第五章 虚拟存储器

5.1 虚拟存储器概述
5.2 请求分页存储管理方式
5.3 页面置换算法
5.4 抖动 （略）
5.5 请求分段存储管理方式

5.1 虚拟存储器概述

- **前述的内存管理方法的共同缺点**
  - 要求程序必须一次性整体装入内存

- **导致两个问题**

① 程序太大，内存放不下

② 程序太多，但内存空间不足

- **解决方案**

① 从物理上增加内存容量，但成本高，且增加是有限的

② 从逻辑上增加内存容量。这正是虚拟存储技术所要解决的主要问题

1. 常规存储器管理方式的特征

① 一次性

- 程序运行时，要求一次性全部装入内存，有些暂时用不到的数据也必须装入，浪费内存

② 驻留性

- 不管是就绪、执行还是阻塞，都必须驻留内存，浪费内存

2. 局部性原理

- **一次性和驻留性在程序执行中是否是必须的呢？**

  不少学者就这一问题进行了广泛研究，其中最著名的是Denning提出的局部性原理，为虚拟存储器的实现奠定了基础

- **1968年，Denning指出：程序执行时将呈现出局部性规律**

  在一较短时间内，程序执行仅局限于某个部分，

  相应的，它所访问的存储空间也局限于某个区域

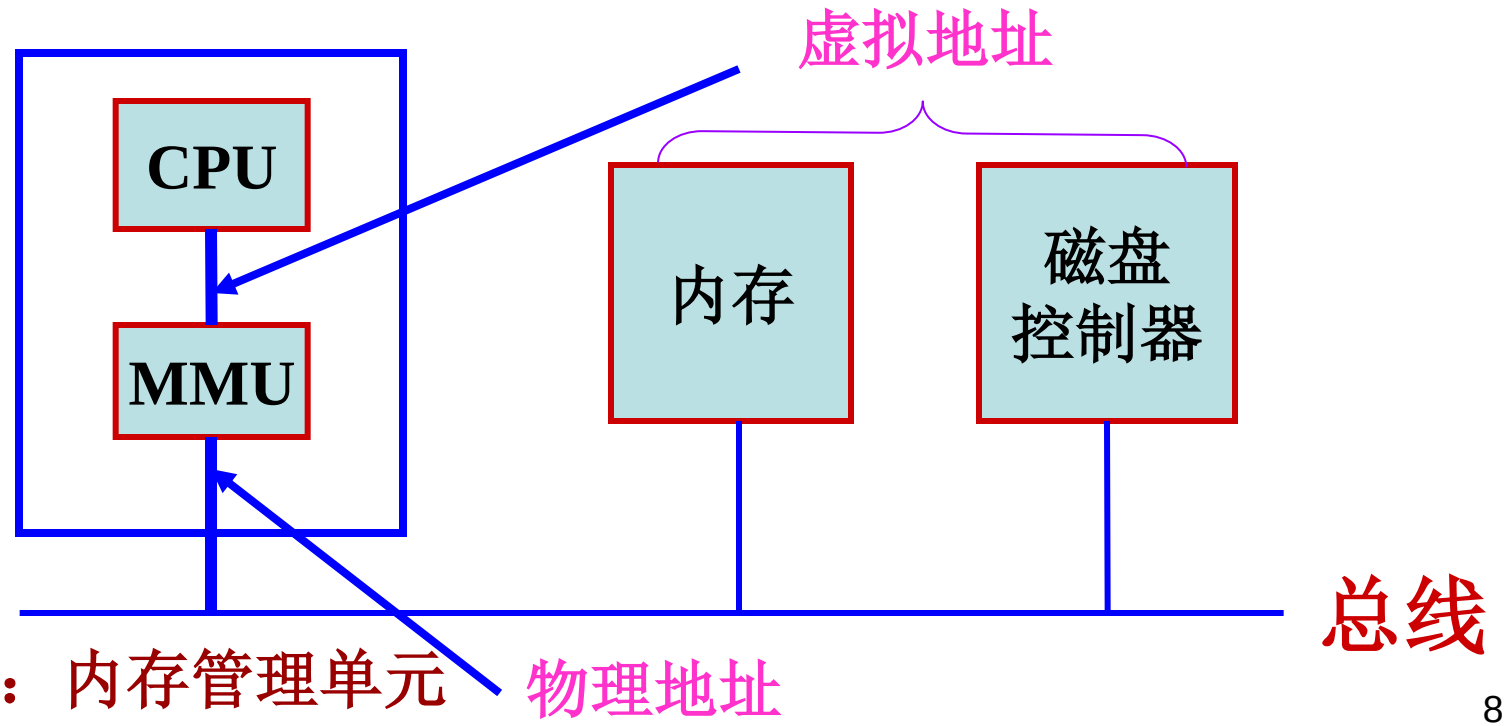
3. 虚拟存储器的基本工作原理

- 基于局部性原理，进程执行时，只需将当前要运行的部分(页面或段)装入内存，其余部分暂留在外存上
- 运行过程中，如果所访问的程序部分不在内存

① 则产生缺页(段)中断，OS利用请求调页(段)功能，将之调入内存，以维持进程的继续执行

② 如果内存已满，无法再装入新的页(段)，则利用置换功能，将内存中暂时不用的页(段)调至外存，腾出足够的内存后，再将之调入内存，以维持进程的继续执行

- **虚存**
  - 虚拟存储器技术允许程序、数据、堆栈超过内存大小，支持多道程序设计技术
  - 它把内存与外存有机的结合起来，从而得到一个容量超过实际内存的更大的“内存”，简称虚存

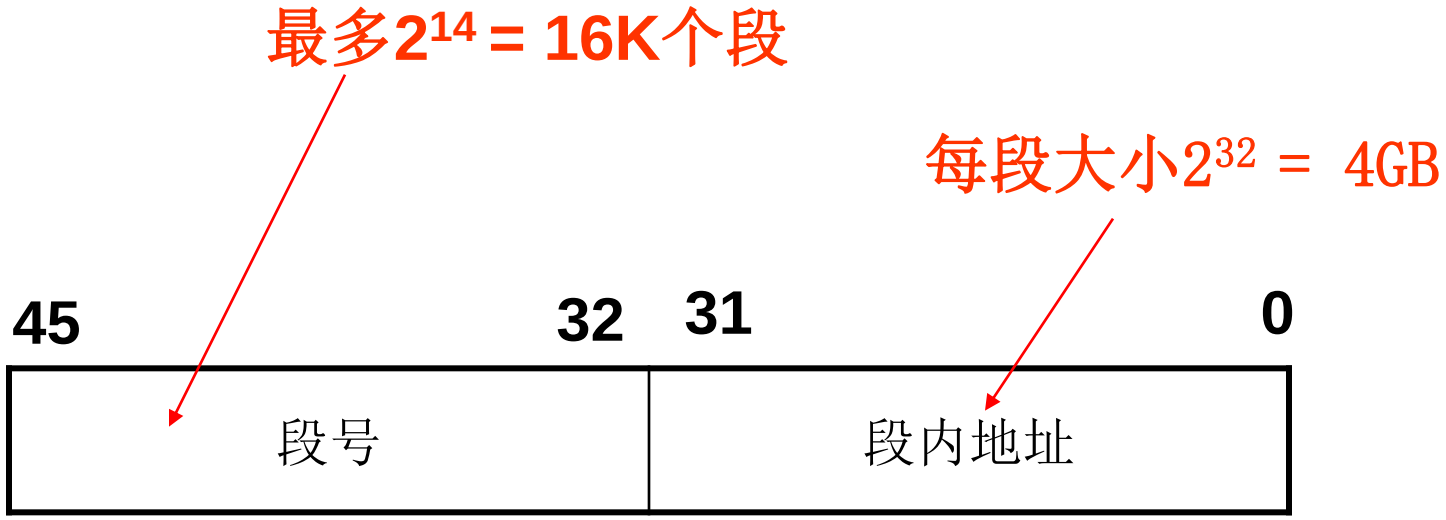


3. 虚拟存储器的定义

- 定义
  - 具有请求调入功能和置换功能，能从逻辑上扩充内存容量的一种存储器系统
- 容量
  - 逻辑容量 = 内存容量 + 外存容量
- 性价比
  - 运行速度接近内存速度，成本却接近于外存
- 总结
  - 虚拟存储技术是一种性能非常优越的存储器管理技术

- 一个虚拟存储器的最大容量是由计算机的地址结构确定的。
- 若给出的有效地址长度为20位，则程序可以寻址的范围为2^{20}=1M，即虚存的容量为：1M。
- 若这个地址长度为24位，则相应的虚存容量为2^{24}=16M。

- **虚拟地址空间**(程序中可用地址范围)**可以大于物理地址空间**(最大可管理的实际内存空间)
- 比如：80486是32位的处理器
  - 所以其地址线为32位，最大可管理4GB实际内存空间
  - 但其虚拟地址为46位，最大可管理64TB虚拟存储空间



- **虚拟存储器的特征**

① **多次性**

- 一个程序可以分多次调入内存
- 虚拟存储器最重要的特征

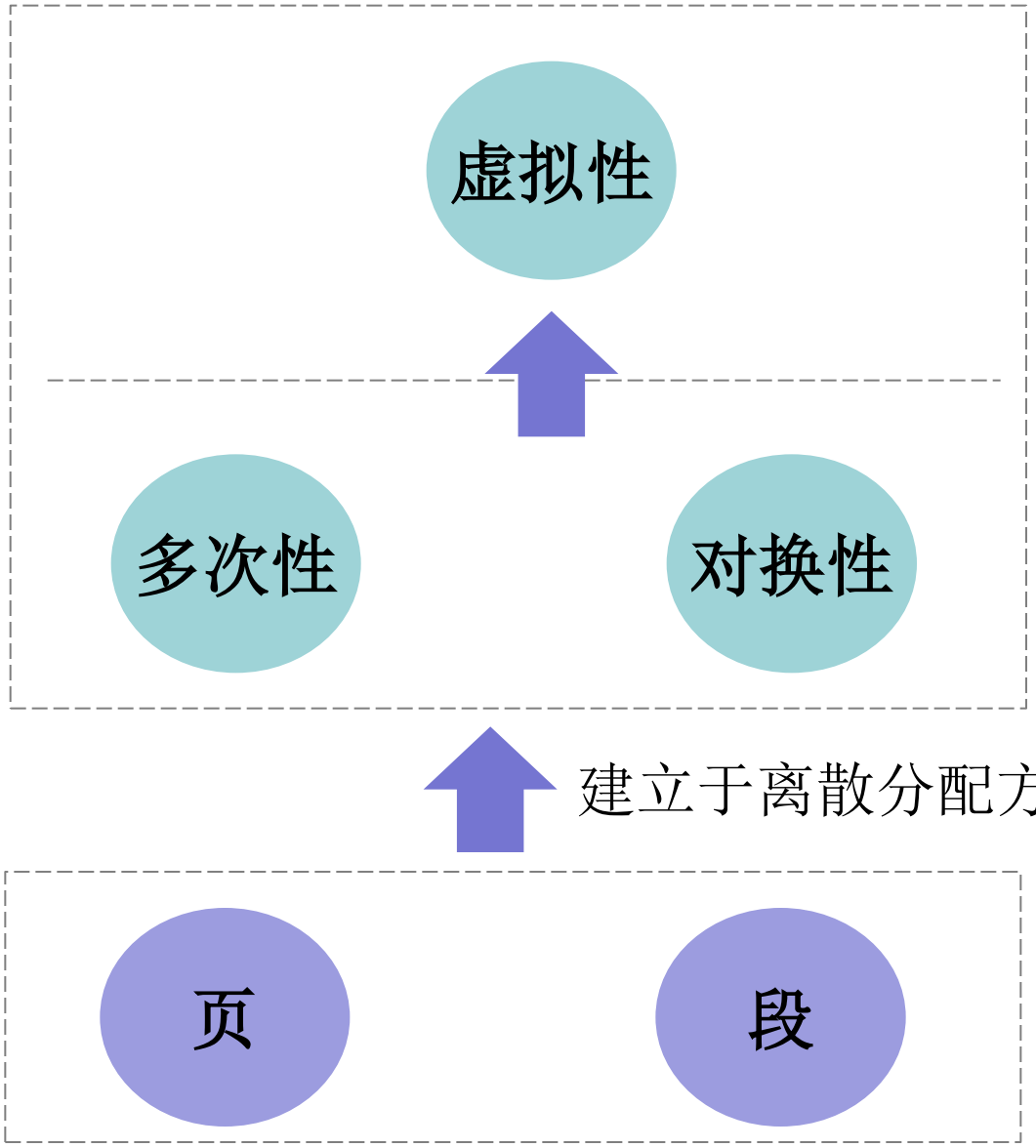
② **对换性**

- 允许程序在执行过程中换进、换出
- 有效提高内存利用率

③ **虚拟性**

- 能从逻辑上扩充内存容量，用户看到的容量远大于实际的内存容量
- 虚拟存储器另一个最重要的特征、实现虚拟存储器最重要的目标

虚拟存储器无一例外地都是建立在离散分配存储管理方式基础上



4. 虚拟存储器的实现方法

- 目前所有的虚拟存储器都是采用下述方式之一：

  ① 请求分页存储管理方式

  ② 请求分段存储管理方式

  ③ 段页式虚拟存储管理方式

- **请求分页存储管理方式**
  - 在分页系统的基础上增加了请求调页、页面置换两大功能所形成的页式虚拟存储系统
  - 为了实现这两大功能，除OS的软件支持外，还必须提供如下的硬件支持：

    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. 请求分段存储管理方式

- 在分段系统的基础上增加了请求调段、分段置换两大功能所形成的段式虚拟存储系统
- 为了实现这两大功能，除OS的软件支持外，还必须提供如下的硬件支持：

① 请求分段的段表机制

② 缺段中断机构

③ 地址变换机构

3. 段页式虚拟存储管理方式

- 目前，许多虚拟存储管理系统是建立在段页式系统的基础上的，通过增加了请求调页、页面置换两大功能所形成的段页式虚拟存储系统
- 如：Intel 80386处理机便支持段页式虚拟存储系统

5.2 请求分页存储管理方式

- **与基本分页系统的相同点：**
  - 把内存空间划分成大小相同、位置固定的块
  - 把程序的虚拟地址空间（就是以前的逻辑地址空间）划分成页
  - 由于页与块大小相同，因此虚拟地址空间中的一页，可以装入到内存中的任何一块中

- **与基本分页系统的不同点：**
  - 程序不必一次性装入内存，只装入目前要用的若干页，其他页仍保存在外存上
  - 运行时，虚拟地址被转换成 [页号, 位移量]
  - 根据页号查找页表时，如果该页已在内存中，必有具体块号与之对应，运行就能继续下去；
  - 如果该页不在内存，必然没有具体块号与之对应，出现“缺页”
  - 此时会产生缺页中断，OS根据页号把缺的页从外存调入内存，以保证进程的正常运行。

5.2.1 请求分页中的硬件支持

- 需要的硬件支持如下：

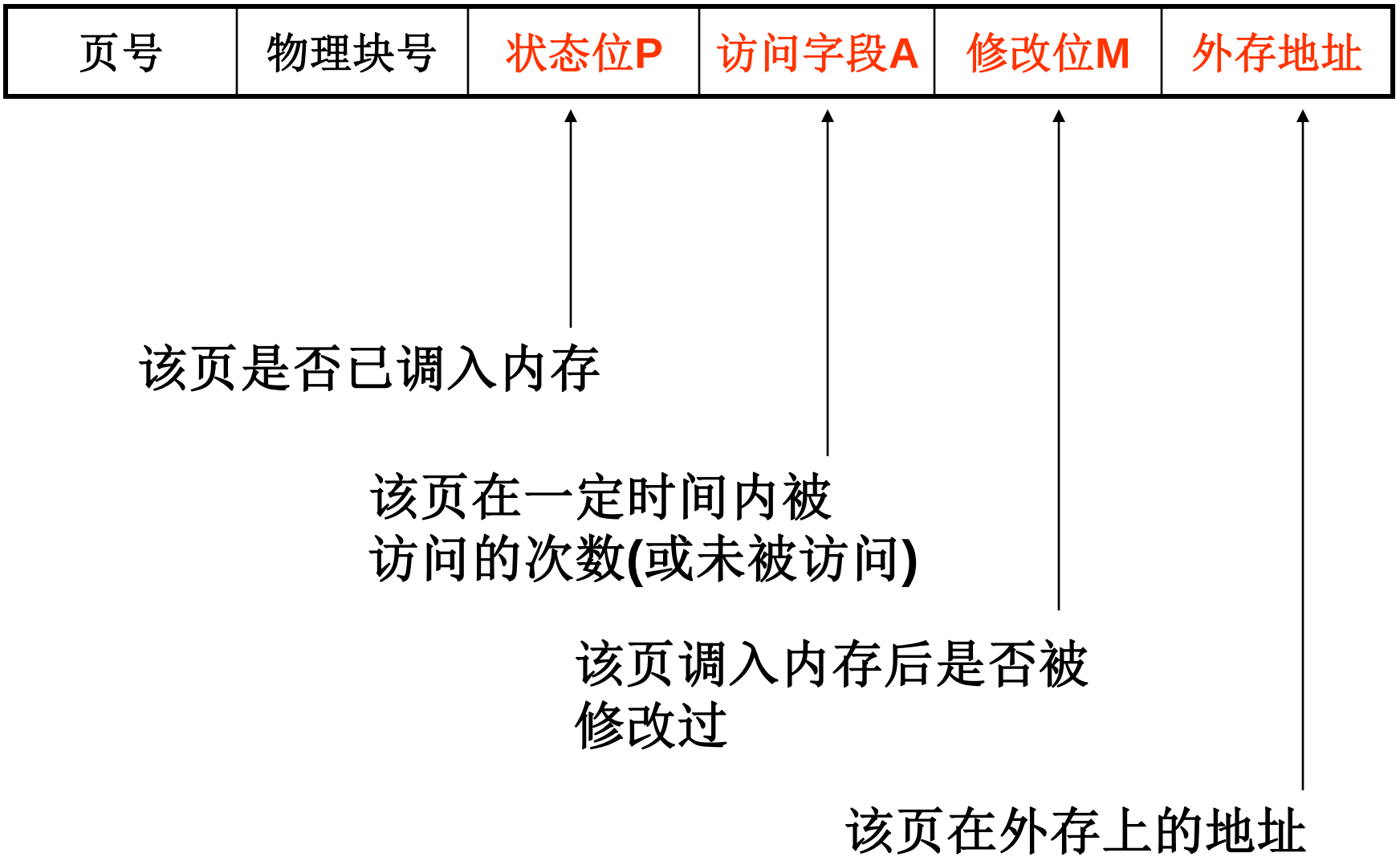
① 请求页表机制

② 缺页中断机构

③ 地址变换机构

1. 请求页表机制

- 与基本分页系统相比，页表的表项有了新的扩充，为实现上述“请求分页”，新增加四个字段


页号
物理块号
状态位P
访问字段A
修改位M
外存地址
该页是否已调入内存
该页在一定时间内被
访问的次数(或未被访问)
该页调入内存后是否被
修改过
该页在外存上的地址

2. 缺页中断机构

- 地址变换过程中，在页表中发现所要访问的页不在内存，则产生缺页中断
- OS接到该中断信号后，就调用缺页中断处理程序，根据页表中给出的外存地址，将该页调入内存，以保证进程继续运行。

- **缺页中断**是一种特殊的中断，与一般中断相比，主要表现为：

① 在指令执行期间产生和处理中断信号。通常，CPU只能在指令之间接受中断；然而，一个缺页中断要求在指令执行中间得到服务，即发现所要访问的指令或数据不在内存时产生缺页中断并立即处理

② 一条指令可能引起多次不同的页面中断

3. 地址变换机构

大家自己回去看看！

5.2.2 请求分页中的内存分配

- 为进程分配内存时，涉及到的三个问题：

① 确定最小物理块数

② 物理块的分配策略

③ 物理块的分配算法

① 最小物理块数的确定

- **最小物理块数：**保证进程正常运行的最小物理块数
  - OS为进程分配的物理块数少于此值时，进程将无法运行
- 与计算机的硬件结构有关，取决于指令的格式、功能和寻址方式
- 有些机器的可能跨两个页面，且源地址和目标地址所涉及的区域也都可能跨两个页面
- 所以，最小物理块数对于不同的机器是不同的

2. 物理块的分配策略

- 两种内存分配策略
  - 固定、可变分配策略
- 两种置换策略
  - 全局置换、局部置换

- **可组合出以下三种适用的策略**

① **固定分配局部置换：**为进程分配一定数目物理块，运行期间保持不变。(困难是：如何确定物理块数目)

② **可变分配全局置换：**为进程分配一定数目物理块。缺页时，从空闲物理块队列(或其他进程占用的物理块)里取出一块空闲的。(若换出时选任一进程的页，这样会使它的块减少，增加其缺页率)

③ **可变分配局部置换：**为进程分配一定数目物理块，发生缺页时，只从该进程的页面中选择一个换出

3. 物理块分配算法

- 下面只讨论“固定分配策略”
- 问题是：如何将系统中可供分配的物理块分配给各个进程？

① 平均分配算法

- 所有可供分配的物理块，平均分配给各个进程
- 缺点：貌似公平，实不公平
- 原因：没考虑各进程本身的大小，进程大的分配的页面可能明显不够，进程小的分配的页面可能太多，造成浪费

② 按比例分配算法

- 依据进程大小，按比例分配物理块

③ 考虑优先权的分配算法

- 对于重要、紧迫的进程，应为它们分配较多的内存空间
- **通常采取的方法**
  - 把内存中可供分配的所有物理块分成两部分：**一部分按比例分配给各进程；另一部分依据各进程优先权，适当分配给各进程**
- 在重要的实时系统中，可能完全按优先权分配物理块

5.2.3 调页策略

- 为使进程能运行，必须事先将要执行的那部分程序和数据调入内存。
- **问题是：**
  - 在何时调入所需页面
  - 从何处调入这些页面
  - 如何调入

1. 何时调入页面

- 如果出现缺页中断，表明企图访问一个不在内存的页面。
- 这时必须立即装入该页面，这种仅当需要时才提取页面的策略，称为请求调页策略；
- 如果对一个页面的访问是可以预期的，那么事先装入页面也是可能的，这样就可以减少缺页中断的产生，把事先提取页面的策略称为预调页策略

- 任何预先调页的策略都是以预测为基础的
- 如果能装入合适的不久要访问的页面，则可减少缺页中断
- 但是，如果预先装入的页面在很长一段时间内不被访问，那么它们会浪费所占空间，而且很可能又被交换到外存，这样的预调页是失败的
- 事实上，预先调页的成功率约为50%，采用这种预先调页策略不可能明显地降低缺页中断率
- 因此，大多数系统采用请求调页策略

2. 从何处调入页面

- 在请求分页系统中，把外存分为两部分

  ① **文件区**：用于存放文件

  ② **对换区**：用于存放对换页面

- 每当发生缺页请求时，系统应从何处将缺页调入内存，不同的OS采用的方法各不相同，可分成三种情况

① 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，进程运行前，须将有关文件从文件区拷贝到对换区

② 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
- 未被修改的页面换出时，不必再将它们导至外存，以后调入时仍从文件区调入
- 修改过的页面换出时调到对换区，需要时从对换区调入

③ UNIX方式

- 凡是未运行过的页面，都从文件区调入
- 曾经运行过、又被换出的页面，下次调入时，从对换区调入
- UNIX系统允许共享页面，某进程请求的页面有可能已由其它进程调入内存，此时无须再从对换区调入

5.3 页面置换算法

- 发生缺页时，要从外存把所需要页面调入到内存
- 如果当时内存中有空闲块，那么页面的调入问题就解决了
- 如果当时内存中已经没有空闲块可供分配使用，那么就必须在内存中选择一页调出内存，以便为即将调入的页面让出空间。这就是所谓的“页面淘汰”问题。

- 页面淘汰首先要研究的是**选择内存中的哪个页面作为淘汰对象**
- 虽然简单地随机选择一个页面淘汰，但如果它是一个经常使用的页面，那么由于很快又用到它，需要把它再一次调入，这种情况如果频繁发生，会增加系统的开销

- **抖动**
  - 系统一直忙于页面的换入/换出，以致大部分CPU时间都用于处理缺页中断和页面淘汰上，很少能顾及到用户进程的实际执行的现象称为“抖动”
- **注意**：**缺页中断不一定引起页面淘汰**。只有内存中没有可分配的空闲块时，缺页中断才会引起页面淘汰

5.3.1 最佳置换算法和先进先出置换算法

1. 最佳置换算法

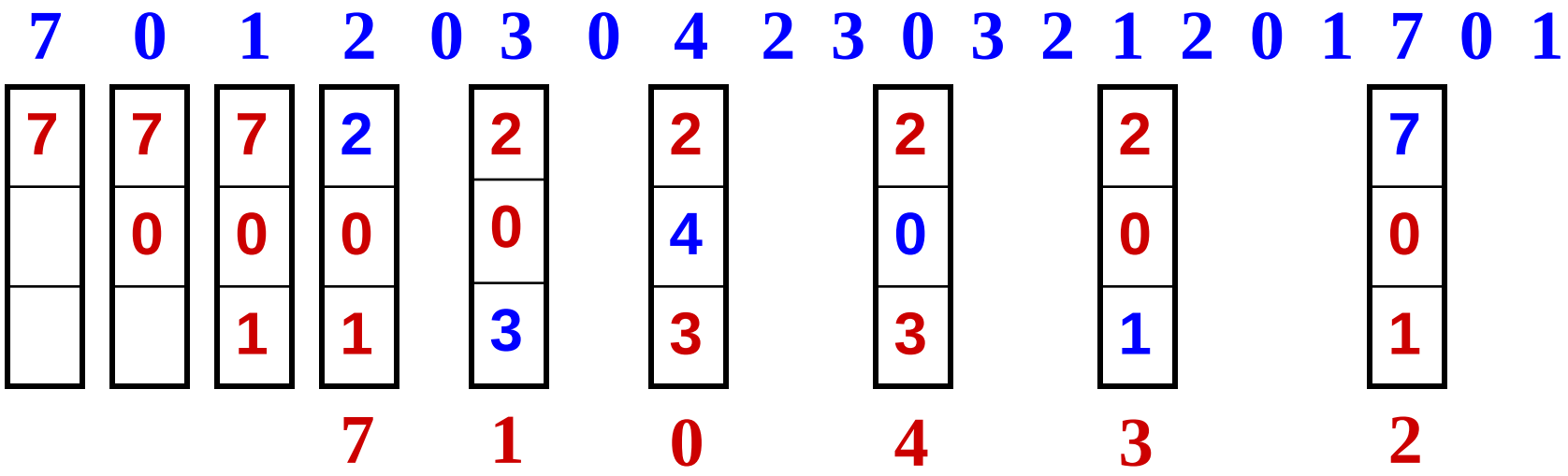
- 最理想的页面置换策略是
  - 从内存中移出永远不再需要的页面；如无这样的页面存在，则应选择最长时间不需要访问的页面
- 这便是最佳置换算法的思想，这种算法首先由Belady于1966年提出

- 最佳置换算法不是一种实际的方法，因为页面访问的未来顺序是不知道的
- 但是，可将其它的实用方法与之比较来评价这些方法的优劣。
- 所以，这种最佳策略具有理论上的意义

例：设系统为某进程分配了3个物理块，页面请求顺序为

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

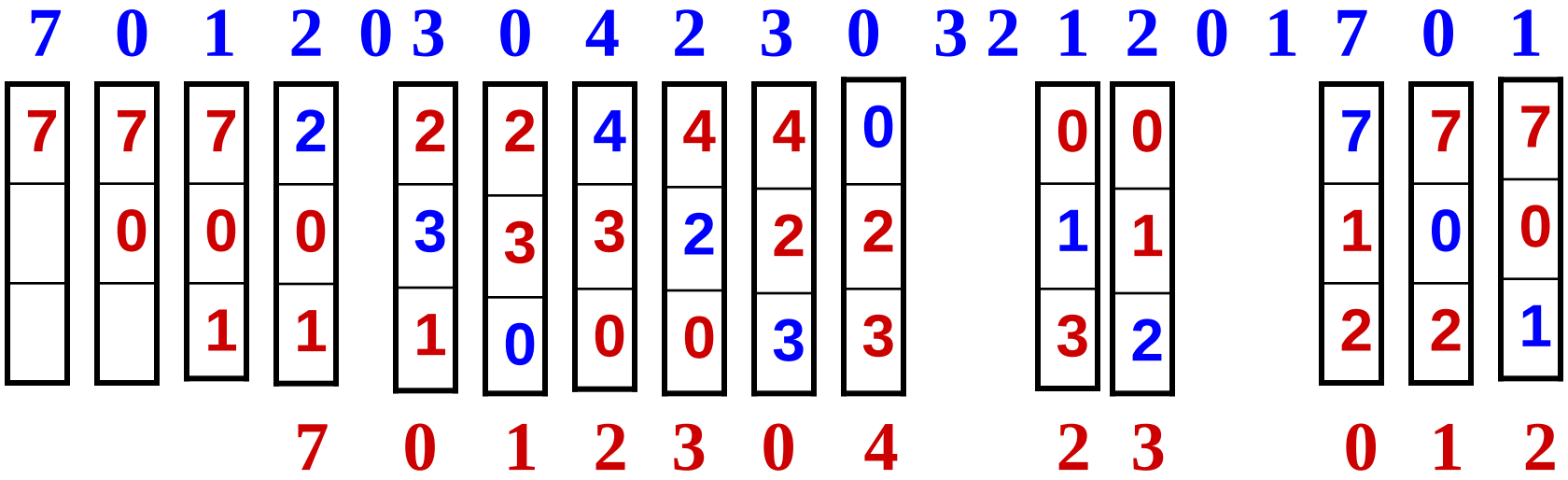
初始时物理块为空，采用最佳置换算法时的缺页情况如下：



- 从上面的演示可知，利用最佳置换算法，发生了6次页面置换。
- 问：发生了几次缺页中断？缺页率多少？
  - 发生了9次缺页中断
  - 缺页率 = 缺页次数/访问次数 = 9/20 = 0.45

2. 先进先出置换算法(FIFO)

- 基本思想
  - 总是选择进程中驻留时间最长的一页淘汰
  - 即：先进入内存的页面先淘汰出内存
  - 理由是，最早调入内存的页，不再被使用的可能性比最近调入内存的页要大


7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 4 4 4 0 0 0 7 7 7
0 0 0 1 3 3 2 2 2 1 1 1 0 0
1 1 0 0 0 3 3 3 2 2 3 2
7 0 1 2 3 0 4 2 3 0 1 2

- 从上面的演示可知，利用FIFO，发生了12次页面置换。
- 问：发生了几次缺页中断？缺页率多少？
  - 发生了15次缺页中断，
  - 缺页率=缺页次数/访问次数=15/20=0.75

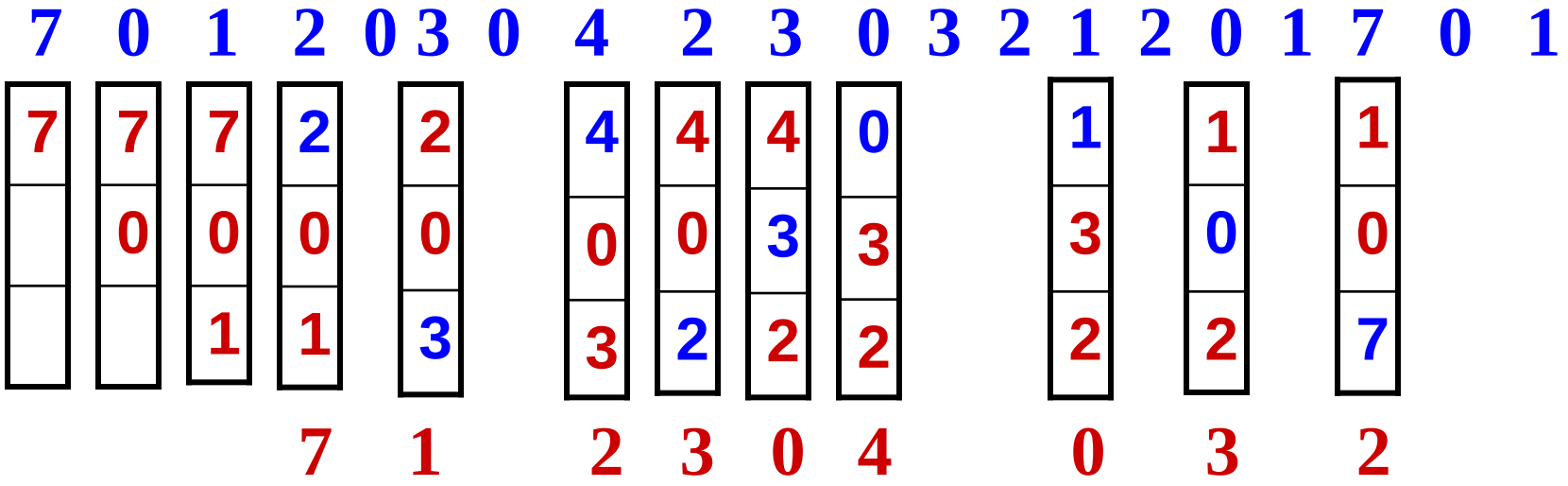
- **缺点：**
  - 可能会造成一些经常被访问的页面被换出，性能较差
  - 存在“Belady现象”：分配的页面数增多但缺页率反而提高的异常现象

5.3.2 最近最久未使用(LRU)置换算法

1. LRU(least Recently Used)算法描述

- 基本思想
  - 利用局部性原理，根据一个进程在执行过程中**过去**的页面访问踪迹来**推测未来**的行为
  - 因为，过去一段时间里不曾被访问的页面，在最近的将来可能也不会被访问

- **实质**：选择最近一段时间内最久不用的页面进行淘汰
- 实现这种技术，是给每个页面一个访问字段来记录一个页面自上次被访问以来所经历的时间 t，并淘汰t 最大的页。
- 它比较普遍适用于各种类型的程序



- 从上面的演示可知，利用LRU，发生了9次页面置换。
- 问：发生了几次缺页中断？缺页率多少？
  - 发生了12次缺页中断，
  - 缺页率=缺页次数/访问次数=12/20=0.60

2. LRU置换算法的硬件支持

① 寄存器

- 为了记录进程在内存中各页的使用情况，须为每个内存中的页面配置一个移位寄存器，可表示为：

$$R=R_{n-1}R_{n-2}R_{n-3}\cdots R_2R_1R_0$$

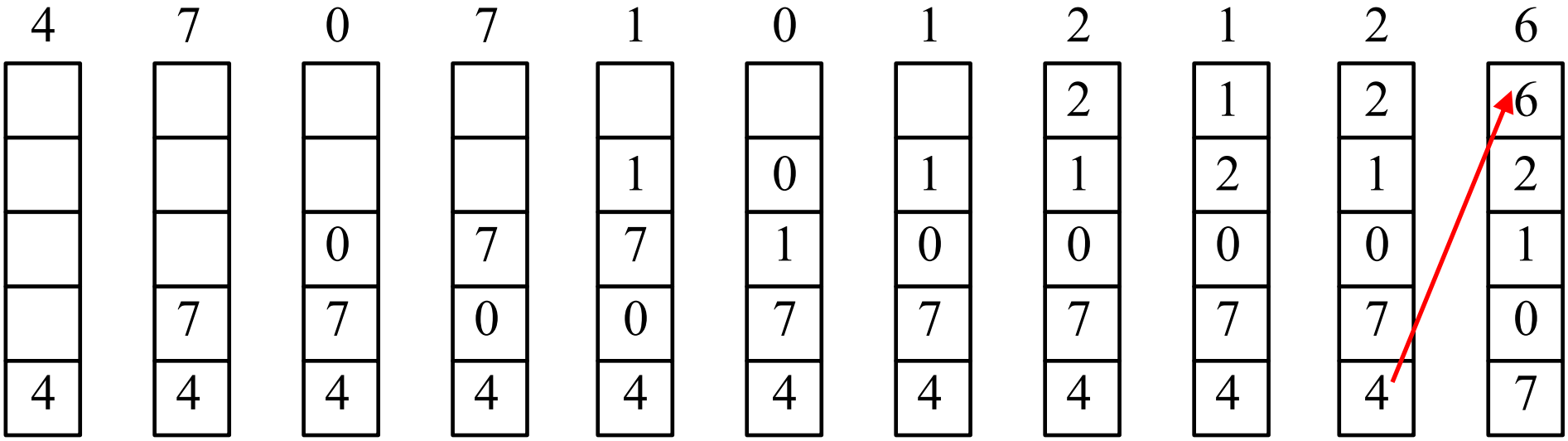
- 进程访问某物理块时，将相应寄存器的R_{n-1}置1
- 移位寄存器定时右移一位
- **具有最小数值的寄存器对应的页面，就是LRU的页面**

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

某进程具有8个页面时的LRU访问情况

② 栈

- 访问某页面时，从栈内取出，再压入栈顶。**LRU是栈底的页面号**



用栈保存当前使用页面时栈的变化情况

- Clock置换算法
- 页面缓冲算法
- 访问内存的有效时间
- “抖动”与工作集

自己回去看！

5.5 请求分段存储管理方式

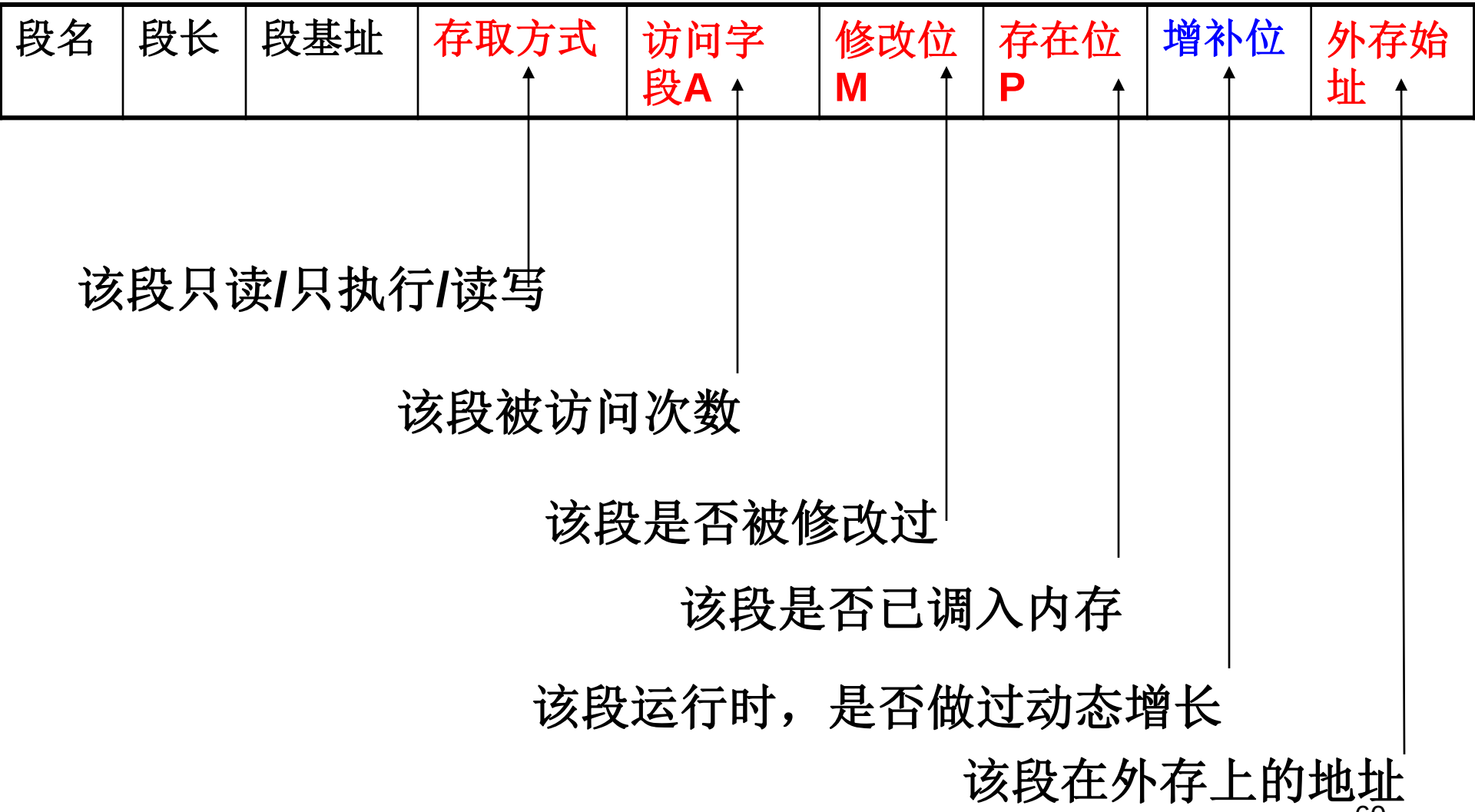
- **基本思想**
  - 与请求分页存储管理类似，只不过换入/换出是以“段”为单位
- **硬件支持**

① 请求段表机制

② 缺段中断机构

③ 地址变换机构

① 请求段表机制


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
该段只读/只执行/读写
该段被访问次数
该段是否被修改过
该段是否已调入内存
该段运行时，是否做过动态增长
该段在外存上的地址

2. 缺段中断机构

- 需要在一条指令执行期间处理缺段中断
- 一条指令执行期间，可能产生多次缺段中断
- 但不会出现一条指令被分割进两个段，或一组逻辑信息位于两个段的情况

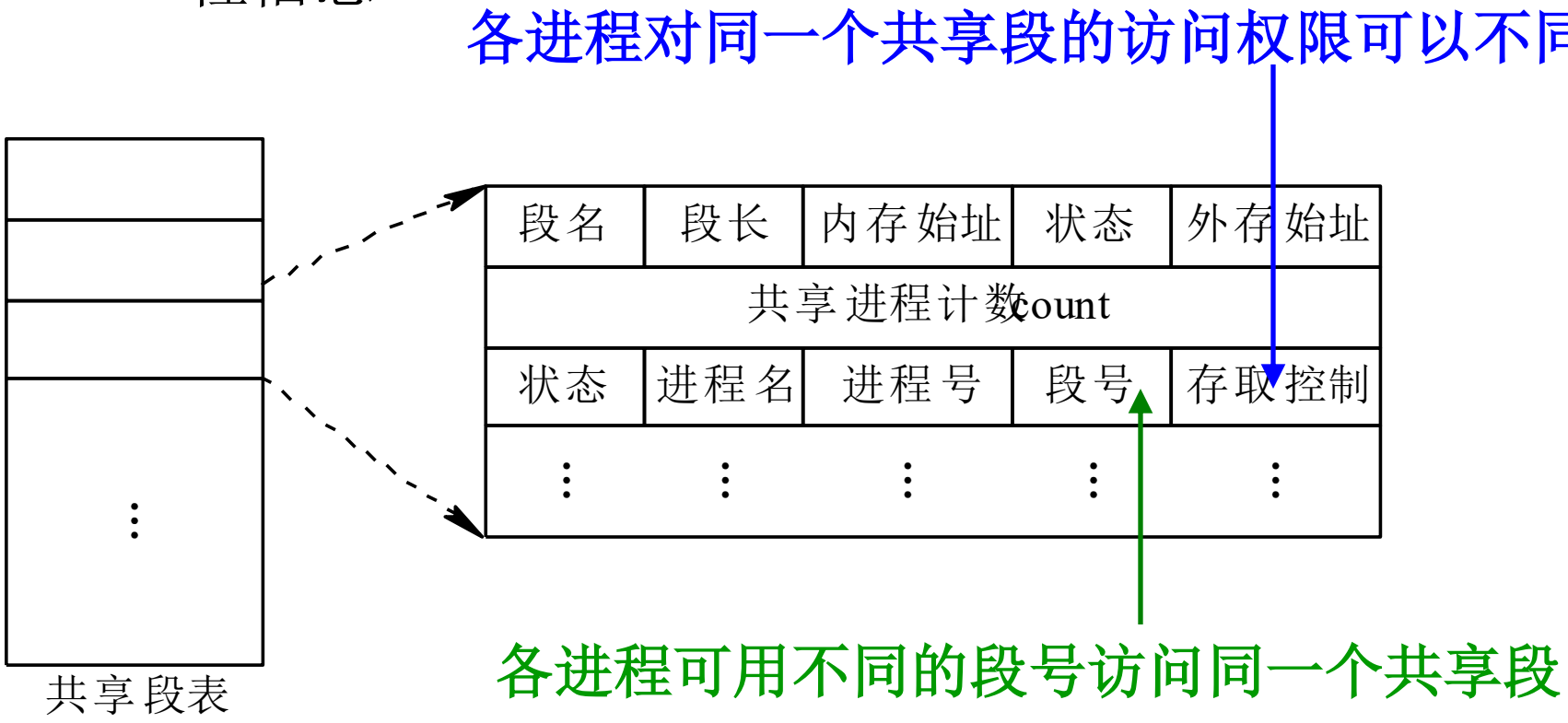
3. 地址变换机构

- 类似分段系统，只是增加了缺段中断处理
- 自己回去看！

5.5.2 分段的共享与保护

1. 共享段表

– 每个表项保存可共享段的信息，及共享此段的进程信息



2. 共享段的分配与回收

① 共享段的分配

- 第一个请求使用该段的进程，由OS为该段分配内存并调入，同时将内存始址填入请求进程的段表的相应项中
- 在共享段表中增加一表项，填写有关数据，令count=1
- 又有其它进程调用该段时，无须再为它分配内存。只需在调用进程的段表中增加一表项，填写该段的物理地址
- 在共享段的段表中，填上调用进程的信息，令count = count+1，以表明有两个进程共享该段

② 共享段的回收

- 当该段的某进程不再需要它时，应将该段释放，包括撤销在该进程段表中的相应表项，令count = count - 1
- 若 count = 0，由OS回收该段的内存，并撤销共享段表中该段的表项，表明此时已没有进程使用该段；
- 否则(即 count > 0)，只是取消调用进程在共享段表中的有关记录

3. 分段保护

① 越界检查

- 根据段表长度和段长，判断访问的地址是否越界

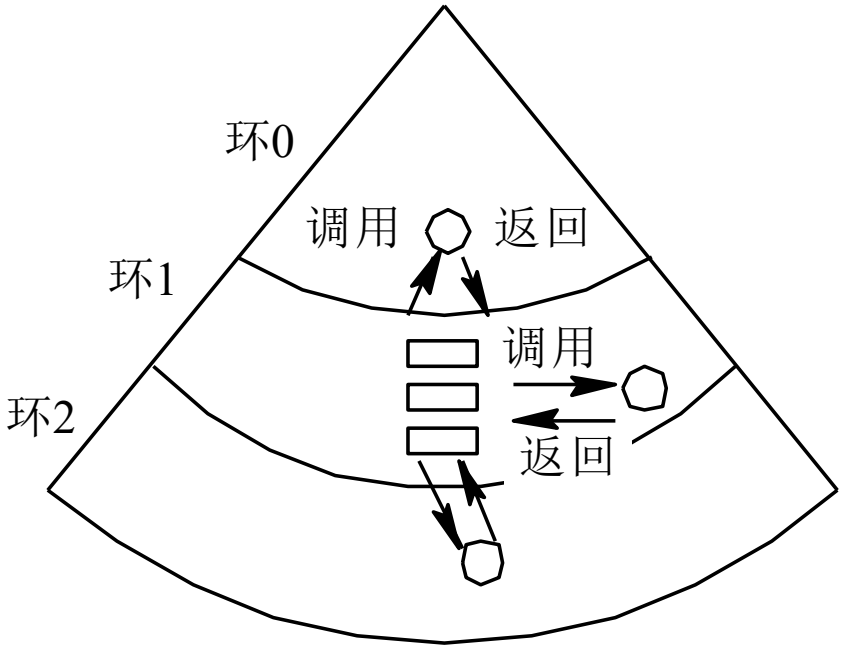
② 存取控制检查

- 只读/只执行/读写

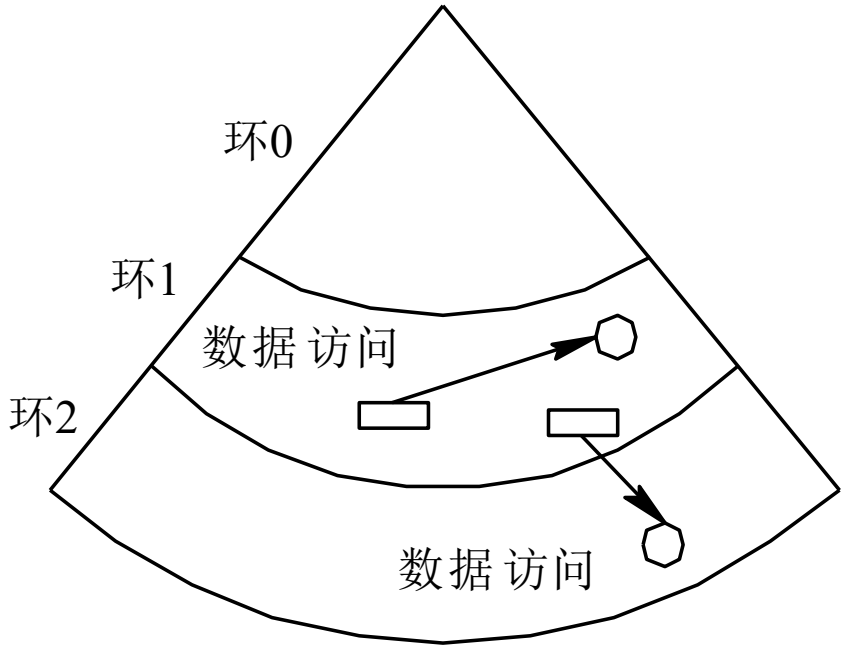
③ 环保护机构

- 一种功能较完善的保护机制
- 低编号的环具有高优先权
- OS内核位于0环内，重要的程序位于中间环，一般程序位于外环

- 对于一个进程：
  - 可以访问相同环、较低特权环中的数据
  - 可以调用相同环、较高特权环中的服务



(*a*) 程序间的控制传输



(*b*) 数据访问

环保护机构

小结

- 虚拟存储的基本原理
- 请求分页、请求分段的工作原理
- 置换算法(最佳、先进先出、LRU)

练习题

1. 虚拟存储器的最大容量(　B　)

A. 为内外存容量之和

B. 由计算机的地址结构决定

C. 任意的

D. 由程序的地址空间决定

2. 在虚拟存储系统中，若进程在内存中占3块(开始时为空)，采用FIFO算法，当执行访问页号序列为1、2、3、4、1、2、5、1、2、3、4、5、6时，将产生(D)次缺页中断

A．7　B．8　C．9　D．10

3. 系统“抖动”现象的发生是由(A)引起的

A. 置换算法选择不当

B. 交换的信息量过大

C. 内存容量不足

D. 请求页式管理方案

4. 实现虚拟存储器的目的是(　D　)

A. 实现存储保护　　B. 实现程序浮动

C. 扩充辅存容量　　D. 扩充内存容量

5. 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(　B　)指令

A. 被中断的前一条　　B. 被中断的

C. 被中断的后一条　　D. 启动时的第一条

6. 在请求分页存储管理中，若采用FIFO页面淘汰算法，当分配的页面数增加时，缺页中断的次数(　D　)

A．减少　　B．增加

C. 无影响　　D．可能增加也可能减少

7. 虚拟存储管理系统的基础是程序的(　A　)理论

A. 局部性　　B．全局性

C. 动态性　　D．虚拟性

8. 下述(A)页面淘汰算法会产生Belady现象

A. 先进先出　　B. 最近最少使用

C. 最不经常使用　　D. 最佳

9. 在请求分页系统中，主要的硬件支持有请求分页的页表机制、缺页中断机构和(C)

A. 时间支持　　B. 空间支持

C. 地址变换机构　　D. 虚拟存储

10. 假设某程序的页面访问序列为1、2、3、4、5、2、3、1、2、3、4、5、1、2、3、4且开始执行时内存中没有页面

- 则在分配给该程序的物理块数是3且采用FIFO方式时缺页次数是(13)
- 在分配给程序的物理块数是4且采用FIFO方式时，缺页次数是(14)
- 在分配给该程序的物理块数是3且采用LRU方式时，缺页次数是(14)
- 在分配给该程序的物理块数为4且采用LRU方式时，缺页次数是(12)

11. 设某进程的页面访问串为1,3,1,2,4，分配的物理块是3块，采用FIFO置换算法时，访问页面4时，要淘汰(　1　)号页面

12. 设某进程的页面访问串为1,3,1,2,4，分配的物理块是3块，采用LRU置换算法时，访问页面4时，要淘汰(　3　)号页面

13. 在下述存储管理技术中，(　D　)处理不当会产生抖动。

A.固定分区　B.可变分区 C.简单分页 D.请求分页

14.使用下面哪个存储管理方法不可以实现虚拟存储？（ A ）

A. 动态分区分配　　**B.** 分页存储管理

C. 分段存储管理　　**D.** 段页式存储管理

15.下面哪个概念在请求分页存储管理系统中不一定涉及？（ C ）

A. 页面置换　**B.** 缺页中断　**C.** 快表　**D.** 页表

16. 关于请求分页存储管理，正确的是(B)

A.采用静态重定位　B．采用动态重定位

C.内存静态分配　D．内存固定分配

17.假设一个机器有32位的地址结构，页面大小是4KB。如果只有一级页表，那么页表里有(2^{20}) 个表项。

18. 在请求分页系统中，若逻辑地址中的页号超过页表寄存器中的页表长度，则会引起(C)中断。

A.输入输出　B.时钟　C.越界　D.缺页

19.在请求分页系统中，页表中的外存地址的作用是 (B)

A. 供分配页面时参考

B. 供置换算法参考

C. 供程序访问时参考

D. 供进程终止时参考

20.下面关于请求分页存储管理说法中，不正确的是（ D ）

A.程序空间页的大小与计算机内存物理块的大小总是一致的

B.地址变换机构必须由相应的硬件支持

C.将用户地址空间分为页号和页内地址用户是感觉不到的

D.用户程序必须全部装入

21. Belady现象是指 (B) 。

A. 淘汰页很可能是一个马上要用的页

B. 当分配到的物理块数增加时，缺页中断的次数有可能反而增加

C. 缺页次数与系统的页面大小有关

D. 引起系统抖动的现象