操作系统原理

PRINCIPLES OF OPERATING SYSTEM

北京大学计算机科学技术系 陈向群
Department of computer science and Technology
Peking University
2015 春季

第8讲

存储模型2
——虚拟存储技术

存储管理

- 虚拟存储技术
- 页表及页表项的设计
- 地址转换过程及TLB引入
- 软件相关策略
- 页面置换算法
- 其他相关技术

虚拟内存、虚拟地址空间、虚拟地址

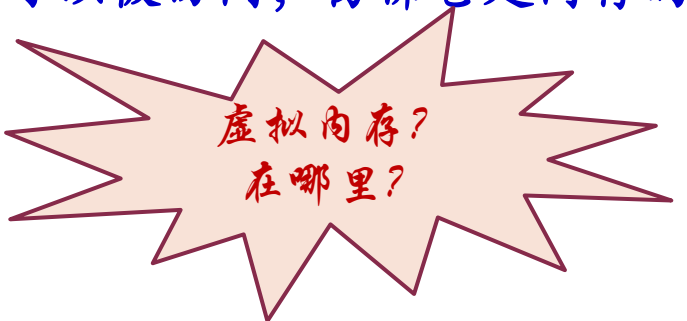
虚拟存储技术

VIRTUAL MEMORY

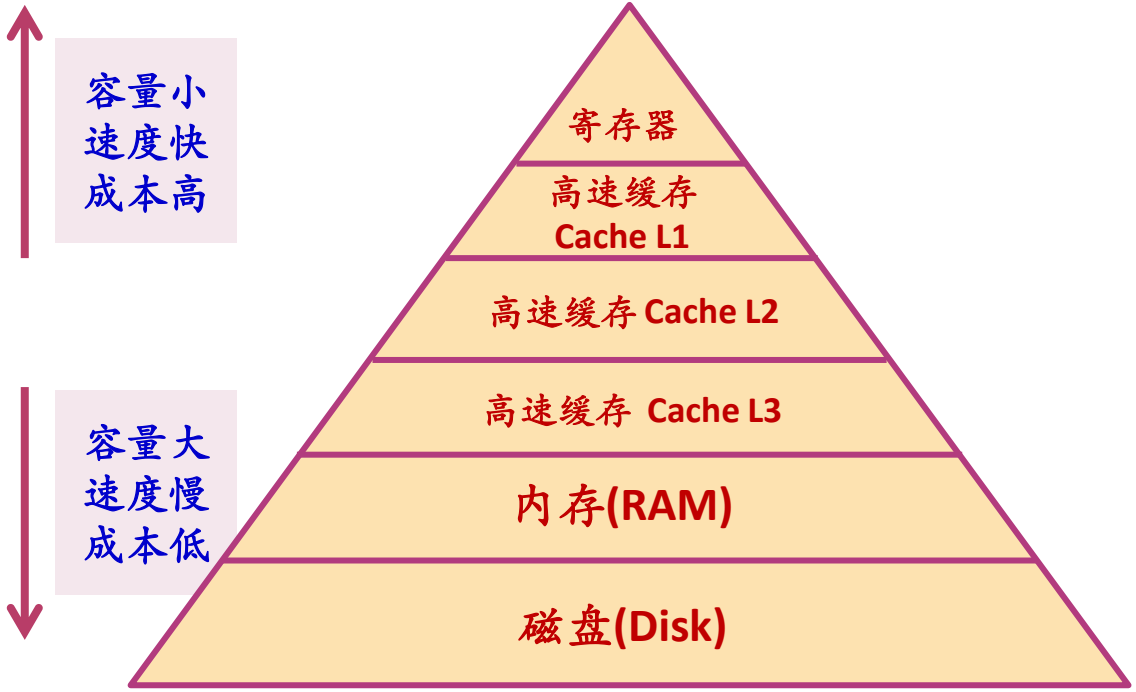
虚拟存储技术

- 所谓虚拟存储技术是指：当进程运行时，先将其一部分装入内存，另一部分暂留在磁盘，当要执行的指令或访问的数据不在内存时，由操作系统自动完成将它们从磁盘调入内存的工作

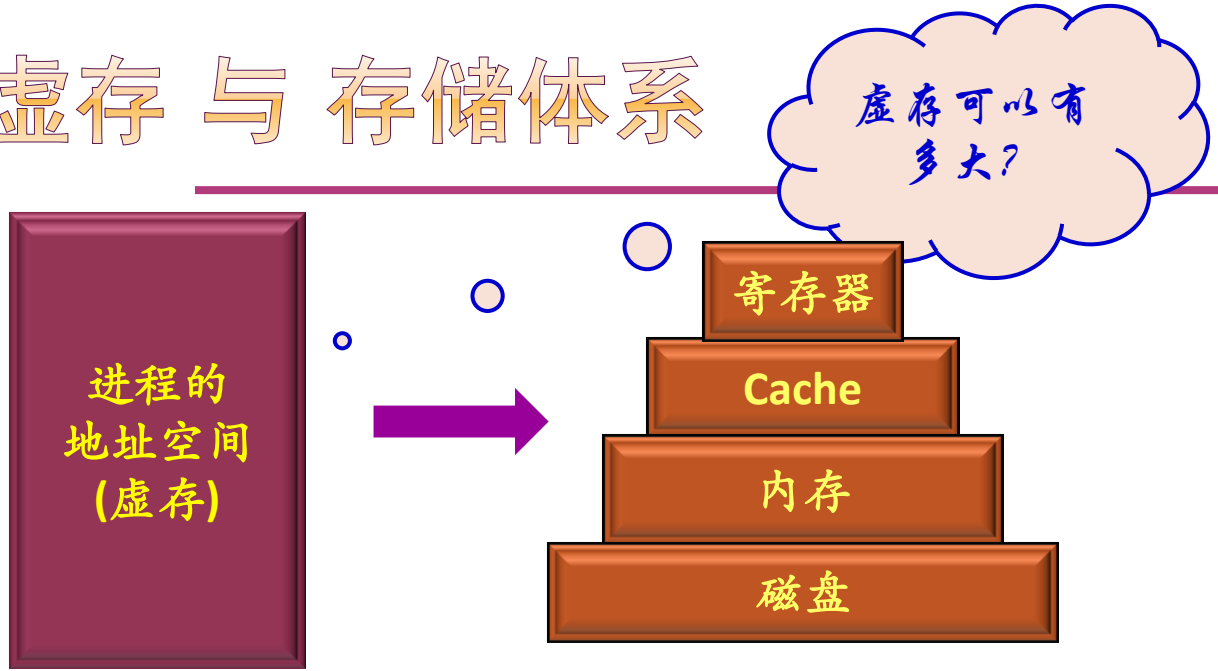
- 虚拟地址空间 即为 *分配给进程的虚拟内存*
- 虚拟地址 是 *在虚拟内存中指令或数据的位置，该位置可以被访问，仿佛它是内存的一部分*



存储器的层次结构


容量小
速度快
成本高
寄存器
高速缓存
Cache L1
高速缓存 Cache L2
高速缓存 Cache L3
内存(RAM)
磁盘(Disk)
容量大
速度慢
成本低

虚存 与 存储体系



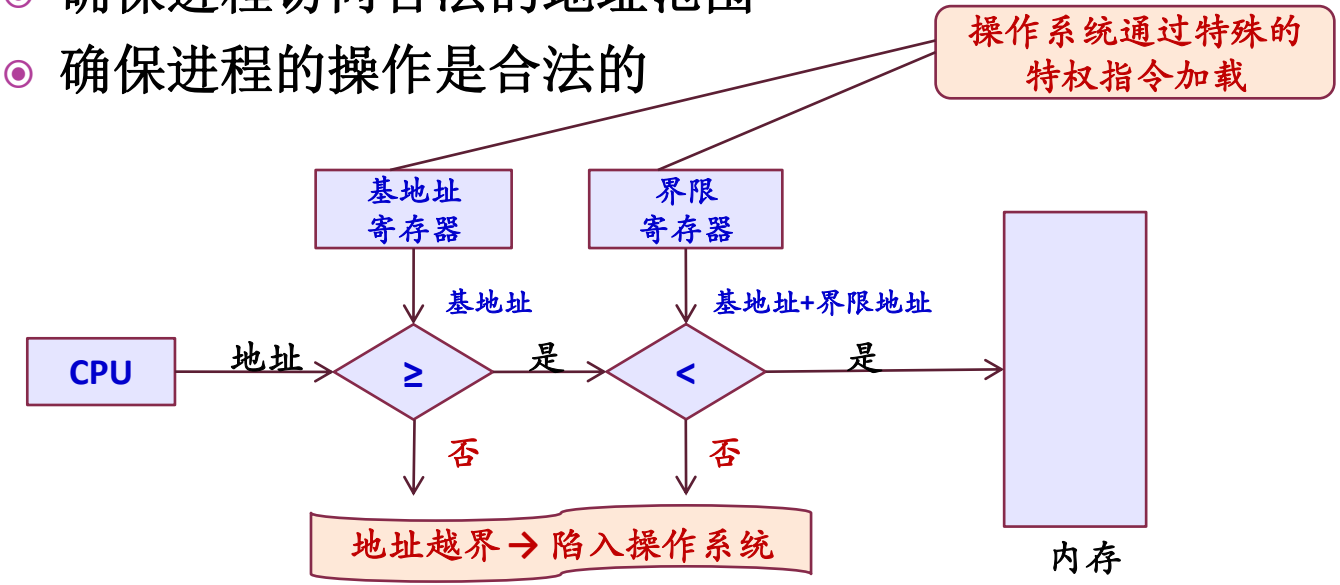
- 把内存与磁盘有机地结合起来使用，从而得到一个容量很大的“内存”，即虚存
- 虚存是对内存的抽象，构建在存储体系之上，由操作系统协调各存储器的使用
- 虚存提供了一个比物理内存空间大得多的地址空间

存储保护

防止地址越界

防止访问越权

- 确保每个进程有独立的地址空间
- 确保进程访问合法的地址范围
- 确保进程的操作是合法的



虚拟页式（PAGING）

虚拟存储技术 + 页式存储管理方案

→ 虚拟页式存储管理系统

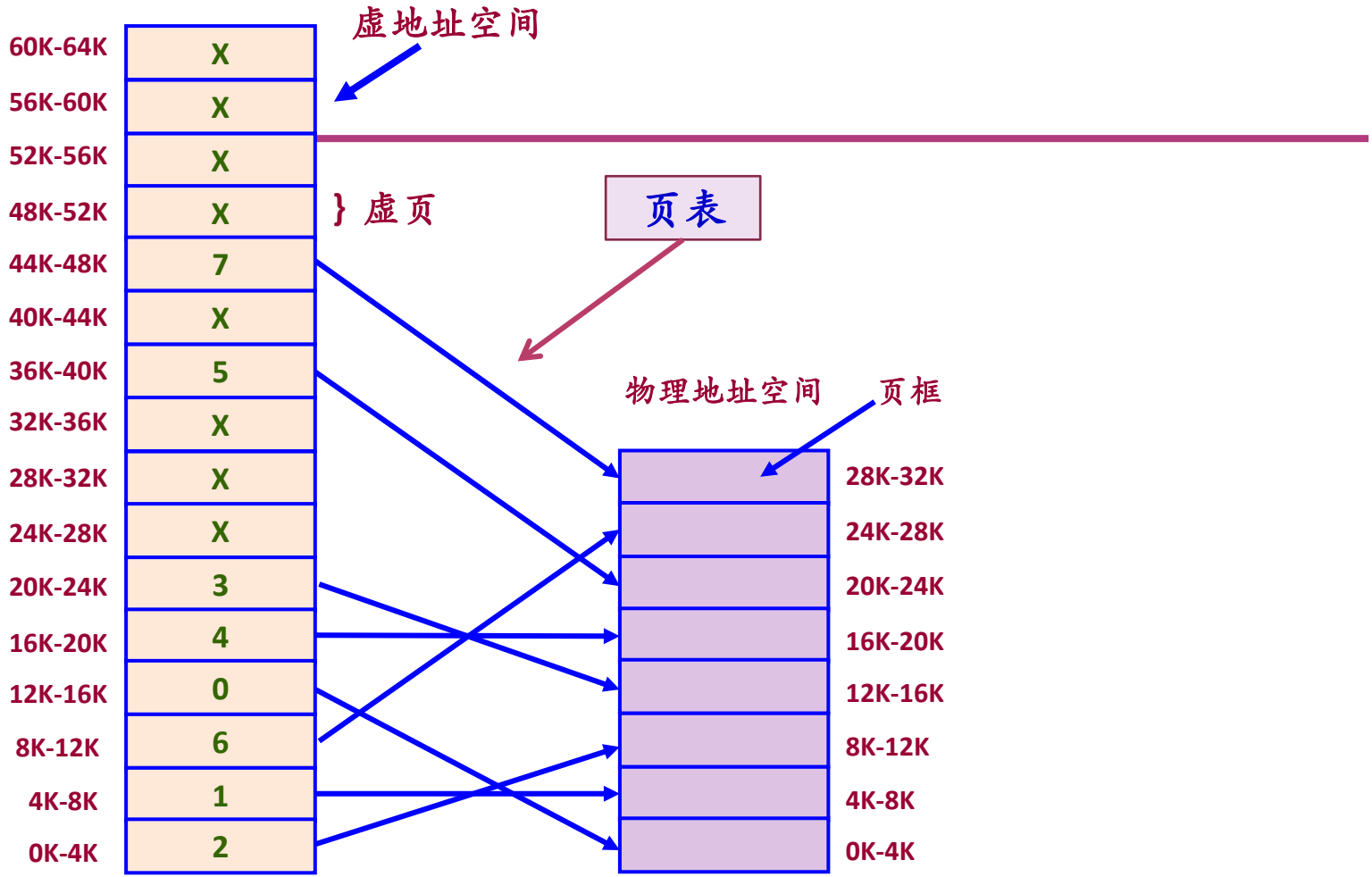
具体有两种方式
1、请求调页（demand paging）√
2、预先调页（prepaging）

- 基本思想
  - 进程开始运行之前，不是装入全部页面，而是装入一个或零个页面
  - 之后，根据进程运行的需要，动态装入其他页面
  - 当内存空间已满，而又需要装入新的页面时，则根据某种算法置换内存中的某个页面，以便装入新的页面

以CPU时间和磁盘空间换取昂贵内存空间，这是操作系统中的资源转换技术

多级页表、X86的页表项示例

页表及页表项的设计


虚地址空间
60K-64K
56K-60K
52K-56K
48K-52K
44K-48K
40K-44K
36K-40K
32K-36K
28K-32K
24K-28K
20K-24K
16K-20K
12K-16K
8K-12K
4K-8K
0K-4K
X
X
X
X
7
X
5
X
X
X
3
4
0
6
1
2
} 虚页
页表
物理地址空间
页框
28K-32K
24K-28K
20K-24K
16K-20K
12K-16K
8K-12K
4K-8K
0K-4K

页表表项设计

- 页表由页表项组成
- 页框号、有效位、访问位、修改位、保护位
  - ✓ 页框号（内存块号、物理页面号、页帧号）
  - ✓ 有效位（驻留位、中断位）：表示该页是在内存还是在磁盘
  - ✓ 访问位：引用位
  - ✓ 修改位：此页在内存中是否被修改过
  - ✓ 保护位：读/可读写

通常，页表项是硬件设计的

关于页表

多级页表

- 32位虚拟地址空间的页表规模？

  页面大小为4K；页表项大小为4字节

  则：一个进程地址空间有 ？页 2^{20}

  其页表需要占 ？页（页表页） 1024

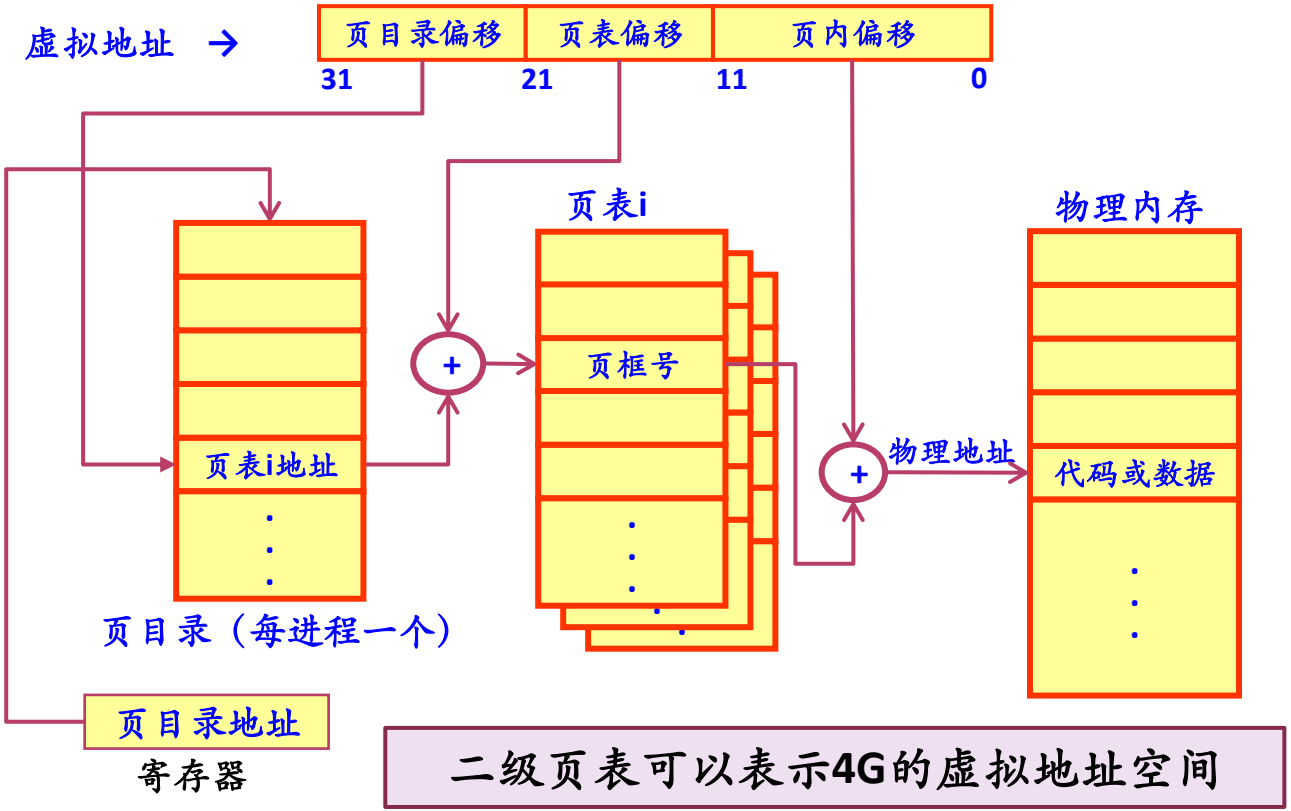
- 64位虚拟地址空间

  页面大小为4K；页表项大小为8字节

  页表规模： 32,000 TB

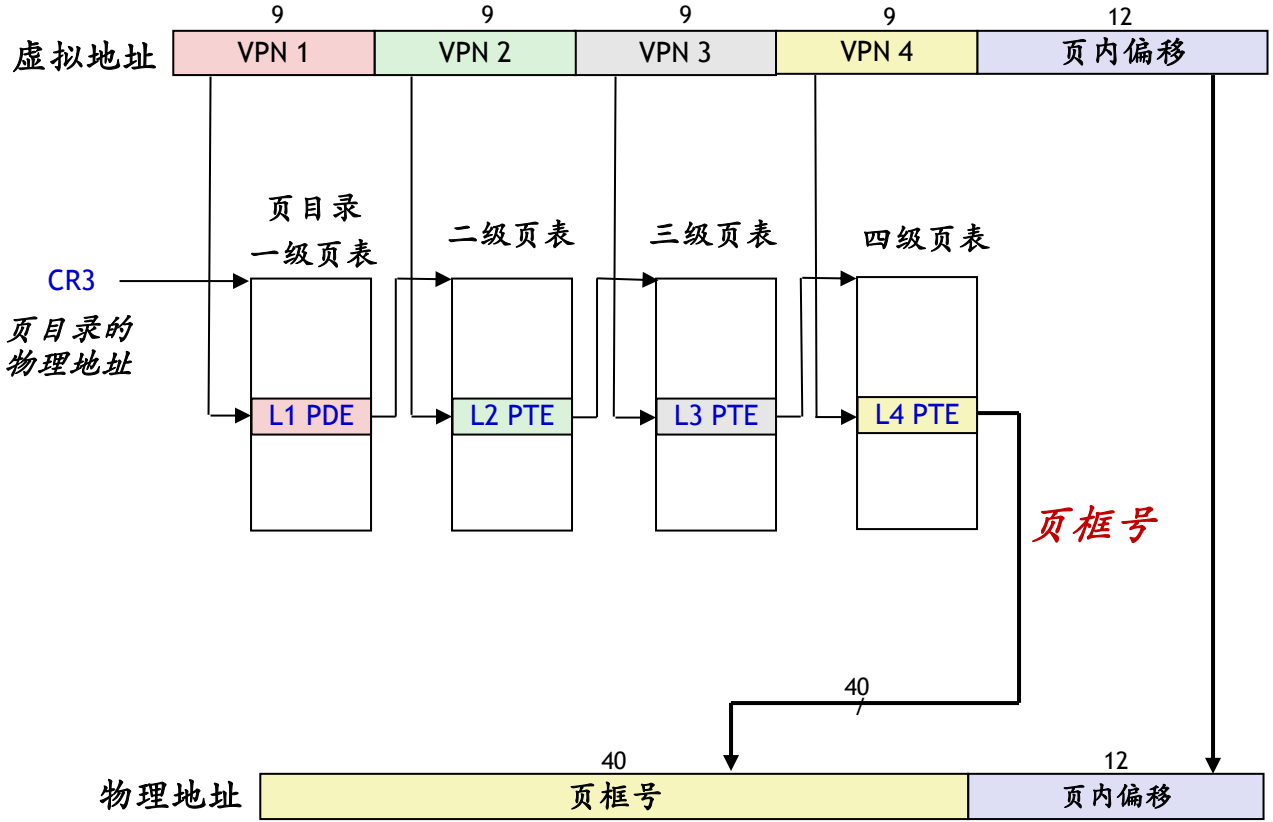
- 页表页在内存中若不连续存放，则需要引入页表页的地址索引表 → **页目录（Page Directory）**

二级页表结构及地址映射



CORE I7 页表结构

虚拟地址空间2^{48}



I386页目录项和页表项

页目录项 PDE (Page Directory Entry)

页表项 PTE (Page Table Entry)

PFN(Page Frame Number)：页框号
P(Present)：有效位
A(Accessed)：访问位
D(Dirty)：修改位
R/W(Read/Write)：只读/可读写
U/S(User/Supervisor)：用户/内核
PWT(Page Write Through)：缓存写策略
PCD(Page Cache Disable)：禁止缓存
PS(Page Size)：大页4M

引入反转(倒排)页表

- 地址转换

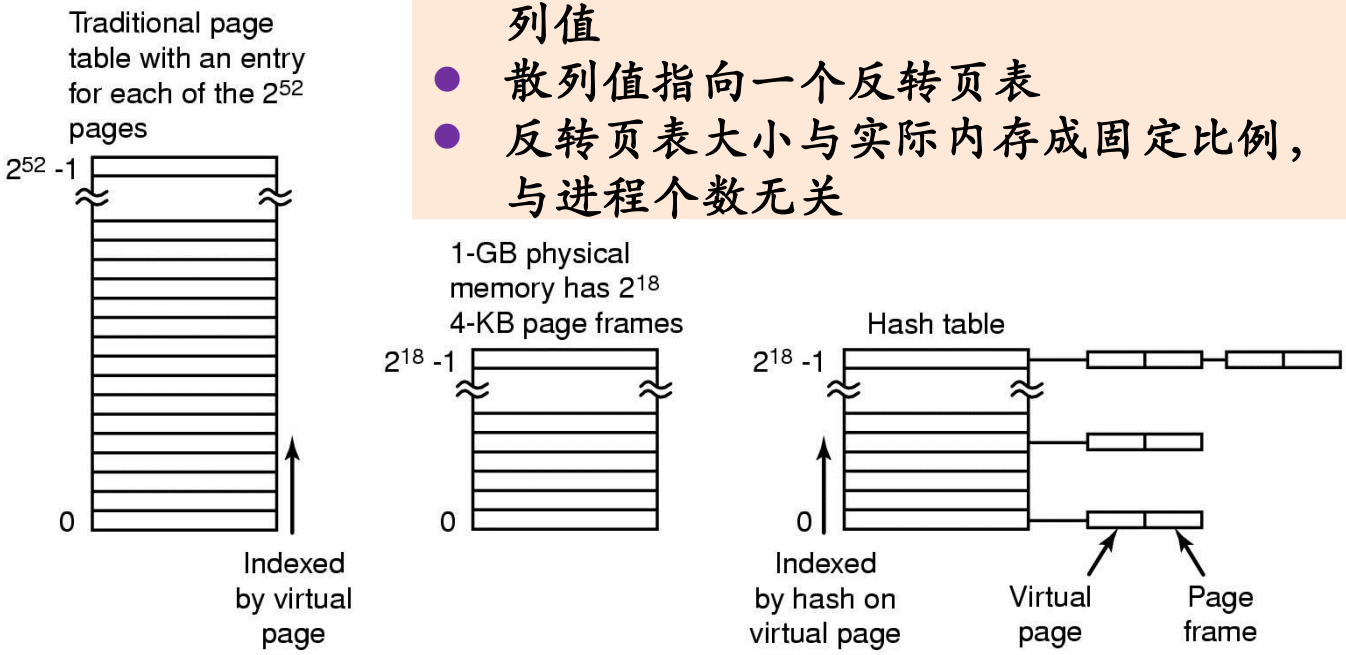
  从虚拟地址空间出发：虚拟地址 → 查页表 → 得到页框号 → 形成物理地址

  每个进程一张页表

- 解决思路
  - 从物理地址空间出发，系统建立一张页表
  - 页表项记录进程i的某虚拟地址(虚页号)与页框号的映射关系

反转(倒排)页表设计

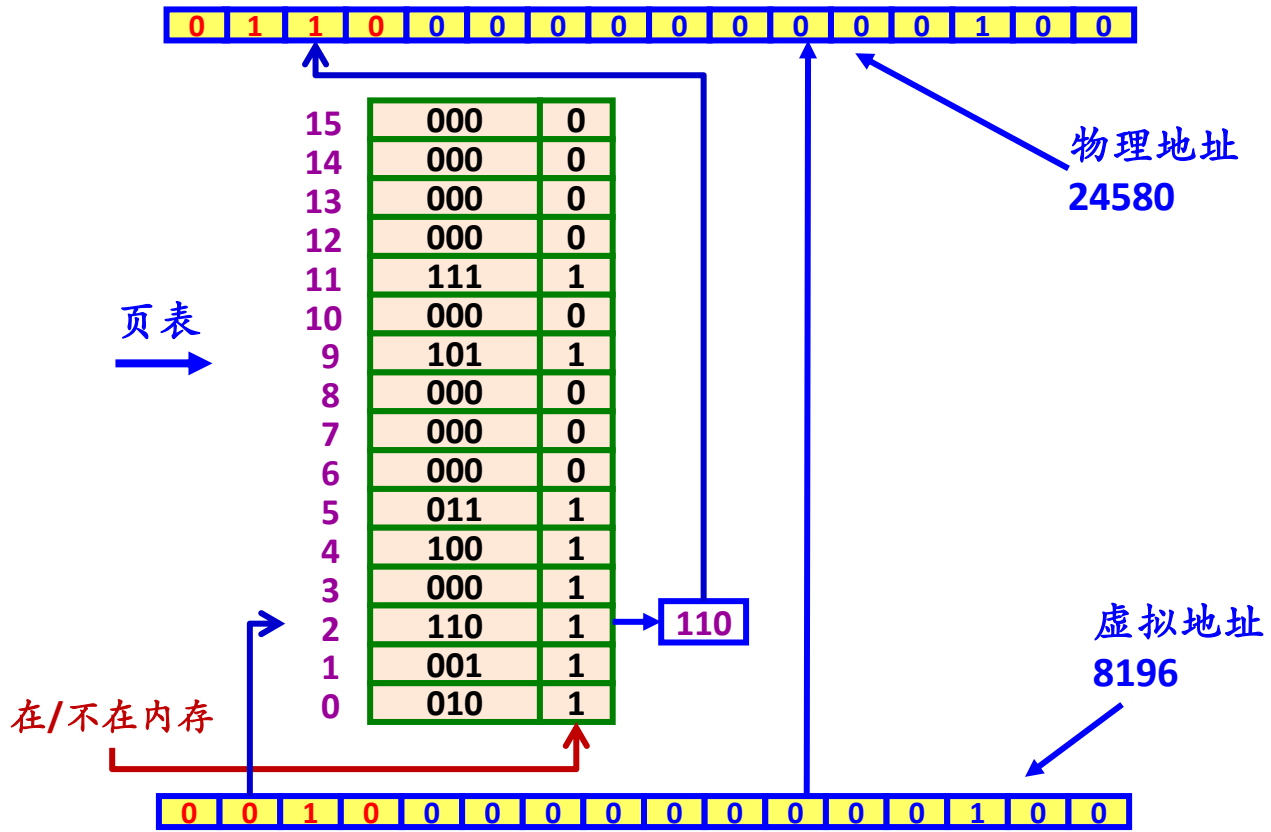
- PowerPC、UltraSPARC和IA-64 等体系结构采用
- 将虚拟地址的页号部分映射到一个散列值
- 散列值指向一个反转页表
- 反转页表大小与实际内存成固定比例，与进程个数无关



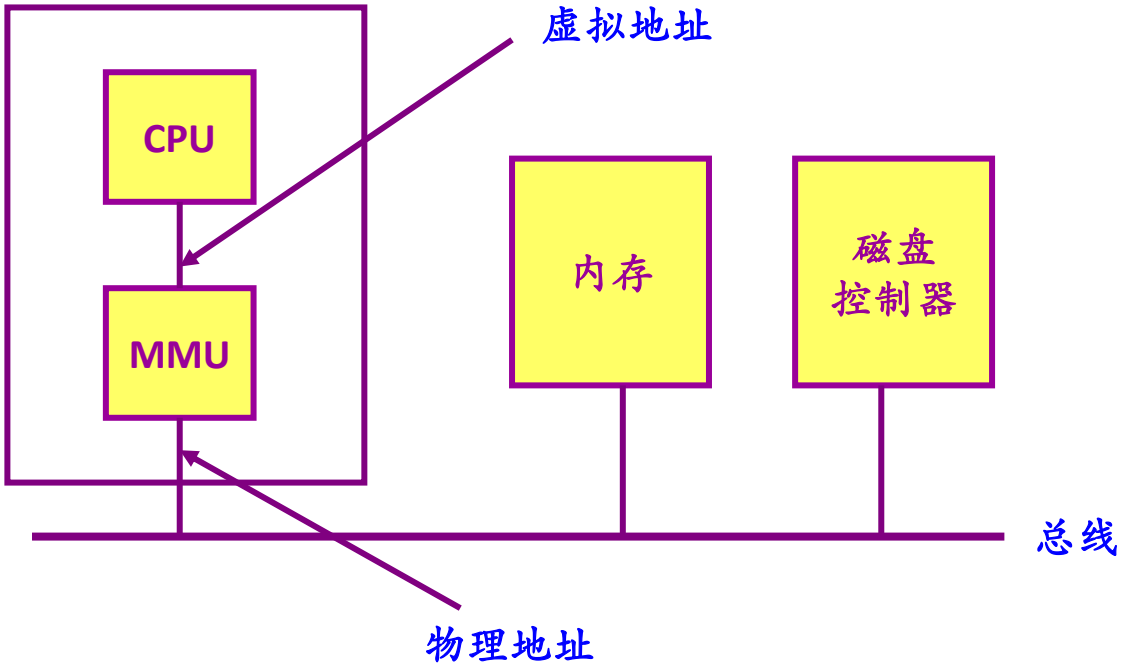
MMU、快表(TLB)

地址转换过程及TLB引入

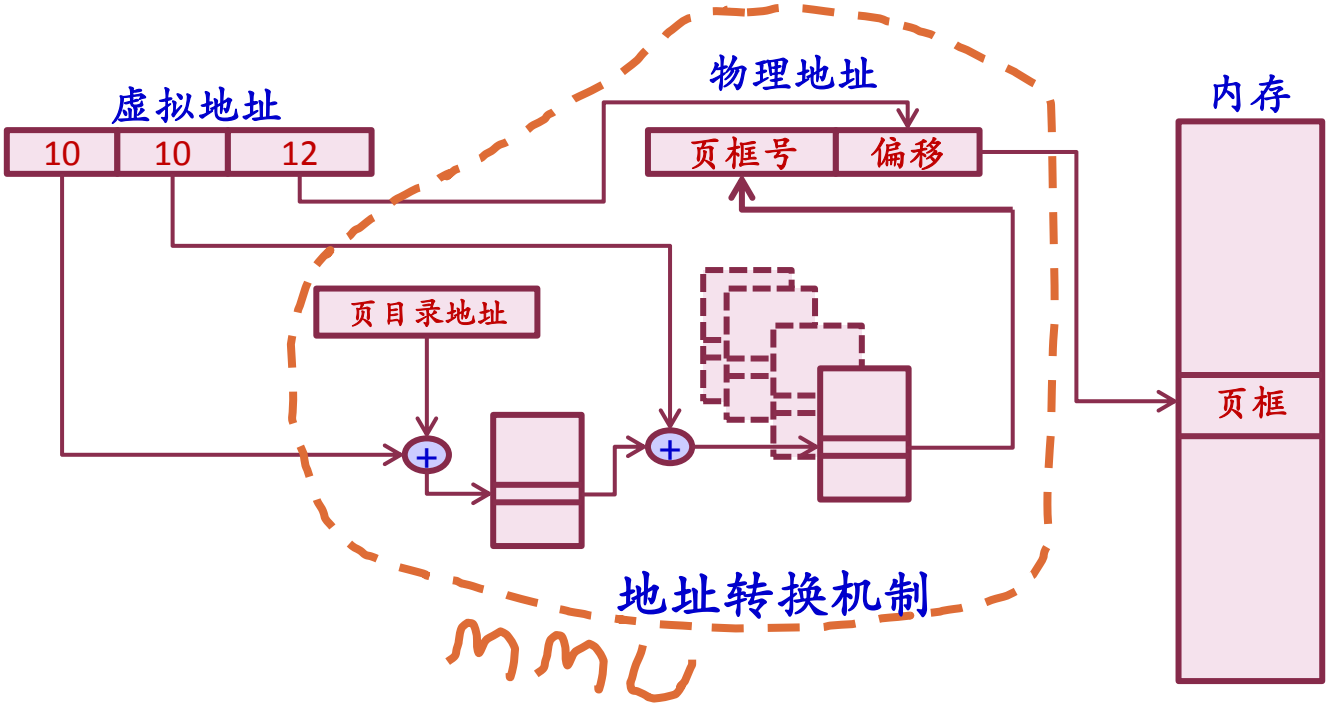
地址转换过程示意


0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0
物理地址
24580
页表
15 000 0
14 000 0
13 000 0
12 000 0
11 111 1
10 000 0
9 101 1
8 000 0
7 000 0
6 000 0
5 011 1
4 100 1
3 000 1
2 110 1
1 001 1
0 010 1
110
虚拟地址
8196
在/不在内存
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0

MMU：内存管理单元



地址转换(映射)



快表(TLB)的引入

问题

- 页表 → 两次或两次以上的内存访问
- CPU的指令处理速度与内存指令的访问速度差异大，CPU的速度得不到充分利用

如何加快地址映射速度，以改善系统性能？

程序访问的局部性原理 → 引入快表(TLB)

快表是什么？

快表的大小、位置

快表的置换问题？

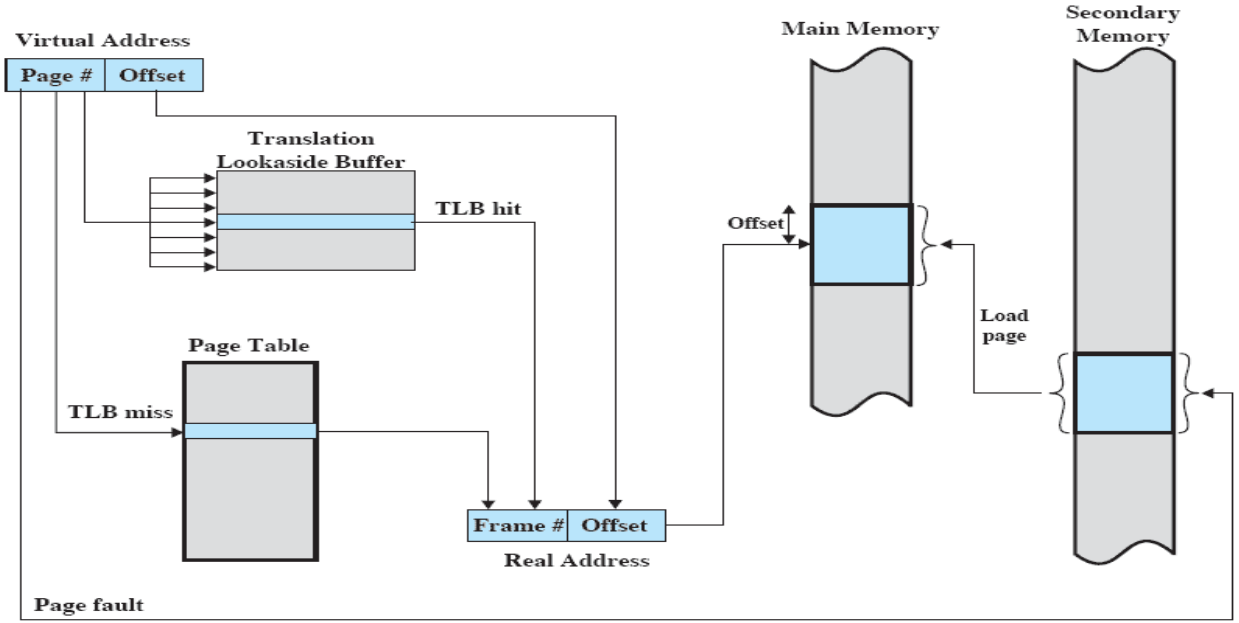
- TLB — Translation Look-aside Buffers

在CPU中引入的高速缓存（Cache），可以匹配CPU的处理速率和内存的访问速度

一种随机存取型存储器，除连线寻址机制外，还有接线逻辑，能按特定的匹配标志在一个存储周期内对所有的字同时进行比较

- 相联存储器（associative memory）
  特点：按内容并行查找
- 保存正在运行进程的页表的子集(部分页表项)

加入TLB后地址转换过程示意



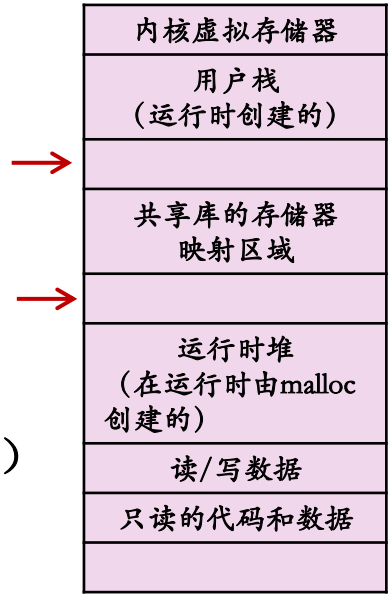
选自William Stallings

缺页异常、受保护、非法地址.....

页错误(页故障)
PAGE FAULT

页错误PAGE FAULT

- 又称 页面错误、页故障、页面失效
- 地址转换过程中硬件产生的异常
- 具体原因
  - 所访问的虚拟页面没有调入物理内存
    → 缺页异常
  - 页面访问违反权限（读/写、用户/内核）
  - 错误的访问地址
  - ……



缺页异常处理

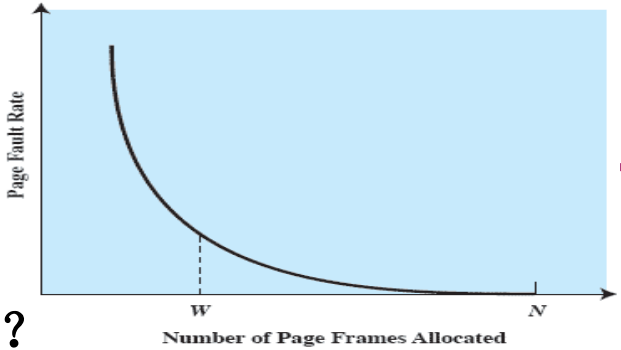
预取一些页面

- 是一种Page Fault
- 在地址映射过程中，硬件检查页表时发现所要访问的页面不在内存，则产生该异常——缺页异常
- 操作系统执行缺页异常处理程序：获得磁盘地址，启动磁盘，将该页调入内存。
  - 如果内存中有空闲页框，则分配一个页框，将新调入页装入，并修改页表中相应页表项的有效位及相应的页框号
  - 若内存中没有空闲页框，则要置换内存中某一页框；若该页框内容被修改过，则要将其写回磁盘

驻留集、置换范围、置换策略、清除策略

软件相关策略

驻留集



- 驻留集大小

  给每个进程分配多少页框？

- 固定分配策略

  进程创建时确定

  可以根据进程类型（交互、批处理、应用类）或者基于程序员或系统管理员的需要来确定

- 可变分配策略

  根据缺页率评估进程局部性表现

✓ 缺页率高→增加页框数

✓ 缺页率低→减少页框数

✓ 系统开销

置换问题

- 置换范围

  计划置换页面的集合是局限在产生缺页中断的进程，还是所有进程的页框？

- 置换策略

  在计划置换的页框集合中，选择换出哪一个页框？

置换范围

- 局部置换策略

  仅在产生本次缺页的进程的驻留集中选择
- 全局置换策略

  将内存中所有未锁定的页框都作为置换的候选

	局部置换	全局置换
固定分配	√	--
可变分配	√	√

1、当一个新进程装入内存时，给它分配一定数目的页框，然后填满这些页框

2．当发生一次缺页异常时，从产生缺页异常进程的驻留集中选择一页用于置换

3．不断重新评估进程的页框分配情况，增加或减少分配给它的页框，以提高整体性能

置换策略

典型思路

- 决定置换当前内存中的哪一个页框
- 所有策略的目标

  → *置换最近最不可能访问的页*
- 根据局部性原理，最近的访问历史和最近将要访问的模式间存在相关性，因此，大多数策略都*基于过去的行为来预测将来的行为*
- 注意：置换策略设计得越精致、越复杂，实现的软硬件开销就越大

- 约束：不能置换被锁定的页框

页框锁定

为什么要锁定页面？

- 采用虚存技术后

  开销 → 使进程运行时间变得不确定

- 给每一页框增加一个锁定位
- 通过设置相应的锁定位，不让操作系统将进程使用的页面换出内存，避免产生由交换过程带来的不确定的延迟
- 例如：操作系统核心代码、关键数据结构、I/O缓冲区

特别是正在I/O的内存页面

Windows中的VirtualLock和VirtualUnLock函数

清除策略(1/2)

- 清除：从进程的驻留集中收回页框
- 虚拟页式系统工作的最佳状态：发生缺页异常时，系统中有大量的空闲页框
- ***结论：在系统中保存一定数目的空闲页框供给比使用所有内存并在需要时搜索一个页框有更好的性能***

- 设计一个分页守护进程（paging daemon），多数时间睡眠着，可定期唤醒以检查内存的状态
- 如果空闲页框过少，分页守护进程通过预定的页面置换算法选择页面换出内存
- 如果页面装入内存后被修改过，则将它们写回磁盘
分页守护进程可保证所有的空闲页框是“干净”的

清除策略(2/2)

- 当进程需要使用一个已置换出的页框时，如果该页框还没有被新的内容覆盖，将它从空闲页框集合中移出即可恢复该页面

页缓冲技术:

- 不丢弃置换出的页，将它们放入两个表之一：如果未被修改，则放到空闲页链表中，如果修改了，则放到修改页链表中
- 被修改的页定期写回磁盘(不是一次只写一个，大大减少I/O操作的数量，从而减少了磁盘访问时间)
- 被置换的页仍然保留在内存中，一旦进程又要访问该页，可以迅速将它加入该进程的驻留集合(代价很小)

置換算法1

页面置换(REPLACEMENT)算法

又称页面淘汰（替换）算法

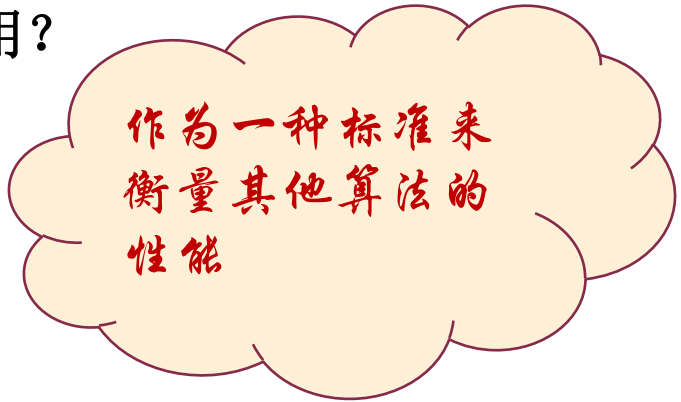
最佳算法→先进先出→第二次机会→时钟算法→最近未使用→最近最少使用→最不经常使用→老化算法→工作集→工作集时钟

设计思想　　算法实现　　算法应用

最佳页面置换算法（OPT）

- 设计思想：
  置换以后不再需要的或最远的将来才会用到的页面

- 实现？
- 作用？



先进先出算法（FIFO）

- 选择在内存中驻留时间最长的页并置换它

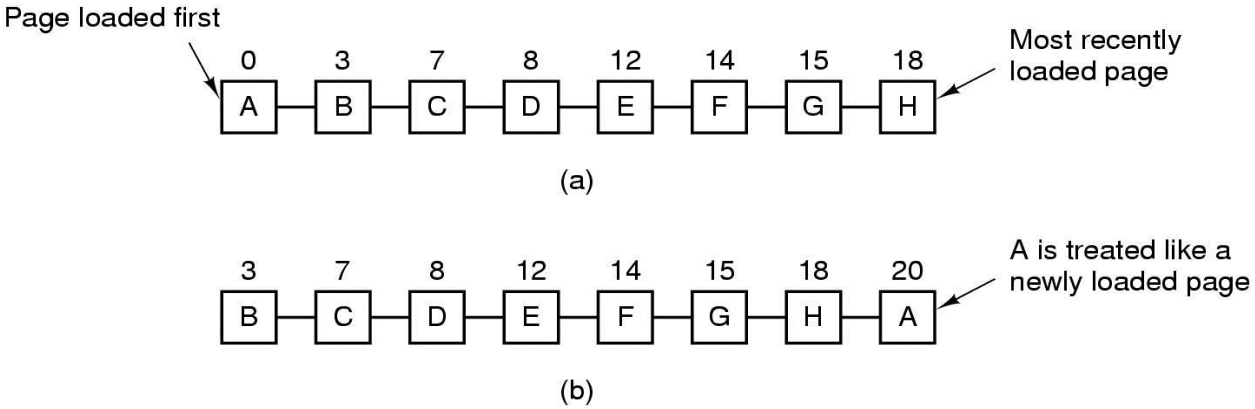
  对照：超市撤换商品

- 实现：页面链表法

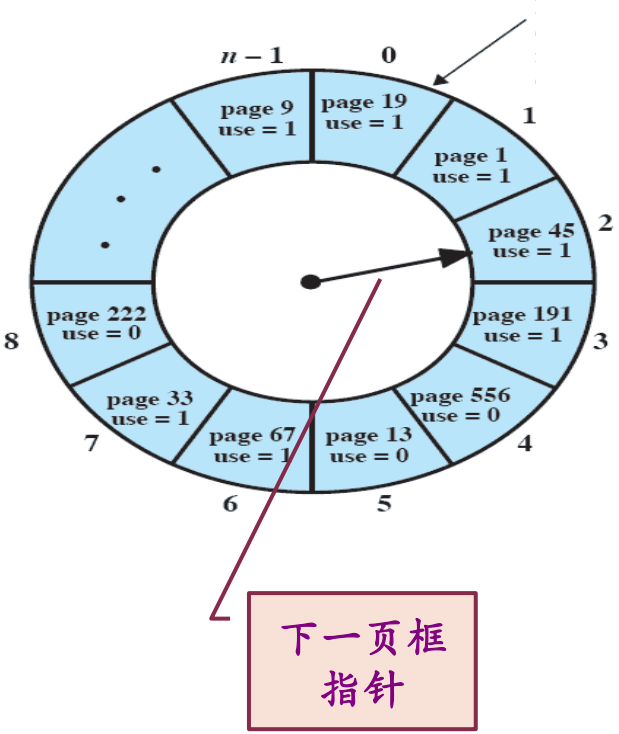
第二次机会算法(SCR)

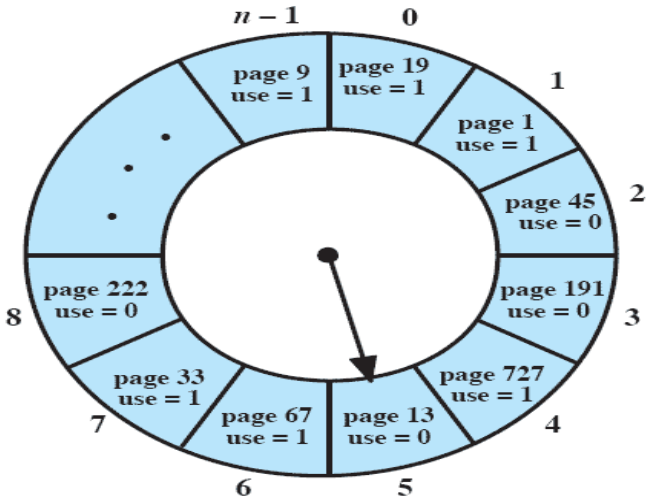
SCR-Second Chance

按照先进先出算法选择某一页面，检查其访问位R，如果为0，则置换该页；如果为1，则给第二次机会，并将访问位置0



时钟算法(CLOCK)


n – 1
0
1
2
3
4
5
6
7
8
page 9
use = 1
page 19
use = 1
page 1
use = 1
page 45
use = 1
page 191
use = 1
page 556
use = 0
page 13
use = 0
page 67
use = 1
page 33
use = 1
page 222
use = 0
下一页框
指针


n – 1
0
1
2
3
4
5
6
7
8
page 9
use = 1
page 19
use = 1
page 1
use = 1
page 45
use = 0
page 191
use = 0
page 727
use = 1
page 13
use = 0
page 67
use = 1
page 33
use = 1
page 222
use = 0

最近未使用算法(NRU)(1/2)

Not Recently Used

选择在最近一段时间内未使用过的一页并置换

实现：设置页表表项的两位

访问位（R）， 修改位（M）

如果硬件没有这些位，则可用软件模拟（做标记）

启动一个进程时，R、M位置0

R位被定期清零（复位）

最近未使用算法(NRU)(2/2)

发生缺页中断时，操作系统检查R，M：

第1类：无访问，无修改

第2类：无访问，有修改

第3类：有访问，无修改

第4类：有访问，有修改

算法思想：

随机从编号最小的非空类中选择一页置换

时钟算法实现

1．从指针的当前位置开始，扫描页框缓冲区，选择遇到的第一个页框（*r*=0；*m*=0）用于置换(本扫描过程中，对使用位不做任何修改)

2．如果第1步失败，则重新扫描，选择第一个（*r*=0；*m*=1）的页框(本次扫描过程中，对每个跳过的页框，将其使用位设置成0)

3．如果第2步失败，指针将回到它的最初位置，并且集合中所有页框的使用位均为0。重复第1步，并且，如果有必要，重复第2步。这样将可以找到供置换的页框

最近最少使用算法(LRU)

Least Recently Used

选择最后一次访问时间距离当前时间最长的一页并置换

即置换未使用时间最长的一页

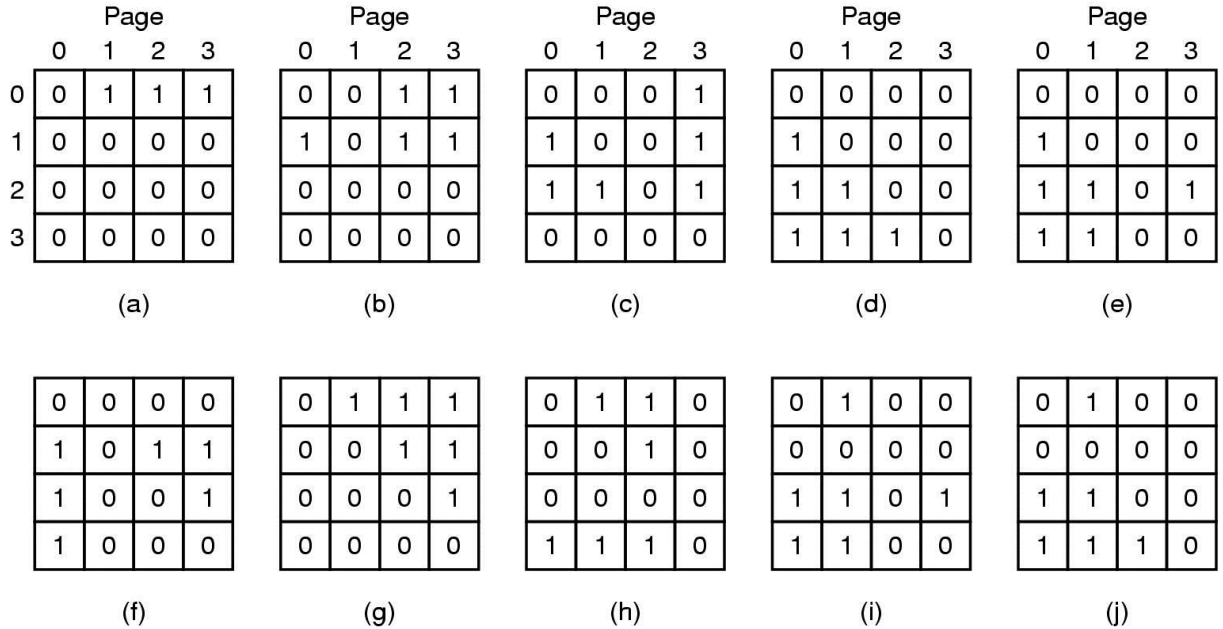
- 性能接近**OPT**

- 实现：时间戳 或 维护一个访问页的栈

  → 开销大

LRU算法的一种硬件实现

- 页面访问顺序0, 1, 2, 3, 2, 1, 0, 3, 2, 3



最不经常使用算法(NFU)

Not Frequently Used
选择访问次数最少的页面置换

- LRU的一种软件解决方案

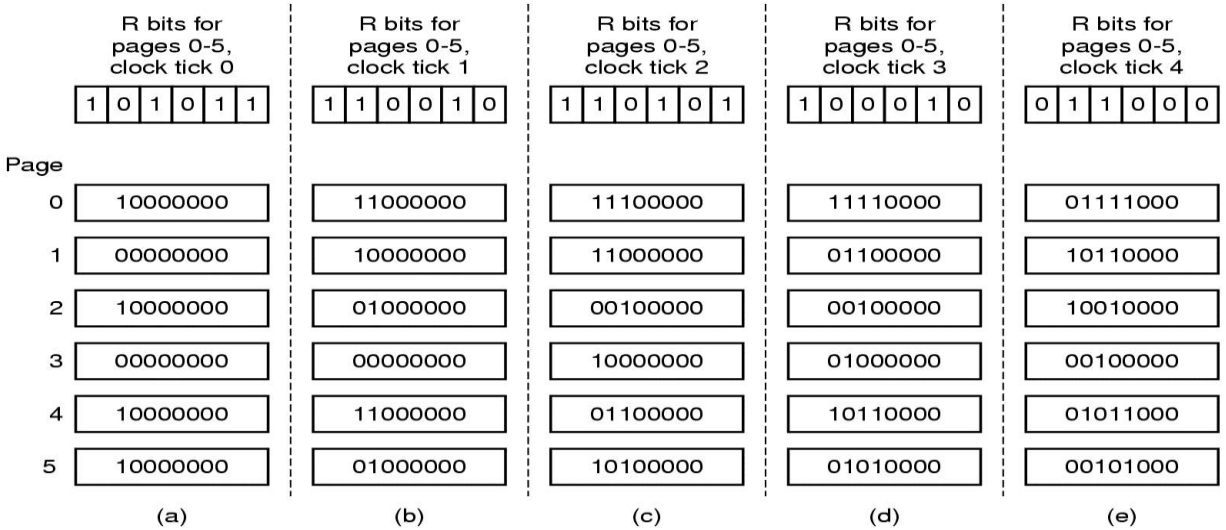
- 实现：
  - 软件计数器，一页一个，初值为0
  - 每次时钟中断时，计数器加R
  - 发生缺页中断时，选择计数器值最小的一页置换

老化算法(AGING)

思考：与LRU的区别

改进（模拟LRU）：计数器在加R前先右移一位

R位加到计数器的最左端



页面置换算法的应用

例子：

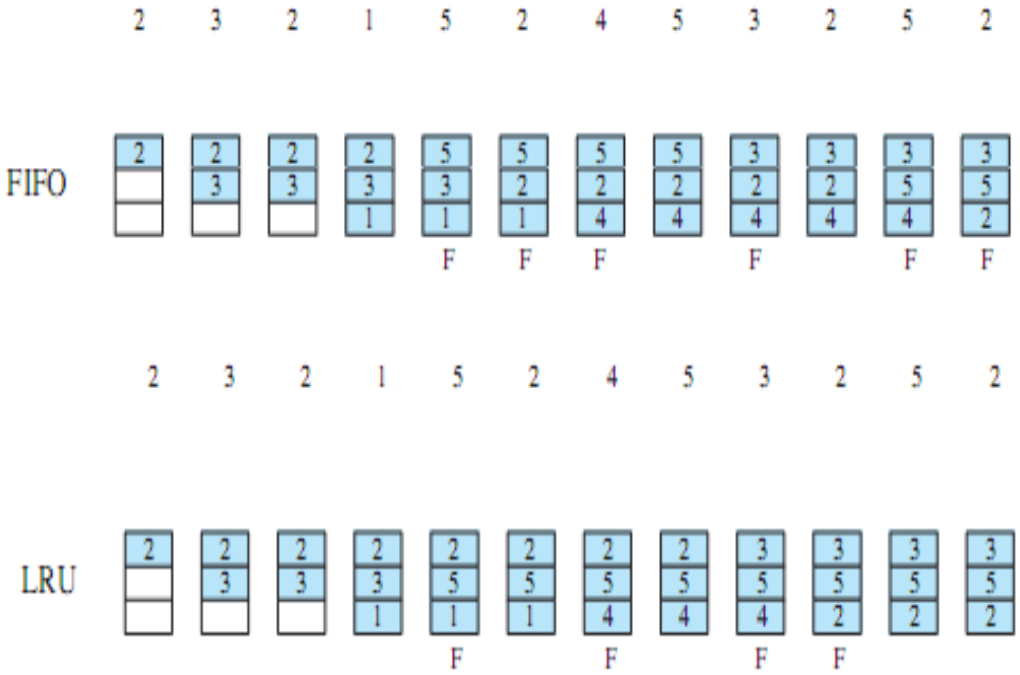
- 系统给某进程分配3个页框(固定分配策略)，初始为空
- 进程执行时，页面访问顺序为：

  2 3 2 1 5 2 4 5 3 2 5 2

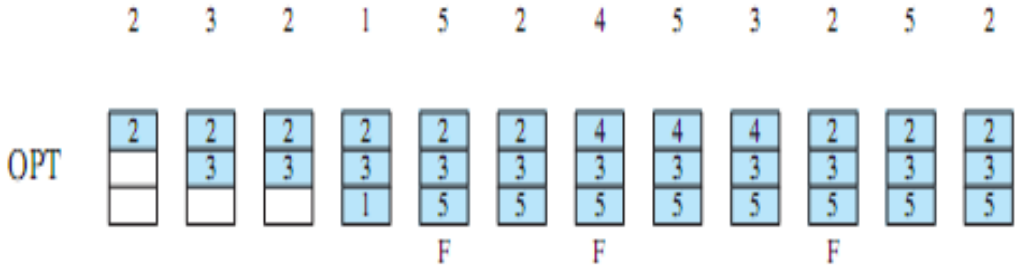
要求：

计算应用FIFO、LRU、OPT算法时的缺页次数

应用FIFO、LRU页面置换算法



应用OPT页面置换算法



BELADY现象

例子：系统给某进程分配 m个页框，初始为空

页面访问顺序为

1 2 3 4 1 2 5 1 2 3 4 5

采用FIFO算法，计算当 m=3 和 m=4 时的缺页中断次数

m=3时，缺页中断9次；m=4时，缺页中断10次

注：FIFO页面置换算法会产生异常现象（Belady现象），即：当分配给进程的物理页面数增加时，缺页次数反而增加

置换算法2——工作集算法

影响缺页次数的因素

- 页面置换算法
- 页面本身的大小 √
- 程序的编制方法 √
- 分配给进程的页框数量 √

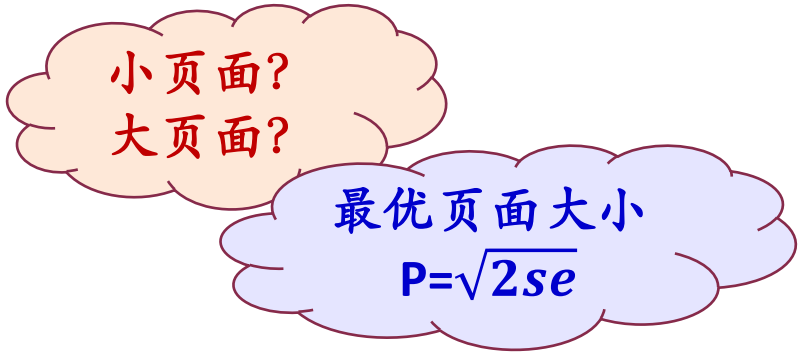
颠簸（Thrashing，抖动）

虚存中，页面在内存与磁盘之间频繁调度，使得调度页面所需的时间比进程实际运行的时间还多，这样导致系统效率急剧下降，这种现象称为颠簸或抖动

页面尺寸问题

- 确定页面大小对于分页的硬件设计非常重要
  而对于操作系统是个可选的参数

- 要考虑的因素：
  - 内部碎片
  - 页表长度
  - 辅存的物理特性



- Intel 80x86/Pentium：4096 或 4M

- 多种页面尺寸：为有效使用TLB带来灵活性，但
  给操作系统带来复杂性

程序编制方法对缺页次数的影响

例子：分配了一个页框；页面大小为128个整数；
矩阵$A_{128X128}$按行存放

程序编制方法1：

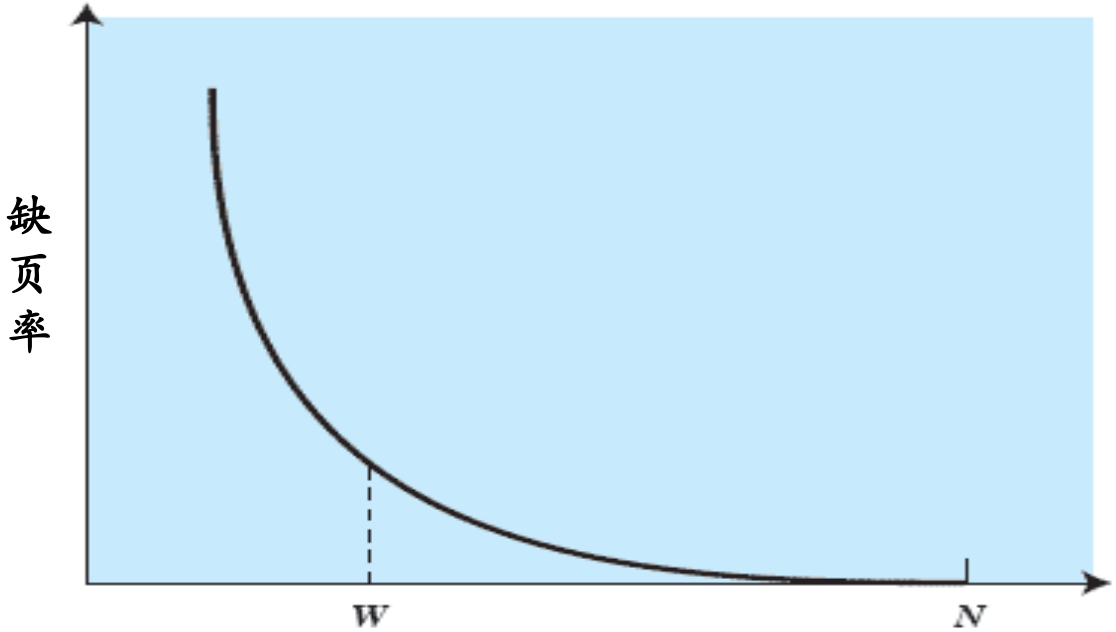
```
for j:=1 to 128
    for i:=1 to 128
            A[i,j]:=0;
```

程序编制方法2：

```
for i:=1 to 128
    for j:=1 to 128
            A[i,j]:=0;
```

缺页
几次？

分配给进程的页框数与缺页率的关系



工作集(WORKING SET)模型(1/3)

基本思想：

根据程序的局部性原理，一般情况下，进程在一段时间内总是集中访问一些页面，这些页面称为活跃页面，如果分配给一个进程的物理页面数太少了，使该进程所需的活跃页面不能全部装入内存，则进程在运行过程中将频繁发生中断

如果能为进程提供与活跃页面数相等的物理页面数，则可减少缺页中断次数

由Denning提出(1968)

工作集(WORKING SET)模型(2/3)

工作集：一个进程当前正在使用的页框集合

工作集W（t，Δ）

= 该进程在过去的Δ个虚拟时间单位中访问到的页面的集合

内容取决于三个因素：

- 访页序列特性
- 时刻t
- 工作集窗口长度(Δ)

窗口越大，工作集就越大

工作集(WORKING SET)模型(3/3)

例：

2615777516234123444343444 1327

$W(t_1,10)=\{1,2,5,6,7\}$

$W(t_2,10)=\{3,4\}$

工作集算法(1/2)

基本思路：

找出一个不在工作集中的页面并置换它

思路：

- 每个页表项中有一个字段：记录该页面最后一次被访问的时间
- 设置一个时间值T
- 判断：

根据一个页面的访问时间是否落在“当前时间-T”之前或之中决定其在工作集之外还是之内

工作集算法(2/2)

实现：

扫描所有页表项，执行操作

1. 如果一个页面的R位是1，则将该页面的最后一次访问时间设为当前时间，将R位清零
2. 如果一个页面的R位是0，则检查该页面的访问时间是否在“当前时间-T”之前

(1) 如果是，则该页面为被置换的页面；

(2) 如果不是，记录当前所有被扫描过页面的最后访问时间里面的最小值。扫描下一个页面并重复1、2

页面置换算法小结

算法	评价
OPT	不可实现，但可作为基准
NRU	LRU的很粗略的近似
FIFO	可能置换出重要的页面
Second Chance	比FIFO有很大的改善
Clock	实现的
LRU	很优秀，但很难实现
NFU	LRU的相对粗略的近似
Aging	非常近似LRU的有效算法
Working set	实现起来开销很大

内存映射技术、写时复制

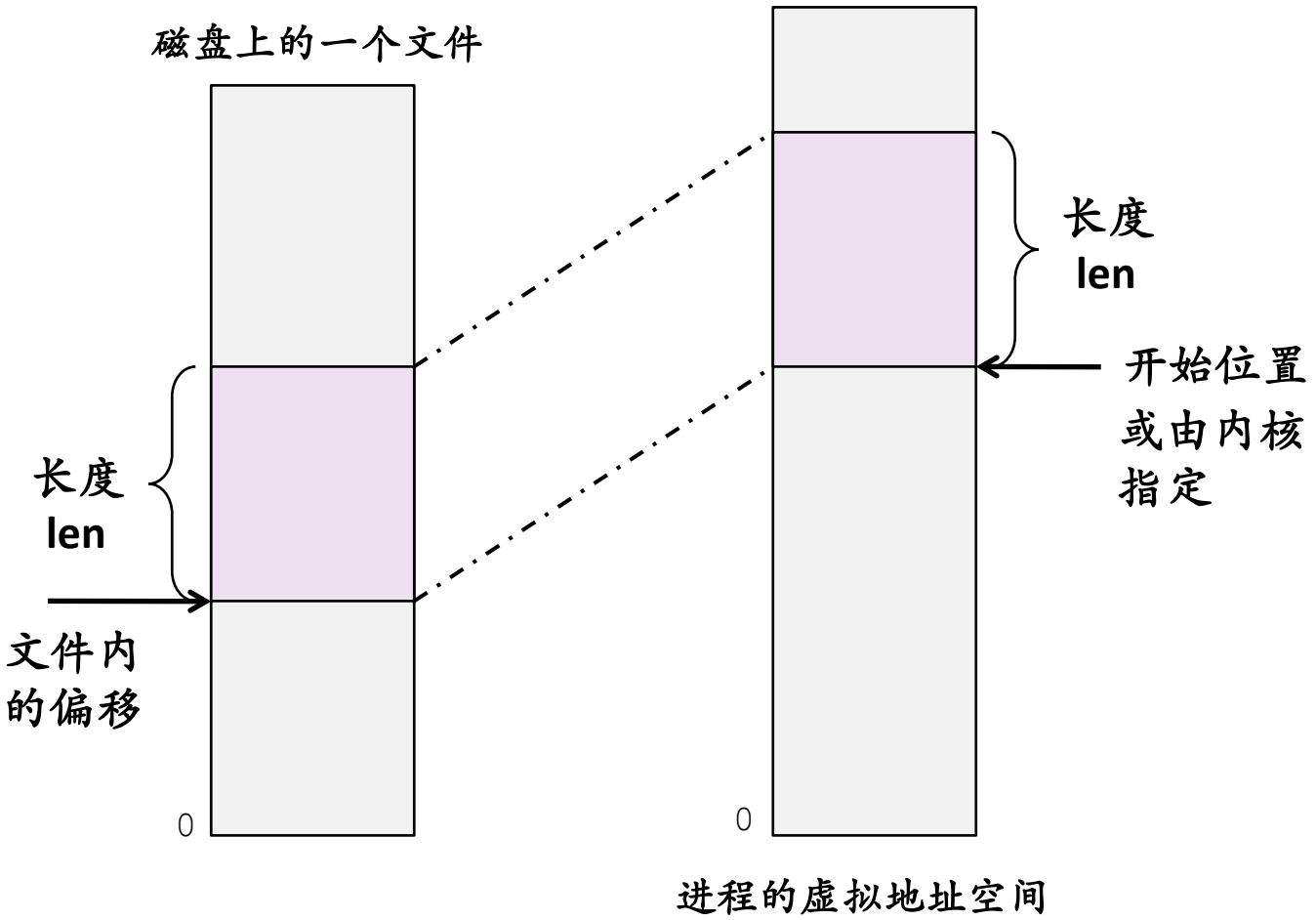
其他相关技术

内存映射文件

- 基本思想

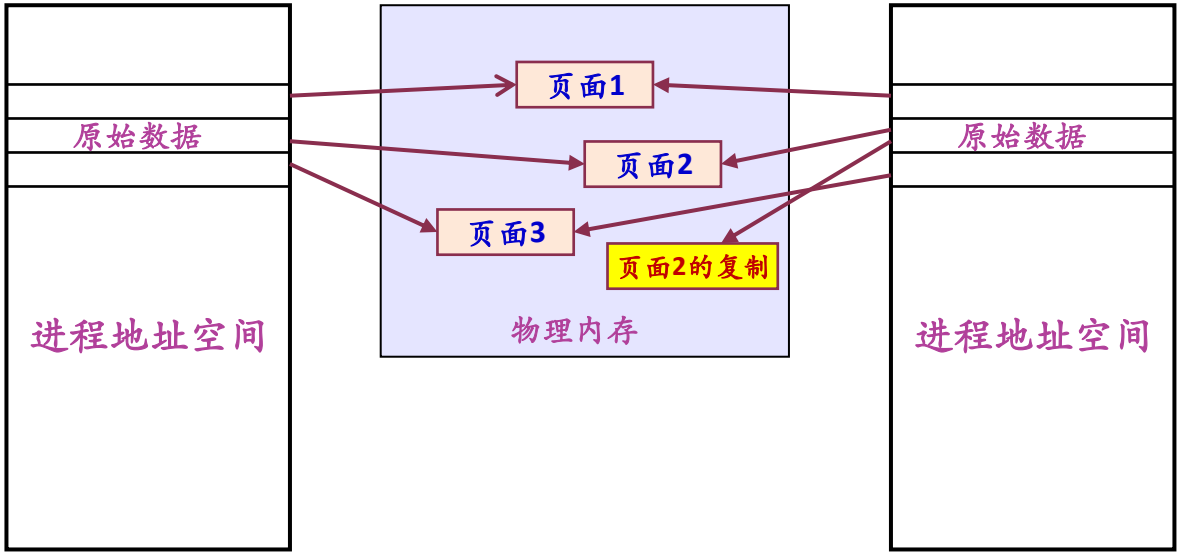
进程通过一个系统调用(mmap)将一个文件(或部分)映射到其虚拟地址空间的一部分，访问这个文件就象访问内存中的一个大数组，而不是对文件进行读写

- 在多数实现中，在映射共享的页面时不会实际读入页面的内容，而是在访问页面时，页面才会被每次一页的读入，磁盘文件则被当作后备存储
- 当进程退出或显式地解除文件映射时，所有被修改页面会写回文件


磁盘上的一个文件
长度
len
文件内
的偏移
0
长度
len
开始位置
或由内核
指定
0
进程的虚拟地址空间

支持写时复制技术

进程试图改变
页面2的数据后



例如：两个进程共享三个页，每页都标志成写时复制

新复制的页面对执行写操作的进程是私有的，对其他共享写时复制页面的进程是不可见的

本讲重点

- 掌握虚拟存储技术的相关概念
- 掌握虚拟页式存储管理方案的实现

  多级页表、反转页表、页表项、地址转换、MMU、快表(TLB)、页错误(Page Fault)、缺页异常处理
- 掌握相关软件策略

  驻留集、置换范围、清除策略

  置换算法：OPT、FIFO、第二次机会、时钟算法、LRU、老化、工作集
- 了解虚存相关的软件技术

  内存映射文件、写时复制

本周要求

- 重点阅读教材

  第3章相关内容：3.3、3.4、3.5.1、3.5.7、3.5.8、3.6.1、3.6.2

- 重点概念

虚拟存储技术：虚拟内存（虚存） 虚拟地址空间 虚拟地址
存储体系 存储保护 内存管理单元MMU
快表TLB 页错误 缺页异常处理
页表 页表项 多级页表 反转页表
驻留集 置换范围 清除策略 页缓冲技术
页面置换算法：最佳、先进先出、第二次机会、时钟、最近最少使用、老化、工作集模型
内存映射文件 写时复制技术

THANKS

The End