第七章　虚拟存储管理

本章学习目标

- 虚拟存储器概念。
- 覆盖和交换技术。
- 请求分页存储管理的逻辑地址结构、段表结构、地址变换过程、缺页中断。
- 请求分页系统的页面置换算法。
- 请求分段存储管理的逻辑地址结构、段表结构、地址变换过程、缺段中断。
- 请求段页式存储管理的地址变换过程。

本章内容

- 7.1 覆盖与交换技术
- 7.2 虚拟存储管理
- 7.3 请求分页存储管理方式
- 7.4 请求分段存储管理方式
- 7.5 请求段页式存储管理方式
- 7.6 存储管理方案总结
- 7.7 Linux存储管理概述

7 虚拟内存

虚拟内存指的是将物理内存进行虚拟化。

虚拟内存主要提供了三个重要能力：

1.将主存看成一个存储在磁盘上的地址空间的高速缓存。

2.为每个进程提供了一致的地址空间，简化了内存管理

3.保护了每个进程的地址空间不受其他进程破坏

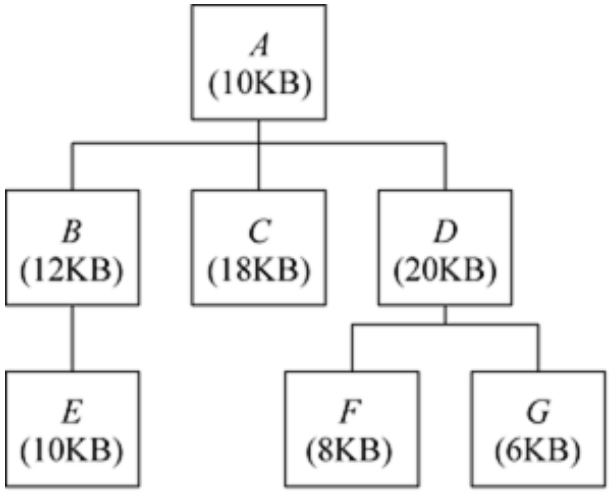
7.1 覆盖和交换技术

- 覆盖和交换技术是解决内存紧张的两种存储管理技术。
- 覆盖和交换技术对内存从逻辑上进行了扩充。
- 给用户提供一个比实际内存容量大的逻辑内存容量。

7.1 覆盖和交换技术

7.1.1 覆盖技术

- 覆盖技术是指程序运行过程中，把同一存储区在不同时刻分配给不同程序段或数据段。可相互覆盖的程序段叫覆盖段，可进行覆盖操作的内存区域叫做覆盖区。



(a) 函数调用结构图

(b) 内存分配图

7.1.2 交换技术

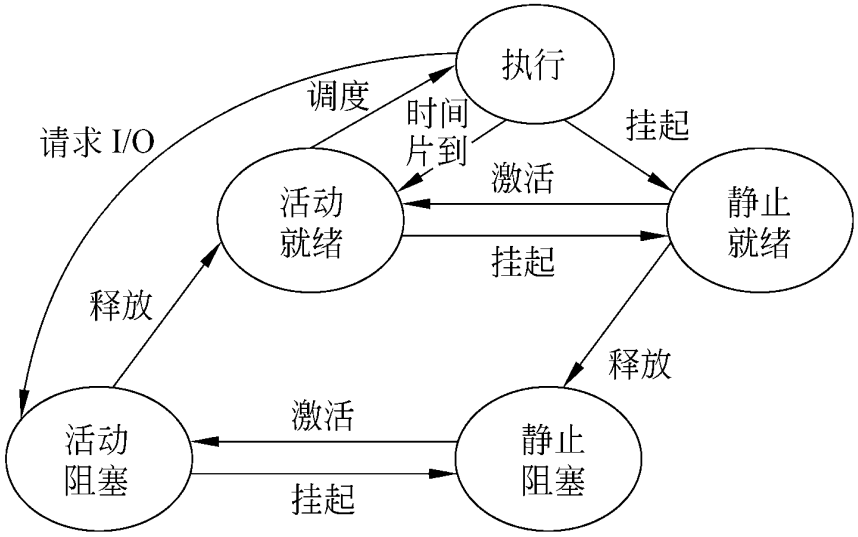
- 交换技术是系统根据需要把内存中暂时不能运行的进程或暂时不用的部分程序和数据移到外存，以便腾出足够的内存空间，把外存中已具备运行条件的进程或部分程序和数据换入，使其运行。交换是提高内存利用率的一种有效措施。
- 三级调度体系中的中级调度就是采用了交换技术。
- 为了实现交换技术，系统必须能实现2方面的功能：对换空间的管理、进程的换出与换入。

7.1.2 交换技术

1. 对换空间的管理

- 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。其与内存动态分区分配方式中所用数据结构相似，常采用空闲分区表或空闲分区链。

2. 进程的换出与换入



7.1 存储管理概述

- 交换技术在使用上要注意以下4点：

（1）要有足够大、足够快的备份存储空间，供系统进行交换操作时直接访问。

（2）必须确保换出进程处于非运行状态。

（3）内存空间紧张时进行交换，当内存空间不紧张时暂停交换。

（4）为了提高CPU的利用率，系统通常要保证进程的执行时间比进程的交换时间要长。

- Linux查看swap空间大小：swapon -s

7.2 虚拟存储管理

7.2.1 程序局部性原理

局部性原理表现在下述两个方面：

① 时间局部性

一条语句执行，不久后有可能接着执行。部分代码频繁执行。

② 空间局部性

一条数据被访问后，相邻的数据块有较大可能后续会接着访问。

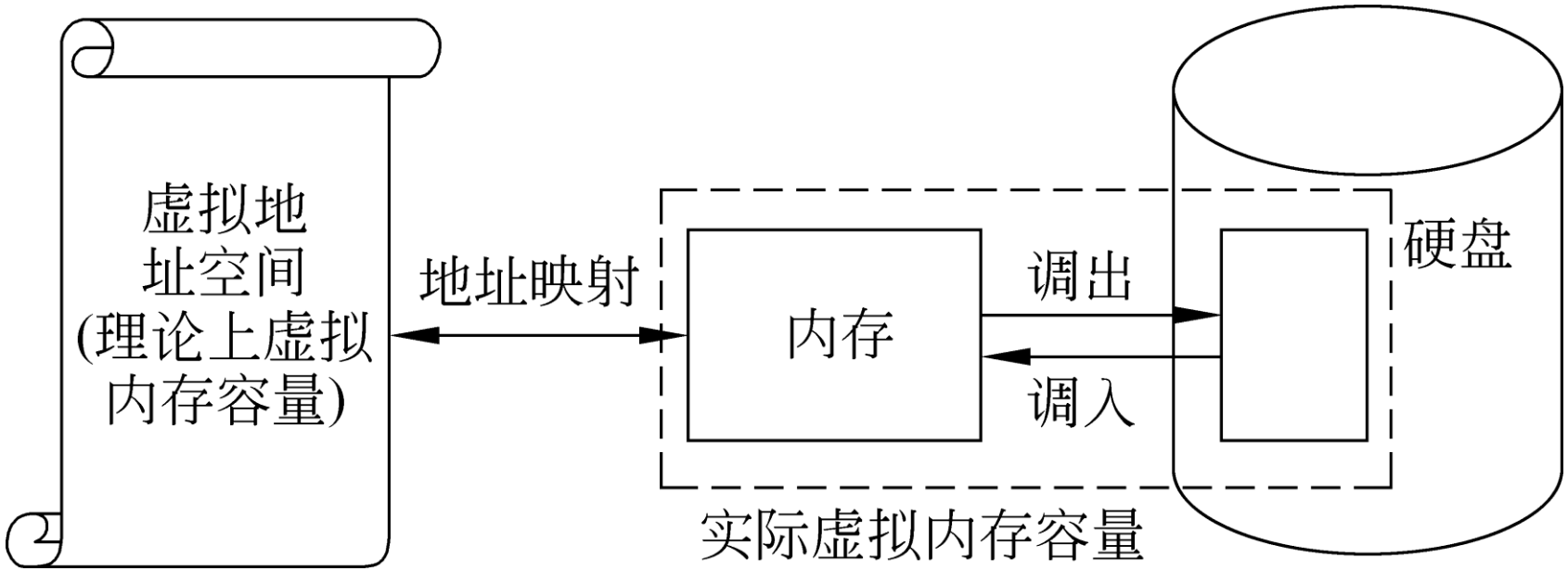
7.2.1 程序局部性原理

2. 虚拟存储器的引入

- 虚拟存储器中存储的进程执行时并不把其全部内容装入内存，而只将其中一部分先装入内存。进程执行过程中用到那些不在内存中的信息时，再把它们换入内存。
- 虚拟存储器容量=物理内存容量+辅存中用于虚存的容量

7.2.1 程序局部性原理

2. 虚拟存储器的引入



7.2.2 虚拟存储器及其特征

1. 虚拟存储器的定义

虚拟存储器的基本思想是：

- 运行之前仅将那些当前要运行的少数页面或段先装入内存便可运行，其余部分放在磁盘上。
- 程序在运行时，如果它所要访问的页（段）已调入内存，便可继续执行下去；但如果程序所要访问的页（段）尚未调入内存（称为缺页或缺段），此时程序应利用操作系统提供的请求调页（段）功能，将它们调入内存，以使进程能继续执行下去。
- 如果此时内存已满，无法再装入新的页（段），则还须再利用页（段）的置换功能，将内存中暂时不用的页（段）调至磁盘上，在腾出足够内存空间后，将要访问的页（段）调入内存，使程序继续执行。

7.2.2 虚拟存储器及其特征

2. 虚拟存储器的特征

① 多次性

一个作业被分成多次调入内存运行。多次性是虚拟存储器最重要的特征，与常规存储器管理的一次性相对应。

② 对换性

系统允许作业在运行过程中进行换进、换出操作。换进和换出能有效地提高内存利用率。

③ 虚拟性

虚拟性是指从逻辑上扩充内存容量，并非实际存在。用户感觉到的很大的虚拟存储容量实际上是一种“假象”

7.3 请求分页存储管理方式

7.3.1 请求分页中的硬件支持

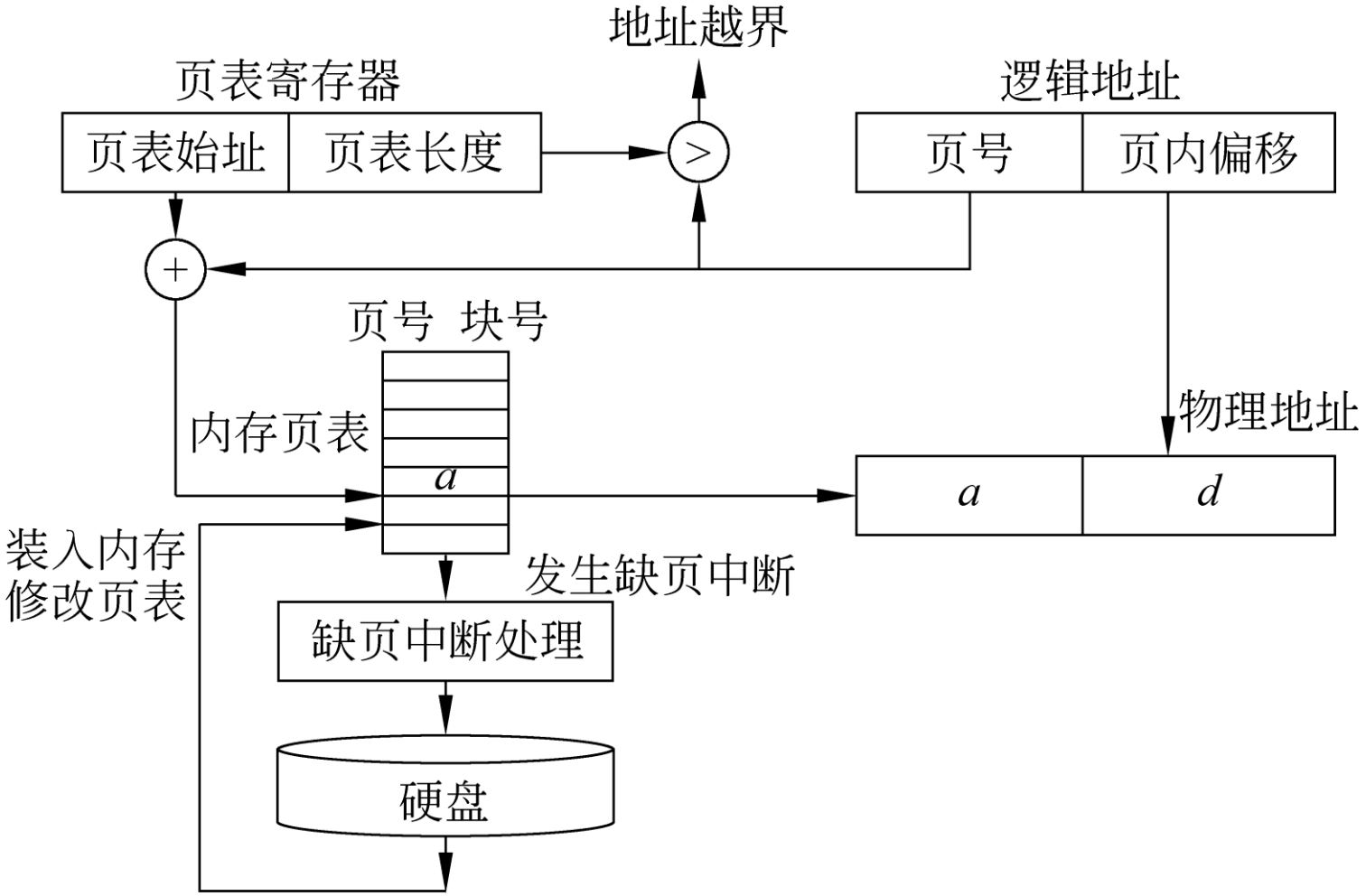
1. 页表

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

- 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，如不存在，发送缺页中断。
- 访问字段 A 用于记录一段时间内被访问次数，用于页面置换算法参考
- 修改位 M 表示该页调入内存后，是否修改过，供是否回写硬盘参考
- 外存地址 D ： 该页在外存上的地址

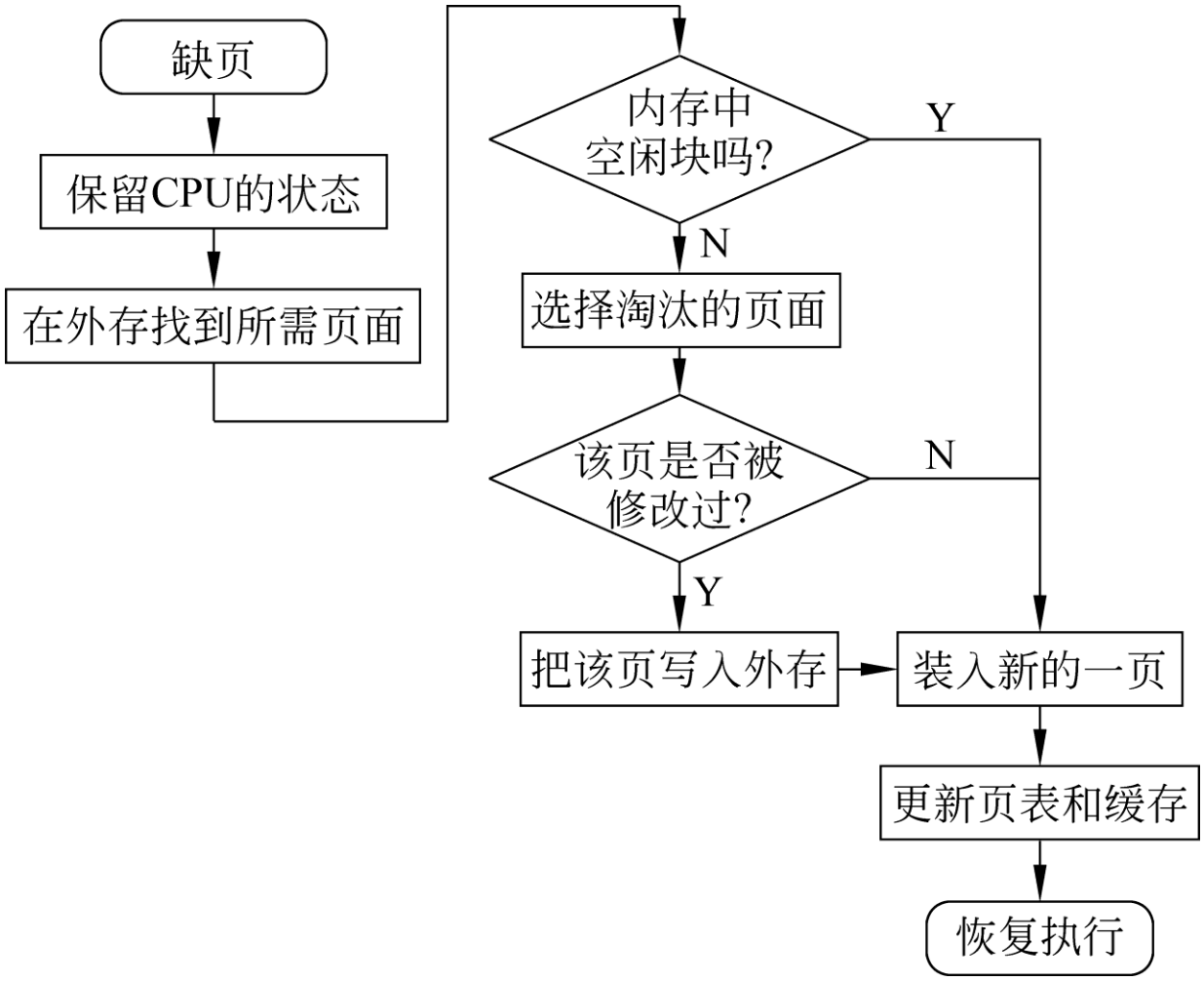
7.3.1 请求分页中的硬件支持

2. 请求分页管理的地址变换机构



7.3.1 请求分页中的硬件支持

3. 缺页中断机构



7.3.2 请求分页中的软件支持

1. 物理块分配算法

- 在采用固定分配策略时，可采用以下几种物理块分配方法。

① 平均分配算法：将系统中所有可供分配的物理块，平均分配给各进程。

② 按比例分配算法：根据进程大小按比例分配。

③ 考虑优先权的分配算法：一部分按照比例分配给各进程；另一进程根据优先权分配。

7.3.2 请求分页中的软件支持

2. 最小物理块数的确定

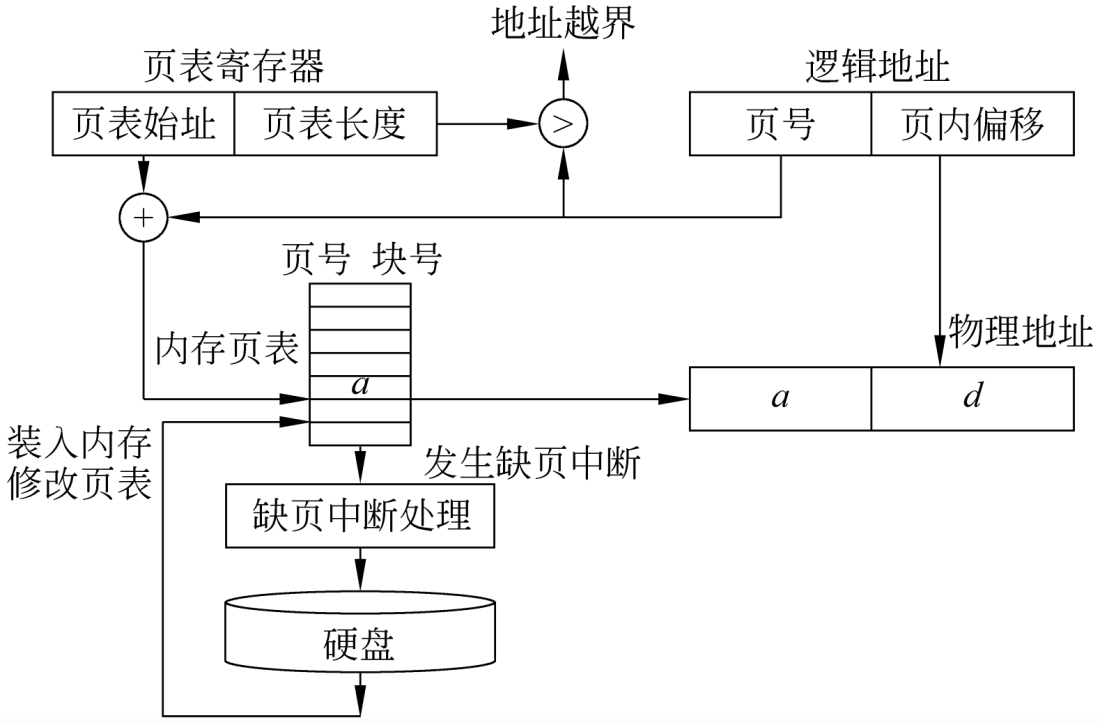
- 进程占用的存储容量越小，缺页率就越大。在为进程分配物理块时，首先应该考虑的问题是保证进程正常运行所需的最少物理块数（称为最小物理块数）。

3. 对换区管理

- 在虚拟存储系统中，将外存分为文件区和对换区。文件区存放用户文件；对换区存放换入换出的页面或段。

7.3.3 页面置换算法

页面置换算法：当进程产生缺页中断时，若内存已无空闲空间时，为了保证该进程能正常运行，系统必须依据一定的算法从内存中选择某页数据或指令送到外存交换区中，所采用的算法称为页面置换算法。算法的好坏直接影响系统整体性能。



7.3.3 页面置换算法的衡量指标

假定进程Pi共有m页，系统分配给它的物理块数为n，这里 m>n。开始时，内存没有装入任何信息。假设进程Pi在运行中成功访问的次数为S，不成功的访问次数为F(即产生缺页中断的次数)，作业执行过程中总访问次数为A。

相关指标如下：

总访问次数 A = S+F ; 置换次数 R = F-n；缺页率： f = F/A;

命中率 s = S/A；置换率 r = R/A

7.3.3 页面置换算法

1. 最佳置换算法

- 最佳置换算法是一种理想化的页面置换算法，保证最低的缺页率。其所选择的被淘汰页面将是以后永不使用的，或是在将来很长时间内不再被访问的页面。

7.3.3 页面置换算法

1. 最佳置换算法

- 某进程在一段时间内的页面访问串为：**1,2,3,4,2,6,2,1,2,3,7,6,3,2,1,2,3,6**。假定只有**4**个物理块，求采用最佳置换算法，缺页中断，置换次数与缺页率和置换率。

时刻	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
引用串	1	2	3	4	2	6	2	1	2	3	7	6	3	2	1	2	3	6
物理块	1	1	1	1	1	1	1	1	1	1	7	7	7	7	1	1	1	1
		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
			3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
				4	4	6	6	6	6	6	6	6	6	6	6	6	6	6
缺页标记	+	+	+	+		+					+				+			

7.3.3 页面置换算法

2. 先进先出FIFO置换算法

- 这种算法的出发点是先装入内存者先被置换。其总是先淘汰那些驻留在内存时间最长的页面。该算法实现简单，可采用一个先进先出的队列实现，不需要硬件的支持

表 6.2　FIFO 置换算法的进程运行情况图

时刻	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
引用串	1	2	3	4	2	6	2	1	2	3	7	6	3	2	1	2	3	6
物理块	1	1	1	1	1	6	6	6	6	6	7	7	7	7	7	7	3	3
		2	2	2	2	2	2	1	1	1	1	6	6	6	6	6	6	6
			3	3	3	3	3	3	2	2	2	2	2	2	1	1	1	1
				4	4	4	4	4	4	3	3	3	3	3	3	2	2	2
缺页标记	+	+	+	+		+		+	+	+	+	+			+	+	+	

7.3.3 页面置换算法

3. 最近最久未使用（LRU）置换算法

- 最近最久未使用的页面置换算法是根据页面调入内存后的使用情况进行决策的。由于无法知道各页面将来的使用情况，只能利用“最近的过去”作为“最近的将来”的近似。

1. 计时法。系统为内存中的每个页面设置一个计时器，用来记录该页面自上次访问以来所经历的时间。淘汰页面时，选择计时器值最大的页面。
2. 移位寄存器法。每个页面设置一个移位寄存器，初值为0。当进程访问某页时，将该页对应的寄存器最高位置为1，每隔一段时间，将寄存器右移一位。淘汰寄存器值最小的页面。
3. 堆栈法。每当访问一次页面时就调整一次，栈顶始终是最新被访问页面的页号，栈底是最近最久未使用的页号。当发生缺页中断时，总是淘汰栈底页号对应的页面。

7.3.3 页面置换算法

3. 最近最久未使用（LRU）置换算法

表 6.3　LRU 置换算法的进程运行情况图

时刻	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
引用串	1	2	3	4	2	6	2	1	2	3	7	6	3	2	1	2	3	6
物理块	1	1	1	1	1	6	6	6	6	6	7	7	7	7	1	1	1	1
		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
			3	3	3	3	3	1	1	1	1	6	6	6	6	6	6	6
				4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
缺页标记	+	+	+	+		+		+		+	+	+			+			

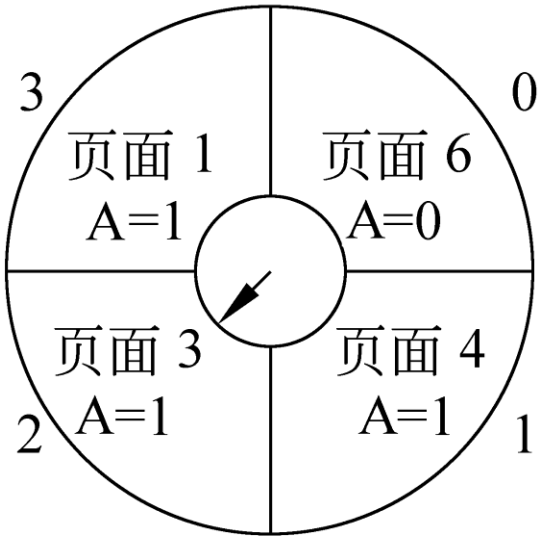
7.3.3 页面置换算法

4. 简单时钟（Clock）置换算法

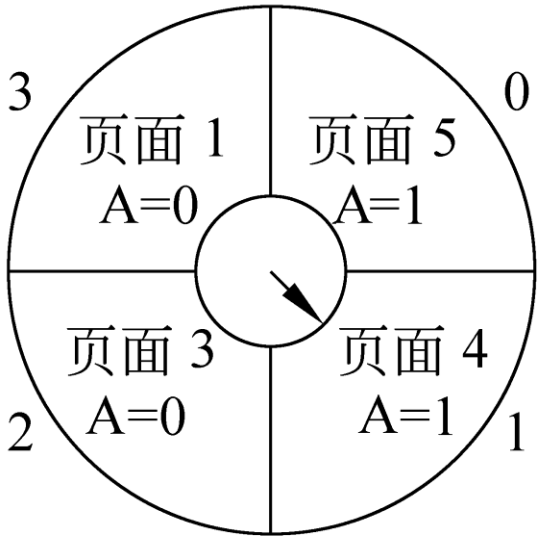
- Clock算法是LRU的近似算法。
- 需要用到页表项当中的访问位，当一个页面被装入内存时，把该位初始化为0。如果这个页面被访问（读/写），则把该位置为1。
- 把各个页面组织成环形链表（类似钟表面），把指针指向最老的页面（FIFO）
- 当发生一个缺页中断时，考察指针所指向的最老页面，若它的访问位为0，立即淘汰；若访问位为1，则把该位置为0，然后指针往下移动一格。如此下去，直到找到被淘汰的页面，然后把指针移动到它的下一格。

7.3.3 页面置换算法

Clock置换算法示例



(a) 置换前



(b) 置换后

7.3.3 页面置换算法

Clock置换算法

表 6.5 Clock 置换算法的进程运行情况图

时刻	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
引用串	1	2	3	4	2	6	2	1	2	3	7	6	3	2	1	2	3	6
物理块	1	1	1	1	1	6	6	6	6	6	7	7	7	7	7	7	3	3
		2	2	2	2	2	2	2	2	2	2	6	6	6	6	6	6	6
			3	3	3	3	3	1	1	1	1	1	1	2	2	2	2	2
				4	4	4	4	4	4	3	3	3	3	3	1	1	1	1
缺页标记	+	+	+	+		+		+		+	+	+		+	+		+	

7.3.3 页面置换算法

5. 改进型时钟（Clock）置换算法

- 在改进型Clock算法中，除须考虑页面的使用情况外，还须再增加一个页面修改因素，即置换代价。其在选择页面换出时，既要是未使用过的页面（访问位用A表示），又要是未被修改过的页面（修改位用M表示）。把同时满足这两个条件的页面作为首选淘汰的页面，这种算法称为改进型Clock置换算法。
- 在淘汰一个页面时，淘汰修改过的页面比淘汰未被修改过的页面开销要大。因为如淘汰页面被修改过，必须将它重新写回磁盘；如淘汰的是未被修改过的页面，就不需要写盘操作。所以，采用将页表中的“访问位”和“修改位”结合起来使用可以改进时钟页面置换算法。

7.3.3 页面置换算法

“访问位”和“修改位”一共4种组合情况。

① 最近没有被访问，没有被修改（A=0, M=0）；

② 最近被访问，没有被修改（A=1, M=0）；

③ 最近没有被访问，但被修改（A=0，M=1）；

④ 最近被访问，也被修改过（A=1，M=1）。

- 第一步，选择最佳淘汰页面。遇到第一个A=0,M=0作为淘汰页面。
- 第二步，如果第一步失败，回到起点，开始查找A=0,M=1作为淘汰页面，扫描过程中需要把A置为0。
- 第三步，如果第二步失败，指针回到起点。再转向第一步。

7.3.4 页面调度性能

1. 抖动（或称颠簸）

- 刚被淘汰的页面又马上被调回内存，调回内存不久后又被淘汰出去，如此频繁进行，这种现象称为抖动（或称颠簸）。它使得系统中页面调度非常频繁，以致CPU大部分时间都花费在内存和外存之间的调入调出上。
- 引起抖动的原因通常与分配给进程的物理块数大小以及页面置换算法有关。解决抖动现象最根本最有效的方法是控制系统中并发进程数量，这通常需要基于程序局部性原理的驻留集和工作集模型实现。

7.3.4 页面调度性能

2. 驻留集

所谓驻留集是指进程在内存中的页面集合，驻留集尺寸是进程驻留在内存中的页面数量。系统为了建立驻留集应采用一定的页面调入策略。

（1）调页策略

（2）驻留集的管理策略

- 内存分配策略
- 页面置换策略

  全局置换，局部置换
- 页面置换策略和内存分配策略配合使用

  固定分配局部置换策略

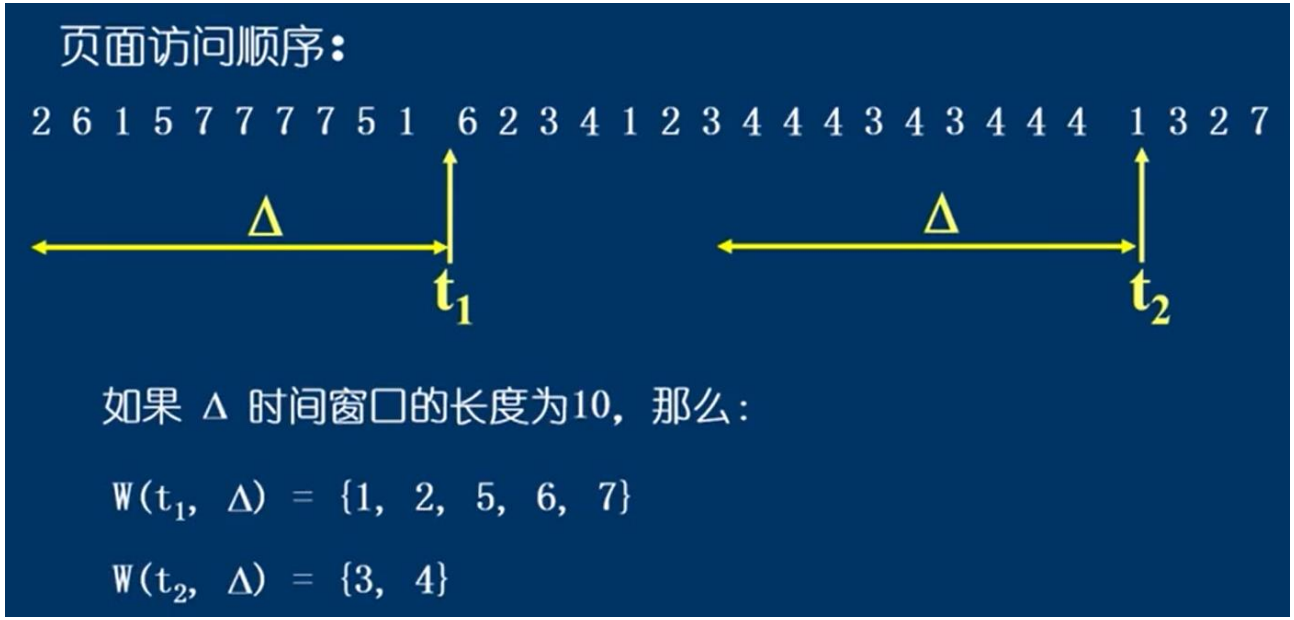
  可变分配全局置换

  可变分配局部置换

7.3.4 页面调度性能

3. 工作集

- 引起抖动的原因通常与分配给进程的物理块数大小以及页面置换算法有关。
- 为了解决抖动现象，基于局部性原理，提出了工作集概念。
- 工作集是指在某一时刻t，进程最近n次内被访问的页面集合，数字n称为工作集窗口，即工作集的大小。如果整个工作集都在内存中，在进入下一个运行阶段之前进程的运行不会引起很多页面故障。如果内存太小无法容纳整个工作集，进程运行将引起大量的页面故障并且速度十分缓慢。



7.3.5 影响缺页率因素

影响缺页率的主要因素有：

（1）分配给作业的物理块数

（2）页面置换算法

（3）页面大小的选择 理论上越大越好，一般0.5~4KB之间

（4）用户程序的编制方法 （局部性特征）

7.3.5 影响缺页率因素

```
Int A[100, 100];

//第一种
for(i = 0;i<100;i++)
    for(j=0;j<100;j++)
      A[i][j]

//第二种
for(j=0;j<100;j++)
    for(i=0;i<100;i++)
      A[i][j]
```

7.3.6 Belady现象

- 直观上，分配给进程的物理块越多，进程执行时发生的缺页次数就越少。但是在某些情况下，当分配的物理块多反而导致缺页次数更多，这种现象称为**Belady异常现象或FIFO置换算法的异常现象**。
- **FIFO算法是基于队列实现的，会出现Belady异常现象。而LRU算法是一种堆栈类算法，理论上可证明，堆栈类算法不可能出现Belady现象。**

7.3.6 Belady现象

- 例如考虑下面的页访问串：4、3、2、1、4、3、5、4、3、2、1、5。采用FIFO页面置换算法，分别分配3个和4个物理块时，各会出现多少次缺页中断？

表 7.6 分配 3 个物理块的进程运行情况图

时刻	1	2	3	4	5	6	7	8	9	10	11	12
引用串	4	3	2	1	4	3	5	4	3	2	1	5
物理块	4	4	4	1	1	1	5	5	5	5	5	5
		3	3	3	4	4	4	4	4	2	2	2
			2	2	2	3	3	3	3	3	1	1
缺页标记	+	+	+	+	+	+	+			+	+	

产生缺页中断的次数为 9 次。

7.3.6 Belady现象

- 例如考虑下面的页访问串：4、3、2、1、4、3、5、4、3、2、1、5。采用FIFO页面置换算法，分别分配3个和4个物理块时，各会出现多少次缺页中断？

分配 4 个物理块时，该进程运行时缺页情况如下表 7.7 所示：

表 7.7 分配 4 个物理块的进程运行情况图

时刻	1	2	3	4	5	6	7	8	9	10	11	12
引用串	4	3	2	1	4	3	5	4	3	2	1	5
物理块	4	4	4	4	4	4	5	5	5	5	1	1
		3	3	3	3	3	3	4	4	4	4	5
			2	2	2	2	2	2	3	3	3	3
				1	1	1	1	1	1	2	2	2
缺页标记	+	+	+	+			+	+	+	+	+	+

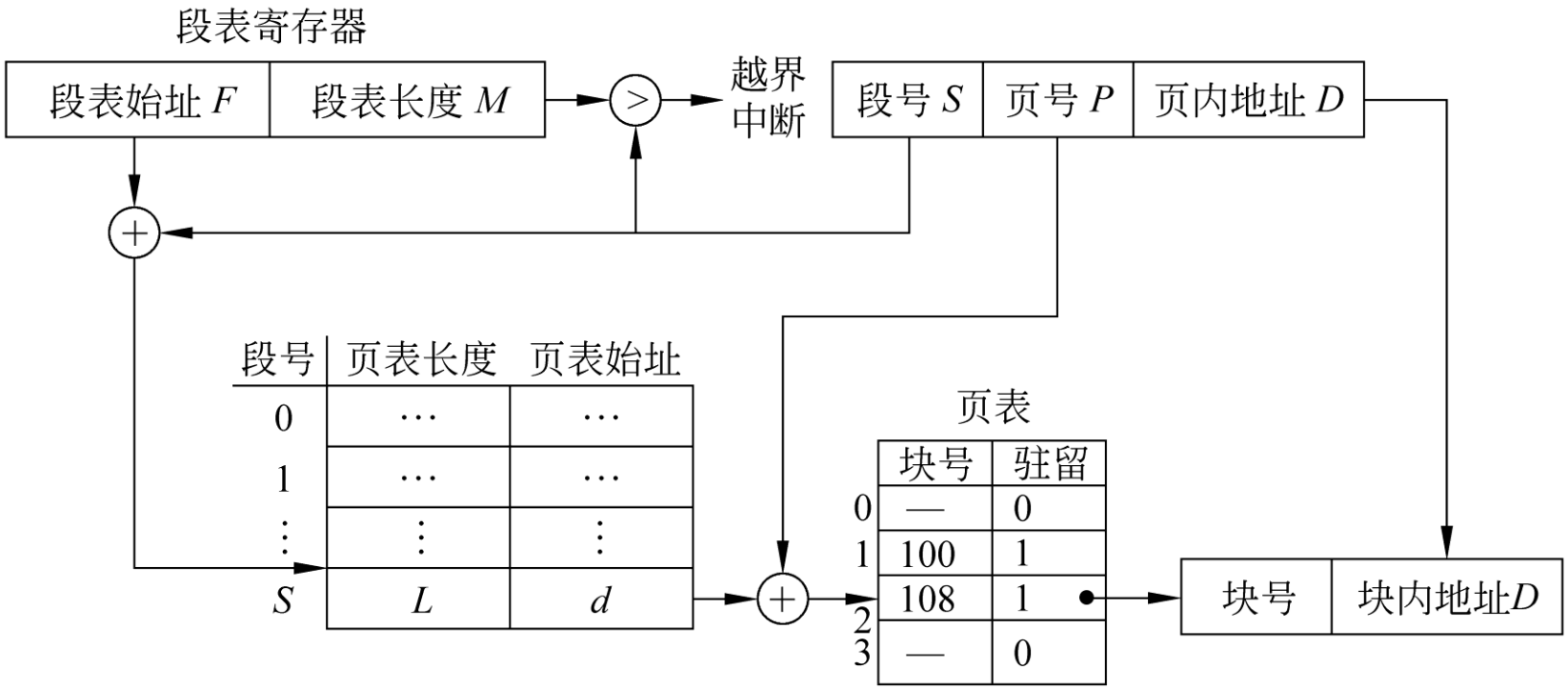
产生缺页中断的次数为 10 次。

7.3.7 请求分页存储管理的优缺点

- 请求页式存储管理的优点如下：
  ① 主存利用率比较高。
  ② 对磁盘管理比较容易。
  ③ 地址映射和变换的速度比较快。
- 请求页式存储管理的缺点如下：
  ① 程序的模块化性能不好。
  ② 页表过长。

7.5 请求段页存储管理方式

1. 地址变换和缺页中断



7.6 存储管理方案总结

表 6.8 几种存储方式总结

方式 功能	单一连续	分区		分页		分段		段页式	
		固定	可变	基本	请求	基本	请求	基本	请求
分配方式	连续	连续		离散		离散		离散	
虚拟空间	一维	一维		一维		二维		二维	
适用环境	单道	多道		多道		多道		多道	
共享	不能	不能		可以，但限制多		可以		可以	
内存扩充	交换	交换		交换	虚拟存储器	交换	虚拟存储器	交换	虚拟存储器

7.6 存储管理方案总结

内存分配单位	内存中整个用户分区	分区		页		段		页	
运行条件	一次性全部装入内存	一次性全部装入内存		全部装入内存	部分装入内存	全部装入内存	部分装入内存	全部装入内存	部分装入内存
地址重地位	静态	静态	动态	静态	动态	静态	动态	静态	动态
硬件支持	保护用寄存器	保护用寄存器，重定位机构		地址变换和保护机构	同左，增加中断机构	地址变换和保护机构	同左，增加中断机构	地址变换机构，保护机构	同左，增加中断机构