

Windows Internals for .NET Developers

Pavel Yosifovich

@zodiacon

zodiacon@live.com

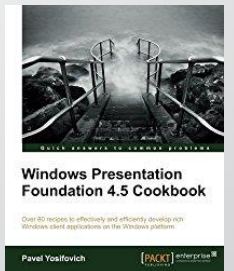
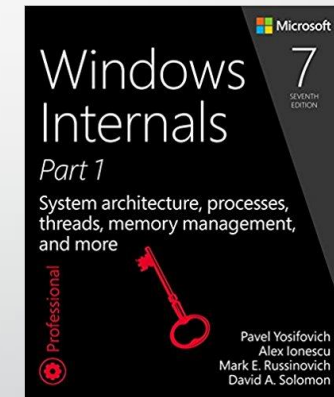


Agenda

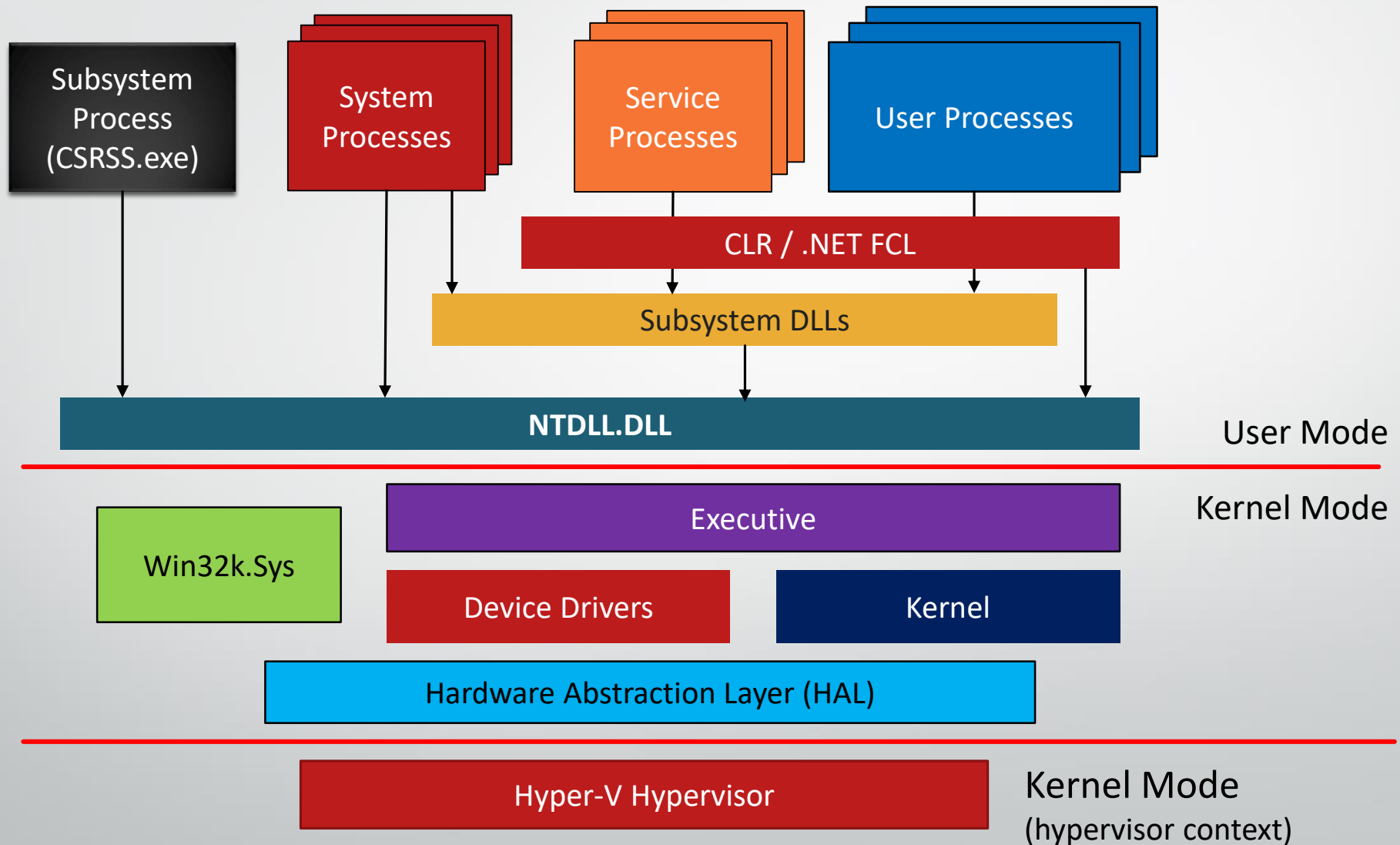
- Windows Architecture and .NET
- Jobs
- Threads
- Kernel Objects
- Windows Runtime
- Summary

About Me

- Developer, Trainer, Author and Speaker
- Co-author: Windows Internals 7th edition, Part 1 (2017)
 - Author: WPF Cookbook (2012), Mastering Windows 8 C++ App Development (2013)
- Pluralsight author
- Author of several open-source tools (<http://github.com/zodiacon>)
- Blogs: <http://blogs.microsoft.co.il/pavely>, <https://scorpiosoftware.net/>



Windows Architecture Overview




.NET Today

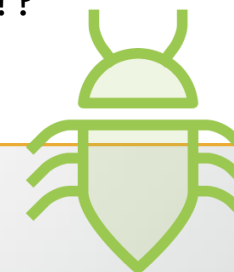
- Part of Windows
 - Fully supported
- User mode only
 - Kernel mode CLR/.NET possible (e.g. “Singularity”)
- Not a complete wrapper over the Windows API
 - Opportunities to extend and optimize
- What about .NET Core?

Windows Numeric Versions



- Windows NT 4 (4.0)
- Windows 2000 (5.0)
- Windows XP (5.1)
- Windows Server 2003, 2003 R2 (5.2)
- Windows Vista, Server 2008 (6.0)
- Windows 7, Server 2008 R2 (6.1) 
- Windows 8, Server 2012 (6.2)
- Windows 8.1, Server 2012 R2 (6.3)
- Windows 10, Server 2016 (10.0)

```
// get version...  
if(version.Major >= 5 && version.Minor >= 1) {  
    // XP or later: good to go!?  
}
```



By default, reported as 6.2

By default, reported as 6.2

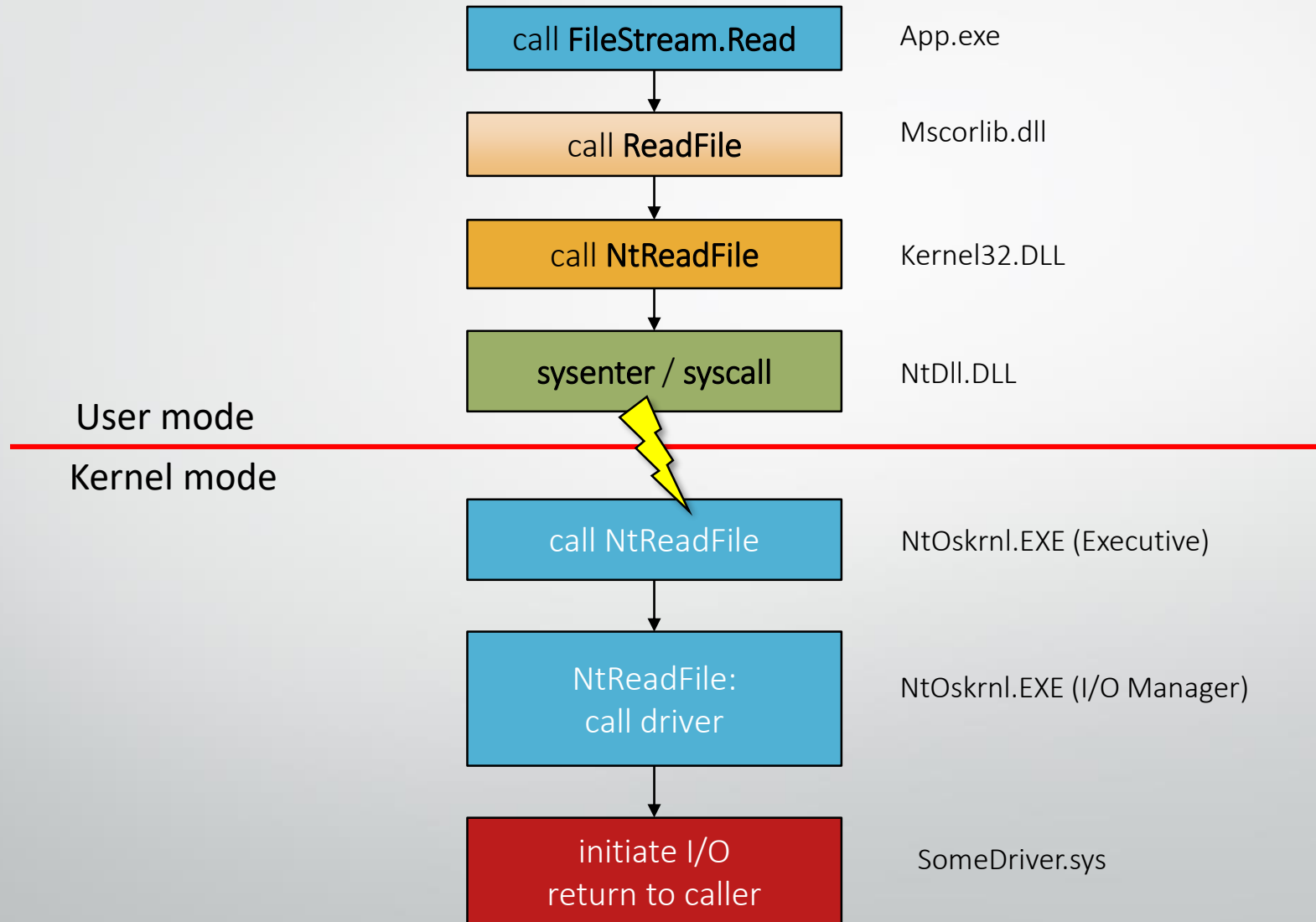
Windows Versions



Windows Subsystem APIs

- Windows API (“Win32”)
 - Classic C API from the first days of Windows NT
- COM based APIs
 - Especially in newer (Vista+) APIs
 - Examples: BITS, DirectX, WIC, DirectShow, Media Foundation, Task Host
- Windows Runtime (WinRT)
 - New unmanaged API available for Windows 8+
 - Built on top of an enhanced version of COM
- The Native (undocumented) API
 - Implemented by Ntdll.dll

Function Call Flow



Jobs

- A job is a kernel object that manages a set of one or more processes
- A job can impose limits on its processes
- Provides accounting information
- Can associate an I/O completion port with a job
- Once a process is assigned to a job, it cannot get out
- Process inside a job that creates a child process
 - Child process are added to the same job by default
 - Unless the **CREATE_BREAKAWAY_FROM_JOB** flag is specified in `CreateProcess` (and the job allows breaking out of it)

Job Limits Examples

- Maximum processes active in a job
- CPU
 - Per-job and per-process CPU time
 - CPU Affinity, process priority class
 - CPU rate control (Windows 8+)
- Memory
 - Minimum and maximum working set
 - Process commit maximum
- Network
 - Maximum bandwidth (Windows 10)
- UI
 - USER and GDI handles, clipboard access, exiting windows, desktop creation/switching

CPU Rate Control

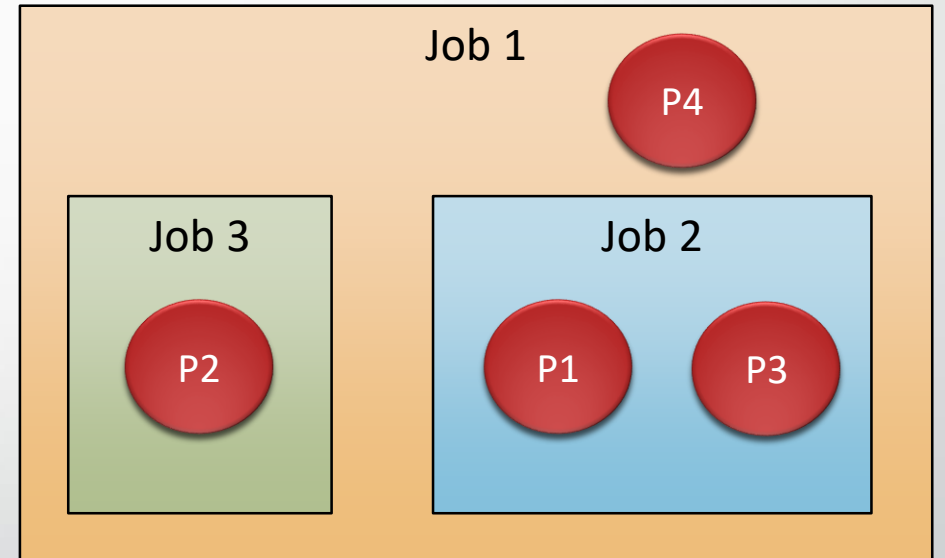


Nested Jobs

- Prior to Windows 8
 - A process could be assigned to a single job
 - Trying to assign a process to a second job just failed
- Windows 8 and later
 - A process can be part of more than one job
 - A job hierarchy is created (if possible)
 - Limits in child jobs can be more restrictive than its parents' jobs limits
 - But not vice versa
- The ability to nest jobs makes jobs much more useful

Nested Jobs Example

- Add P1 to Job 1
- Add P1 to Job 2
- Add P2 to Job 1
- Add P2 to Job 3
- Add P3 to Job 2
- Add P4 to Job 1

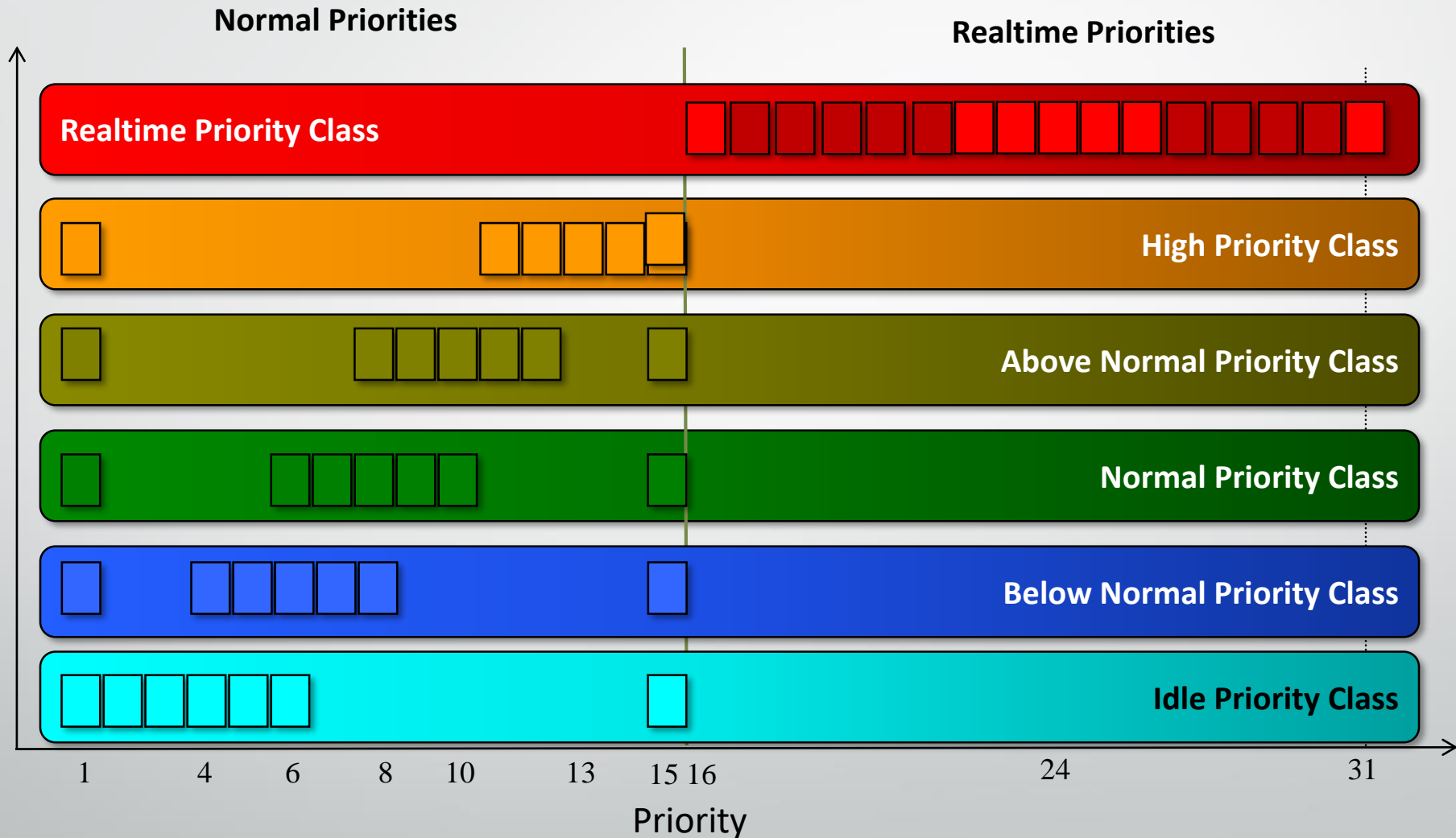


Jobs API



- **CreateJobObject**
 - Create a job object or open a named job object
- **OpenJobObject**
 - Open a named job object
- **AssignProcessToJobObject**
 - Add a process to a job
- **Set(Query)InformationJobObject**
 - Set various limits on the job (query job information)
- **TerminateJobObject**
 - Terminate all processes belonging to the job

Thread Priorities (Windows Subsystem View)



Manipulating Priorities

- Process
 - Win32: `SetPriorityClass`
 - .NET: `Process.PriorityClass` property
- Thread
 - Win32: `SetThreadPriority`
 - .NET: `Thread.Priority` property
 - Only the middle five levels are exposed

Thread and Process Priorities

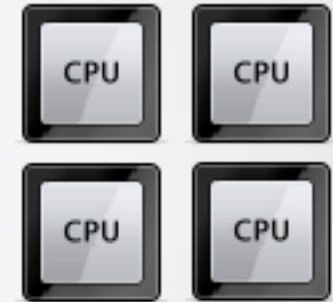
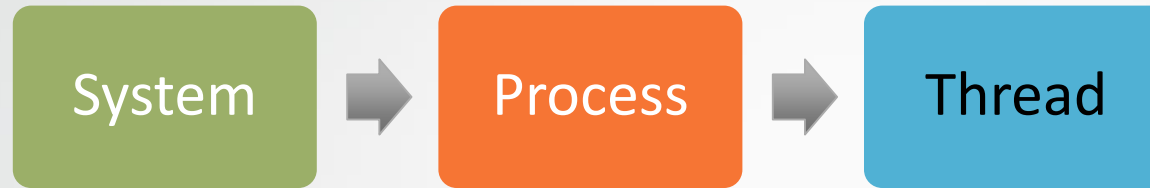


Processor Affinity

- Soft affinity
 - Ideal processor
- Hard affinity
 - Always respected
- CPU Sets (Windows 10)
- .NET indirectly supports soft and hard affinity
 - But not CPU sets

CPU Sets

- A different kind of affinity



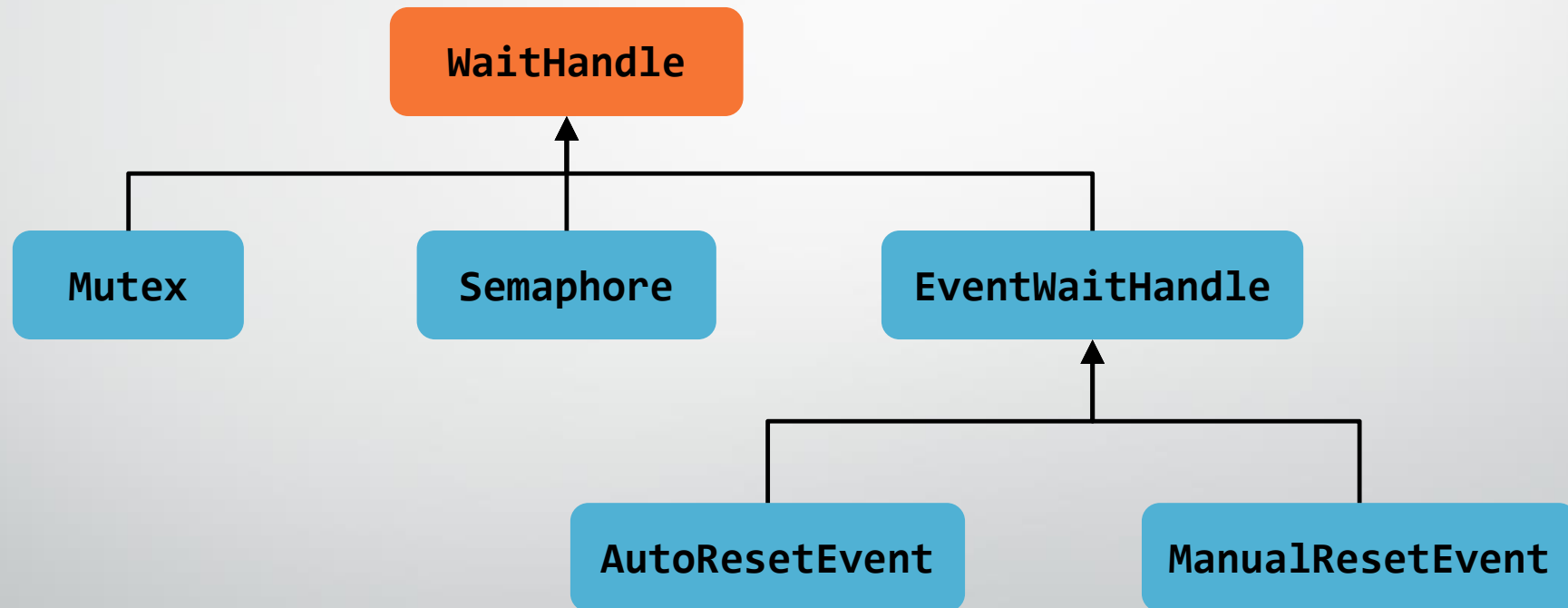
- The Windows API provides control over process and thread CPU sets
 - `SetProcessDefaultCpuSets`,
`SetThreadSelectedCpuSets`
- System CPU Sets control is available through an undocumented `NtSetSystemInformation` call

CPU Sets



Waitable Kernel Objects

- Also known as “Dispatcher Objects”



Wrapping Kernel Objects



Kernel Objects Namespace

- By default, all named objects created in *\Sessions\x\BaseNamedObjects*
- Session 0 objects created in *\BaseNamedObjects*
 - Can create/open named objects in session 0 with “Global\” prefix
 - Cannot be used by UWP processes
- Private namespaces

Object Names and Namespaces



The Windows Runtime and .NET

- The Windows Runtime
 - Unmanaged API primarily for use in UWP apps
- C# projection is excellent
- Parts of the Windows Runtime are usable from classic (desktop) apps

WinRT in Classic Apps



Summary

- The .NET framework is rich in functionality
- However, the Windows platform has more than .NET can cover
- Interop is sometimes key
- Experiment!