Windows Kernel

History, Evolution, Future

Weber Ress – H2HC 6th

about_me

Weber Ress

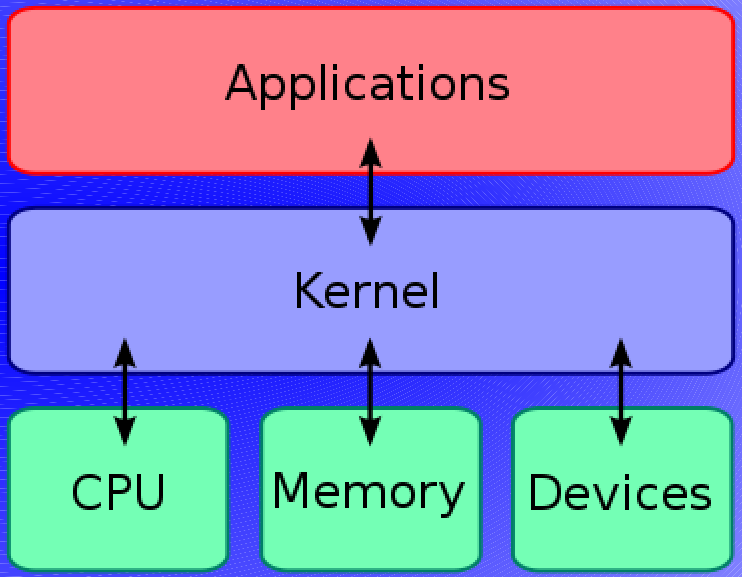
www.WeberRess.com

@WeberRess

Microsoft® Most Valuable Professional

Kernel Pure Concept

Kernel can be thought of as the bridge between application and the actual data processing done at the hardware level



Kernel Basic Facilities

Process Management

Memory Management

Device Management

System Calls

Kernel Design Decisions

Design Decisions from 60’s, 70’s and 80’s

Design build to an inocence and secure world

CPU design decisions share the same idea

Kernel Design Decisions

Fault Tolerance

Rings Strategy

x86 – 4 rings

Multics – 8 rings

NT – use 2 rings

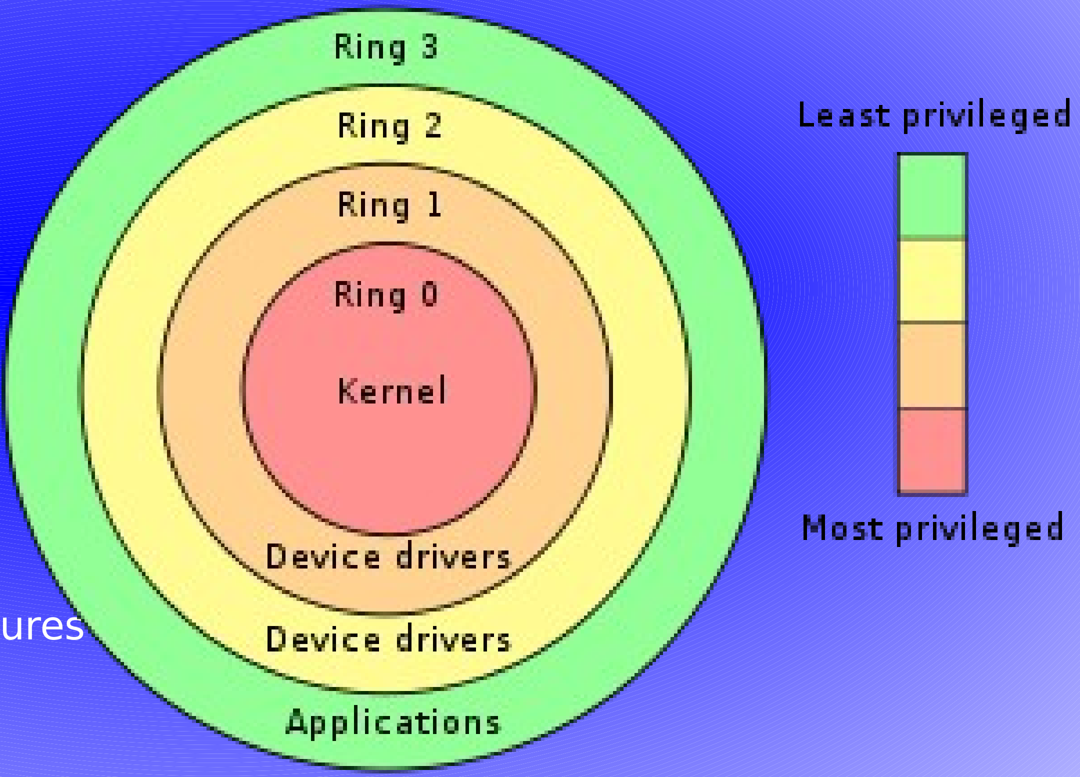
OS/2 – use 3 rings

Spaces

Kernel Space

User Space

More details @ CPU manufactures redbook's



Kernel Design Decisions

From 60’s and 70’s, the kernel design is focused in:

Monolithic

- Linux, *BSD, Solaris, MS-DOS, Windows 3.x, 9x, ME, Mac OS up to 8.6

Microkernel

- Minix, Mach, QNX, L4

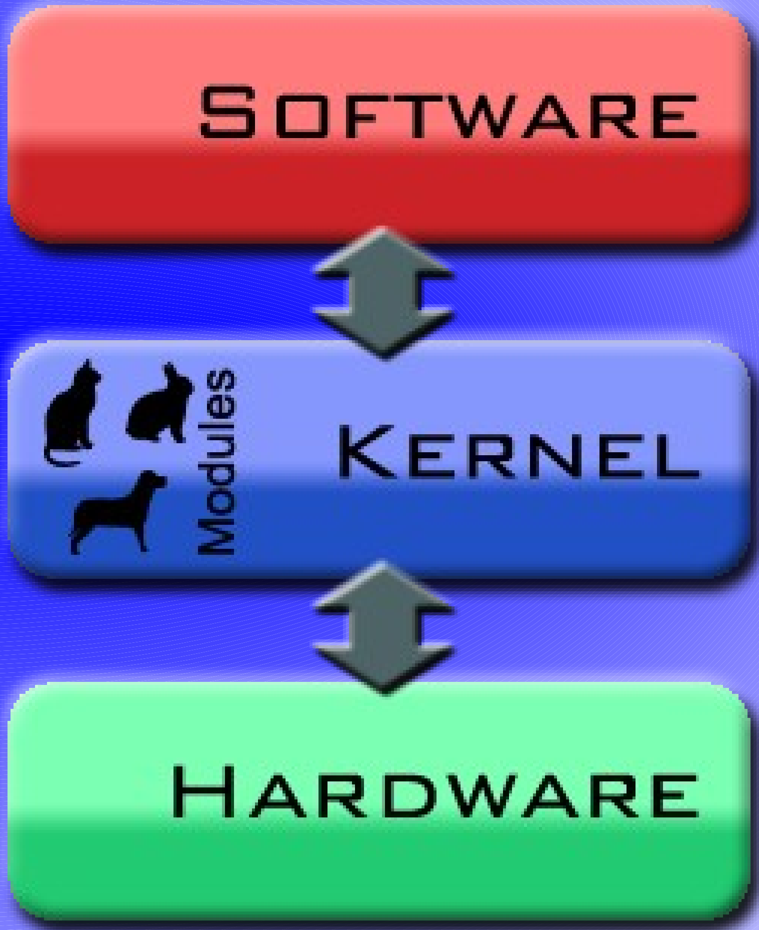
Hybrid

Kernel Design Decisions

Monolithic

all OS services run along with the main kernel thread, thus also residing in the same memory area. This approach provides rich and powerful hardware access.

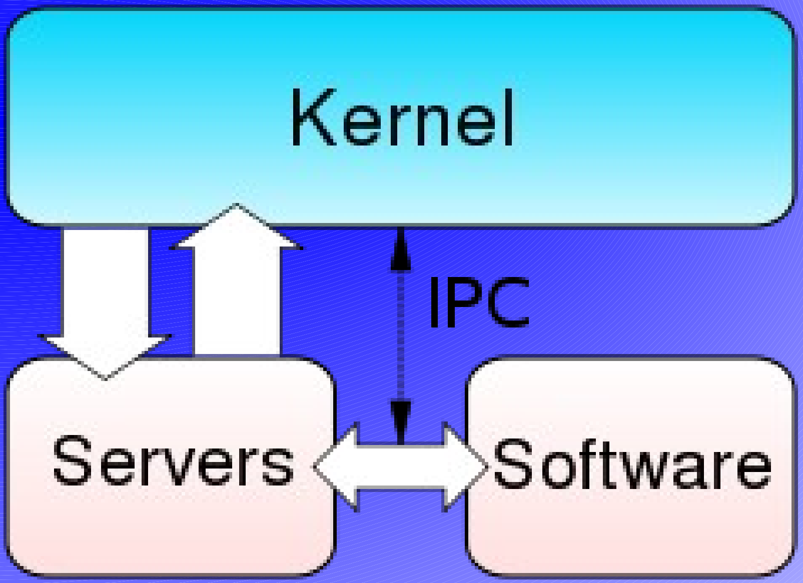
Main disadvantage: bug in a device driver / module might crash the entire system. Large kernel are hard to maintain.



Kernel Design Decisions

Microkernel

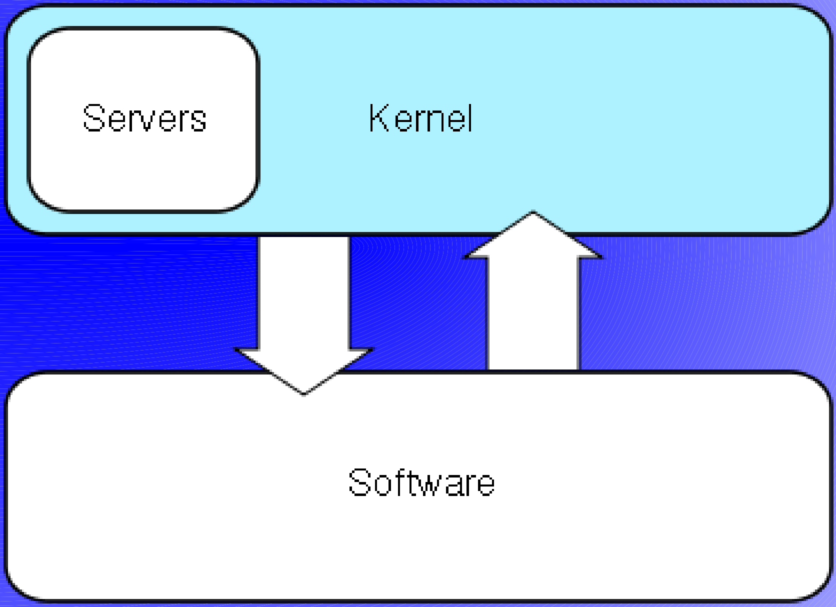
Run most of the operating system services in user space as servers, aiming to improve maintainability and modularity of the operating system.



Kernel Design Decisions

Hybrid Kernel

Hybrid kernels are a compromise between the monolithic and microkernel designs. This implies running some services (such as the network stack or the filesystem) in kernel space to reduce the performance overhead of a traditional microkernel, but still running kernel code (such as device drivers) as servers in user space.



Kernel in Real World

Hybrid

Windows: 88,5 %

Mac OS: 6,8%

Monolithic

Linux: 4,2%

source: www.w3schools.org

Microkernel

QNX is the only viable commercial microkernel; most used in airports and Space Shuttle.

Windows Kernel

Started in 1988, led by David Cutler from Digital

6 guys from Digital, 1 guy from Microsoft.

3 years of development, from scratch

Inspired in DEC VMS, DEC RSX-11 and PRISM

The idea was to have a common code base with a custom Hardware Abstraction Layer (HAL) for each platform (Have you seen this ?)

Windows NT 3.1 was initially developed using non-x86 development systems and then ported to the x86 architecture

Windows Kernel

Main Design Goals:

Hardware and software portability: (Intel IA-32, MIPS R3000/R4000 and Alpha, with PowerPC, Itanium and AMD64), and a private version to Sun SPARC architecture.

Reliability: Nothing should be able to crash the OS. Anything that crashes the OS is a bug

Compatibility: DOS, Win16, Win32, OS/2, POSIX applications

Windows Kernel

in Windows NT 3.x, several I/O driver subsystems, such as video and printing, were user-mode subsystems.

In Windows NT 4, the video, server and printer spooler subsystems were integrated into the kernel.

In Windows 7 and 2008 R2, many subsystems was integrated into the kernel (UAC, TPM)

Windows Kernel

Windows NT's kernel mode code further distinguishes between the "kernel", whose primary purpose is to implement processor and architecture dependent functions, and the "executive".

Both the kernel and the executive are linked together into the single loaded module **ntoskrnl.exe**;

From outside this module there is little distinction between the kernel and the executive. Routines from each are directly accessible, as for example from kernel-mode device drivers.

Windows Kernel

While the x86 architecture supports four different privilege levels (numbered 0 to 3), only the two extreme privilege levels are used.

Usermode programs are run with CPL 3, and the kernel runs with CPL 0. These two levels are often referred to as "ring 3" and "ring 0", respectively.

Design decision to achieve code portability to RISC platforms, that only support two privilege levels

The original Multics system had eight rings, but many modern systems have fewer.

Windows Kernel

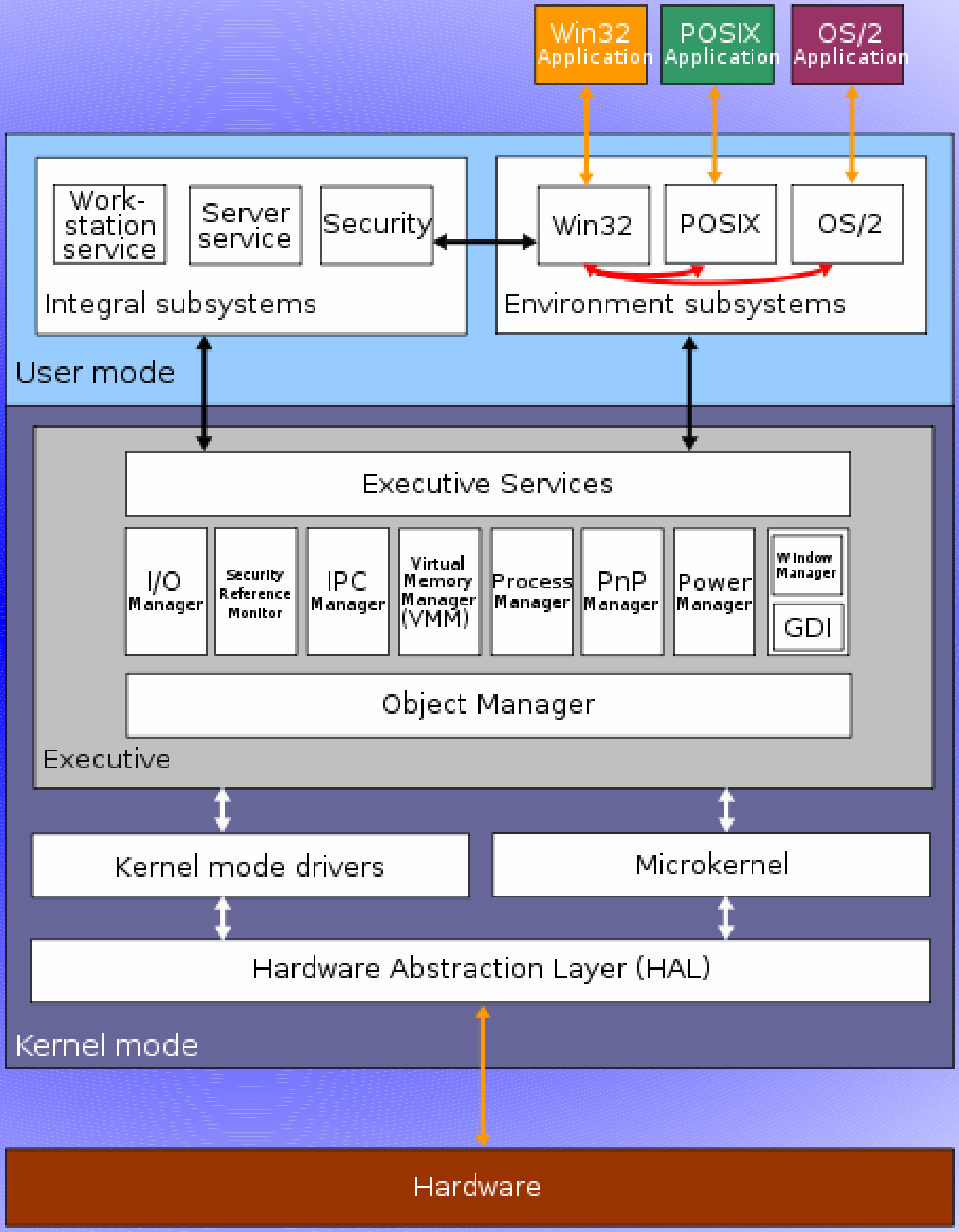
Effective use of ring architecture requires close cooperation between hardware and the operating system (the Apple advantage ??)

Operating systems designed to work on multiple hardware platforms may make only limited use of rings if they are not present on every supported platform.

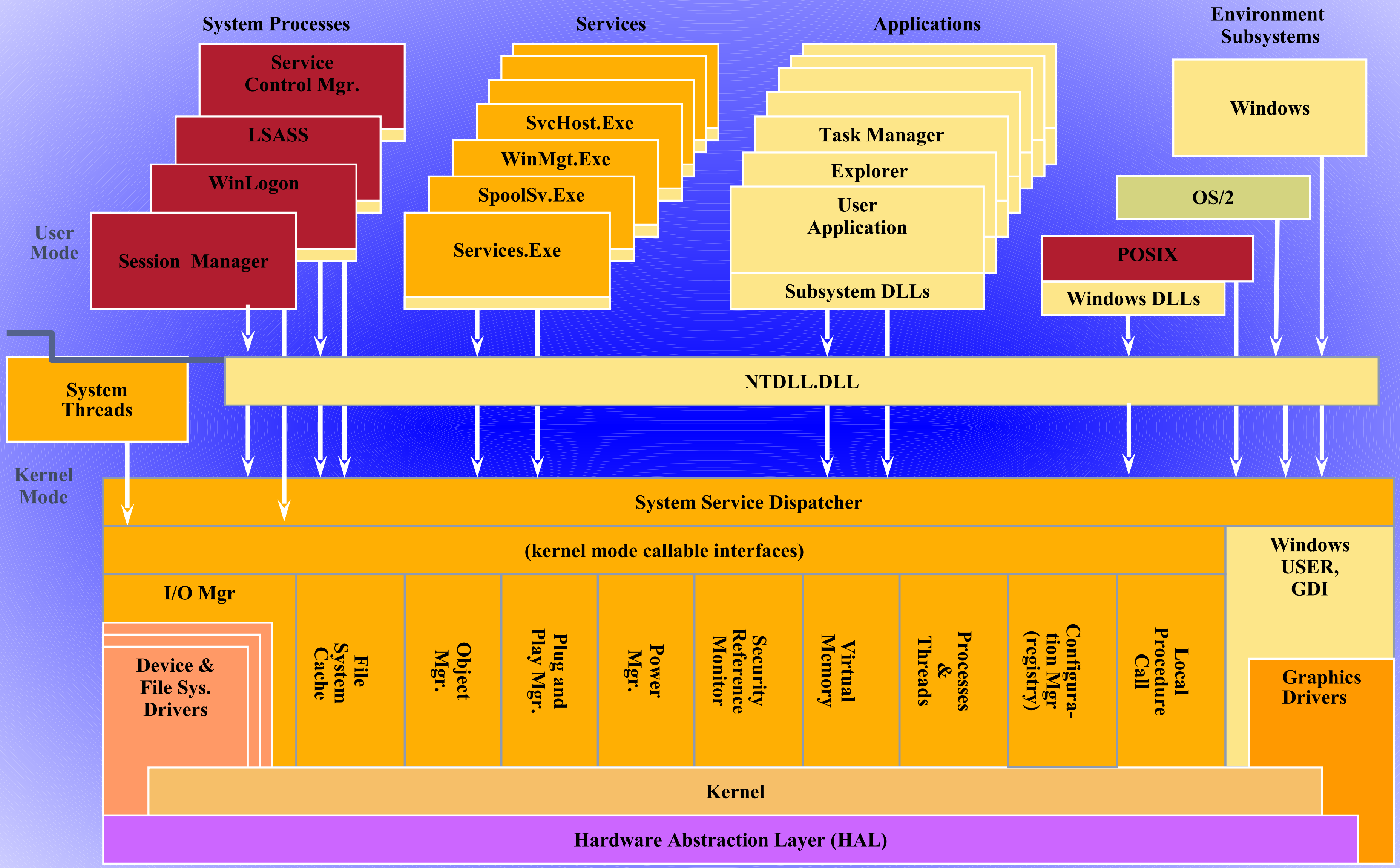
Often the security model is simplified to "kernel" and "user" even if hardware provides finer granularity through rings.

More security ? Build your hardware and OS (Microsoft future ? Apple first vision ?)

Windows Kernel

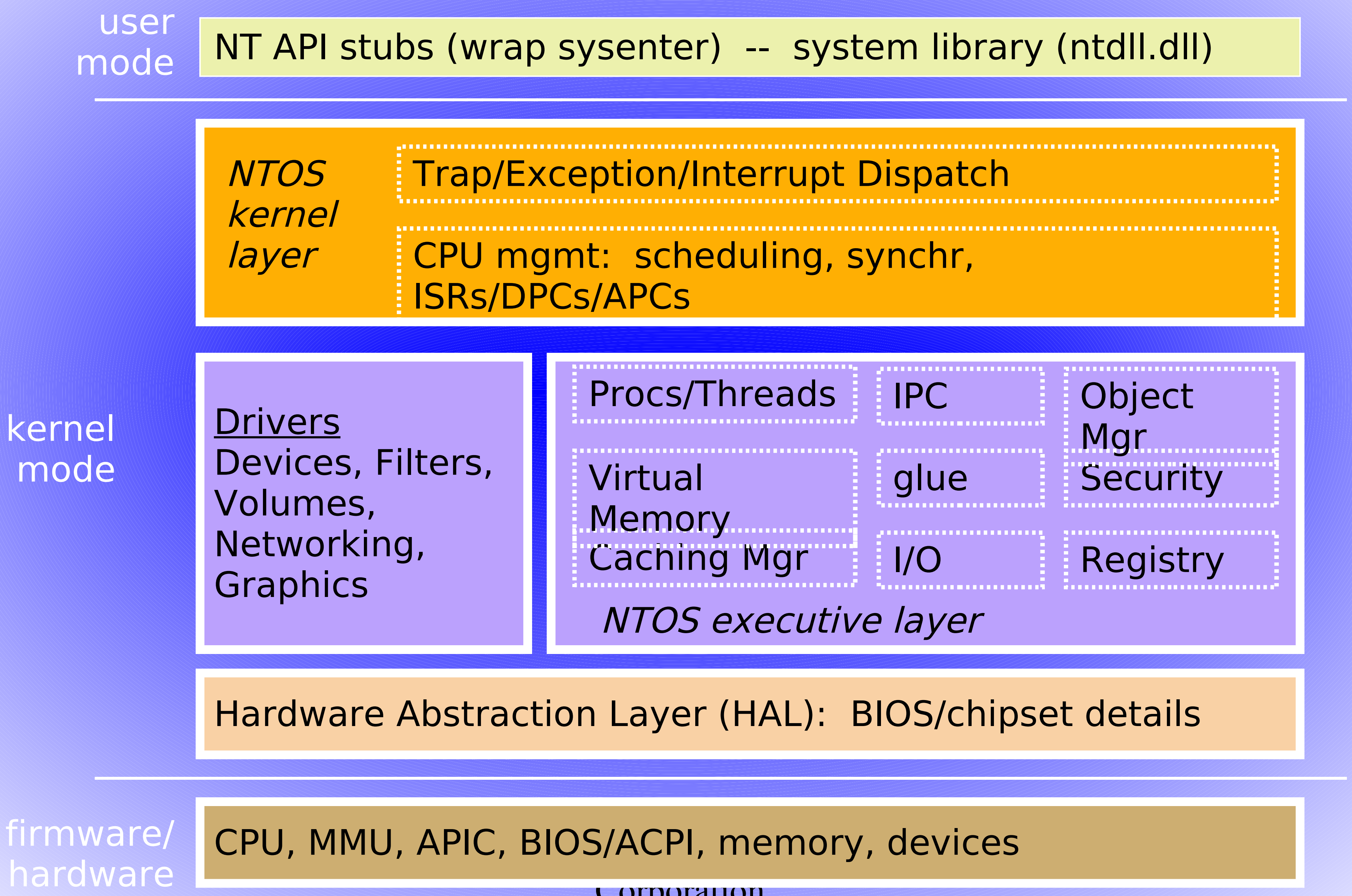


Windows Architecture



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

Architecture of Windows


user mode
NT API stubs (wrap sysenter) -- system library (ntdll.dll)
NTOS kernel layer
Trap/Exception/Interrupt Dispatch
CPU mgmt: scheduling, synchr, ISRs/DPCs/APCs
kernel mode
Drivers
Devices, Filters, Volumes, Networking, Graphics
Procs/Threads
IPC
Object Mgr
Virtual Memory
glue
Security
Caching Mgr
I/O
Registry
NTOS executive layer
Hardware Abstraction Layer (HAL): BIOS/chipset details
firmware/ hardware
CPU, MMU, APIC, BIOS/ACPI, memory, devices

Windows Vista Kernel Changes

Algorithms, scalability, code maintainability

CPU timing: Uses Time Stamp Counter (TSC)

- Interrupts not charged to threads
- Timing and quanta are more accurate

Communication

- ALPC: Advanced Lightweight Procedure Calls
- Kernel-mode RPC
- New TCP/IP stack (integrated IPv4 and IPv6)

I/O

- Remove a context switch from I/O Completion Ports
- I/O cancellation improvements

Memory management

- Address space randomization (DLLs, stacks)

Windows Vista Kernel Changes

Many improvements in a era with slow machines (Intel Core Duo / Core 2 Duo was very expensise)

Modern software in old machines = no responsiveness

Other mistakes, sure… ▪ , but it’s an evolution

Windows 7 Kernel

First general “public” revision of Windows Kernel

Improvements on performance (big mistake in Vista); What matters is responsiveness !

Kernel Dispacher Lock was replaced by “*more complex symbolic system of semantics that lets threads execute in a more parallel, efficient fashion*".

Previous Windows Kernel don’t understand the new global reality: Multi Core

Windows 7 Kernel

Current Windows Kernel systems have a global dispatcher lock which essentially stop all cores to prevent objects from being accessed by more than one core.

Windows 7/2008 Server will use processor groups where threads will be assigned to groups of cores.

Windows 7 / 2008 can use 256 cores

Windows 7 Kernel

00000000 00000000 00000000 00000000 = thread can run on all processors (affinity is basically off)

00000000 00000000 00000000 00000001 = thread runs only on the first processor

00000000 00000000 00000000 00000010 = thread runs only on the second processor

00000000 00000000 00000000 00000100 = thread runs only on the third processor

00000000 00000000 00000000 00000111 = thread can be distributed across the first three processors

Windows 7 Kernel Changes

MinWin

- Change how Windows is built
- Lots of DLL refactoring
- API Sets (virtual DLLs)

Working-set management

- Runaway processes quickly start reusing own pages
- Break up kernel working-set into multiple working-sets
  - System cache, paged pool, pageable system code

Security

- Better UAC, new account types, less BitLocker blockers

Energy efficiency

MinWin

MinWin is first step at creating architectural partitions

Can be built, booted and tested separately from the rest of the system

Higher layers can evolve independently

An engineering process improvement, not a microkernel NT!

MinWin was defined as set of components required to boot and access network

Kernel, file system driver, TCP/IP stack, device

Kernel Future

- Many-core challenge
- New driving force in software innovation:
  - Amdahl’s Law (1) overtakes Moore’s Law as high-order bit
- Heterogeneous cores?
- OS Scalability
- Loosely –coupled OS: mem + cpu + services?
- Energy efficiency
- Shrink-wrap and Freeze-dry applications?
- Hypervisor/Kernel/Runtime relationships
- Move kernel scheduling (cpu/memory) into run-times?
- [illegible]

Thank you

Weber Ress

www.WeberRess.com

@WeberRess