
  - 内存地址寄存器
  - 数据寄存器
  - 堆栈指针寄存器
12. ( )用作连接大量的低速或中速I/O设备。
- 数据选择通道
  - 字节多路通道
  - 数据多路通道
  - I/O处理器
13. 在下列问题中,( )不是设备分配中应考虑的问题。
- 及时性
  - 设备的固有属性
  - 设备独立性
  - 安全性
14. 将系统中的每台设备按某种原则统一进行编号,这些编号作为区分硬件和识别设备的代号,该编号称为设备的( )。
- 绝对号
  - 相对号
  - 型号
  - 符号
15. 关于通道、设备控制器和设备之间的关系,以下叙述中正确的是( )。
- 设备控制器和通道可以分别控制设备

关注公众号【乘龙考研】  
一手更新 稳定有保障

- B. 对于同一组输入/输出命令，设备控制器、通道和设备可以并行工作  
C. 通道控制设备控制器、设备控制器控制设备工作  
D. 以上答案都不对
16. 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
17. 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
18. 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
19. 将系统调用参数翻译成设备操作命令的工作由（ ）完成。  
A. 用户层 I/O                                    B. 设备无关的操作系统软件  
C. 中断处理                                    D. 设备驱动程序
20. 一个典型的文本打印页面有 50 行，每行 80 个字符，假定一台标准的打印机每分钟能打印 6 页，向打印机的输出寄存器中写一个字符的时间很短，可忽略不计。若每打印一个字符都需要花费 50 $\mu$ s 的中断处理时间（包括所有服务），使用中断驱动 I/O 方式运行这台打印机，中断的系统开销占 CPU 的百分比为（ ）。  
A. 2%                    B. 5%                    C. 20%                    D. 50%
21. 【2010 统考真题】本地用户通过键盘登录系统时，首先获得键盘输入信息的程序是（ ）。  
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
22. 【2011 统考真题】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
23. 【2012 统考真题】操作系统的 I/O 子系统通常由 4 个层次组成，每层明确定义了与邻近层次的接口，其合理的层次组织排列顺序是（ ）。  
A. 用户级 I/O 软件、设备无关软件、设备驱动程序、中断处理程序  
B. 用户级 I/O 软件、设备无关软件、中断处理程序、设备驱动程序  
C. 用户级 I/O 软件、设备驱动程序、设备无关软件、中断处理程序  
D. 用户级 I/O 软件、中断处理程序、设备无关软件、设备驱动程序
24. 【2013 统考真题】用户程序发出磁盘 I/O 请求后，系统的处理流程是：用户程序 → 系统调用处理程序 → 设备驱动程序 → 中断处理程序。其中，计算数据所在磁盘的柱面号、磁头号、扇区号的程序是（ ）。

- |           |             |
|-----------|-------------|
| A. 用户程序   | B. 系统调用处理程序 |
| C. 设备驱动程序 | D. 中断处理程序   |
25. 【2017 统考真题】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

关注公众号【乘龙考研】  
一手更新 稳定有保障

## 二、综合应用题

01. DMA 方式与中断控制方式的主要区别是什么？
02. DMA 方式与通道方式的主要区别是什么？
03. 在 32 位 100MHz 的单总线计算机系统中 (10ns 一个周期)，磁盘控制器使用 DMA 以 40MB/s 的速率从存储器中读出数据或向存储器写入数据。假设计算机在没有被周期挪用的情况下，在每个循环周期中读取并执行一个 32 位的指令。这样做，磁盘控制器使指令的执行速度降低了多少？
04. 某计算机系统中，时钟中断处理程序每次执行时间为 2ms (包括进程切换开销)，若时钟中断频率为 60Hz，问 CPU 用于时钟中断处理的时间比率为多少？
05. 考虑 56kb/s 调制解调器的性能，驱动程序输出一个字符后就阻塞，当一个字符打印完毕后，产生一个中断通知阻塞的驱动程序，输出下一个字符，然后阻塞。若发消息、输出一个字符和阻塞的时间总和为 0.1ms，则由于处理调制解调器而占用的 CPU 时间比率是多少？假设每个字符有一个开始位和一个结束位，共占 10 位。

### 5.1.7 答案与解析

#### 一、单项选择题

01. B

可寻址是块设备的基本特征，A 选项不正确；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。分配共享设备是不会引起进程死锁的，D 选项不正确。

02. C

虚拟设备是指采用虚拟技术将一台独占设备转换为若干逻辑设备。引入虚拟设备是为了克服独占设备速度慢、利用率低的特点。这种设备并不是物理地变成共享设备，一般的独享设备也不能转换为共享设备，只是用户在使用它们时“感觉”是共享设备，是逻辑的概念。

03. D

DMA 方式主要用于块设备，磁盘是典型的块设备。这道题也要求读者了解什么是 I/O 控制方式，选项 A、B、C 显然都不是 I/O 控制方式。

04. A

中断寄存器位于计算机主机；不存在 I/O 地址寄存器；编程空间一般是由体系结构和操作系统共同决定的。控制寄存器和状态寄存器分别用于接收上层发来的命令并存放设备状态信号，是设备控制器与上层的接口；至于控制命令，每种设备对应的设备控制器都对应一组相应的控制命

关注公众号【乘龙考研】  
一手更新 稳定有保障

令，CPU 通过控制命令控制设备控制器。

**05. C**

接口用来传输信号，I/O 逻辑即设备控制器，用来实现对设备的控制。

**06. D**

设备映射表中记录了逻辑设备所对应的物理设备，体现了两者的对应关系。对设备映射表来说，不能实现具体的功能及管理物理设备。

**07. A**

DMA 是一种不经过 CPU 而直接从主存存取数据的数据交换模式，它在 I/O 设备和主存之间建立了一条直接数据通路，如磁盘。当然，这条数据通路只是逻辑上的，实际并未直接建立一条物理线路，而通常是通过总线进行的。

**08. A**

设置通道后，CPU 只需向通道发送一条 I/O 指令。通道在收到该指令后，便从内存中取出本次要执行的通道程序，然后执行该通道程序，仅当通道完成规定的 I/O 任务后，才向 CPU 发出中断信号。因此通道用于完成内存与外设的信息交换。

**09. A**

通道是一种特殊的处理器，所以属于硬件技术。SPOOLing、缓冲池、内存覆盖都是在内存的基础上通过软件实现的。

**10. C**

在 DMA 方式中，设备和内存之间可以成批地进行数据交换而不用 CPU 干预，CPU 只参与预处理和结束过程。

**11. D**

命令/状态寄存器控制 DMA 的工作模式并给 CPU 反映它当前的状态，地址寄存器存放 DMA 作业时的源地址和目标地址，数据寄存器存放要 DMA 转移的数据，只有堆栈指针寄存器不需要在 DMA 控制器中存放。

**12. B**

字节多路通道，它通常含有许多非分配型子通道，其数量可达几十到几百个，每个通道连接一台 I/O 设备，并控制该设备的 I/O 操作。这些子通道按时间片轮转方式共享主通道。各个通道循环使用主通道，各个通道每次完成其 I/O 设备的一个字节的交换，然后让出主通道的使用权。这样，只要字节多路通道扫描每个子通道的速率足够快，而连接到子通道上的设备的速率不太高时，便不至于丢失信息。

**13. A**

设备的固有属性决定了设备的使用方式；设备独立性可以提高设备分配的灵活性和设备的利用率；设备安全性可以保证分配设备时不会导致永久阻塞。设备分配时一般不需要考虑及时性。

**14. A**

计算机系统为每台设备确定一个编号以便区分和识别设备，这个确定的编号称为设备的绝对号。

**15. C**

三者的控制关系是层层递进的，只有 C 选项正确。

**16. A**

通道为特殊的处理器，所以不属于软件。其他几项均正确。

**17. C**

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（用来传送

数据)的过程中, CPU与通道并行, 通道完成通道程序的执行时, 便发I/O中断向CPU报告。

**18. C**

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。

**19. B**

系统调用是操作系统提供给用户程序的通用接口, 属于设备无关软件的功能, 不会因为具体设备的不同而改变。而设备驱动程序负责执行操作系统发出的I/O命令, 它因设备不同而不同。

**20. A**

这台打印机每分钟打印  $50 \times 80 \times 6 = 24000$  个字符, 即每秒打印 400 个字符。每个字符打印中断需要占用 CPU 时间  $50\mu s$ , 所以每秒用于中断的系统开销为  $400 \times 50\mu s = 20ms$ 。若使用中断驱动 I/O, 则 CPU 剩余的 980ms 可用于其他处理, 中断的开销占 CPU 的 2%。因此, 使用中断驱动 I/O 方式运行这台打印机是有意义的。

**21. B**

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。

**22. B**

输入/输出软件一般从上到下分为 4 个层次: 用户层、与设备无关的软件层、设备驱动程序及中断处理程序。与设备无关的软件层也就是系统调用的处理程序。

当用户使用设备时, 首先在用户程序中发起一次系统调用, 操作系统的内核接到该调用请求后, 请求调用处理程序进行处理, 再转到相应的设备驱动程序, 当设备准备好或所需数据到达后, 设备硬件发出中断, 将数据按上述调用顺序逆向回传到用户程序中。

**23. A**

考查内容同上题。设备管理软件一般分为 4 个层次: 用户层、与设备无关的系统调用处理层、设备驱动程序及中断处理程序。

**24. C**

计算柱面号、磁头号和扇区号的工作是由设备驱动程序完成的。题中的功能因设备硬件的不同而不同, 因此应由厂家提供的设备驱动程序实现。

**25. B**

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。

## 二、综合应用题

### 01. 【解答】

DMA 控制方式与中断控制方式的主要区别如下:

- 1) 中断控制方式在每个数据传送完成后中断 CPU, 而 DMA 控制方式则在所要求传送的一批数据全部传送结束时中断 CPU。
- 2) 中断控制方式的数据传送在中断处理时由 CPU 控制完成, 而 DMA 控制方式则在 DMA 控制器的控制下完成。不过, 在 DMA 控制方式中, 数据传送的方向、存放数据的内存始址及传送数据的长度等仍然由 CPU 控制。
- 3) DMA 方式以存储器为核心, 中断控制方式以 CPU 为核心。因此 DMA 方式能与 CPU 并

行工作。

- 4) DMA 方式传输批量的数据，中断控制方式的传输则以字节为单位。

#### 02. 【解答】

在 DMA 控制方式中，在 DMA 控制器控制下设备和主存之间可以成批地进行数据交换而不用 CPU 干预，这样既减轻了 CPU 的负担，又大大提高了 I/O 数据传送的速度。通道控制方式与 DMA 控制方式类似，也是一种以内存为中心实现设备与内存直接交换数据的控制方式。不过在通道控制方式中，CPU 只需发出启动指令，指出通道相应的操作和 I/O 设备，该指令就可以启动通道并使通道从内存中调出相应的通道程序执行。与 DMA 控制方式相比，通道控制方式所需的 CPU 干预更少，并且一个通道可以控制多台设备，进一步减轻了 CPU 的负担。另外，对通道来说，可以使用一些指令灵活改变通道程序，这一点 DMA 控制方式无法做到。

#### 03. 【解答】

在 32 位单总线的系统中，磁盘控制器使用 DMA 传输数据的速率为 40MB/s，即每 100ns 传输 4B（32 位）的数据。控制器每读取 10 个指令就挪用一个周期。因此，磁盘控制器使指令的执行速度降低了 10%。

#### 04. 【解答】

时钟中断频率为 60Hz，因此中断周期为 1/60s，每个时钟周期中用于中断处理的时间为 2ms，因此比率为  $0.002/(1/60) = 12\%$ 。

#### 05. 【解答】

因为一个字符占 10 位，因此在 56kb/s 的速率下，每秒传送  $56000/10 = 5600$  个字符，即产生 5600 次中断。每次中断需 0.1ms，因此处理调制解调器占用的 CPU 时间共为  $5600 \times 0.1\text{ms} = 560\text{ms}$ ，占 56% 的 CPU 时间。

## 5.2 设备独立性软件

在学习本节时，请读者思考以下问题：

- 1) 当处理机和外部设备的速度差距较大时，有什么办法可以解决问题？
- 2) 什么是设备的独立性？引入设备的独立性有什么好处？

### 5.2.1 与设备无关的软件

与设备无关的软件是 I/O 系统的最高层软件，它的下层是设备驱动程序，其间的界限因操作系统和设备的不同而有所差异。比如，一些本应由设备独立性软件实现的功能，也可能放在设备驱动程序中实现。这样的差异主要是出于对操作系统、设备独立性软件和设备驱动程序运行效率等多方面因素的权衡。总体而言，设备独立性软件包括执行所有设备公有操作的软件。

### 5.2.2 高速缓存与缓冲区

#### 1. 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，对访问高速缓存要比访问原始磁盘数据更为高效。例如，正在运行进程的数据既存储在磁盘上，又存储在物理内存上，也被复制到 CPU 的二级和一级高速缓存中。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：一种是在内存中开辟一个单独的空间作为磁盘高速缓存，大小固定；另一种是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。

## 2. 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）。

根据系统设置缓冲器的个数，缓冲技术可以分为如下几种：

### (1) 单缓冲

在主存中设置一个缓冲区。当设备和处理机交换数据时，先将数据写入缓冲区，然后需要数据的设备或处理机从缓冲区取走数据，在缓冲区写入或取出的过程中，另一方需等待。

如图 5.6 所示，在块设备输入时，假定从磁盘把一块数据输入到缓冲区的时间为  $T$ ，操作系统将该缓冲区中的数据传送到用户区的时间为  $M$ ，而 CPU 对这一块数据处理的时间为  $C$ 。

在研究每块数据的处理时间时，有一个技巧：假设一种初始状态，然后计算下一次到达相同状态时所需要的时间，就是处理一块数据所需要的时间。在单缓冲中，这种初始状态为：工作区是满的，缓冲区是空的。如题目无明确说明，通常认为缓冲区的大小和工作区的大小相等。

假设  $T > C$ ，从初始状态开始，当工作区数据处理完后，时间为  $C$ ，缓冲区还没充满，当缓冲区充满时，经历了  $T$  时间，停止再冲入数据，然后缓冲区向工作区传送数据，当工作区满了后，缓冲区的数据同时也为空，用时为  $M$ ，到达下一个开始状态，整个过程用时  $M+T$ ；若  $T < C$ ，同理，整个过程用时  $M+C$ 。故单缓冲区处理每块数据的用时为  $\max(C, T) + M$ 。

### (2) 双缓冲

根据单缓冲的特点，CPU 在传送时间  $M$  内处于空闲状态，由此引入双缓冲。I/O 设备输入数据时先装填到缓冲区 1，在缓冲区 1 填满后才开始装填缓冲区 2，与此同时处理机可以从缓冲区 1 中取出数据送入用户进程，当缓冲区 1 中的数据处理完后，若缓冲区 2 已填满，则处理机又从缓冲区 2 中取出数据送入用户进程，而 I/O 设备又可以装填缓冲区 1。注意，必须等缓冲区 2 充满才能让处理机从缓冲区 2 取出数据。双缓冲机制提高了处理机和输入设备的并行程度。

为了研究双缓冲处理一块数据的用时，我们先规定一种初始状态：工作区是空的，其中一个缓冲区是满的，另外一个缓冲区是空的；我们不妨假设缓冲区 1 是空的，缓冲区 2 是满的。

如图 5.7 所示，我们假设  $T < C + M$ ，缓冲区 2 开始向工作区传送数据，缓冲区 1 开始冲入数据，

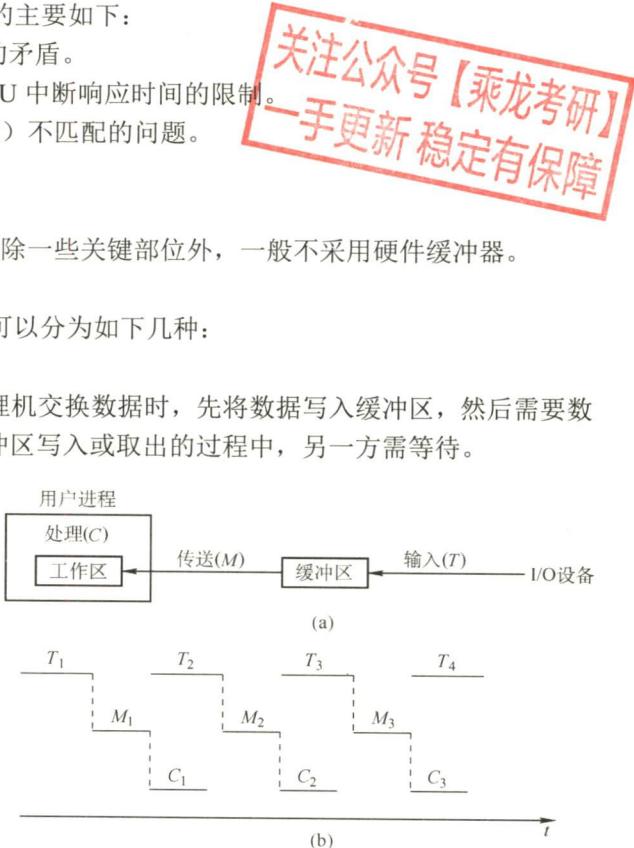
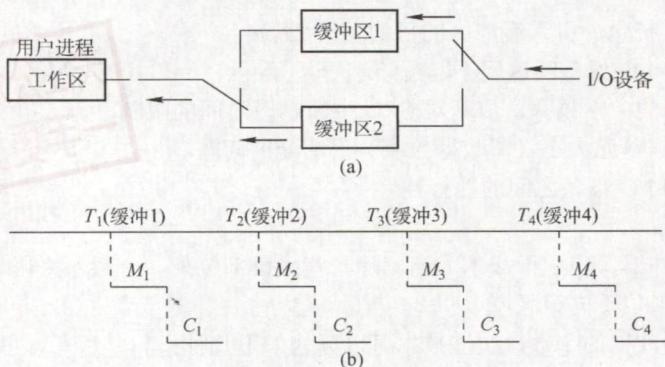


图 5.6 单缓冲工作示意图

当工作区充满数据后，缓冲区为空，时间为  $M$ ，然后工作区开始处理数据，缓冲区 1 继续冲入数据，因为此时只有一个 I/O 设备，所以缓冲区 2 虽然为空，但不能冲入数据。当缓冲区 1 充满数据后，工作区的数据还未处理完毕，时间为  $T$ ，当工作区数据处理完毕后，此时工作区为空，缓冲区 1 满，缓冲区 2 为空，达到下一个初始状态，用时  $C + M$ 。



关注公众号【乘龙考研】  
一手更新稳定有保障

图 5.7 双缓冲工作示意图

我们再来分析  $T > C + M$  的情况。缓冲区 2 开始向工作区传送数据，缓冲区 1 开始冲入数据，当工作区充满数据并处理完后，用时  $C + M$ ，但缓冲区 1 的数据还未充满；当时间为  $T$  时，缓冲区 1 的数据充满，到达下一个初始状态。

总结：双缓冲区处理一块数据的用时为  $\max(C + M, T)$ 。

若  $M + C < T$ ，则可使块设备连续输入；若  $C + M > T$ ，则可使 CPU 不必等待设备输入。对于字符设备，若采用行输入方式，则采用双缓冲可使用户在输入第一行后，在 CPU 执行第一行中的命令的同时，用户可继续向第二缓冲区输入下一行数据。而单缓冲情况下则必须等待一行数据被提取完毕才可输入下一行的数据。

若两台机器之间通信仅配置了单缓冲，如图 5.8(a)所示，则它们在任意时刻都只能实现单方向的数据传输。例如，只允许把数据从 A 机传送到 B 机，或从 B 机传送到 A 机，而绝不允许双方同时向对方发送数据。为了实现双向数据传输，必须在两台机器中都设置两个缓冲区，一个用作发送缓冲区，另一个用作接收缓冲区，如图 5.8(b)所示。

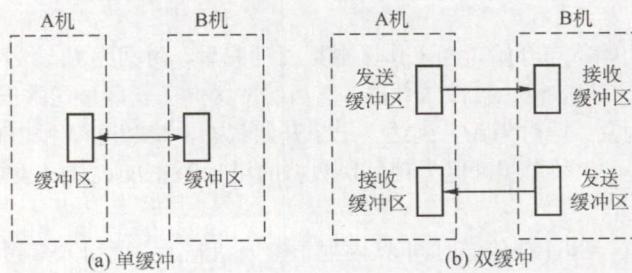


图 5.8 双机通信时缓冲区的设置

### (3) 循环缓冲

包含多个大小相等的缓冲区，每个缓冲区中有一个链接指针指向下一个缓冲区，最后一个缓冲区指针指向第一个缓冲区，多个缓冲区构成一个环形。

循环缓冲用于输入/输出时，还需要有两个指针  $in$  和  $out$ 。对输入而言，首先要从设备接收数据到缓冲区中， $in$  指针指向可以输入数据的第一个空缓冲区；当运行进程需要数据时，从循环缓

冲区中取一个装满数据的缓冲区，并从此缓冲区中提取数据，out 指针指向可以提取数据的第一个满缓冲区。输出则正好相反。

#### (4) 缓冲池

由多个系统公用的缓冲区组成，缓冲区按其使用状况可以形成三个队列：空缓冲队列、装满输入数据的缓冲队列（输入队列）和装满输出数据的缓冲队列（输出队列）。还应具有 4 种缓冲区：用于收容输入数据的工作缓冲区、用于提取输入数据的工作缓冲区、用于收容输出数据的工作缓冲区及用于提取输出数据的工作缓冲区，如图 5.9 所示。

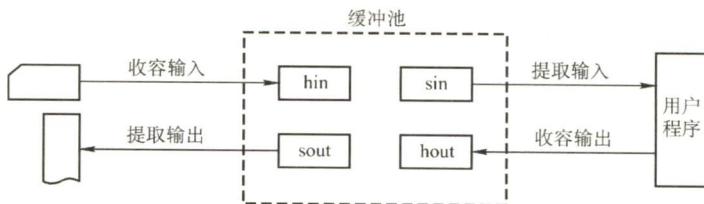


图 5.9 缓冲池的工作方式

当输入进程需要输入数据时，便从空缓冲队列的队首摘下一个空缓冲区，把它作为收容输入工作缓冲区，然后把输入数据输入其中，装满后再将它挂到输入队列队尾。当计算进程需要输入数据时，便从输入队列取得一个缓冲区作为提取输入工作缓冲区，计算进程从中提取数据，数据用完后再将它挂到空缓冲队列尾。当计算进程需要输出数据时，便从空缓冲队列的队首取得一个空缓冲区，作为收容输出工作缓冲区，当其中装满输出数据后，再将它挂到输出队列队尾。当要输出时，由输出进程从输出队列中取得一个装满输出数据的缓冲区，作为提取输出工作缓冲区，当数据提取完后，再将它挂到空缓冲队列的队尾。

对于循环缓冲和缓冲池，我们只是定性地介绍它们的机理，而不去定量研究它们平均处理一块数据所需要的时间。而对于单缓冲和双缓冲，我们只要按照上面的模板分析，就可以解决任何计算单缓冲和双缓冲情况下数据块处理时间的问题，以不变应万变。

### 3. 高速缓存与缓冲区的对比

高速缓存是可以保存数据拷贝的高速存储器，访问高速缓存比访问原始数据更高效，速度更快。高速缓存和缓冲区的对比见表 5.1。

表 5.1 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，即高速缓存上有的，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

### 5.2.3 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。从设备的特性来看，采用下述三种使用方式的设备分别称为独占设备、共享设备和虚拟设备。

- 1) 独占式使用设备。进程分配到独占设备后，便由其独占，直至该进程释放该设备。
- 2) 分时式共享使用设备。对于共享设备，可同时分配给多个进程，通过分时共享使用。
- 3) 以 SPOOLing 方式使用外部设备。SPOOLing 技术实现了虚拟设备功能，可以将设备同时分配给多个进程。这种技术实质上就是实现了对设备的 I/O 操作的批处理。

## 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)：一个设备控制表就代表一个设备，而这个控制表中的表项就是设备的各个属性，如图 5.10 所示。凡因请求本设备而未得到满足的进程，应将其 PCB 按某种策略排成一个设备请求队列，设备队列的队首指针指向该请求队列队首 PCB。

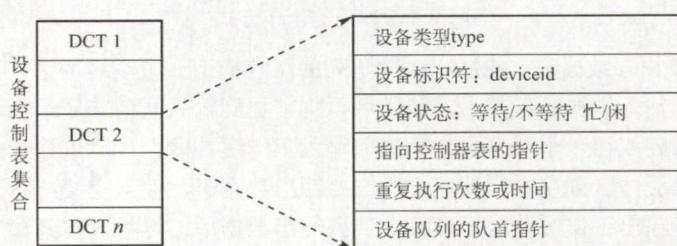
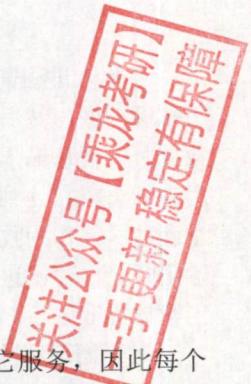


图 5.10 设备控制表 DCT



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 [图 5.11(a)] 有一个表项存放指向相应通道控制表 (CHCT) [图 5.11(b)] 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。

系统设备表 (SDT)：整个系统只有一张 SDT，如图 5.11(c)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。

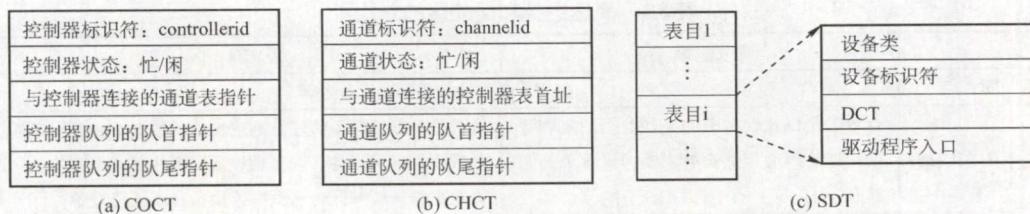


图 5.11 控制器控制表 COCT、通道控制表 CHCT 和系统设备表 SDT

在多道程序系统中，进程数多于资源数，因此要有一套合理的分配原则，主要考虑的因素有：I/O 设备的固有属性、I/O 设备的分配算法、I/O 设备分配的安全性以及 I/O 设备的独立性。

## 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。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来。

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。  
①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。一旦分配，这些设备、控制器就一直为该作业所占用，直到该作业被撤销。静态分配方式不会出现死锁，但设备的使用效率低。  
②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，由系统按某种策略给进程分配所需要的设备、控制器，一旦用完，便立即释放。这种方式有利于提高设备利用率，但若分配算法使用不当，则有可能造成进程死锁。

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。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，而在它阻塞时也不保持任何资源。其优点是设备分配安全，缺点是 CPU 和 I/O 设备是串行工作的。
- 2) 不安全分配方式。进程在发出 I/O 请求后仍继续运行，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。优点是一个进程可同时操作多个设备，使进程推进迅速；缺点是有可能造成死锁。

#### 5. 逻辑设备名到物理设备名的映射

为了提高设备分配的灵活性和设备的利用率，方便实现 I/O 重定向，引入了设备独立性。设备独立性是指应用程序独立于具体使用的物理设备。

为了实现设备独立性，在应用程序中使用逻辑设备名来请求使用某类设备，在系统中设置一张逻辑设备表（Logical Unit Table，LUT），用于将逻辑设备名映射为物理设备名。LUT 表项包括逻辑设备名、物理设备名和设备驱动程序入口地址；当进程用逻辑设备名来请求分配设备时，系统为它分配一台相应的物理设备，并在 LUT 中建立一个表目，当以后进程再利用该逻辑设备名请求 I/O 操作时，系统通过查找 LUT 来寻找对应的物理设备和驱动程序。

在系统中可采取两种方式设置逻辑设备表：

- 1) 在整个系统中只设置一张 LUT。这样，所有进程的设备分配情况都记录在同一张 LUT 中，因此不允许 LUT 中具有相同的逻辑设备名，主要适用于单用户系统。
- 2) 为每个用户设置一张 LUT。每当用户登录时，系统便为该用户建立一个进程，同时也为之建立一张 LUT，并将该表放入进程的 PCB 中。

##### 5.2.4 SPOOLing 技术（假脱机技术）

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，它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。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 系统的组成如图 5.12 所示。

###### (1) 输入井和输出井

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

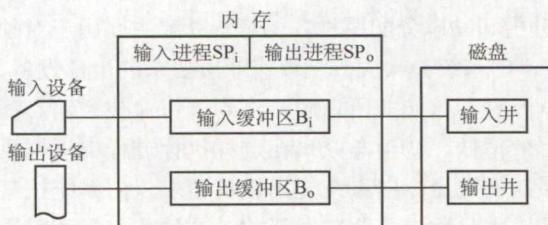


图 5.12 SPOOLing 系统的组成

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。一个进程的输入（或输出）数据保存为一个文件，所有进程的数据输入（或输出）文件链接成一个输入（或输出）队列。

### (2) 输入缓冲区和输出缓冲区

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数据，以后再传送到输出设备。

### (3) 输入进程和输出进程

输入/输出进程用于模拟脱机输入/输出时的外围控制机。用户要求的数据从输入设备经过输入缓冲区送到输入井，当 CPU 需要输入数据时，直接从输入井读入内存。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 技术的实例。当用户进程请求打印输出时，SPOOLing 系统同意打印，但是并不真正立即把打印机分配给该进程，而由假脱机管理进程完成两项任务：

- 1) 在磁盘缓冲区中为之申请一个空闲盘块，并将要打印的数据送入其中暂存。
- 2) 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，再将该表挂到假脱机文件队列上。

这两项工作完成后，虽然还没有任何实际的打印输出，但是对于用户进程而言，其打印任务已完成。对用户而言，系统并非立即执行真实的打印操作，而只是立即将数据输出到缓冲区，真正的打印操作是在打印机空闲且该打印任务已排在等待队列队首时进行的。

SPOOLing 系统的特点如下：①提高了 I/O 的速度，将对低速 I/O 设备执行的 I/O 操作演变为对磁盘缓冲区中数据的存取，如同脱机输入/输出一样，缓和了 CPU 和低速 I/O 设备之间的速度不匹配的矛盾；②将独占设备改造为共享设备，在假脱机打印机系统中，实际上并没有为任何进程分配设备；③实现了虚拟设备功能，对每个进程而言，它们都认为自己独占了一个设备。

前面我们提到过 SPOOLing 技术是一种以空间换时间的技术，我们很容易理解它牺牲了空间，因为它开辟了磁盘上的空间作为输入井和输出井，但它又如何节省时间呢？

从前述内容我们了解到，磁盘是一种高速设备，在与内存交换数据的速度上优于打印机、键盘、鼠标等中低速设备。试想一下，若没有 SPOOLing 技术，CPU 要向打印机输出要打印的数据，打印机的打印速度比较慢，CPU 就必须迁就打印机，在打印机把数据打印完后才能继续做其他的工作，浪费了 CPU 的不少时间。在 SPOOLing 技术下，CPU 要打印机打印的数据可以先输出到磁盘的输出井中（这个过程由假脱机进程控制），然后做其他的事情。若打印机此时被占用，则 SPOOLing 系统就会把这个打印请求挂到等待队列上，待打印机有空时再把数据打印出来。向磁盘输出数据的速度比向打印机输出数据的速度快，因此就节省了时间。

## 5.2.5 设备驱动程序接口

如果每个设备驱动程序与操作系统的接口都不同，那么每次出现一个新设备时，都必须为此修改操作系统。因此，要求每个设备驱动程序与操作系统之间都有着相同或相近的接口。这样会使得添加一个新设备驱动程序变得很容易，同时也便于开发人员编制设备驱动程序。

对于每种设备类型，例如磁盘，操作系统都要定义一组驱动程序必须支持的函数。对磁盘而言，这些函数自然包含读、写、格式化等。驱动程序中通常包含一张表格，这张表格具有针对这

些函数指向驱动程序自身的指针。装载驱动程序时，操作系统记录这个函数指针表的地址，所以当操作系统需要调用一个函数时，它可以通过这张表格发出间接调用。这个函数指针表定义了驱动程序与操作系统其余部分之间的接口。给定类型的所有设备都必须服从这一要求。

与设备无关的软件还要负责将符号化的设备名映射到适当的驱动程序上。例如，在 UNIX 中，设备名/dev/disk0 唯一确定了一个特殊文件的 i 结点，这个 i 结点包含了主设备号（用于定位相应的驱动程序）和次设备号（用来确定要读写的具体设备）。

在 UNIX 和 Windows 中，设备是作为命名对象出现在文件系统中的，因此针对文件的常规保护规则也适用于 I/O 设备。系统管理员可以为每个设备设置适当的访问权限。

## 5.2.6 本节小结

本节开头提出的问题的参考答案如下。

1) 当处理机和外部设备的速度差距较大时，有什么办法可以解决问题？

可采用缓冲技术来缓解 CPU 与外设速度上的矛盾，即在某个地方（一般为主存）设立一片缓冲区，外设与 CPU 的输入/输出都经过缓冲区，这样外设和 CPU 就都不用互相等待。

2) 什么是设备的独立性？引入设备的独立性有什么好处？

设备独立性是指用户在编程序时使用的设备与实际设备无关。一个程序应独立于分配给它的某类设备的具体设备，即在用户程序中只指明 I/O 使用的设备类型即可。

设备独立性有以下优点：① 方便用户编程。② 使程序运行不受具体机器环境的限制。③ 便于程序移植。

## 5.2.7 本节习题精选

### 一、单项选择题

01. 设备的独立性是指( )。
- 设备独立于计算机系统
  - 系统对设备的管理是独立的
  - 用户编程时使用的设备与实际使用的设备无关
  - 每台设备都有一个唯一的编号
02. 引入高速缓冲的主要目的是( )。
- 提高 CPU 的利用率
  - 提高 I/O 设备的利用率
  - 改善 CPU 与 I/O 设备速度不匹配的问题
  - 节省内存
03. 为了使并发进程能有效地进行输入和输出，最好采用( )结构的缓冲技术。
- 缓冲池
  - 循环缓冲
  - 单缓冲
  - 双缓冲
04. 缓冲技术中的缓冲池在( )中。
- 主存
  - 外存
  - ROM
  - 寄存器
05. 设从磁盘将一块数据传送到缓冲区所用的时间为 80μs，将缓冲区中的数据传送到用户区所用的时间为 40μs，CPU 处理一块数据所用的时间为 30μs。若有多块数据需要处理，并采用单缓冲区传送某磁盘数据，则处理一块数据所用的总时间为( )。
- 120μs
  - 110μs
  - 150μs
  - 70μs
06. 某操作系统采用双缓冲区传送磁盘上的数据。设从磁盘将数据传送到缓冲区所用的时间为  $T_1$ ，将缓冲区中的数据传送到用户区所用的时间为  $T_2$ （假设  $T_2$  远小于  $T_1$ ），CPU 处理数据所用的时间为  $T_3$ ，则处理该数据，系统所用的总时间为( )。



- A.  $T_1 + T_2 + T_3$       B.  $\max(T_2, T_3) + T_1$       C.  $\max(T_1, T_3) + T_2$       D.  $\max(T_1, T_3)$
07. 若 I/O 所花费的时间比 CPU 的处理时间短得多，则缓冲区（ ）。
 

A. 最有效	B. 几乎无效
C. 均衡	D. 以上答案都不对
08. 缓冲区管理着重要考虑的问题是（ ）。
 

A. 选择缓冲区的大小	B. 决定缓冲区的数量
C. 实现进程访问缓冲区的同步	D. 限制进程的数量
09. 考虑单用户计算机上的下列 I/O 操作，需要使用缓冲技术的是（ ）。
 

I. 图形用户界面下使用鼠标	II. 多任务操作系统下的磁带驱动器（假设没有设备预分配）
III. 包含用户文件的磁盘驱动器	IV. 使用存储器映射 I/O，直接和总线相连的图形卡

 A. I、III      B. II、IV      C. II、III、IV      D. 全选
10. 以下（ ）不属于设备管理数据结构。
 

A. PCB	B. DCT	C. COCT	D. CHCT
--------	--------	---------	---------
11. 下列（ ）不是设备的分配方式。
 

A. 独享分配	B. 共享分配	C. 虚拟分配	D. 分区分配
---------	---------	---------	---------
12. 下面设备中属于共享设备的是（ ）。
 

A. 打印机	B. 磁带机	C. 磁盘	D. 磁带机和磁盘
--------	--------	-------	-----------
13. 提高单机资源利用率的关键技术是（ ）。
 

A. SPOOLing 技术	B. 虚拟技术
C. 交换技术	D. 多道程序设计技术
14. 虚拟设备是靠（ ）技术来实现的。
 

A. 通道	B. 缓冲	C. SPOOLing	D. 控制器
-------	-------	-------------	--------
15. SPOOLing 技术的主要目的是（ ）。
 

A. 提高 CPU 和设备交换信息的速度	B. 提高独占设备的利用率
C. 减轻用户编程负担	D. 提供主、辅存接口
16. 在采用 SPOOLing 技术的系统中，用户的打印结果首先被送到（ ）。
 

A. 磁盘固定区域	B. 内存固定区域	C. 终端	D. 打印机
-----------	-----------	-------	--------
17. 采用 SPOOLing 技术的计算机系统，外围计算机需要（ ）。
 

A. 一台	B. 多台	C. 至少一台	D. 0 台
-------	-------	---------	--------
18. SPOOLing 系统由（ ）组成。
 

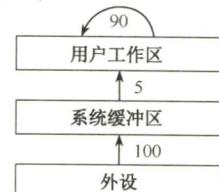
A. 预输入程序、井管理程序和缓输出程序
B. 预输入程序、井管理程序和井管理输出程序
C. 输入程序、井管理程序和输出程序
D. 预输入程序、井管理程序和输出程序
19. 在 SPOOLing 系统中，用户进程实际分配到的是（ ）。
 

A. 用户所要求的外设	B. 外存区，即虚拟设备
C. 设备的一部分存储区	D. 设备的一部分空间
20. 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
21. 下面关于 SPOOLing 的叙述中, 不正确的是( )。  
A. SPOOLing 系统中不需要独占设备  
B. SPOOLing 系统加快了作业执行的速度  
C. SPOOLing 系统使独占设备变成共享设备  
D. SPOOLing 系统提高了独占设备的利用率
22. ( ) 是操作系统中采用的以空间换取时间的技术。  
A. SPOOLing 技术 B. 虚拟存储技术 C. 覆盖与交换技术 D. 通道技术
23. 采用假脱机技术, 将磁盘的一部分作为公共缓冲区以代替打印机, 用户对打印机的操作实际上是对于磁盘的存储操作, 用以代替打印机的部分由( ) 完成。  
A. 独占设备 B. 共享设备 C. 虚拟设备 D. 一般物理设备
24. 下面关于独占设备和共享设备的说法中, 不正确的是( )。  
A. 打印机、扫描仪等属于独占设备  
B. 对独占设备往往采用静态分配方式  
C. 共享设备是指一个作业尚未撤离, 另一个作业即可使用, 但每个时刻只有一个作业使用  
D. 对共享设备往往采用静态分配方式
25. 【2009 统考真题】程序员利用系统调用打开 I/O 设备时, 通常使用的设备标识是( )。  
A. 逻辑设备名 B. 物理设备名 C. 主设备号 D. 从设备号
26. 【2011 统考真题】某文件占 10 个磁盘块, 现要把该文件的磁盘块逐个读入主存缓冲区, 并且送到用户区进行分析, 假设一个缓冲区与一个磁盘块大小相同, 把一个磁盘块读入缓冲区的时间为 100μs, 将缓冲区的数据传送到用户区的时间是 50μs, CPU 对一块数据进行分析的时间为 50μs。在单缓冲区和双缓冲区结构下, 读入并分析完该文件的时间分别是( )。  
A. 1500μs, 1000μs B. 1550μs, 1100μs C. 1550μs, 1550μs D. 2000μs, 2000μs
27. 【2012 统考真题】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
28. 【2013 统考真题】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(如右图所示)。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
29. 【2015 统考真题】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
30. 【2016 统考真题】下列关于 SPOOLing 技术的叙述中, 错误的是( )。

关注公众号【乘龙考研】  
一手更新 稳定有保障



- 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
31. 【2020 统考真题】对于具备设备独立性的系统，下列叙述中，错误的是（ ）。
- A. 可以使用文件名访问物理设备
  - B. 用户程序使用逻辑设备名访问物理设备
  - C. 需要建立逻辑设备与物理设备之间的映射关系
  - D. 更换物理设备后必须修改访问该设备的应用程序
32. 【2022 统考真题】下列关于驱动程序的叙述中，不正确的是（ ）。
- A. 驱动程序与 I/O 控制方式无关
  - B. 初始化设备是由驱动程序控制完成的
  - C. 进程在执行驱动程序时可能进入阻塞态
  - D. 读/写设备的操作是由驱动程序控制完成的

## 二、综合应用题

01. 用于设备分配的数据结构有哪些？它们之间的关系是什么？
02. 输入/输出软件一般分为 4 个层次：用户层、与设备无关的软件层、设备驱动程序和中断处理程序。请说明以下各工作是在哪一层完成的：
- 1) 为磁盘读操作计算磁道、扇区和磁头。
  - 2) 向设备寄存器写命令。
  - 3) 检查用户是否有权使用设备。
  - 4) 将二进制整数转换成 ASCII 码以便打印。
03. 一个串行线能以最大 50000B/s 的速度接收输入。数据平均输入速率是 20000B/s。若用轮询来处理输入，不管是否有输入数据，轮询例程都需要  $3\mu s$  来执行。在下一个字节到达之前未从控制器中取走的字节将丢失。那么最大的安全的轮询时间间隔是多少？
04. 在某系统中，从磁盘将一块数据输入缓冲区需要花费的时间为  $T$ ，CPU 对一块数据进行处理的时间为  $C$ ，将缓冲区的数据传送到用户区所花的时间为  $M$ ，那么在单缓冲和双缓冲情况下，系统处理大量数据时，一块数据的处理时间为多少？
05. 在某系统中，若采用双缓冲区（每个缓冲区可存放一个数据块），将一个数据块从磁盘传送到缓冲区的时间为  $80\mu s$ ，从缓冲区传送到用户的时间为  $20\mu s$ ，CPU 计算一个数据块的时间为  $50\mu s$ 。总共处理 4 个数据块，每个数据块的平均处理时间是多少？
06. 一个 SPOOLing 系统由输入进程 I、用户进程 P、输出进程 O、输入缓冲区、输出缓冲区组成。进程 I 通过输入缓冲区为进程 P 输入数据，进程 P 的处理结果通过输出缓冲区交给进程 O 输出。进程间数据交换以等长度的数据块为单位。这些数据块均存储在同一磁盘上。因此，SPOOLing 系统的数据块通信原语保证始终满足  $i + o \leq max$ ，其中  $max$  为磁盘容量（以该数据块为单位）， $i$  为磁盘上输入数据块的总数， $o$  为磁盘上输出数据块的总数。该 SPOOLing 系统运行时：只要有输入数据，进程 I 终究会将它放入输入缓冲区；只要输入缓冲区有数据块，进程 P 终究会读入、处理，并产生结果数据，写到输出缓冲区；只要输出缓冲区有数据块，进程 O 终究会输出它。  
请说明该 SPOOLing 系统在什么情况下死锁。请说明如何修正约束条件以避免死锁，同时仍允许输入数据块和输出数据块均存储在同一个磁盘上。

## 5.2.8 答案与解析

### 一、单项选择题

关注公众号【乘龙考研】  
一手更新 稳定有保障

01. 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。

02. C

CPU与I/O设备执行速度通常是不对等的，前者快、后者慢，通过高速缓冲技术来改善两者不匹配的问题。

03. A

缓冲池是系统的共用资源，可供多个进程共享，并且既能用于输入又能用于输出。其一般包含三种类型的缓冲：①空闲缓冲区；②装满输入数据的缓冲区；③装满输出数据的缓冲区。为了管理上的方便，可将相同类型的缓冲区链成一个队列。B、C、D属专用缓冲。

04. A

输入井和输出井是在磁盘上开辟的存储空间，而输入/输出缓冲区则是在内存中开辟的，因为CPU速度比I/O设备高很多，缓冲池通常在主存中建立。

05. A

采用单缓冲区传送数据时，设备与处理机对缓冲区的操作是串行的，当进行第*i*次读磁盘数据送至缓冲区时，系统再同时读出用户区中第*i*-1次数据进行计算，此两项操作可以并行，并与数据从缓冲区传送到用户区的操作串行进行，所以系统处理一块数据所用的总时间为  $\max(80\mu s, 30\mu s) + 40\mu s = 120\mu s$ 。

06. D

处理该数据所用的总时间，即可以默认初始状态缓冲区1已将数据传送到用户区。然后分情况讨论：若  $T_3 > T_1$ ，即CPU处理数据块比数据传送慢，意味着I/O设备可连续输入，磁盘将数据传送到缓冲区，再传送到用户区，与CPU处理数据可视为并行处理，时间的花费取决于CPU最大花费时间，则系统所用总时间为  $T_3$ 。若  $T_3 < T_1$ ，即CPU处理数据比数据传送快，此时CPU不必等待I/O设备，磁盘将数据传送到缓冲区，与缓冲区中数据传送到用户区及CPU数据处理可视为并行执行，则花费时间取决于磁盘将数据传送到缓冲区所用时间  $T_1$ 。所以选D。

07. B

缓冲区主要解决输入/输出速度比CPU处理的速度慢而造成数据积压的矛盾。所以当I/O花费的时间比CPU处理时间短很多时，缓冲区没有必要设置。

08. C

在缓冲机制中，无论是单缓冲、多缓冲还是缓冲池，由于缓冲区是一种临界资源，所以在使用缓冲区时都有一个申请和释放（即互斥）的问题需要考虑。

09. D

在鼠标移动时，若有高优先级的操作产生，为了记录鼠标活动的情况，必须使用缓冲技术，I正确。由于磁盘驱动器和目标或源I/O设备间的吞吐量不同，必须采用缓冲技术，II正确。为了能使数据从用户作业空间传送到磁盘或从磁盘传送到用户作业空间，必须采用缓冲技术，III正确。为了便于多幅图形的存取及提高性能，缓冲技术是可以采用的，特别是在显示当前一幅图形又要得到下一幅图形时，应采用双缓冲技术，IV正确。

综上所述，本题正确答案为选项D。

10. A

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。

**11. D**

设备的分配方式主要有独享分享、共享分配和虚拟分配，选项 D 是内存的分配方式。

**12. C**

共享设备是指在一个时间间隔内可被多个进程同时访问的设备，只有磁盘满足。打印机在一个时间间隔内被多个进程访问时打印出来的文档就会乱；磁带机旋转到所需的读写位置需要较长时间，若一个时间间隔内被多个进程访问，磁带机就只能一直旋转，没时间读写。

**13. D**

在单机系统中，最关键的资源是处理器资源，最大化地提高处理器利用率，就是最大化地提高系统效率。多道程序设计技术是提高处理器利用率的关键技术，其他均为设备和内存的相关技术。

**14. C**

SPOOLing 技术是操作系统中采用的一种将独占设备改造为共享设备的技术。通过这种技术处理后的设备通常称为虚拟设备。

**15. B**

SPOOLing 技术可将独占设备改造为共享设备，其主要目的是提高系统资源/独占设备的利用率。

**16. A**

输入井和输出井是在磁盘上开辟的两大存储空间。输入井模拟脱机输入时的磁盘设备，用于暂存 I/O 设备输入的数据；输出井模拟脱机输出时的磁盘，用于暂存用户程序的输出数据。为了缓和 CPU，打印结果首先送到位于磁盘固定区域的输出井。

**17. D**

SPOOLing 技术需要使用磁盘空间（输入井和输出井）和内存空间（输入/输出缓冲区），不需要外围计算机的支持。

**18. A**

SPOOLing 系统主要包含三部分，即输入井和输出井、输入缓冲区和输出缓冲区以及输入进程和输出进程。这三部分由预输入程序、井管理程序和缓输出程序管理，以保证系统正常运行。

**19. B**

通过 SPOOLing 技术可将一台物理 I/O 设备虚拟为 I/O 设备，同样允许多个用户共享一台物理 I/O 设备，所以 SPOOLing 并不是将物理设备真的分配给用户进程。

**20. D**

构成 SPOOLing 系统的基本条件是不仅要有大容量、高速度的外存作为输入井和输出井，而且还要有 SPOOLing 软件，因此 A 错误、B 不够全面，同时利用 SPOOLing 技术提高了系统和 I/O 设备的利用率，进程不必等待 I/O 操作的完成，因此 C 选项也不正确。

**21. A**

因为 SPOOLing 技术是一种典型的虚拟设备技术，它通过将独占设备虚拟成共享设备，使得多个进程共享一个独占设备，从而加快作业的执行速度，提高独占设备的利用率。既然是将独占设备虚拟成共享设备，所以必须先有独占设备才行。

**22. A**

SPOOLing 技术需有高速大容量且可随机存取的外存支持，通过预输入及缓输出来减少 CPU

等待慢速设备的时间，将独享设备改造成共享设备。

**23. C**

打印机是独享设备，利用SPOOLing技术可将打印机改造为可供多个用户共享的虚拟设备。

**24. D**

独占设备采用静态分配方式，而共享设备采用动态分配方式。

**25. A**

用户程序对I/O设备的请求采用逻辑设备名，而程序实际执行时使用物理设备名，它们之间的转换是由设备无关软件层完成的。主设备和从设备是总线仲裁中的概念。

**26. B**

在单缓冲区中，当上一个磁盘块从缓冲区读入用户区完成时，下一磁盘块才能开始读入，也就是当最后一块磁盘块读入用户区完毕时所用的时间为 $150 \times 10 = 1500\mu s$ ，加上处理最后一个磁盘块的时间 $50\mu s$ ，得 $1550\mu s$ 。双缓冲区中，不存在等待磁盘块从缓冲区读入用户区的问题，10个磁盘块可以连续从外存读入主存缓冲区，加上将最后一个磁盘块从缓冲区送到用户区的传输时间 $50\mu s$ 及处理时间 $50\mu s$ ，也就是 $100 \times 10 + 50 + 50 = 1100\mu s$ 。

**27. B**

对于选项A，重排I/O请求次序也就是进行I/O调度，使进程之间公平地共享磁盘访问，减少I/O完成所需要的平均等待时间。对于选项C，缓冲区结合预读和滞后写技术对于具有重复性及阵发性的I/O进程改善磁盘I/O性能很有帮助。对于选项D，优化文件物理块的分布可以减少寻找时间与延迟时间，从而提高磁盘性能。在一个磁盘上设置多个分区与改善设备I/O性能并无多大联系，相反还会带来处理的复杂性，降低利用率。

**28. C**

数据块1从外设到用户工作区的总时间为105，在这段时间中，数据块2未进行操作。在数据块1进行分析处理时，数据块2从外设到用户工作区的总时间为105，这段时间是并行的。再加上数据块2进行处理的时间90，总共是300，答案为选项C。

**29. A**

磁盘和内存的速度差异，决定了可以将内存经常访问的文件调入磁盘缓冲区，从高速缓存中复制的访问比磁盘I/O的机械操作要快很多。

**30. D**

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，选项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，选项B正确。SPOOLing技术实现了将独占设备改造成共享设备的技术，选项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，选项D错误。

**31. D**

设备可视为特殊文件，选项A正确。用户使用逻辑设备名来访问物理文件，有利于设备独立性，选项B正确。通过逻辑设备名访问物理设备时，需要建立逻辑设备和物理设备之间的映射关系，选项C正确。应用程序按逻辑设备名访问设备，再经驱动程序的处理来控制物理设备，若更换物理设备，则只需更换驱动程序，而无须修改应用程序，选项D错误。

**32. A**

厂家在设计一个设备时，通常会为该设备编写驱动程序，主机需要先安装驱动程序，才能使

用设备。当一个设备被连接到主机时，驱动程序负责初始化设备（如将设备控制器中的寄存器初始化），选项 B 正确。当进程在执行驱动程序时，可能会因为设备忙碌而进入阻塞态，选项 C 正确。设备的读/写操作本质就是在设备控制器和主机之间传送数据，而只有厂家知道设备控制器的内部实现，因此也只有厂家提供的驱动程序能控制设备的读/写操作，选项 D 正确。厂家会根据设备特性，在驱动程序中实现一种合适的 I/O 控制方式，选项 A 错误。

## 二、综合应用题

### 01. 【解答】

用于设备分配的数据结构有系统设备表 (SDT)、设备控制表 (DCT)、控制器控制表 (COCT) 和通道控制表 (CHCT)。

SDT 整个系统中只有一张，它记录系统中全部设备的情况，是系统范围的数据结构。每个设备有一张 DCT，系统为每个设备配置一张 DCT，以记录本设备的情况。每个控制器有一张 COCT，系统为每个控制器都设置一张用于记录本控制器情况的 COCT。系统为每个通道配置一张 CHCT，以记录通道情况。SDT 中每个表目有一个指向 DCT 的指针，DCT 中的每个表目有一个指向 COCT 的指针，COCT 中有一个 CHCT 指针，CHCT 中有一个 COCT 指针。

### 02. 【解答】

分析：首先，我们来看这些功能是不是应该由操作系统来完成。操作系统是一个代码相对稳定的软件，它很少发生代码的变化。若 1) 由操作系统完成，则操作系统就必须记录逻辑块和磁盘细节的映射，操作系统的代码会急剧膨胀，而且对新型介质的支持也会引起代码的变动。若 2) 也由操作系统完成，则操作系统需要记录不同生产厂商的不同数据，而且后续新厂商和新产品也无法得到支持。

因为 1) 和 2) 都与具体的磁盘类型有关，因此为了能够让操作系统尽可能多地支持各种不同型号的设备，1) 和 2) 应由厂商所编写的设备驱动程序完成。3) 涉及安全与权限问题，应由与设备无关的操作系统完成。4) 应由用户层来完成，因为只有用户知道将二进制整数转换为 ASCII 码的格式（使用二进制还是十进制、有没有特别的分隔符等）。

### 03. 【解答】

串行线接收数据的最大速度为  $50000\text{B/s}$ ，即每  $20\mu\text{s}$  接收 1B，而轮询例程需  $3\mu\text{s}$  来执行，因此最大的安全轮询时间间隔是  $17\mu\text{s}$ 。

### 04. 【解答】

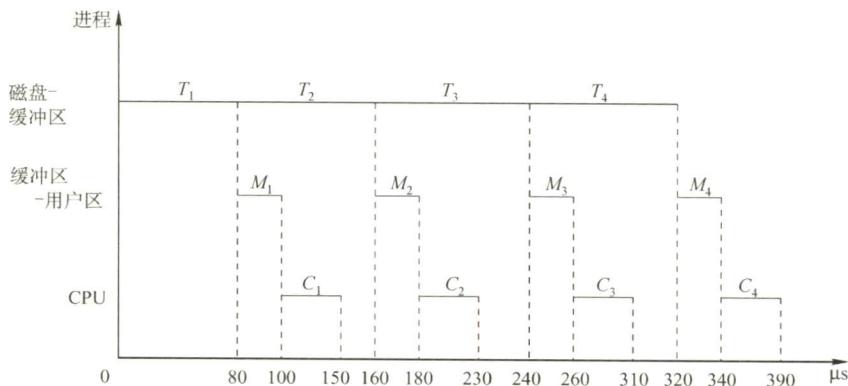
- 1) 在单缓冲的情况下，应先从磁盘把一块数据输入缓冲区，所花费的时间为  $T$ ；然后由操作系统将缓冲区的数据传送到用户区，所花的时间为  $M$ ；接下来便由 CPU 对这一块数据进行计算，计算时间为  $C$ 。由于 CPU 的计算操作与磁盘的数据输入操作可以并行，因此一块数据的处理时间为  $\max(C, T) + M$ 。
- 2) 在双缓冲的情况下，初始时两个缓冲区都是空的。应先从磁盘把一块数据输入第一个缓冲区，当装满第一个缓冲区后，操作系统可将第一个缓冲区的数据传送到用户区并对第一块数据进行计算，与此同时可将磁盘输入数据送入第二个缓冲区；当计算完成后，若第二个缓冲区已装满数据，则又可以将第二个缓冲区中的数据传送至用户区并对第二块数据进行计算，与此同时可将磁盘输入数据送入第一个缓冲区，如此反复交替使用两个缓冲区。CPU 处理一个缓冲区中的数据的耗时为  $C + M$ ，而准备好另一个缓冲区内的数据的耗时为  $T$ 。因此，当  $C + M > T$  时，CPU 刚处理完一个缓冲区的数据，另一个缓冲区的数据就已经准备好了，就可以紧接着处理下一块数据，因此平均来看，每处理一块数据耗时为  $C + M$ 。而当  $C + M < T$  时，CPU 处理完一个缓冲区的数据，另一个缓冲区的数据还没准备好，因此每隔  $T$  的时间，才可以开始处理下一块数据，平均来看处理一

块数据耗时为  $T$ 。综上，采用双缓冲区，处理一块数据平均耗时为  $\max(C + M, T)$ 。

注意：在无缓冲的情况下，为了读取磁盘数据，应先从磁盘把一块数据输入用户数据区，所花费的时间为  $T$ ；然后由 CPU 对一块数据进行计算，计算时间为  $C$ ，所以每块数据的处理时间为  $T + C$ 。

### 05. 【解答】

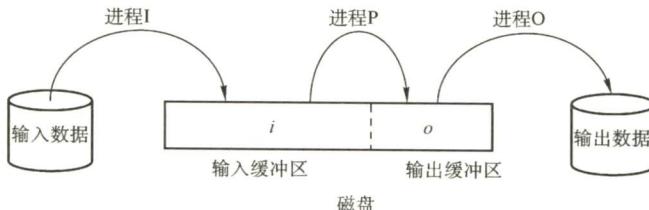
4个数据块的处理过程如下图所示，总耗时  $390\mu s$ ，每个数据块的平均处理时间为  $390\mu s/4 = 97.5\mu s$ 。



从中看到，处理  $n$  个数据块的总耗时为  $(80n + 20 + 50)\mu s = (80n + 70)\mu s$ ，每个数据块的平均处理时间为  $(80n + 70)/n \mu s$ ，当  $n$  较大时，平均时间近似为  $\max(C, T) = 80\mu s$ 。

### 06. 【解答】

此系统的示意图如下图所示。



下面找到一种导致该 SPOOLing 系统死锁的情况：当磁盘上输入数据块总数  $i = \max$  时，磁盘上输出数据块的总数  $o$  必然为 0。此时，进程 I 发现输入缓冲区已满，所以不能再把输入数据放入缓冲区；进程 P 此时有一个处理完的数据，打算把结果数据放入缓冲区，但也发现没有空闲的空间可以放结果数据，因为  $o = 0$ ；所以没有输出数据可以输出，于是进程 O 也无事可做。这时进程 I、P、O 各自都等待着一个事件的发生，若没有外力的作用，它们将一直等待下去，这种僵局显然是死锁。只需将条件修改为  $i + o \leq \max$ ，且  $i \leq \max - 1$ ，就不会再发生死锁。

## 5.3 磁盘和固态硬盘

在学习本节时，请读者思考以下问题：

- 1) 在磁盘上进行一次读写操作需要哪几部分时间？其中哪部分时间最长？
- 2) 存储一个文件时，当一个磁道存储不下时，剩下的部分是存在同一个盘面的不同磁道好，还是存在同一个柱面上的不同盘面好？

本节主要介绍磁盘管理的方式。学习本节时，要重点掌握计算一次磁盘操作的时间，以及对于给定访盘的磁道序列，按照特定算法求出磁头通过的总磁道数及平均寻道数。

### 5.3.1 磁盘

磁盘（Disk）是由表面涂有磁性物质的物理盘片，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 5.13 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，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 5.14 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。扇区是磁盘可寻址的最小单位，磁盘上能存储的物理块数目由扇区数、磁道数及磁盘面数决定，磁盘地址用“柱面号·盘面号·扇区号”表示。

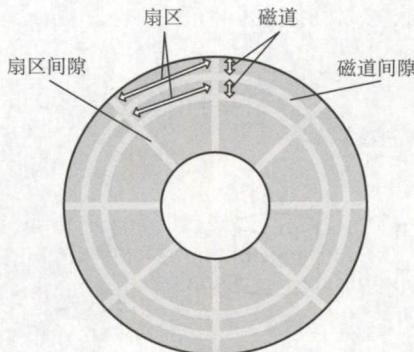


图 5.13 磁盘盘片

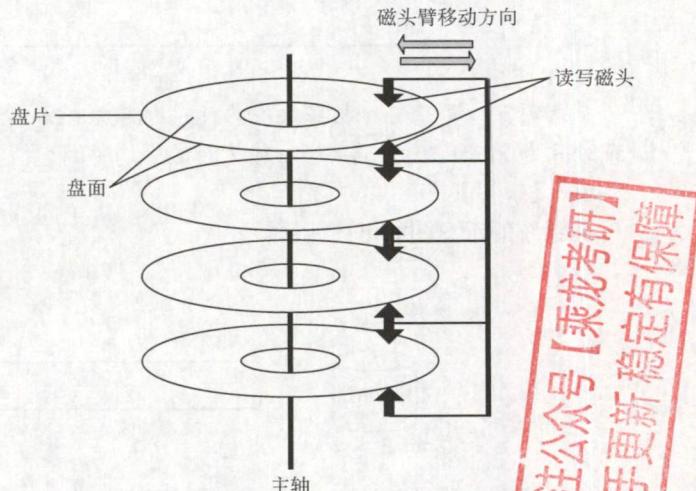


图 5.14 磁盘驱动器

关注公众号【乘龙考研】  
一手更新 稳定有保障

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。

操作系统中几乎每介绍一类资源及其管理时，都要涉及一类调度算法。用户访问文件，需要操作系统的服务，文件实际上存储在磁盘中，操作系统接收用户的命令后，经过一系列的检验访问权限和寻址过程后，最终都会到达磁盘，控制磁盘把相应的数据信息读出或修改。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。

### 5.3.2 磁盘的管理

#### 1. 磁盘初始化

一个新的磁盘只是一个磁性记录材料的空白盘。在磁盘可以存储数据之前，必须将它分成扇

区，以便磁盘控制器能够进行读写操作，这个过程称为低级格式化（或称物理格式化）。低级格式化为每个扇区使用特殊的数据结构，填充磁盘。每个扇区的数据结构通常由头部、数据区域（通常为 512B 大小）和尾部组成。头部和尾部包含了一些磁盘控制器的使用信息。

大多数磁盘在工厂时作为制造过程的一部分就已低级格式化，这种格式化能够让制造商测试磁盘，并且初始化逻辑块号到无损磁盘扇区的映射。对于许多磁盘，当磁盘控制器低级格式化时，还能指定在头部和尾部之间留下多长的数据区，通常选择 256 或 512 字节等。

## 2. 分区

在可以使用磁盘存储文件之前，操作系统还要将自己的数据结构记录到磁盘上，分为两步：第一步是，将磁盘分为由一个或多个柱面组成的分区（即我们熟悉的 C 盘、D 盘等形式的分区），每个分区的起始扇区和大小都记录在磁盘主引导记录的分区表中；第二步是，对物理分区进行逻辑格式化（创建文件系统），操作系统将初始的文件系统数据结构存储到磁盘上，这些数据结构包括空闲空间和已分配的空间以及一个初始为空的目录。

因扇区的单位太小，为了提高效率，操作系统将多个相邻的扇区组合在一起，形成一簇（在 Linux 中称为块）。为了更高效地管理磁盘，一簇只能存放一个文件的内容，文件所占用的空间只能是簇的整数倍；如果文件大小小于一簇（甚至是 0 字节），也要占用一簇的空间。

## 3. 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。为此，自举程序找到磁盘上的操作系统内核，将它加载到内存，并转到起始地址，从而开始操作系统的运行。

自举程序通常存放在 ROM 中，为了避免改变自举代码而需要改变 ROM 硬件的问题，通常只在 ROM 中保留很小的自举装入程序，而将完整功能的引导程序保存在磁盘的启动块上，启动块位于磁盘的固定位置。具有启动分区的磁盘称为启动磁盘或系统磁盘。

引导 ROM 中的代码指示磁盘控制器将引导块读入内存，然后开始执行，它可以从非固定的磁盘位置加载整个操作系统，并且开始运行操作系统。下面以 Windows 为例来分析引导过程。Windows 允许将磁盘分为多个分区，有一个分区为引导分区，它包含操作系统和设备驱动程序。Windows 系统将引导代码存储在磁盘的第 0 号扇区，它称为主引导记录 (MBR)。引导首先运行 ROM 中的代码，这个代码指示系统从 MBR 中读取引导代码。除了包含引导代码，MBR 还包含：一个磁盘分区表和一个标志（以指示从哪个分区引导系统），如图 5.15 所示。当系统找到引导分区时，读取分区的第一个扇区，称为引导扇区，并继续余下的引导过程，包括加载各种系统服务。

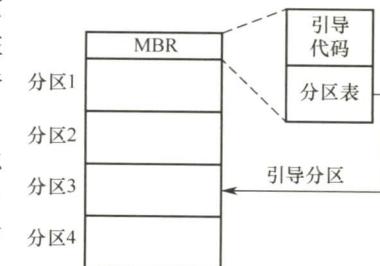


图 5.15 Windows 磁盘的引导

由于磁盘有移动部件且容错能力弱，因此容易导致一个或多个扇区损坏。部分磁盘甚至在出厂时就有坏块。根据所用的磁盘和控制器，对这些块有多种处理方式。

对于简单磁盘，如采用 IDE 控制器的磁盘，坏块可手动处理，如 MS-DOS 的 Format 命令执行逻辑格式化时会扫描磁盘以检查坏块。坏块在 FAT 表上会标明，因此程序不会使用它们。

对于复杂的磁盘，控制器维护磁盘内的坏块列表。这个列表在出厂低级格式化时就已初始化，并在磁盘的使用过程中不断更新。低级格式化将一些块保留作为备用，操作系统看不到这些块。控制器可以采用备用块来逻辑地替代坏块，这种方案称为扇区备用。

对坏块的处理实质上就是用某种机制使系统不去使用坏块。

### 5.3.3 磁盘调度算法

一次磁盘读写操作的时间由寻找（寻道）时间、旋转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

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

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) 旋转延迟时间  $T_r$ 。磁头定位到某一磁道的扇区所需要的时间，设磁盘的旋转速度为  $r$ ，则

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

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：

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

式中， $r$  为磁盘每秒的转数， $N$  为一个磁道上的字节数。

在磁盘存取时间的计算中，寻道时间与磁盘调度算法相关；而延迟时间和传输时间都与磁盘旋转速度相关，且为线性相关，所以在硬件上，转速是磁盘性能的一个非常重要的参数。

总平均存取时间  $T_a$  可以表示为

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

虽然这里给出了总平均存取时间的公式，但是这个平均值是没有太大实际意义的，因为在实际的磁盘 I/O 操作中，存取时间与磁盘调度算法密切相关。

目前常用的磁盘调度算法有以下几种。

(1) 先来先服务 (First Come First Served, 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，如图 5.16 所示。该算法的优点是具有公平性。若只有少量进程需要访问，且大部分请求都是访问簇聚的文件扇区，则有望达到较好的性能；若有大量进程竞争使用磁盘，则这种算法在性能上往往接近于随机调度。所以，实际磁盘调度中会考虑一些更为复杂的调度算法。

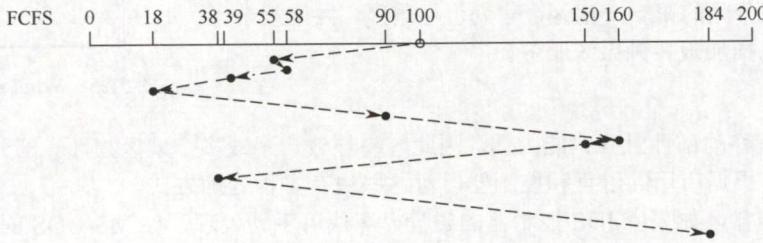


图 5.16 FCFS 磁盘调度算法

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 FCFS 算法时磁头的运动过程如图 5.16 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  $= 498/9 = 55.3$ 。

### (2) 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。这种算法会产生“饥饿”现象。如图 5.17 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 号磁道的访问被无限期地延迟，即被“饿死”。

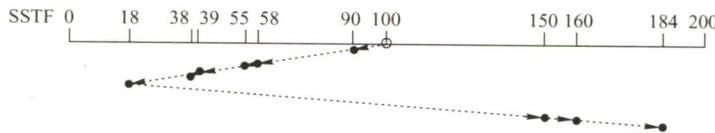


图 5.17 SSTF 磁盘调度算法

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，磁头初始位置是磁道 100，采用 SSTF 算法时磁头的运动过程如图 5.17 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  $= 248/9 = 27.5$ 。

### (3) 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，如图 5.18 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，因此它在访问局部性方面不如 FCFS 算法和 SSTF 算法好。

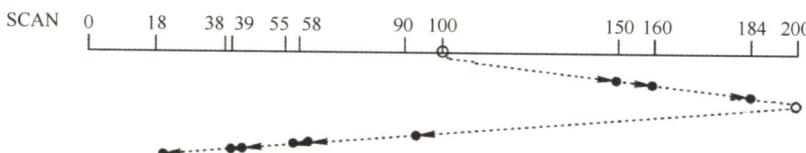


图 5.18 SCAN 磁盘调度算法

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图 5.18 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  $= 282/9 = 31.33$ 。

### (4) 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 5.19 所示。

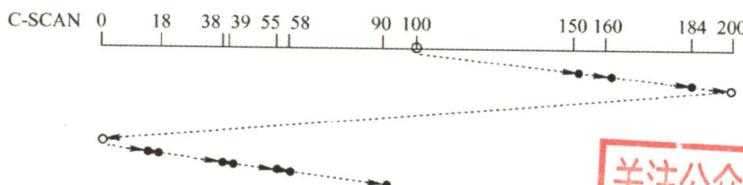


图 5.19 C-SCAN 磁盘调度算法

关注公众号【乘龙考研】  
一手更新 稳定有保障

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。这种形式的 SCAN 算法和 C-SCAN 算法称为 LOOK 调度（见图 5.20）和 C-LOOK（见图 5.21）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。

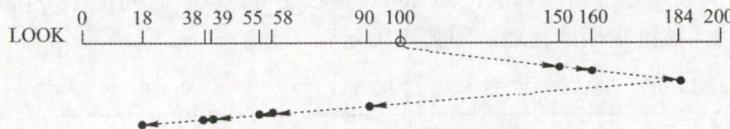


图 5.20 LOOK 磁盘调度算法

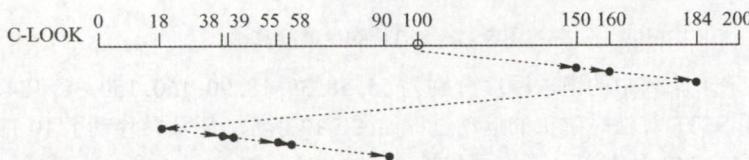


图 5.21 C-LOOK 磁盘调度算法

注意，若无特别说明，也可以默认 SCAN 算法和 C-SCAN 算法为 LOOK 和 C-LOOK 调度（请读者认真领悟，并通过结合后面的习题进一步加深对以上相关算法的理解）。

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图 5.19 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  $50 + 10 + 24 + 16 + 200 + 18 + 20 + 1 + 16 + 3 + 32 = 390/9 = 43.33$ 。

请读者注意区分磁盘调度算法中的循环扫描算法和页面调度算法中的 CLOCK 算法。

对比以上几种磁盘调度算法，FCFS 算法太过简单，性能较差，仅在请求队列长度接近于 1 时才较为理想；SSTF 算法较为通用和自然；SCAN 算法和 C-SCAN 算法在磁盘负载较大时比较占优势。它们之间的比较见表 5.2。

表 5.2 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	寻道性能较好，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	—

除减少寻找时间外，减少延迟时间也是提高磁盘传输效率的重要因素。可以对盘面扇区进行交替编号，对磁盘片组中的不同盘面错位命名。假设每个盘面有 8 个扇区，磁盘片组共 8 个盘面，则可以采用如图 5.22 所示的编号。

磁盘是连续自转设备，磁头读/写一个物理块后，需要经过短暂的处理时间才能开始读/写下一块。假设逻辑记录数据连续存放在磁盘空间中，若在盘面上按扇区交替编号连续存放，则连续读/写多条记录时能减少磁头的延迟时间；同柱面不同盘面的扇区若能错位编号，连续读/写相邻两个盘面的逻辑记录时也能减少磁头延迟时间。

以图 5.22 为例，在随机扇区访问情况下，定位磁道中的一个扇区平均需要转过 4 个扇区，这时，延迟时间是传输时间的 4 倍，这是一种非常低效的方式。理想的情况是不需要定位而直接连

续读取扇区，没有延迟时间，这样磁盘数据存取效率可以成倍提高。但由于读取扇区的顺序是不可预测的，所以延迟时间不可避免。图 5.22 中的编号方式是读取连续编号扇区时的一种方法。

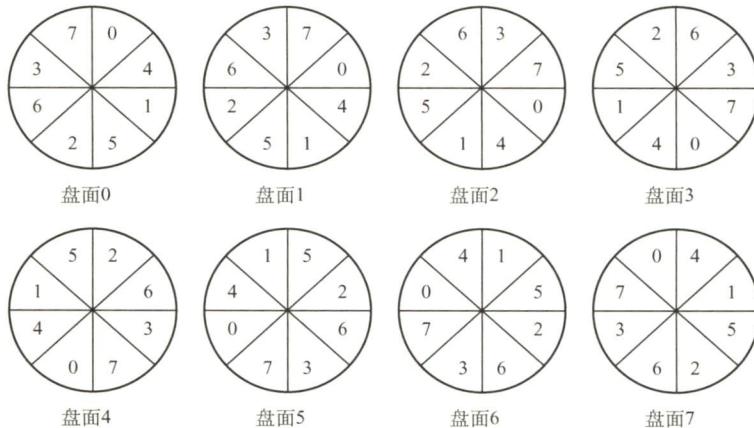


图 5.22 磁盘片组扇区编号

关注公众号【乘龙考研】  
一手更新 稳定有保障

磁盘寻块时间分为三个部分，即寻道时间、延迟时间和传输时间，寻道时间和延迟时间属于“找”的时间，凡是“找”的时间都可以通过一定的方法削减，但传输时间是磁盘本身性质所决定的，不能通过一定的措施减少。

### 5.3.4 固态硬盘

#### 1. 固态硬盘的特性

固态硬盘(SSD)是一种基于闪存技术的存储器。它与U盘并无本质差别，只是容量更大，存取性能更好。一个SSD由一个或多个闪存芯片和闪存翻译层组成，如图5.23所示。闪存芯片替代传统旋转磁盘中的机械驱动器，而闪存翻译层将来自CPU的逻辑块读写请求翻译成对底层物理设备的读写控制信号，因此闪存翻译层相当于扮演了磁盘控制器的角色。

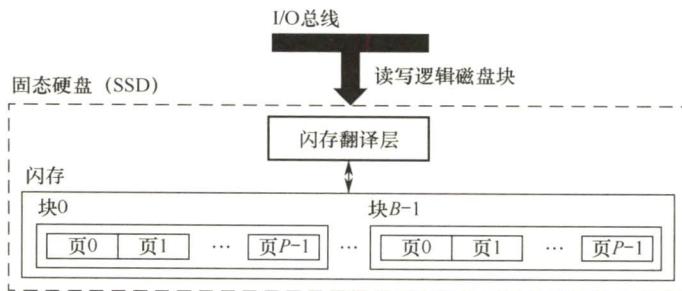


图 5.23 固态硬盘( SSD )

在图5.23中，一个闪存由 $B$ 块组成，每块由 $P$ 页组成。通常，页的大小是512B~4KB，每块由32~128页组成，块的大小为16KB~512KB。数据是以页为单位读写的。只有在一页所属的块整个被擦除后，才能写这一页。不过，一旦擦除一块，块中的每页就都可以直接再写一次。某块进行若干次重复写后，就会磨损坏，不能再使用。

随机写很慢，有两个原因。首先，擦除块比较慢，通常比访问页高一个数量级。其次，如果写操作试图修改包含已有数据的页 $P_i$ ，那么这个块中所有含有用数据的页都必须被复制到一个新(擦除过的)块中，然后才能进行对页 $P_i$ 的写操作。

比起传统磁盘，SSD 有很多优点，它由半导体存储器构成，没有移动的部件，因而随机访问速度比机械磁盘要快很多，也没有任何机械噪声和震动，能耗更低、抗震性好、安全性高等。

随着技术的不断发展，价格也不断下降，SSD 会有望逐步取代传统机械硬盘。

## 2. 磨损均衡 (Wear Leveling)

固态硬盘也有缺点，闪存的擦写寿命是有限的，一般是几百次到几千次。如果直接用普通闪存组装 SSD，那么实际的寿命表现可能非常令人失望——读写数据时会集中在 SSD 的一部分闪存，这部分闪存的寿命会损耗得特别快。一旦这部分闪存损坏，整块 SSD 也就损坏了。这种磨损不均衡的情况，可能会导致一块 256GB 的 SSD，只因数兆空间的闪存损坏而整块损坏。

为了弥补 SSD 的寿命缺陷，引入了磨损均衡。SSD 磨损均衡技术大致分为两种：

1) 动态磨损均衡。写入数据时，自动选择较新的闪存块。老的闪存块先歇一歇。

2) 静态磨损均衡。这种技术更为先进，就算没有数据写入，SSD 也会监测并自动进行数据分配，让老的闪存块承担无须写数据的存储任务，同时让较新的闪存块腾出空间，平常的读写操作在较新的闪存块中进行。如此一来，各闪存块的寿命损耗就都差不多。

有了这种算法加持，SSD 的寿命就比较可观了。例如，对于一个 256GB 的 SSD，如果闪存的擦写寿命是 500 次，那么就需要写入 125TB 数据，才寿终正寝。就算每天写入 10GB 数据，也要三十多年才能将闪存磨损坏，更何况很少有人每天往 SSD 中写入 10GB 数据。

## 5.3.5 本节小结

本节开头提出的问题的参考答案如下。

1) 在磁盘上进行一次读写操作需要哪几部分时间？其中哪部分时间最长？

在磁盘上进行一次读写操作花费的时间由寻道时间、延迟时间和传输时间决定。其中寻道时间是将磁头移动到指定磁道所需要的时间，延迟时间是磁头定位到某一磁道的扇区（块号）所需要的时间，传输时间是从磁盘读出或向磁盘写入数据所经历的时间。一般来说，寻道时间因为要移动磁臂，所以占用时间最长。

2) 存储一个文件时，当一个磁道存储不下时，剩下部分是存在同一个盘面的不同磁道好，还是存在同一个柱面上的不同盘面好？

上一问已经说到，寻道时间对于一次磁盘访问的影响是最大的，若存在同一个盘面的不同磁道，则磁臂势必要移动，这样会大大增加文件的访问时间，而存在同一个柱面上的不同盘面就不需要移动磁道，所以一般情况下存在同一个柱面上的不同盘面更好。

## 5.3.6 本节习题精选

### 一、单项选择题

01. 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

02. 既可以顺序读写，又可以按任意次序读写的存储器有（ ）。

- |              |             |           |         |
|--------------|-------------|-----------|---------|
| I. 光盘        | II. 磁带      | III. U 盘  | IV. 磁盘  |
| A. II、III、IV | B. I、III、IV | C. III、IV | D. 仅 IV |

03. 磁盘调度的目的是缩短（ ）时间。

- A. 找道
- B. 延迟
- C. 传送
- D. 启动

04. 磁盘上的文件以（ ）为单位读/写。

- A. 块                    B. 记录                    C. 柱面                    D. 磁道
05. 在磁盘中读取数据的下列时间中，影响最大的是( )。
- A. 处理时间            B. 延迟时间            C. 传送时间            D. 寻找时间
06. 硬盘的操作系统引导扇区产生在( )。
- A. 对硬盘进行分区时            B. 对硬盘进行低级格式化时  
C. 硬盘出厂时自带            D. 对硬盘进行高级格式化时
07. 在下列有关旋转延迟的叙述中，不正确的是( )。
- A. 旋转延迟的大小与磁盘调度算法无关  
B. 旋转延迟的大小取决于磁盘空闲空间的分配程序  
C. 旋转延迟的大小与文件的物理结构有关  
D. 扇区数据的处理时间对旋转延迟的影响较大
08. 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
09. 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
10. 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. 以上答案都不对
11. 已知某磁盘的平均转速为  $r$  秒/转，平均寻找时间为  $T$  秒，每个磁道可以存储的字节数为  $N$ ，现向该磁盘读写  $b$  字节的数据，采用随机寻道的方法，每道的所有扇区组成一个簇，其平均访问时间是( )。
- A.  $(r + T)b/N$             B.  $b/NT$             C.  $(b/N + T)r$             D.  $bT/N + r$
12. 设磁盘的转速为 3000 转/分，盘面划分为 10 个扇区，则读取一个扇区的时间为( )。
- A. 20ms            B. 5ms            C. 2ms            D. 1ms
13. 一个磁盘的转速为 7200 转/分，每个磁道有 160 个扇区，每扇区有 512B，那么理想情况下，其数据传输率为( )。
- A.  $7200 \times 160 \text{KB/s}$             B.  $7200 \text{KB/s}$             C.  $9600 \text{KB/s}$             D.  $19200 \text{KB/s}$
14. 设一个磁道访问请求序列为 55, 58, 39, 18, 90, 160, 150, 38, 184，磁头的起始位置为 100，若采用 SSTF（最短寻道时间优先）算法，则磁头移动( )个磁道。
- A. 55            B. 184            C. 200            D. 248
15. 假定磁带的记录密度为 400 字符/英寸 ( $1\text{in} = 0.0254\text{m}$ )，每条逻辑记录为 80 字符，块间隙为 0.4 英寸，现有 3000 个逻辑记录需要存储，存储这些记录需要长度为( )的磁带，磁带利用率是( )。
- A. 1500 英寸，33.3%            B. 1500 英寸，43.5%  
C. 1800 英寸，33.3%            D. 1800 英寸，43.5%
16. 下列关于固态硬盘(SSD)的说法中，错误的是( )。
- A. 基于闪存的存储技术            B. 随机读写性能明显高于磁盘  
C. 随机写比较慢            D. 不易磨损
17. 下列关于固态硬盘的说法中，正确的是( )。

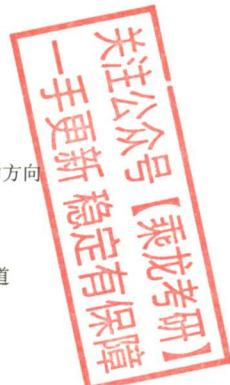
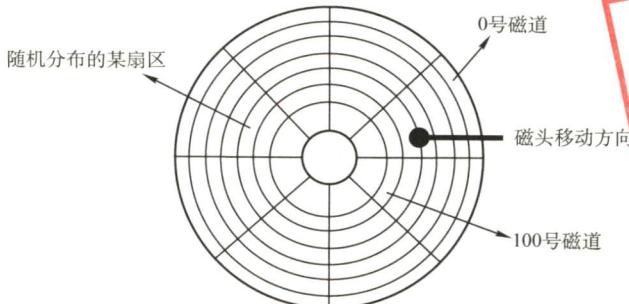
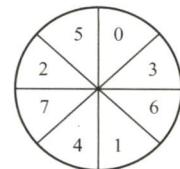
关注公众号【乘龙考研】  
一手更新 稳定有保障

- A. 固态硬盘的写速度比较慢，性能甚至弱于常规硬盘  
 B. 相比常规硬盘，固态硬盘优势主要体现在连续存取的速度  
 C. 静态磨损均衡算法比动态磨损均衡算法的表现更优秀  
 D. 写入时，每次选择使用长期存放数据而很少擦写的存储块
18. 【2009 统考真题】假设磁头当前位于第 105 道，正在向磁道序号增加的方向移动。现有一个磁道访问请求序列为 35, 45, 12, 68, 110, 180, 170, 195，采用 SCAN 调度（电梯调度）算法得到的磁道访问序列是（ ）。  
 A. 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
19. 【2015 统考真题】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
20. 【2017 统考真题】下列选项中，磁盘逻辑格式化程序所做的工作是（ ）。  
 I. 对磁盘进行分区  
 II. 建立文件系统的根目录  
 III. 确定磁盘扇区校验码所占位数  
 IV. 对保存空闲磁盘块信息的数据结构进行初始化  
 A. 仅 II      B. 仅 II、IV      C. 仅 III、IV      D. 仅 I、II、IV
21. 【2017 统考真题】某文件系统的簇和磁盘扇区大小分别为 1KB 和 512B。若一个文件的大小为 1026B，则系统分配给该文件的磁盘空间大小是（ ）。  
 A. 1026B      B. 1536B      C. 1538B      D. 2048B
22. 【2018 统考真题】系统总是访问磁盘的某个磁道而不响应对其他磁道的访问请求，这种现象称为磁臂黏着。下列磁盘调度算法中，不会导致磁臂黏着的是（ ）。  
 A. 先来先服务（FCFS）      B. 最短寻道时间优先（SSTF）  
 C. 扫描算法（SCAN）      D. 循环扫描算法（CSCAN）
23. 【2021 统考真题】某系统中磁盘的磁道数为 200（0~199），磁头当前在 184 号磁道上。用户进程提出的磁盘访问请求对应的磁道号依次为 184, 187, 176, 182, 199。若采用最短寻道时间优先调度算法（SSTF）完成磁盘访问，则磁头移动的距离（磁道数）是（ ）。  
 A. 37      B. 38      C. 41      D. 42

## 二、综合应用题

01. 假定有一个磁盘组共有 100 个柱面，每个柱面有 8 个磁道，每个磁道划分成 8 个扇区。现有一个 5000 条逻辑记录的文件，逻辑记录的大小与扇区大小相等，该文件以顺序结构被存放在磁盘组上，柱面、磁道、扇区均从 0 开始编址，逻辑记录的编号从 0 开始，文件信息从 0 柱面、0 磁道、0 扇区开始存放。试问，该文件编号为 3468 的逻辑记录应存放在哪个柱面的第几个磁道的第几个扇区上？
02. 假设磁盘的每个磁道分成 9 个块，现在一个文件有 A, B, …, I 共 9 条记录，每条记录的大小与块的大小相等，设磁盘转速为 27ms/转，每读出一块后需要 2ms 的处理时间。若忽略其他辅助时间，且一开始磁头在即将要读 A 记录的位置，试问：  
 1) 若将这些记录顺序存放在一个磁道上，则顺序读取该文件要多少时间？  
 2) 若要求顺序读取的时间最短，则应该如何安排文件的存放位置？

03. 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 FCFS 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) FCFS; 2) SSTF; 3) SCAN; 4) LOOK; 5) C-SCAN; 6) C-LOOK。
04. 某软盘有 40 个磁道，磁头从一个磁道移至相邻磁道需要 6ms。文件在磁盘上非连续存放，逻辑上相邻数据块的平均距离为 13 磁道，每块的旋转延迟时间及传输时间分别为 100ms 和 25ms，问读取一个 100 块的文件需要多少时间？若系统对磁盘进行了整理，让同一文件的磁盘块尽可能靠拢，从而使逻辑上相邻数据块的平均距离降为 2 磁道，这时读取一个 100 块的文件需要多少时间？
05. 有一个交叉存放信息的磁盘，信息在其上的存放方法如右图所示。每个磁道有 8 个扇区，每个扇区 512B，旋转速度为 3000 转/分，顺时针读扇区。假定磁头已在读取信息的磁道上，0 扇区转到磁头下需要 1/2 转，且设备对应的控制器不能同时进行输入/输出，在数据从控制器传送至内存的这段时间内，从磁头下通过的扇区数为 2，请回答：
- 1) 依次读取一个磁道上的所有扇区需要多少时间？
  - 2) 该磁盘的数据传输速率是多少？
06. 【2010 统考真题】如下图所示，假设计算机系统采用 C-SCAN（循环扫描）磁盘调度策略，使用 2KB 的内存空间记录 16384 个磁盘块的空闲状态。



- 1) 请说明在上述条件下如何进行磁盘块空闲状态的管理。
  - 2) 设某单面磁盘的旋转速度为 6000 转/分，每个磁道有 100 个扇区，相邻磁道间的平均移动时间为 1ms。若在某时刻，磁头位于 100 号磁道处，并沿着磁道号增大的方向移动（见上图），磁道号请求队列为 50, 90, 30, 120，对请求队列中的每个磁道需读取 1 个随机分布的扇区，则读完这 4 个扇区点共需要多少时间？要求给出计算过程。
  - 3) 若将磁盘替换为随机访问的 Flash 半导体存储器（如 U 盘、固态硬盘等），是否有比 C-SCAN 更高效的磁盘调度策略？若有，给出磁盘调度策略的名称并说明理由；若无，说明理由。
07. 【2019 统考真题】某计算机系统中的磁盘有 300 个柱面，每个柱面有 10 个磁道，每个磁道有 200 个扇区，扇区大小为 512B。文件系统的每簇包含 2 个扇区。请回答下列问题：
- 1) 磁盘的容量是多少？
  - 2) 设磁头在 85 号柱面上，此时有 4 个磁盘访问请求，簇号分别为 100260, 60005, 101660 和 110560。采用最短寻道时间优先 SSTF 调度算法，系统访问簇的先后次序是什么？
  - 3) 簇号 100530 在磁盘上的物理地址是什么？将簇号转换成磁盘物理地址的过程由 I/O

系统的什么程序完成？

**08. 【2021 统考真题】**某计算机用硬盘作为启动盘，硬盘的第一个扇区存放主引导记录，其中包含磁盘引导程序和分区表。磁盘引导程序用于选择引导哪个分区的操作系统，分区表记录硬盘上各分区的位置等描述信息。硬盘被划分成若干分区，每个分区的第一个扇区存放分区引导程序，用于引导该分区中的操作系统。系统采用多阶段引导方式，除了执行磁盘引导程序和分区引导程序，还需要执行 ROM 中的引导程序。回答下列问题：

- 1) 系统启动过程中操作系统的初始化程序、分区引导程序、ROM 中的引导程序、磁盘引导程序的执行顺序是什么？
- 2) 把硬盘制作作为启动盘时，需要完成操作系统的安装、磁盘的物理格式化、逻辑格式化、对磁盘进行分区，执行这 4 个操作的正确顺序是什么？
- 3) 磁盘扇区的划分和文件系统根目录的建立分别是在第 2) 问的那个操作中完成的？

### 5.3.7 答案与解析

#### 一、单项选择题

**01. D**

磁盘是可共享设备（分时共享），是指某段时间内可以有多个用户进行访问。但某一时刻只能有一个作业可以访问。

**02. B**

顺序访问按从前到后的顺序对数据进行读写操作，如磁带。直接访问或随机访问，则可以按任意的次序对数据进行读写操作，如光盘、磁盘、U 盘等。

**03. A**

磁盘调度是对访问磁道次序的调度，若没有合适的磁盘调度，则寻找时间会大大增加。

**04. A**

文件以块为单位存放于磁盘中，文件的读写也以块为单位。

**05. D**

磁盘调度中，对读/写时间影响最大的是寻找时间，寻找过程为机械运动，时间较长，影响较大。

**06. D**

操作系统的引导程序位于磁盘活动分区的引导扇区，因此必然产生在分区之后。分区是将磁盘分为由一个或多个柱面组成的分区（即 C 盘、D 盘等形式），每个分区的起始扇区和大小都记录在磁盘主引导记录的分区表中。而对于高级格式化（创建文件系统），操作系统将初始的文件系统数据结构存储到磁盘上，文件系统在磁盘上布局介绍详见第 4 章。

**07. D**

磁盘调度算法是为了减少寻找时间。扇区数据的处理时间主要影响传输时间。选项 B、C 均与旋转延迟有关，文件的物理结构与磁盘空间的分配方式相对应，包括连续分配、链接分配和索引分配。连续分配的磁盘中，文件的物理地址连续；而链接分配方式的磁盘中，文件的物理地址不连续，因此与旋转延迟都有关。

**08. C**

选项 A 是进程调度算法；选项 B 是页面淘汰算法；选项 D 可以用于进程调度和作业调度。只有选项 C 是磁盘调度算法。

**09. B**

关注公众号【乘龙考研】  
一手更新 稳定有保障

最短寻找时间优先算法中，当新的距离磁头比较近的磁盘访问请求不断被满足时，可能会导致较远的磁盘访问请求被无限延迟，从而导致“饥饿”现象。

**10. B**

先来先服务算法根据磁盘请求的时间先后进行调度，因而可能随时改变磁头方向。而电梯调度、循环扫描算法均限制磁头的移动方向。

**11. A**

将每道的所有扇区组成一个簇，意味着可以将一个磁道的所有存储空间组织成一个数据块组，这样有利于提高存储速度。读写磁盘时，磁头首先找到磁道，称为寻道，然后才可以将信息从磁道里读出或写入。读写完一个磁道后，磁头会继续寻找下一个磁道，完成剩余的工作，所以在随机寻道的情况下，读写一个磁道的时间要包括寻道时间和读写磁道时间，即  $T + r$  秒。由于总的数据量是  $b$  字节，它要占用的磁道数为  $b/N$  个，所以总平均读写时间为  $(r + T)b/N$  秒。

**12. C**

访问每条磁道的时间为  $60/3000\text{s} = 0.02\text{s} = 20\text{ms}$ ，即磁盘旋转一圈的时间为  $20\text{ms}$ ，每个盘面 10 个扇区，因此读取一个扇区的时间为  $20\text{ms}/10 = 2\text{ms}$ 。

**13. C**

磁盘的转速为 7200 转/分 = 120 转/秒，转一圈经过 160 个扇区，每个扇区为 512B，所以数据传输率 =  $120 \times 160 \times 512 / 1024\text{KB/s} = 9600\text{KB/s}$ 。

**14. D**

对于 SSTF 算法，寻道序列应为 100, 90, 58, 55, 39, 38, 18, 150, 160, 184；移动磁道次数分别为 10, 32, 3, 16, 1, 20, 132, 10, 24，故磁头移动总次数为 248。另外也可以画出草图来解答，从 100 寻道到 18 需要 82 次，然后加上从 18 到 184 需要的  $184 - 18 = 166$  次，共移动  $166 + 82 = 248$  次。

**15. C**

一个逻辑记录所占的磁带长度为  $80/400 = 0.2$  英寸，因此存储 3000 条逻辑记录需要的磁带长度为  $(0.2 + 0.4) \times 3000 = 1800$  英寸，利用率为  $0.2/(0.2 + 0.4) = 33.3\%$ 。

**16. D**

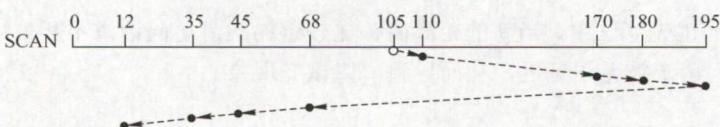
固态硬盘基于闪存技术，没有机械部件，随机读写不需要机械操作，因此速度明显高于磁盘，选项 A 和 B 正确。选项 C 已在考点讲解中解释过。SSD 的缺点是容易磨损，选项 D 错误。

**17. C**

SSD 的写速度慢于读速度，但不至于比常规机械硬盘差，选项 A 错误。SSD 基于闪存技术，没有机械部件，随机存取速度很快，传统机械硬盘因为需要寻道和找扇区的时间，所以随机存取速度慢；传统机械硬盘转速很快，连续存取比随机存取快得多，因此 SSD 的优势主要体现在随机存取的速度上，选项 B 错误。静态磨损算法在没有写入数据时，SSD 监测并自动进行数据分配，因此表现更为优秀，选项 C 正确。因为闪存的擦除速度较慢，若每次都选择写入存放有数据的块，会极大地降低写入速度，选项 D 混淆了静态磨损均衡，静态磨损均衡是指在没有写入数据时，SSD 监测并自动进行数据分配，从而使得各块的擦写更加均衡，并不是说写入时每次都选择存放老数据的块。

**18. A**

SCAN 算法的原理类似于电梯。首先，当磁头从 105 道向序号增加的方向移动时，便会按照从小到大的顺序服务所有大于 105 的磁道号（110, 170, 180, 195）；往回移动时又会按照从大到小的顺序进行服务（68, 45, 35, 12），结果如下图所示。

**19. C**

SCAN 算法就是电梯调度算法。顾名思义，若开始时磁头向外移动，就一直要到最外侧，然后返回向内侧移动，就像电梯若往下则一直要下到底层才会再上升一样。当前磁头位于 58 号并从外侧向内侧移动，先依次访问 130、180 和 199，然后返回向外侧移动，依次访问 42 和 15，因此磁头移过的磁道数是 $(199 - 58) + (199 - 15) = 325$ 。

**20. B**

新磁盘是空白的，必须分成各个扇区以便磁盘控制器能读和写，这个过程称为低级格式化（或物理格式化）。低级格式化为磁盘的每个扇区采用特别的数据结构，包括校验码，III 错误。为了使用磁盘存储文件，操作系统还需要将自己的数据结构记录在磁盘上。这分为两步。第一步是将磁盘分为由一个或多个柱面组成的分区，每个分区可以作为一个独立的磁盘，I 错误。在分区之后，第二步是逻辑格式化（创建文件系统）。在这一步，操作系统将初始的文件系统数据结构存储到磁盘上。这些数据结构包括空闲和已分配的空间及一个初始为空的目录，II、IV 正确。

**21. D**

绝大多数操作系统为改善磁盘访问时间，以簇为单位进行空间分配，因此答案选择选项 D。

**22. A**

当系统总是持续出现某个磁道的访问请求时，均持续满足最短寻道时间优先、扫描算法和循环扫描算法的访问条件，会一直服务该访问请求。而先来先服务按照请求次序进行调度，比较公平，因此选择选项 A。

**23. C**

最短寻道时间优先算法总是选择调度与当前磁头所在磁道距离最近的磁道。可以得出访问序列 184, 182, 187, 176, 199，从而求出移动距离之和是  $0 + 2 + 5 + 11 + 23 = 41$ 。

## 二、综合应用题

**01. 【解答】**

该磁盘有 8 个盘面，一个柱面大小为  $8 \times 8 = 64$  个扇区，即 64 条逻辑记录。由于所有磁头是固定在一起的，因此在存放数据时，先存满扇区，后存满磁道，再存满柱面。

编号为 3468 的逻辑记录对应的柱面号为  $3468 / 64 = 54$ ；对应的磁道号为  $(3468 \bmod 64) \text{ DIV } 8 = 1$ ；对应的扇区号为  $(3468 \bmod 64) \bmod 8 = 4$ 。

**02. 【解答】**

磁盘转速为 27ms/转，每个磁道存放 9 条记录，因此读出 1 条记录的时间为  $27/9 = 3\text{ms}$ 。

1) 读出并处理记录 A 需要 5ms，此时磁头已转到记录 B 的中间，因此为了读出记录 B，必须再转接近一圈（从记录 B 的中间到记录 B）。后续 8 条记录的读取及处理与此类似，但最后一条记录的读取与处理只需 5ms。于是，处理 9 条记录的总时间为

$$8 \times (27 + 3) + (3 + 2) = 245\text{ms}$$

【另解】注意，从开始读 A 到最后读完 I 一共转了 9 圈，即处理完前 8 条记录 + 读第 9 条记录的时间一共是  $27 \times 9 = 243\text{ms}$ ，加上最后的 2ms 处理时间，一共是  $243 + 2 = 245\text{ms}$ 。

2) 由于每读出一条记录后需要 2ms 的处理时间，当读出并处理记录 A 时，不妨设记录 A 放在第 1 个盘块中，读写头已移到第 2 个盘块的中间，为了能顺序读到记录 B，应将它放

到第3个盘块中，即应将记录按下表顺序存放：

盘块	1	2	3	4	5	6	7	8	9
记录	A	F	B	G	C	H	D	I	E

这样，处理一条记录并将磁头移到下一条记录的时间是

$$3 \text{ (读出)} + 2 \text{ (处理)} + 1 \text{ (等待)} = 6\text{ms}$$

所以，处理9条记录的总时间为

$$6 \times 8 + (3 + 2) = 53\text{ms}$$

#### 03. 【解答】

- 1) FCFS：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，仔细的读者会发现：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要快捷一些？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  $222 + 18 + 271 + 99 + 217 + 182 = 1009$ 。
- 5) C-SCAN：移动磁道的顺序为 345, 123, 105, 0, 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  $222 + 18 + 769 + 182 + 217 + 99 = 1507$ 。

#### 04. 【解答】

磁盘整理前，逻辑上相邻数据块的平均距离为 13 磁道，读一块数据需要的时间为

$$13 \times 6 + 100 + 25 = 203\text{ms}$$

因此，读取一个 100 块的文件需要的时间为

$$203 \times 100 = 20300\text{ms}$$

磁盘整理后，逻辑上相邻数据块的平均距离为 2 磁道，读一块数据需要的时间为

$$2 \times 6 + 100 + 25 = 137\text{ms}$$

因此，读取一个 100 块的文件需要的时间为

$$137 \times 100 = 13700\text{ms}$$

#### 05. 【解答】

磁盘逆时针方向旋转按扇区来看即 0, 3, 6, … 这个顺序。每个号码连续的扇区正好相隔 2 个扇区，即数据从控制器传送到内存的时间，所以相当于磁头连续工作。

1) 由题中条件知，旋转速度为 3000 转/分 = 50 转/秒，即 20ms/转。

读一个扇区需要的时间为  $20/8 = 2.5\text{ms}$ 。

读一个扇区并将扇区数据送入内存需要的时间为  $2.5 \times 3 = 7.5\text{ms}$ 。

故读出一个磁道上的所有扇区需要的时间为  $20/2 + 8 \times 7.5 = 70\text{ms} = 0.07\text{s}$ 。

关注公众号【乘龙考研】  
一手更新 稳定有保障

2) 每个磁道的数据量为  $8 \times 512 = 4\text{KB}$ 。

故数据传输速率为  $4\text{KB}/0.07\text{s} = 4 \times 1024\text{B}/(1000 \times 0.07\text{s}) = 58.5\text{kB/s}$ 。

**注意：**表示存储容量、文件大小时 K 等于 1024（通常用大写的 K），表示传输速率时 k 等于 1000（通常用小写的 k），注意区别。

#### 06. 【解答】

1) 用位图表示磁盘的空闲状态。每位表示一个磁盘块的空闲状态，共需  $16384/32 = 512$  个字

$$= 512 \times 4\text{B} = 2\text{KB}$$

2) 采用 C-SCAN 调度算法，访问磁道的顺序和移动的磁道数如下表所示：

被访问的下一个磁道号	移动距离（磁道数）
120	20
30	90
50	20
90	40

移动的磁道数为  $20 + 90 + 20 + 40 = 170$ ，因此总的移动磁道时间为 170ms。

由于转速为 6000 转/分，因此平均旋转延迟为 5ms，总的旋转延迟时间 = 20ms。

由于转速为 6000 转/分，因此读取一个磁道上的一个扇区的平均读取时间为 0.1ms，扇区的平均读取时间为 0.1ms，总的读取扇区的时间为 0.4ms。

综上，读取上述磁道上所有扇区所花的总时间为 190.4ms。

3) 采用先来先服务 (FCFS) 调度策略更高效。因为 Flash 半导体存储器的物理结构不需要考虑寻道时间和旋转延迟，可直接按 I/O 请求的先后顺序服务。

#### 07. 【解答】

1) 磁盘容量 = 磁盘的柱面数 × 每个柱面的磁道数 × 每个磁道的扇区数 × 每个扇区的大小 =  $(300 \times 10 \times 200 \times 512/1024)\text{KB} = 3 \times 10^5\text{KB}$ 。

2) 磁头在 85 号柱面上，对 SSTF 算法而言，总是访问当前柱面距离最近的地址。注意每个簇包含 2 个扇区，通过计算得到，85 号柱面对应的簇号为 85000~85999。通过比较得出，系统最先访问离 85000~85999 最近的 100260，随后访问离 100260 最近的 101660，然后访问 110560，最后访问 60005。顺序为 100260, 101660, 110560, 60005。

3) 第 100530 簇在磁盘上的物理地址由其所在的柱面号、磁头号、扇区号构成。

$$\text{柱面号} = \lfloor \text{簇号}/\text{每个柱面的簇数} \rfloor = \lfloor 100530/(10 \times 200/2) \rfloor = 100.$$

$$\text{磁头号} = \lfloor (\text{簇号} \% \text{每个柱面的簇数})/\text{每个磁道的簇数} \rfloor = \lfloor 530/(200/2) \rfloor = 5.$$

$$\text{扇区号} = \text{扇区地址} \% \text{每个磁道的扇区数} = (530 \times 2) \% 200 = 60.$$

将簇号转换成磁盘物理地址的过程由磁盘驱动程序完成。

#### 08. 【解答】

1) 执行顺序依次是 ROM 中的引导程序、磁盘引导程序、分区引导程序、操作系统的初始化程序。启动系统时，首先运行 ROM 中的引导代码 (bootstrap)。为执行某个分区的操作系统的初始化程序，需要先执行磁盘引导程序以指示引导到哪个分区，然后执行该分区的引导程序，用于引导该分区的操作系统。

2) 4 个操作的执行顺序依次是磁盘的物理格式化、对磁盘进行分区、逻辑格式化、操作系统的安装。磁盘只有通过分区和逻辑格式化后才能安装系统和存储信息。物理格式化（又称低级格式化，通常出厂时就已完成）的作用是为每个磁道划分扇区，安排扇区在磁道中的排列顺序，并对已损坏的磁道和扇区做“坏”标记等。随后将磁盘的整体存储空间

划分为相互独立的多个分区(如Windows中划分C盘、D盘等),这些分区可以用作多种用途,如安装不同的操作系统和应用程序、存储文件等。然后进行逻辑格式化(又称高级格式化),其作用是对扇区进行逻辑编号、建立逻辑盘的引导记录、文件分配表、文件目录表和数据区等。最后才是操作系统的安装。

- 3) 由上述分析可知,磁盘扇区的划分是在磁盘的物理格式化操作中完成的,文件系统根目录的建立是在逻辑格式化操作中完成的。

关注公众号【乘龙考研】  
一手更新 稳定有保障

## 5.4 本章疑难点

### 1. 设备分配

- 1) 分配设备。首先根据I/O请求中的物理设备名查找系统设备表(SDT),从中找出该设备的DCT,再根据DCT中的设备状态字段,可知该设备是否忙。若忙,便将请求I/O进程的PCB挂到设备队列上;若空闲,则按照一定的算法计算设备分配的安全性,若安全则将设备分配给请求进程,否则仍将其PCB挂到设备队列上。
- 2) 分配控制器。系统把设备分配给请求I/O的进程后,再到其DCT中找出与该设备连接的控制器的COCT,从COCT中的状态字段中可知该控制器是否忙碌。若忙,则将请求I/O进程的PCB挂到该控制器的等待队列上;若空闲,则将控制器分配给进程。
- 3) 分配通道。在该COCT中又可找到与该控制器连接的通道的CHCT,再根据CHCT内的状态信息,可知该通道是否忙碌。若忙,则将请求I/O的进程挂到该通道的等待队列上;若空闲,则将该通道分配给进程。只有在上述三者都分配成功时,这次设备的分配才算成功。然后,便可启动该I/O设备进行数据传送。

为使独占设备的分配具有更强的灵活性,提高分配的成功率,还可从以下两方面对基本的设备分配程序加以改进:

- 1) 增加设备的独立性。进程使用逻辑设备名请求I/O。这样,系统首先从SDT中找出第一个该类设备的DCT。若该设备忙,则又查找第二个该类设备的DCT。仅当所有该类设备都忙时,才把进程挂到该类设备的等待队列上;只要有一个该类设备可用,系统便进一步计算分配该设备的安全性。
- 2) 考虑多通路情况。为防止I/O系统的“瓶颈”现象,通常采用多通路的I/O系统结构。此时对控制器和通道的分配同样要经过几次反复,即若设备(控制器)所连接的第一个控制器(通道)忙时,则应查看其所连接的第二个控制器(通道),仅当所有控制器(通道)都忙时,此次的控制器(通道)分配才算失败,才把进程挂到控制器(通道)的等待队列上。而只要有一个控制器(通道)可用,系统便可将它分配给进程。

设备分配过程中,先后分别访问的数据结构为SDT→DCT→COCT→CHCT。要成功分配一个设备,必须要:①设备可用;②控制器可用;③通道可用。所以,“设备分配,要过三关”。

### 2. 提高磁盘I/O速度的方法

- 1) 提前读。在读磁盘当前块时,把下一磁盘块也读入内存缓冲区。
- 2) 延迟写。仅在缓冲区首部设置延迟写标志,然后释放此缓冲区并将其链入空闲缓冲区链表的尾部,当其他进程申请到此缓冲区时,才真正把缓冲区信息写入磁盘块。
- 3) 虚拟盘。是指用内存空间去仿真磁盘,又叫RAM盘。虚拟盘是一种易失性存储器。虚拟盘常用于存放临时文件。

## 参 考 文 献

- [1] 汤小丹. 计算机操作系统[M]. 西安: 西安电子科技大学出版社, 2014.
- [2] 李善平. 操作系统学习指导和考试指导[M]. 杭州: 浙江大学出版社, 2004.
- [3] Tanenbaum A. S. 现代操作系统[M]. 北京: 机械工业出版社, 2017.
- [4] Abraham Silberschatz. 操作系统概念[M]. 北京: 机械工业出版社, 2018.
- [5] William Stallings. 操作系统: 精髓与设计原理[M]. 北京: 机械工业出版社, 2017.
- [6] 本书编写组. 计算机专业基础综合考试大纲解析[M]. 北京: 高等教育出版社, 2009.
- [7] 李春葆等. 操作系统统考辅导教程[M]. 北京: 清华大学出版社, 2010.
- [8] 崔魏等. 计算机学科专业基础综合辅导讲义[M]. 北京: 原子能出版社, 2011.
- [9] 翔高教育. 计算机学科专业基础综合复习指南[M]. 上海: 复旦大学出版社, 2009.
- [10] Bryant R. E.等. 深入理解计算机系统[M]. 北京: 机械工业出版社, 2010.

更多考研考证类日更网盘群&笔记类资料 请关注微信公众言【秉龙考研】

经久才是王道

# 十五年考研磨练

## 2024年王道计算机考研课程

- ◆ 阶段一：计算机考研零基础入门
- ◆ 阶段二：大纲考点、经典习题精讲
- ◆ 阶段三：暑期强化提升训练
- ◆ 阶段四：真题、模拟题、考前冲刺
- ◆ 阶段五：复试机试课程



加客服领券购课

抄/底/价/299元起

- 2024年数据结构考研复习指导
- 2024年计算机组成原理考研复习指导
- 2024年操作系统考研复习指导
- 2024年计算机网络考研复习指导
- 2024年计算机专业基础综合考试历年真题解析
- 2024年计算机专业基础综合考试冲刺模拟题
- 计算机考研——机试指南（第2版）



责任编辑：谭海平  
封面设计：张 显

ISBN 978-7-121-44472-2



9 787121 444722 >

定价：69.00 元