The Windows Operating System

Zoltán Micskei

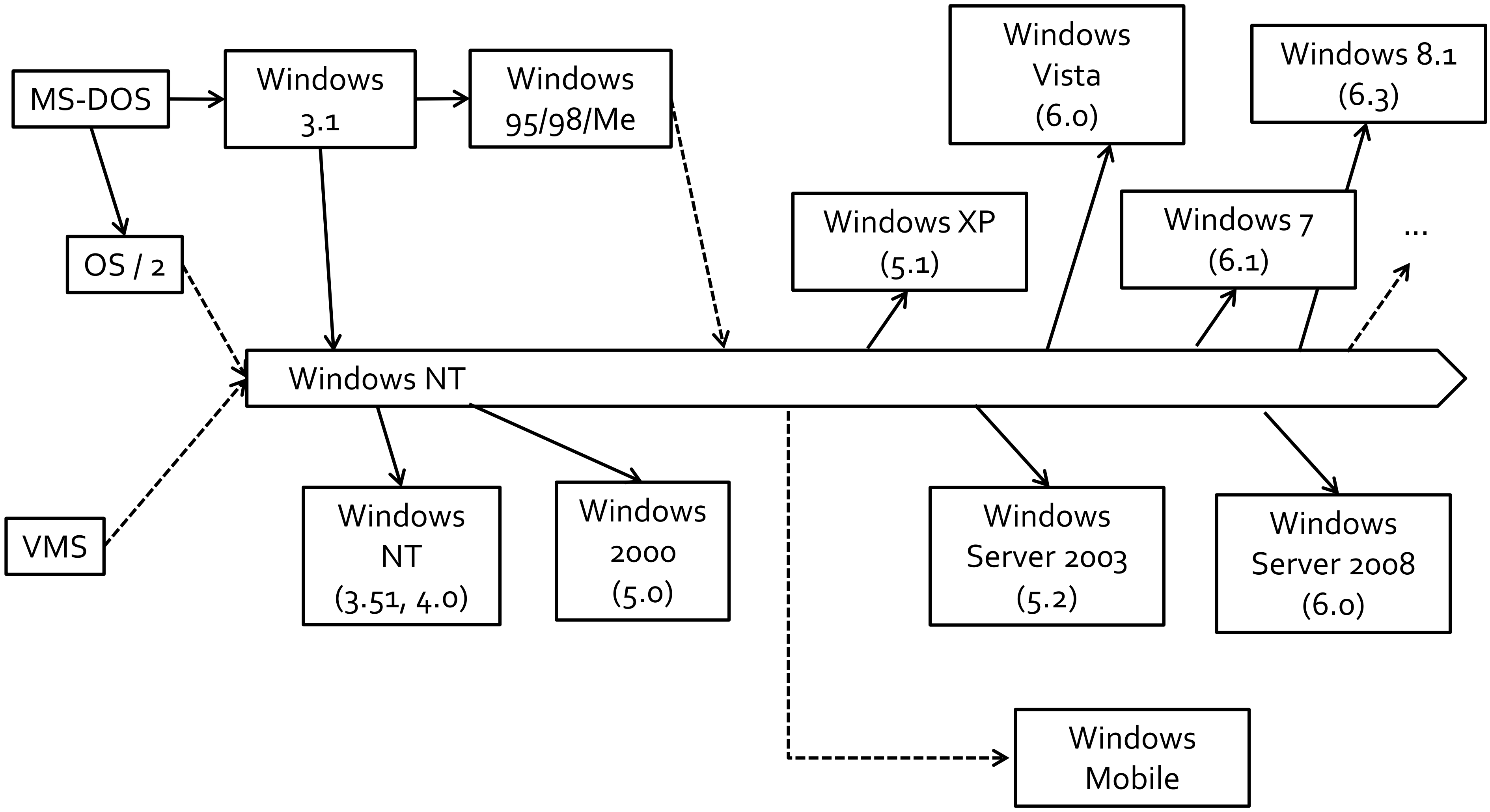
http://mit.bme.hu/~micskeiz

FAULT TOLERANT SYSTEMS RESEARCH GROUP

Copyright Notice

- These materials are part of the *Windows Operating System Internals Curriculum Development Kit,* developed by David A. Solomon and Mark E. Russinovich with Andreas Polze
- Microsoft has licensed these materials from David Solomon Expert Seminars, Inc. for distribution to academic organizations solely for use in academic environments (and not for commercial use)
- http://www.academicresourcecenter.net/curriculum/pfv.aspx?ID=6191
- © 2000-2005 David A. Solomon and Mark Russinovich

Windows family



The history of Windows NT

- New operating system in 1988
  - Originally: OS/2 3.0
  - Change: Successor of Windows 3.0
- Creator:
  - Dave Cutler (creator of VMS at Digital)
- Windows NT name
  - NT = New Technology
  - Windows NT = WNT = ?

Releases

- Produ r
  - Inc y 5-6 times)

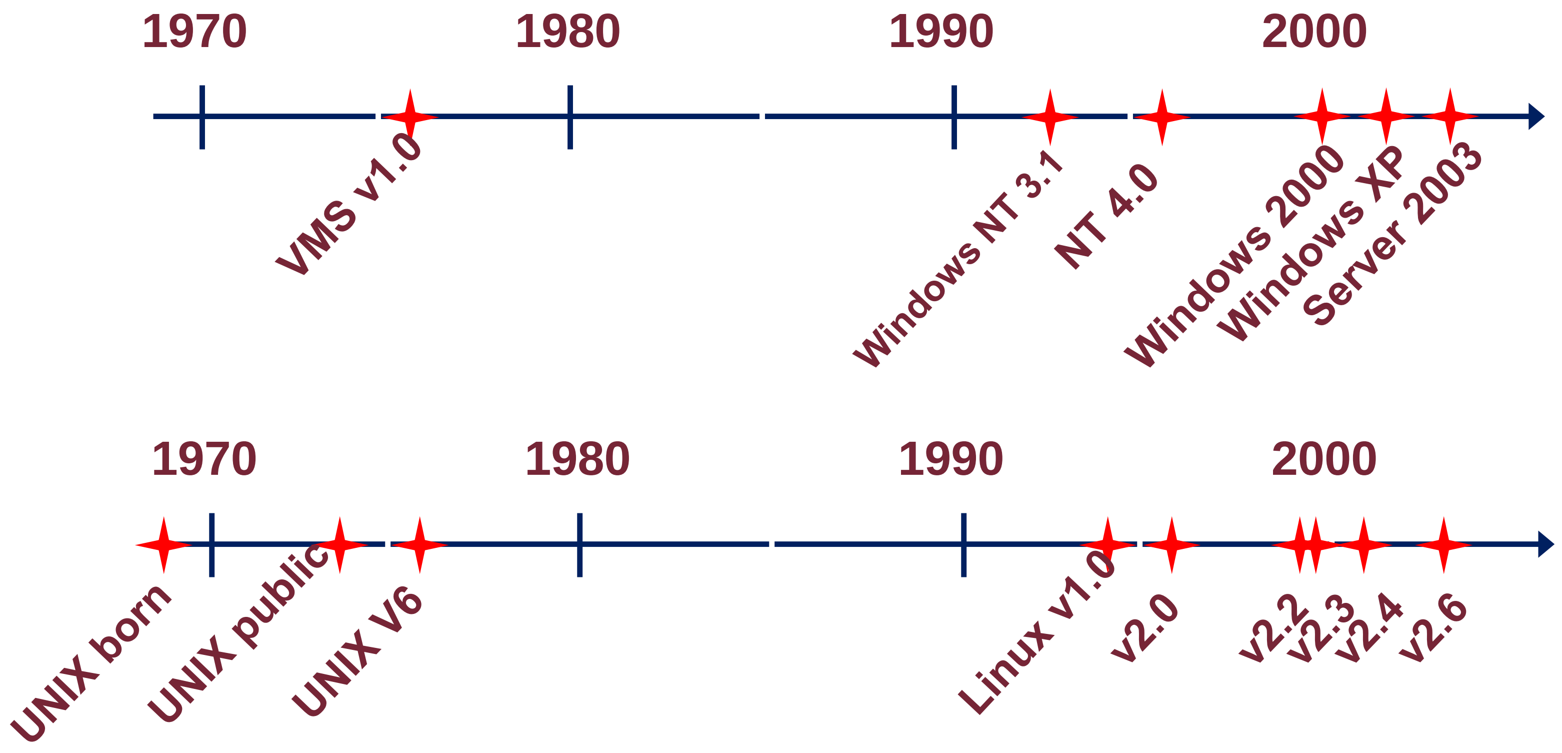
DEMO
cmd.exe

- 6 developers at the beginning
- 200 dev, 140 testers at end
- 6M LOC
- Whole compiling: 5 hours

Build#	V	Date
297	PDC d eloper release	Jul 1992
511	NT 3.1	Jul 1993
807	NT 3.5	4
1057	NT 3.51	5
1381	NT 4.0	6
2195	Windows 2000 (NT 5.0)	9
2600	Windows XP (NT 5.1)	Aug 2001
3790	Windows Server 2003 (NT 5.2)	Mar 2003
6000	Windows Vista RTM	Nov 2006
9600	Windows 8.1 RTM	Aug 2013

- 1400 dev, 1700 testers
- 29M LOC
- 50 GB source code
- Compiling: 8 hours
- Stress test: 1000 machines

Windows and Linux



The Structure of Windows

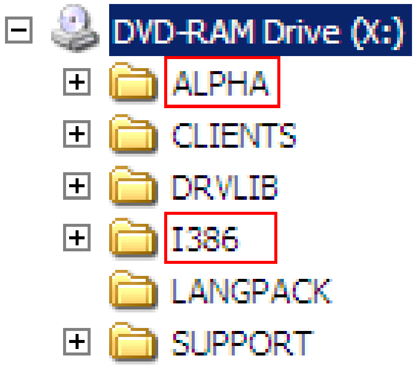
Questions

What does smss.exe do on my machine??

What is WoW?

Design goals

- Portability
  - Multiple processor architectures:
    - Originally: Intel x86, MIPS, Alpha, PowerPC
    - Windows XP: Intel x86
    - Windows Server 2003: x86, x64, IA64 (Itanium)
    - Windows 8: x86, x64, ARM (?)
  - HW specific part separated
  - Kernel: written in C



Design goals

- Portability
- Extensibility
  - Modular design
  - Well-defined interfaces
  - Unicode (even in kernel)

Design goals

- Portability
- Extensibility
- Reliability
  - Windows 3.0: shared address space
  - Security standards

Design goals

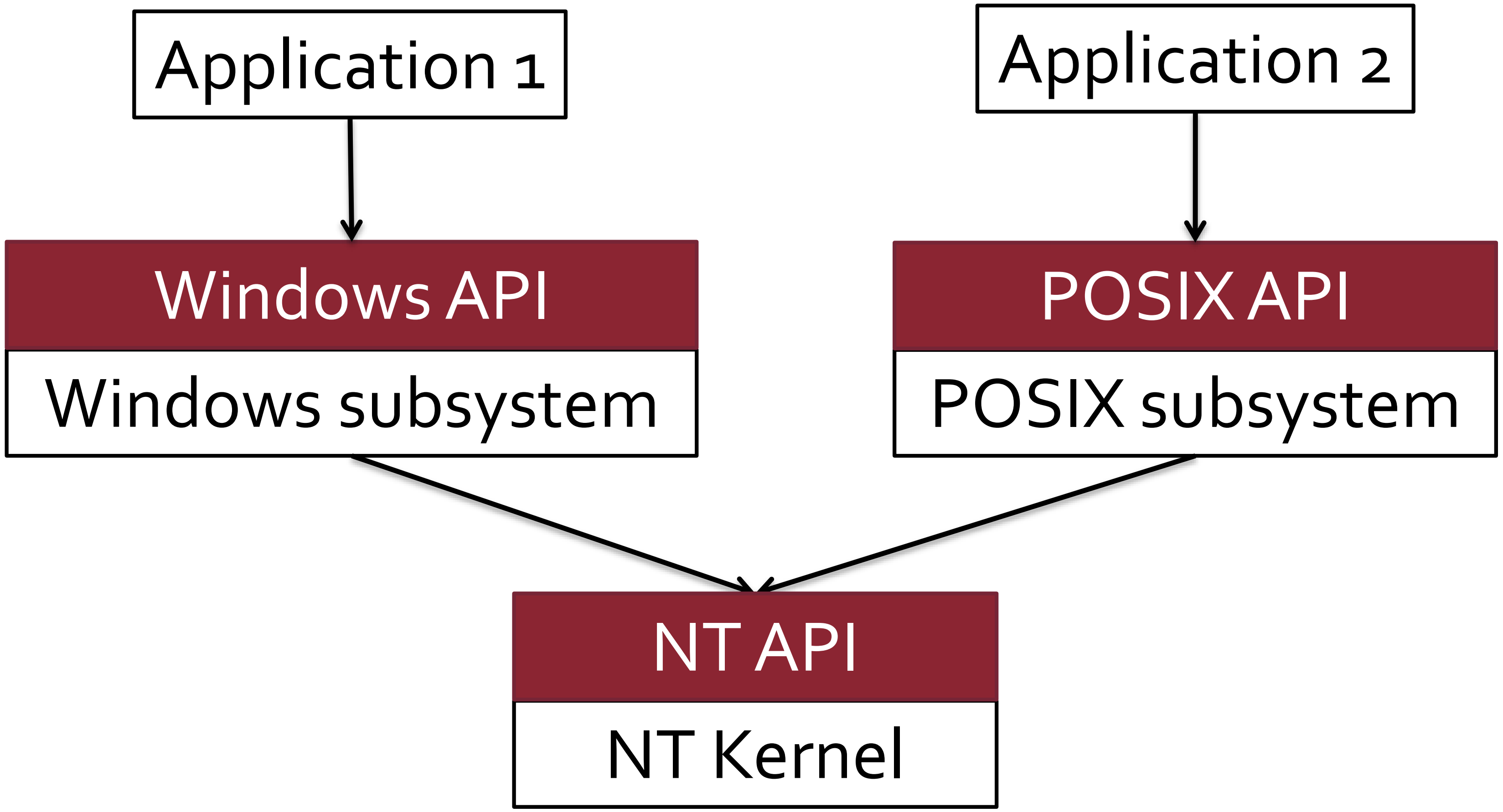
- Portability
- Extensibility
- Reliability
- Performance
  - 32 bit, preemptive, ***multi-threaded***, ***reentrant***
  - Symmetric Multiprocessing (SMP)
  - Asynchronous I/O
  - Optimized for client-server applications

Design goals

- Portability
- Extensibility
- Reliability
- Performance
- Compatibility, support for
  - DOS and 16 bit Windows API
  - POSIX
  - OS/2

Multiple personalities

- How to support Win32, POSIX and OS/2 API?
- Solution: environment subsystem

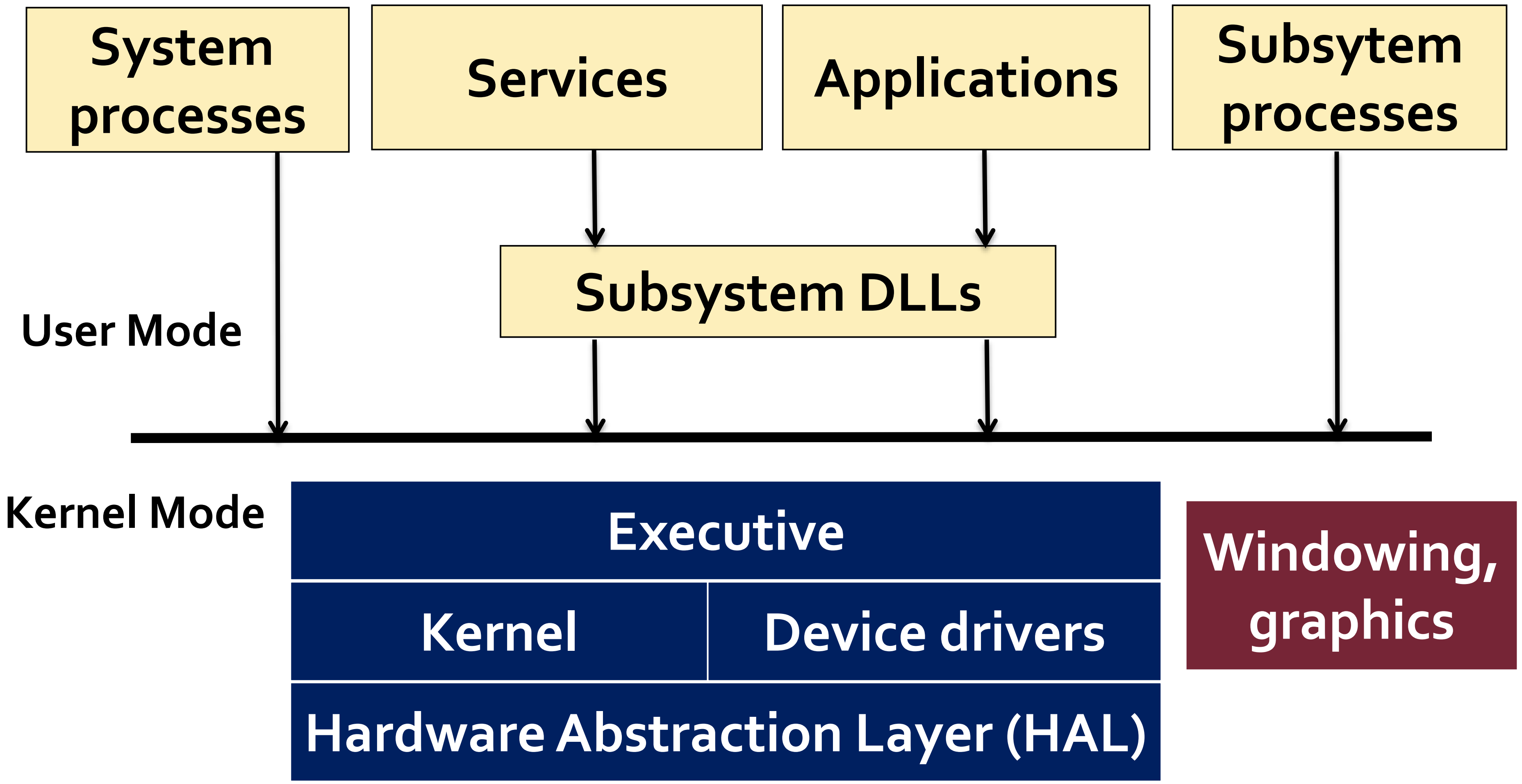


DEMO

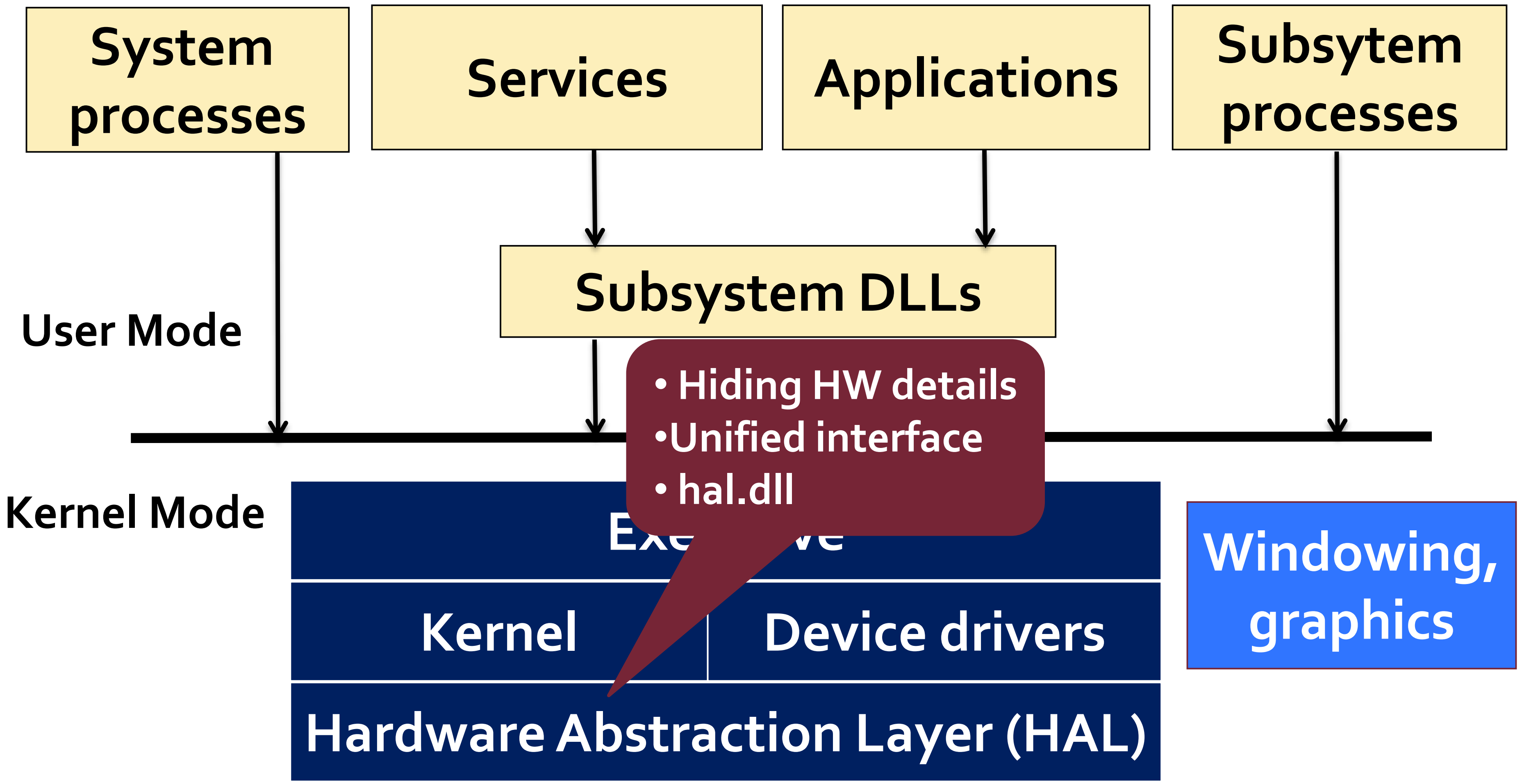
Exetype.exe

- Which subsytem do they belong?
  - cmd.exe
  - notepad.exe
  - smss.exe

Simplified architecture

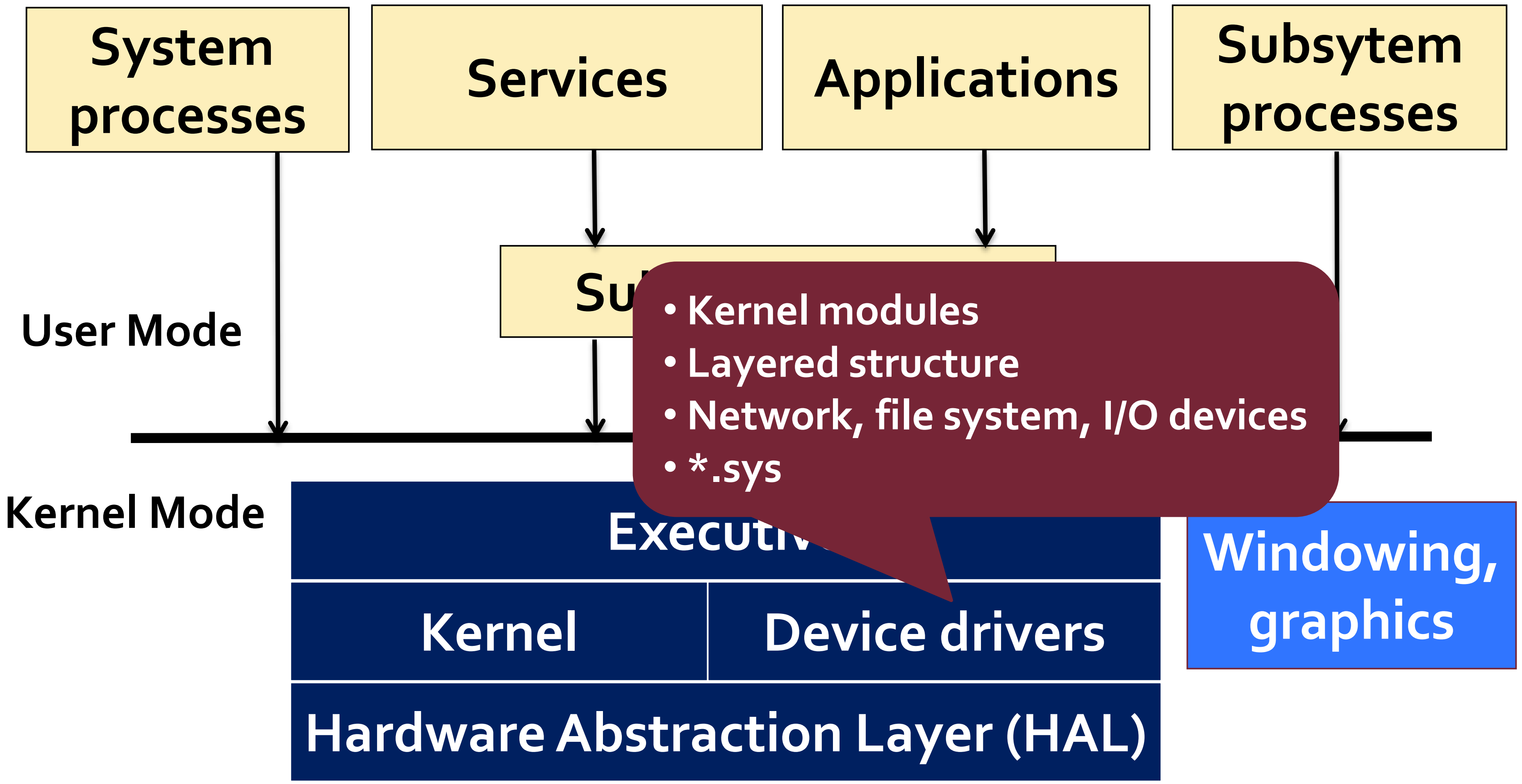


Simplified architecture



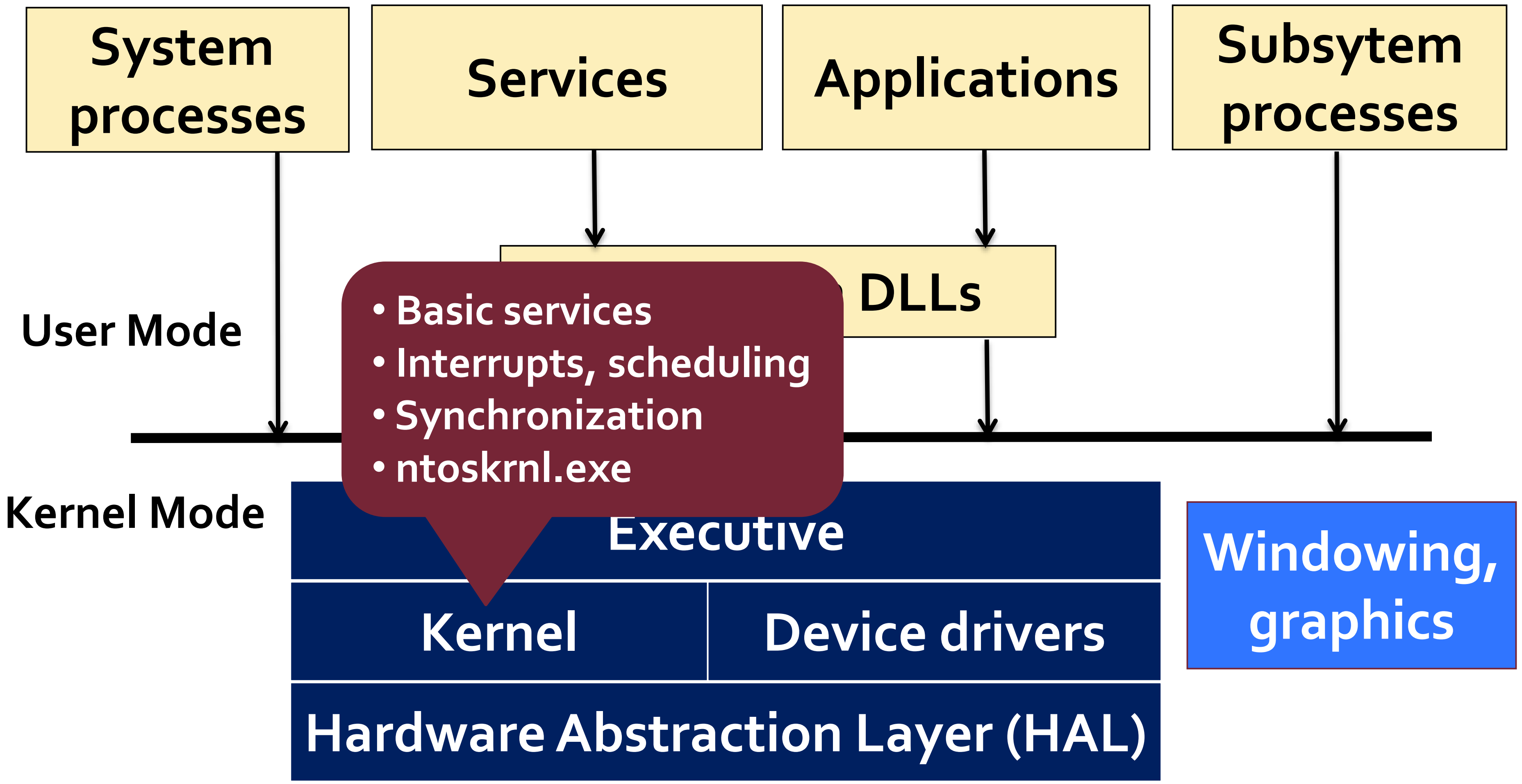
- Hiding HW details
- Unified interface
- hal.dll

Simplified architecture



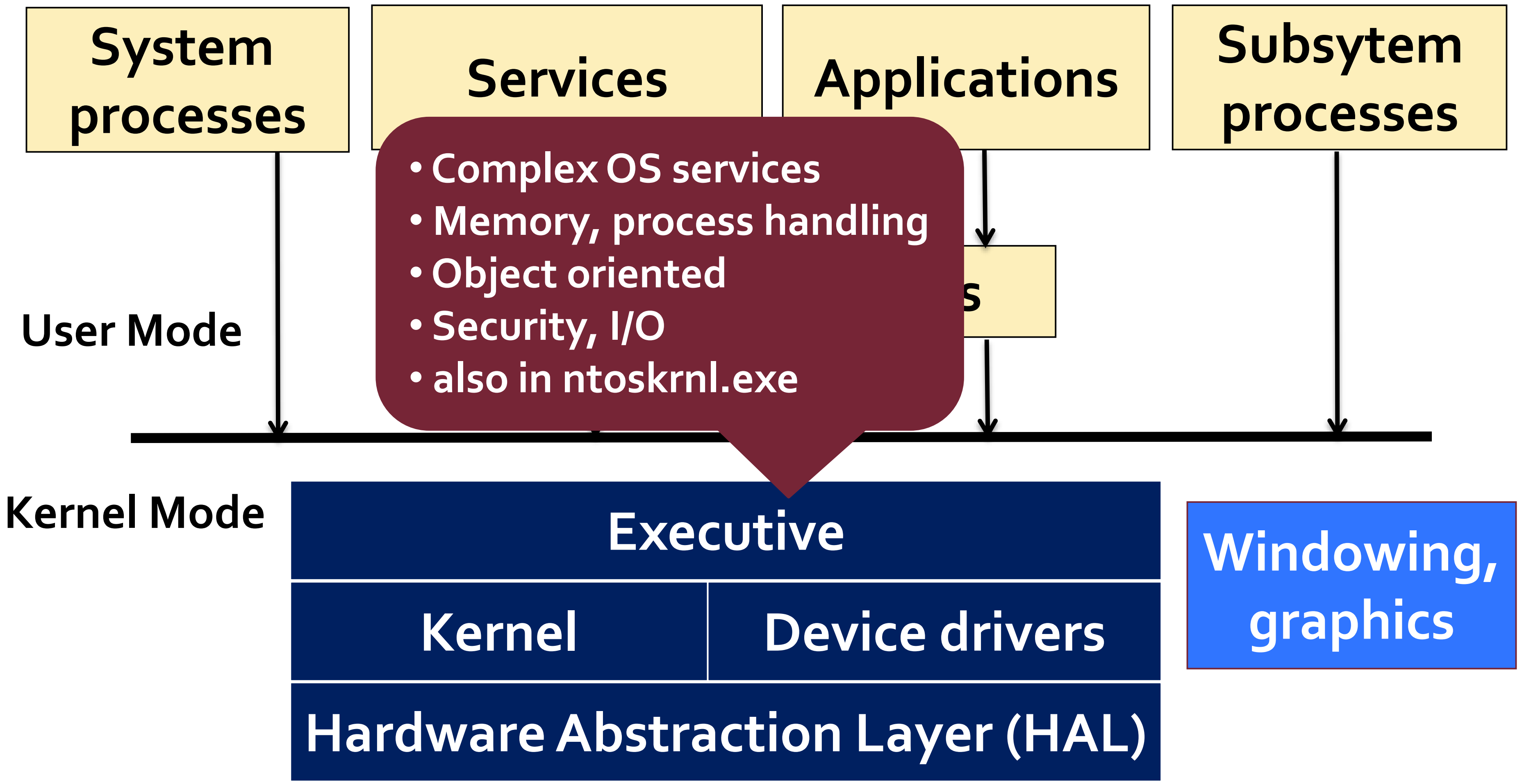
- Kernel modules
- Layered structure
- Network, file system, I/O devices
- *.sys

Simplified architecture



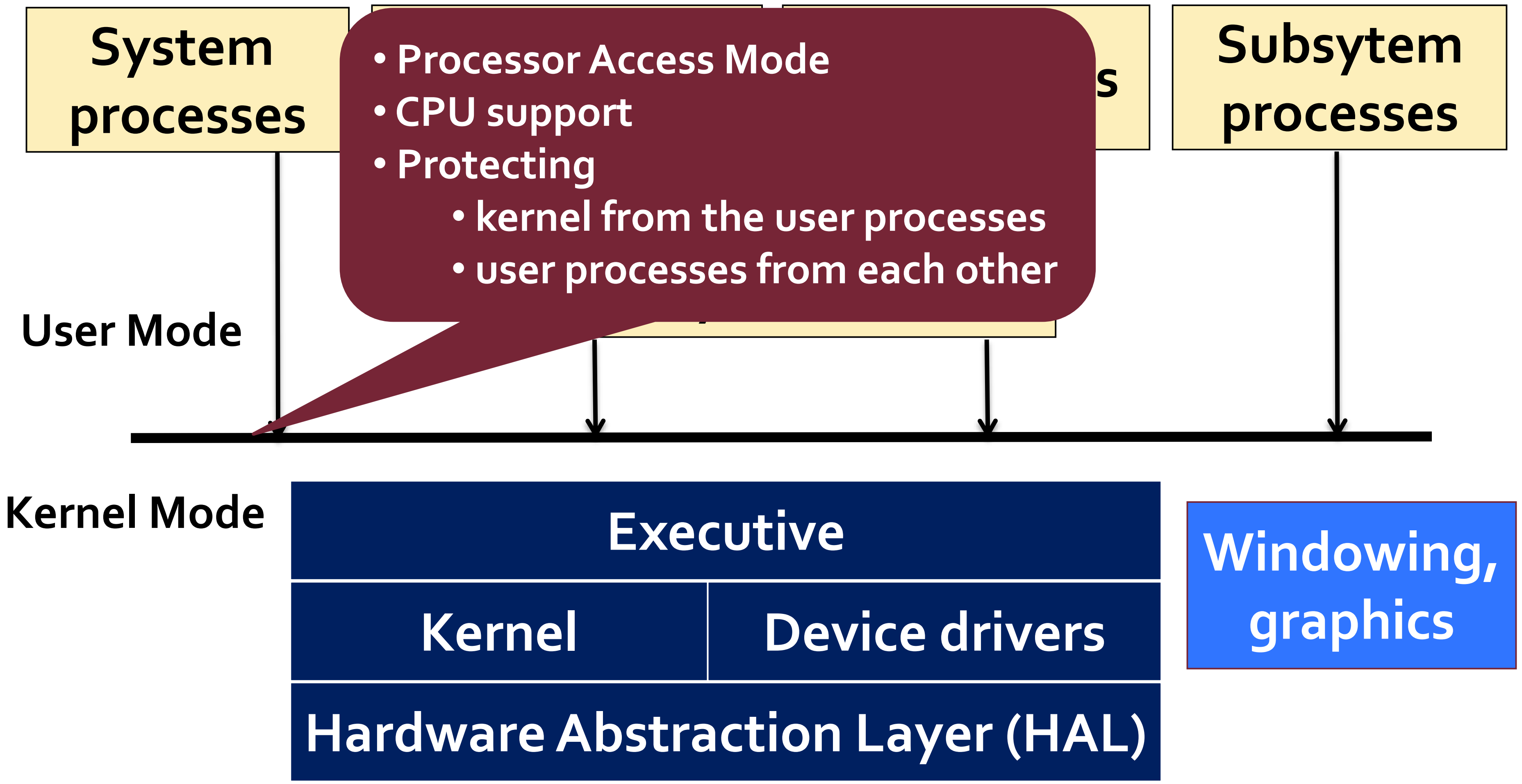
- Basic services
- Interrupts, scheduling
- Synchronization
- ntoskrnl.exe

Simplified architecture



- Complex OS services
- Memory, process handling
- Object oriented
- Security, I/O
- also in ntoskrnl.exe

Simplified architecture

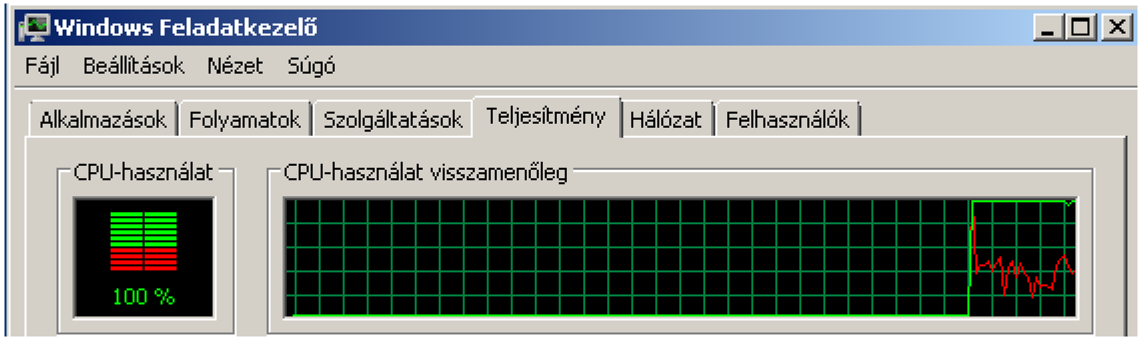


- Processor Access Mode
- CPU support
- Protecting
  - kernel from the user processes
  - user processes from each other

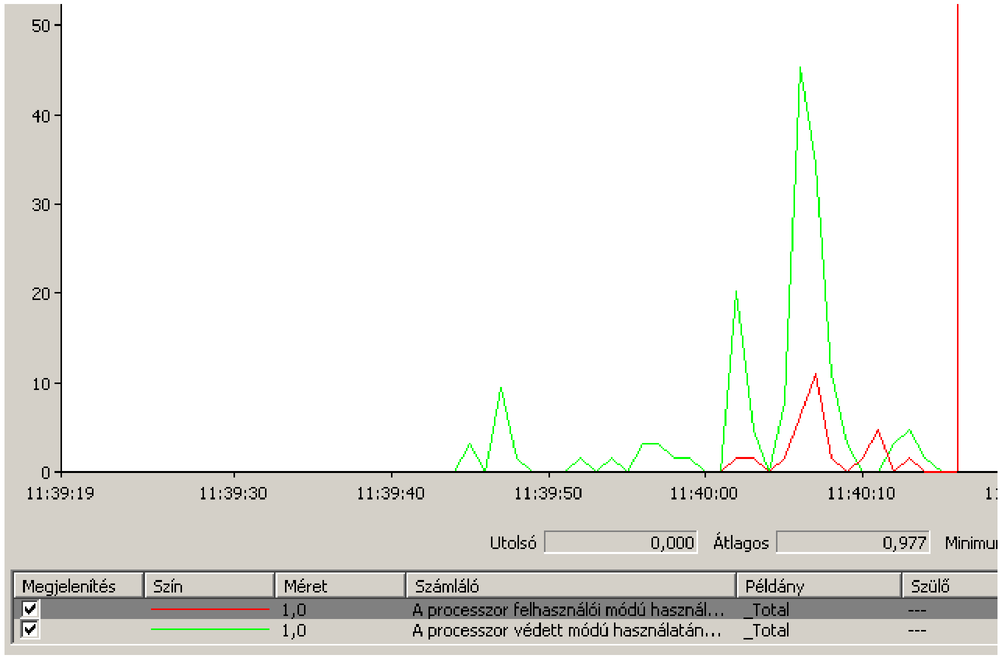
DEMO

Time spent in user and kernel mode

- Task Manager


Windows Feladatkezelő
Fájl Beállítások Nézet Súgó
Alkalmazások Folyamatok Szolgáltatások Teljesítmény Hálózat Felhasználók
CPU-használat
100 %
CPU-használat visszamenőleg

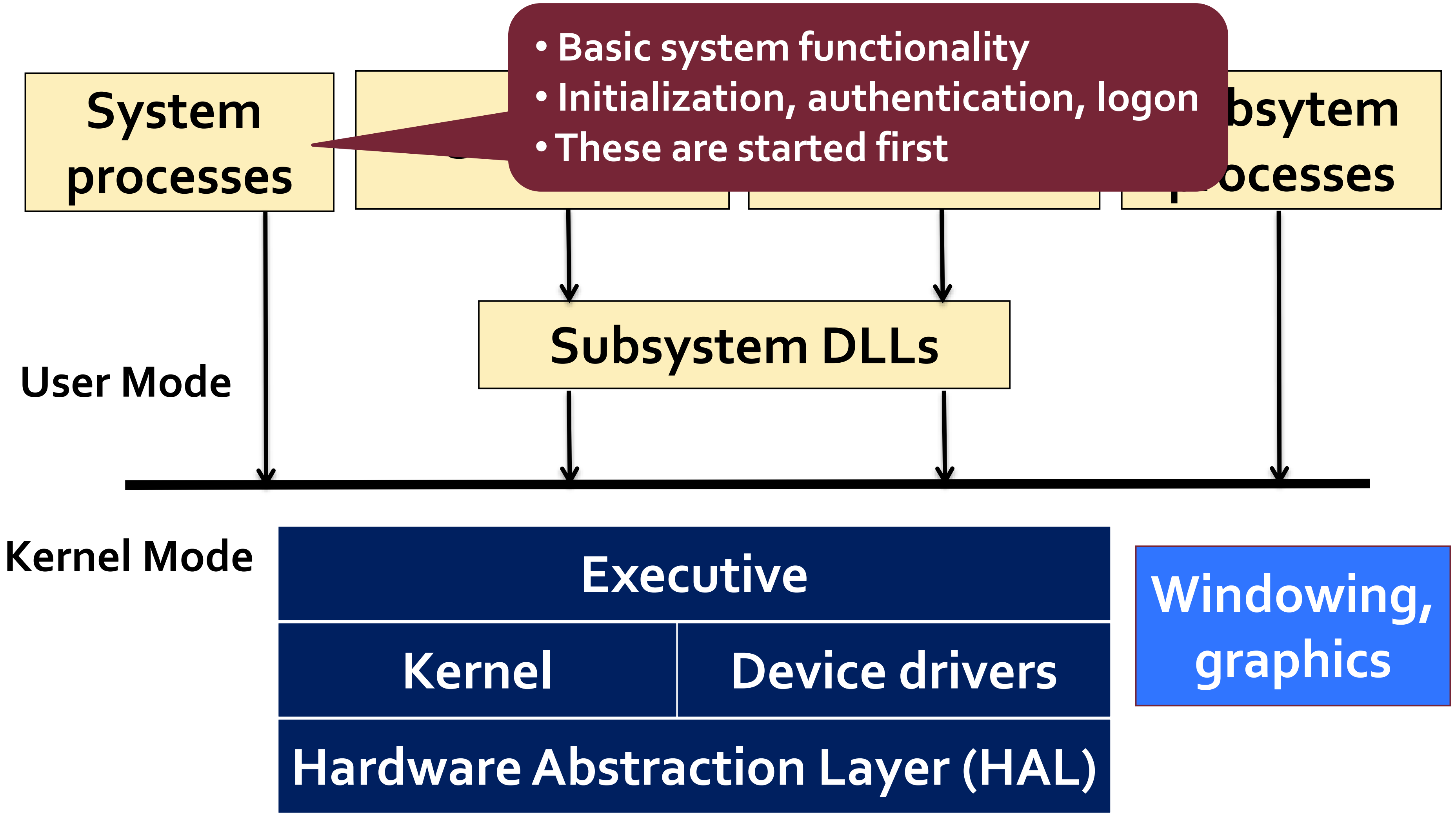
- Performance counters



Utolsó 0,000 Átlagos 0,977 Minimu

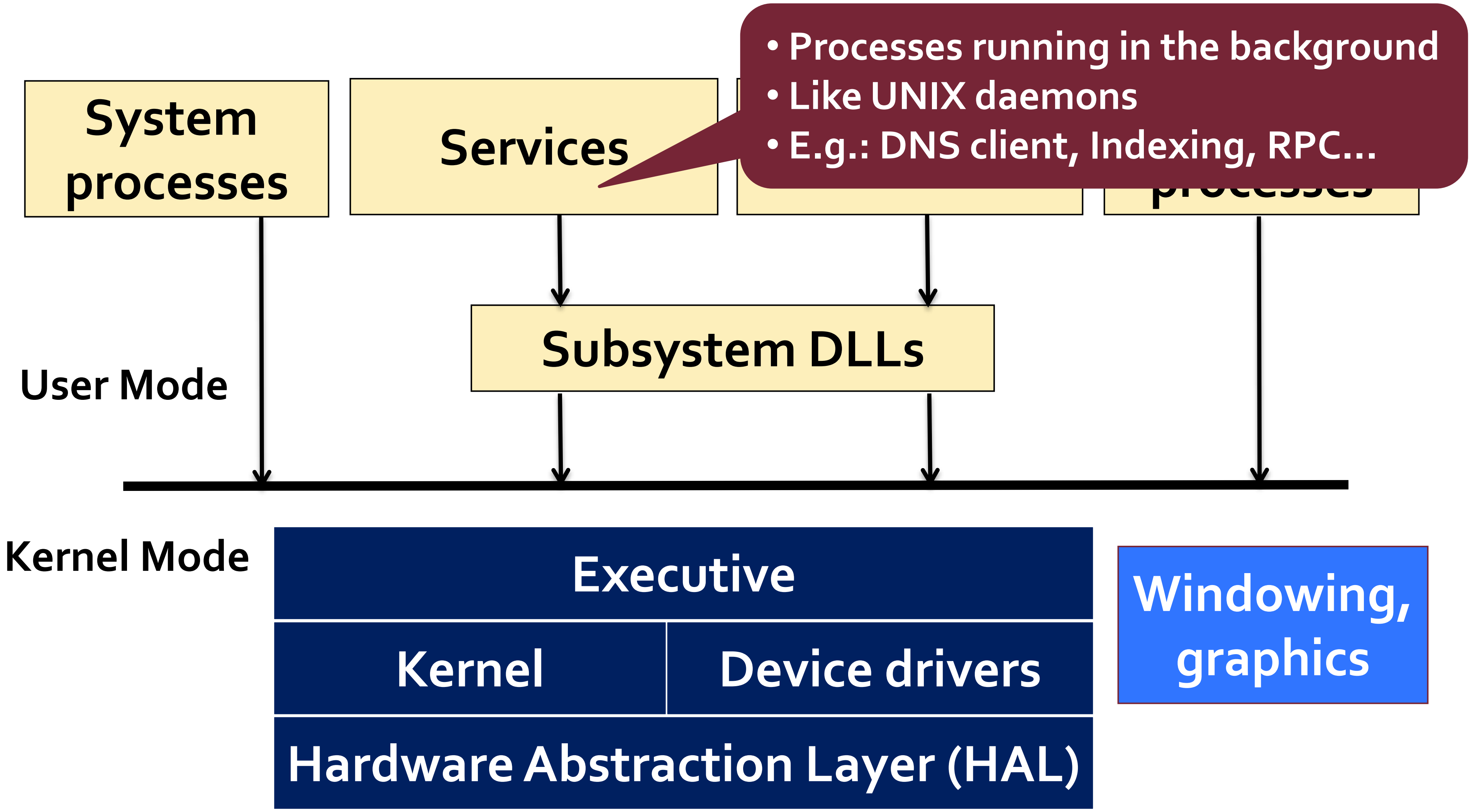
Megjelenítés	Szín	Méret	Számláló	Példány	Szülő
✓	🟥	1,0	A processzor felhasználói módú használ...	_Total	---
✓	🟩	1,0	A processzor védett módú használatán...	_Total	---

Simplified architecture



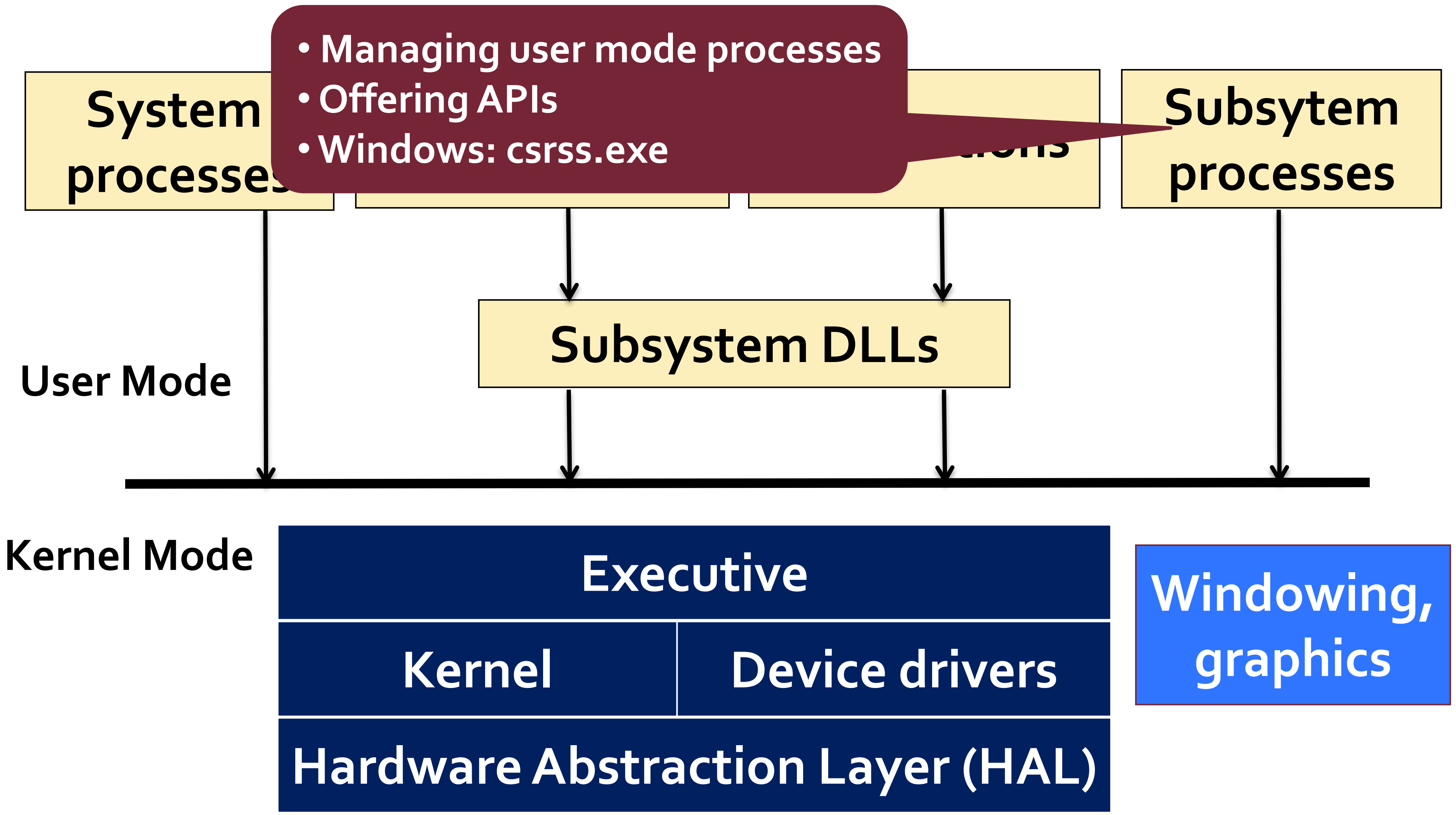
- Basic system functionality
- Initialization, authentication, logon
- These are started first

Simplified architecture



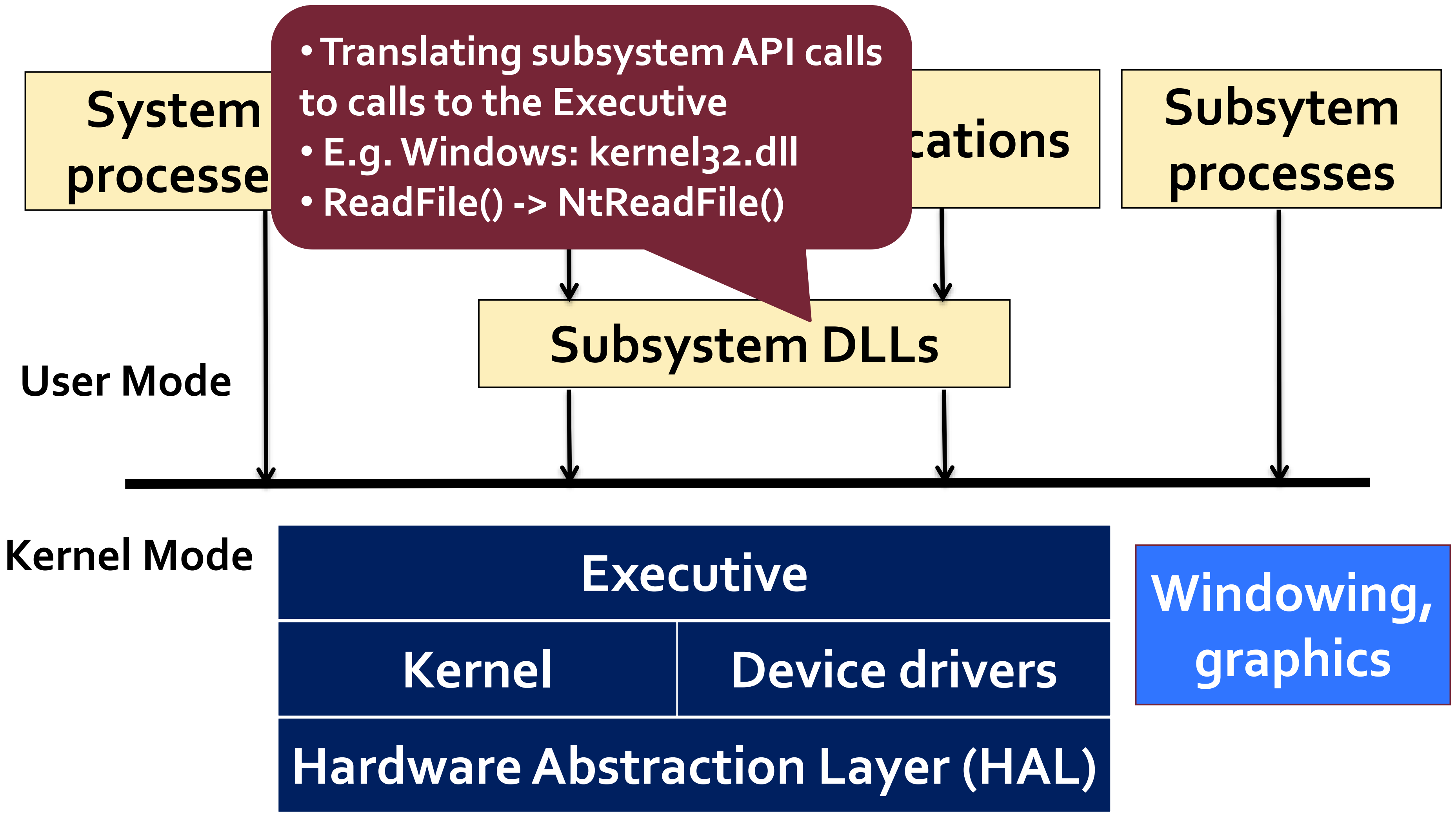
- Processes running in the background
- Like UNIX daemons
- E.g.: DNS client, Indexing, RPC…

Simplified architecture



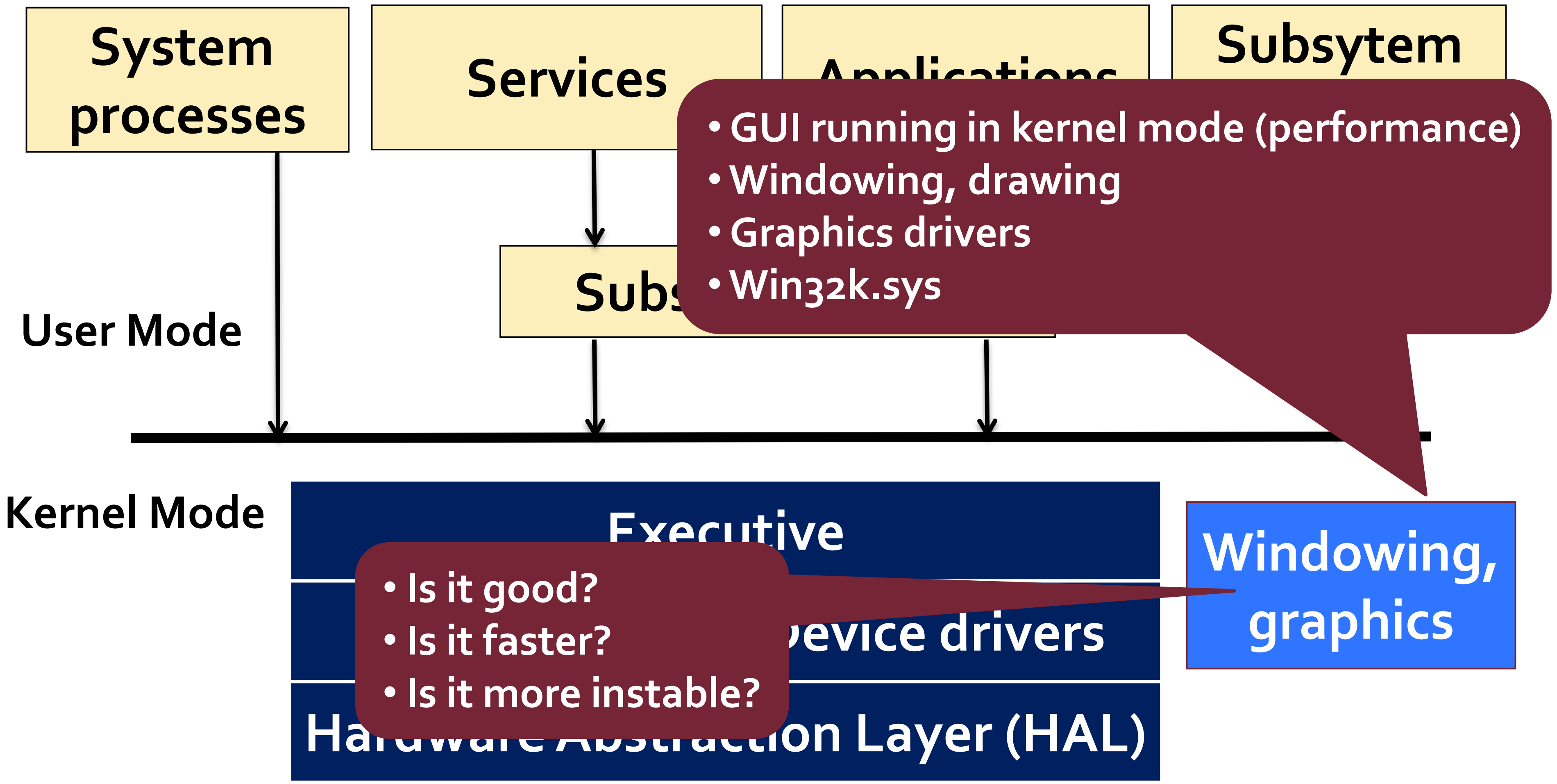
- Managing user mode processes
- Offering APIs
- Windows: csrss.exe

Simplified architecture



- Translating subsystem API calls to calls to the Executive
- E.g. Windows: kernel32.dll
- ReadFile() -> NtReadFile()

Simplified architecture



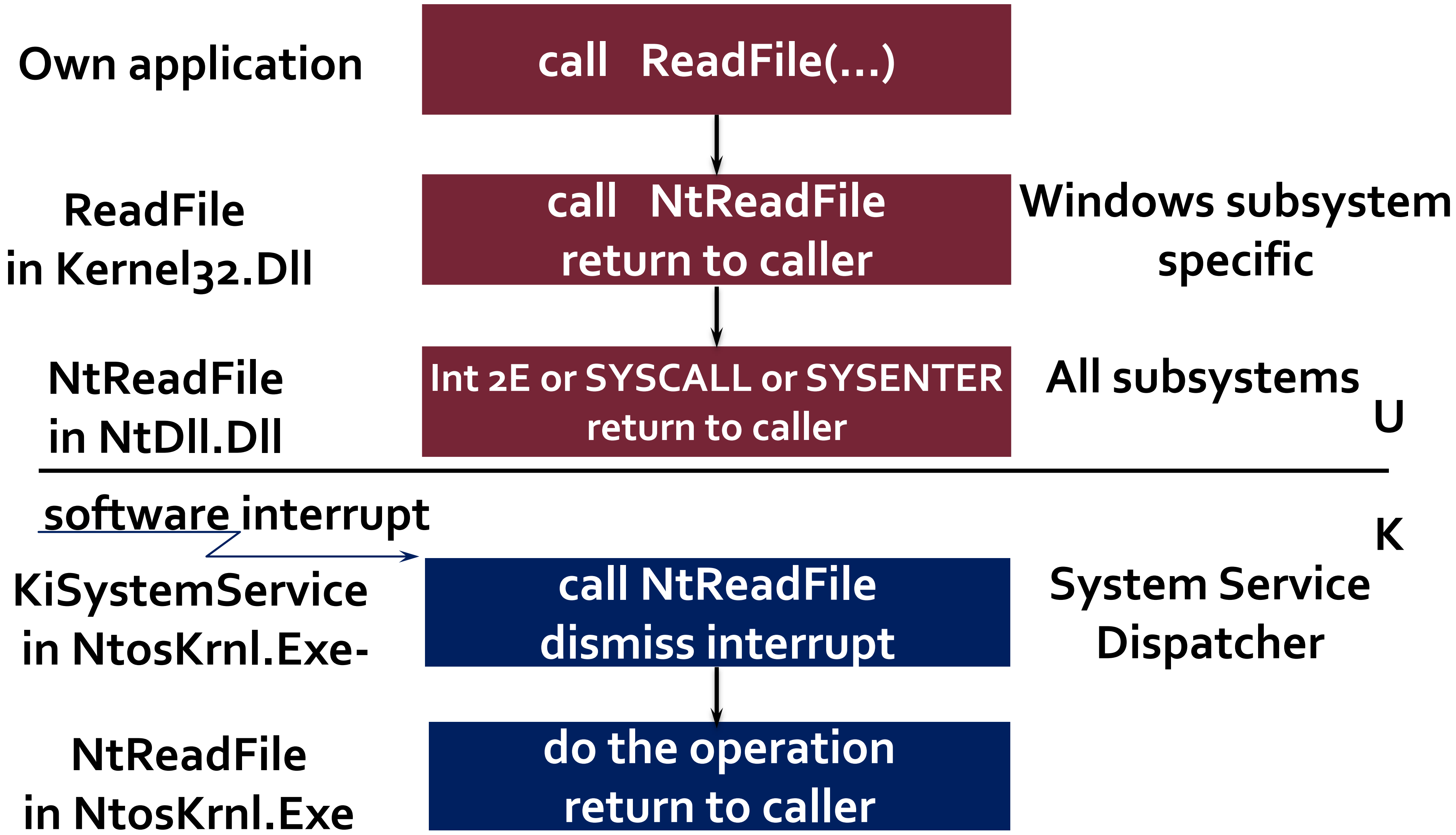
- GUI running in kernel mode (performance)
- Windowing, drawing
- Graphics drivers
- Win32k.sys

- Is it good?
- Is it faster?
- Is it more instable?

DEMO

- Documented kernel calls in the Windows DDK
- Documented Windows API calls in the Windows SDK
- List of services

Calling a Windows Kernel functions


Own application
call ReadFile(…)
ReadFile in Kernel32.Dll
call NtReadFile return to caller
Windows subsystem specific
NtReadFile in NtDll.Dll
Int 2E or SYSCALL or SYSENTER return to caller
All subsystems
U
software interrupt
K
KiSystemService in NtosKrnl.Exe-
call NtReadFile dismiss interrupt
System Service Dispatcher
NtReadFile in NtosKrnl.Exe
do the operation return to caller

DEMO

Tracing calls:

application → kernel32.dll → ntdll.dll

- Dependency walker
- WinDbg debugger

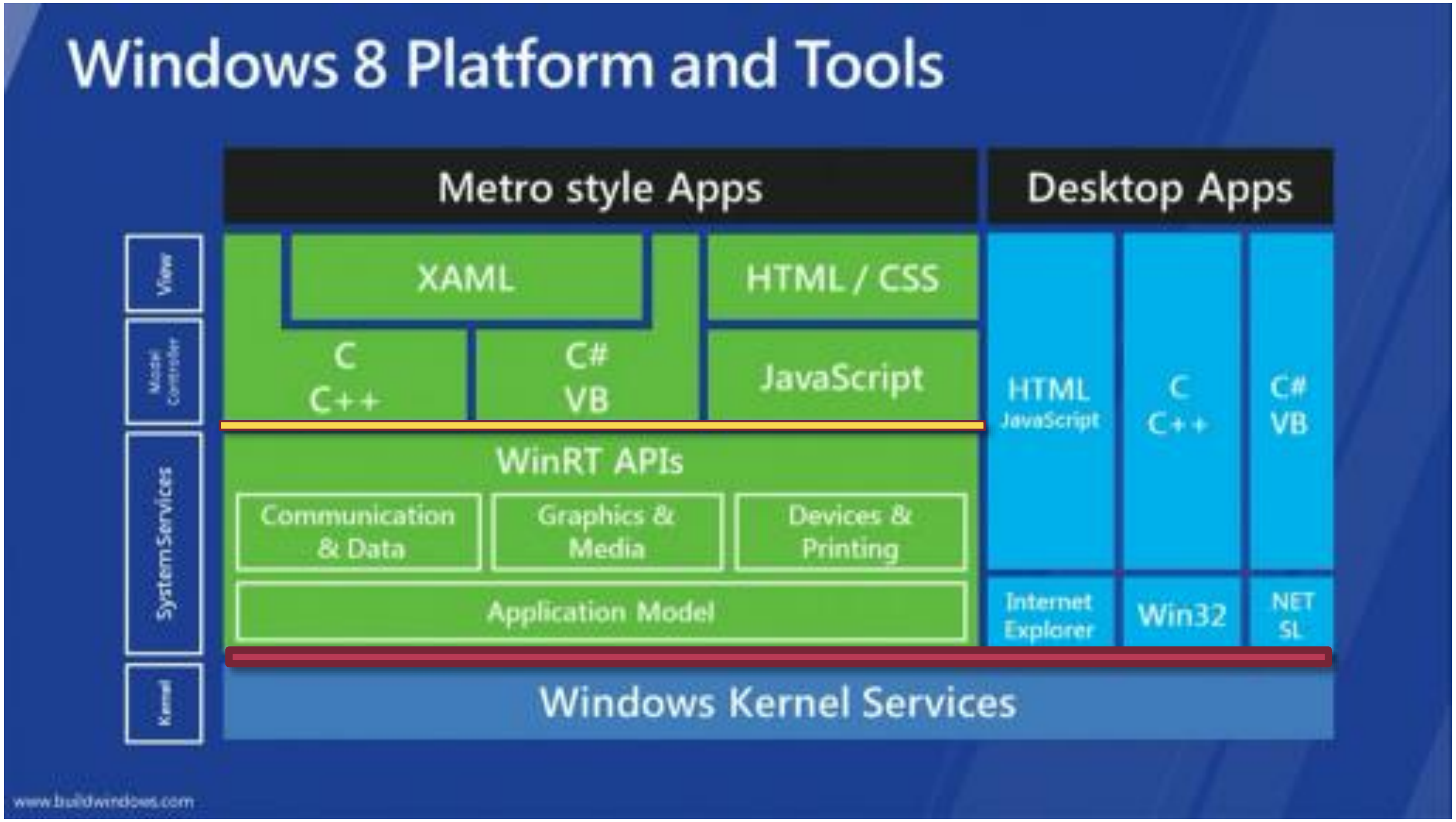
- Windows API function
  - E.g. ReadFile
  - documented in the SDK
- System services
  - Functions of the Executice callable from user mode
- Windows internal functions
  - Callable only from kernel mode

The Windows kernel

- Monolithic or microkernel?
- Shows mikrokernel-like properties
  - Only minimal functionality in the kernel
  - Kernel only callable on well-defined interfaces
  - Part of the OS runs in user mode
- However
  - Protected components run in one address space
- (other names)

(Windows 8: WinRT)

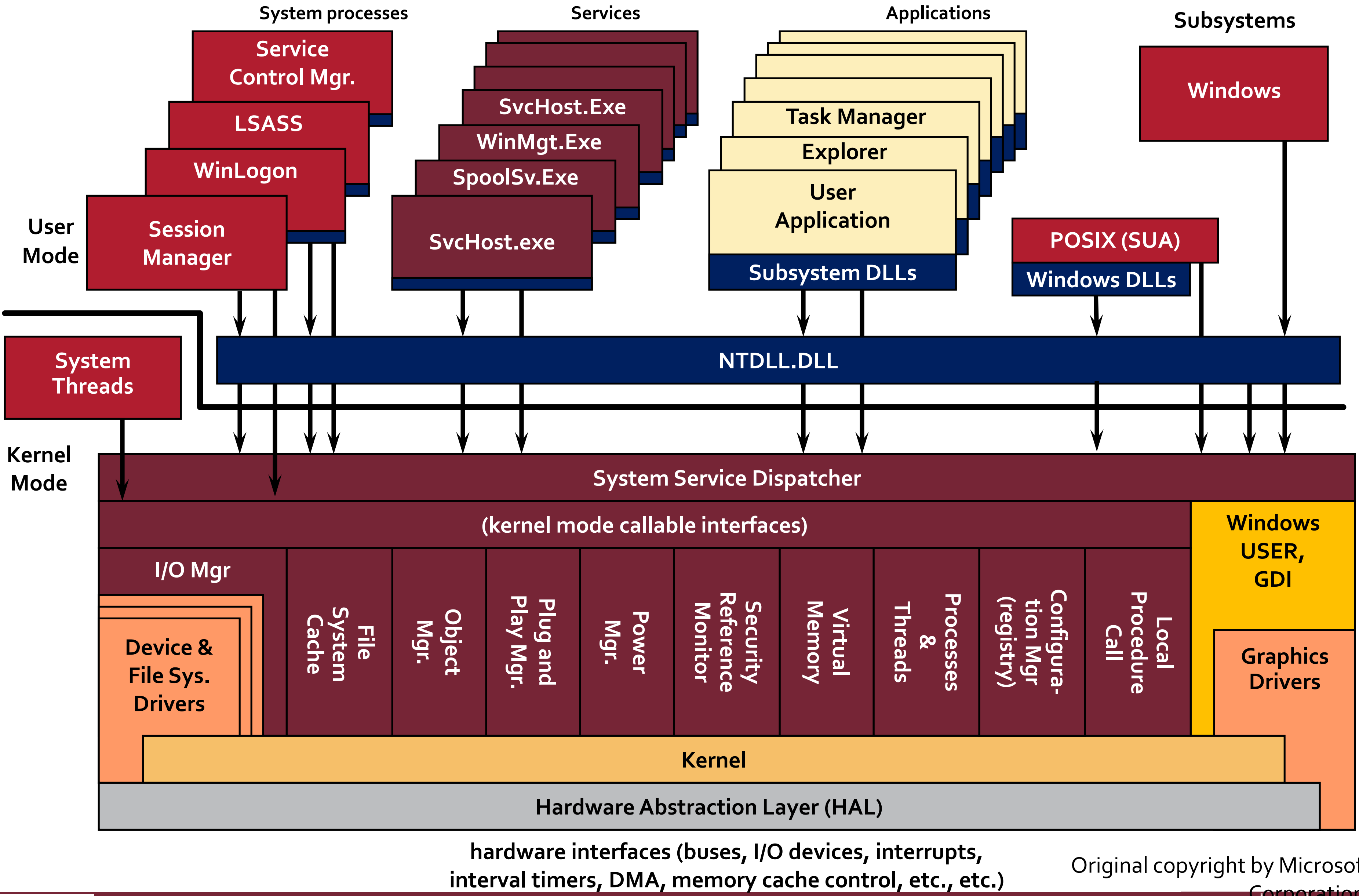
- One more layer
- Support for Metro / Immersive apps



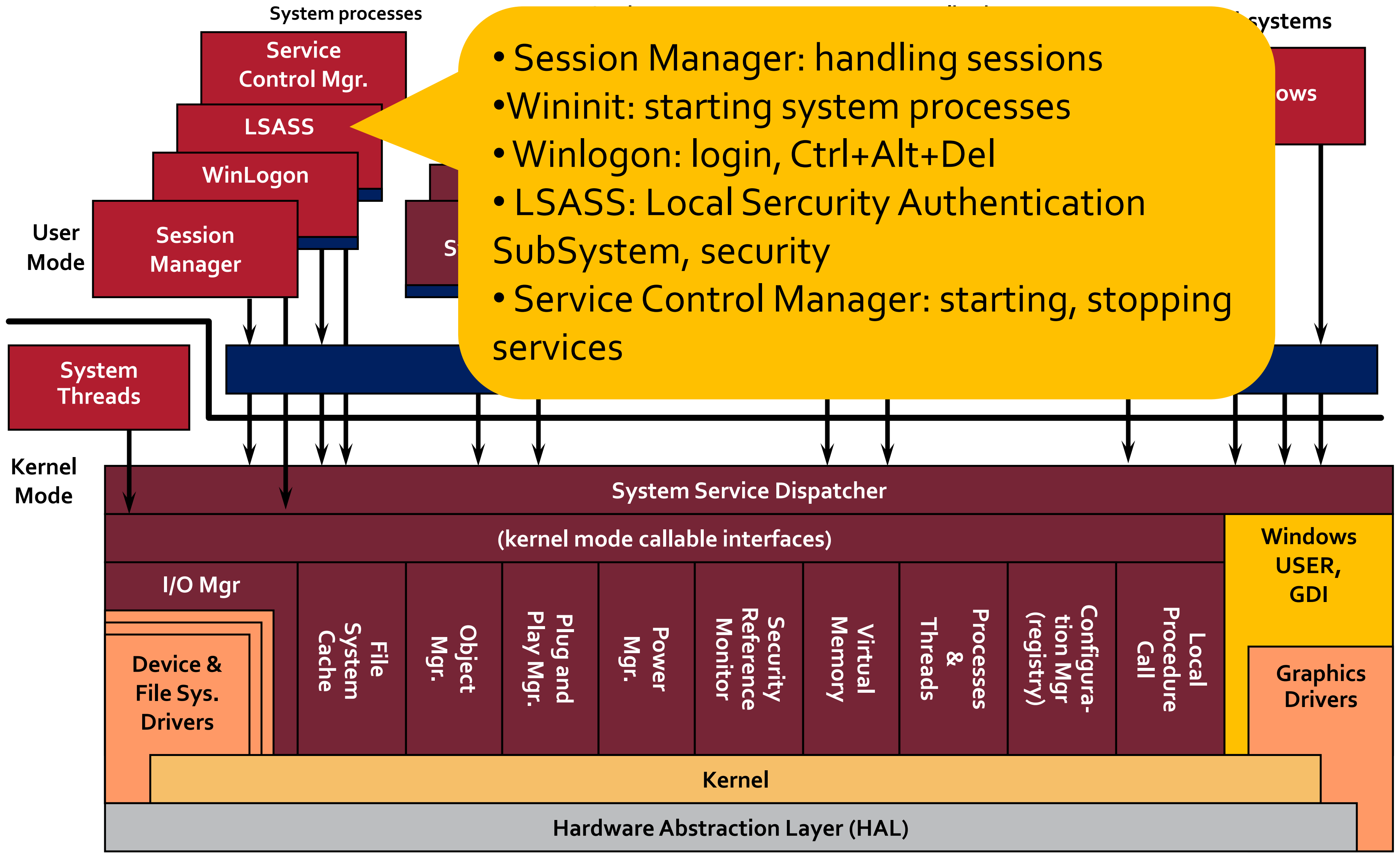
Windows on ARM (WOA)

- Separate product
  - ~ consumer device
- WOA (and Windows 8): new design goals
  - thin and light design
  - long battery life
  - integrated quality

Not so simplified architecture



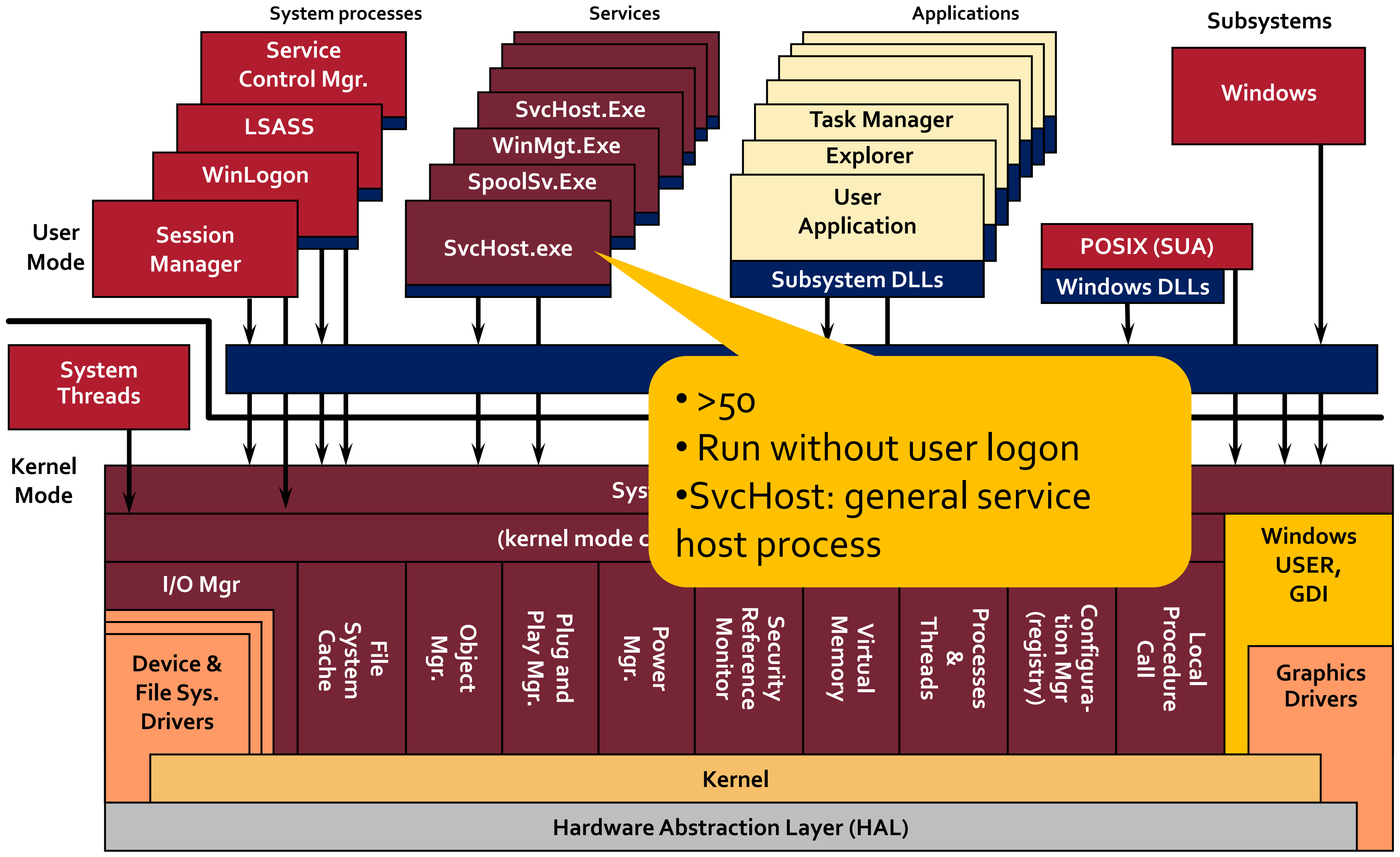
Not so simplified architecture



- Session Manager: handling sessions
- Wininit: starting system processes
- Winlogon: login, Ctrl+Alt+Del
- LSASS: Local Sercurity Authentication SubSystem, security
- Service Control Manager: starting, stopping services

hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)

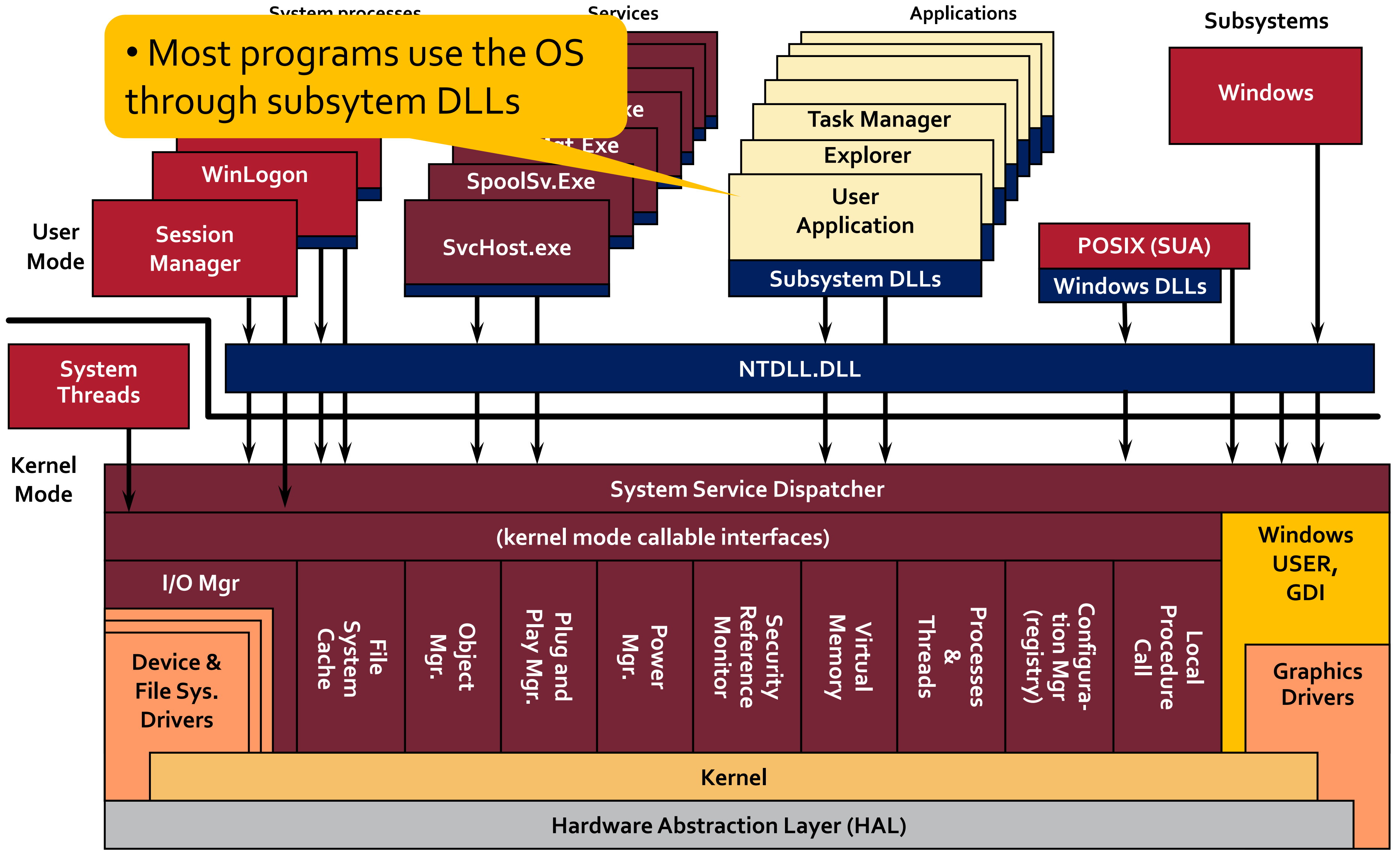
Not so simplified architecture



- >50
- Run without user logon
- SvcHost: general service host process

hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)

Not so simplified architecture

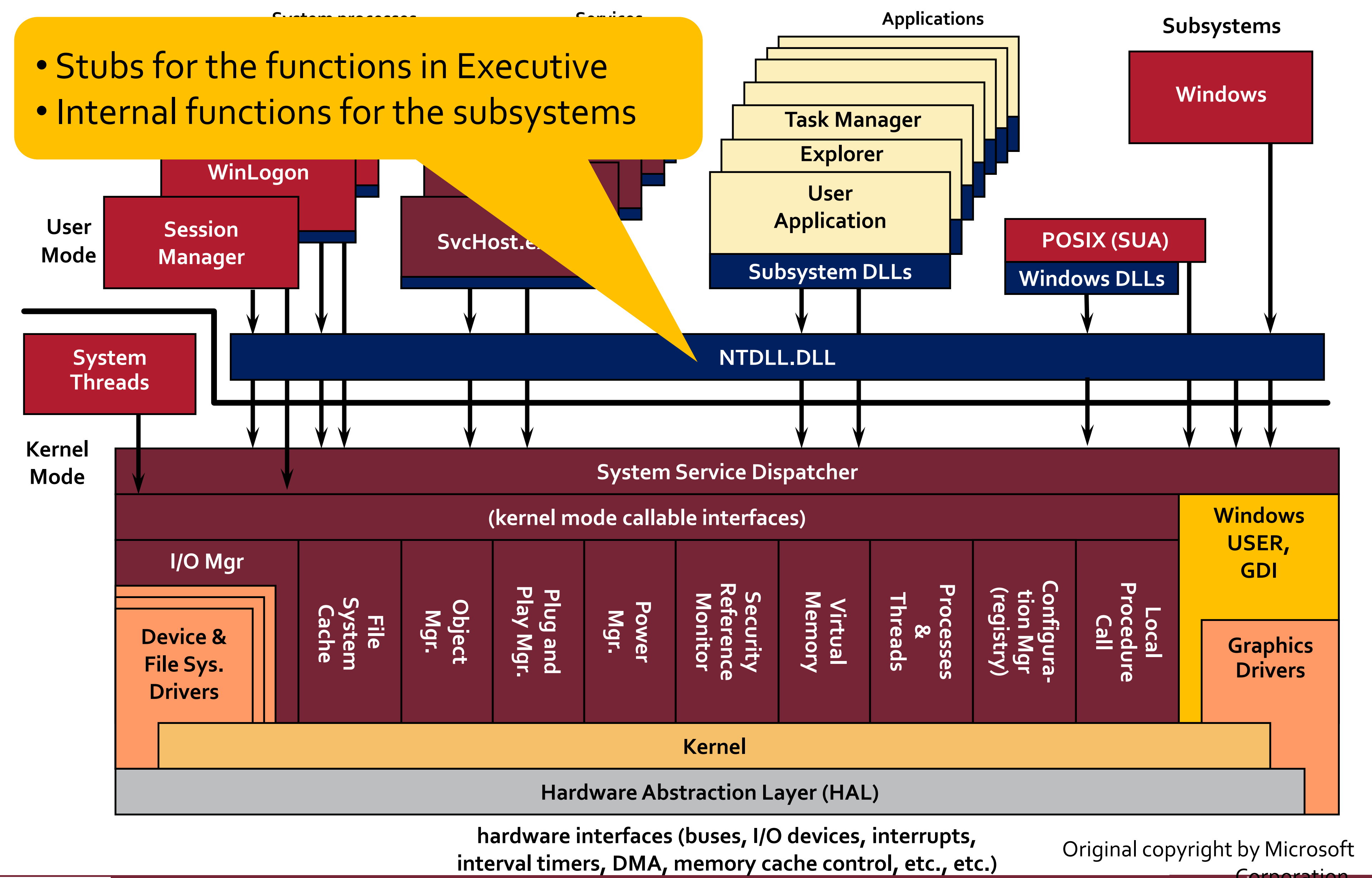


- Most programs use the OS through subsytem DLLs

hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)

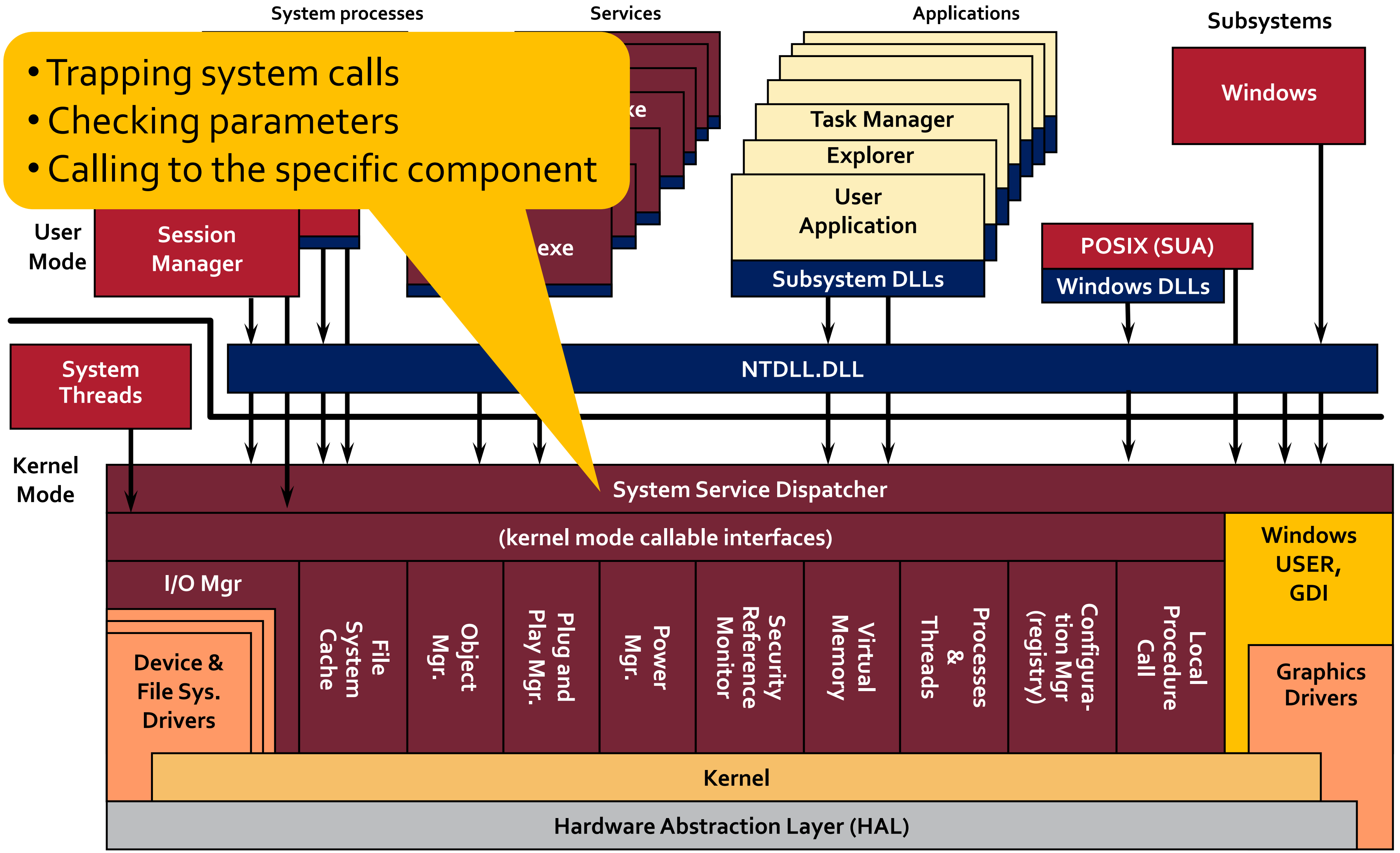
Original copyright by Microsoft Corporation.

Not so simplified architecture



- Stubs for the functions in Executive
- Internal functions for the subsystems

Not so simplified architecture

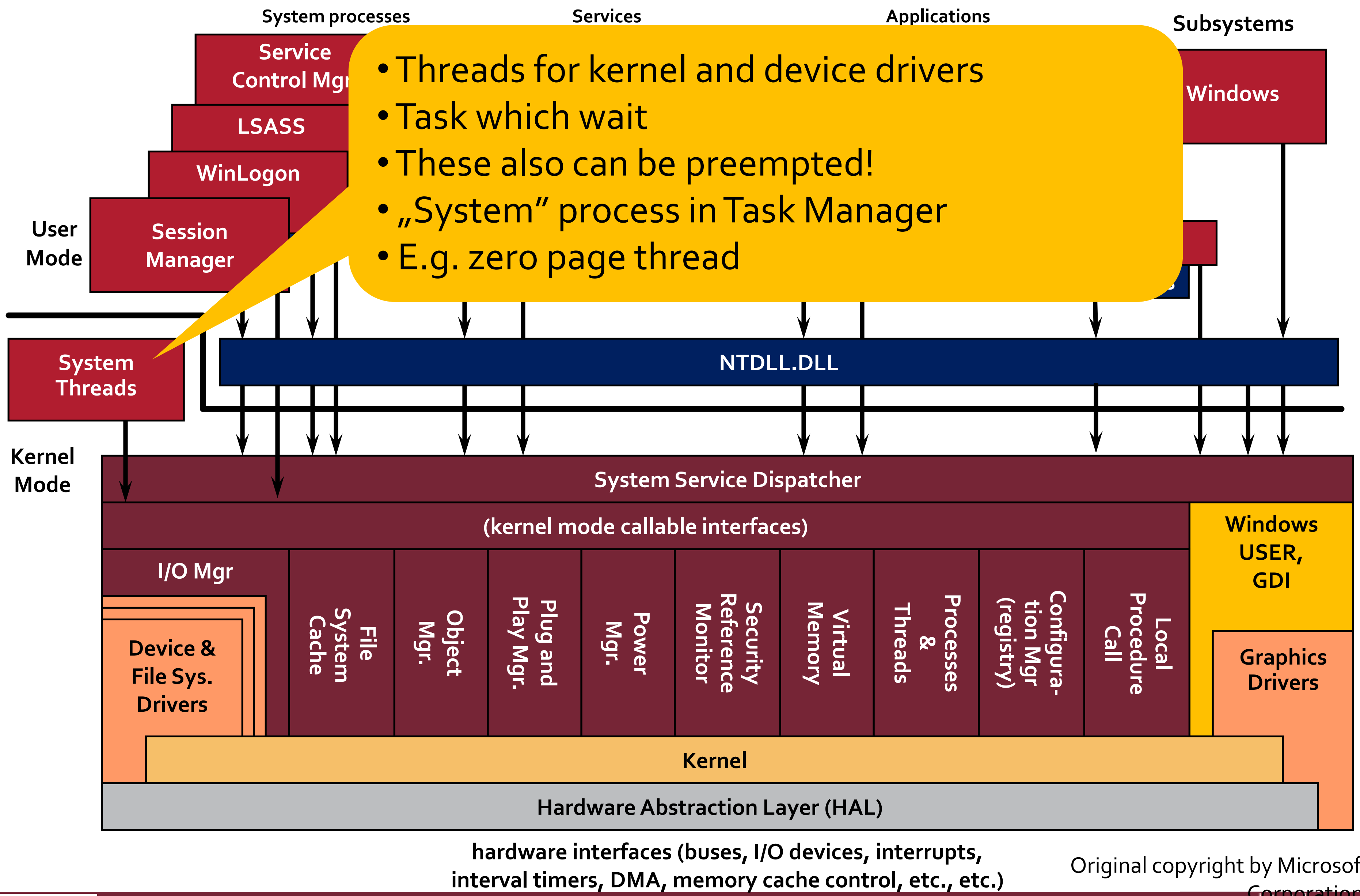


- Trapping system calls
- Checking parameters
- Calling to the specific component

hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)

Original copyright by Microsoft Corporation.

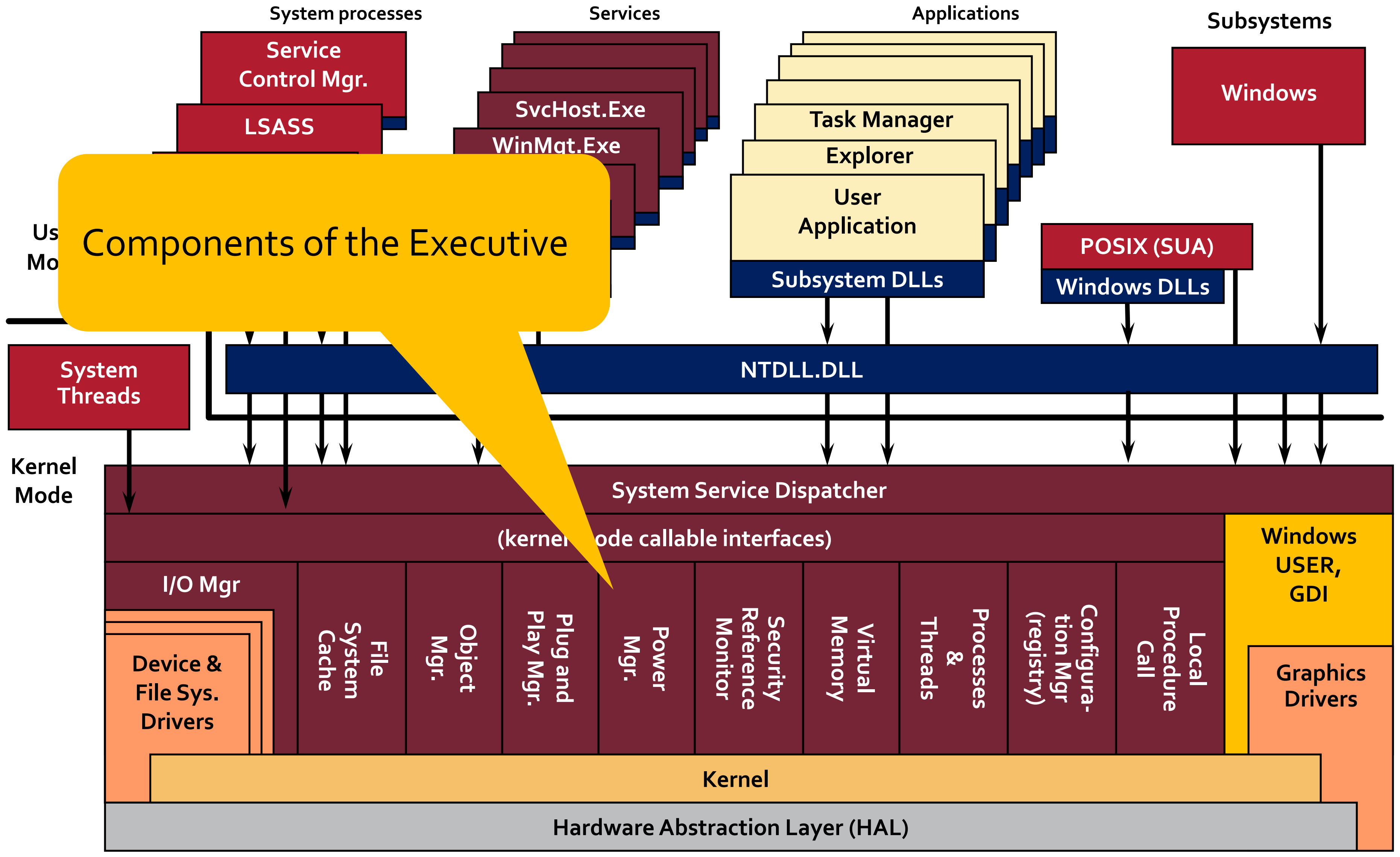
Not so simplified architecture



- Threads for kernel and device drivers
- Task which wait
- These also can be preempted!
- „System” process in Task Manager
- E.g. zero page thread

Original copyright by Microsoft Corporation.

Not so simplified architecture



hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)

DEMO

Process Explorer

Base OS components:

- NTOSKRNL.EXE: Executive and kernel
- HAL.DLL: Hardware abstraction layer
- NTDLL.DLL: Stubs for the Executive

System processes:

- SMSS.EXE: Session manager process
- WINLOGON.EXE: Logon process
- SERVICES.EXE: Service controller process
- LSASS.EXE: Local Security Authority Subsystem

Windows subsystem, GUI:

- CSRSS.EXE: Windows subsystem process
- WIN32K.SYS: USER and GDI kernel-mode components
- KERNEL32/USER32/GDI32.DLL: Windows subsystem DLLs

Windows versions

- The same source scales from
  - 1 CPU, 1 GB memory (Windows Vista Starter)
  - 64 CPU, 2 TB memory (Windows Server 2008 Datacenter Edition)
- Depending on settings in the registry:
  - Server or client
  - Type of server
- Differences
  - Defaults values for scheduling, memory mgmt
  - Licensing limits

Tools to dig in..

- Windows SDK
  - Successor of the Platform SDK, .NET Framework SDK
  - C/C++ headers, API description, compiles
- Windows Driver Kit
  - Successor of the Windows DDK
  - C/C++ headers, documentation, static verifyers
- Windows Debugging Tools
  - User and kernel mode debugger
- Sysinternals
  - Company of Mark Russinovich (MS bought it)
  - Process Explorer, Filemon, liveKd…
- Windows Support Tools, Windows Resource Kit
- …

To read

- Mark E. Russinovich and David A. Solomon with Alex Ionescu: ***Microsoft Windows Internals***, 6th Edition, Microsoft Press, 2012.
  - Everything about Windows
- MSDN, Building Windows 8, http://blogs.msdn.com/b/b8/