Windows Internals

Module 6: Processes & Threads (Part 2)

Pavel Yosifovich

CTO, CodeValue

pavely@codevalue.net

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

pluralsight
hardcore developer training

Thread Stacks

- **Every user mode thread has two stacks**
    - In kernel space (12K (x86), 24K (x64))
        - Resides in physical memory (most of the time)
    - In user space (may be large)
        - By default 1MB is reserved, 64KB committed
        - A guard page is placed just below the last committed page, so that the stack can grow
        - Can change the initial size
            - Using linker settings as new defaults
            - On a thread by thread basis in the call to **CreateThread** / **CreateRemoteThread(Ex)**
            - Can specify a new committed or reserved size, but not both
            - Committed is assumed, unless the flag **STACK_SIZE_PARAM_IS_A_RESERVATION** is used

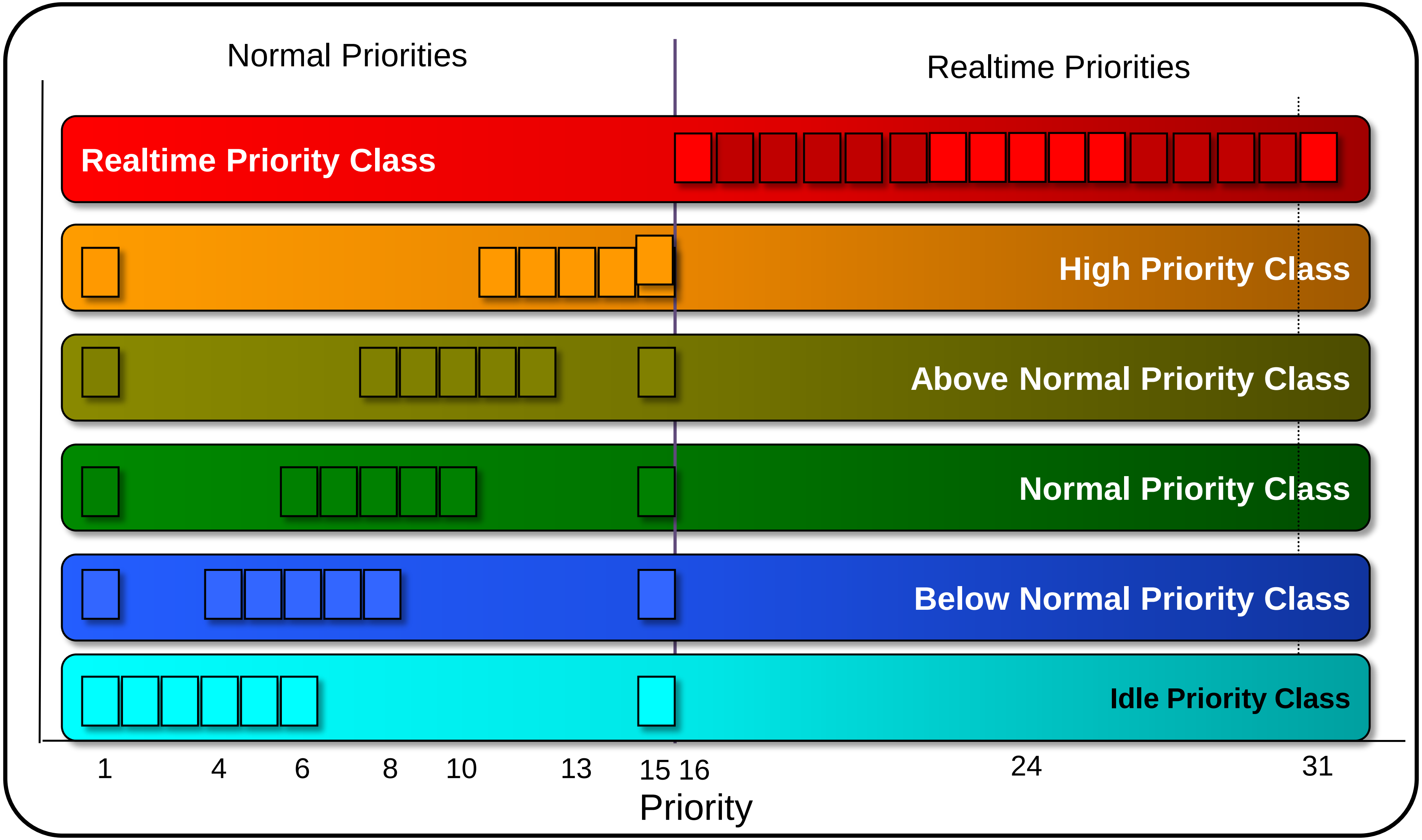
Demo

Thread stacks

Thread Priorities

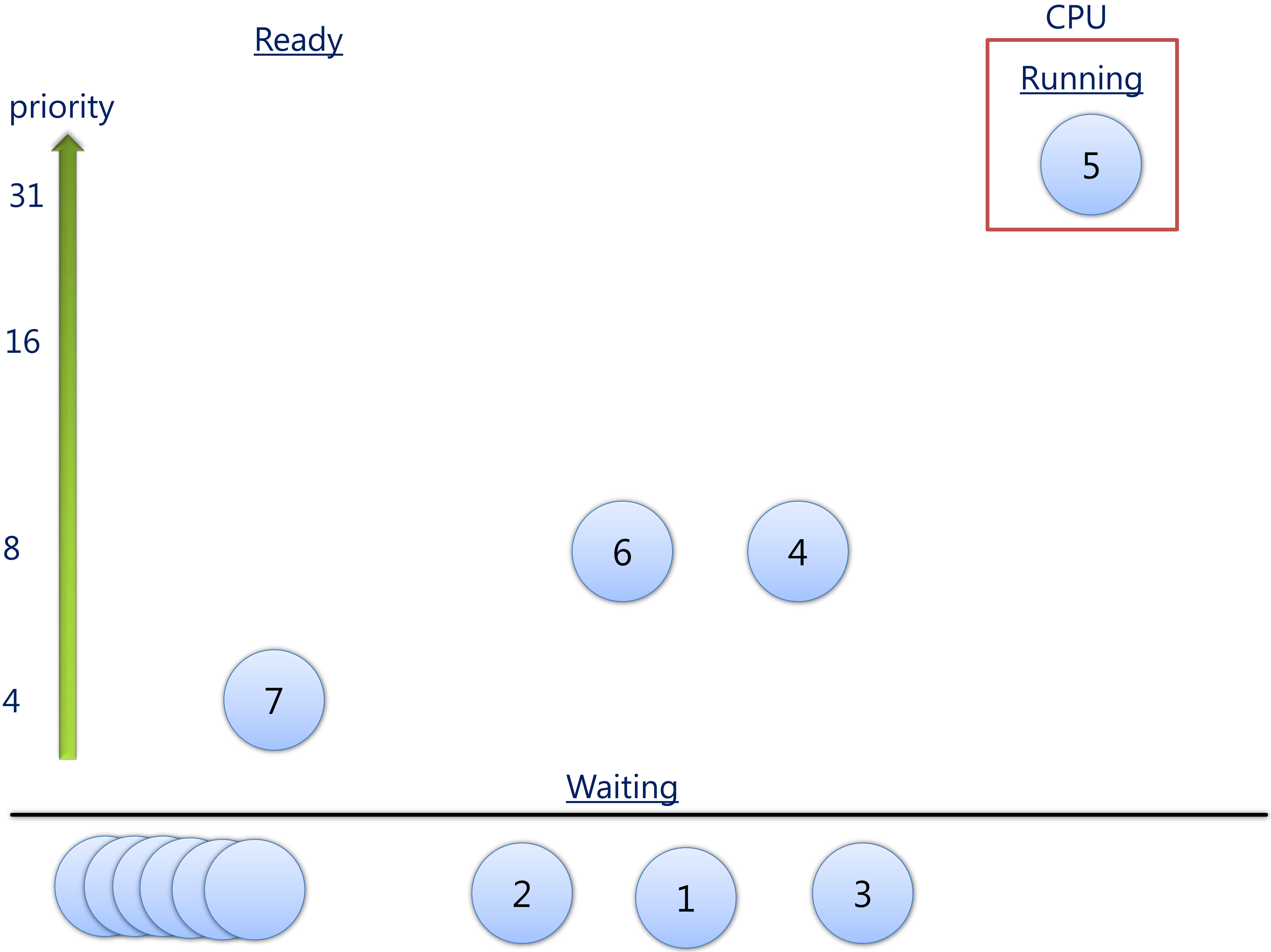
- **Thread priorities are between 1 and 31 (31 being the highest)**
- **Priority 0 is reserved for the zero page thread**
- **The Windows API mandates thread priority be based on a process priority class (base priority)**
- **A thread's priority can be changed around the base priority**
- **APIs (Win32)**
  - **SetPriorityClass** – changing process base priority
  - **SetThreadPriority** – change the thread priority offset from the parent's base priority
- **API (kernel)**
  - **KeSetPriorityThread** – change thread priority to some absolute value

Thread Priorities (Win32 View)



Demo

Thread Priorities



Ready

CPU

Running

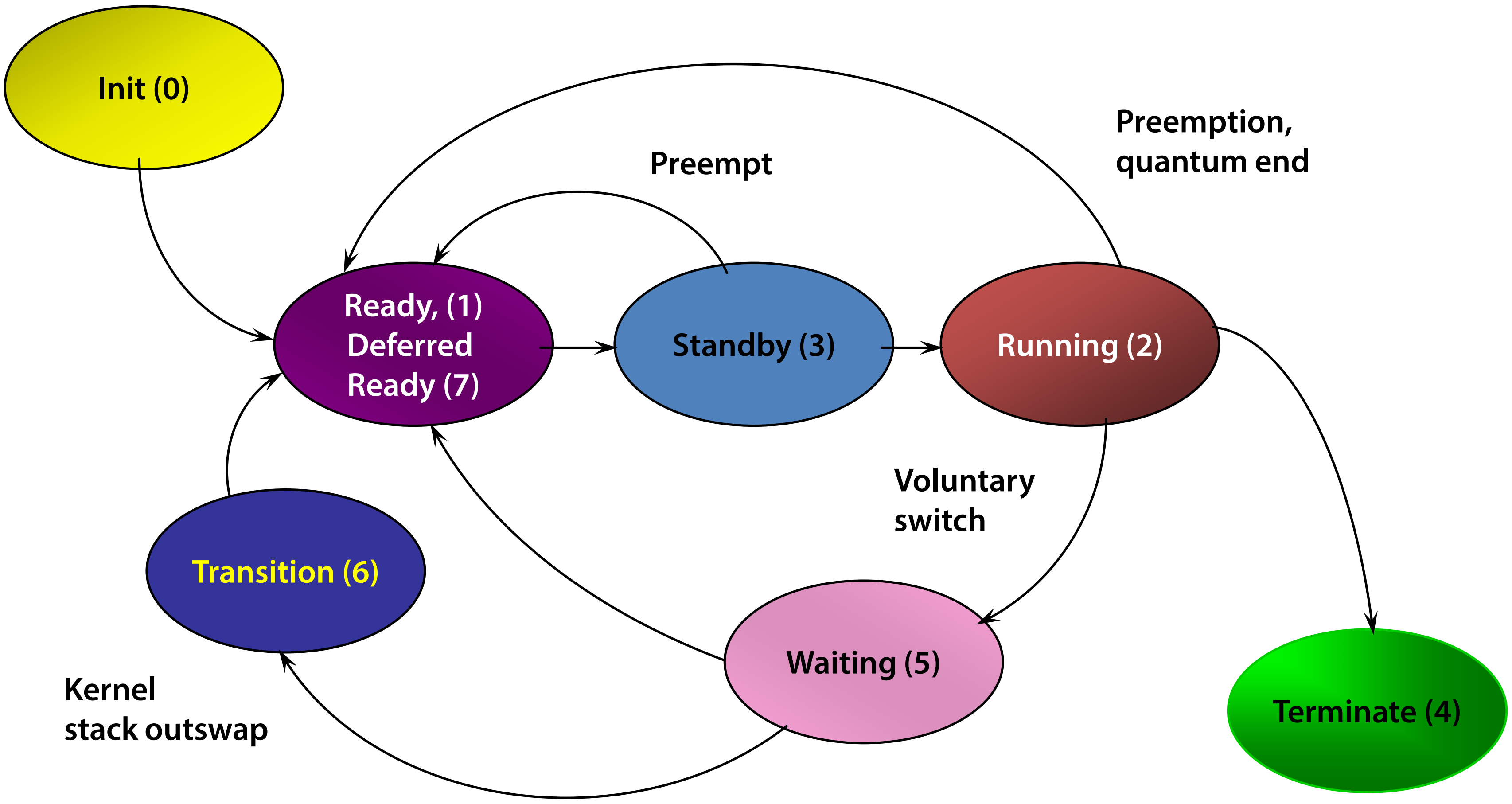
priority

Waiting

Thread Scheduling (single processor)

- **Priority based, preemptive, time-sliced**
  - Highest priority thread runs first
  - If time slice (quantum) elapses, and there is another thread with the same priority in the Ready state – it runs
    - Otherwise, the same thread runs again
  - If thread A runs, and thread B (with a higher priority) receives something it waited upon (message, kernel object signaling, etc.), thread A is preempted and thread B becomes the Running thread
- **Voluntary switch**
  - A thread entering a wait state is dropped from the scheduler's Ready list
- **Typical time slice is 30 msec on client, 180 msec on server**
- **On an MP system with n logical processors, n concurrent threads may be running**

Thread States



The Scheduler

- **Scheduling routines are called when scheduling events occur**
  - Interval Timer interrupts checks for quantum end and timed wait completion
  - I/O Completion calls
  - Changes in thread priority
  - Changing state of waitable object other threads are waiting on
  - Entering a wait on one or more objects
  - Entering Sleep

Demo

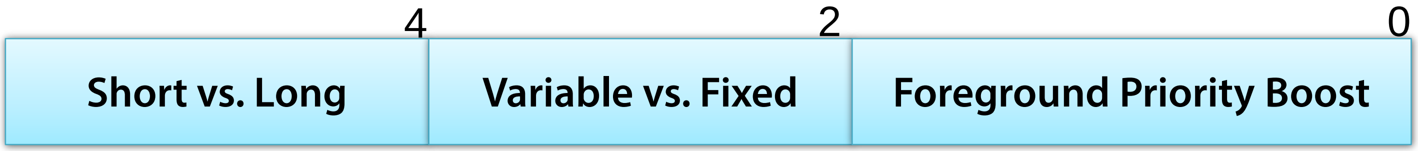
Thread Scheduling

The Quantum

- **Scheduler clock tick is typically**
  - 10 msec (uniprocessor)
  - 15 msec (multiprocessor)
- **Can determine with clockres.exe utility from SysInternals**
- **Default client quantum is 2 clock ticks**
- **Default server quantum is 12 clock ticks**
- **Quantum can be modified by using the registry or a Job**
- **Quantum boosting**
  - On a system configured for short, variable quantum
    - The foreground process gets triple quantum
    - For any process with a priority class above Idle

Quantum Control

- **Registry key: HKLM\SYSTEM\CCS\Control\PriorityControl**
- **Value: Win32PrioritySeparation**



4 2 0

Short vs. Long | Variable vs. Fixed | Foreground Priority Boost

- **Short vs. Long**
  - 1=long, 2=short
  - 0, 3=deafult (long for Server, short for Client)
- **Variable vs. Fixed**
  - 1=boost priority of foreground process, 2=don’t boost
  - 0, 3=default (boost for Client, don’t boost for Server)
- **Foreground quantum boost**
  - Index into a table

	Short			Long		
Variable	6	12	18	12	24	36
Fixed	18	18	18	36	36	36

Demo

Thread Quantum