第五章　虚拟存储器

5.1 虚拟存储器概述

第四章所介绍的各种存储器管理方式有一个共同的特点，即它们都要求将一个作业全部装入内存后方能运行。于是，出现了下面这样两种情况：

(1) 有的作业很大，其所要求的内存空间超过了内存总容量，作业不能全部被装入内存，致使该作业无法运行；

(2) 有大量作业要求运行，但由于内存容量不足以容纳所有这些作业，只能将少数作业装入内存让它们先运行，而将其它大量的作业留在外存上等待。

解决方案：

1. 增加内存容量

2. 逻辑上扩充内存 ←→ 虚拟存储技术

5.1 虚拟存储器概述

5.1.1 常规存储管理方式的特征和局部性原理

1. 常规存储器管理方式的特征

我们把前一章中所介绍的各种存储器管理方式统称为传统存储器管理方式，它们全都具有如下两个共同的特征：

(1) 一次性

作业必须一次性全部装入内存方能开始工作。

事实上，许多作业运行时，并非需要用到全部程序和数据。

(2) 驻留性

作业被装入内存后，整个作业都一直驻留在内存中，其中任何部分都不会被换出，直至作业运行结束。

一次性和驻留性导致不用的程序或数据占据内存空间，存在浪费。可否打破一次性和驻留性？？

5.1 虚拟存储器概述

5.1.1 常规存储管理方式的特征和局部性原理

2. 局部性原理

程序运行时存在的局部性现象，很早就已被人发现，但直到1968年，P.Denning才真正指出：程序在执行时将呈现出局部性规律，即在一较短的时间内，程序的执行仅局限于某个部分，相应地，它所访问的存储空间也局限于某个区域。

局限性又表现在下述两个方面：

(1) 时间局限性。

(2) 空间局限性。

如：程序的顺序执行

过程调用中的区域转移

循环结构

对数据结构的处理

5.1 虚拟存储器概述

5.1.1 常规存储管理方式的特征和局部性原理

3. 虚拟存储器的基本工作情况

基于局部性原理可知，应用程序在运行之前没有必要将之全部装入内存，而仅须将那些当前要运行的少数页面或段先装入内存便可运行，其余部分暂留在盘上。

5.1 虚拟存储器概述

5.1.2 虚拟存储器的定义和特征

1. 虚拟存储器的定义

虚拟存储器，是指具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统。其逻辑容量由内存容量和外存容量之和所决定，运行速度接近内存，每位成本接近外存。

2. 虚拟存储器的特征

与传统的存储器管理方式比较，虚拟存储器具有以下三个重要特征：

(1) 多次性。作业被分成多次调入内存执行。

(2) 对换性。一个作业中的程序和数据，无须在作业运行时一直常驻内存。

(3) 虚拟性。 从逻辑上扩充内存容量，使用户看到的内存容量远大于实际内存容量。

虚拟性以多次性和对换性为基础；多次性建立在离散分配的基础上。

5.1 虚拟存储器概述

5.1.3 虚拟存储器的实现方法

1. 分页请求系统

1) 硬件支持

请求调页、页面置换。主要的硬件支持有：

(1) 请求分页的页表机制。

(2) 缺页中断机构。

(3) 地址变换机构。

2) 实现请求分页的软件

5.1 虚拟存储器概述

5.1.3 虚拟存储器的实现方法

2. 请求分段系统

1) 硬件支持

主要的硬件支持有：

(1) 请求分段的段表机制。

(2) 缺段中断机构。

(3) 地址变换机构。

2) 软件支持

5.2 请求分页存储管理方式

5.2.1 请求分页中的硬件支持

为了实现请求分页，系统必须提供一定的硬件支持。计算机系统除了要求一定容量的内存和外存外，还需要有请求页表机制、缺页中断机构以及地址变换机构。

请求分页存储管理方式：每次调入和换出的基本单位都是长度固定的页面。

请求分段存储管理方式：每次调入和换出的基本单位都是长度不固定的段。

5.2 请求分页存储管理方式

5.2.1 请求分页中的硬件支持

1. 请求页表机制

在请求分页系统中需要的主要数据结构是请求页表，其基本作用仍然是将用户地址空间中的逻辑地址映射为内存空间中的物理地址。为了满足页面换进换出的需要，在请求页表中又增加了四个字段。这样，在请求分页系统中的每个页表应含以下诸项：

页号	物理块号	状态位 P	访问字段 A	修改位 M	外存地址

1）状态位（存在位）P：指示进程的该页是否已经调入内存，供进程访问时参考。

2）访问字段A：相应的页在一段时间内被访问的次数，或者本页最近已有多长时间未被访问，供页面置换算法参考。

3）修改位M：该页调入内存后是否被修改过。若修改过则写回外存；若未修改过，则使用外存中的副本，不必写回。

4）外存地址：该页在外存上的地址，通常是物理块号，供调入该页时参考。

5.2 请求分页存储管理方式

5.2.1 请求分页中的硬件支持

每当要访问的页面不在内存时，便产生一次缺页中断，请求OS将所缺之页调入内存。

步骤：① 保护CPU环境

② 分析中断原因

③ 转入缺页中断处理程序处理

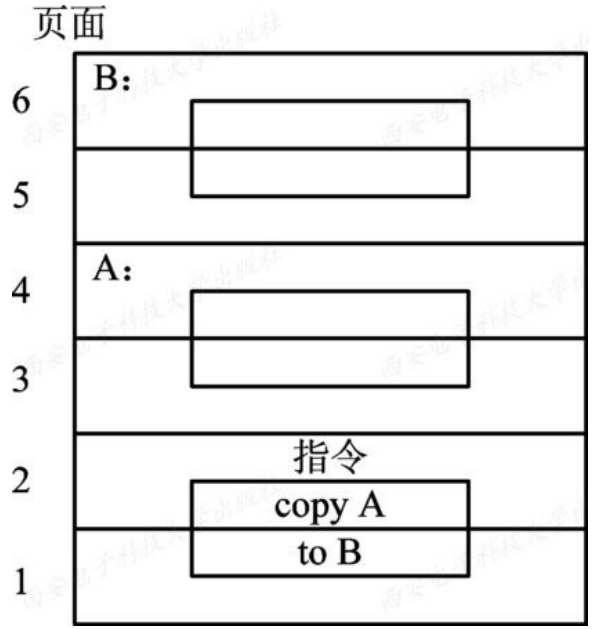
④ 恢复CPU环境

2. 缺页中断机构

(1) 在指令执行期间产生和处理中断信号。

普通中断：CPU执行一条指令结束后检查并响应中断

(2) 一条指令在执行期间可能产生多次缺页中断。如图，产生6次中断。



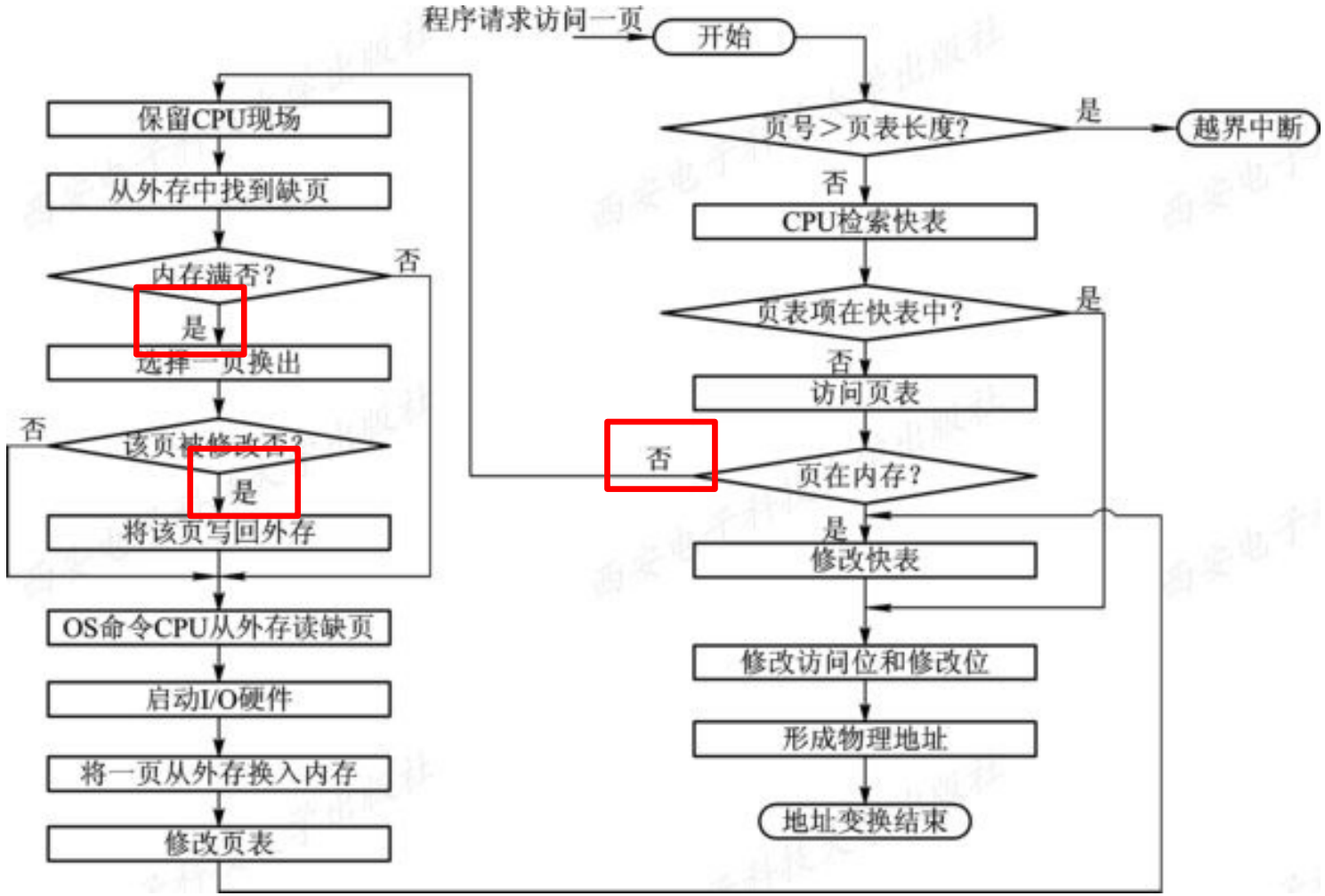
涉及6次缺页中断的指令

5.2 请求分页存储管理方式

5.2.1 请求分页中的硬件支持

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，为实现虚拟存储器，再增加了某些功能所形成的，如产生和处理缺页中断，以及从内存中换出一页的功能等等。图展示出了请求分页系统中的地址变换过程。



5.2 请求分页存储管理方式

5.2.2 请求分页中的内存分配

需要考虑的三个问题：

1. 为保证进程的正常运行，所需要的最小物理块数

2.为每个进程分配物理块时，采用何种物理块分配策略？物理块固定或者可变？

3. 物理块分配算法。如：是否平均？ 或者 按比例？

5.2 请求分页存储管理方式

5.2.2 请求分页中的内存分配

1. 为保证进程的正常运行，所需要的最小物理块数

取决于：指令格式、功能、寻址方式

2. 物理块分配策略

①固定分配局部置换。为每个进程分配一组固定数目的物理块，缺页时从已分配的页面中换出页面。难点：难以确定物理块数量。

②可变分配全局置换。为每个进程分配一定数量的物理块，在进程运行期间，若遇缺页，可从空闲物理块或者所有进程的全部物理块中换出一块，为之分配。

③可变分配局部置换。为每个进程分配一定数量的物理块，缺页时从已分配的页面中换出页面。若缺页频繁，则附加物理块；若缺页率低，则减少物理块。

3. 物理块分配算法。如：是否平均？ 或者 按比例？

5.2 请求分页存储管理方式

5.2.2 请求分页中的内存分配

3. 物理块分配算法

在采用固定分配策略时，如何将系统中可供分配的所有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，即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，即根据进程的大小按比例分配物理块。如果系统中共有n个进程，每个进程的页面数为S_i，则系统中各进程页面数的总和为：

$$S = \sum_{i=1}^{n} S_i$$

又假定系统中可用的物理块总数为m，则每个进程所能分到的物理块数为b_i可由下式计算：

$$b_i = \frac{S_i}{S} \times m$$

这里，b_i应该取整，它必须大于最小物理块数。

5.2 请求分页存储管理方式

5.2.2 请求分页中的内存分配

3. 物理块分配算法

(3) 考虑优先权的分配算法。在实际应用中，为了照顾到重要的、紧迫的作业能尽快地完成，应为它分配较多的内存空间。

通常采取的方法是把内存中可供分配的所有物理块分成两部分：

一部分按比例地分配给各进程；

另一部分则根据各进程的优先权进行分配，为高优先进程适当地增加其相应份额。

在有的系统中，如重要的实时控制系统，则可能是完全按优先权为各进程分配其物理块的。

5.2 请求分页存储管理方式

5.2.3 页面调入策略

为使进程能够正常运行，必须事先将要执行的那部分程序和数据所在的页面调入内存。现在的问题是：

(1) 系统应在何时调入所需页面；

(2) 系统应从何处调入这些页面；

(3) 是如何进行调入的。

5.2 请求分页存储管理方式

5.2.3 页面调入策略

1. 何时调入页面

(1) 预调页策略。以预测为基础的预调页策略。

(2) 请求调页策略。进程请求调入。

2. 从何处调入页面（对换区连续分配方式，速度快；文件区离散分配方式，速度慢）

(1) 系统拥有足够的对换区空间。这时可以全部从对换区调入所需页面，以提高调页速度。

(2) 系统缺少足够的对换区空间。这时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，则不必再将它们重写到磁盘(换出)，以后再调入时，仍从文件区直接调入。但对于那些可能被修改的部分，在将它们换出时便须调到对换区，以后需要时再从对换区调入。

(3) UNIX方式。 与进程有关的文件放文件区；曾经运行但被换出的页面，存放于对换区。UNIX允许页面共享，所请求的页面若由其它进程调入内存，则无需再次调入。

5.2 请求分页存储管理方式

5.2.3 页面调入策略

3. 页面调入过程

每当程序所要访问的页面未在内存时(存在位为“0”)，便向CPU发出一缺页中断，中断处理程序：

①首先保留CPU环境，分析中断原因后转入缺页中断处理程序。

②查找页表得到在外存中的物理块，若内存能容纳新页，则启动I/O，将所缺之页调入内存，修改页表；若内存已满，置换页表，并根据被置换页面的修改位判定是否写回磁盘，修改相关页表。

5.2 请求分页存储管理方式

5.2.3 页面调入策略

4. 缺页率

假设一个进程的逻辑空间为n页，系统为其分配的内存物理块数为m(m≤n)。如果在进程的运行过程中，访问页面成功(即所访问页面在内存中)的次数为S，访问页面失败(即所访问页面不在内存中，需要从外存调入)的次数为F，则该进程总的页面访问次数为A = S + F，那么该进程在其运行过程中的缺页率即为

$$f = \frac{F}{A}$$

影响缺页率的因素：

①页面大小

②进程所分配物理块的数目

③页面置换算法

④程序固有特性

5.2 请求分页存储管理方式

5.2.3 页面调入策略

4. 缺页率

在缺页中断处理时，当由于空间不足，需要置换部分页面到外存时，选择被置换页面还需要考虑到置换的代价，如页面是否被修改过。没有修改过的页面可以直接放弃，而修改过的页面则必须进行保存，所以处理这两种情况时的时间也是不同的。

假设被置换的页面被修改的概率是β，其缺页中断处理时间为t_a，被置换页面没有被修改的缺页中断时间为t_b，那么，缺页中断处理时间的计算公式为：

$$t=\beta\times t_a+(1-\beta)\times t_b$$

5.3 页面置换算法

在进程运行过程中，若其所要访问的页面不在内存，而需把它们调入内存，但内存已无空闲空间时，为了保证该进程能正常运行，系统必须从内存中调出一页程序或数据送到磁盘的对换区中。但应将哪个页面调出，须根据一定的算法来确定。通常，把选择换出页面的算法称为页面置换算法(Page-Replacement Algorithms)。置换算法的好坏将直接影响到系统的性能。

不好的置换算法产生抖动。

好的置换算法应将以后不再访问的页面换出，或将较长时间不会再访问的页面调出。

5.3 页面置换算法

5.3.1 最佳置换算法和先进先出置换算法

1. 最佳(Optimal)置换算法 （理论值，供参考）

最佳置换算法是由Belady于1966年提出的一种理论上的算法。其所选择的被淘汰页面将是以后永不使用的，或许是在最长(未来)时间内不再被访问的页面。采用最佳置换算法通常可保证获得最低的缺页率。但由于人们目前还无法预知，一个进程在内存的若干个页面中，哪一个页面是未来最长时间内不再被访问的，因而该算法是无法实现的，但可以利用该算法去评价其它算法。

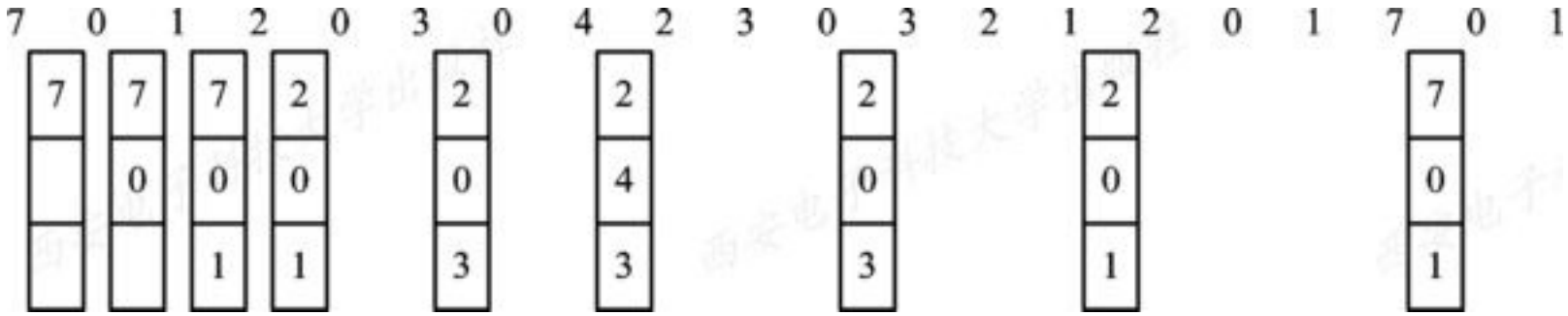


图5-3　利用最佳页面置换算法时的置换图

5.3 页面置换算法

5.3.1 最佳置换算法和先进先出置换算法

2. 先进先出(FIFO)页面置换算法

FIFO算法是最早出现的置换算法。该算法总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面予以淘汰。该算法实现简单，只需把一个进程已调入内存的页面按先后次序链接成一个队列，并设置一个指针，称为替换指针，使它总是指向最老的页面。但该算法与进程实际运行的规律不相适应，因为在进程中，有些页面经常被访问，比如，含有全局变量、常用函数、例程等的页面，FIFO算法并不能保证这些页面不被淘汰。

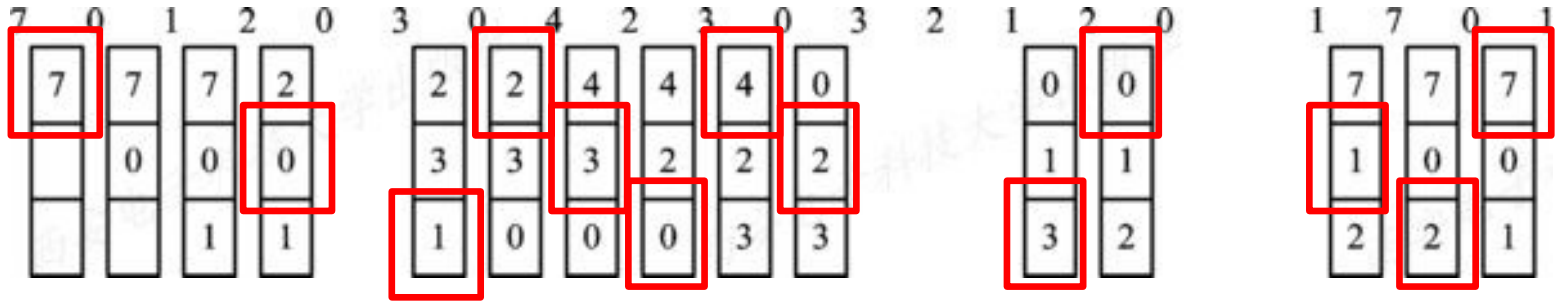


图5-4 利用FIFO置换算法时的置换图

5.3 页面置换算法

5.3.2 最近最久未使用和最少使用置换算法

1. 最近最久未使用LRU(Least Recently Used)置换算法的描述

FIFO置换算法的性能之所以较差，是因为它所依据的条件是各个页面调入内存的时间，而页面调入的先后并不能反映页面的使用情况。最近最久未使用(LRU)的页面置换算法是根据页面调入内存后的使用情况做出决策的。

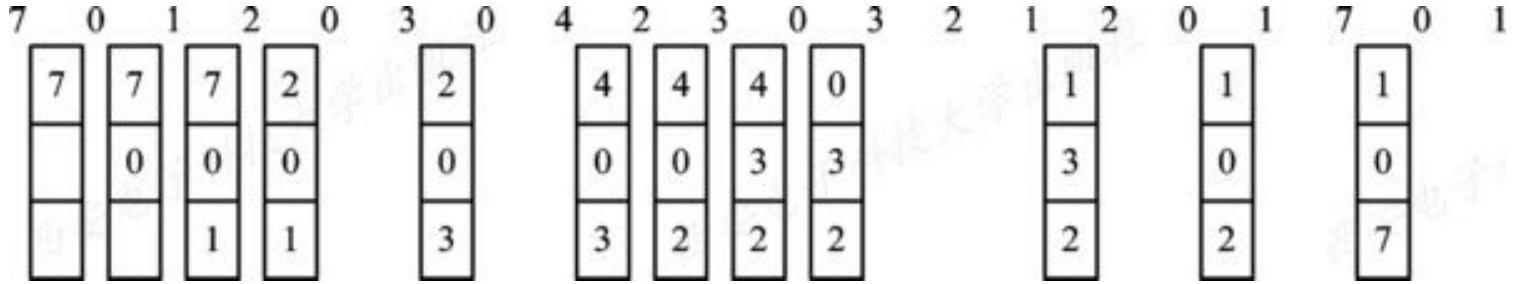


图5-5　LRU页面置换算法

5.3 页面置换算法

5.3.2 最近最久未使用和最少使用置换算法

2. LRU置换算法的硬件支持

1) 寄存器

为了记录某进程在内存中各页的使用情况，须为每个在内存中的页面配置一个移位寄存器，可表示为$R = R_{n-1}R_{n-2}R_{n-3} \ldots R_2R_1R_0$

当进程访问某物理块时，要将相应寄存器的R_{n-1}位置成1。此时，定时信号将每隔一定时间(例如100 ms)将寄存器右移一位。如果我们把n位寄存器的数看作是一个整数，那么，具有最小数值的寄存器所对应的页面，就是最近最久未使用的页面。

实页 \ R	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
1	0	1	0	1	0	0	1	0
2	1	0	1	0	1	1	0	0
3	0	0	0	0	0	1	0	0
4	0	1	1	0	1	0	1	1
5	1	1	0	1	0	1	1	0
6	0	0	1	0	1	0	1	1
7	0	0	0	0	0	1	1	1
8	0	1	1	0	1	1	0	1

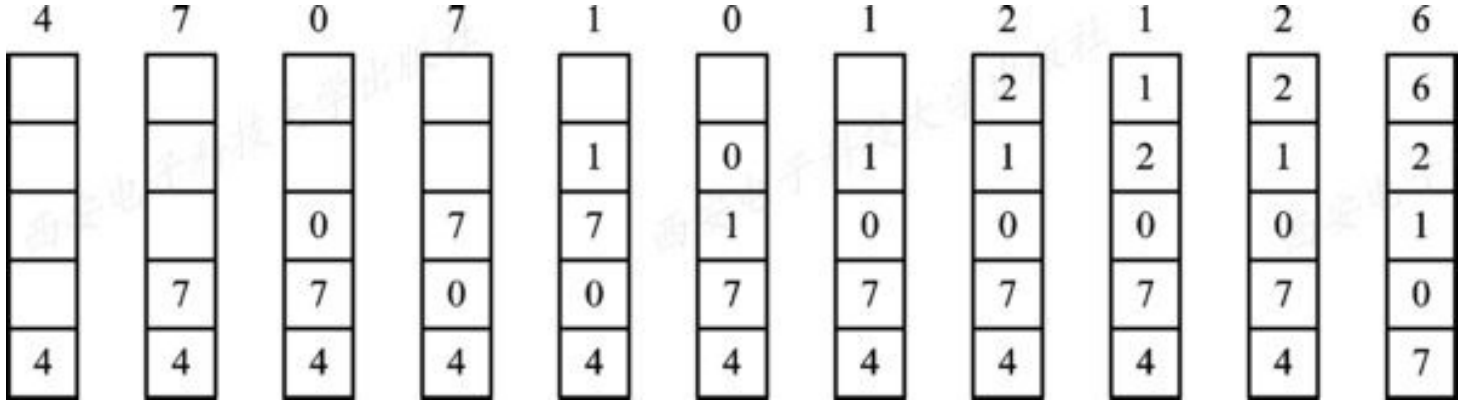
5.3 页面置换算法

5.3.2 最近最久未使用和最少使用置换算法

2. LRU置换算法的硬件支持

2) 栈

可利用一个特殊的栈保存当前使用的各个页面的页面号。每当进程访问某页面时，便将该页面的页面号从栈中移出，将它压入栈顶。因此，栈顶始终是最新被访问页面的编号，而栈底则是最近最久未使用页面的页面号。假定现有一进程，它分有五个物理块，所访问的页面的页面号序列为：4，7，0，7，1，0，1，2，1，2，6



用栈保存当前使用页面时栈的变化情况

5.3 页面置换算法

5.3.2 最近最久未使用和最少使用置换算法

3. 最少使用(Least Frequently Used，LFU)置换算法

在采用LFU算法时，应为在内存中的每个页面设置一个移位寄存器，用来记录该页面被访问的频率。该置换算法选择在最近时期使用最少的页面作为淘汰页。

5.3 页面置换算法

5.3.3 Clock置换算法

1. 简单的Clock置换算法

当利用简单Clock算法时，只需为每页设置一位访问位，再将内存中的所有页面都通过链接指针链接成一个循环队列。

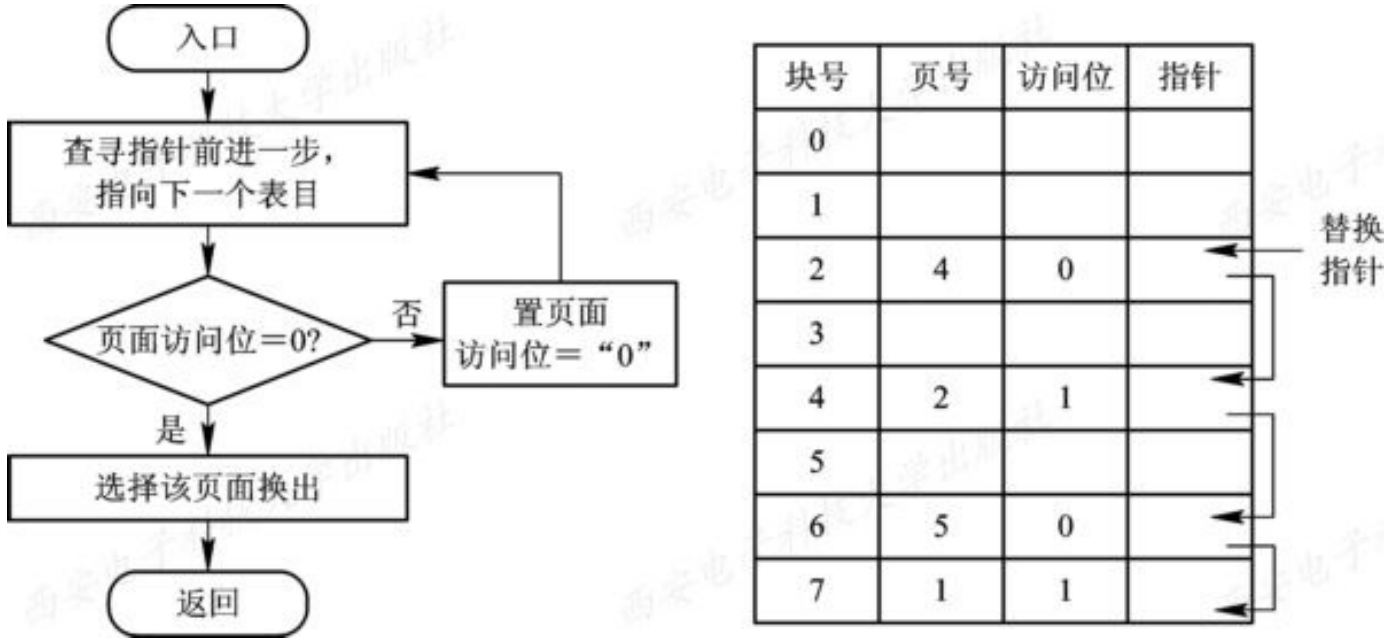


图5-8　简单Clock置换算法的流程和示例

5.3 页面置换算法

5.3.3 Clock置换算法

2. 改进型Clock置换算法

在将一个页面换出时，如果该页已被修改过，便须将该页重新写回到磁盘上；但如果该页未被修改过，则不必将它拷回磁盘。

换而言之，对于修改过的页面，在换出时所付出的开销比未修改过的页面大，或者说，置换代价大。

在改进型Clock算法中，除须考虑页面的使用情况外，还须再增加一个因素——置换代价。

1类（A=0，M=0）：未访问、未修改

2类（A=0，M=1）：未访问、已修改

3类（A=1，M=0）：已访问、未修改

4类（A=1，M=1）：已访问、已修改

依次找第一类与第二类，若找不到，则访问位A置0，继续找一类与二类。

5.3 页面置换算法

5.3.4 页面缓冲算法(Page Buffering Algorithm，PBA)

1. 影响页面换进换出效率的若干因素

(1) 页面置换算法。

(2) 写回磁盘的频率。

(3) 读入内存的频率。

2. 页面缓冲算法PBA

1) 空闲页面链表

2) 修改页面链表

5.3 页面置换算法

5.3.4 页面缓冲算法(Page Buffering Algorithm，PBA)

2. 页面缓冲算法PBA

1) 空闲页面链表。

当有一个未被修改的页面要换出时，实际上并不将它换出到外存，而是把它们所在的物理块挂在空闲链表的末尾。这些页面有数据，以后进程需要时，可从链表上直接取下，避免去外存执行I/O操作。

2) 修改页面链表。

当进程需要将一个已修改的页面换出时，系统并不立即把它换出到外存上，而是将它所在的物理块挂在修改页面链表的末尾。以降低该页面写回磁盘的频率，降低将磁盘内容读入内存的频率。

5.3 页面置换算法

5.3.4 页面缓冲算法(Page Buffering Algorithm，PBA)

2. 页面缓冲算法PBA

PBA算法的主要特点是：

① 显著地降低了页面换进、换出的频率，使磁盘I/O的操作次数大为减少，因而减少了页面换进、换出的开销；

② 正是由于换入换出的开销大幅度减小，才能使其采用一种较简单的置换策略，如先进先出(FIFO)算法，它不需要特殊硬件的支持，实现起来非常简单。

5.3 页面置换算法

5.3.5 访问内存的有效时间

与基本分页存储管理方式不同，在请求分页管理方式中，内存有效访问时间不仅要考虑访问页表和访问实际物理地址数据的时间，还必须要考虑到缺页中断的处理时间。

（1）被访问页在内存中，且其对应的页表项在快表中。

有效访问时间（EAT）=查找快表时间（λ）+访问物理地址所需时间（t）

$$EAT = \lambda + t$$

（2）被访问页在内存中，且其对应的页表项不在快表中。

有效访问时间（EAT） = 一次读取快表（ λ ）+一次读取页表（ t ）

+ 一次更新快表（ λ ） +一次读取程序或数据（t）

$$EAT = \lambda + t + \lambda + t = 2(\lambda + t)$$

5.3 页面置换算法

5.3.5 访问内存的有效时间

与基本分页存储管理方式不同，在请求分页管理方式中，内存有效访问时间不仅要考虑访问页表和访问实际物理地址数据的时间，还必须要考虑到缺页中断的处理时间。

（3）被访问页面不在内存中。

EAT = 一次读取快表（ λ ）+ 一次读取页表（ t ） + ε（缺页中断时间）
+一次读取程序或数据（ t ） + 一次更新快表（ λ ）

EAT = 2（ λ + t ）+ ε（缺页中断时间）

5.3 页面置换算法

5.3.5 访问内存的有效时间

与基本分页存储管理方式不同，在请求分页管理方式中，内存有效访问时间不仅要考虑访问页表和访问实际物理地址数据的时间，还必须要考虑到缺页中断的处理时间。

（4）被访问页面不在内存中 且 考虑命中率和缺页率。

$$EAT = \lambda + a \times t + (1-a) \times [t + f \times (\varepsilon + \lambda + t) + (1-f) \times (\lambda + t)]$$

5.3 页面置换算法

5.3.5 访问内存的有效时间

与基本分页存储管理方式不同，在请求分页管理方式中，内存有效访问时间不仅要考虑访问页表和访问实际物理地址数据的时间，还必须要考虑到缺页中断的处理时间。

（4）被访问页面不在内存中 且 考虑命中率和缺页率。

$$EAT = \lambda + a \times t + (1-a) \times [t + f \times (\varepsilon + \lambda + t) + (1-f) \times (\lambda + t)]$$

访问块表时间: λ 访问快表时间

在快表中命中，则直接访问内存：a × t

没命中的概率： （1-a）

没命中的操作： 查页表访问一次内存 t

有缺页： 中断ε +修改快表λ +访问内存t 缺页率：f

无缺页： 修改快表λ +访问内存t 不缺页概率：1-f

5.4 “抖动”与工作集

由于请求分页式虚拟存储器系统的性能优越，在正常运行情况下，它能有效地减少内存碎片，提高处理机的利用率和吞吐量，故是目前最常用的一种系统。但如果在系统中运行的进程太多，进程在运行中会频繁地发生缺页情况，这又会对系统的性能产生很大的影响，故还须对请求分页系统的性能做简单的分析。

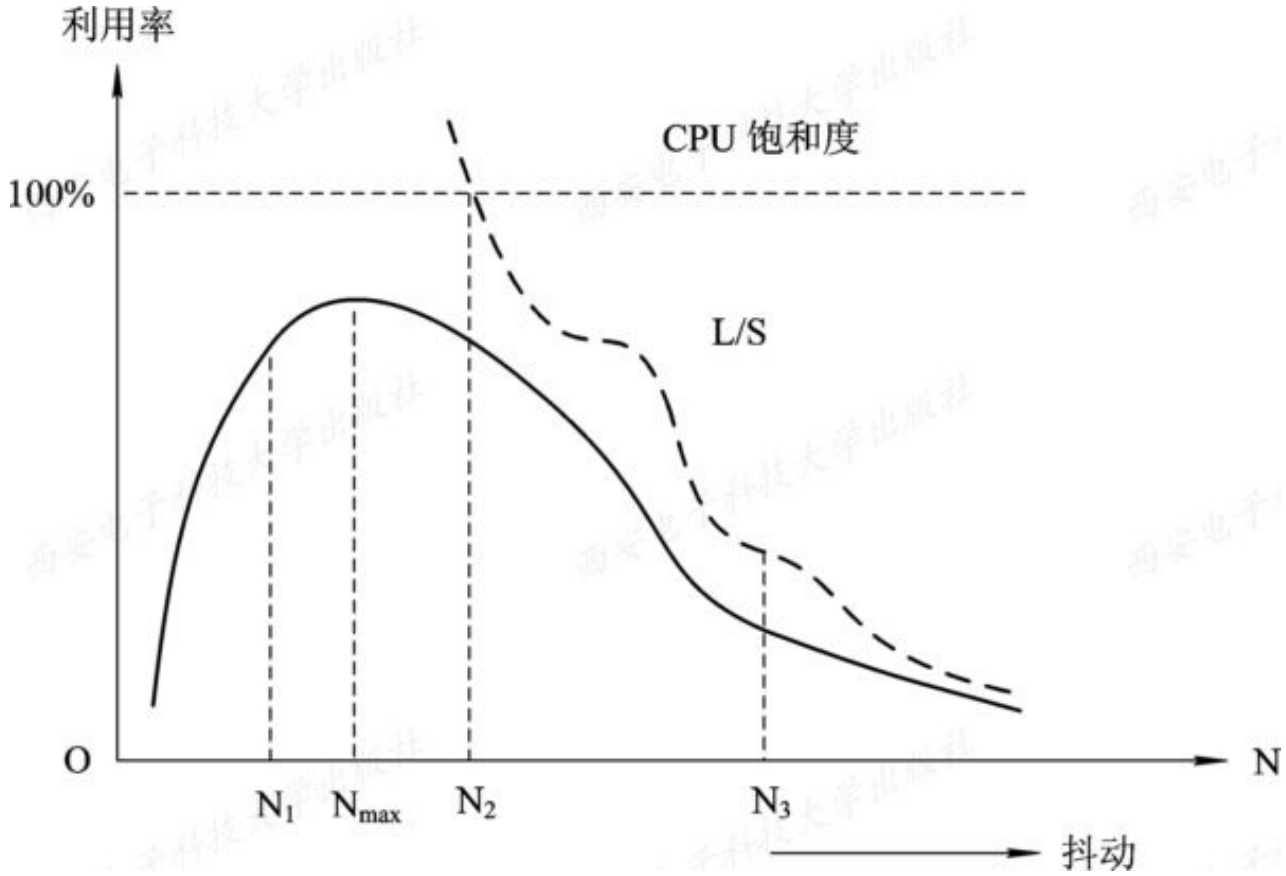
5.4 “抖动”与工作集

5.4.1 多道程序度与“抖动”

1. 多道程序度与处理机的利用率

由于虚拟存储器系统能从逻辑上扩大内存，这时，只需装入一个进程的部分程序和数据便可开始运行，故人们希望在系统中能运行更多的进程，即增加多道程序度，以提高处理机的利用率。但处理机的实际利用率却如图中的实线所示。

纵轴：CPU利用率



横轴：进程数量

5.4 “抖动”与工作集

5.4.1 多道程序度与“抖动”

2. 产生“抖动”的原因

发生“抖动”的根本原因是，同时在系统中运行的进程太多，由此分配给每一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之页调入内存。

这会使得在系统中排队等待页面调进/调出的进程数目增加。

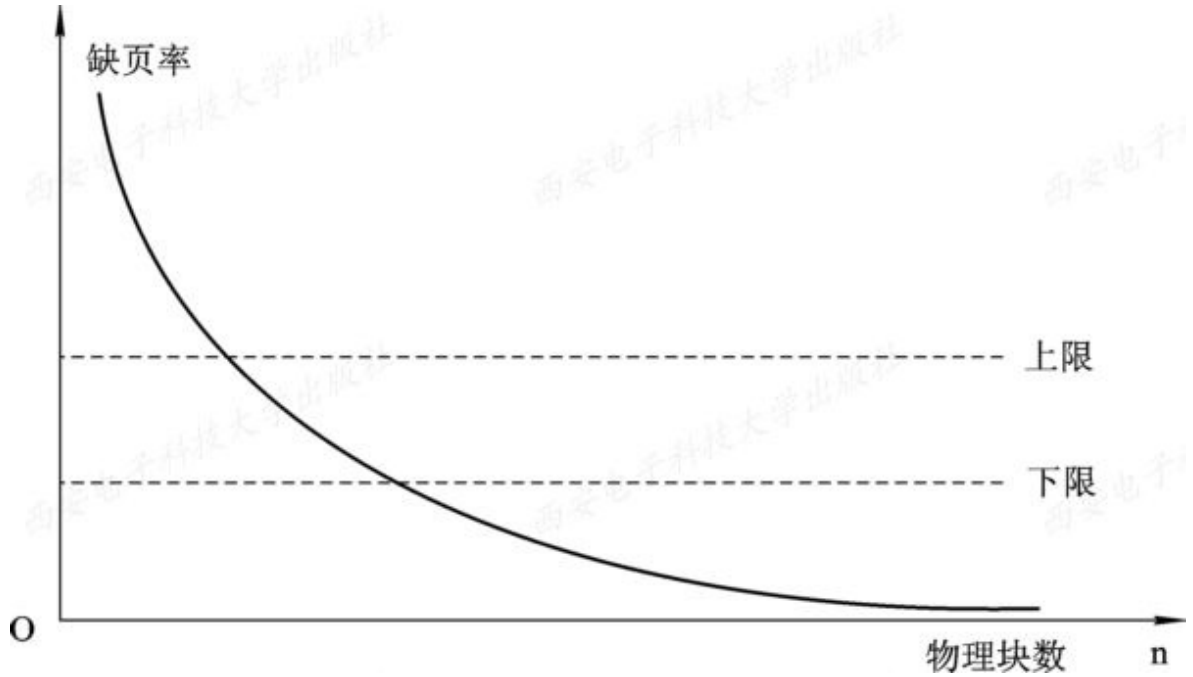
显然，对磁盘的有效访问时间也随之急剧增加，造成每个进程的大部分时间都用于页面的换进/换出，而几乎不能再去做任何有效的工作，从而导致发生处理机的利用率急剧下降并趋于0的情况。我们称此时的进程是处于“抖动”状态。

5.4 “抖动”与工作集

5.4.2 工作集

1. 工作集的基本概念

进程发生缺页率的时间间隔与进程所获得的物理块数有关。图展示出了缺页率与物理块数之间的关系。



缺页率与物理块数之间的关系

5.4 “抖动”与工作集

5.4.2 工作集

2. 工作集的定义

所谓工作集，是指在某段时间间隔Δ里，进程实际所要访问页面的集合。

Denning指出，虽然程序只需要少量的几页在内存便可运行，但为了较少地产生缺页，应将程序的全部工作集装入内存中。

然而我们无法事先预知程序在不同时刻将访问哪些页面，故仍只有像置换算法那样，用程序的过去某段时间内的行为作为程序在将来某段时间内行为的近似。

即：调入调出的参考单位是工作集在内存中，保证工作集中的页面在内存中。

5.4 “抖动”与工作集

引用页序列	窗口大小 3	4	5
24	24	24	24
15	15 24	15 24	15 24
18	18 15 24	18 15 24	18 15 24
23	23 18 15	23 18 15 24	23 18 15 24
24	24 23 18	—	—
17	17 24 23	17 24 23 18	17 24 23 18 15
18	18 17 24	—	—
24	—	—	—
18	—	—	—
17	—	—	—
17	—	—	—
15	15 17 18	15 17 18 24	—
24	24 15 17	—	—
17	—	—	—
24	—	—	—
18	18 24 17	—	—

5.4 “抖动”与工作集

5.4.3 “抖动”的预防方法

为保证系统有较大的吞吐量，必须防止“抖动”的发生。

1. 采取局部置换策略

在页面分配和置换策略中，如果采取的是可变分配方式，则为了预防发生“抖动”，可采取局部置换策略。进程局部空间内抖动，不影响整体；但频繁的I/O操作，影响其它进程的I/O操作。

2. 把工作集算法融入到处理机调度中

当调度程序发现处理机利用率低下时，它将试图从外存调入一个新作业进入内存，来改善处理机的利用率。

调入时，检查每个进程在页面驻留的程序是否足够多，若足够多则少缺页，可调入新进程；否则，不调入新作业。

5.4 “抖动”与工作集

5.4.3 “抖动”的预防方法

3. 利用“L=S”准则调节缺页率

Denning于1980年提出了“L=S”的准则来调节多道程序度，其中L是缺页之间的平均时间，S是平均缺页服务时间，即用于置换一个页面所需的时间。如果是L远比S大，说明很少发生缺页，磁盘的能力尚未得到充分的利用；反之，如果是L比S小，则说明频繁发生缺页，缺页的速度已超过磁盘的处理能力。只有当L与S接近时，磁盘和处理机都可达到它们的最大利用率。理论和实践都已证明，利用“L=S”准则，对于调节缺页率是十分有效的。

4. 选择暂停的进程

当多道程序度偏高时，已影响到处理机的利用率，为了防止发生“抖动”，系统必须减少多道程序的数目。暂停优先级低的进程，将空间分配给缺页率偏高的进程。

5.5 请求分段存储管理方式

5.5.1 请求分段中的硬件支持

为了实现请求分段式存储管理，应在系统中配置多种硬件机构，以支持快速地完成请求分段功能。与请求分页系统相似，在请求分段系统中所需的硬件支持有段表机制、缺段中断机构，以及地址变换机构。

5.5 请求分段存储管理方式

5.5.1 请求分段中的硬件支持

1. 请求段表机制

在请求分段式管理中所需的主要数据结构是请求段表。在该表中除了具有请求分页机制中有的访问字段A、修改位M、存在位P和外存始址四个字段外，还增加了存取方式字段和增补位。这些字段供程序在调进、调出时参考。下面给出请求分段的段表项。

段名	段长	段基址	存取方式	访问字段 A	修改位 M	存在位 P	增补位	外存始址

存取方式：只执行、只读、允许读/写

增 补 位：请求分段特有字段，表示本段在运行过程中是否做过动态增长

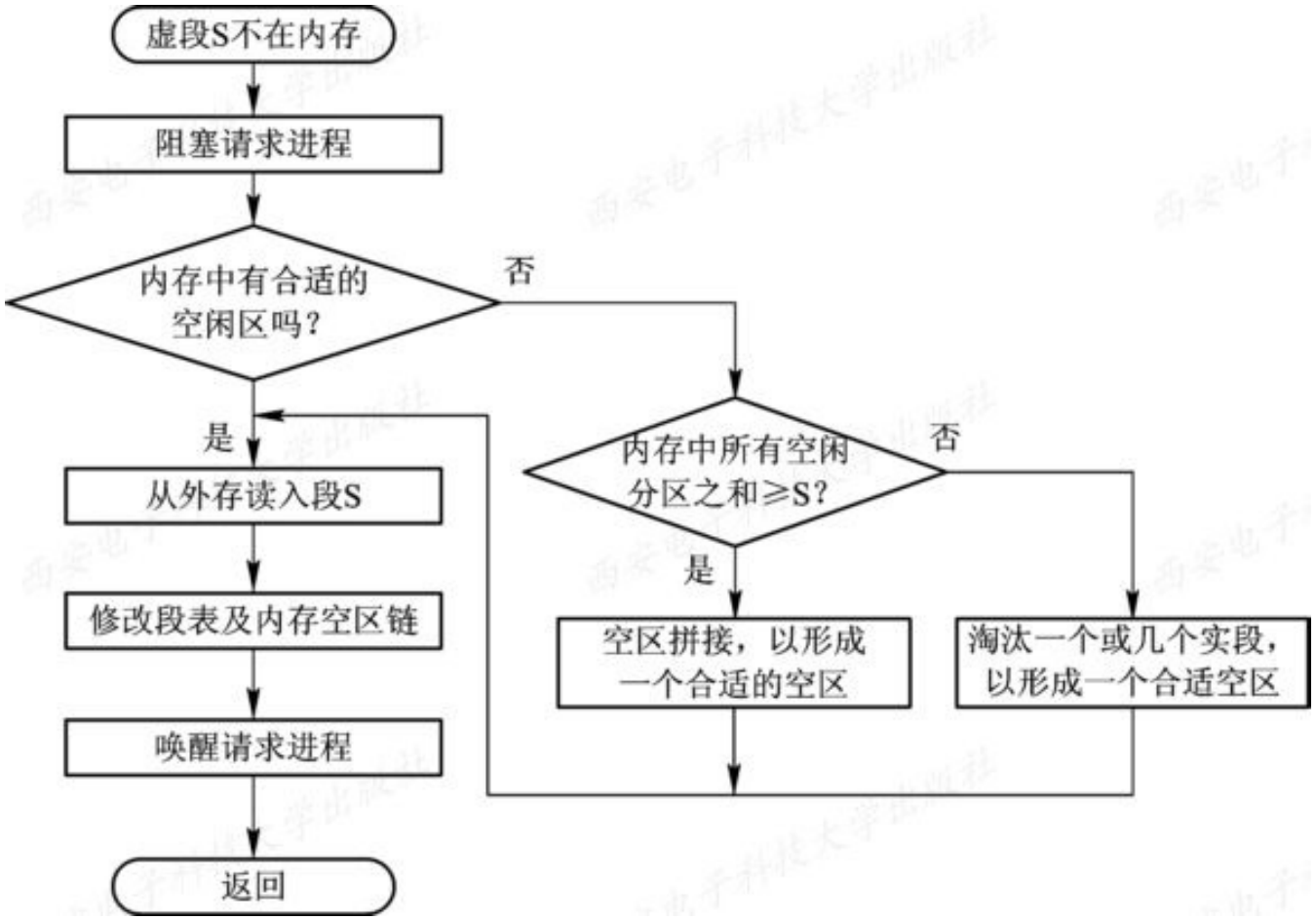
外存始址：在外存中的起始盘块号

5.5 请求分段存储管理方式

5.5.1 请求分段中的硬件支持

2. 缺段中断机构

在请求分段系统中采用的是请求调段策略。每当发现运行进程所要访问的段尚未调入内存时，便由缺段中断机构产生一缺段中断信号，进入OS后，由缺段中断处理程序将所需的段调入内存。与缺页中断机构类似，缺段中断机构同样需要在一条指令的执行期间产生和处理中断，以及在一条指令执行期间，可能产生多次缺段中断。**但由于分段是信息的逻辑单位，因而不可能出现一条指令被分割在两个分段中，和一组信息被分割在两个分段中的情况。**

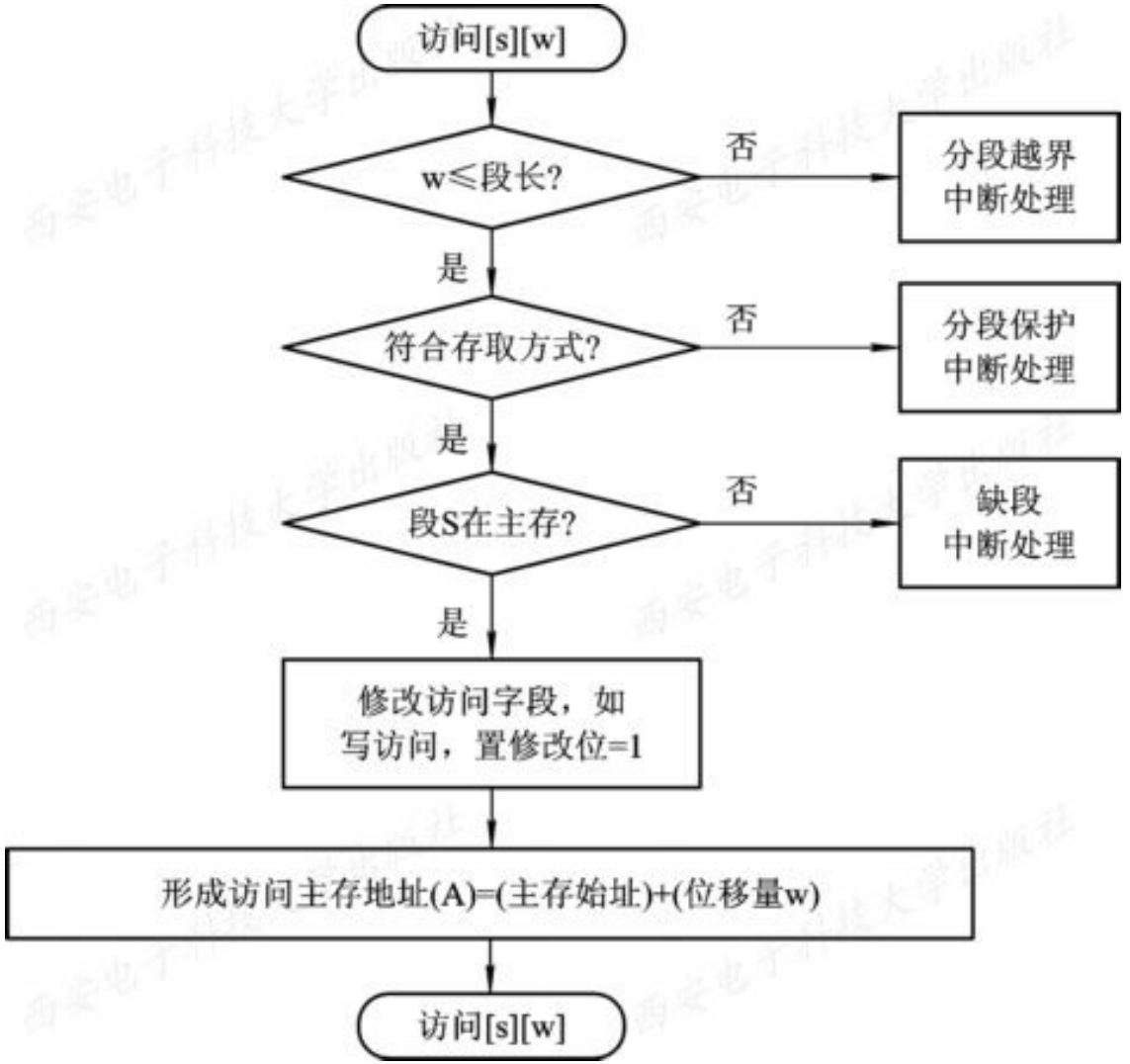


5.5 请求分段存储管理方式

5.5.1 请求分段中的硬件支持

3. 地址变换机构

请求分段系统中的地址变换机构是在分段系统地址变换机构的基础上形成的。因为被访问的段并非全在内存，所以在地址变换时，若发现所要访问的段不在内存，必须先将所缺的段调入内存，并修改段表，然后才能再利用段表进行地址变换。为此，在地址变换机构中又增加了某些功能，如缺段中断的请求及处理等。



5.5 请求分段存储管理方式

5.5.2 分段的共享与保护

分段有利于共享和保护，本部分内容介绍相应的数据结构。

1. 共享段表

(1) 共享进程计数count。

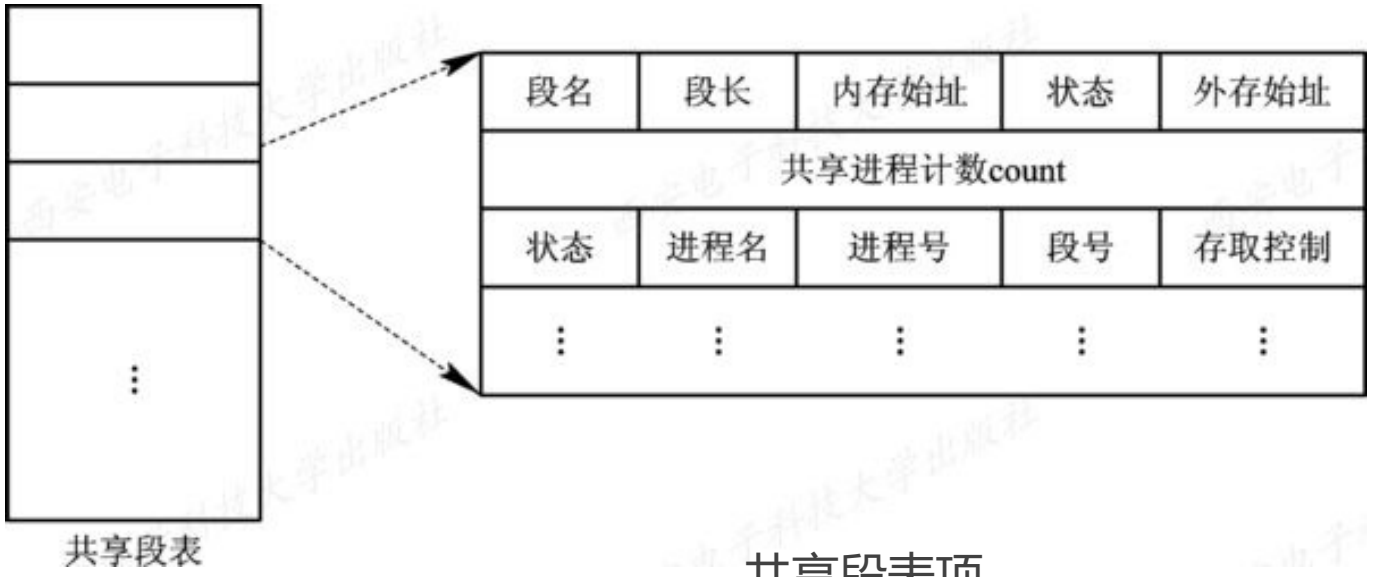
(2) 存取控制字段。

主文件：允许读写；

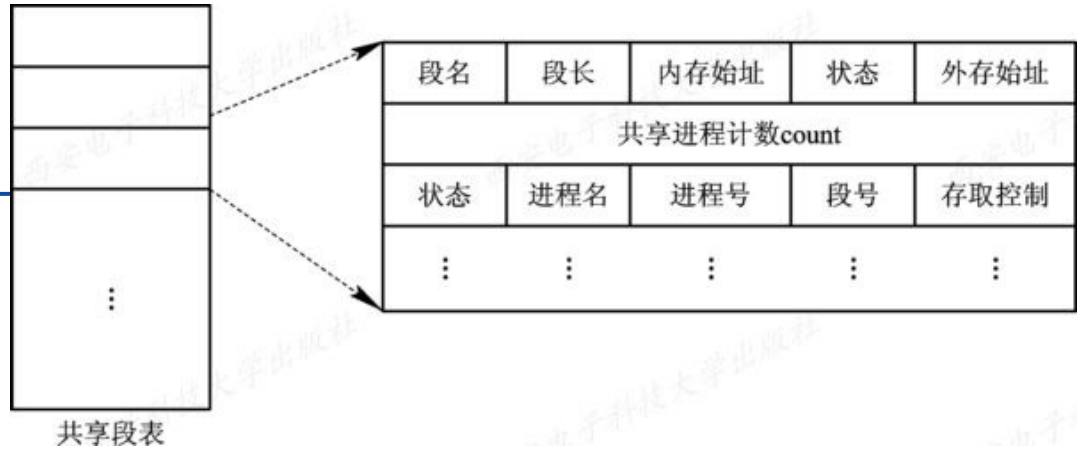
其他文件：只读或者只执行。

(3) 段号。

不同的进程中用不同的段号。



5.5 请求分段存储管理方式



5.5.2 分段的共享与保护

2. 共享段的分配与回收

1) 共享段的分配

第一个进程，将段调入内存，增加共享段表项，count=1；

后续进程，填写相关内容，count++。

2) 共享段的回收

count--；若count==0，则系统回收该段；否则，取消调用者在共享段表中的记录。

5.5 请求分段存储管理方式

5.5.2 分段的共享与保护

3. 分段保护

在分段系统中，由于每个分段在逻辑上是相对独立的，因而比较容易实现信息保护。目前，常采用以下几种措施来确保信息的安全。

1) 越界检查

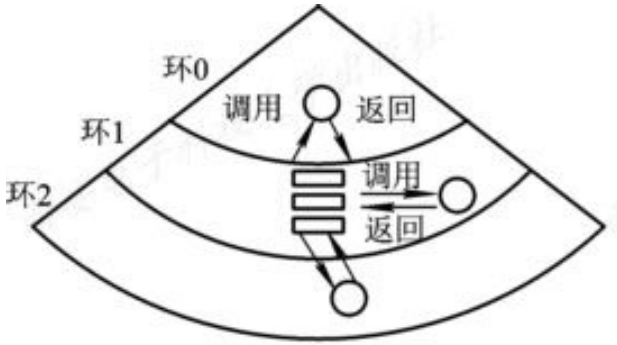
2) 存取控制检查：只读、只执行、读/写

3) 环保护机构。

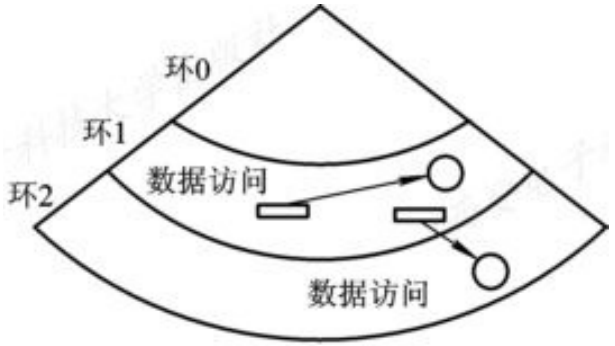
低编号的环具有较高优先级。

①一个程序可以调用驻留在相同环或较高特权环中的服务；

②一个程序可以访问驻留在相同环或较低特权环中的数据。



(a) 程序间的控制传输



(b) 数据访问

环保护机构

习题

1. 常规存储器管理方式具有哪两大特征？它对系统性能有何影响？
2. 什么是程序运行时的时间局限性和空间局限性？
3. 虚拟存储器有哪些特征？其中最本质的特征是什么？
4. 实现虚拟存储器需要哪些硬件支持？
5. 实现虚拟存储器需要哪几个关键技术？
6. 在请求分页系统中，页表应包括哪些数据项？每项的作用是什么？
7. 试比较缺页中断机构与一般的中断，它们之间有何明显的区别？
8. 试说明请求分页系统中的地址变换过程。
9. 何谓固定分配局部置换和可变分配全局置换的内存分配策略？
10. 在请求分页系统中，应从何处将所需页面调入内存？

习题

11. 试说明在请求分页系统中页面的调入过程。

12. 在请求分页系统中，常采用哪几种页面置换算法?

13. 在一个请求分页系统中，采用FIFO页面置换算法时，假如一个作业的页面走向为4、3、2、1、4、3、5、4、3、2、1、5，当分配给该作业的物理块数M分别为3和4时，试计算在访问过程中所发生的缺页次数和缺页率，并比较所得结果。

14. 实现LRU算法所需的硬件支持是什么?

15. 试说明改进型Clock置换算法的基本原理。

16. 影响页面换进换出效率的若干因素是什么?

17. 页面缓冲算法的主要特点是什么? 它是如何降低页面换进、换出的频率的?

18. 在请求分页系统中，产生“抖动”的原因是什么?

19. 何谓工作集? 它是基于什么原理确定的?

20. 当前可以利用哪几种方法来防止“抖动”？

习题

21. 试说明如何利用“L=S”准则来调节缺页率，以避免“抖动”的发生。
22. 为了实现请求分段式存储管理，应在系统中增加配置哪些硬件机构?
23. 在请求段表机制中，应设置哪些段表项?
24. 说明请求分段系统中的缺页中断处理过程。
25. 请对共享段表中的各项作简要说明。
26. 如何实现共享分段的分配和回收?

1. 分页系统中若页面较小，虽有利于（　　），但会引起（　　）；而页面较大，虽可减少（　　），但会引起（　　）。

2. 分页系统中，页表的作用是实现（　　）到（　　）的转换。

3. 在分页系统中为实现地址变换而设置了页表寄存器，其中存放了处于（　　）状态进程的（　　）和（　　）；而其他进程的上述信息则被保存在（　　）中。

4. 引入分段主要是满足用户的需要，具体包括（　　）、（　　）、（　　）、（　　）等方面。

5. 在页表中最基本的数据项是（　　）；而在段表中则是（　　）和（　　）。

6. 把逻辑地址分成页号和页内地址是由（　　）进行的，故分页系统的作业地址空间是（　　）维的；把逻辑地址分成段号和段内地址是由（　　）进行的，故分段系统的作业地址空间是（　　）维的。

7. 在段页式系统中（　　），为获得一条指令或数据，都需三次访问内存。第一次从内存中取得（　　）；第二次从内存中取得（　　）；第三次从内存中取得（　　）。

答：

1. 分页系统中若页面较小，虽有利于（**减少块内碎片**），但会引起（**页表太长**）；而页面较大，虽可减少（**页表长度**），但会引起（**块内碎片增大**）。

2. 分页系统中，页表的作用是实现（**页号**）到（**物理块号**）的转换。

3. 在分页系统中为实现地址变换而设置了页表寄存器，其中存放了处于（**执行**）状态进程的（**页表长度**）和（**页表起始地址**）；而其他进程的上述信息则被保存在（**它们的PCB**）中。

4. 引入分段主要是满足用户的需要，具体包括（**便于编程**）、（**分段共享**）、（**分段保护**）、（**动态链接**）等方面。

5. 在页表中最基本的数据项是（**物理块号**）；而在段表中则是（**段的内存基址**）和（**段长**）。

6. 把逻辑地址分成页号和页内地址是由（**机器硬件**）进行的，故分页系统的作业地址空间是（**一**）维的；把逻辑地址分成段号和段内地址是由（**程序员**）进行的，故分段系统的作业地址空间是（**二**）维的。

7. 在段页式系统中**(无快表)**，为获得一条指令或数据，都需三次访问内存。第一次从内存中取得（**页表起始地址**）；第二次从内存中取得（**块号**）；第三次从内存中取得（**指令或数据**）。

虚拟存储器最基本的特征是（A）；该特征主要是基于（B）；实现虚拟存储器最关键的技术是（C）。

A：（1）一次性　　（2）多次性　　（3）交换性　　（4）离散性　　（5）驻留性

B：（1）计算机的高速性（2）大容量的内存（3）大容量的硬盘（4）循环性原理（5）局部性原理

C：（1）内存分配　　（2）置换算法　　（3）请求调页(段)（4）对换空间管理

虚拟存储器最基本的特征是（A）；该特征主要是基于（B）；实现虚拟存储器最关键的技术是（C）。

A：（1）一次性　　**（2）多次性**　　（3）交换性　　（4）离散性　　（5）驻留性

B：（1）计算机的高速性（2）大容量的内存（3）大容量的硬盘（4）循环性原理　**（5）局部性原理**

C：（1）内存分配　　（2）置换算法　　**（3）请求调页(段)**（4）对换空间管理

分析：

虚拟存储器的基本特征是（**多次性**）和（**对换性**），因而决定了实现虚拟存储器的关键技术是（**请求调页/段**）和（**页/段置换**）。

虚拟存储器管理系统的基础是程序的局部性理论。此理论的基本含义是（A）。局部性有两种表现形式，时间局部性和（B），它们的意义分别是（C）和（D）。根据局部性理论，Denning提出了（E）。

A，B：（1）代码的顺序执行　　（2）程序执行时对主存的访问是不均匀的

（3）数据的局部性　　（4）变量的连续访问

（5）指令的局部性　　（6）空间的局部性

C，D：（1）最近被访问的单元,很可能在不久的将来还要被访问

（2）最近被访问的单元，很可能它附近的单元也即将被访问

（3）结构化程序设计，很少出现转移语句

（4）程序中循环语句的执行时间一般很长

（5）程序中使用的数据局部于各子程序。

E：　（1）chache结构的思想　（2）先进先出（FIFO页面置换算法）

（3）工作集理论　　（4）最近最久未用（LRU页面置换算法）

虚拟存储器管理系统的基础是程序的局部性理论。此理论的基本含义是（A）。局部性有两种表现形式，时间局部性和（B），它们的意义分别是（C）和（D）。根据局部性理论，Denning提出了（E）。

A，B：（1）代码的顺序执行　　**（2）程序执行时对主存的访问是不均匀的--A**

（3）数据的局部性　　（4）变量的连续访问

（5）指令的局部性　　**（6）空间的局部性--B**

C，D：**（1）最近被访问的单元,很可能在不久的将来还要被访问--C**

（2）最近被访问的单元，很可能它附近的单元也即将被访问--D

（3）结构化程序设计，很少出现转移语句

（4）程序中循环语句的执行时间一般很长

（5）程序中使用的数据局部于各子程序。

E：　（1）chache结构的思想　（2）先进先出（FIFO页面置换算法）

（3）工作集理论　　（4）最近最久未用（LRU页面置换算法）

从下列关于虚拟存储器的论述中，判断对错

(1) 在请求段页式系统中，以页为单位管理用户的虚空间，以段为单位管理内存空间。

(2) 在请求段页式系统中，以段为单位管理用户的虚空间，以页为单位管理内存空间。

(3) 为提高请求分页系统中内存的利用率，允许用户使用不同大小的页面。

(4) 在虚拟存储器中，为了能让更多的作业同时运行，通常只应装入10%~30%的作业后便启动运行。

(5) 实现虚拟存储器的最常用的算法，是最佳适应算法OPT。

(6) 由于有了虚拟存储器，于是允许用户使用比内存更大的地址空间。

6、从下列关于虚拟存储器的论述中，选出两条正确的论述。

(1) 在请求段页式系统中，以页为单位管理用户的虚空间，以段为单位管理内存空间（×）

(2) 在请求段页式系统中，以段为单位管理用户的虚空间，以页为单位管理内存空间（√）

(3) 为提高请求分页系统中内存的利用率，允许用户使用不同大小的页面（×，**页面大小相同**）

(4) 在虚拟存储器中，为了能让更多的作业同时运行，通常只应装入10%~30%的作业后便启动运行（×）

(5) 实现虚拟存储器的最常用的算法，是最佳适应算法OPT（×，**OPT是理想的算法，现实并不存在**）

(6) 由于有了虚拟存储器，于是允许用户使用比内存更大的地址空间（√）