Windows Internals

Module 2: Basic Concepts

Pavel Yosifovich

CTO, CodeValue

pavely@codevalue.net

http://blogs.Microsoft.co.il/blogs/pavely

pluralsight
hardcore developer training

Contents

User mode vs. kernel mode

- **Thread access mode**
- **User mode**
  - Allows access to non-operating system code & data only
  - No access to the hardware
  - Protects user applications from crashing the system
- **Kernel mode**
  - Privileged mode for use by the kernel and device drivers only
  - Allows access to all system resources
  - Can potentially crash the system

Processes

- **Process**
  - A set of resources used to execute a program
- **A process consists of**
  - A private virtual address space
  - An executable program, referring to an image file on disk which contains the initial code and data to be executed
  - A table of handles to various kernel objects
  - A security context (access token), used for security checks when accessing shared resources
  - One or more threads that execute code

Demo

Task Manager

Demo

Process Explorer

Threads

- **Thread**
  - Entity that is scheduled by the kernel to execute code
- **A thread contains**
  - The state of CPU registers
  - Current access mode (user mode or kernel mode)
  - Two stacks, one in user space and one in kernel space
  - A private storage area, called Thread Local Storage (TLS)
  - Optional security token
  - Optional message queue and Windows the thread creates
  - A priority, used in thread scheduling
  - A state: running, ready, waiting

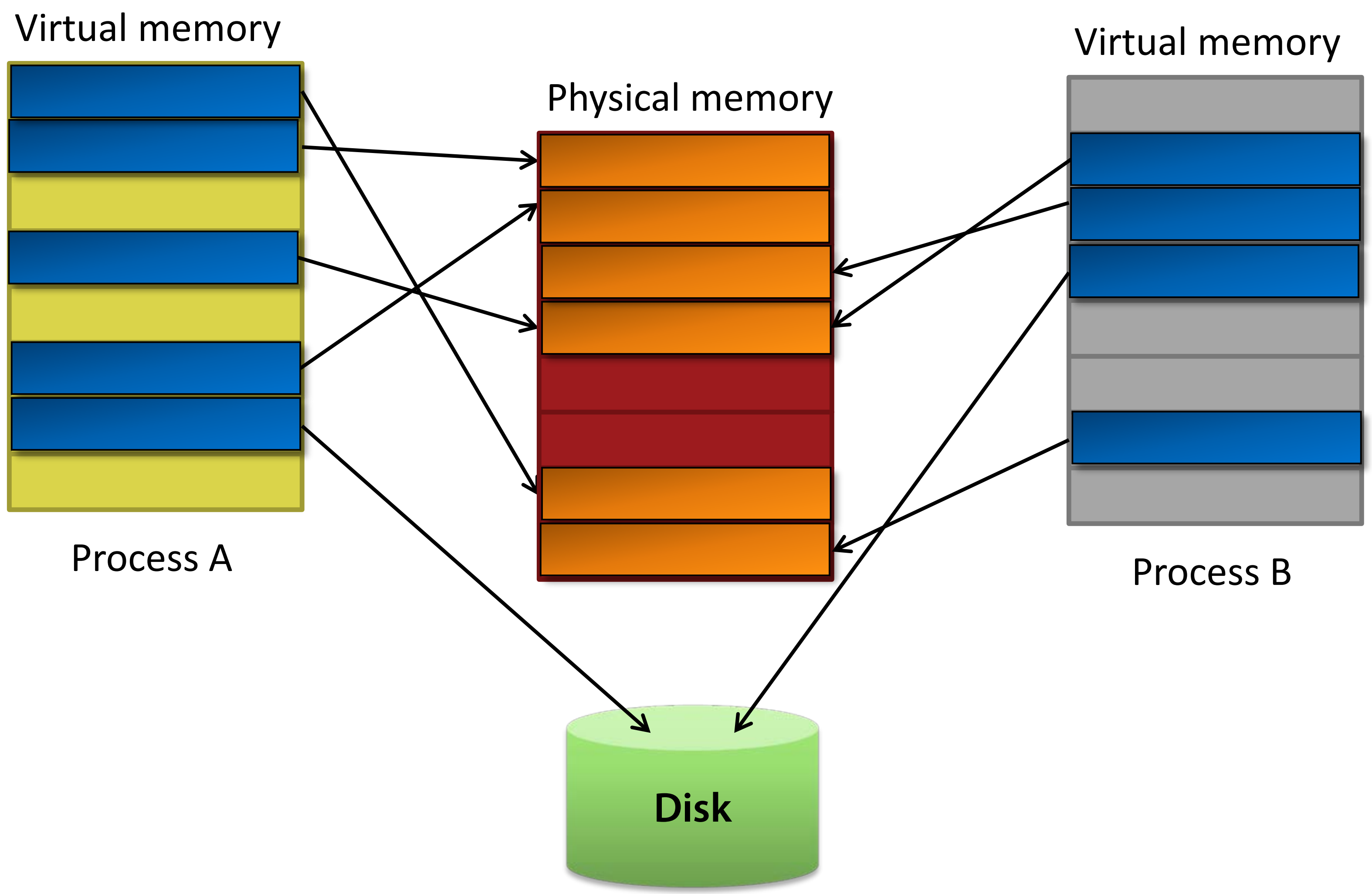
Demo

Threads

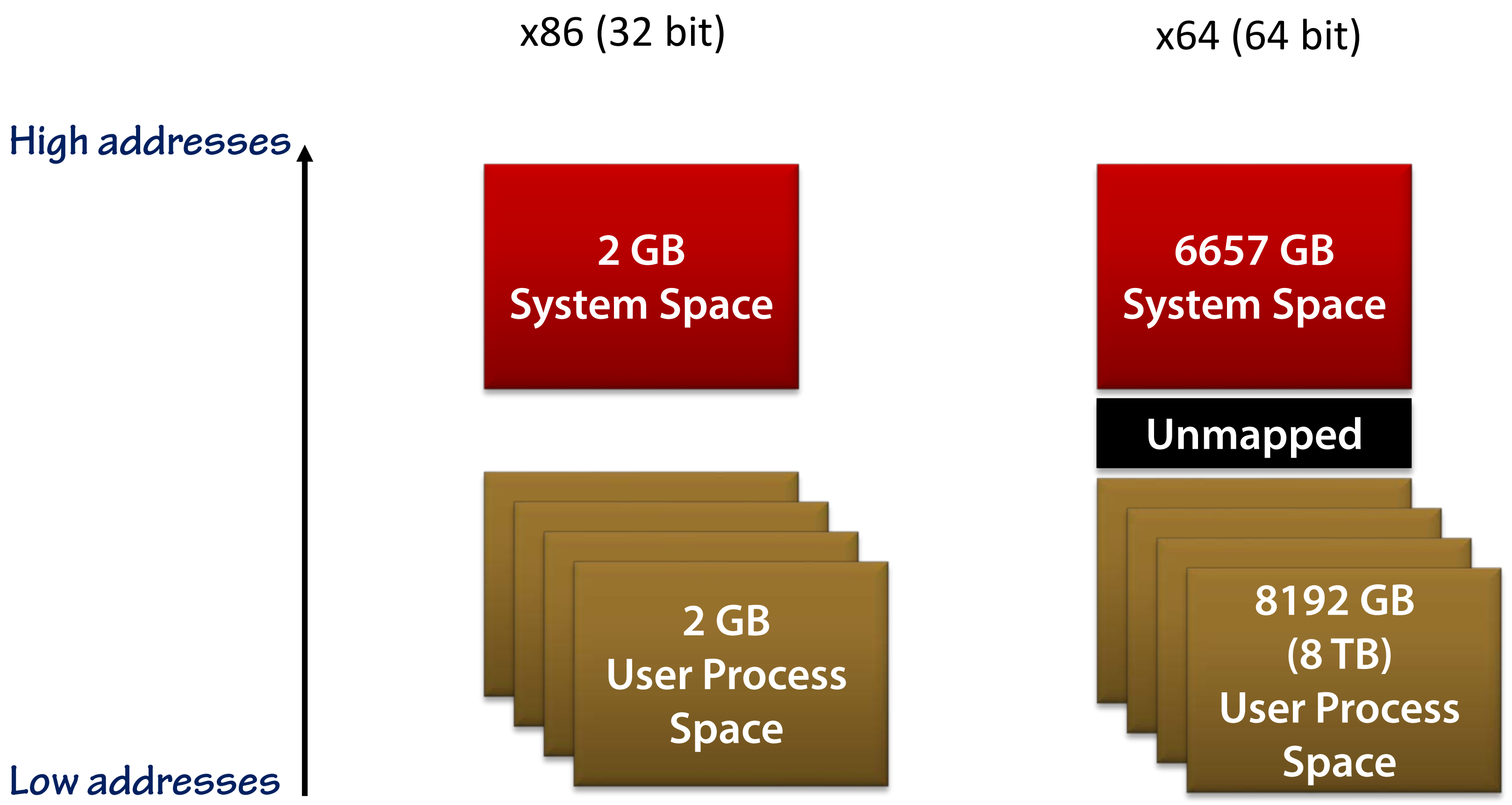
Virtual Memory

- **Each process “sees” a flat linear memory**
- **Internally, virtual memory may be mapped to physical memory, but may also be stored on disk**
- **Processes access memory regardless of where it actually resides**
  - The memory manager handles mapping of virtual to physical pages
  - Processes cannot (and need not) know the actual physical address of a given address in virtual memory

Virtual Memory Mapping



Virtual Memory Layout



Demo

Virtual Memory

Objects and Handles

- **Objects are runtime instances of static structures**
  - Examples: process, mutex, event, desktop, file
- **Reside in system memory space**
- **Kernel code can obtain direct pointer to an object**
- **User mode code can only obtain a handle to an object**
  - Shields user code from directly accessing an object
- **Objects are reference counted**
- **The Object Manager is the entity responsible for creating, obtaining and otherwise manipulating objects**

Demo

Objects and handles

Summary

- A process is a management container for threads to execute code
- A Thread executes code on a CPU
- Multiple threads can execute concurrently on multiple CPUs
- Per process virtual memory provides a private address space isolated from other processes
- Kernel objects are accessed from user mode using private process handles