操作系统A

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

2020 Autumn

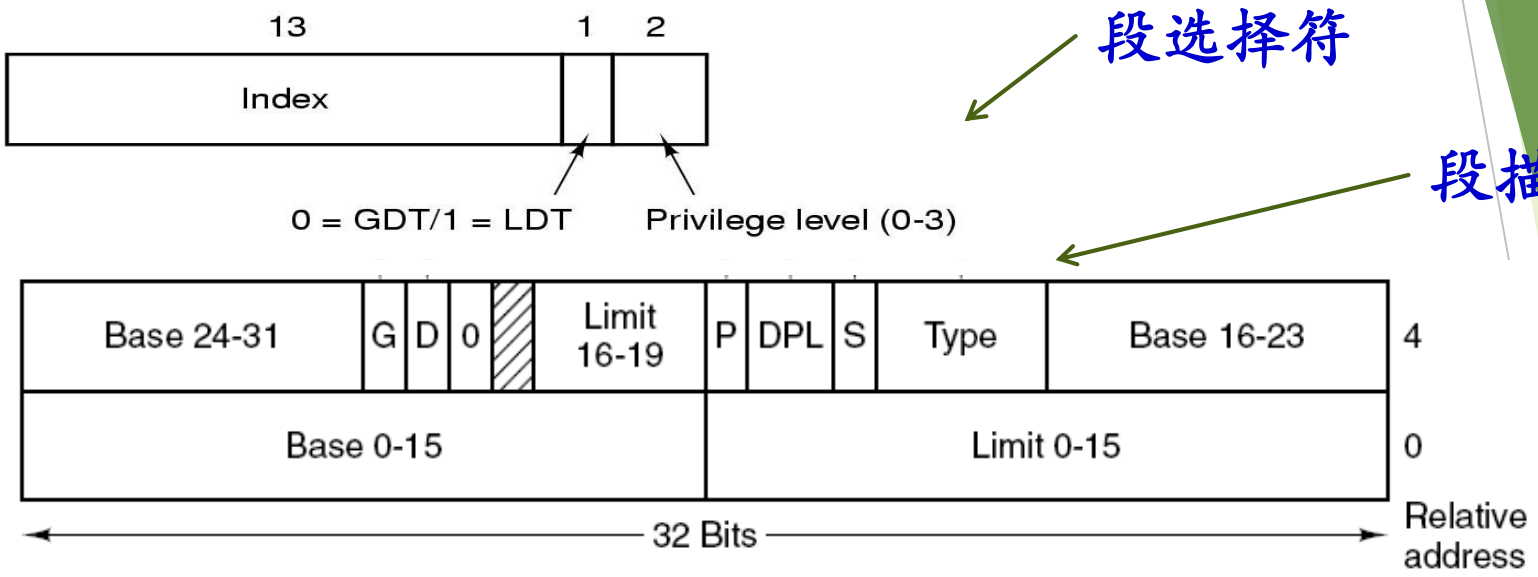
存储管理

- Intel x86 虚拟内存机制
  - 段页式、段描述符、地址转换
- Windows虚拟内存管理
  - 内存管理器组、地址空间布局、地址转换机制、缺页处理
  - 内存分配方式
  - 工作集
  - 物理内存管理

段式寻址、页式及地址转换……

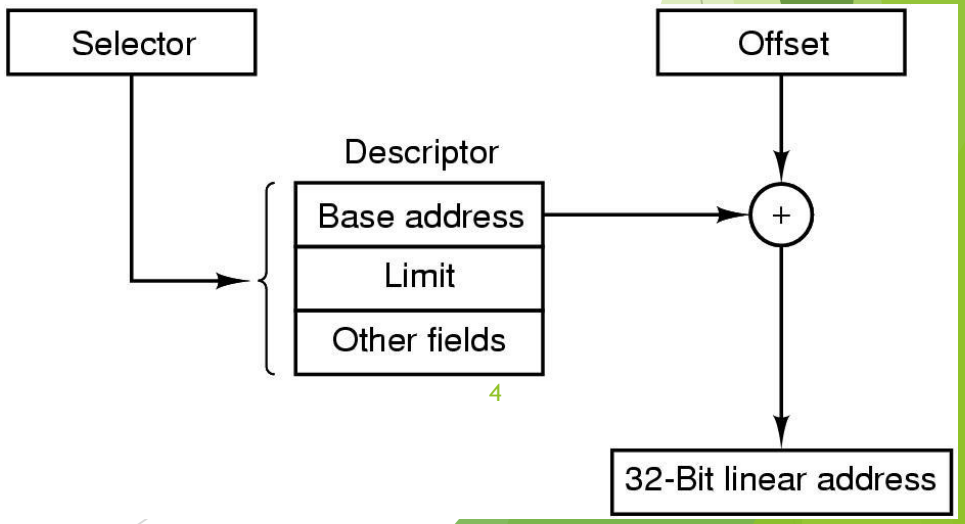
Intel x86
虚拟内存机制

保护模式下的寻址方式



思考：

- ✓ x86提供几种工作模式?
- ✓ 保护模式下的寻址方式?
- ✓ 地址保护如何完成?
- ✓ TLB是怎么刷新的?
- ✓ 如何绕过分段机制?



逻辑地址到物理地址的转换

分页机制开启

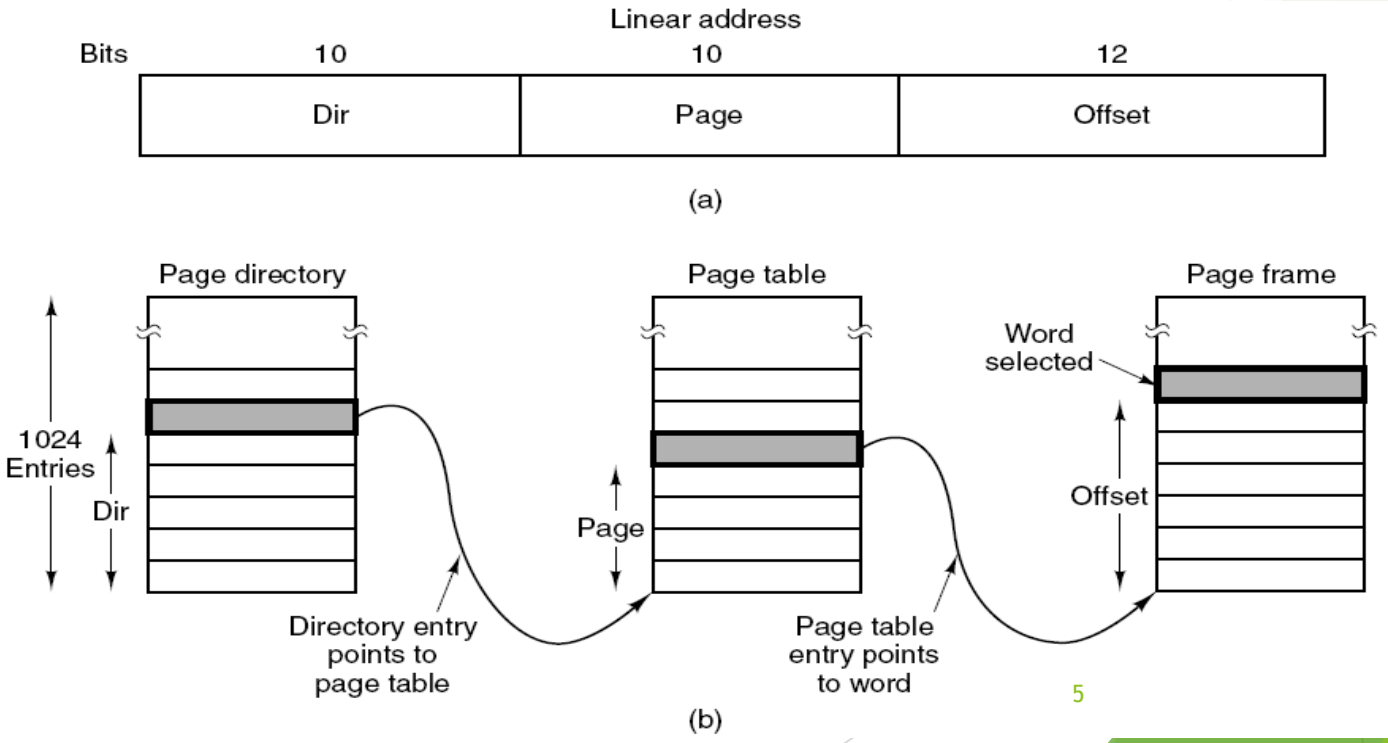
逻辑地址 → 段式转换 → 线性地址 → 页式转换 → 物理地址

控制寄存

CR0

CR2

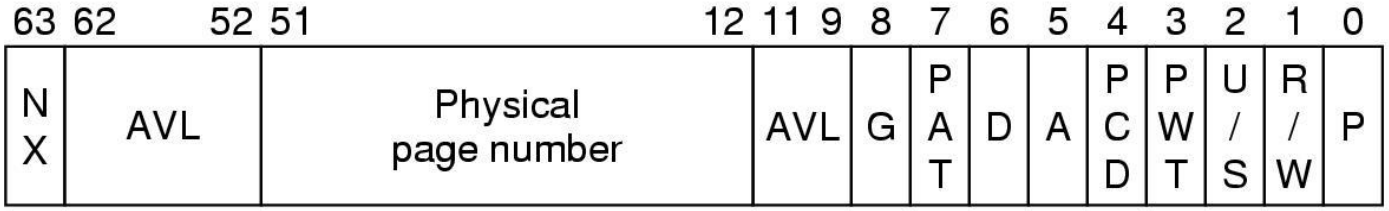
CR3



x86页目录项和页表项

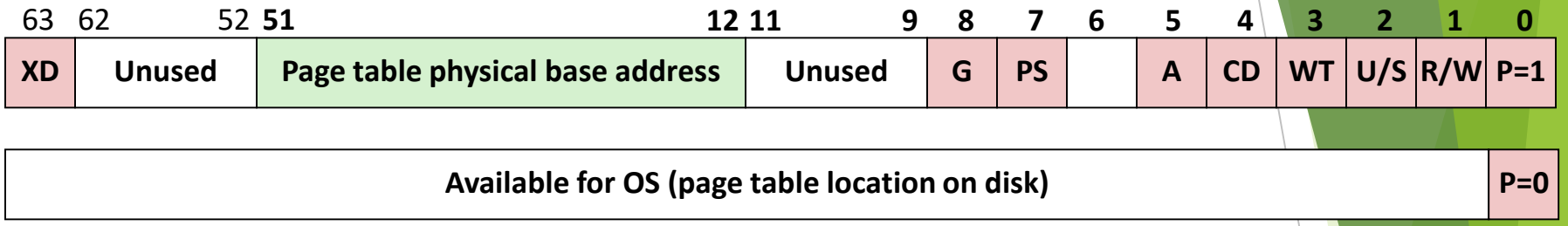
页目录项 PDE (Page Directory Entry)

页表项 PTE (Page Table Entry)



AMD x64体系结构中页表项

Core i7 Level 1-3 Page Table Entries



Each entry references a 4K child page table

P: Child page table present in physical memory (1) or not (0).

R/W: Read-only or read-write access access permission for all reachable pages.

U/S: user or supervisor (kernel) mode access permission for all reachable pages.

WT: Write-through or write-back cache policy for the child page table.

CD: Caching disabled or enabled for the child page table.

A: Reference bit (set by MMU on reads and writes, cleared by software).

PS: Page size either 4 KB or 4 MB (defined for Level 1 PTEs only).

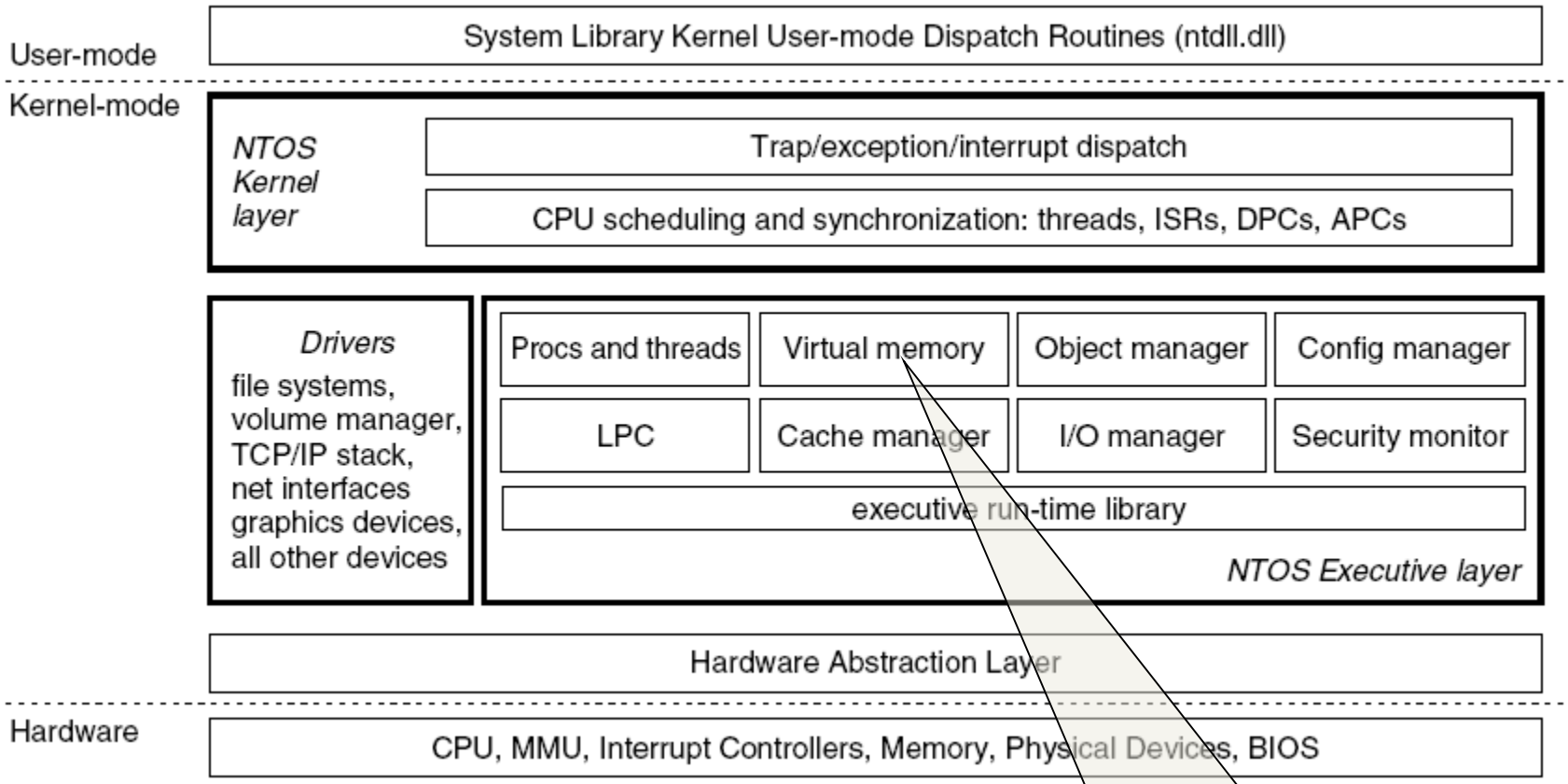
G: Global page (don't evict from TLB on task switch)

Page table physical base address: 40 most significant bits of physical page table address (forces page tables to be 4KB aligned)

基本概念、主要方式

Windows虚拟内存管理

Windows的体系结构



内存管理器
位于Ntoskrnl.exe文件中

内存管理器的组成部分

- 一组执行体系统服务程序

  用于虚存的分配、回收和管理。大多数服务是通过Win32 API或内核态的设备驱动程序接口的形式呈现

- 一个页面错误陷阱处理程序

  用于解决硬件检测到的内存管理异常

  负责分配页框，或把磁盘上的数据装入内存

内存管理器的组成部分

关键组件

- *工作集管理器(MmWorkingSetManager)*：当空闲内存低于某一界限时，便启动所有的内存管理策略，如：工作集的修整、老化和已修改页面的写入等(16)
- *进程/栈交换器(KeSwapProcessOrStack)*：(23)

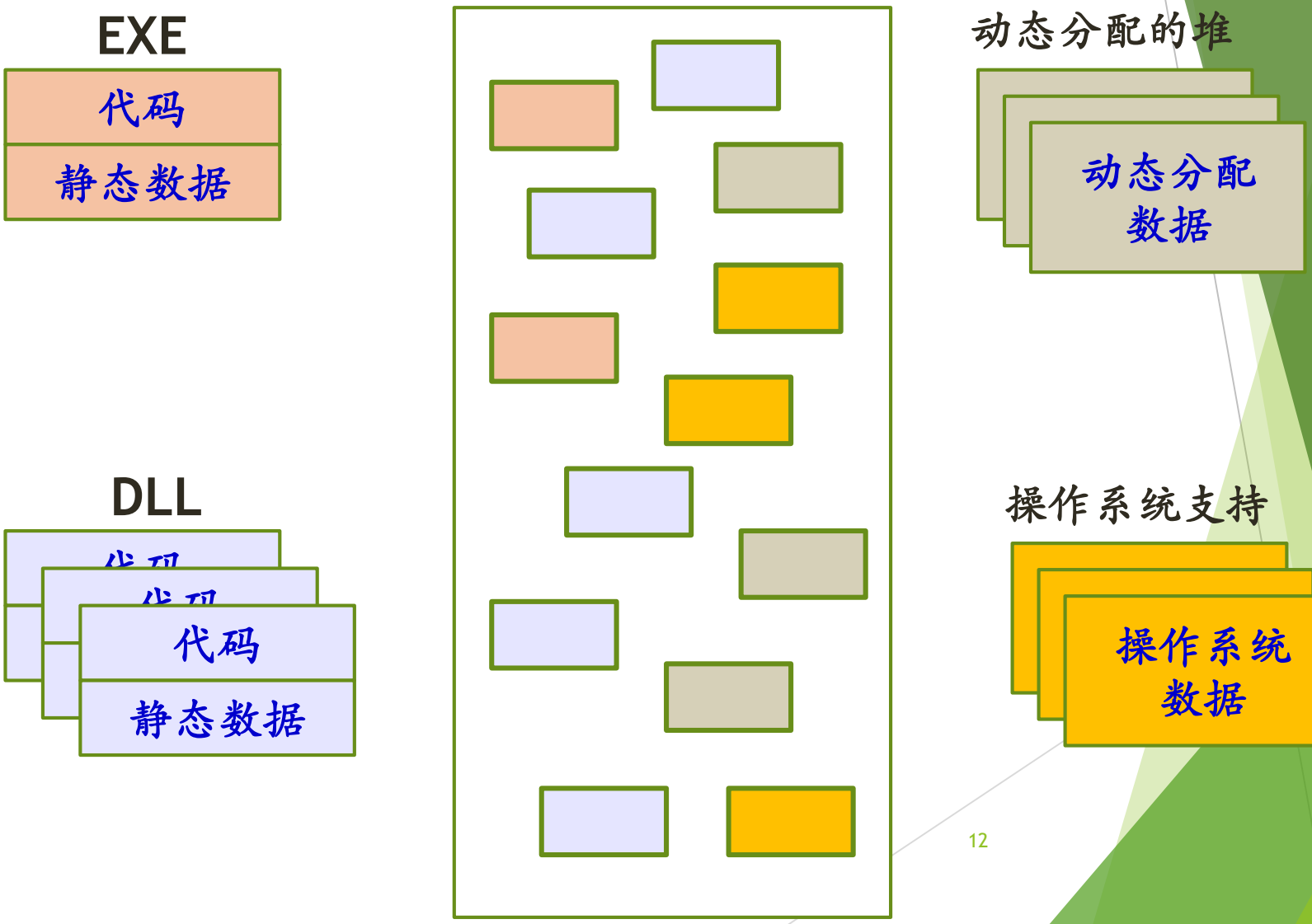
  完成进程和内核线程堆栈的换入和换出操作
- *修改页面写出器(MiModifiedPageWriter)*：(17)

  将“脏”页写回到适当的页文件
- *映射页面写出器(MiMappedPageWriter)*：(17)

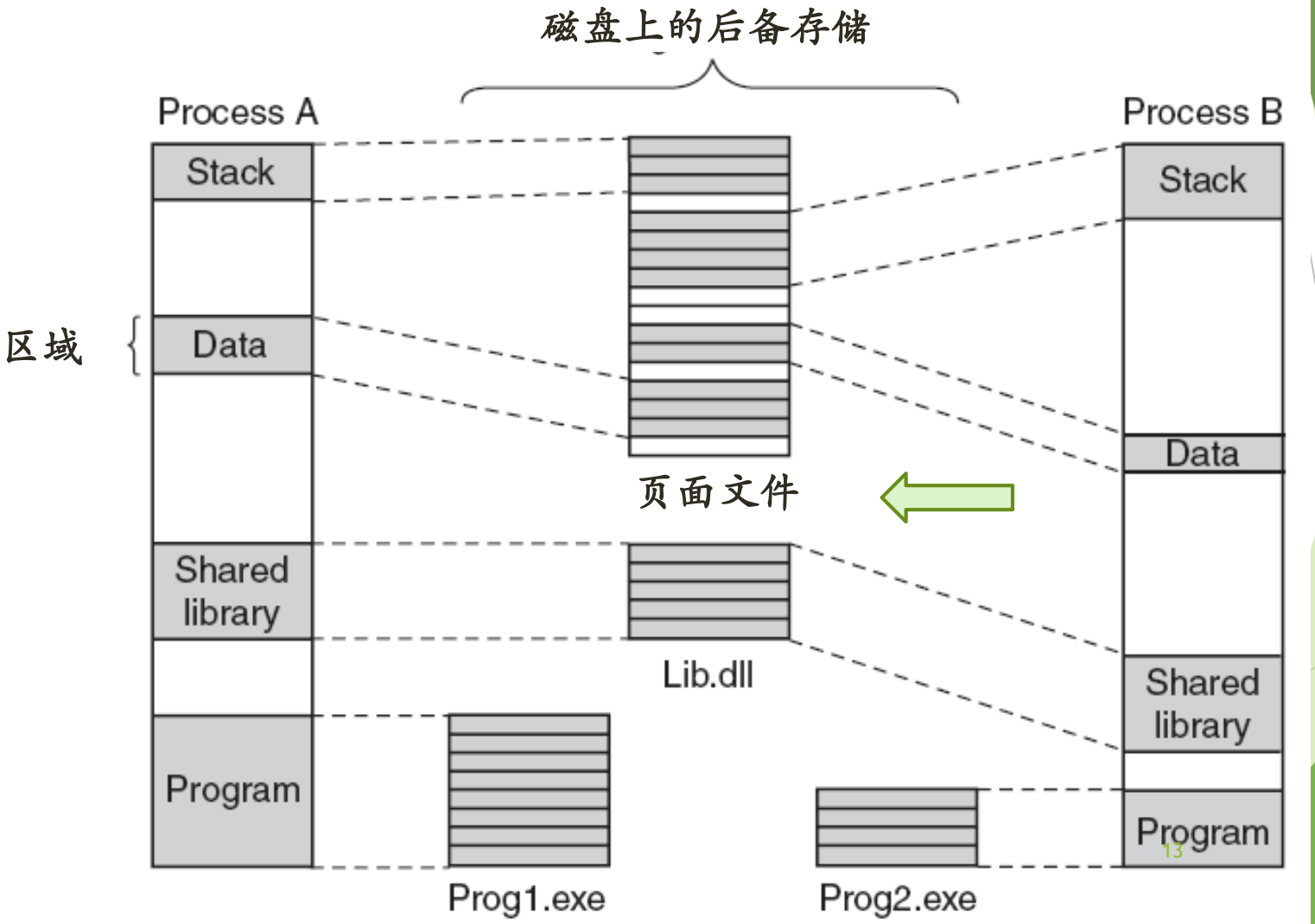
  将映射文件中的“脏”页写回磁盘
- *零页线程(MmZeroPageThread)*：(0)

  将空闲链表中的页框清零

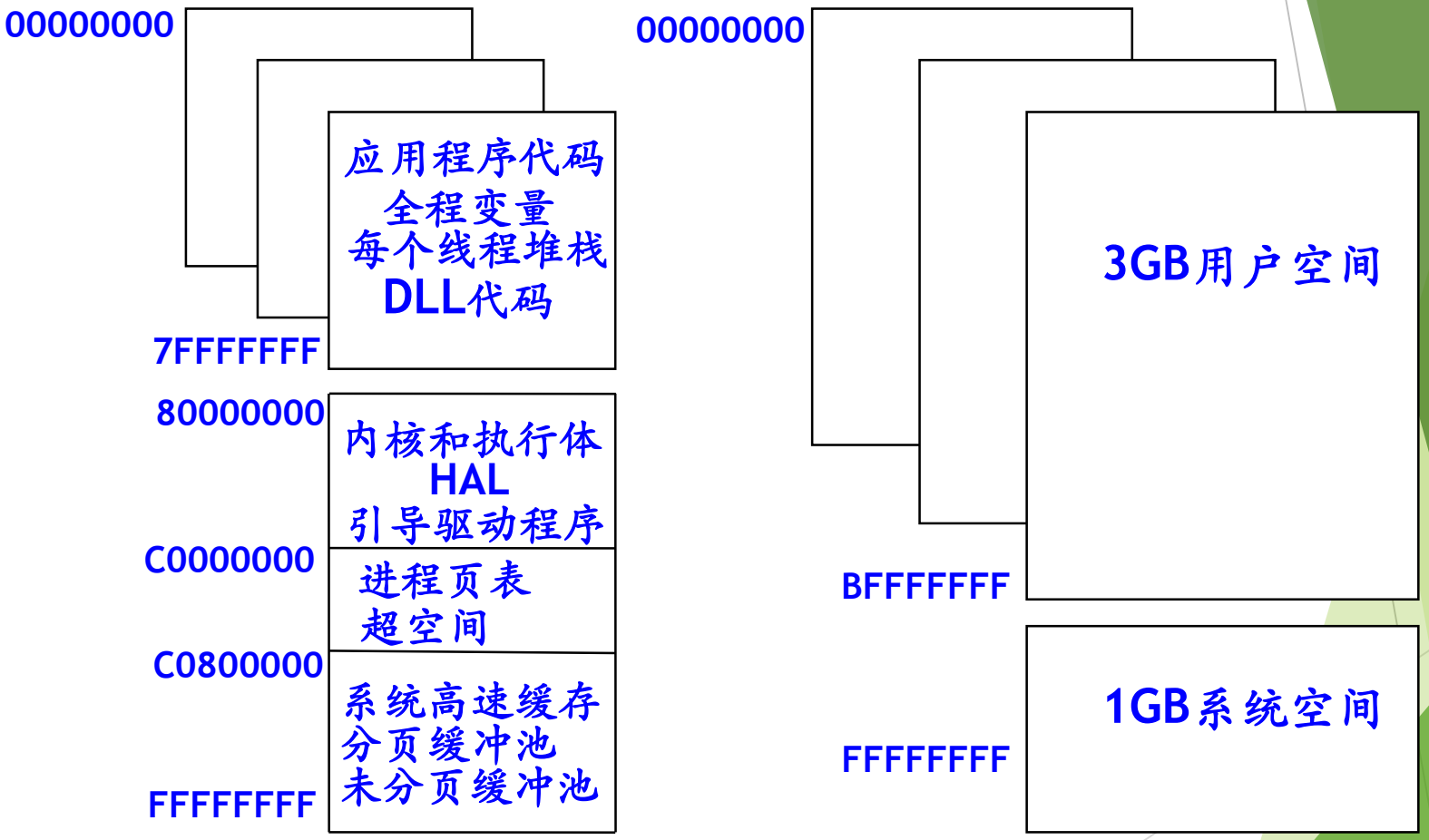
Windows的虚拟内存



磁盘上的相关内容

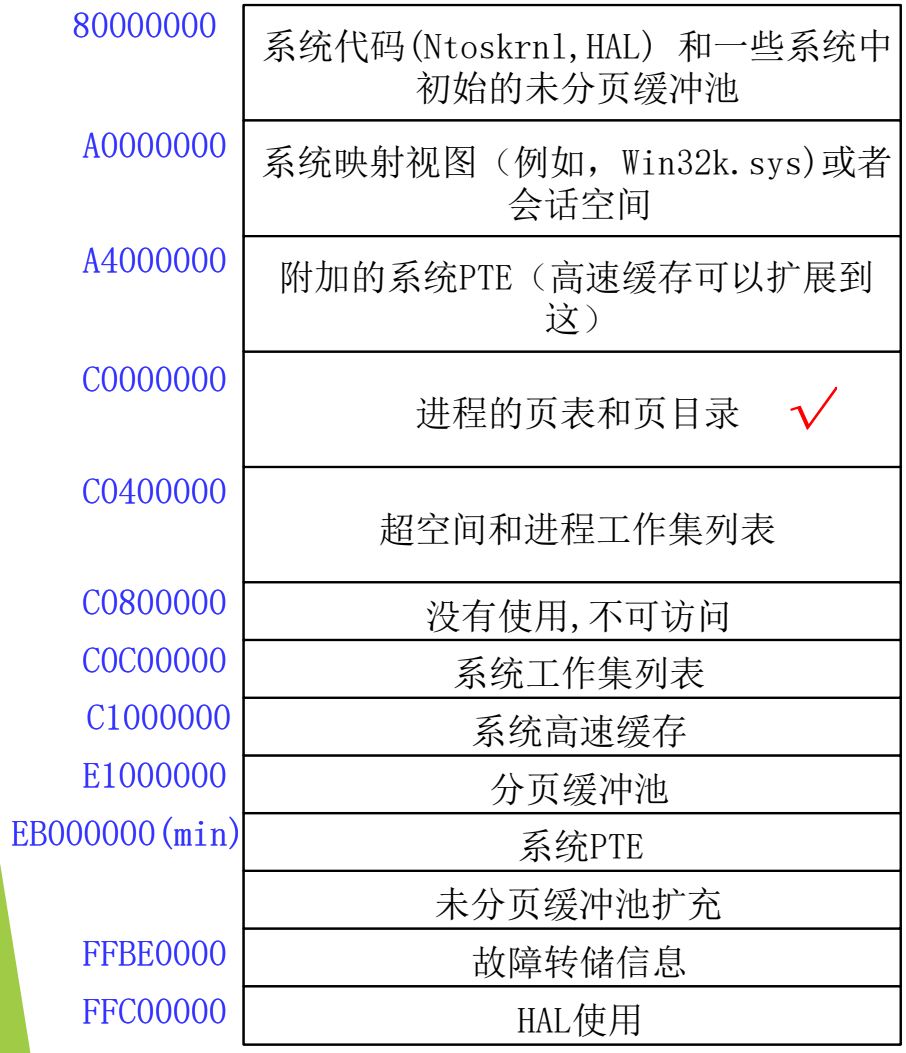

磁盘上的后备存储
Process A
Process B
Stack
Data
Shared library
Program
区域
页面文件
Lib.dll
Prog1.exe
Prog2.exe
Stack
Data
Shared library
Program

地址空间布局(1/3)



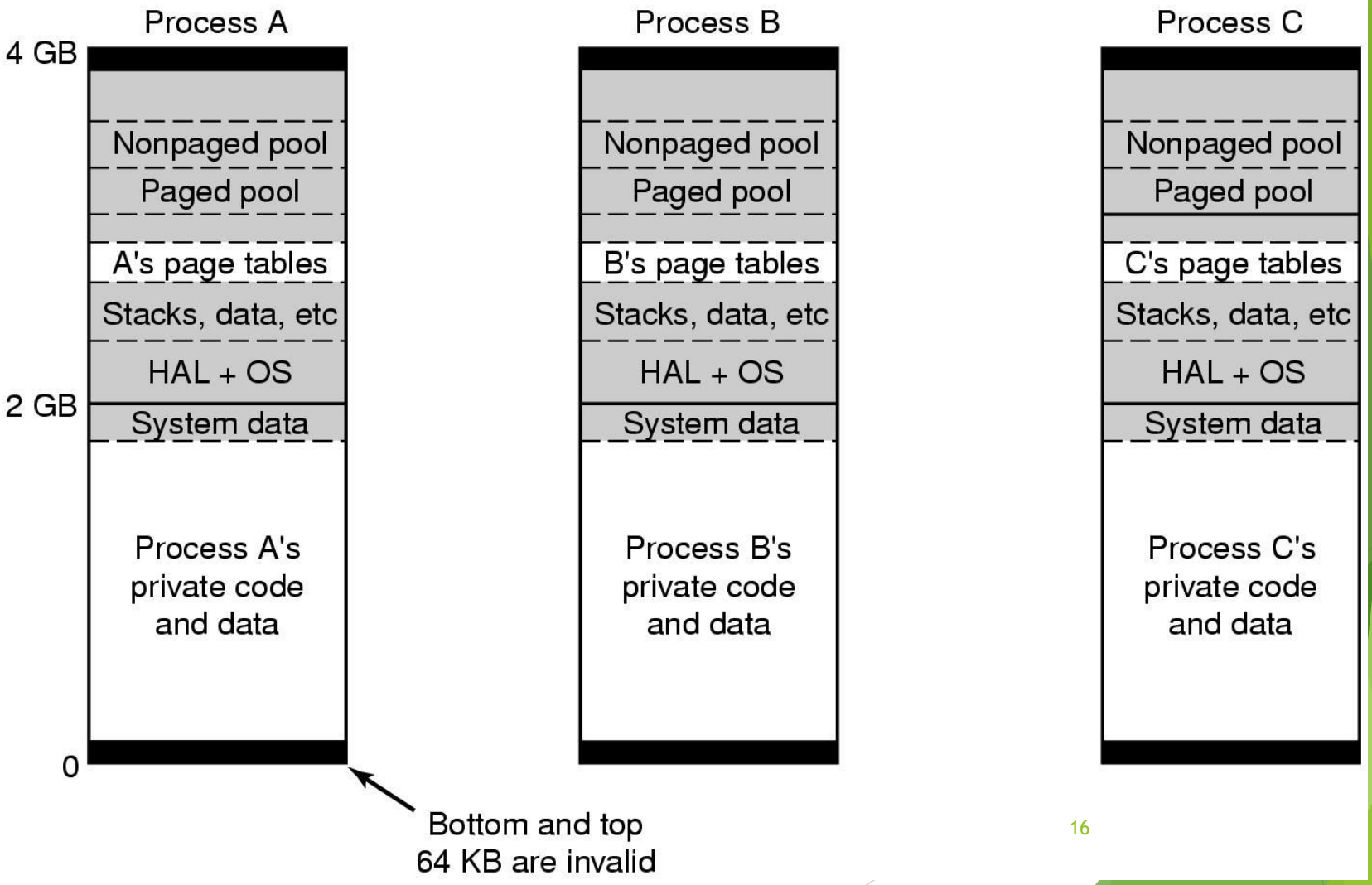
在boot.ini中加入／3GB标志

地址空间布局(2/3)

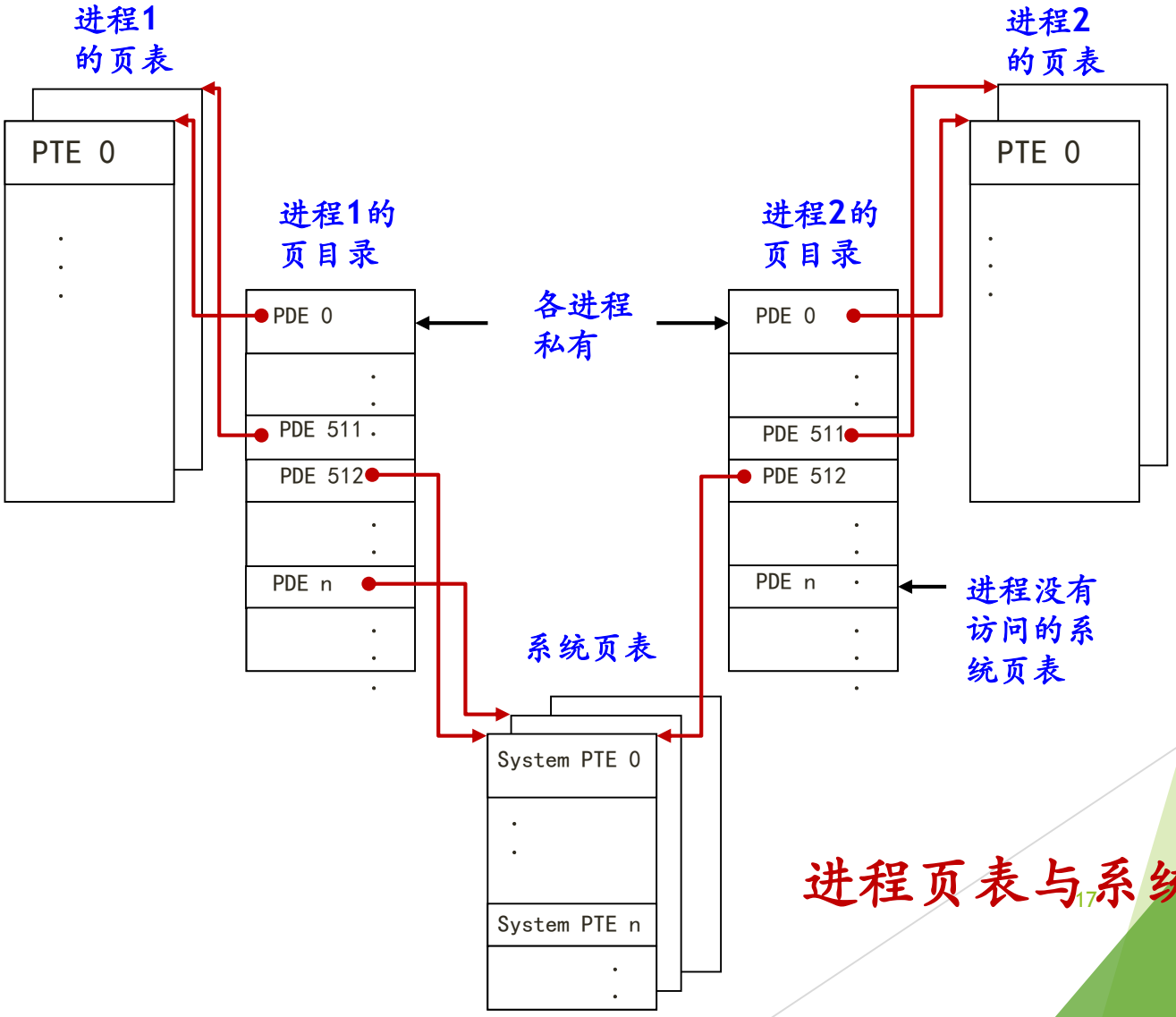

80000000
系统代码(Ntoskrnl,HAL) 和一些系统中初始的未分页缓冲池
A0000000
系统映射视图（例如，Win32k.sys)或者会话空间
A4000000
附加的系统PTE（高速缓存可以扩展到这）
C0000000
进程的页表和页目录
C0400000
超空间和进程工作集列表
C0800000
没有使用,不可访问
C0C00000
系统工作集列表
C1000000
系统高速缓存
E1000000
分页缓冲池
EB000000(min)
系统PTE
未分页缓冲池扩充
FFBE0000
故障转储信息
FFC00000
HAL使用


00000000
不可访问
00010000
环境变量
进程的参数
进程堆(Heap)
进程载入的模组
线程栈(Stack)
线程TEB
7FFDE000
第一个线程的线程环境块(TEB)
7FFDF000
进程环境块(PEB)
7FFE0000
共享用户数据页
7FFF0000
不可访问

地址空间布局(3/3)

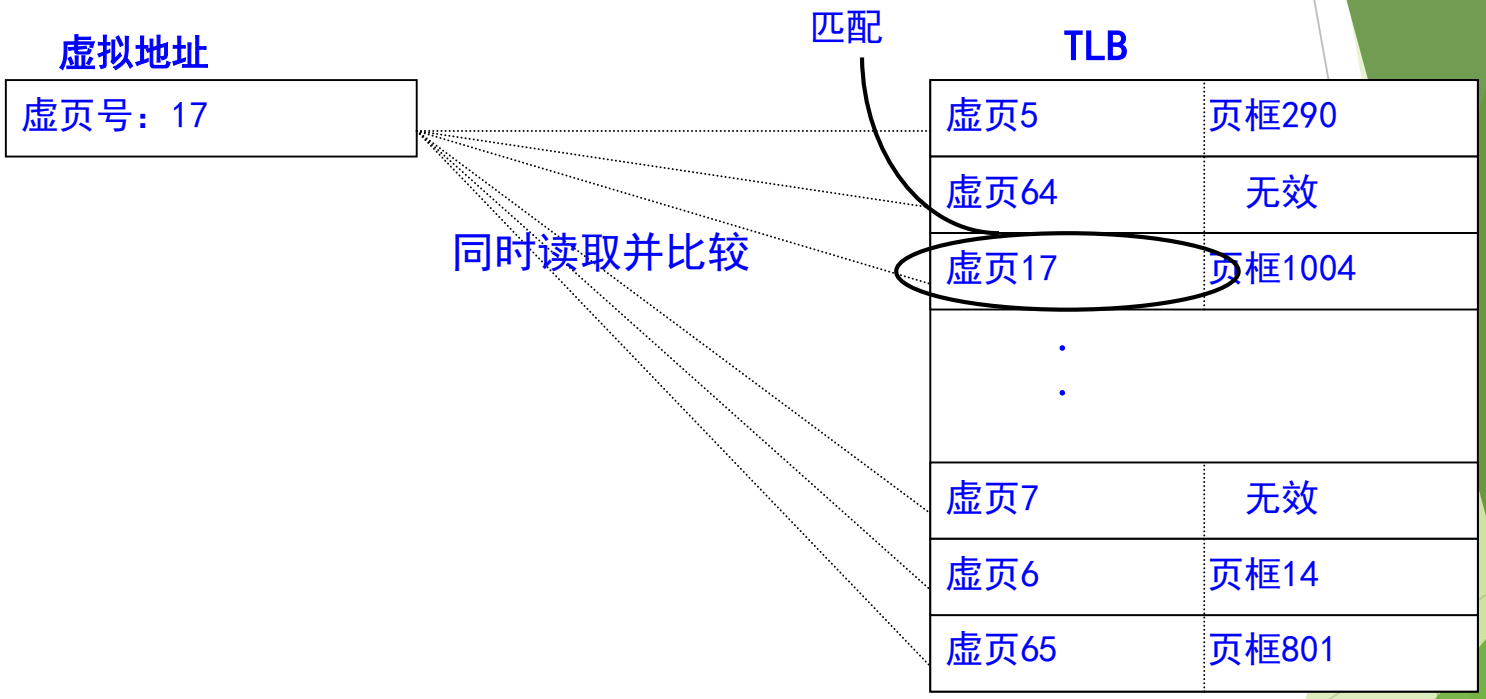

Process A
Process B
Process C
4 GB
2 GB
0
Nonpaged pool
Paged pool
A's page tables
B's page tables
C's page tables
Stacks, data, etc
HAL + OS
System data
Process A's private code and data
Process B's private code and data
Process C's private code and data
Bottom and top 64 KB are invalid

地址转换机制



进程页表与系统页表

快表TLB


虚拟地址
虚页号：17
匹配
TLB
同时读取并比较
虚页5
页框290
虚页64
无效
虚页17
页框1004
虚页7
无效
虚页6
页框14
虚页65
页框801

Windows虚拟内存管理

缺页异常处理

缺页异常

- 页目录项PDE和页表项PTE的最低位为1，有效(Valid)，表示该页映射了物理内存
- 当要访问的虚拟页在物理内存中时，该虚拟页对应的 PDE、PTE 都有效，CPU 根据相应的PDE、PTE自动把虚拟地址转换成物理地址，完成访问——这一过程操作系统不需介入
- 如果要访问的虚拟页在不物理内存，此时的PTE无效（最低位为 0 ）
- 对无效PTE的格式的定义，由操作系统负责

页面失效——无效的PTE

- 所引用的页面没有被提交
- 违反权限的页面访问
- 修改一个私有共享的写时复制页面
- 需要扩大栈
- 所引用的页已被提交但尚未被映射
- 请求一个零页面

Windows缺页异常处理

- 当某条指令访问的页不在物理内存中时，CPU 仍然会自动通过页目录和页表把虚拟地址转换成物理地址，在地址转换过程中，CPU将会发现对应地址的页表项无效，从而就会引发页面错误（Page Fault）异常，该异常的中断号是0xe，有时也称为缺页中断
- 发生缺页中断时，CPU 自动将引发异常时访问的虚拟地址存入寄存器CR2
- 在发生异常时，CPU 自动根据中断号，在中断描述符表中找到相应的中断描述符，根据中断描述符中的地址，转到异常处理程序ntoskrnl!KiTrap0E
- 异常处理程序 KiTrap0E是操作系统提供的，它将会调用MmAccessFault，MmAccessFault 通过 CR2 中的访问地址，计算出相应的 PDE／PTE地址，通过分析PTE中的内容（无效的 PDE／PTE的格式由操作系统定义），可以知道是哪种情况引起的异常，并根据情况作出相应的处理

Windows虚拟内存管理

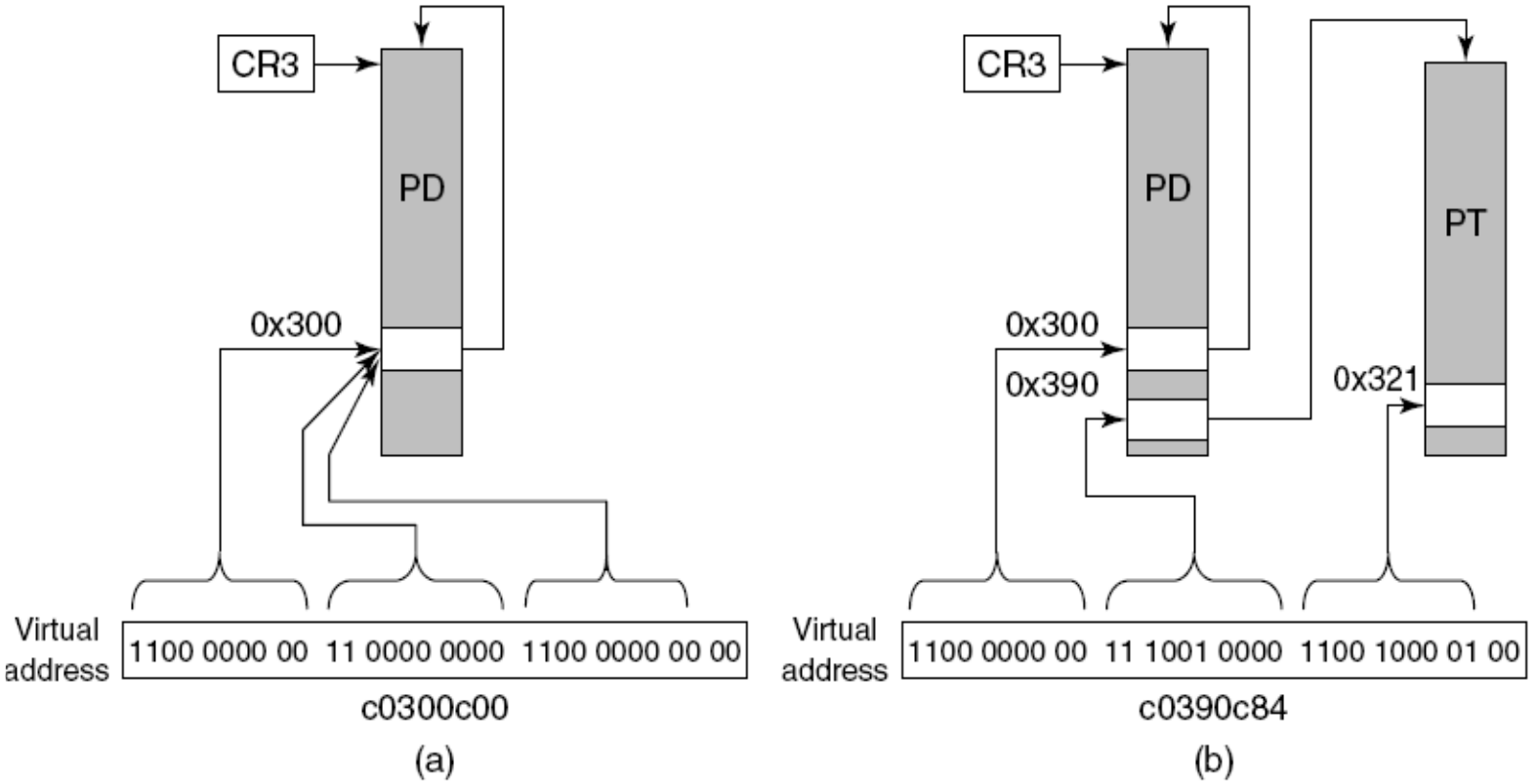
页目录和自映射机制

页目录

- 内存管理器创建的特殊页，用于映射进程所有页表的位置
- 其物理地址保存在KPROCESS中
- 硬件访问页目录、页表和页：通过PFN完成
- 内核：通过虚地址来对它们进行访问

  x86中它还同时被映射到地址0xC0300000处
- 专用寄存器（x86中为CR3）用于保存页目录的物理地址

页目录自映射机制



自映射：PD[0xc0300000>>22]即PD

虚拟地址(a)：(PTE *)(0xc0300c00)指向PD[0x300]，即自映射页目录项

虚拟地址(b)：(PTE *)(0xc0390c84)指向PTE，对应虚拟地址0xe4321000

页目录自映射机制(续)

宏MiGetPdeAddress():

给定一个虚拟地址va，计算出对应的PDE

```
((PMMPTE)(((((ULONG)(va))>>22)<< 2) + PDE_BASE))
```

宏MiGetPteAddress():

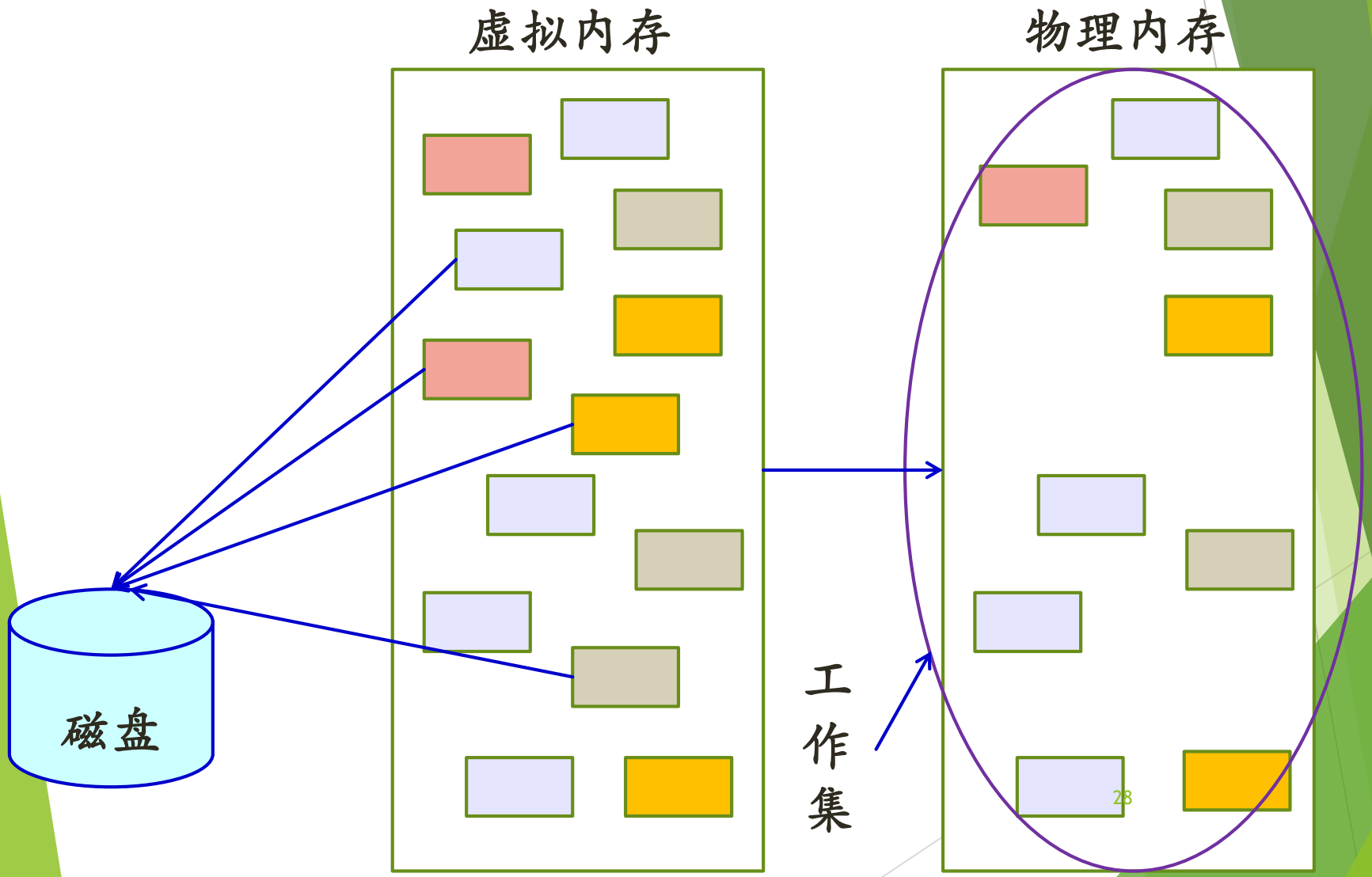
给定一个虚拟地址va，计算出对应的PTE

```
((PMMPTE)(((((ULONG)(va))>>12)<< 2) + PTE_BASE))
```

Windows虚拟内存管理

工作集

Windows的工作集(1/3)



Windows的工作集(2/3)

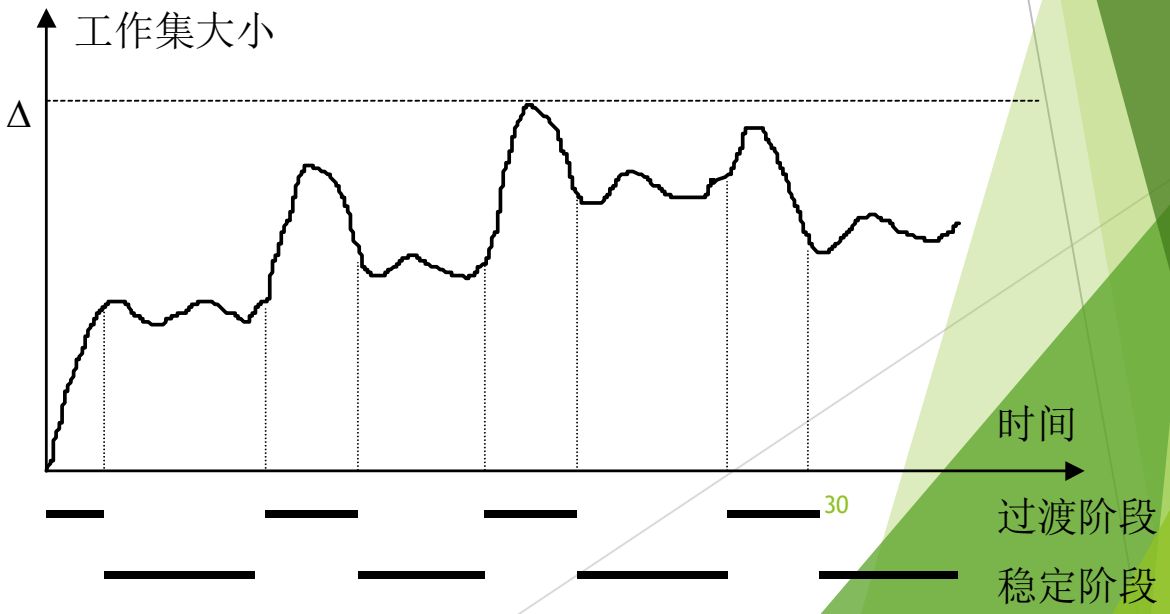
- 工作集

  驻留在物理内存中的虚拟页面的子集

  - ✓ 进程工作集：为每个进程分配的一定数量的页框
  - ✓ 系统工作集：为可分页的系统代码和数据分配的页框
  - 内存管理器利用请求式页面调度算法以及簇方式将页面装入内存
  - 置页策略：选择页框应使CPU内存高速缓存不必要的震荡最小

Windows的工作集(3/3)

- 工作集大小的变化：进程开始执行后，随着访问新页面逐步建立较稳定的工作集
- 当内存访问的局部性区域的位置大致稳定时，工作集大小也大致稳定
- 局部性区域的位置改变时，工作集快速扩张和收缩过渡到下一个稳定值



页面置换算法

动态改变大小和构成

- 基于工作集模型
  - 由已装入内存的页框组成
  - 最小值/最大值
  - 当可用页框数量降低到一定程度时，启动工作集修整策略

平衡集管理器线程调用工作集管理器

- 周期性检查

  大量可用内存、内存开始紧张、内存紧缺

Windows虚拟内存管理

用户空间内存分配方式

用户空间内存分配方式

- 以页为单位的虚拟内存分配方式

  函数（Virtualxxx）

- 内存映射文件

  函数（CreateFileMapping，MapViewOfFile）

- 内存堆方法

  （Heapxxx 和早期的接口Localxxx 和Globalxxx）

以页为单位的虚拟内存分配(1/3)

- 进程地址空间0x0~0x7FFFFFFF，用户程序必须经过“保留”和“提交”两个阶段使用一段地址范围
- VirtualAlloc和VirtualAllocEx函数实现这些功能，用户程序可以首先保留地址空间，然后向此地址空间提交物理页面
- 保留地址空间是为线程将来使用所保留的一块虚拟地址。在已保留的区域中，提交页面必须指出将物理存储器提交到何处以及提交多少；提交页面在访问时会转变为物理内存中的有效页面
- VirtualFree或VirtualFreeEx函数回收页面或释放地址空间

以页为单位的虚拟内存分配(2/3)

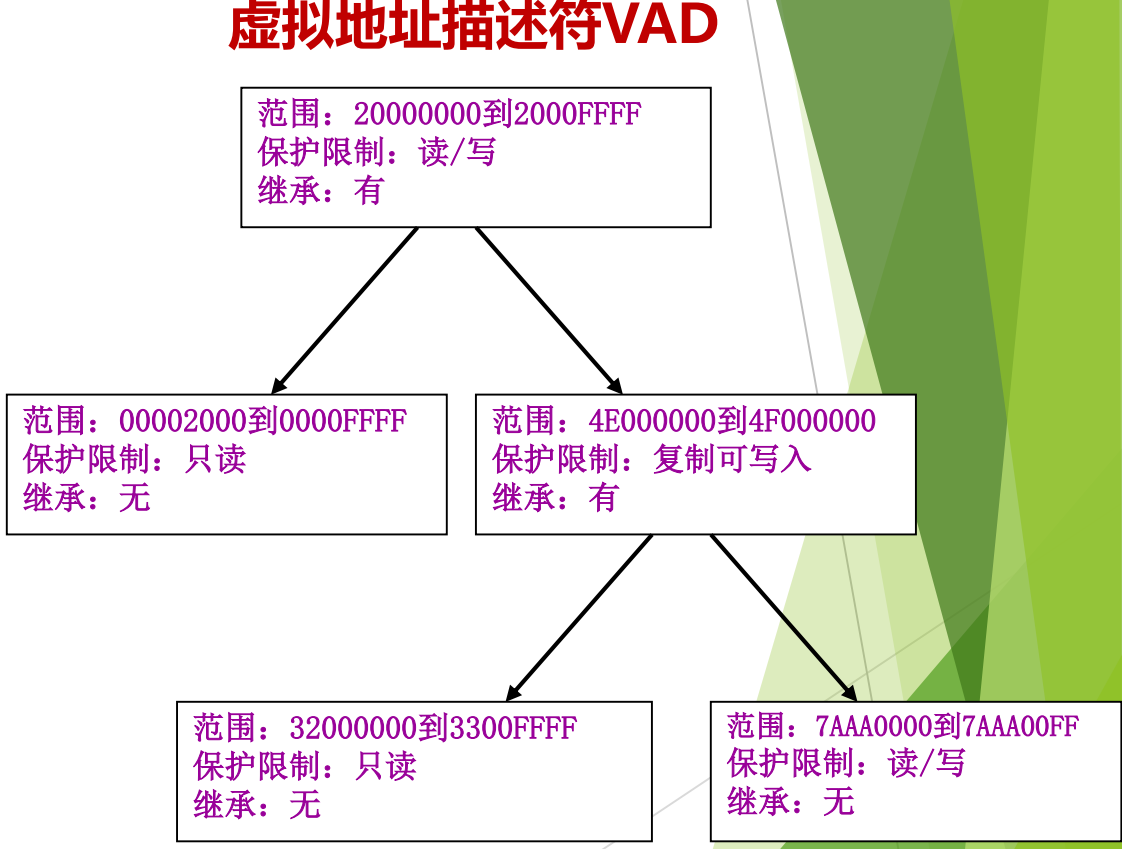
问题?

- 使用 VirtualAlloc 可以在用户地址空间中保留或者提交指定地址和大小的一段地址空间，**那么系统如何知道指定的这段地址空间是不是已经被分配（保留或者提交）？**
- 对于指定地址空间是否已经被提交了物理内存，可以通过**页目录和页表**来判断，不过这样做很麻烦；而对于指定地址空间是否已经被保留，通过页目录和页表是没有办法判断的
- Windows中使用 VAD 来解决这个问题

以页为单位的虚拟内存分配(3/3)

对于每一个进程，Windows 的内存管理器维护一组虚拟地址描述符（VAD）来描述一段被分配的进程虚拟空间的状态

虚拟地址描述信息被构造成一棵自平衡二叉树以使查找更有效率



VAD二叉排序树的根的地址保存在进程结构 EPROCESS 中

利用区域对象实现内存映射文件

通过区域对象服务提供的大数据流和内存共享服务

- 区域对象（section object）在Win32子系统中称为文件映射对象
- 表示可以被两个或更多进程所共享的内存块，区域对象也可以基于页文件
  - CreateFileMapping创建区域对象
  - MapViewOfFile函数映射区域对象的一部分（叫做区域视图），并指定映射范围
- 当两个进程对同一区域对象建立视图时，就发生了对该区域对象的共享

利用区域对象实现内存映射文件

- 使用区域对象将一个文件映射到进程的地址空间，之后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
- 当程序访问一个无效的页面（不在物理内存中的页面）时，引起缺页中断，内存管理器会自动将这个页面从映射文件调入内存
- 如果应用程序修改了页面，内存管理器在页面进行常规调度时将修改写回文件（或者应用程序可以通过利用Win32的FlushViewOfFile函数来刷新一个视图）

内存堆方法

- 内存堆方法适合于大量的小型内存申请
- 堆(heap)是保留的地址空间中一个或多个页组成的区域，这个地址区域可以由堆管理器按更小块划分和分配
- 堆管理器是执行体中分配和回收可变内存的函数集
- 进程启动时带有一个缺省进程堆，通常是1MB大小。为了从缺省堆中分配内存，线程必须调用GetProcessHeap函数得到一个指向它的句柄。有了句柄后，线程可以调用HeapAlloc和HeapFree来从堆中分配和回收内存块
- 进程也可以使用HeapCreate函数创建另外的私有堆。当进程不再需要私有堆时，可以通过调用HeapDestroy释放虚拟地址空间

Windows虚拟内存管理

物理内存管理

物理内存管理

- 页框号数据库（PFN数据库）
- 结构MMPFN（24字节）

  保存每一个物理页的相关信息
- 全局变量 MmPfnDatabase

  保存页框号数据库的首地址

页框的状态(1/2)

- 活动（Active）/有效（Valid）

  该页框在某个进程的工作集中

  此进程的对应页表项是有效的，从高20bit可获得PFN

- 过渡（Transition）

  系统正在从一个文件将内容读入该页框，或者正在向一个文件写出该页框的内容

- 空闲（Free）

  该页框中的内容不再被需要

- 零初始化（zeroed）

  该页框空闲并且已经被用零初始化

- 坏（Bad）

  该页框存在硬件错误，不能被使用

页框的状态(2/2)

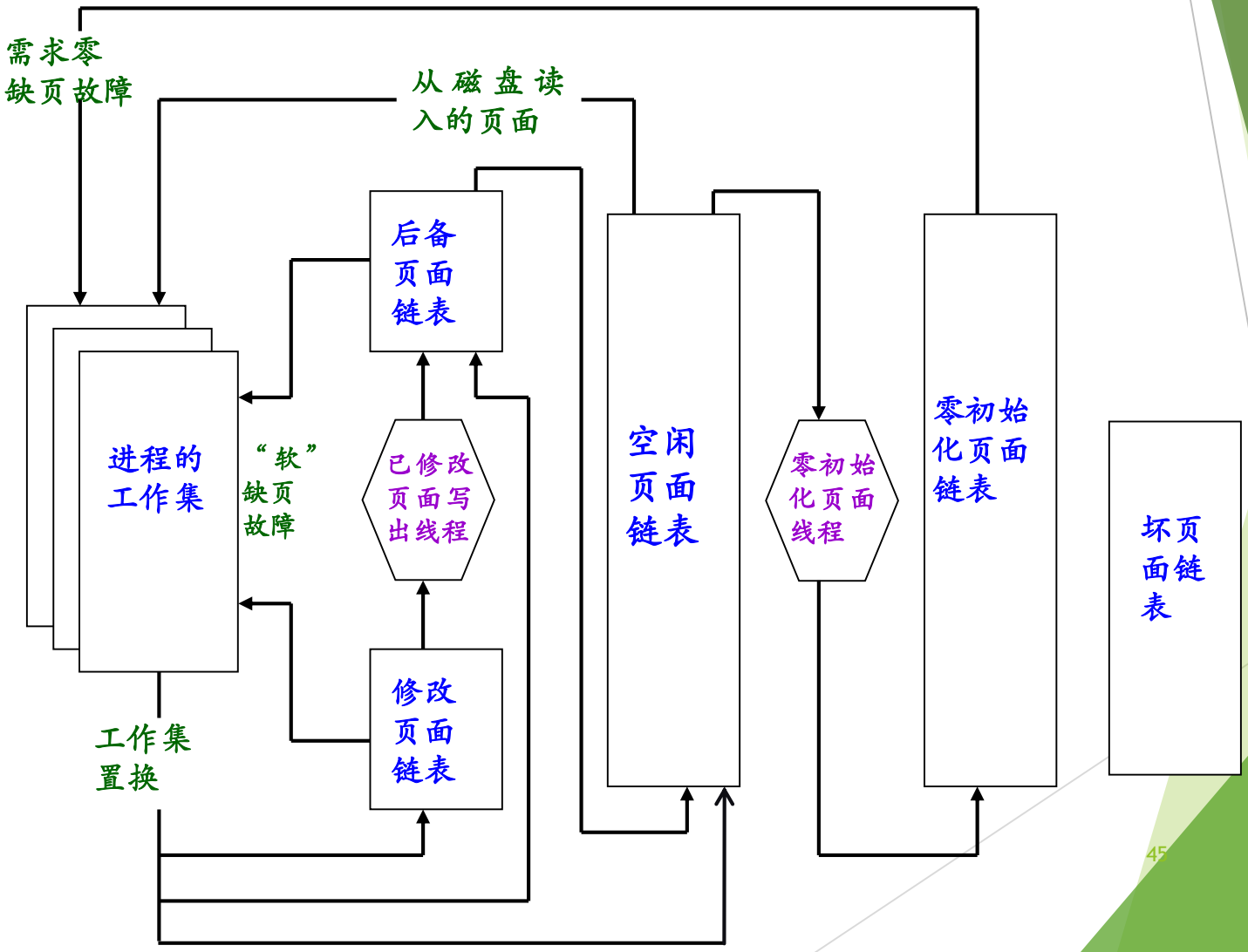
- **后备(standby)**
  - 该页框曾经在某个进程的工作集中，且该页框的内容在被此进程使用时没有改变
  - 该页框现在已被移出该进程的工作集，不过页框中的内容仍是此进程的内容；即对应的PTE中的高20bit仍然是该页框号，只是该PTE被标记为 invalid 和 transition
  - 当此进程需要再次访问这一页内容时，只需要重新设定该PTE的标志，并把该PTE变为有效，即把该页框从 standby 状态变为active(valid) 状态即可

- **修改（Modified）**
  - 该页框曾经在某个进程的工作集中，且该页框的内容在被此进程使用时有改变
  - 该页框现在已被移出该进程的 工作集，不过页框中的内容仍是在此进程的内容；对应的PTE中的高20bit仍然是该页框号，只是该PTE被标记为 invalid 和 transition
  - 当此进程需要再次访问这一页内容时，只需要重新设定该PTE的标志，并把该PTE变为有效，即把该页框从 modified 状态变为 active(valid) 状态就可以了
  - 在该页框被系统作为其他用途使用之前，需要将该页框中的内容写入硬盘中的页文件中

页表与页框号数据库

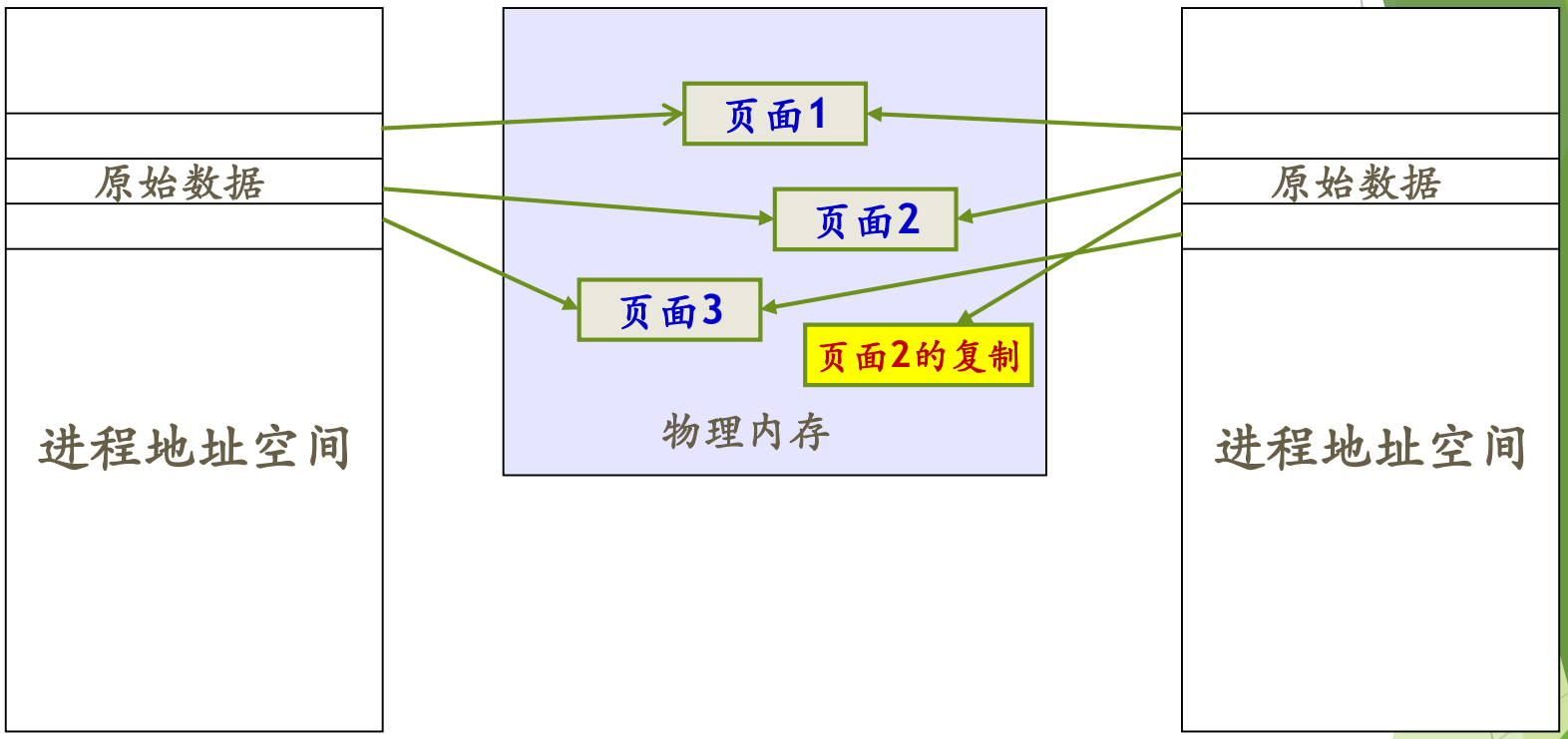

页框号数据库
页表
State
Cnt
WS
Other
PT
Next
链表头
Standby
Modified
Free
Zeroed
14 Clean X
13 Dirty X
12 Clean
11 Active 20
10 Clean
9 Dirty
8 Active 4
7 Dirty
6 Free X
5 Free
4 Zeroed X
3 Active 6
2 Zeroed
1 Active 14
0 Zeroed

页框数据库实现与应用


需求零
缺页故障
从磁盘读
入的页面
后备
页面
链表
进程的
工作集
“软”
缺页
故障
已修改
页面写
出线程
空闲
页面
链表
零初始
化页面
线程
零初始
化页面
链表
坏页
面链
表
修改
页面
链表
工作集
置换

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，每页个都标志成写时复制

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

重点小结

- Intel Pentium虚存机制
  - 保护模式下寻址方式
  - 地址转换：段式转换与页式转换
- Windows虚拟内存管理
  - 地址空间布局
  - 虚拟页面状态
  - 区域对象与内存映射文件
  - 物理内存管理
  - 工作集模型
  - 写时复制技术支持

思考题

- 总结Windows自映射机制的实现原理。
- 总结Windows的物理内存管理。

Thanks

The End