作者：希赛教育软考学院　来源：希赛网　2014年05月20日

考点分析

第4章　操作系统

操作系统是软件设计师的一个必考知识点，每次考试的分数在7分左右，主要涉及操作系统的5大管理功能，即进程管理、存储管理、文件管理、作业管理和设备管理。

4.1　考点分析

本节把历次考试中操作系统方面的试题进行汇总，得出本章的考点，如表4-1所示。

表4-1 操作系统试题知识点分布

考试时间	分数	考查知识点
10.11	5	虚拟存储（2）、UNIX的Shell程序（1）、PV操作（1）、银行家算法（1）
11.05	4	虚拟存储器（1）、页式存储（1）、进程调度（2）
11.11	5	树形文件目录（3）、PV操作（2）
12.05	4	进程调度（2）、UNIX设备管理（1）、文件管理（1）
12.11	5	进程调度（4）、存储管理（1）
13.05	7	进程状态（2）、虚拟设备（1）、LRU（2）、响应时间与吞吐量（1）、位示图（1）
13.11	7	响应时间和作业吞吐量（1）、银行家算法（2）、页式存储管理（1）、多级目录结构（1）、设备驱动程序（2）
14.05	7	PV操作（2）、文件系统（2）、缺页中断（2）、段式存储（1）

根据表4-1,我们可以得出操作系统的考点主要有以下几个方面：

（1）存储管理：主要考查虚拟存储器，特别是页式存储管理。

（2）进程管理：包括进程状态、PV操作、银行家算法和死锁问题等。在下午考试中，也出现过1次考查PV操作的试题，占15分。

（3）文件管理：主要考查文件的组织和结构、位示图等。

（4）作业管理：包括作业状态及转换、作业调度算法、响应时间和吞吐量等。

（5）设备管理：包括UNIX设备管理、设备驱动程序和虚拟设备等。

（6）Shell程序：主要考查UNIX的Shell程序。

对这些知识点进行归类，然后按照重要程度进行排列，如表4-2所示。其中的星号（*）代表知识点的重要程度，星号越多，表示越重要。

在本章的后续内容中，我们将对这些知识点进行逐个讲解。

表4-2 操作系统各知识点重要程度

知识点	10.11	11.05	11.11	12.05	12.11	13.05	13.11	14.05	合计	比例	重要程度
存储管理	2	2			1	2	1	3	11	25.00%	★★★★
进程管理	2	2	2	2	4	2	2	2	18	40.91%	★★★★★
文件管理			3	1			1	2	7	15.91%	★★★
作业管理						1	1		2	4.55%	★
设备管理				1		2	2		5	11.36%	★★
Shell程序	1								1	2.27%	★

存储管理

4.2 存储管理

对于本知识点，主要考查虚拟存储器，特别是页式存储管理。

所谓虚拟存储技术，即在内存中保留一部分程序或数据，在外存（硬盘）中放置整个地址空间的副本。程序运行过程中可以随机访问内存中的数据或程序，但需要的程序或数据不在内存时，就将内存中部分内容根据情况写回外存，然后从外存调入所需程序或数据，实现作业内部的局部转换，从而允许程序的地址空间大于实际分配的存储区域。它在内存和外存之间建立了层次关系，使得程序能够像访问主存一样访问外存，主要用于解决计算机主存储器的容量问题。其逻辑容量由主存和外存容量之和以及CPU可寻址的范围来决定，其运行速度接近于主存速度，成本也下降。可见，虚拟存储技术是一种性能非常优越的存储器管理技术，故被广泛地应用于大、中、小型机器和微型机中。

虚拟存储器允许用户用比主存容量大得多的地址空间来编程，以运行比主存实际容量大得多的程序。用户编程所用的地址称为逻辑地址（又称虚地址），而实际的主存地址则称为物理地址（又称实地址）。每次访问内存时都要进行逻辑地址到物理地址的转换。实际上，超过主存在实际容量的那些程序和数据是存放在辅助存储器中，当使用时再由辅存调入。地址变换以及主存和辅存间的信息动态调度是硬件和操作系统两者配合完成的。

虚拟存储器的分类

4.2.1 虚拟存储器的分类

虚拟存储器可以分为单一连续分区、固定分区、可变分区、可重定位分区、非请求页式、请求页式、段页式7种。

（1）单一连续分区。把所有用户区都分配给唯一的用户作业，当作业被调度时，进程全部进入内存，一旦完成，所有主存恢复空闲，因此，它不支持多道程序设计。

（2）固定分区。这是支持多道程序设计的最简单的存储管理方法，它把主存划分成若干个固定的和大小不同的分区，每个分区能够装入一个作业，分区的大小是固定的，算法简单，但是容易生成较多的存储器碎片。

（3）可变分区。引入可变分区后虽然主存分配更灵活，也提高了主存利用率，但是由于系统在不断地分配和回收中，必定会出现一些不连续的小的空闲区，尽管这些小的空闲区的总和超过某一个作业要求的空间，但是由于不连续而无法分配，产生了碎片。解决碎片的方法是拼接（或称紧

凑），即向一个方向（例如向低地址端）移动已分配的作业，使那些零散的小空闲区在另一方向连成一片。分区的拼接技术，一方面是要求能够对作业进行重定位，另一方面系统在拼接时要耗费较多的时间。

（4）可重定位分区。这是克服固定分区碎片问题的一种存储分配方法，它能够把相邻的空闲存储空间合并成一个完整的空区，还能够整理存储器内各个作业的存储位置，以达到消除存储碎片和紧缩存储空间的目的。紧缩工作需要花费大量的时间和系统资源。

（5）非请求页式。非请求分页式将存储空间和作业的地址空间分成若干个等分部分在分页式，要求把进程所需要的页面全部调入主存后作业方能运行，因此，当内存可用空间小于作业所需的地址空间时，作业无法运行。它克服了分区存储管理中碎片多和紧缩处理时间长的缺点，支持多道程序设计，但不支持虚拟存储。

（6）请求页式。非请求分页式将存储空间和作业的地址空间分成若干个等分部分在分页式，当进程需要用到某个页面时将该页面调入主存，把那些暂时无关的页面留在主存外。它支持虚拟存储，克服了分区存储管理中碎片多和紧缩处理时间长的缺点，支持多道程序设计，但是它不能实现对最自然的以段为单位的共享与存储保护（因为程序通常是以段为单位划分的，所以以段为单位最自然）。

（7）段页式。这是分段式和分页式结合的存储管理方法，充分利用了分段管理和分页管理的优点。作业按逻辑结构分段，段内分页，内存分块。作业只需部分页装入即可运行，所以支持虚拟存储，可实现动态连接和装配。

现在，最常见的虚存组织有分段技术、分页技术、段页式技术三种。我们把这3种存储组织总结如表4-3所示。

表4-3 常见的虚存组织

项目	段式管理	页式管理	段页式管理
划分方式	段（不定长） 每个作业一张段表	页（定长） 每个进程一张页表	先将主存分为等长页，每个作业一张段表（通常有一个基号指向它），每段对应一组页表
虚地址	（s，d），即（段号，段内偏移）	（p，d），即（页号，页内偏移）	（s，p，d）即（段号、段内页号、页内偏移）
虚实转换	段表内找出起始地址，然后+段内偏移	页表内找出起始地址，然后+页内偏移	先在段表中找到页表的起始地址，然后在页表中找到起始地址，最后+页内偏移
主要优点	简化了任意增长和收缩的数据段管理，利于进程间共享过程和数据	消除了页外碎片	结合了段与页的优点 便于控制存取访问
主要缺点	段外碎片降低了利用率	存在页内碎片	增长复杂度，增加硬件 存在页内碎片

说明：段内偏移也称为段内地址，页内偏移也称为页内地址。

例如：某页式存储系统的地址变换过程如图4-1所示。假定页面的大小为8K,图4-1中所示的十进制逻辑地址9612经过地址变换后，形成的物理地址a应为十进制多少呢？

因为8K=213,所以页内地址有13位。逻辑地址9612转换成二进制，得到10 0101 1000 1100,这里的低13位为页内偏移量，最高一位则为页号，所以逻辑地址9612的页号为1,根据图4-1的对照表，即物理块号为3（二进制形式为11）。把物理块号和页内偏移地址拼合得到110 0101 1000 1100,再转换为十进制，得到25996.

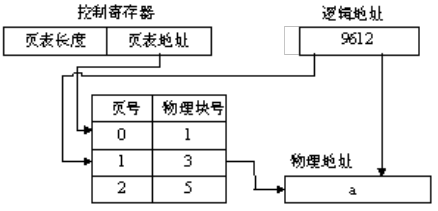

图4-1 页式存储系统的地址变换过程

在现行的虚存组织方面，最常见的就是段页式管理。在进行虚实地址转换时，可以采用的公式如下：

$$(((x)+s)+p)\times 2^{n}+d$$

其中x为基号，s为段号，p为段内页号，d为页内偏移，n的值为d的总位数，（x）表示x里的内容。

上一节　本书简介　下一节

局部性原理

4.2.2 局部性原理

虚拟存储管理的理论基础是程序的局部性原理。

程序局部性原理是指程序在执行时呈现出局部性规律，即在一段时间内，程序的执行仅限于程序的某一部分。相应地，执行所访问的存储空间也局限于某个内存区域。局部性又表现为时间局部性和空间局部性。时间局部性是指如果程序中的某条指令一旦执行，则不久以后该指令可能再次执行。如果某数据被访问，则不久以后该数据可能再次被访问。空间局部性是指一旦程序访问了某个存储单元，则不久之后，其附近的存储单元也将被访问。

根据程序的局部性理论，Denning提出了工作集理论。工作集是指进程运行时被频繁访问的页面集合。显然只要使程序的工作集全部在内存中，就可以大大减少进程的缺页次数。否则会使进程在运行中频繁出现缺页中断，从而出现频繁的页面调入/调出现象，造成系统性能下降，甚至出现"抖动".

划分工作集可以按定长时间、或定长页面两种方法进行。当颠簸现象发生时，说明系统负荷过大，通常采用处理器均衡调度，淘汰低优先级进程；另一种是控制缺页率，当缺页率达到上限时，则增加内存分配量；达到下限时，就减少内存分配量。

上一节　本书简介　下一节

虚存管理

4.2.3 虚存管理

在虚存的管理中涉及载入（调入）、放置（放入分区）和交换（swapping）等问题。

（1）调入策略：即何时将一页或一段从外存中调入主存，通常有两种策略，一种是请求调入法，即需要使用时才调入；另一种是先行调入法，即将预计要使用的页/段先行调入主存。

（2）放置策略：也就是调入后，放在主存的什么位置，这与实存管理基本上是一致的。

（3）置换策略：由于实际主存是小于虚存的，因此可能会发生内存中已满，但需要使用的页不在主存中这一情况（称为缺页中断）。这时就需要进行置换，即将一些主存中的页淘汰到外存，腾出空间给要使用的页，这个过程也称为Swapping.常见的置换算法如下：

最优算法（OPT）：淘汰不用的或最远的将来才用的页。这是一种理想算法，不可能实现，只是用来作为衡量算法效率的参照物。

随机算法（RAND）：随机淘汰。这种算法开销小，但性能不稳定。

先进先出算法（FIFO）：选择最早调入（也是驻留时间最长）的页。

最近最少使用算法（LRU）：选择离当前时刻最近的一段时间内使用得最少的页。

例如：某虚拟存储系统采用LRU页面淘汰算法，假定系统为每个作业分配3个页面的主存空间，其中一个页面用来存放程序。现有某作业的部分语句如下：

```
Var A: Array[1150,1100] OF integer;
i,j: integer;
FOR i:=1 to 150 DO
FOR j:=1 to 100 DO
A[i,j]:=0;
```

设每个页面可存放150个整数变量，变量i、j放在程序页中。初始时，程序及变量i、j已在内存，其余两页为空，矩阵A按行序存放。

在上述程序片段执行过程中，则共产生100次缺页中断（所谓缺页中断，就是指CPU所需要的数据不在内存中，需要从外存调入内存）。这是因为采用了3个页面来存储，由于第1个页面用来存放程序及i、j.所以只有2个页面用来存放数组。整个数组有150×100=15000个整数，而每一页可存放150个整数变量，所以整个程序执行完，共产生15000÷150=100次缺页。因为矩阵A按行序存放，每一行100个整数，最后保留在2个内存页面的内容是矩阵A的最后3行的数值。

上一节　本书简介　下一节

进程管理

4.3 进程管理

本部分内容主要包括进程的状态、信号量与PV操作（并发控制）、死锁问题与银行家算法等方面的知识点。

上一节　本书简介　下一节

进程的状态

4.3.1 进程的状态

一个进程从创建而产生至撤销而消亡的整个生命周期，可以用一组状态加以刻画，为了便于管理进程，把进程划分为几种状态，分别有三态模型、五态模型。

1.三态模型

按进程在执行过程中的不同状况至少定义三种不同的进程状态：

（1）运行态：占有处理器正在运行。

（2）就绪态：具备运行条件，等待系统分配处理器以便运行。

（3）等待态（阻塞态）：不具备运行条件，正在等待某个事件的完成。

一个进程在创建后将处于就绪状态。每个进程在执行过程中，任一时刻当且仅当处于上述三种状态之一。同时，在一个进程执行过程中，它的状态将会发生改变。图4-2表示进程的状态转换。

图4-2 进程三态模型及其状态转换

运行状态的进程将由于出现等待事件而进入等待状态，当等待事件结束之后等待状态的进程将进入就绪状态，而处理器的调度策略又会引起运行状态和就绪状态之间的切换。引起进程状态转换的具体原因如下：

（1）运行态→等待态：等待使用资源；如等待外设传输；等待人工干预。

（2）等待态→就绪态：资源得到满足；如外设传输结束；人工干预完成。

（3）运行态→就绪态：运行时间片到；出现有更高优先权进程。

（4）就绪态→运行态：CPU空闲时选择一个就绪进程。

2.五态模型

在三态模型中，总是假设所有的进程都在内存中。事实上，可能出现这样一些情况，例如由于进程的不断创建，系统的资源已经不能满足进程运行的要求，这个时候就必须把某些进程挂起，对换到磁盘镜像区中，暂时不参与进程调度，起到平滑系统操作负荷的目的。引起进程挂起的原因是多样的，主要有：

（1）系统中的进程均处于等待状态，处理器空闲，此时需要把一些阻塞进程对换出去，以腾出足够的内存装入就绪进程运行。

（2）进程竞争资源，导致系统资源不足，负荷过重，此时需要挂起部分进程以调整系统负荷，保证系统的实时性或让系统正常运行。

（3）把一些定期执行的进程（如审计程序、监控程序、记账程序）对换出去，以减轻系统负荷。

（4）用户要求挂起自己的进程，以便根据中间执行情况和中间结果进行某些调试、检查和改正。

（5）父进程要求挂起自己的后代子进程，以进行某些检查和改正。

（6）操作系统需要挂起某些进程，检查运行中资源使用情况，以改善系统性能；或当系统出现故障或某些功能受到破坏时，需要挂起某些进程以排除故障。

图4-3给出了具有挂起进程功能的系统中的进程状态。在此类系统中，进程增加了两个新状态：静止就绪态和静止阻塞态。为了区别，而把三态模型中的等待态改名为活跃阻塞态，就绪态改名为活跃就绪态。静止就绪态表明了进程具备运行条件但目前在二级存储器中，只有当它被对换到主存

才能被调度执行。静止阻塞态则表明了进程正在等待某一个事件且在二级存储器中。

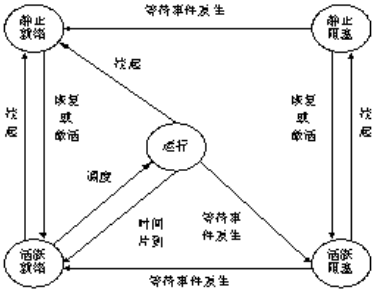

图4-3 具有挂起功能系统的进程状态及其转换

引起进程状态转换的具体原因如下：

（1）活跃阻塞态→静止阻塞态：如果当前不存在活跃就绪进程，那么至少有一个等待态进程将被对换出去成为静止阻塞态；操作系统根据当前资源状况和性能要求，可以决定把活跃阻塞态进程对换出去成为静止阻塞态。

（2）静止阻塞态→静止就绪态：引起进程等待的事件发生之后，相应的静止阻塞态进程将转换为静止就绪态。

（3）静止就绪态→活跃就绪态：当内存中没有活跃就绪态进程，或者静止就绪态进程具有比活跃就绪态进程更高的优先级，系统将把静止就绪态进程转换成活跃就绪态。

（4）活跃就绪态→静止就绪态：操作系统根据当前资源状况和性能要求，也可以决定把活跃就绪态进程对换出去成为静止就绪态。

（5）静止阻塞态→活跃阻塞态：当一个进程等待一个事件时，原则上不需要把它调入内存。但是，当一个进程退出后，主存已经有了一大块自由空间，而某个静止阻塞态进程具有较高的优先级并且操作系统已经得知导致它阻塞的事件即将结束，此时便发生了这一状态变化。

不难看出：一个挂起进程等同于不在主存的进程，因此挂起的进程将不参与进程调度直到它们被对换进主存。一个挂起进程具有如下特征：

（1）该进程不能立即被执行。

（2）挂起进程可能会等待一个事件，但所等待的事件是独立于挂起条件的，事件结束并不能导致进程具备执行条件。

（3）进程进入挂起状态是由于操作系统、父进程或进程本身阻止它的运行。

（4）结束进程挂起状态的命令只能通过操作系统或父进程发出。

（5）阻塞态：进入阻塞态通常是因为在等待I/O完成或等待分配到所需资源。

上一节　本书简介　下一节

作者：希赛教育软考学院　来源：希赛网　2014年05月20日

信号量与PV操作

4.3.2 信号量与PV操作

对于本知识点的考查，重点在于理解信号量与PV操作的基本概念，能够正确地理解在互斥、同步方面的控制应用，并能够灵活地运用，相对来说是个难点。

在操作系统中，进程之间经常会存在互斥（都需要共享独占性资源时）和同步（完成异步的两个进程的协作）两种关系。为了有效地处理这两种情况，W·Dijkstra在1965年提出信号量和PV操作。

（1）信号量：是一种特殊的变量，表现形式是一个整型S和一个队列。

（2）P操作：S=S-1,若S<0,进程暂停执行，进入等待队列。

（3）V操作：S=S+1,若S≤0,唤醒等待队列中的一个进程。

1.互斥控制

也就是为了保护共享资源，不让多个进程同时访问这个共享资源，换句话说，就是阻止多个进程同时进入访问这些资源的代码段，这个代码段称为临界区，而这种一次只允许一个进程访问的资源称为临界资源。为了实现进程互斥地进入自己的临界区，代码可以如下所示：

P（信号量）

临界区

V（信号量）

由于只允许一个进程进入，因此信号量S的初值应该为1.该值表示可以允许多少个进程进入，当S<0时，其绝对值就是等待使用临界资源的进程数，也就是等待队列中的进程数。而当一个进程从临界区出来时，执行V操作（S=S+1），如果等待队列中还有进程（S≤0），则调入一个新的进程进入（唤醒）。

2.同步控制

最简单的同步形式是进程A在另一个进程B到达L2以前，不应前进到超过点L1,这样就可以使用程序，如下所示：

进程A　　进程B

…　　…

L1:P（信号量）　　L2:V（信号量）

…　　…

因此，要确保进程B执行V操作之前，不让进程A的运行超过L1,就要设置信号量S的初值为0.这样，如果进程A先执行到L1,那么执行P操作（S=S-1）后，则S<0,就停止执行。直到进程B执行到L2时，将执行V操作（S=S+1），唤醒A以继续执行。

3.生产者-消费者问题

生产者-消费者是一个经典的问题，它不仅要解决生产者进程与消费者进程的同步关系，还要处理缓冲区的互斥关系，因此通常需要三个信号量来实现，如表4-4所示。

表4-4 生产者-消费者问题

信号量	功能类别	功能说明
empty	管理同步	说明空闲的缓冲区数量，最早没有产生东西。因此，其初始值应为缓冲区的最大数
full	管理同步	说明已填充的缓冲区数量，其初始值应为0
mutex	管理互斥	保证同时只有一个进程在写缓冲区（因此，其初始值应为1）

如果对缓冲区的读写无须进行互斥控制的话，那么就可以省去mutex信号量。

4.理解P、V操作

信号量与PV操作的概念比较抽象，在历年的考试中总是难倒许多考生，其实主要还是没有能够正确地理解信号量的含义。

（1）信号量与P、V操作是用来解决并发问题的，而在并发问题中最重要的是互斥与同步两个关系，也就是说只要有这两个关系存在，信号量就有用武之地。因此，在解题时，应该先从寻找互斥

与同步关系开始。这个过程可以套用简单互斥、简单同步、生产者-消费者问题。

（2）通常来说，一个互斥或一个同步关系可以使用一个信号量来解决，但要注意经常会忽略一些隐藏的同步关系。例如：在生产者-消费者问题中，就有两个同步关系，一个是判断是否还有足够的空间给生产者存放产物，另一个是判断是否有足够的内容让消费者使用。

（3）信号量的初值通常就是表示资源的可用数。而且通常对于初始为0的信号量，会先做V操作。

（4）在资源使用之前，将会使用P操作；在资源用完之后，将会使用V操作。在互斥关系中，P、V操作是在一个进程中成对出现的；而在同步关系中，则P、V操作一定是在两个进程甚至是多个进程中成对出现的。

5.一个例子

在考试时，可能会出现一些需要综合应用的问题，需要考生根据基本的概念，结合实际问题进行解答。

例如：在某并发系统中，有一个发送进程A、一个接收进程B、一个环形缓冲区BUFFER、信号量S1和S2.发送进程不断地产生消息并写入缓冲区BUFFER,接收进程不断地从缓冲区BUFFER取消息。假设发送进程和接收进程可以并发地执行，那么，当缓冲区的容量为N时，如何使用P、V操作才能保证系统的正常工作。发送进程A和接收进程B的工作流程如图4-4所示。请在图4-4中的（1）~（4）处填写正确的操作。

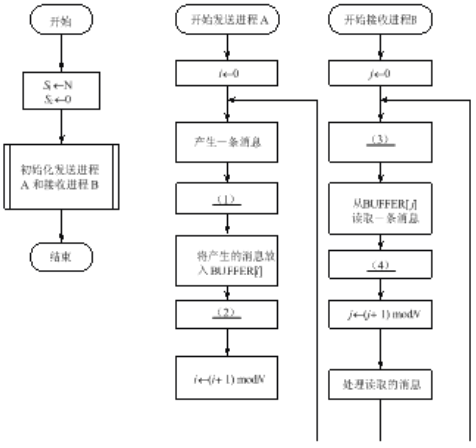

图4-4 PV操作实例一

根据题意，很显然，这是一个"生产者-消费者"问题，根据该问题的特性，通常需要3个信号量来实现：两个用来管理缓冲区同步，信号量empty表示空闲缓冲区数量，初值为缓冲区最大数N,信号量full表示已填充缓冲区数量，初值为0;一个用于管理互斥，由信号量mutex保证只有一个进程在写缓冲区，初值为1.但在本题中，进程A和进程B允许并发地访问缓冲区，因此无须管理互斥，就不需要使用信号量mutex了。因此只需定义两个信号量：S_1和S_2,初值为N的S_1在此承担的是信号量empty的功能，初值为0的S2在此则承担的是信号量full的功能。

通过这样的分析，不难得出：（1）处应该是P（S_1），将空闲缓冲区数量减1;（2）处应该是V（S_2），将已填充的缓冲区数量加1;（3）处则是P（S_2），（4）处为V（S_1）。

在这个例子的基础上，如果系统中有多个发送进程和接收进程，进程间的工作流程如图4-5所示，其中空（1）~（4）的内容与图4-4相同。发送进程产生消息并顺序地写入环形缓冲区BUFFER,接收者进程顺序地从BUFFER中取消息，且每条消息只能读取一次。为了保证进程间的正确通信，增加了信息量SA和SB.请说明信息量SA和SB的物理意义，在图4-5中的（5）和（6）处填入正确的内

容，并从图4-5的（a）~（l）中选择四个位置正确地插入P（S_A），V（S_A），P（S_B），V（S_B）。

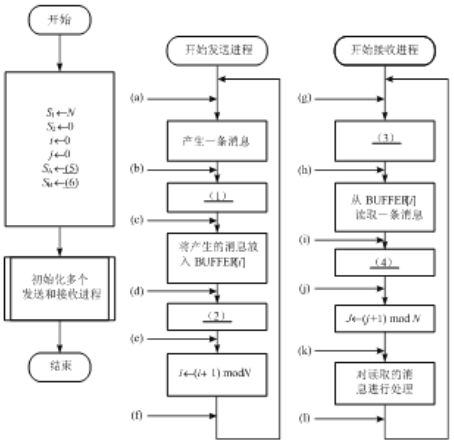

图4-5 PV操作实例二

图4-5所涉及的问题在普通的"生产者-消费者"问题上增加了一些复杂度："系统中有多个发送进程和接收进程",根据题意，我们可以得知它要完成的控制是：发送进程顺序写入，接收进程顺序读取，而且每条消息都只能够读取一次。这显然是两个互斥的问题，即多个发送进程在写缓冲区时是互斥关系，多个接收进程读缓冲区也是互斥关系。因此，信号量SA和SB分别实现这两个用来完成两个进程的互斥控制。

（1）S_A:初值为1,表示允许同时对缓冲区进行写操作的进程数量。

（2）S_B:初值为1,表示允许同时对缓冲区进行读操作的进程数量。

当然，两个对调也是可以的。在发送进程和接收进程中分别有一组信号量SA和SB的P、V操作。因此，接下来的问题就是找插入点。互斥控制的要点在于判断出临界区的范围，也就是哪部分程序必须互斥进入，否则将出现问题。根据这一点，我们可以进行如下分析。

（1）发送进程：在进程产生消息之后准备写入缓冲区时，这时就需要进行互斥判断，因此在位置（b）应插入P（S_A）；而直到完成"i=（i+1）mod N"操作后，才完成缓冲区操作，因此必须在位置（f）插入V（S_A）。

（2）接收进程：由于接收进程是负责读数据的，如果数据区是空的则应该等待，因此必须先完成P（S_2）操作，来决定其是否需要阻塞。如果没有阻塞时，再进入临界区，因此应该在位置（h）处操作P（S_B）；而"对读取的消息进行处理"已显然在临界区之外，因此应该在位置（k）插入V（S_B）。

上一节　本书简介　下一节

第 4 章：操作系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

死锁问题

4.3.3 死锁问题

死锁是指多个进程之间，互相等待对方的资源，而在得到对方资源之前又不释放自己的资源，

这样，造成循环等待的一种现象。如果一个进程在等待一个不可能发生的事，则进程就死锁了。如果一个或多个进程产生死锁，就会造成系统死锁。

1.死锁发生的必要条件

（1）互斥条件：即一个资源每次只能被一个进程使用，在操作系统中这是真实存在的情况。

（2）保持和等待条件：有一个进程已获得了一些资源，但因请求其他资源被阻塞时，对已获得的资源保持不放。

（3）不可剥夺条件：有些系统资源是不可剥夺的，当某个进程已获得这种资源后，系统不能强行收回，只能由进程使用完时自己释放。

（4）环路等待条件：若干个进程形成环形链，每个都占用对方要申请的下一个资源。

2.银行家算法

所谓银行家算法是指在分配资源之前，先看清楚，如果资源分配下去后，是否会导致系统死锁。如果会死锁，则不分配，否则就分配。

对于这些内容，关键在于融会贯通地理解与应用，为了帮助考生更好地理解，下面，我们通过一个例子来说明银行家算法的应用。

假设系统中有三类互斥资源R1、R2和R3,可用资源数分别是9、8和5.在T0时刻系统中有P1、P2、P3、P4和P5五个进程，这些进程对资源的最大需求量和已分配资源数如表4-5所示。

进程按照P1→P2→P4→P5→P3序列执行，系统状态安全吗？如果按P2→P4→P5→P1→P3的序列呢？

在这个例子中，我们先看一下未分配的资源还有哪些？很明显，还有2个R1未分配，1个R2未分配，而R3全部分配完毕。

表4-5 进程对资源的最大需求量和已分配资源数

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

按照P1→P2→P4→P5→P3的顺序执行时，首先执行P1,这时由于其R1、R2和R3的资源数都未分配够，因而开始申请资源，得到还未分配的2个R1,1个R2.但其资源仍不足（没有R3资源），从而进入阻塞状态，并且这时所有资源都已经分配完毕。因此，后续的进程都无法得到能够完成任务的资源，全部进入阻塞，形成死循环，死锁发生了。

而如果按照P2→P4→P5→P1→P3的序列执行时：

（1）首先执行P2,它还差1个R2资源，系统中还有1个未分配的R2,因此满足其要求，能够顺利结束进程，释放出2个R1、2个R2、1个R3.这时，未分配的资源就是：4个R1、2个R2、1个R3.

（2）然后执行P4,它还差一个R3,而系统中刚好有一个未分配的R3,因此满足其要求，也能够顺利结束，并释放出其资源。因此，这时系统就有5个R1、4个R2、1个R3……

根据这样的方式推下去，会发现按这种序列可以顺利地完成所有的进程，而不会出现死锁现象。从这个例子中，我们也可以体会到，死锁的四个条件是如何起作用的。只要打破任何一个条件，都不会产生死锁。

3.解决死锁的策略

（1）死锁预防："解铃还需系铃人",随便破坏导致死锁这任意一个必要条件就可以预防死锁。例

如：要求用户申请资源时一次性申请所需要的全部资源，这就破坏了保持和等待条件；将资源分层，得到上一层资源后，才能够申请下一层资源，它破坏了环路等待条件。预防通常会降低系统的效率。

（2）死锁避免：避免是指进程在每次申请资源时判断这些操作是否安全，典型算法是银行家算法。但这种算法会增加系统的开销。

（3）死锁检测：前两者是事前措施，而死锁的检测则是判断系统是否处于死锁状态，如果是，则执行死锁解除策略。

（4）死锁解除：这是与死锁检测结合使用的，它使用的方式就是剥夺。即将某进程所拥有的资源强行收回，分配给其他的进程。

上一节　本书简介　下一节

文件管理

4.4 文件管理

文件管理是对外部存储设备上的以文件方式存放的信息的管理。从历年试题来看，有涉及文件的组织和结构、位示图等的试题。

上一节　本书简介　下一节

文件的基本概念

4.4.1 文件的基本概念

本节简单介绍文件的结构、访问方式和控制块等基本概念。

1.文件的结构

文件的结构是指文件的组织形式，从用户观点所看到的文件组织形式，称为文件的逻辑结构。一般文件的逻辑结构可以分为两种，分别是无结构的字符流文件和有结构的记录文件。记录文件由记录组成，即文件内的信息划分成多个记录，以记录为单位组织和使用信息。记录文件有顺序文件、索引顺序文件、索引文件和直接文件。

文件的物理结构是指文件在存储设备上的存放方法。文件的物理结构侧重于提高存储器的利用效率和降低存取时间。文件的存储设备通常划分为大小相同的物理块，物理块是分配和传输信息的基本单位。文件的物理结构涉及文件存储设备的组织策略和文件分配策略，决定文件信息在存储设备上的存储位置。常用的文件分配策略有顺序分配（连续分配）、链接分配（串联分配）、索引分配。

2.文件的访问方式

用户通过对文件的访问（读写）来完成对文件的查找、修改、删除和添加等操作。常用的访问方法有两种，即顺序访问和随机访问。

3.文件控制块

文件控制块是系统在管理文件时所必需的信息的数据结构，是文件存在的唯一标志，简称为FCB.文件目录就是文件控制块的有序集合。FCB的内容包括相应文件的基本属性，大致可以分成4个部分。

（1）基本信息：如文件名、文件类型和文件组织等。

（2）保护信息：如口令、所有者名、保存期限和访问权限等。

（3）位置信息；如存储位置、文件长度等。

（4）使用信息：如时间信息、最迟使用者等。

第 4 章：操作系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

树形目录结构

4.4.2 树形目录结构

文件控制块的集合称为文件目录，文件目录也被组织成文件，常称为目录文件。文件管理的一个重要方面是对文件目录进行组织和管理。文件系统一般采用一级目录结构、二级目录结构和多级目录结构。DOS、UNIX、Windows系统都是采用多级树形目录结构。

在多级树形目录结构中，整个文件系统有一个根，然后在根上分枝，任何一个分枝上都可以再分枝，枝上也可以长出树叶。根和枝称为目录或文件夹。而树叶则是一个个的文件。实践证明，这种结构的文件系统效率比较高。例如：图4-6就是一个树形目录结构，其中方框代表目录，圆形代表文件。

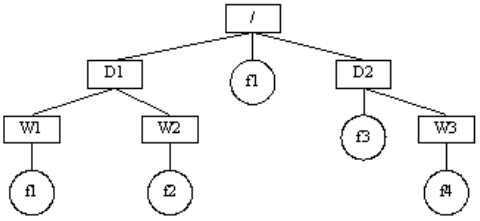

图4-6 树形文件结构

在树形目录结构中，树的根节点为根目录，数据文件作为树叶，其他所有目录均作为树的节点。系统在建立每一个目录时，都会自动为它设定两个目录文件，一个是".",代表该目录自己，另一个是"",代表该目录的父目录。对于根目录，"."和""都代表其自己。

从逻辑上讲，用户在登录到系统中之后，每时每刻都处在某个目录之中，此目录被称作工作目录或当前目录，工作目录是可以随时改变的。

对文件进行访问时，需要用到路径的概念。路径是指从树形目录中的某个目录层次到某个文件的一条道路。在树形目录结构中，从根目录到任何数据文件之间，只有一条唯一的通路，从树根开

始，把全部目录文件名与数据文件名依次用"/"连接起来，构成该数据文件的路径名，且每个数据文件的路径名是唯一的。这样，可以解决文件重名问题，不同路径下的同名文件不一定是相同的文件。例如：在图4-6中，根目录下的文件f1和/D1/W1目录下的文件f1可能是相同的文件，也可能是不相同的文件。

用户在对文件进行访问时，要给出文件所在的路径。 路径又分相对路径和绝对路径。绝对路径是指从根目录开始的路径，也称为完全路径；相对路径是从用户工作目录开始的路径。应该注意到，在树形目录结构中到某一确定文件的绝对路径和相对路径均只有一条。绝对路径是确定不变的，而相对路径则随着用户工作目录的变化而不断变化。

用户要访问一个文件时，可以通过路径名来引用。例如：在图4-6中，如果当前路径是D1,则访问文件f2的绝对路径是/D1/W2/f2,相对路径是W2/f2.如果当前路径是W1,则访问文件f2的绝对路径仍然是/D1/W2/f2,但相对路径变为/W2/f2.

在Windows系统中，有两种格式的文件，分别是FAT32（FAT16）文件和NTFS文件。NTFS在使用中产生的磁盘碎片要比FAT32少，安全性也更高，而且支持单个文件的容量更大，超过了4GB,特别适合现在的大容量存储。NTFS可以支持的分区（如果采用动态磁盘则称为卷）大小可以达到2TB.而Windows 2000中的FAT32支持分区的大小最大为32GB.

上一节　本书简介　下一节

存储空间管理

4.4.3 存储空间管理

由于文件存储设备是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问题，它包括空闲块的组织、空闲块的分配与空闲块的回收等问题。

1.空闲表法

空闲表法属于连续分配，系统为外存上的所有空闲区建立一张空闲表，每个空闲区对应一个空闲表项，包括序号、第一空闲盘块号和空闲盘块数。

2.空闲链表法

将所有空闲盘区，拉成一条空闲链，根据构成链所用的基本元素的不同，可把链表分成两种形式：

（1）空闲盘块链。将磁盘上所有空闲区空间，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链给用户。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。空闲盘块链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，可能要重复多次操作。

（2）空闲盘区链。将磁盘上所有空闲盘区拉成一条链，在每个盘区上包含若干用于指示下一个空闲盘区的指针，指明盘区大小的信息。分配盘块时，通常采用首次适应算法（显式链接法）。在回收时，要将回收区与空闲盘区相合并。

3.位示图法

位示图用二进制位表示磁盘中的一个盘块的使用情况，0表示空闲，1表示已分配。磁盘上的所有盘块都与一个二进制位相对应，由所有的二进制位构成的集合，称为位示图。

位示图法的优点是很容易找到一个或一组相邻的空闲盘块。位示图小，可以把它保存在内存中，从而节省了磁盘的启动操作。

例如：某文件管理系统在磁盘上建立了位示图，记录磁盘的使用情况。若系统中字长为32位，磁盘上的物理块依次编号为：0、1、2、…，那么8192号物理块的使用情况在位示图中的第257个字中描述。这是因为系统中字长为32位，所以每个字可以表示32个物理块的使用情况。

4.成组链接法

成组链接法将空闲表和空闲链表法结合形成的一种空闲盘块管理方法，适用于大型文件。

空闲盘块的组织如下：

（1）空闲盘块号栈。存放当前可用的一组空闲盘块的盘块号和栈中尚有的空闲盘块号数N.（N还可做栈顶指针用）。

（2）文件区中的所有空闲盘块，被分成若干组。

（3）将每一组含有的盘块总数N和该组所有的盘块号，记入其前一组的第一个盘块的s.free（0）~s.free（99）中。

（4）将第一组的盘块总数和所有的盘块号，记入空闲盘块号栈中，作为当前可供分配的空闲盘块号。

（5）最末一组只有99个盘块，其盘块号记入其前一组的s.free（1）~s.free（99）中。而在s.free（0）中则存放"0",作为空闲盘块的结束标志。

空闲盘块的分配与回收过程如下：

（1）分配过程（调用盘块分配程序完成）

检查空闲盘块号栈是否上锁，若未上锁，从栈顶取出一空闲盘块号，将与之对应的盘块分配给用户，栈顶指针下移；若该盘号已是栈底，即 s.free（0），这是当前栈中最后一个可分配的盘块号，由于在该盘块号所对应的盘块中记有下一组可用的盘块号，因此，需调用磁盘读过程，将栈底盘块号所对应盘块的内容读入栈中，作为新的盘块号栈的内容。并把原栈底对应的盘块号的盘块分配出去。然后，再分配一相应的缓冲区。最后，把栈中的空闲盘块数减1,并返回。

（2）回收过程（调用盘块回收程序完成）

将回收盘块的盘块号记入空闲盘块号栈的顶部，并执行空闲盘块数加1操作，当栈中空闲盘块数达到100时，记入新回收的盘块中，再将其盘块号作为新栈底。

管道

4.4.4 管道

管道通信方式的中间介质是文件，通常称这种文件为管道文件。两个进程利用管道文件进行通

信时，一个进程为写进程，另一个进程为读进程。写进程通过写端（发送端）往管道文件中写入信息；读进程通过读端（接收端）从管道文件中读取信息。两个进程协调不断地进行写、读，便会构成双方通过管道传递信息的流水线。

管道（命名管道）用缓冲区存储数据，普通文件用磁盘存储数据。另外，命名管道的重要作用是用于进程间的通信。在Unix系统中，管道命令可以起到输出重定向的作用。例如：

$ ps > a.txt

$ sort a.txt > b.txt

其中">"是输出重定向符号，把标准输出重定向到另一个文件，如果该文件已经存在，则覆盖。如果使用"》"符号，则把标准输出追加到另一个文件的尾部。这2条命令的作用是，首先把进程列表输出到文件a.txt中（ps的作用是显示当前正在运行的进程列表）。然后，再对文件a.txt进行排序，把排序的结果写入b.txt中。如果这2条命令接上管道，则如下：

$ ps | sort > b.txt

其中，中"|"是管道符号。这条命令的作用与前面2条命令的作用是相同的。

上一节 本书简介 下一节

作业管理

4.5 作业管理

本知识点主要考查作业状态及转换、作业调度算法、响应时间和吞吐量等。

操作系统中用来控制作业的进入、执行和撤销的一组程序称为作业管理程序，这些控制功能也能通过把作业细化，通过进程的执行来实现。

在作业管理中，系统为每一个作业建立-个作业控制块JCB.系统通过JCB感知作业的存在。JCB包括的主要内容有作业名、作业状态、资源要求、作业控制方式、作业类型以及作业优先权等。

上一节 本书简介 下一节

作业的状态

第 4 章：操作系统 作者：希赛教育软考

作业的状态

一个作业从提交给计算机系统到执行结束退出系统，一般都要经历提交、后备、执行和完成四个状态。其状态转换如图4-7所示。

（1）提交状态。作业由输入设备进入外存储器（也称输入井）的过程称为提交状态。处于提交状态的作业，其信息正在进入系统。

（2）后备状态。当作业的全部信息进入外存后，系统就为该作业建立一个作业控制块。

（3）执行状态。一个后备作业被作业调度程序选中分配了必要的资源并讲入了内存，作业调度程序同时为其建立了相应的进程后，该作业就由后备状态变成了执行状态。

图4-7 作业的状态及其转换

（4）完成状态：当作业正常运行结束，它所占用的资源尚未全部被系　　的状态为完成状态。

上一节　本书简介　下一节

处理机调度

4.5.2 处理机调度

处理机调度通常分为三级调度，即低级调度、中级调度和高级调度。

（1）高级调度。高级调度也称为作业调度。高级调度的主要功能是在批处理作业的后备作业队列中选择一个或者一组作业，为它们建立进程，分配必要的资源，使它们能够运行起来。

（2）中级调度。中级调度也称为交换调度，中级调度决定进程在内、外存之间的调入、调出。其主要功能是在内存资源不足时将某些处于等待状态或就绪状态的进程调出内存，腾出空间后，再将外存上的就绪进程调入内存。

（3）低级调度。低级调度也称为进程调度，低级调度的主要功能是确定处理器在就绪进程间的分配。

上一节　本书简介　下一节

作业调度算法

4.5.3 作业调度算法

作业调度主要完成从后备状态到执行状态的转变，及从执行状态到完成状态的转变。作业调度算法有：

（1）先来先服务（FCFS）。按作业到达的先后次序调度，它不利于短作业。

（2）短作业优先（SJF）。按作业的估计运行时间调度，估计运行时间短的作业优先调度。它不利于长作业，可能会使一个估计运行时间长的作业迟迟得不到服务。

（3）响应比高者优先（HRN）。综合上述两者，既考虑作业估计运行时间，又考虑作业等待时间，响应比是：HRN=（估计运行时间+等待时间）/估计运行时间。

（4）优先级调度。根据作业的优先级别，优先级高者先调度。

计算机系统性能指标以系统响应速度和作业吞吐量为代表。系统响应时间是指用户发出完整请求到系统完成任务给出响应的时间间隔。作业吞吐量是指单位时间内系统完成的任务量。若一个给定系统持续地收到用户提交的任务请求，则系统的响应时间将对作业吞吐量造成一定影响。若每个任务的响应时间越短，则系统的空闲资源越多，整个系统在单位时间内完成的任务量将越大；反之，若响应时间越长，则系统的空闲资源越少，整个系统在单位时间内完成的任务量将越少。

上一节　本书简介　下一节

设备管理

4.6 设备管理

本知识点包括UNIX设备管理、设备驱动程序和虚拟设备等。

上一节　本书简介　下一节

设备管理的功能

4.6.1 设备管理的功能

在计算机系统中，除了处理器和内存之外，其他的大部分硬设备称为外部设备。它包括输入/输出设备，辅存设备及终端设备等。设备管理程序一般要提供下述功能：

（1）提供和进程管理系统的接口。当进程要求设备资源时，该接口将进程要求转达给设备管理程序。

（2）进行设备分配。按照设备类型和相应的分配算法把设备和其他有关的硬件分配给请求该设备的进程，并把未分配到所请求设备或其他有关硬件的进程放入等待队列。

（3）实现设备和设备、设备和CPU等之间的并行操作。

（4）进行缓冲区管理。主要减少外部设备和内存与CPU之间的数据速度不匹配的问题，系统中一般设有缓冲区（器）来暂放数据。设备管理程序负责进行缓冲区分配、释放及有关的管理工作。

要注意的是在UNIX系统中，是把输入/输出设备当作特殊文件来处理的。

上一节　本书简介　下一节

设备驱动程序

4.6.2 设备驱动程序

设备驱动程序是一种可以使计算机和设备通信的特殊程序，可以说相当于硬件的接口，操作系统只能通过这个接口，才能控制硬件设备的工作，假如某设备的驱动程序未能正确安装，便不能正常工作。正因为这个原因，驱动程序在系统中的所占的地位十分重要，一般当操作系统安装完毕后，首要的便是安装硬件设备的驱动程序。

驱动程序的任务：首先，其作用是将硬件本身的功能告诉操作系统，接下来的主要功能就是完成硬件设备电子信号与操作系统及软件的高级编程语言之间的互相翻译。当操作系统需要使用某个硬件时，比如：让声卡播放音乐，它会先发送相应指令到声卡驱动程序，声卡驱动程序接收到后，马上将其翻译成声卡才能听懂的电子信号命令，从而让声卡播放音乐。

因此，驱动程序处在操作系统和硬件之间，与上层软件无关。

上一节　本书简介　下一节

虚拟设备

4.6.3 虚拟设备

SPOOLING（Simultaneous Peripheral Operation On Line）的意思是外部设备同时联机操作，又称为假脱机输入输出操作，采用一组程序或进程模拟一台I/O处理器。SPOOLING系统的组成如图4-8所示。

该技术利用了专门的外围控制机将低速I/O设备上的数据传送到高速设备上，或者相反。但是当引入多道程序后，完全可以利用其中的一道程序来模拟脱机输入时的外围控制机的功能，把低速的I/O设备上的数据传送到高速磁盘上；再利用另一道程序来模拟脱机输出时的外围控制机的功能，把高速磁盘上的数据传送到低速的I/O设备上。这样便可以在主机的控制下实现脱机输入、输出的功能。此时的外围操作与CPU对数据的处理同时进行，将这种在联机情况下实现的同时外围操作称为SPOOLING,或称为假脱机操作。

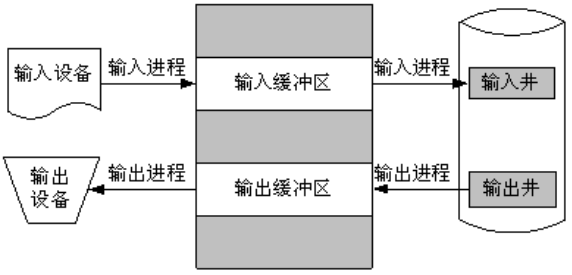

图4-8 SPOOLING系统示意图

采用假脱机技术，可以将低速的独占设备改造成一种可共享的设备，而且一台物理设备可以对应若干台虚拟的同类设备。SPOOLING系统必须有高速、大容量并且可随机存取的外存（如磁盘或磁鼓）支持。

Shell程序

4.7 Shell程序

本知识点主要考查UNIX的Shell程序的基础知识。UNIX系统有很多种产品，每种产品的命令略有不同，但基本一致。

1.Shell变量

Shell环境定义了UNIX与用户进行交互的方式。环境特性由包含名称和值的环境变量定义。以下是一些常见的缺省设置的环境变量。

（1）HOME:定义用户的主目录；CD命令的缺省目录。

（2）LOGNAME:包含用户名。

（3）MAIL:确定系统查找邮件的位置。

（4）PATH:设置系统用于查找并执行命令的目录。

（5）SHELL:决定运行的Shell.

（6）TERM:指定准备输出的终端类型。

（7）TZ:提供当前时区及其与格林威治标准时间的差值。

（8）EDITOR:确定缺省的编辑器。

（9）DISPLAY:指定窗口显示主机。

在Shell命令中，echo 命令可以显示Shell变量的内容或值。在Shell中有一些特殊意义的变量，由Shell自己管理，这些特殊变量的含义如表4-6所示。

表4-6 Shell程序中的保留字符及其含义

保留字符	含　义
$	Shell 变量名的开始
|	管道，将标准输出转到下一个命令的标准输入
#	注释开始
&	在后台执行一个进程
?	匹配一个字符
*	匹配 0 到多个字符(与 DOS 不同，可在文件名中间使用，并且含“.”)
$-	使用 set 及执行时传递给 Shell 的标志位
$!	最后一个后台进程的标识符
$#	保存程序命令行参数的数目
$*	以("$1$2…")的形式保存所有输入的命令行参数
$@	以("$1""$2"…)的形式保存所有输入的命令行参数
$?	保存前一个命令的返回码
$0	当前 Shell 的名字
$n	位置参数
$$	当前命令的进程标识符

2.重定向

在UNIX中，cat命令的功能是从命令行给出的文件中读取数据，并将这些数据直接送到标准输出。

输出重定向是指把命令（或可执行程序）的标准输出或标准错误输出重新定向到指定文件中。

这样，该命令的输出就不显示在屏幕上，而是写入到指定文件中。

输出重定向比输入重定向更常用，很多情况下都可以使用这种功能。例如：如果某个命令的输出很多，在屏幕上不能完全显示，那么将输出重定向到一个文件中，然后再用文本编辑器打开这个文件，就可以查看输出信息；如果想保存一个命令的输出，也可以使用这种方法。还有，输出重定向可以用于把一个命令的输出当作另一个命令的输入。

输出重定向的一般形式为：命令>文件名。

如果>符号后边的文件已存在，那么这个文件将被重写。 为避免输出重定向中指定文件只能存放当前命令的输出重定向的内容，Shell提供了输出重定向的一种追加手段。输出追加重定向与输出重定向的功能非常相似，区别仅在于输出追加重定向的功能是把命令（或可执行程序）的输出结果追加到指定文件的最后，而该文件原有内容不被破坏。如果要将一条命令的输出结果追加到指定文件的后面，可以使用追加重定向操作符》.形式为：命令》文件名。

上一节　本书简介　下一节

考点分析

第5章　程序设计语言

根据考试大纲，在程序设计语言方面，主要考查以下内容：

（1）汇编、编译、解释系统的基础知识和基本工作原理。

（2）程序设计语言的基本成分：数据、运算、控制和传输，程序调用的实现机制。

（3）各类程序设计语言的主要特点和适用情况。

5.1　考点分析

本节把历次考试中程序设计语言方面的试题进行汇总，得出本章的考点，如表5-1所示。

表5-1 程序设计语言试题知识点分布

考试时间	分数	考查知识点
10.11	2	非确定性有限自动机（2）
11.05	3	确定性有限自动机（2）、正规式（1）
11.11	3	词法分析（1）、有限自动机（2）
12.05	4	逻辑式语言（1）、语法与语用（2）、文法（1）
12.11	3	语言的分类（1）、自动机和正规式（2）
13.05	3	语言比较（1）、函数调用（1）、循环语句（1）
13.11	9	C语言结构体（1）、C和C++的比较（2）、动态语义（1）、语言的分类（1）、正规式（2）、全局变量（1）、上下文无关文法（1）
14.05	5	编译器（1）、文法（1）、程序语言的分类（1）、有限自动机（1）、确定性有限自动机（1）

根据表5-1,我们可以得出程序设计语言的考点主要有以下5个：

（1）基本概念：包括汇编、编译、解释系统的基础知识和基本工作原理。

（2）语言的分类：包括各种语言的特点和应用场合。

（3）控制结构：包括语句结构、函数调用、全局变量等。

（4）文法：包括文法的类型，主要考查上下文无关文法。

（5）自动机与正规式：包括非确定性有限自动机、确定性有限自动机、正规式。