Windows 2000 – Memory Management

„Ausgewählte Betriebssysteme“
Institut Betriebssysteme

Fakultät Informatik

Outline

- Components, Services
- „Kinds of Memory“
- Address Space Layout
- Address Translation
- Physical Address Extension & Address Windowing Extensions
- Memory Objects and their usage

Components

- Memory Manager exists fully in Ntoskrnl.exe (no parts in HAL)
- Components:
  - Set of executive system services for allocating, de-allocating and managing virtual memory
  - Trap handler for hardware-detected memory management exceptions
  - Several key components (running in the context of different kernel-mode system threads)

Configuring

- Registry value in HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management
  - ClearPageFileAtShutdown
  - DisablePagingExecutive
  - IoPageLockLimit
  - ...

MM Services

- MM provides services to:
  - Allocate and free virtual memory
  - Share memory between processes
  - Map files into memory
  - Flush virtual pages to disk
  - Lock virtual pages in memory
  - ...
- A process handle can be supplied with call → specifies process, whose memory is being manipulated

MM Services (2)

- Most services exposed through Win32 API (*Virtualxxx, CreateFileMapping, MapViewOfFile, Heapxxx*)
- Also services to:
  - Allocate and de-allocate physical memory and
  - Lock pages in physical memory (e.g. DMA)
- Also support function from executive to:
  - Allocate and de-allocate memory from system heaps
  - Manipulate look-aside lists

Reserved and Commited Pages

- Reserved pages:
  - Are reserved for the process
  - Access results in access violation because memory is not mapped to storage
- Committed pages:
  - Are backed by physical memory
  - Are either private or mapped to a view of a section
- Pages are written to disk during normal modified page writing
- This approach can reduce memory usage
- Reserving memory is relatively fast
- Reserving only updates VAD structures (see below)

Locking Memory

- Device Drivers use kernel-mode functions
  - Remain in memory until explicitly unlocked
  - Can‘t lock more pages than resident available page count allows
  - Also, each page uses PTE, which is a limited resource
- Win32 applications (user-mode) lock pages in process working set
  - These pages can be paged (all threads in wait state)
  - If thread restarts all locked pages have to be loaded
  - Number of locked pages < working set size - 8

Allocation Granularity

- Aligns region of reserved address space to multiple of system allocation granularity (currently 64K)
- Size of reserved region is multiple of system page size (on x86 4K)

Protecting Memory

- System controlled translation of virtual to physical addresses
- Hardware-controlled memory protection
- Section objects have ACLs, which are checked when process tries to open it
- Thread needs appropriate rights to create section (e.g. permissions on file)

Copy on Write

- Same as Linux
- Used to optimize memory access
- Used by POSIX subsystem to implement `fork()`

Outline

- Components, Services
- „Kinds of Memory“
- Address Space Layout
- Address Translation
- Physical Address Extension & Address Windowing Extensions
- Memory Objects and their usage

System Memory Pools

- Two dynamically sized pools for kernel-mode components
- Non-paged pool:
  - Guaranteed to reside in physical memory
  - One for general use
  - One for emergency use (4 pages) – should not be used
  - Maximum: 256 MB
- Paged pool:
  - Used below DPC IRQL
  - On uniprocessor 3 pools, on multiprocessors 5
  - Multiple pools reduce blocking frequency
  - Maximum: 491‘875 MB

Heap Functions

- Managed by heap manager, which is a set of functions to allocate and de-allocate variable amounts of memory
- One default heap per process, which is synchronized by default
- New heaps can be created (MSDN: HeapCreate)
- Thread needs handle to heap to allocate memory from it
- Allows process to allocate memory, which is smaller than a page

Look-Aside Lists

- Pool that contains fixed size blocks
- Faster than general pools:
  - Don‘t use spin locks
  - No need to search for fitting space
- Executive components can create lists, that match size of data-structures
- Memory added from paged/non-paged pool
- Memory freed automatically depending on allocation frequency
- MSDN: ExInitializePagedLookasideList

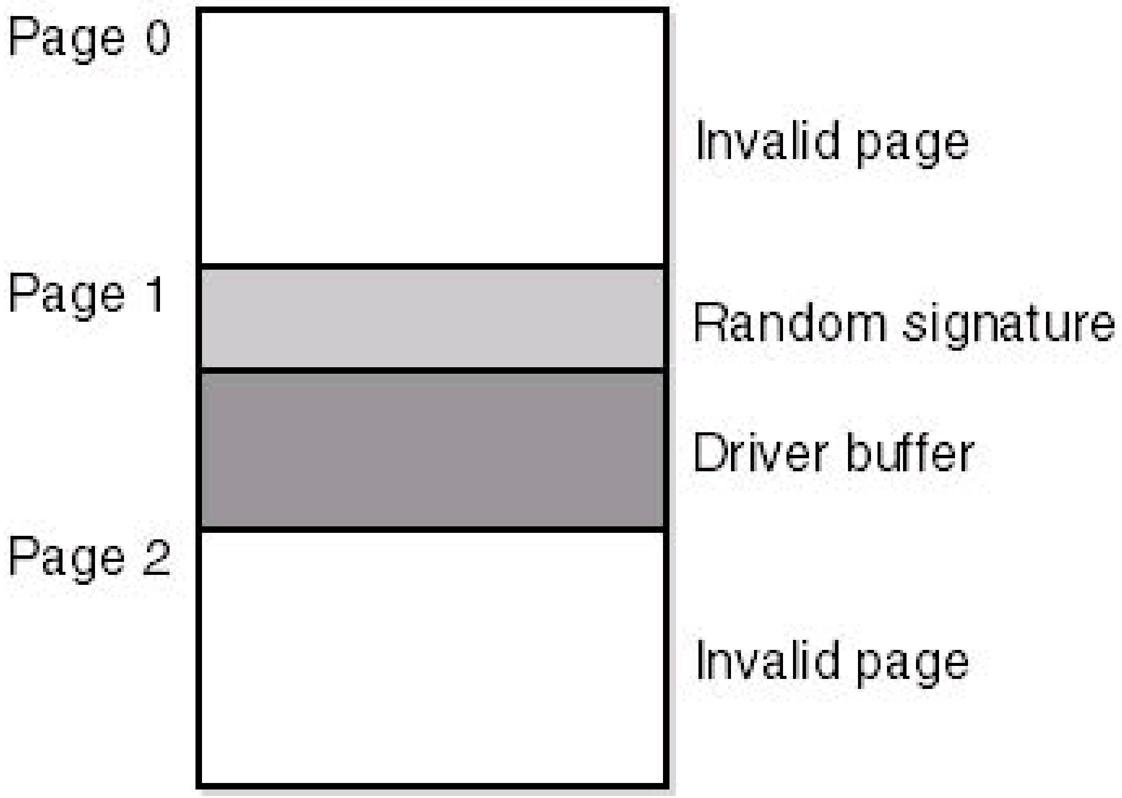
Driver Verifier

- Consist of support for system components (memory manager, I/O manager, HAL, Win32k.sys)
- \Winnt\System32\Verifier.exe configures which driver to verify
- On boot Verifier checks loaded drivers against verify list
- Replaces references to kernel functions with Driver Verifier‘s versions (approximately 40)

Driver Verifier (Special Pool)

- Brackets allocated memory with invalid pages
- Default: check for overrun error (allocated memory placed at end of page)
- Rest of page filled with random signature
- IRQL level checked
- Pool Tracking:
  - Device specifies tag on allocation
  - Poolmon tool can show how much memory is assigned to tag

Special Pool



© Mark Russinovich & David Solomon (used with permission of authors)

Driver Verifier (IRQL Checking)

- Forces pageable memory out of working set if IRQL is elevated
- Forces system crash if driver at wrong IRQL
- Low Resource Simulation:
  - Memory allocation randomly fails
  - Starts 7 minutes after boot

Outline

- Components, Services
- „Kinds of Memory“
- Address Space Layout
- Address Translation
- Physical Address Extension & Address Windowing Extensions
- Memory Objects and their usage

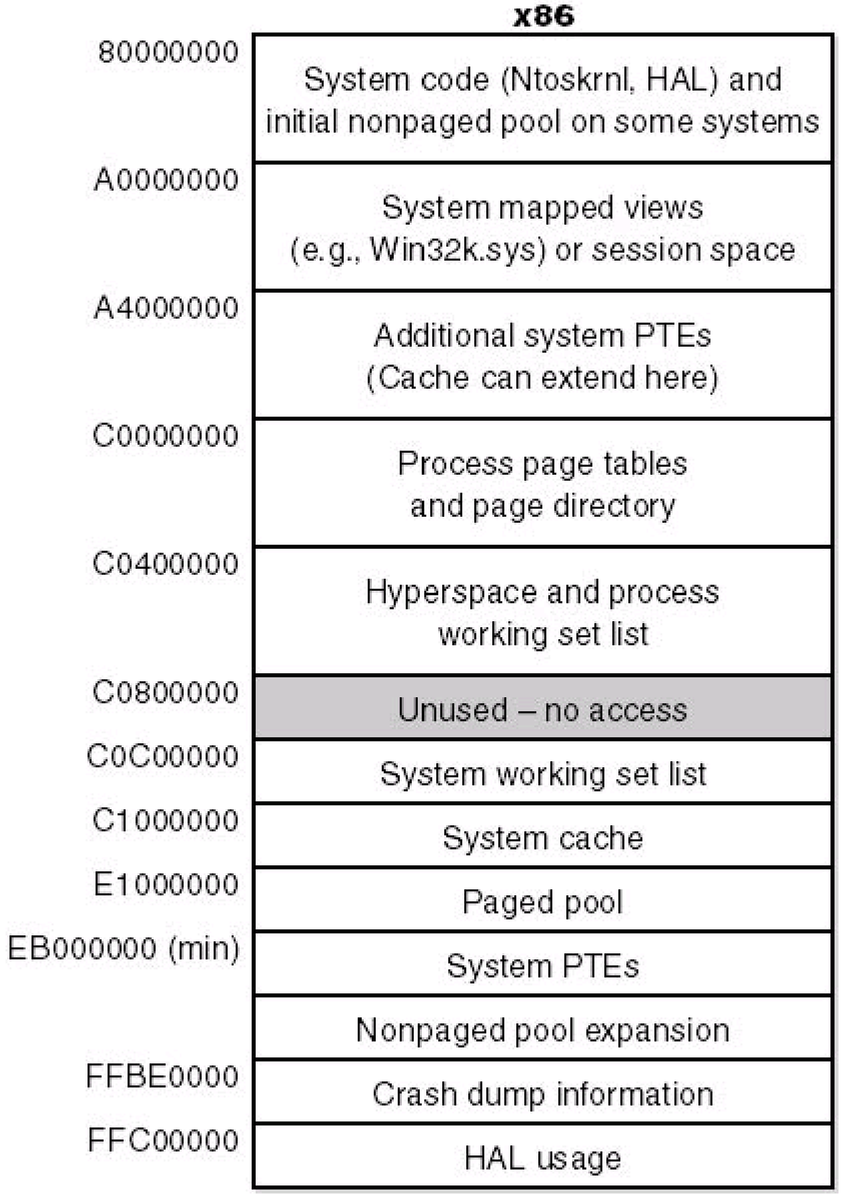
Address Space Layout

- Default: 2GB user / 2GB system
- Optional: 3GB user / 1GB system
  - Boot Advanced Server and Datacenter Server with /3GB flag
  - Application linked with /LARGEADDRESSAWARE
  - If Professional or Server booted with /3GB system restricted to 1GB, but user still 2GB
- Consumer Windows: 2GB user / 1GB shared / 1GB system

User Address Space

1. 0-64KB no-access region
2. 2GB – min.192KB private process address space
   (0x10000 – 0x7ffeffff)
3. 4KB thread environment block (TEB) for first thread
   (0x7ffde00 – 0x7ffdefff)
4. 4KB process environment block (PEB)
   (0x7ffdf00 – 0x7ffdffff)
5. 4KB shared user data page (ro) (clock, ...)
   (0x7ffe000 – 0x7ffe0fff)
6. 60KB no-access region (remainder of 64 KB)
7. 64KB no-access region (border protection)

System Address Space


x86
80000000
System code (Ntoskrnl, HAL) and initial nonpaged pool on some systems
A0000000
System mapped views (e.g., Win32k.sys) or session space
A4000000
Additional system PTEs (Cache can extend here)
C0000000
Process page tables and page directory
C0400000
Hyperspace and process working set list
C0800000
Unused – no access
C0C00000
System working set list
C1000000
System cache
E1000000
Paged pool
EB000000 (min)
System PTEs
Nonpaged pool expansion
FFBE0000
Crash dump information
FFC00000
HAL usage

© Mark Russinovich & David Solomon (used with permission of authors)

System Address Space (2)

- System code: system image, HAL , drivers for booting
- System mapped views: map Win32k.sys
- Session space: user session specific information
- Hyperspace: map process working set list, used to temporarily map physical pages for special operations (zero page, ...)
- System cache: map files which are open in system cache
- HAL usage: HAL-specific structures

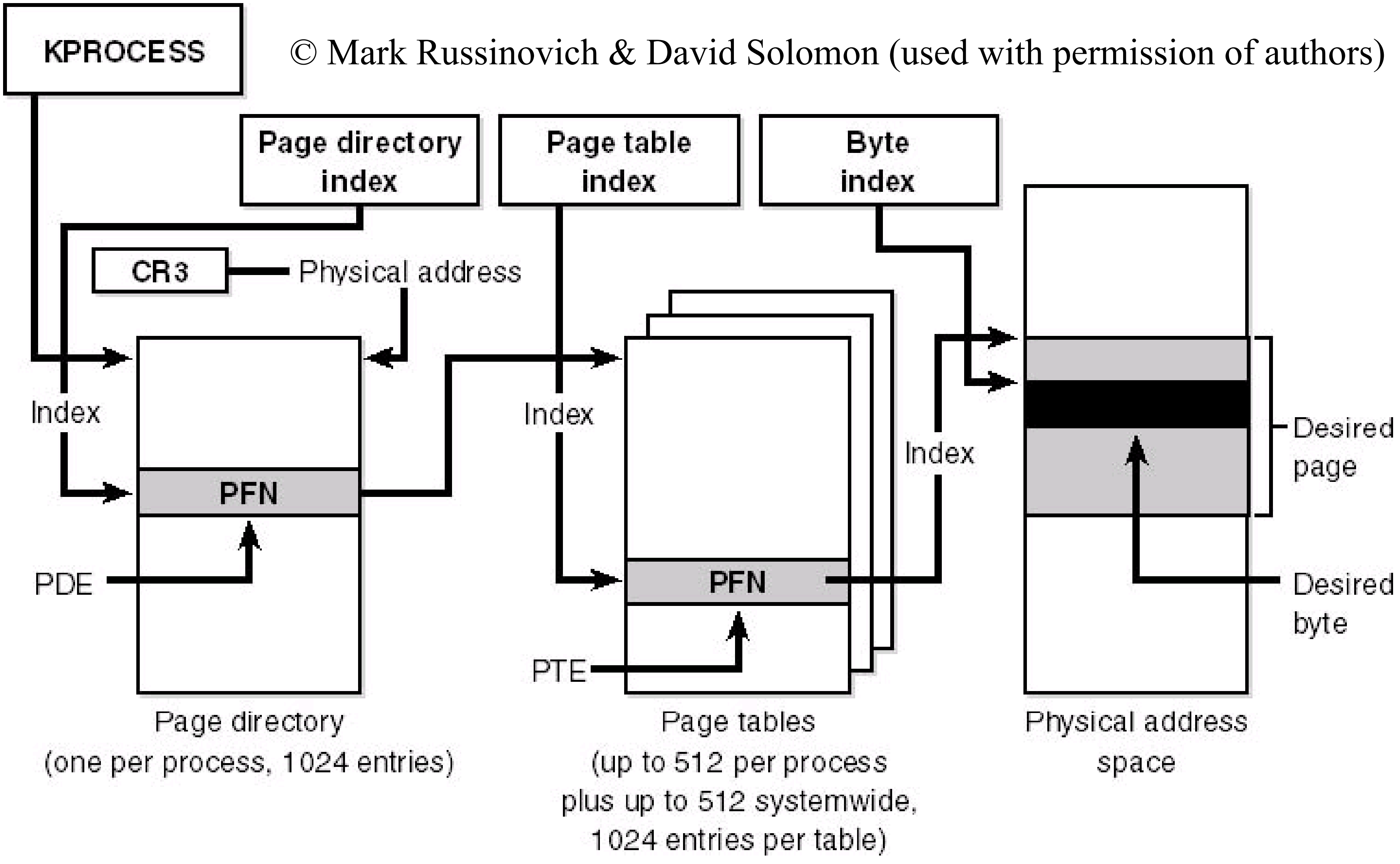
Session Space

- Session consist of:
  - Processes and system objects that represent single user
  - Paged pool area used by Win32k.sys to allocate session-private GUI objects
  - Own copy of Win32 subsystem process and logon process
- Session wide data structures are mapped to „session space“ region

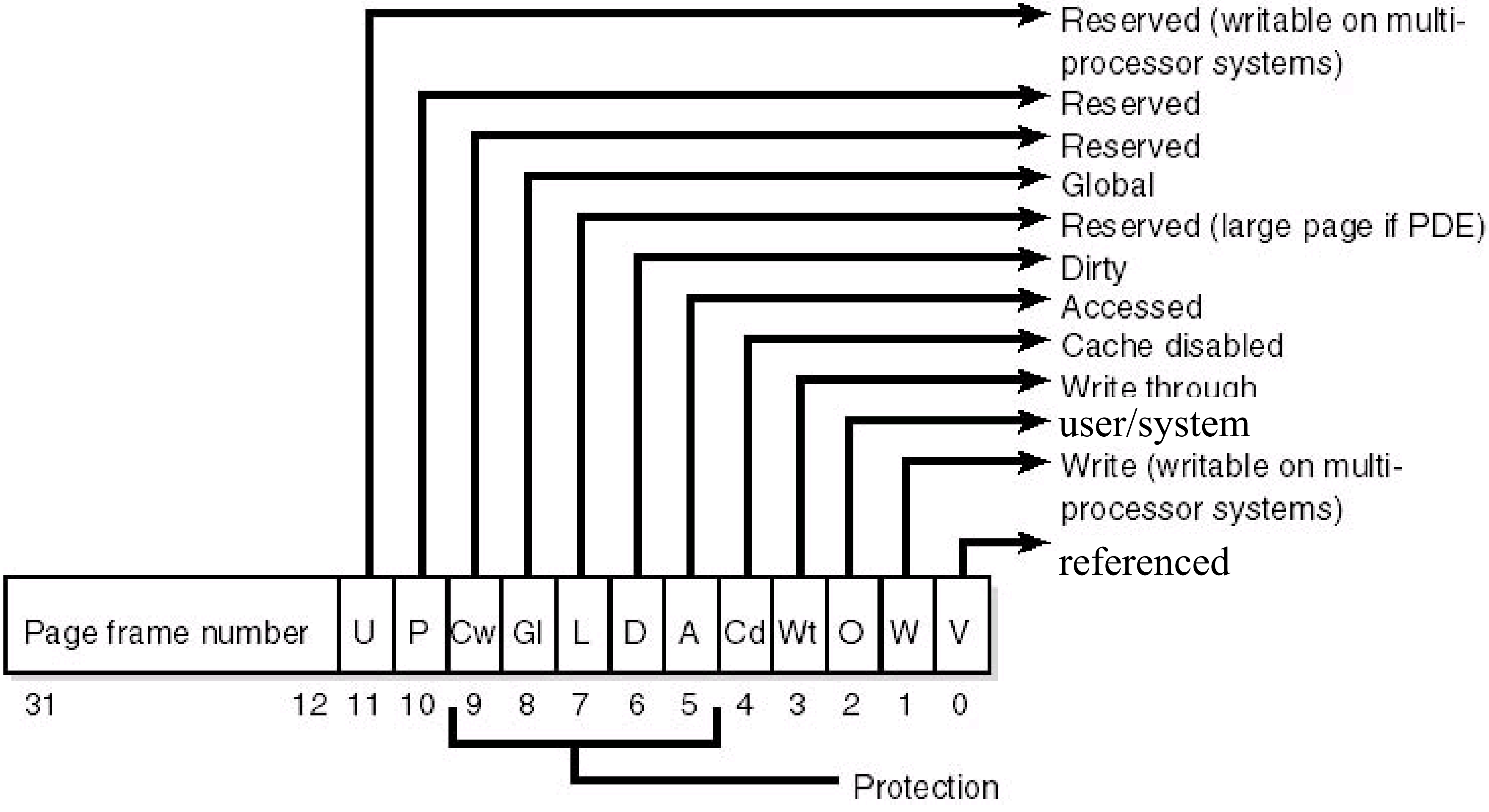
Outline

- Components, Services
- „Kinds of Memory“
- Address Space Layout
- Address Translation
- Physical Address Extension & Address Windowing Extensions
- Memory Objects and their usage

Address Translation



Page Table Entry



© Mark Russinovich & David Solomon (used with permission of authors)

PTE Bits

Name of Bit	Meaning
Accessed	Page has been read.
Cache disabled	Disables caching for that page.
Dirty	Page has been written to.
Global	Translation applies to all processes.
Large page	Indicates that the PDE maps a 4-MB page.
Owner (user/system)	Indicates whether user-mode code can access the page or whether the page is limited to kernel-mode access.
Valid (referenced)	Indicates whether the translation maps to a page in physical memory.
Write through	Disables caching of writes to this page so that changes are immediately flushed to disk.
Write	Uniprocessor: whether the page is read/write or read-only; Multiprocessor: whether the page is writable.

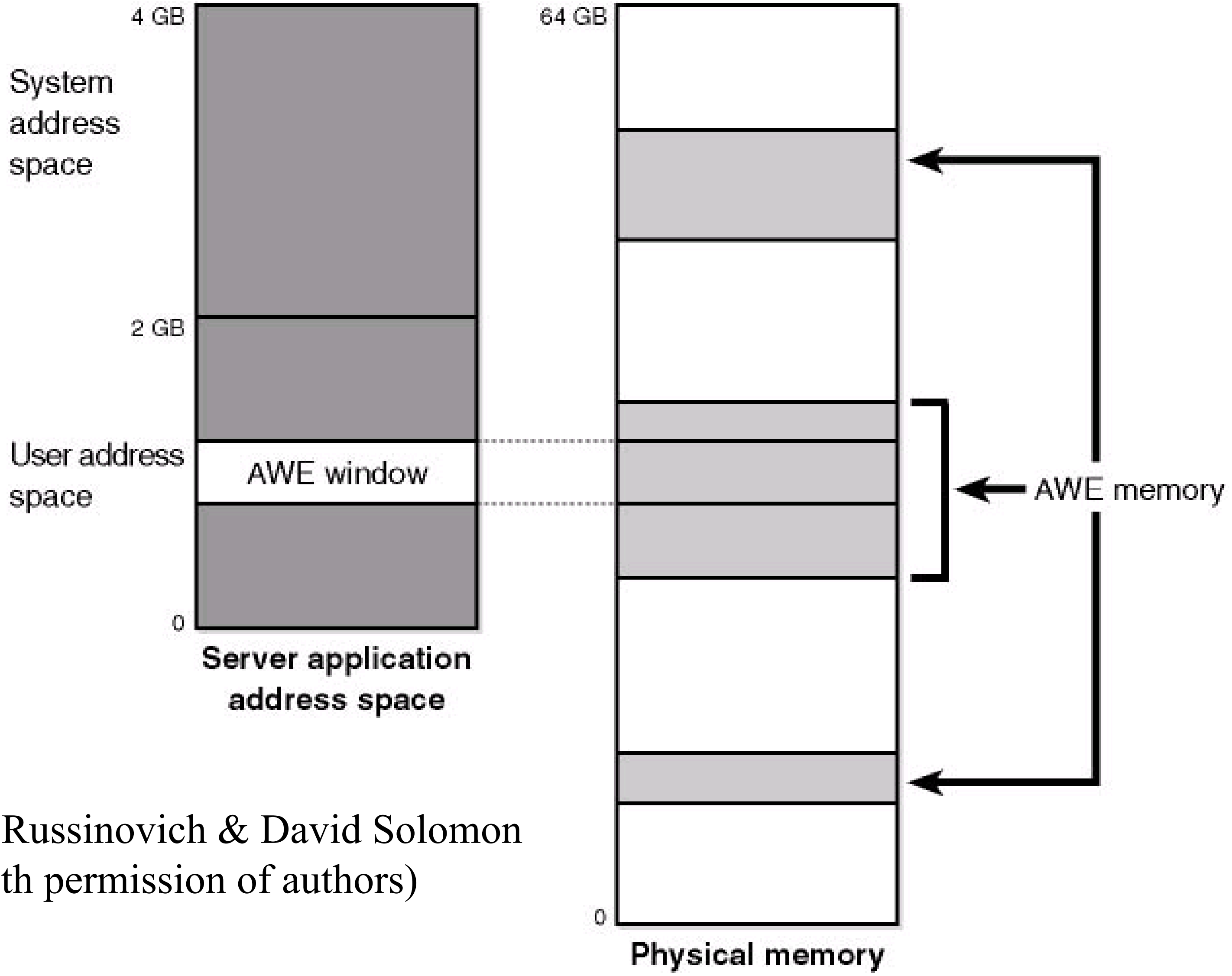
Outline

- Components, Services
- „Kinds of Memory“
- Address Space Layout
- Address Translation
- Physical Address Extension & Address Windowing Extensions
- Memory Objects and their usage

Address Windowing Extensions

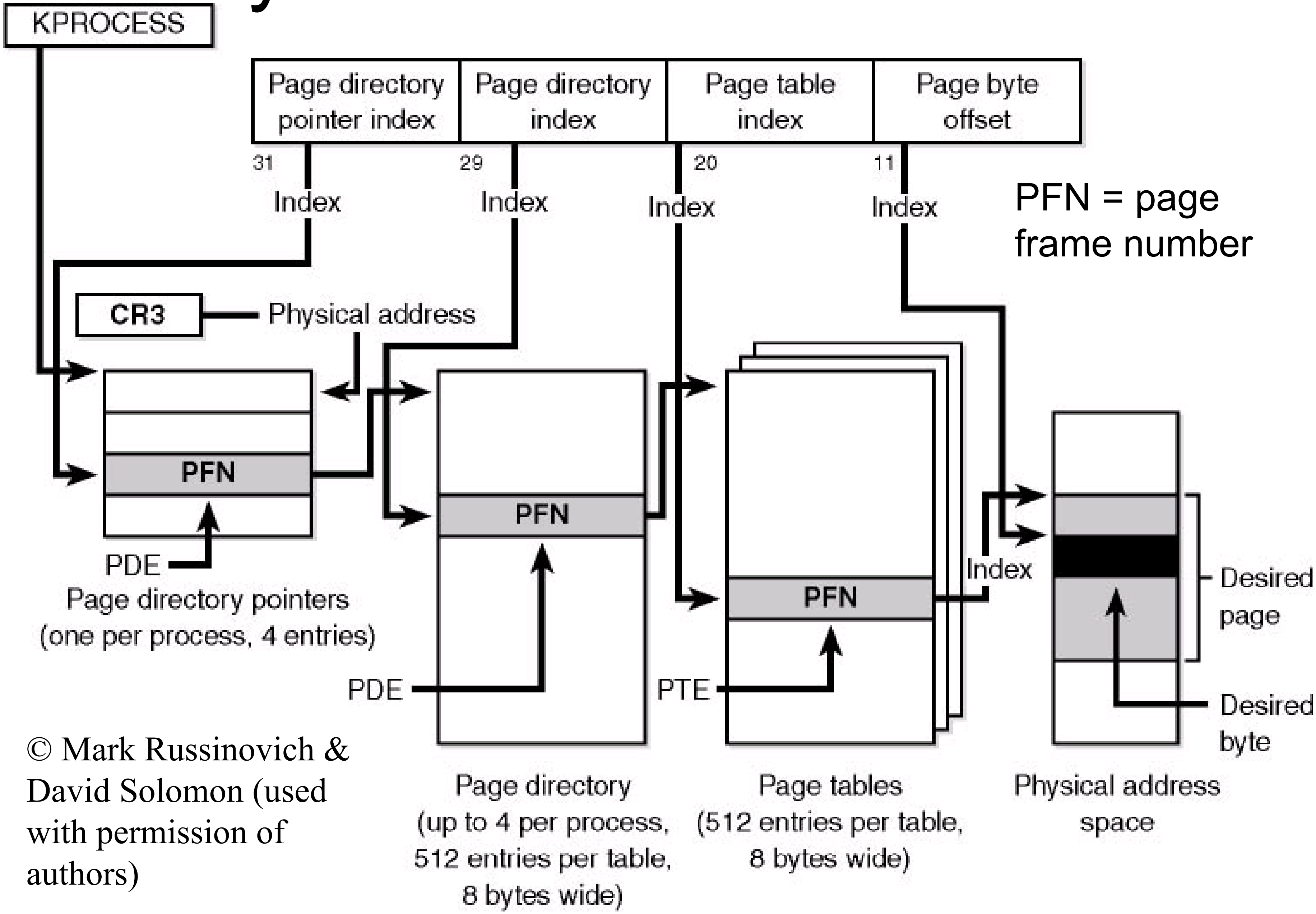
- Views of physical memory are mapped into virtual memory region
- Depending on Windows 2000 version different physical memory sizes are supported
- Pages can‘t be shared between processes
- The same physical page can‘t be mapped to more than one virtual address in the same process
- Page protection is limited to read/write

Address Windowing Extensions



© Mark Russinovich & David Solomon
(used with permission of authors)

Physical Address Extension


KPROCESS
Page directory pointer index
Page directory index
Page table index
Page byte offset
31
29
20
11
Index
Index
Index
Index
CR3
Physical address
PFN
PDE
Page directory pointers (one per process, 4 entries)
PFN
PDE
Page directory (up to 4 per process, 512 entries per table, 8 bytes wide)
PFN
PTE
Page tables (512 entries per table, 8 bytes wide)
Index
Desired page
Desired byte
Physical address space

PFN = page frame number

© Mark Russinovich & David Solomon (used with permission of authors)

Page Fault Handling

Reason for Fault	Result
Accessing a page that isn't resident in memory but is on disk in a page file or a mapped file	Allocate a physical page and read the desired page from disk and into the working set
Accessing a page that is on the standby or modified list	Transition the page to the process or system working set
Accessing a page that isn't committed (for example, reserved address space or address space that isn't allocated)	Access violation
Accessing a page from user mode that can be accessed only in kernel mode	Access violation
Writing to a page that is read-only	Access violation
Accessing a demand-zero page	Add a zero-filled page to the process working set

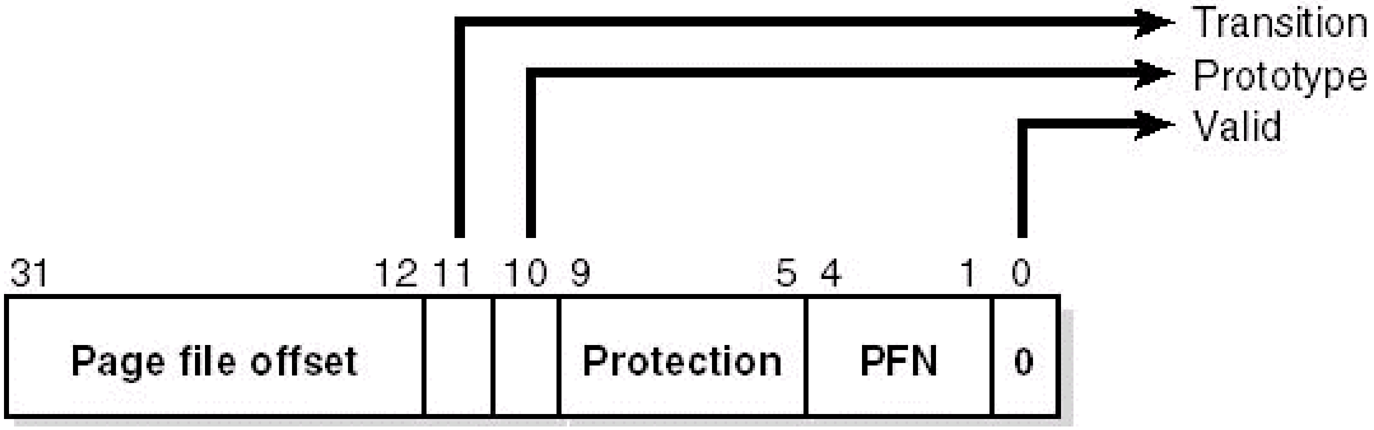
Page Fault Handling (2)

Reason for Fault	**Result**
Writing to a guard page	Guard-page violation (if a reference to a user-mode stack, perform automatic stack expansion)
Writing to a copy-on-write page	Make process-private (or session-private) copy of page and replace original in process, session, or system working set
Referencing a page in system space that is valid but not in the process page directory (for example, if paged pool expanded after the process page directory was created)	Copy page directory entry from master system page directory structure and dismiss exception
On a multiprocessor system, writing to a page that is valid but hasn't yet been written to	Set dirty bit in PTE

Outline

- Components, Services
- „Kinds of Memory“
- Address Space Layout
- Address Translation
- Physical Address Extension & Address Windowing Extensions
- Memory Objects and their usage

Invalid PTEs



PFN = page
file number

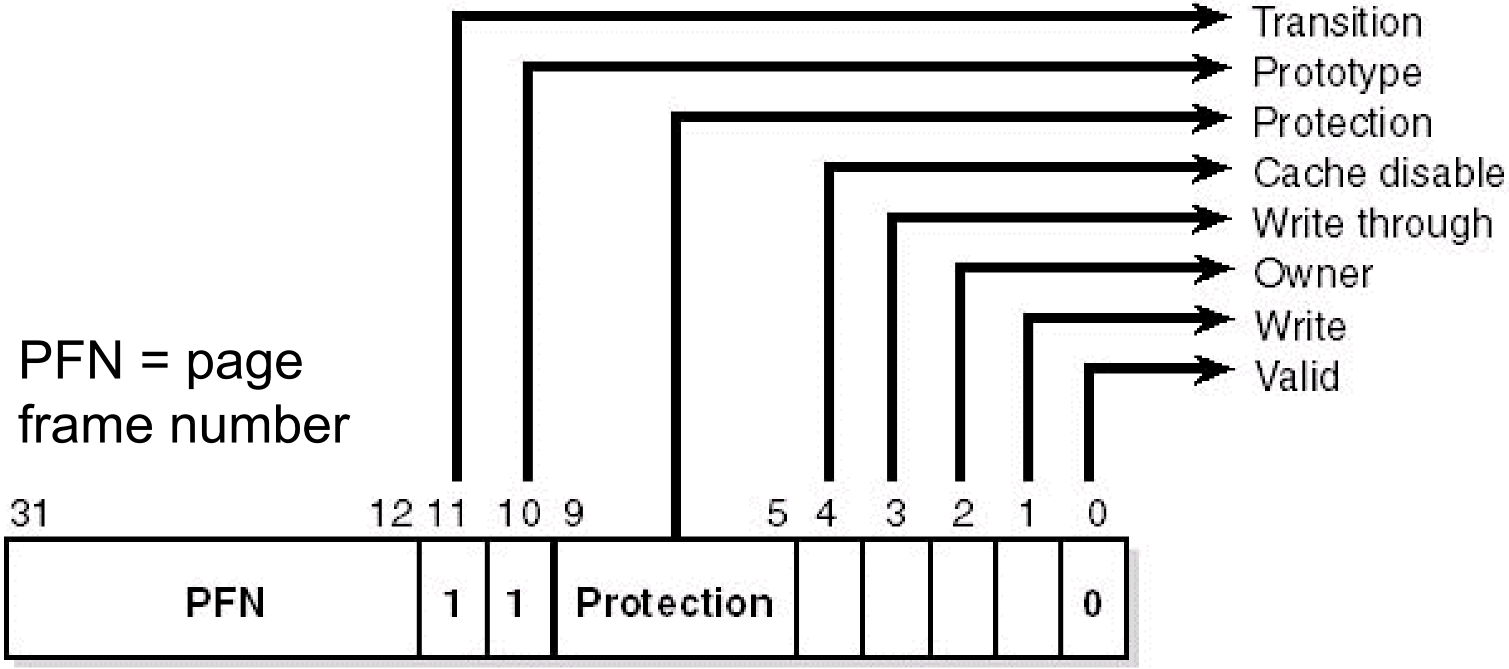
© Mark Russinovich & David Solomon (used with permission of authors)

Invalid PTEs

- Page file:
  - Page is in page file
  - Transition (11), Prototype (10) clear
  - Bits 1-4 contain page file number
  - Bits 12-31 contain page file offset
- Demand zero:
  - Page has to be replaced with zeroed page
  - As above, but page frame number and offset are zero

Invalid PTEs

PFN = page frame number



© Mark Russinovich & David Solomon (used with permission of authors)

Invalid PTEs

- Transition:
  - Page is in memory on standby, modified or modified-no-write list
  - Page is moved from list to working set
  - Transition and Prototype are 1
- Unknown:
  - PTE is zero or page table doesn‘t exist
  - Virtual Address Descriptor is examined
  - If memory is committed PTE is created

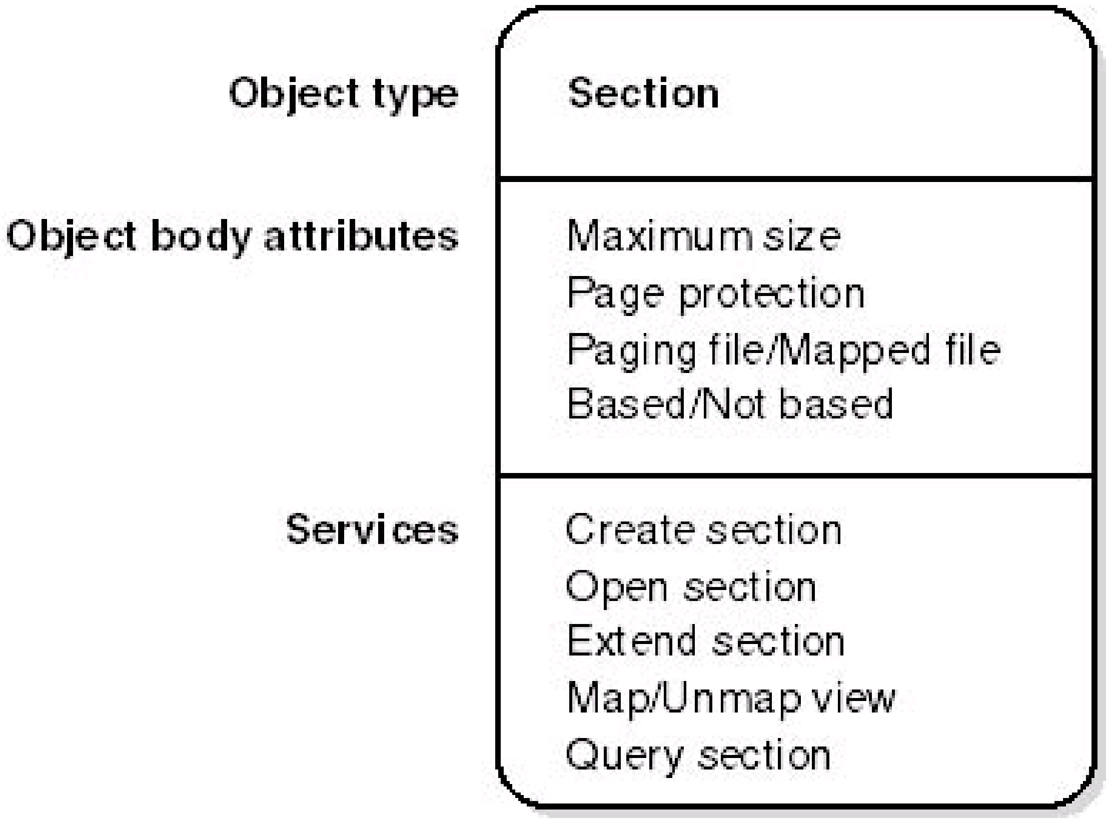
Shared Memory and Mapped Files

- Underlying primitives are called *section objects*
- Section object can be connected to file or committed memory
- If section has name other processes can open it
- Or grant access through handle inheritance or duplication
- Section object can refer to files larger than address space: map view of a section

Section Object

- Managed by Object manager
- Represents a block of memory that two or more processes can share
- Executive uses them to load images into memory
- Cache manager uses them to access data in cached file
- Memory manager does automatic update of disk file (write) and memory (read)
- Contains prototype PTEs

Section Object

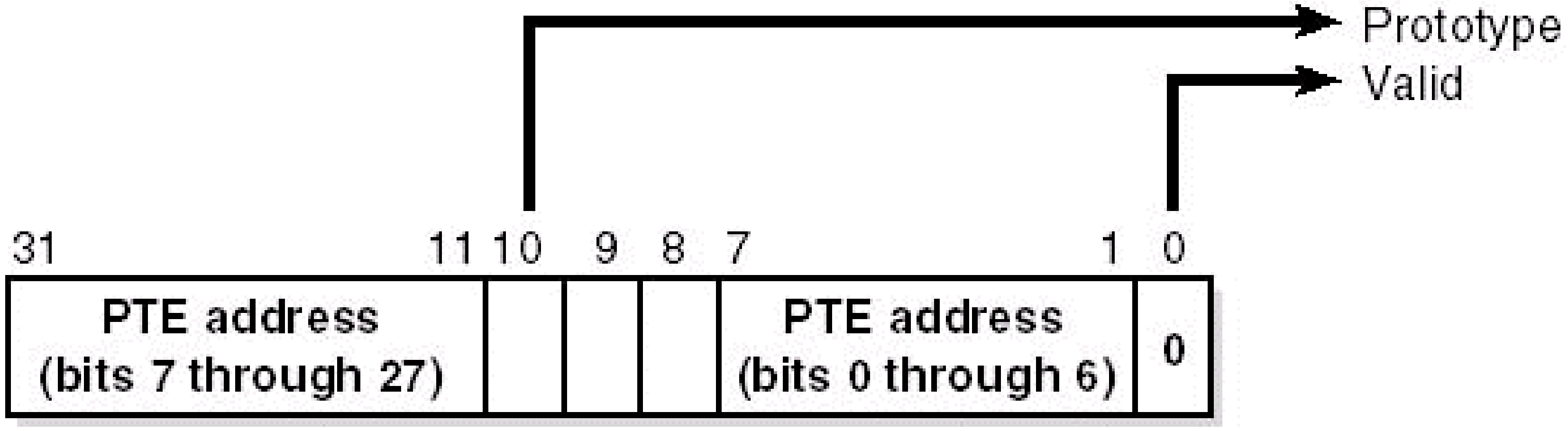


Object type	**Section**
Object body attributes	Maximum size Page protection Paging file/Mapped file Based/Not based
Services	Create section Open section Extend section Map/Unmap view Query section

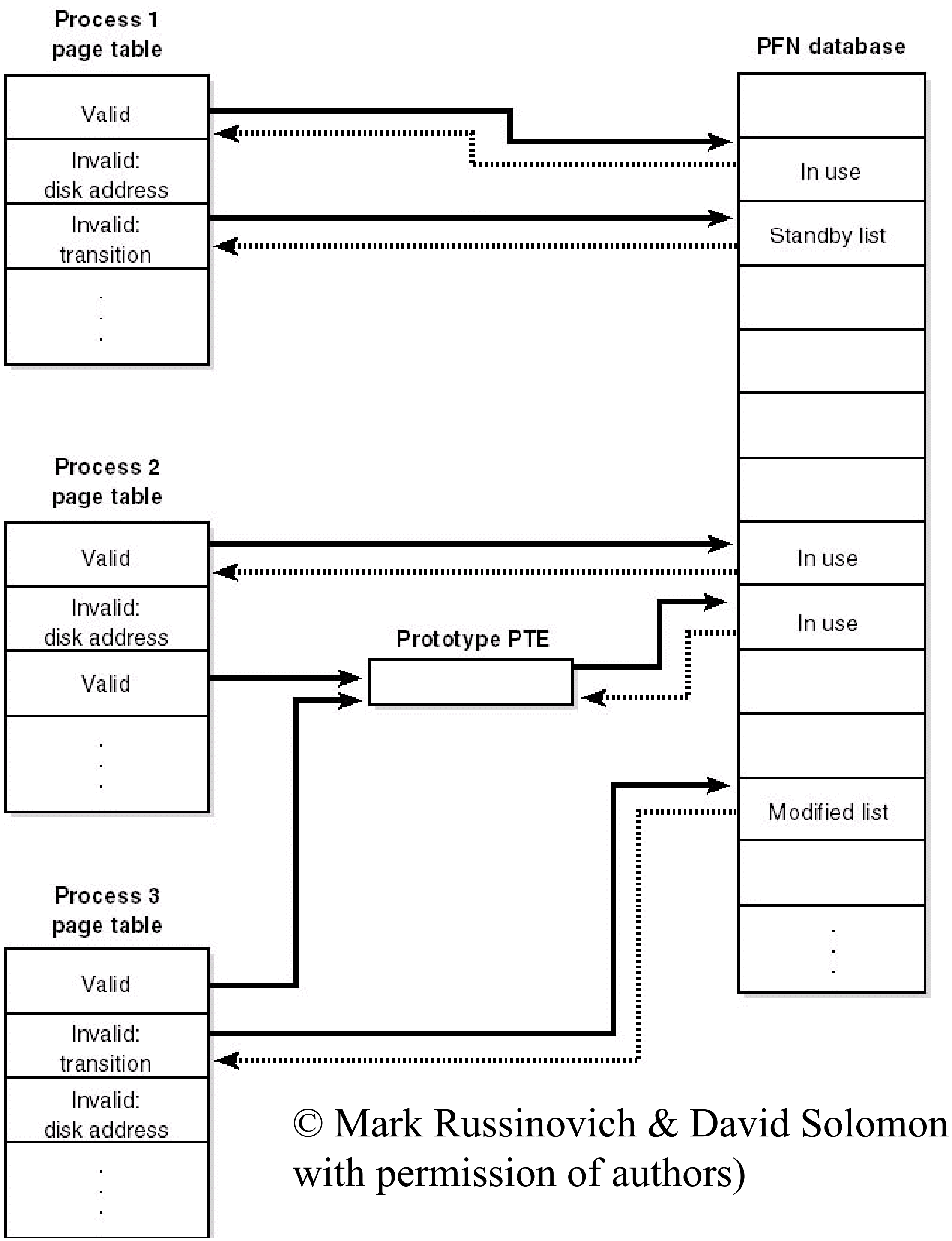
Prototype PTEs

- Used to manage shared pages in sections
- Created when section is created
- Central location to manage shared page
- „real“ PTE points to prototype PTE

Invalid PTE pointing to Prototype PTE



© Mark Russinovich & David Solomon (used with permission of authors)



© Mark Russinovich & David Solomon (used with permission of authors)

Collided Page Faults

- Page fault on page, which is currently in-paged
- Pager detects collision by examining page frame number (PFN) database (see below)
- Page uses event from PFN database entry to wait for
- When event enters signaled state all waiting threads are satisfied

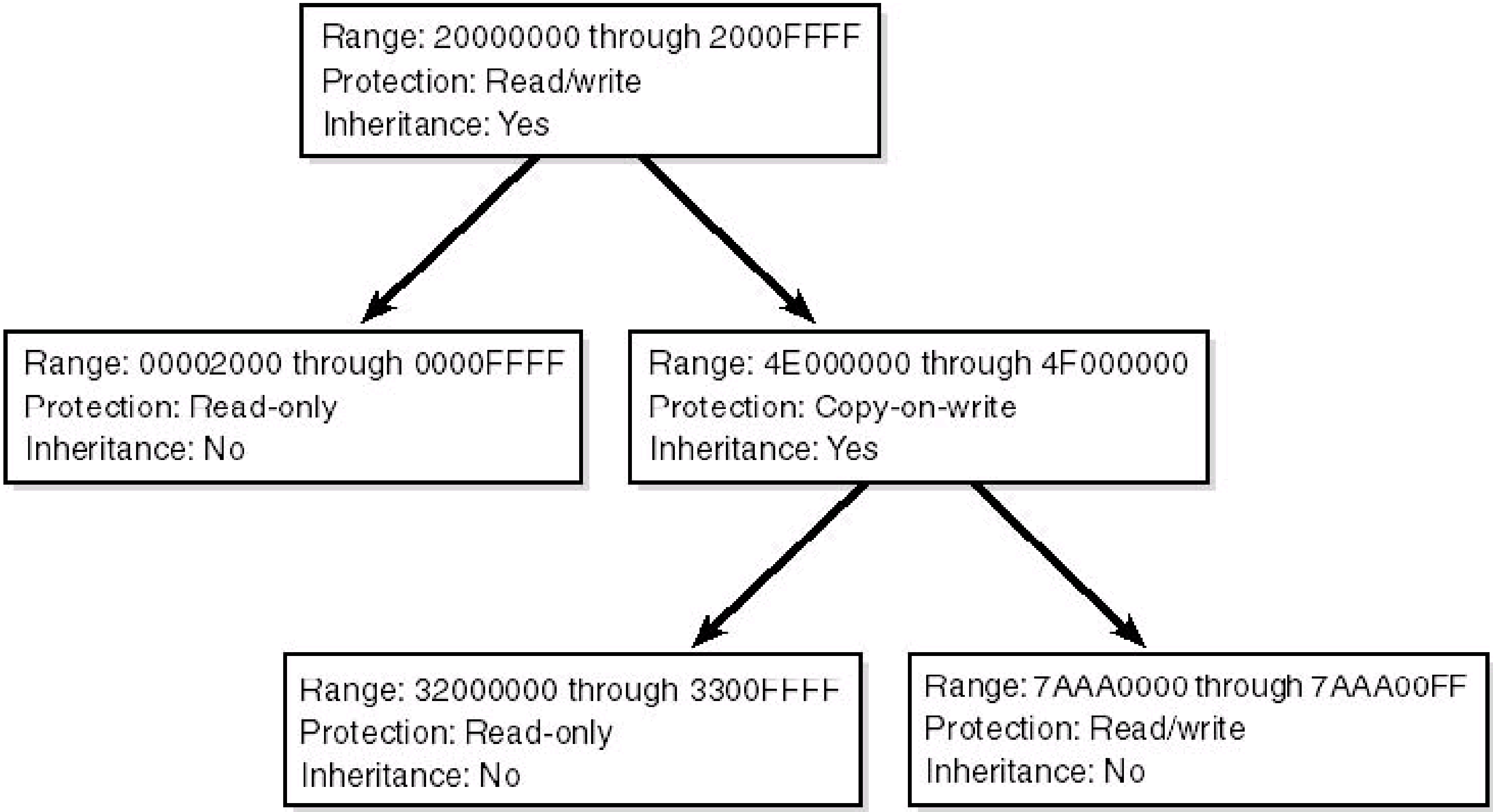
Page File

- Up to 16 page files supported
- Space not reserved until pages are written to disk
- System process maintains open handle to each page file (can‘t be deleted)

Virtual Address Descriptors

- Memory manager uses lazy evaluation to construct page tables
- Memory manager cannot use page tables to determine allocated memory
- Allocated ranges are stored in VAD
- VADs are stored in self-balanced binary tree
- If access to address outside of VADs and not committed → invalid access

VAD (2)



Range: 20000000 through 2000FFFF
Protection: Read/write
Inheritance: Yes

Range: 00002000 through 0000FFFF
Protection: Read-only
Inheritance: No

Range: 4E000000 through 4F000000
Protection: Copy-on-write
Inheritance: Yes

Range: 32000000 through 3300FFFF
Protection: Read-only
Inheritance: No

Range: 7AAA0000 through 7AAA00FF
Protection: Read/write
Inheritance: No

Working Sets

- Paging policies:
  - Fetch policy:
    - When to bring page into memory
  - Placement policy:
    - Where to put page in physical memory
    - Consider cache size
  - Replacement policy:
    - When to replace a page

Fetch policy

- Windows 2000 uses demand-paging with clustering:
  - Read a page when it is faulted
  - Read a small number of pages ahead (cluster)
  - Cluster size depends on physical memory size (8 for code in images, 4 for data in images, 8 for other pages)

Replacement Policy

- Multiprocessor:
  - Variation of local FIFO algorithm
- Uniprocessor:
  - Clock algorithm
- Number of frames for each process make up its working set

Working Set Management

- Process starts with default minimum and maximum WS size
- Maximum can‘t exceed system wide maximum (available pages – 512 or 1984MB)
- On page fault:
  - Examine WS limits and free memory
  - If WS below max and free pages → increase WS
  - If WS below max and no free pages → replace pages

Working Set Management (2)

- Automatic WS trimming if low on memory
  - Page removed from WS if above min
  - Page removed if not accessed (Access bit in PTE – Clock)
  - Don‘t use Access bit on multiprocessor (requires cache invalidation)
  - If thread incurs more than a few page-faults since last trimming, trimming is stopped for that thread (assumes that faulted pages have been trimmed)

Balance Set Manager

- Activated once per second or explicitly (system running low on free memory)
- When activated once per second:
  - Wake up swapper every fourth time
  - Check look-aside lists and adjust size
  - Trim working sets
- Swapper:
  - Looks for threads which have been in wait for 3-7 seconds
  - Swaps thread‘s kernel stack
  - If all threads of process are swapped, process is swapped

PFN Database

- Page Frame Number database
- Working Set describes resident pages
- PFN database describes state of each page in physical memory
- Page is in one of eight states
- Database is array of structure representing each physical page of memory

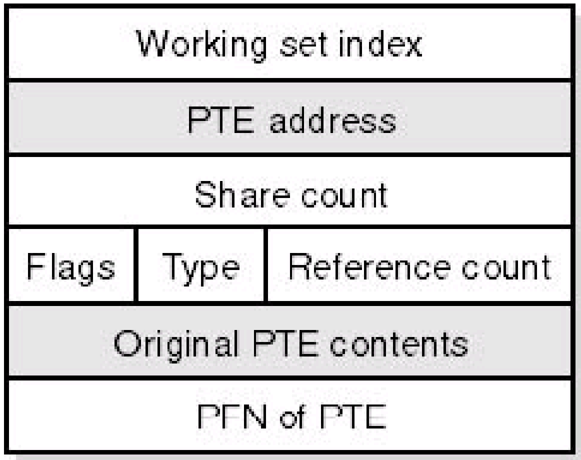
PFN Data Structures

- PFN database entry can have different states
- Fields have different meanings depending on state
- States:
  - see next slide...

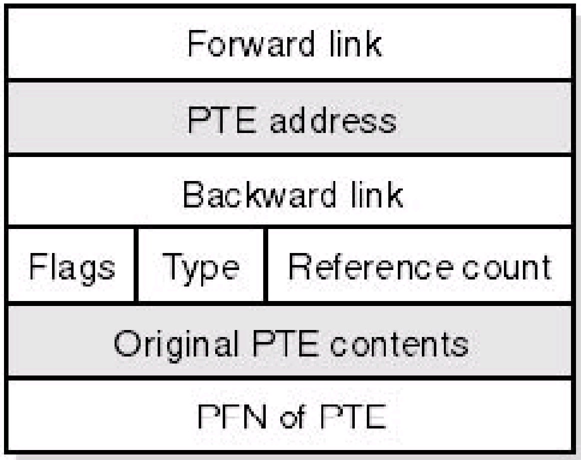
Page States

Status	**Description**
Active / Valid	Page is part of a working set or not in any working set and a valid PTE points to it
Transition	I/O to the page is in progress. Page isn‘t owned by working set and not on any paging list
Standby	Page was just removed from working set and has not been modified since last written to disk.
Modified	Page was just removed from working set and has been modified since last written to disk.
Modified no-write	Same as Modified, but marked as not to be written to disk.
Free	Page is free, but has unspecified dirty data in it.
Zeroed	Page is free and has been initialized with zeros.
Bad	Page can‘t be used. (e.g. Parity error)

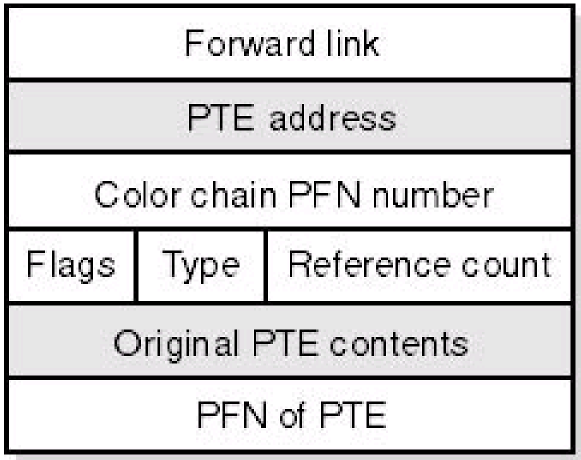
PFN Data Structures



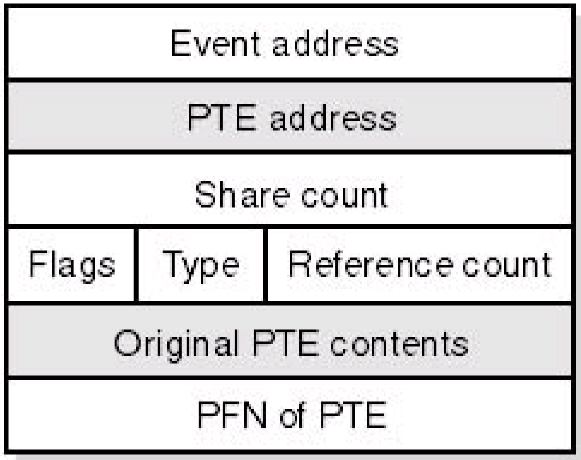
PFN for a page in a working set



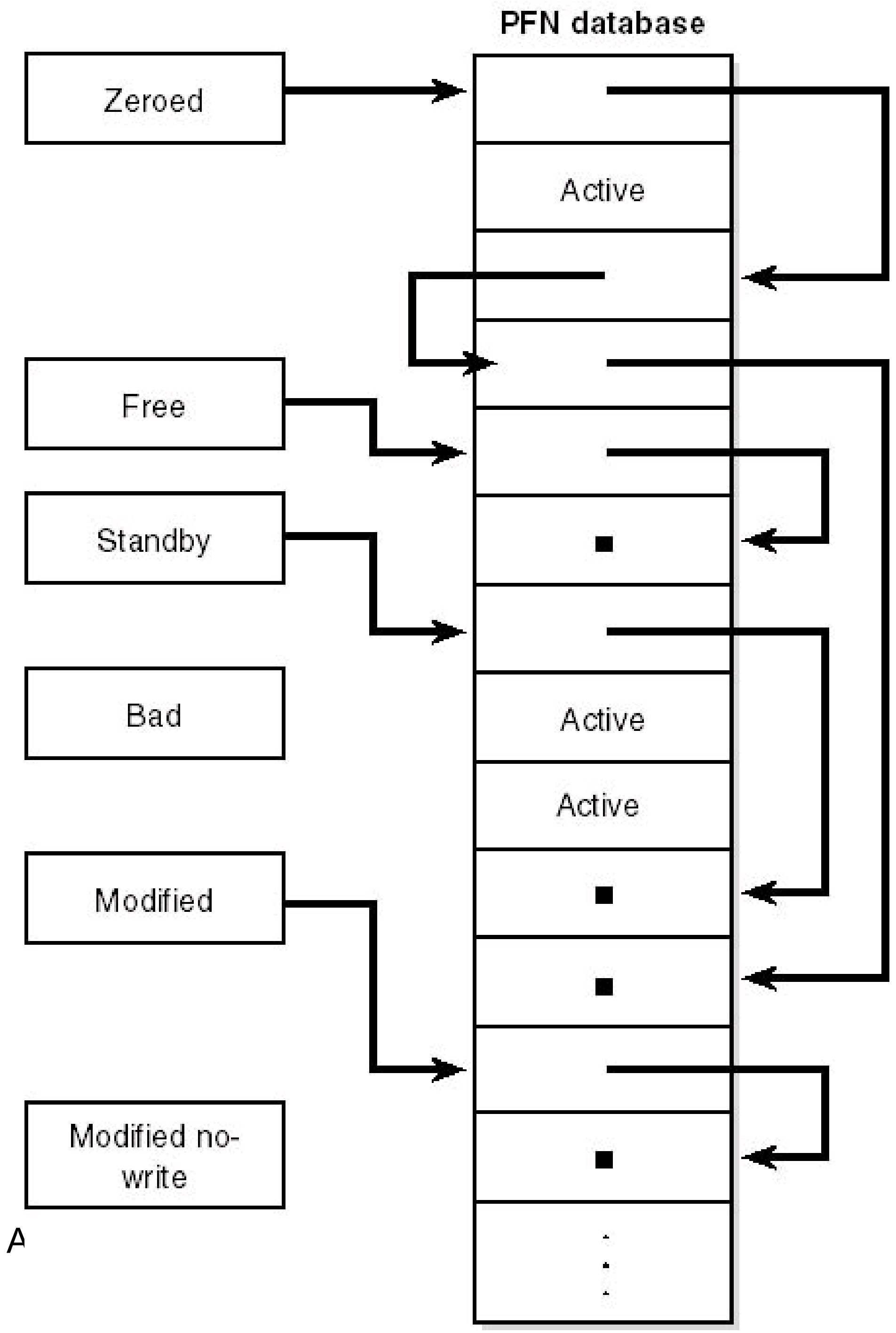
PFN for a page on the standby or the modified list



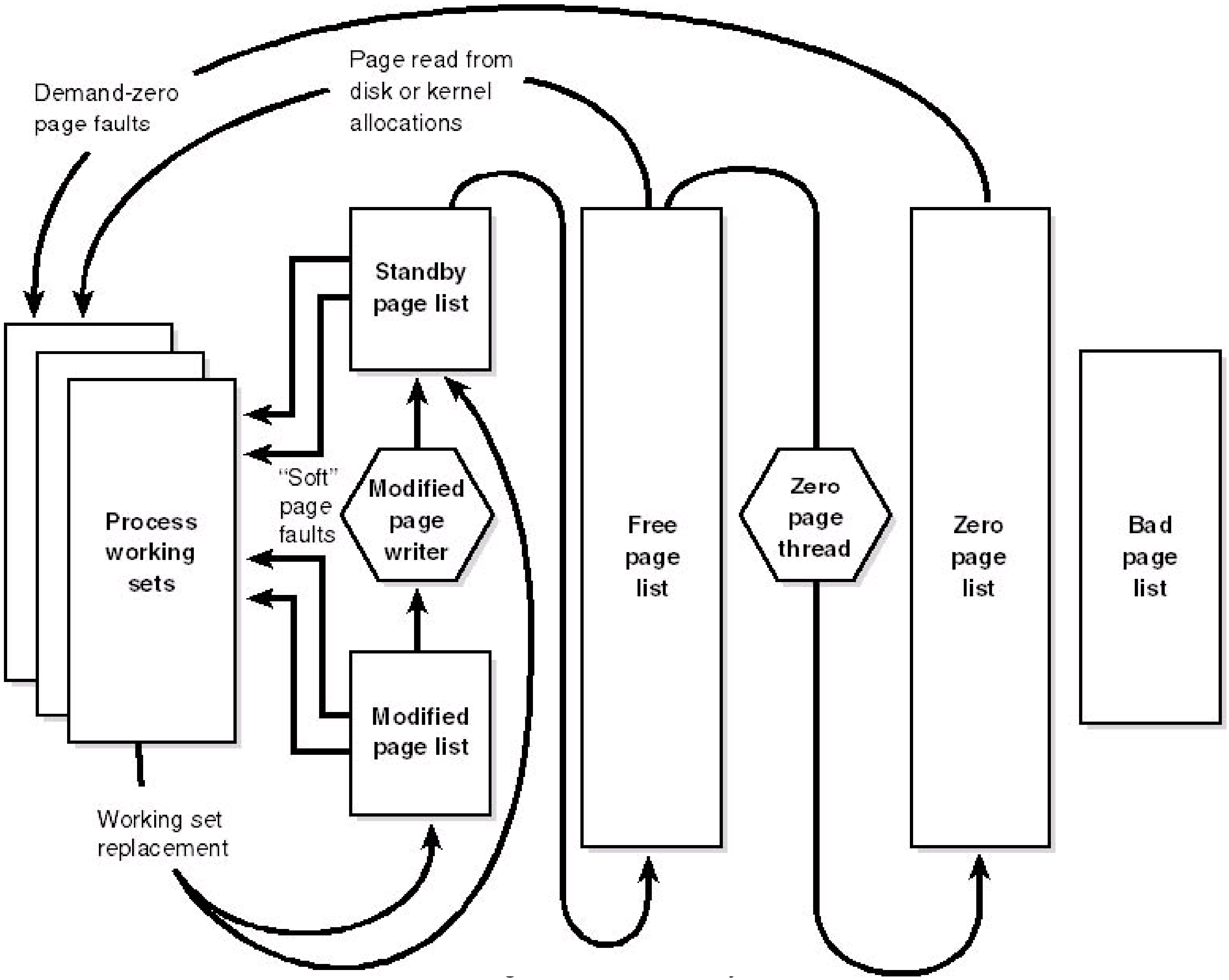
PFN for a page on the zero or free list



PFN for a page with an I/O in progess



© Mark Russinovich & David Solomon (used with permission of authors)



© Mark Russinovich & David Solomon (used with permission of authors)

List Dynamics

- Demand-Zero Page-Fault:
  1. Use zero page, if none available:
  2. Use free page and zero it, if none available:
  3. Use standby page and zero it
- Zero pages
  – Generated from free list by *zero page thread*
  – Runs when free list has 8 or more pages
  – Runs at priority 0 (lowest)

Modified Page Writer

- Two threads:
  - One to write to page file
  - One to write to mapped files
  - Avoid deadlock (write to mapped file causes page fault, which requires free page, which requires dirty page to be written)
- Invoked if:
  - Modified page exceed maximum value (>300)
  - Number of available pages goes below minimum of free pages
  - Regularly every 300 seconds

Modified Page Writer (2)

- Write pages:
  - Remove from modified list and set PTE to transition
  - Initiate I/O
  - After I/O completed successfully, frame is placed at end of standby list
  - Every 5 min.: search for contiguous frames from mapped file (one I/O)
  - If frame is referenced by other process during I/O: share count is increased → frame remains in modified list