# 第16章　Windows 的存储器管理

# 本章主要讨论内容

1. 进程地址空间的布局
2. 进程空间的主存分配，所涉及的数据结构
3. 地址转换
4. 主存空间管理（页框数据库）
5. 页调度策略

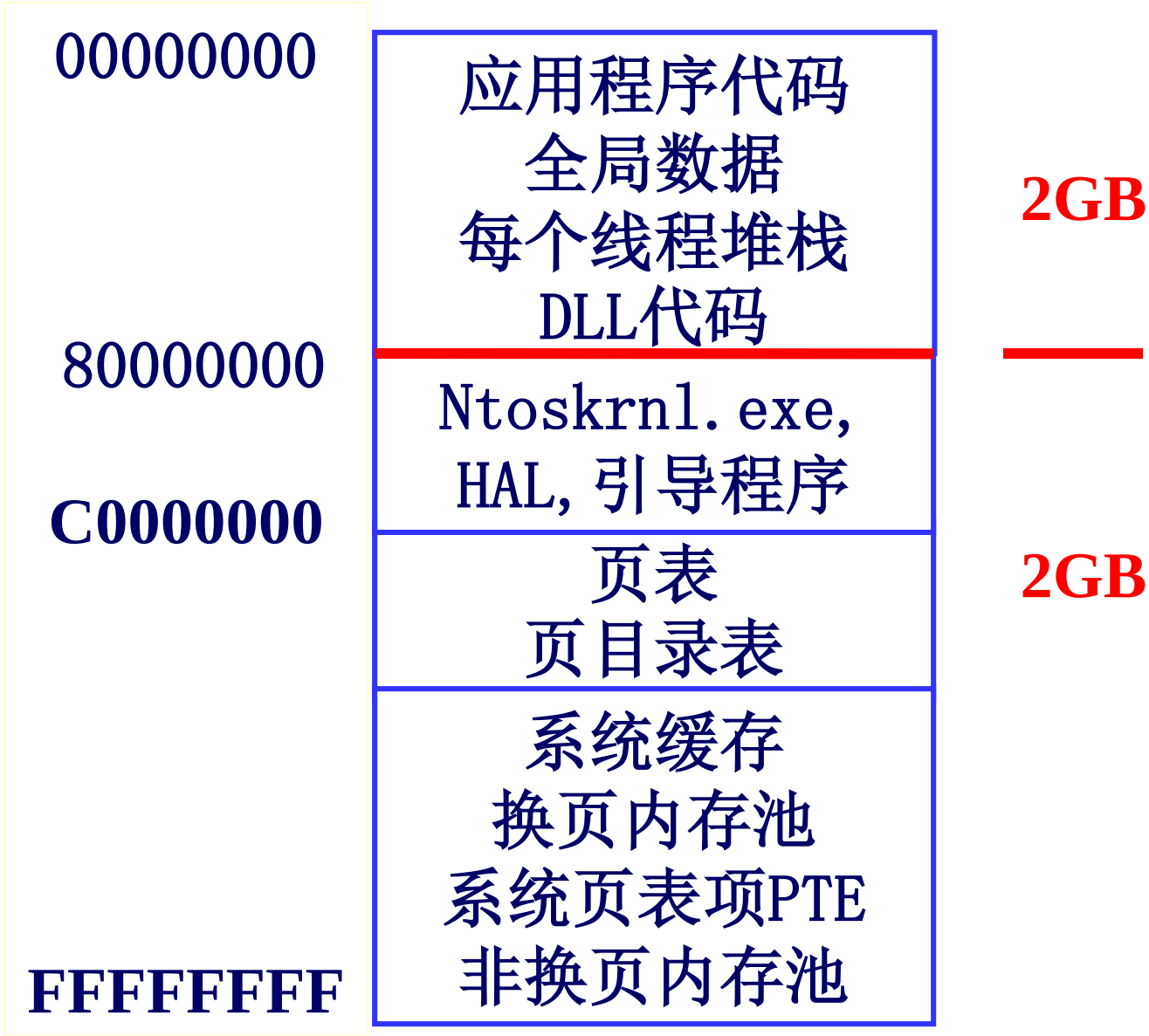
# 16.1 存储器管理基本概念

- 1．提供存储管理所需的系统服务。包括分配、释放和保护虚存和物理主存，写时拷贝，获得有关虚拟页的信息，主存共享和文件映射等。
- 2．提供运行在核心态系统线程上的几个例程：
- ①平衡工作集管理器。每秒运行一次，维护空闲主存数量不低于某一界限并及时调整进程的工作集。
- ②进程/堆栈交换程序。完成进程和线程核心堆栈的换入和换出操作。
- ③更改页写入器。定期将"脏"页写回到磁盘。
- ④废弃段线程。负责系统高速缓存和页文件的扩大和缩小。
- ⑤零页线程。负责维护系统有足够多的零填充的空闲页存在。

# 16.1.1　进程地址空间的布局

- 在32位的地址空间上，允许每个用户进程占有4G的虚存空间。低2GB为进程的私有地址空间，高2GB为进程公用的操作系统空间。
- Windows 的企业版有一个引导选项，允许用户拥有3GB的地址空间。

# x86系统虚拟地址空间布局



# 页文件（交换区）

- 页文件：是指作为主存补充的磁盘的那部分空间。如果计算机有128MB物理主存，同时在磁盘上有256MB的页文件，那么就认为计算机拥有384MB虚拟主存。
- Windows 2000支持最多16个页文件。当系统启动时，打开页文件。一旦打开页文件，在系统运行期间不能删除，因为系统进程为每个页文件都维持一个打开的句柄。

# 16.1.2　进程空间的主存分配

Windows 管理进程私有地址空间采用两种描述方式：

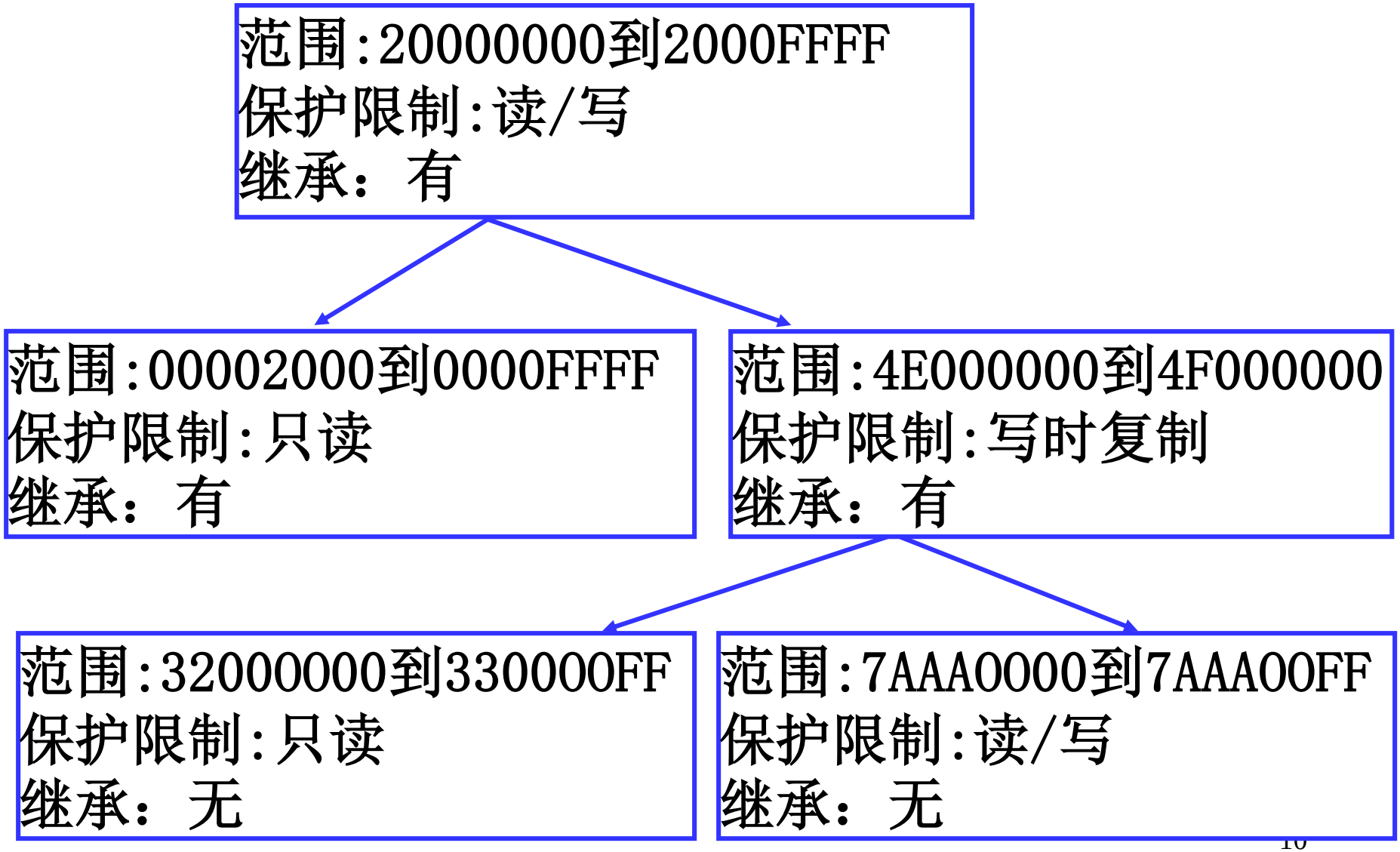
1. 虚拟地址描述符（VAD，Virtual Address Descriptor）
2. 区域对象（Section Object）

# 1．虚拟地址描述符VAD

- 存储器管理器采用请求页式调度算法。
- 进程页表的构建一直推迟到访问页时才建立。（“懒惰”方式）
- 当一个线程要求分配一块连续虚存时，存储器管理器并不立即为其构造页表，而是为它建立一个VAD结构，记录该地址空间的相关信息。进程的页表依据VAD来建立。

- **VAD结构**包括被分配的地址域、该域是共享的还是私有的、该域的存取保护以及是否可继承等信息。
- 存储管理器通过维护一组**VAD**结构，记录每个进程地址空间的状态。一个进程的一组**VAD**结构构成一棵自平衡二叉树，以便快速查找。

# 虚拟地址描述符树


范围:20000000到2000FFFF
保护限制:读/写
继承：有
范围:00002000到0000FFFF
保护限制:只读
继承：有
范围:4E000000到4F000000
保护限制:写时复制
继承：有
范围:32000000到330000FF
保护限制:只读
继承：无
范围:7AAA0000到7AAA00FF
保护限制:读/写
继承：无

# 2. 区域对象

- 在 Win32 子系统中，“区域对象”(section object)被称为“文件映射对象”，是一个可被多个进程共享的存储区。
- 一个区域对象可被一个或多个进程打开。
- 区域对象可被映射到页文件或磁盘上的其他文件。

# 区域对象的主要作用

I. 利用区域对象可将一个可执行的映像装入主存。然后访问这个文件就象访问主存中的一个大数组，而不是对文件进行读/写操作。

II. 使用区域对象可将一个大于进程地址空间的文件映射到进程地址空间。

III. 高速缓存管理器利用区域对象访问一个被缓冲文件中的数据。

# 区域对象的结构

对象类型

对象体属性

服务程序

区域对象

1）最大尺寸
2）页保护方式
3）页文件/映像文件
4）基准的/非基准的

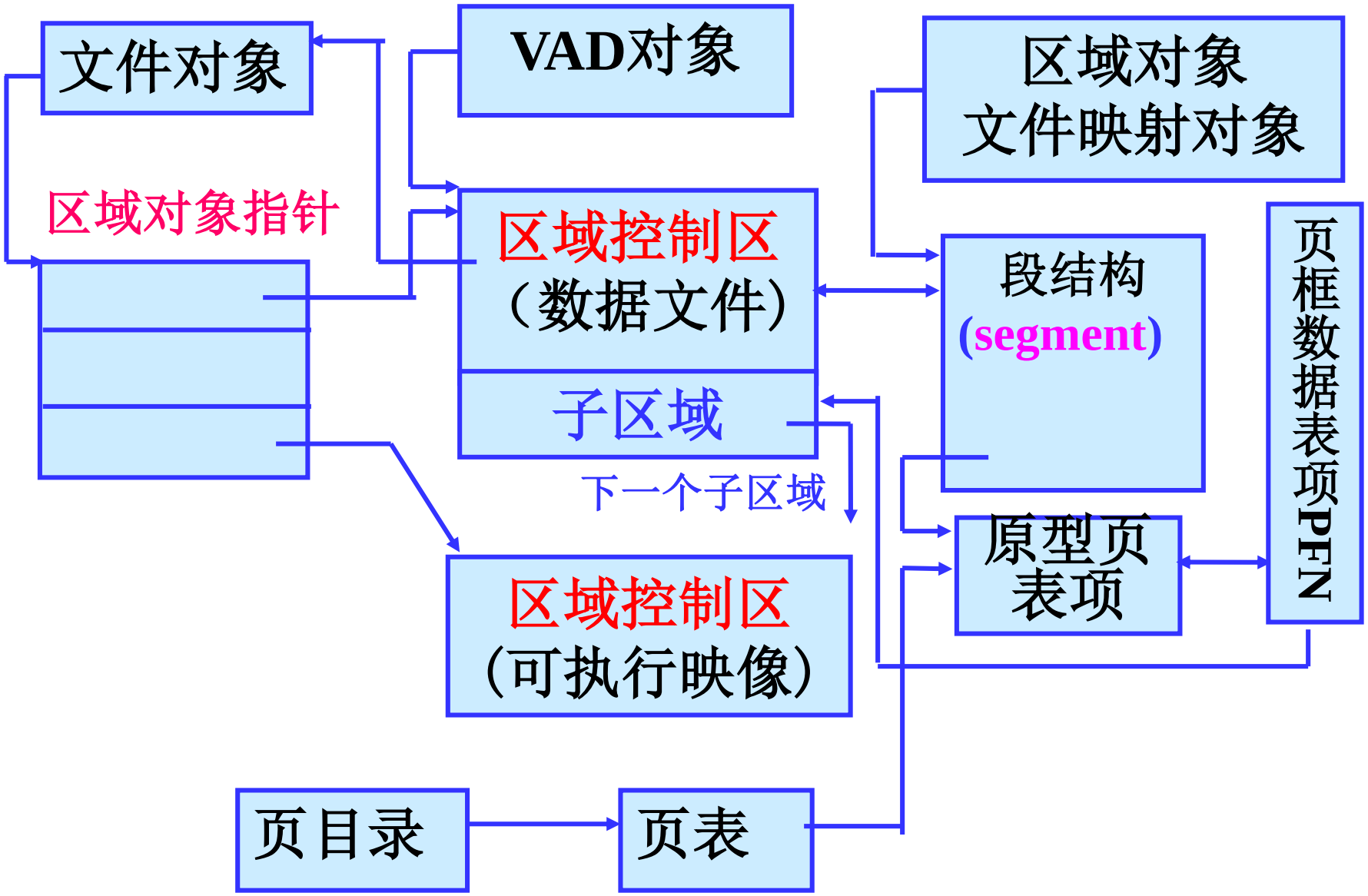
创建区域
打开区域
扩展区域
映射/废除一个映射视口
查询区域

1) 最大尺寸：区域可增长到最大字节数；如果映射一个文件，就是文件的大小。
2) 页保护方式：分配给该区域的所有页框的保护方式。
3) 页文件/映像文件：指出区域是否被创建为空（基于页文件），或是加载一个文件（基于映像文件）。
4) 基准的/非基准的：基准的要求共享区域出现在不同进程的相同虚地址处，非基准的要求共享区域可以出现在不同进程的不同虚地址处。

# 区域对象与映射文件之间的关系

**A.** 对于每个打开的文件(文件对象)，都有一个单独的区域对象指针结构。该结构为所有类型的文件访问维护数据一致性。

**B.** 区域对象指针结构由三个指针组成。其中两个控制区域分别用来映射数据文件和可执行文件。




系统内部区域各数据结构之间的关系

C. 控制区域依次指向“子区域”结构，控制区域也指向一个在换页内存池中分配的“段”结构。该段结构依次指向用于映射到区域对象的实际页面的原型页表项。进程页表指向这些原型页表项。

D. 换页内存池位于系统空间。

# Win32子系统实现文件映射的过程

- 一个进程要访问一个非常大的区域对象，只需在自己的地址空间保留一部分空间，来映射该区域对象的一部分。被进程映射的那部分叫做该区域的一个视口。

# 实现文件映射的过程

① 用**CreateFile()**创建/打开一个被映射文件
② 用**CreateFileMapping()**创建一个与被映射文件大小相等的区域对象。
③ 用**MapViewOfFile()**将区域对象的一个视口映射到进程保留的某部分地址空间，之后进程就可以像访问主存一样访问文件。
④ 访问完成，用**UnMapWiewOfFile()**解除被映射的视口。

# 3．虚存的分配

- 进程私有的2G地址空间的页可能是空闲的，或被保留，或被提交。
  - 被保留：已预留虚存，还没分配物理主存
  - 被提交：已分配物理主存或交换区。
- 分配主存时，可以先保留地址空间，后提交物理主存；也允许保留和提交同时实现。

# 分配主存的两阶段

① 第一阶段只保留地址空间，特别适合线程正在创建大的动态数据结构的情况。为防止进程的其他线程占用这段虚拟地址空间，可预留所需要的虚拟地址域，并用一个虚拟地址描述符记录之。

② 第二阶段在已保留的地址空间中分配物理主存，并建立虚实映射。

# 4．共享主存和写时复制技术

- 是用来节约主存的一种优化技术。
- 若没有进程向共享主存页写时，两个进程就共享之。一旦一个进程向共享主存页写时，系统就把该页复制到主存的另一个页框中，并更新该进程页表，指向此复制的页框。

# 16.2 Windows 地址转换

# 16.2.1 地址转换所涉及的数据结构

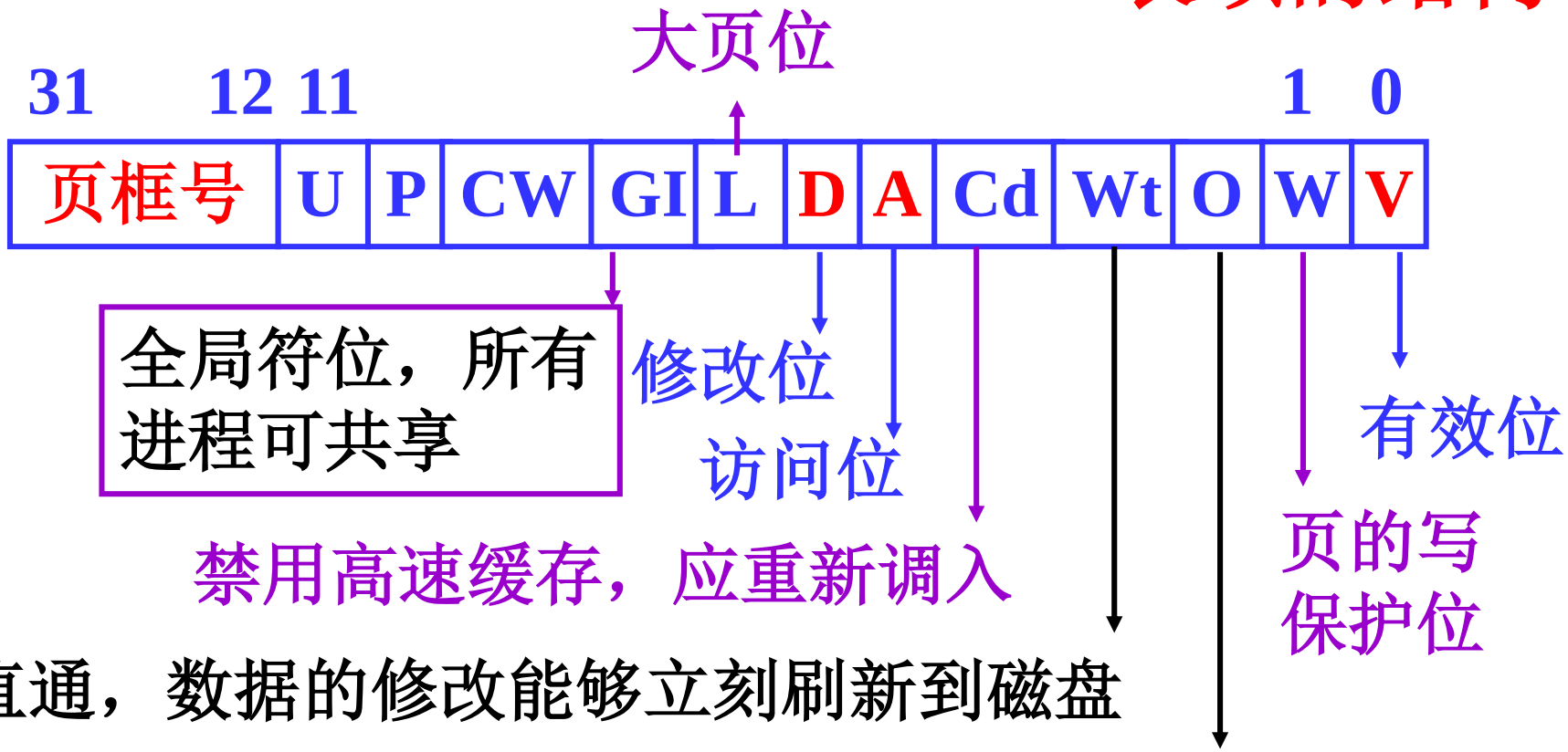
- 采用二级页表结构：页目录表、页表。

# 1．页表和页目录表的结构

- 页目录表的每一项记录一个页表的地址。进程页表不再占用连续的主存空间。
- 32位的虚地址被分解为：页目录索引、页表索引、页内字节索引。
- 进程页目录的物理地址被保存在核心进程块中(KPROCESS)。

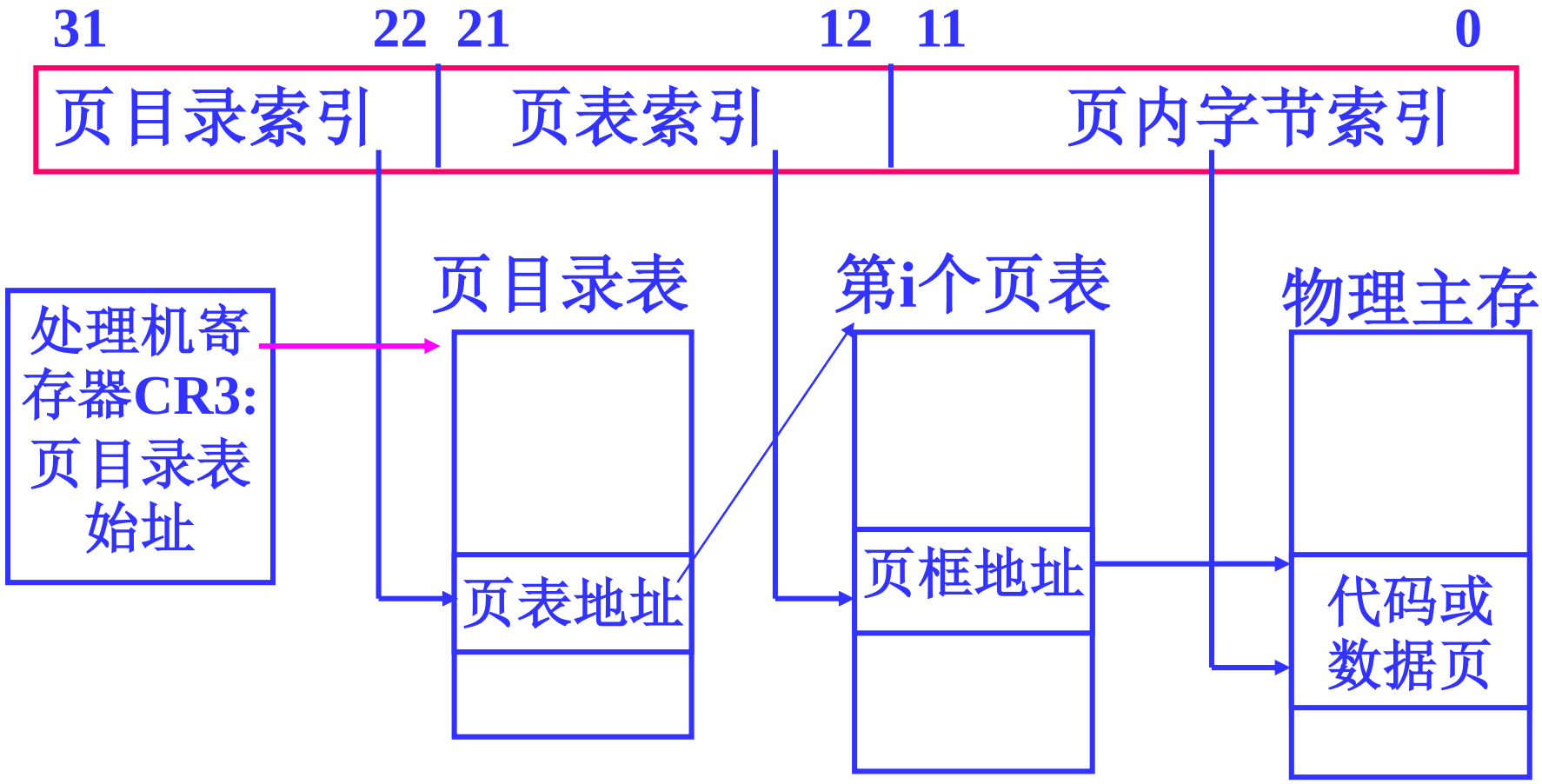
❖ 页表和页目录表的结构相同

**x86硬件页表项的结构**



# 2. 虚拟地址变换过程

32位的地址被分解为三部分：



# 3．页框数据库

## （1）页框数据库的作用

- ✓进程页表用于跟踪虚拟页在物理主存的位置，页框数据库用于跟踪物理主存的使用情况。
- ✓页框数据库是一个数组，其索引号从0到主存的页框总数-1。

主存中的页框可能处于以下八种状态：

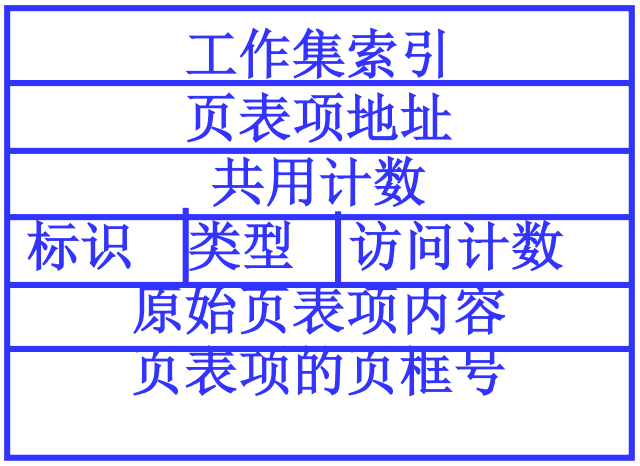
1. 活动（有效）： 是进程工作集的一部分。
2. 过渡： 不在进程工作集中，但未被破坏或该页框的I/O正在进行中
3. 备用：已不属于工作集，页表项仍然指向该页，但被标记为无效和处于过渡状态。
4. 更改：已不属于工作集，修改未写磁盘，页表项仍指向该页，被标记为无效和处于过渡状态。
5. 更改不写入： 更改但不将该页框写入磁盘
6. 空闲： 不属于任何一个用户进程的空闲页框
7. 零初始化： 由零初始化线程进行了清零的空闲页框
8. 坏页框。

# 页框链

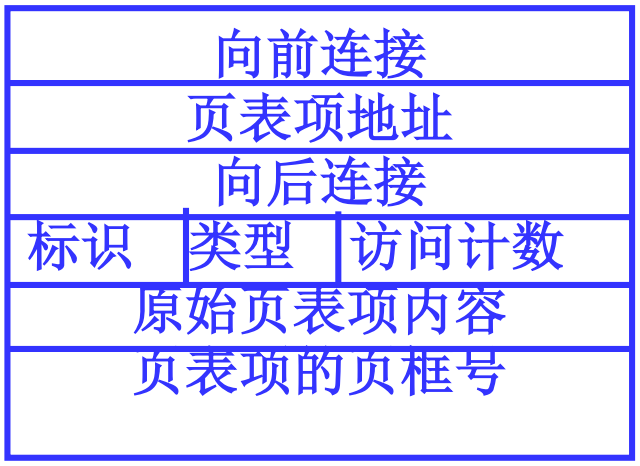
- 为了便于快速定位一个页框，系统把页框数据库中未被使用的、状态相同的页框链在一起，形成六个链表：零初始化的、空闲的、备用的、更改的、更改不写入的、坏的。
- 活动页框和过渡页框不在链表中，活动页框由进程的页表来管理。

# （2）页框数据结构

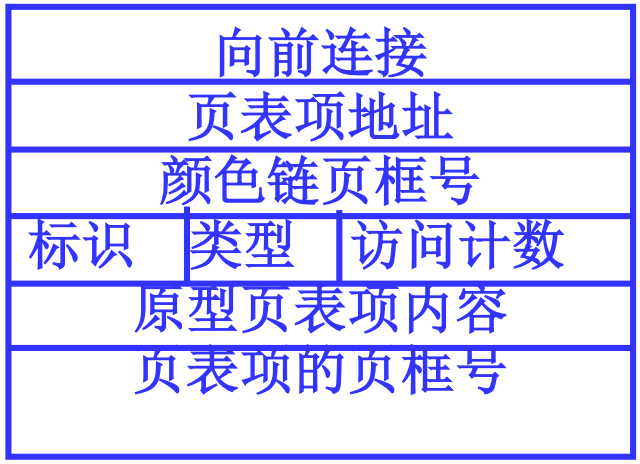
- 为了描述页框数据库中的各页的状态，每个页框都对应一个数据结构，叫**PFN**。
- 有四种页框数据结构，用于描述不同状态的页。



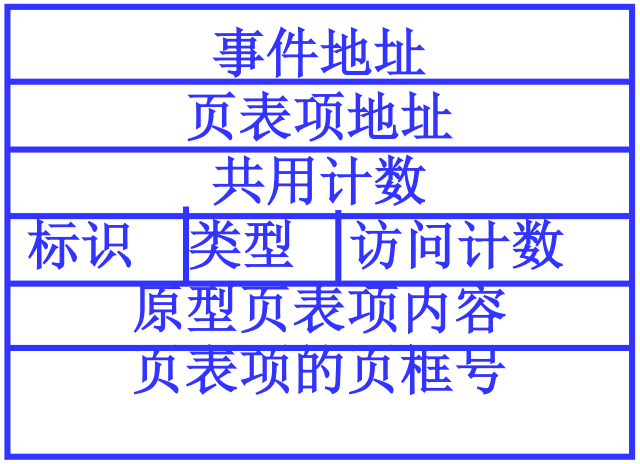

a. 活动的PFN




b. 备用或更改链表中PFN




c. 零或空闲链表中PFN

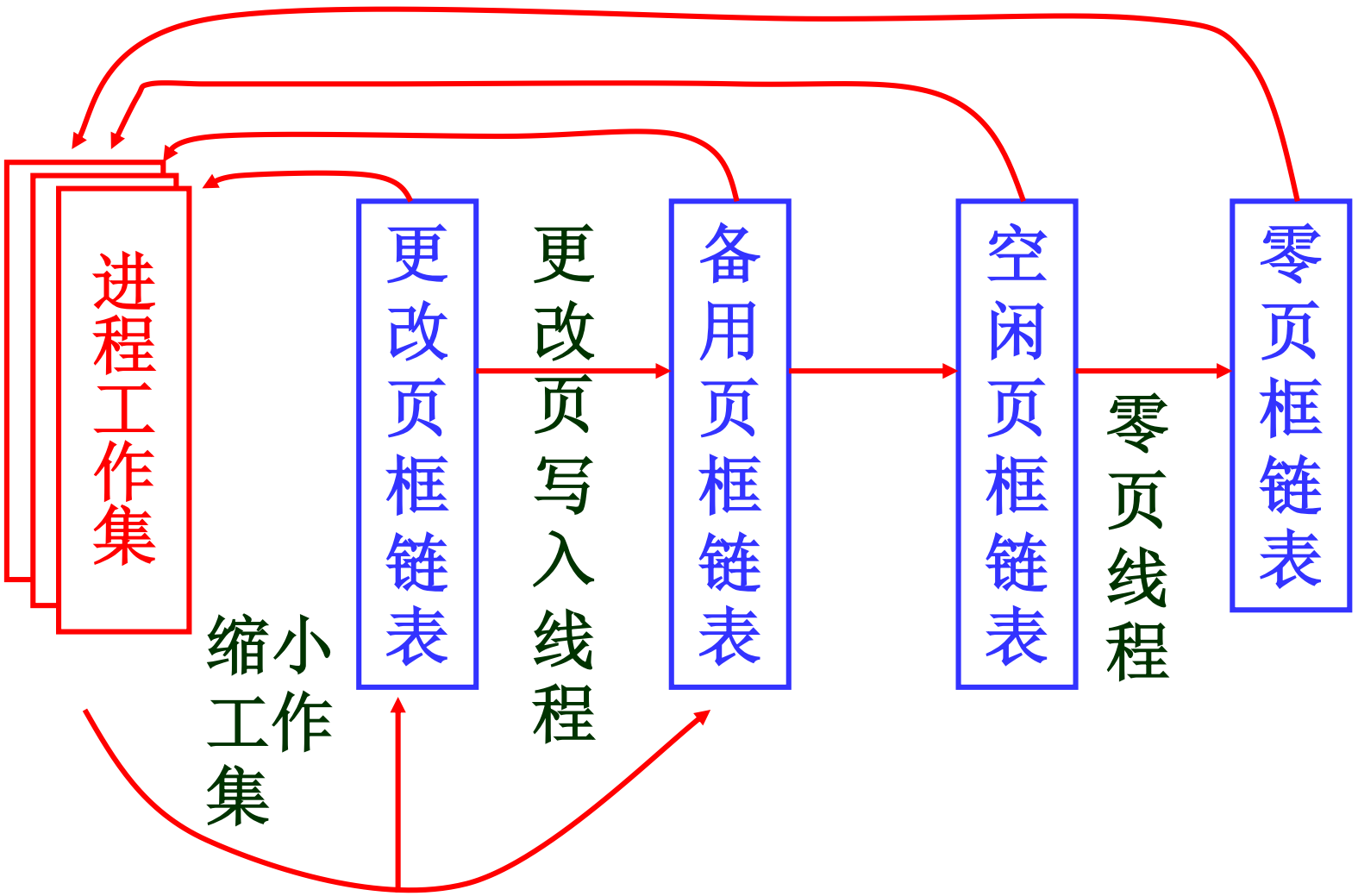



d. 正在进行I/O中PFN

# 四种页框数据结构的相同域

- 页表项地址：指向此页的页表项的虚拟地址
- 类型：页框类型
- 访问计数：对此页的访问计数
- 原型页表项内容：表示该页框项包含指向该原始页框的页表项的内容。此页是可共享的
- 页表项的页框号：
- 标识：见表16.1

# 页框的状态转换图



# 16.2.2 页错误处理

1. 无效页处理：当被访问的页无效时，产生“无效页错误”。内核中断处理程序将这类错误分派给存储器故障处理程序。

- 访问一个未知页，其页表项为0或页表不存在。
- 所访问的页没有驻留在主存，在外存页文件或映像文件
- 所访问的页在备用链表或更改链表。
- 页访问违约。与访问权限不符。

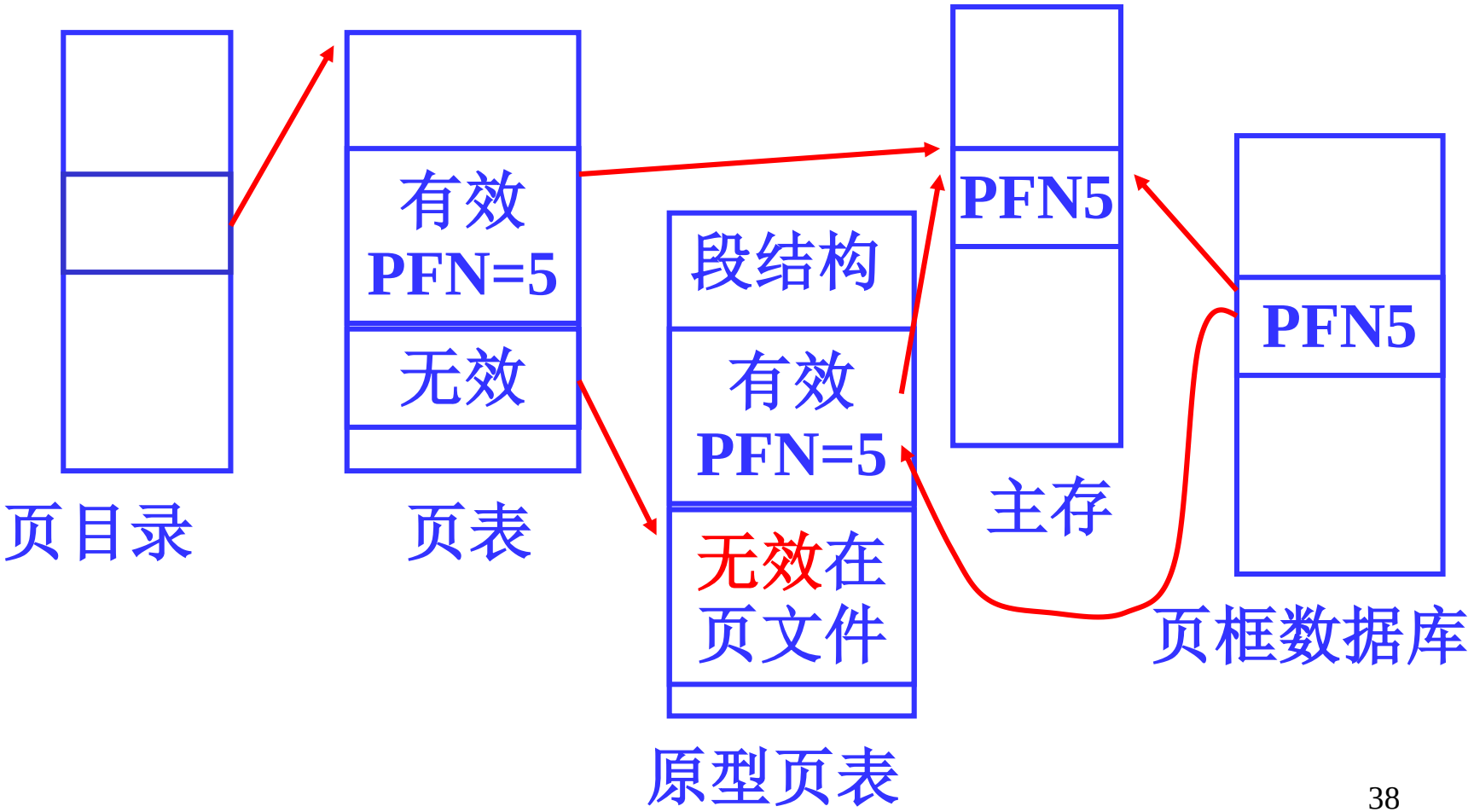
# 2. 原型页表项

- 当一个页框被两个或多个进程共享时，存储器管理器依靠一个称为“原型页表项”（Prototype PTE）的软件结构来映射这些被共享的页框。
- 当一个区域对象第一次被创建时，这些原型页表项“按段”同时被创建。

# 引入原型页表项的目的

- 引入原型页表项是为了尽可能地减少对各进程的页表项的影响。
- 当一个共享页被换出到磁盘时，只需修改原型页表项，各进程页表项仍保持不变。当共享页被换入主存时，系统只需更新原型页表项，使之指向新分配的页框。此后，当进程访问该页时，实际的页表项才得以更新。

# 原型页表项PTE，页框数据结构PFN



# 16.3 页调度策略

① 调页策略：将所缺的页及其前后的一些页装入主存。试图减少调页I/O次数。

② 置页策略：将虚拟页放到物理主存。

③ 置换策略：在多处理器系统中，采用了局部先进先出置换策略。而在单处理器系统中，更接近于最近最久未使用策略(LRU，也称为“时钟页面置换算法”)。实现局部和全局置换的一种组合模式。

1. 进程工作集：为每个进程分配一定数量的页框，称为“进程工作集”。当缺页错误产生时，检测进程的工作集限制和系统中空闲主存的数量。如果系统有足够多的空闲页框，则允许工作集规模增大；否则，只能替换工作集中的页。
2. 系统工作集：为可分页的系统代码和数据分配一定数量的页框，称为“系统工作集”。

# 平衡工作集管理器

- 是在系统初始化时创建的一个系统线程。
- 调整进程和系统工作集。
- 当系统缺页率很高，或者空闲链表中的页框太少时，存储器管理器就会唤醒平衡工作集管理器，开始修剪工作集。