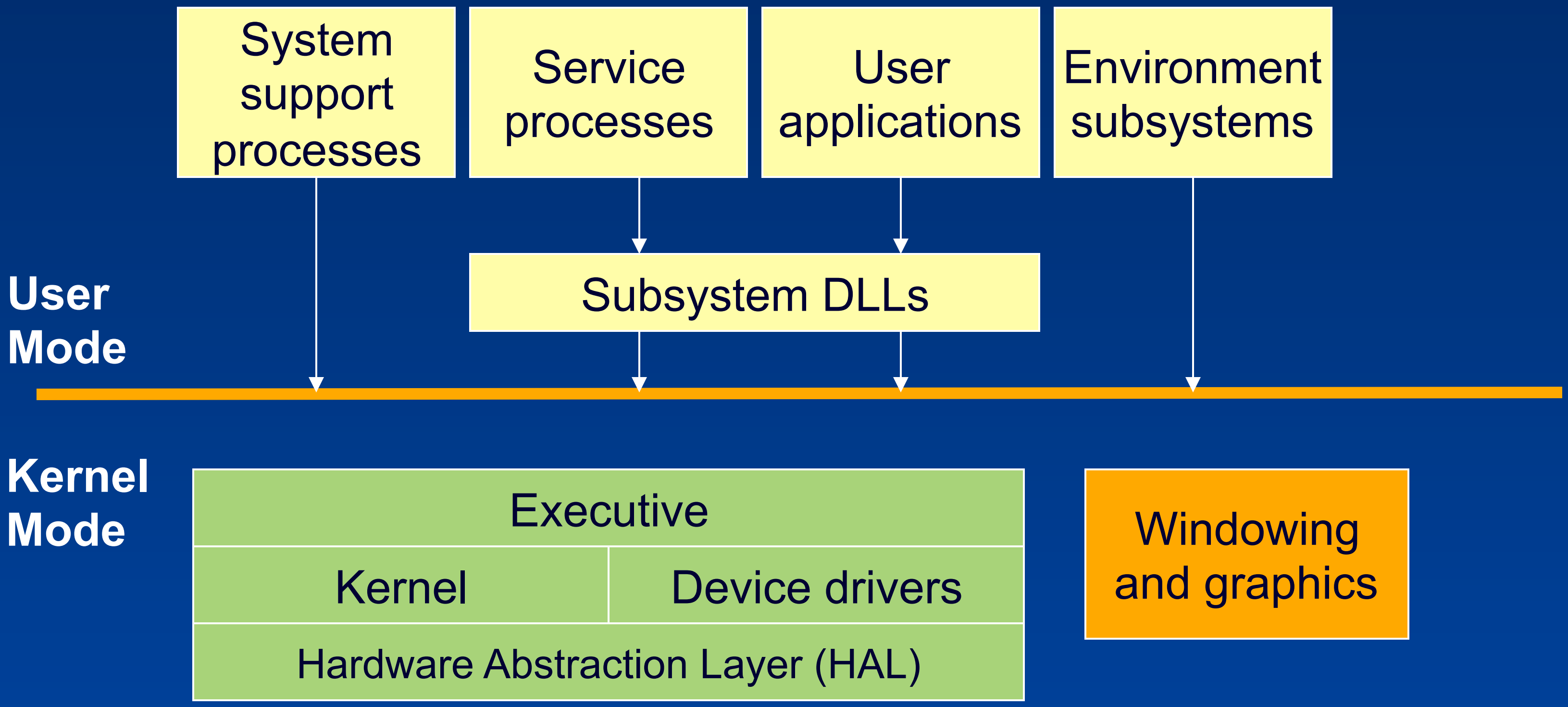
# Structuring of the Windows Operating System

## Roadmap for This Lecture

- Architecture Overview
  - Key windows system files
  - Design Attributes and Features
- Key System Components
- System Threads
- System Processes / Services

# Simplified OS Architecture



# OS Architecture

- Multiple personality OS design
  - user applications don't call the native Windows operating system services directly
- Subsystem DLLs is to translate a documented function into the appropriate internal (and undocumented) Windows system service calls.
- Environment subsystem processes
  - Manage client processes in their world
  - Impose semantics such as process model, security
- Originally three environment subsystems: Windows, POSIX, and OS/2
  - POSIX: **P**ortable **O**perating **S**ystem **I**nterface for Uni**x**
  - Windows 2000 only included Windows and POSIX
  - Windows XP only includes Windows
    - Enhanced POSIX subsystem available with Services for Unix
    - Included with Windows Server 2003 R2

## Kernel-Mode Components: Core OS

- *Executive*
  - base operating system services,
  - memory management, process and thread management,
  - security, I/O, interprocess communication.
- *Kernel*
  - low-level operating system functions,
  - thread scheduling, interrupt and exception dispatching,
  - multiprocessor synchronization.
  - provides a set of routines and basic objects that the rest of the executive uses to implement higher-level constructs.
- Both contained in file Ntoskrnl.exe

# Kernel-Mode Components: Drivers

- *Device drivers* (*.sys)
  - hardware device drivers translate user I/O function calls into specific hardware device I/O requests
  - virtual devices - system volumes and network protocols
- *Windowing and Graphics Driver* (Win32k.sys)
  - graphical user interface (GUI) functions (USER and GDI)
  - windows, user interface controls, and drawing
- *Hardware Abstraction Layer* (Hal.dll)
  - isolates the kernel, device drivers, and executive from hardware
  - Hides platform-specific hardware differences (motherboards)

## User-Mode Processes

- *System support processes*
  - Logon process, session manager, etc.
- *Services processes*
  - Windows services independent of user logons
  - Task Scheduler, printer spooler, etc.
- *User applications*
  - Windows 32-bit, 64-bit, Windows 3.1 16-bit, MS-DOS 16-bit and POSIX 32-bit
- *Environment subsystems*
  - OS environment *a.k.a. personalities*
  - Windows, POSIX, OS/2 (dropped after Windows 2000)

# Key Windows System Files

Core OS components:

| | |
|---|---|
| NTOSKRNL.EXE** | Executive and kernel |
| HAL.DLL | Hardware abstraction layer |
| NTDLL.DLL | Internal support functions and system service dispatch stubs to executive functions |

Core system (support) processes:

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

Windows subsystem:

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

# NTOSKRNL.EXE

- **Core operating system image**
  - contains Executive & Kernel
  - Also includes entry points for routines actually implemented in Hal.Dll
  - Many functions are exposed to user mode via NtDll.Dll and the environment subsystems (t.b.d.)
- **Four retail variations:**

| | |
|---|---|
| NTOSKRNL.EXE | Uniprocessor |
| NTKRNLMP.EXE | Multiprocessor |

  - Windows 2000 adds PAE (page address extension) versions – must boot /PAE (32-bit Windows only); also used for processors with hardware no execute support (explained in Memory Management unit)

| | |
|---|---|
| NTKRNLPA.EXE | Uniprocessor w/extended addressing support |
| NTKRPAMP.EXE | Multiprocessor w/extended addressing support |

- **Two checked build (debug) variations:**

| | |
|---|---|
| NTOSKRNL.EXE, | |
| NTKRNLMP.EXE | Debug multiprocessor |
| NTKRNLPA.EXE, | |
| NTKRPAMP.EXE | Debug multiprocessor w/extended addressing |

# Portability

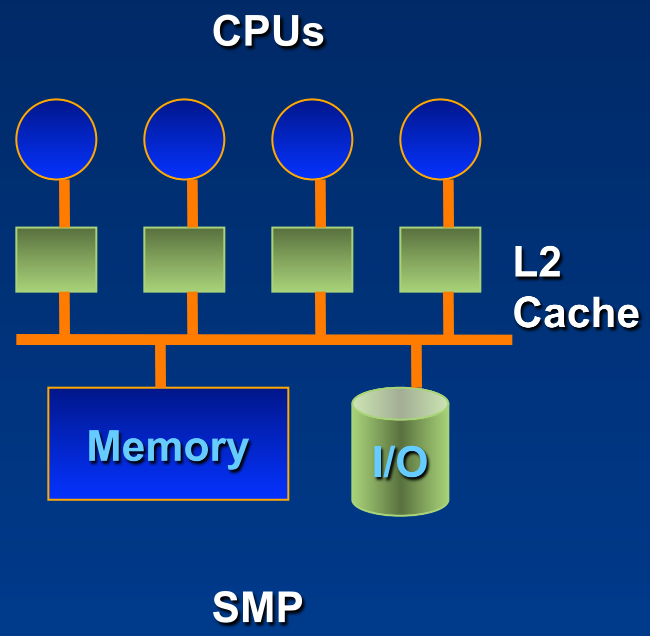
- When Windows NT was designed, there was no dominant processor architecture
  - Therefore it was designed to be portable
- How achieved?
  - Most Windows OS code and device drivers is written in C
    - HAL and kernel contain some assembly language
  - Some components are written in C++:
    - windowing/graphics subsystem driver
    - volume manager
  - Hardware-specific code is isolated in low level layers of the OS (such as Kernel and the HAL)
    - Provides portable interface
- XP and Server 2003 support
  - Intel Itanium IA-64
  - AMD64
  - Intel 64-bit Extension Tech (EM64T)

# Reentrant and Asynchronous Operation

- Windows kernel is fully reentrant
  - Kernel functions can be invoked by multiple threads simultaneously
  - No serialization of user threads when performing system calls
- I/O system works fully asynchronously
  - Asynchronous I/O improves application's throughput
  - Synchronous wrapper functions provide ease-of-programming

# Symmetric Multiprocessing (SMP)

- No master processor
  - All the processors share just one memory space
  - Interrupts can be serviced on any processor
  - Any processor can cause another processor to reschedule what it's running
- Maximum # of CPUs stored in registry
  - HKLM\System\CurrentControlSet
    \Control\Session Manager
    \LicensedProcessors
- Current implementation limit is # of bits in a native word
  - 32 processors for 32-bit systems
  - 64 processors for 64-bit systems
  - Not an architectural limit—just implementation



# UP vs MP File Differences

- These files are updated when moving from UP to MP:

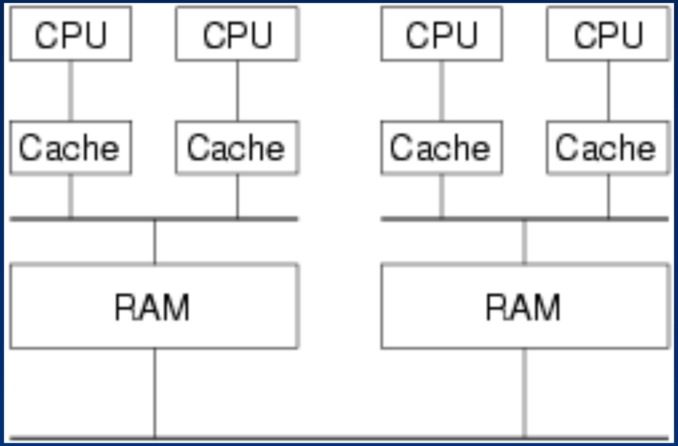
| *Name of file on system disk* | *Name of uniprocessor version on CD-ROM* | *Name of multiprocessor version on CD-ROM* |
|---|---|---|
| NTOSKRNL.EXE | \I386\NTOSKRNL.EXE | \I386\NTKRNLMP.EXE |
| NTKRNLPA.EXE | \I386\NTKRNLMP.EXE | \I386\NTKRPAMP.EXE |
| HAL.DLL | Depends on system type | Depends on system type |

- Everything else is the same (drivers, EXEs, DLLs)
  - NT4: Win32k.sys, Ntdll.dll, and Kernel32.dll had uniprocessor versions

# Hyperthreading

- New technology in newer Intel processors
  - Makes a single processor appear as  multi-processor to the OS
  - Also called simultaneous multithreading technology (SMT)
- Chip maintains many separate CPU states (“logical processors”)
  - Execution engine & onboard cache is shared
- XP & Server 2003 and later Windows are “hyperthreading aware”
  - Logical processors don’t count against physical processor limits
  - Scheduling algorithms take into account logical vs physical processors
    - Applications can also optimize for it (new Windows function in Server 2003)

# NUMA



- NUMA (non uniform memory architecture) systems
  - Groups of physical processors (called "nodes") that have local memory
    - Connected to the larger system through a cache-coherent interconnect bus
  - Still an SMP system (e.g. any processor can access all of memory)
    - But node-local memory is faster
- Scheduling algorithms take this into account
  - Tries to schedule threads on processors within the same node
  - Tries to allocate memory from local memory for processes with threads on the node
- New Windows APIs to allow applications to optimize

# SMP Scalability

- Scalability is a function of parallelization and resource contention
  - Can’t make a general statement
  - Depends on what you are doing and if the code involved scales well
- Kernel is scalable
  - OS can run on any available processor and on multiple processors at the same time
  - Fine-grained synchronization within the kernel as well as within device drivers allows more components to run concurrently on multiple processors
  - Concurrency has improved with every release
- Applications <u>can</u> be scalable
  - Threads can be scheduled on any available CPU
  - Processes can contain multiple threads that can execute simultaneously on multiple processors
  - Programming mechanisms provided to facilitate scalable server applications
    - Most important is I/O completion ports

## Differences between Client and Server Versions

- Number of CPUs supported  (# of packages, not cores)
- Amount of physical memory supported
- Number of concurrent networks
- Support for Tablet PC and Media Center Ed.
- Features like BitLocker, DVD burning, WinFax and 100+ config values
- Layered services unique for Servers, e.g. directory services and clustering

# Built On the Same Core OS

- **Through Windows 2000, core operating system executables were identical**
  - NTOSKRNL.EXE, HAL.DLL, xxxDRIVER.SYS, etc.
  - As stated earlier, XP & Server 2003 have different kernel versions
- **Registry indicates system type (set at install time)**
  - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control \ProductOptions
    - ProductType: WinNT=Workstation, ServerNT=Server not a domain controller, LanManNT=Server that is a Domain Controller
    - ProductSuite: indicates type of Server (Advanced, Datacenter, or for NT4: Enterprise Edition, Terminal Server, …)

## Built On the Same Core OS (Cont'd)

- **Code in the operating system tests these values and behaves slightly differently in a few places**
  - Licensing limits (number of processors, number of network connections, etc.)
  - Boot-time calculations (mostly in the memory manager)
  - Default length of time slice
  - See DDK: MmIsThisAnNtasSystem (check if it's server)

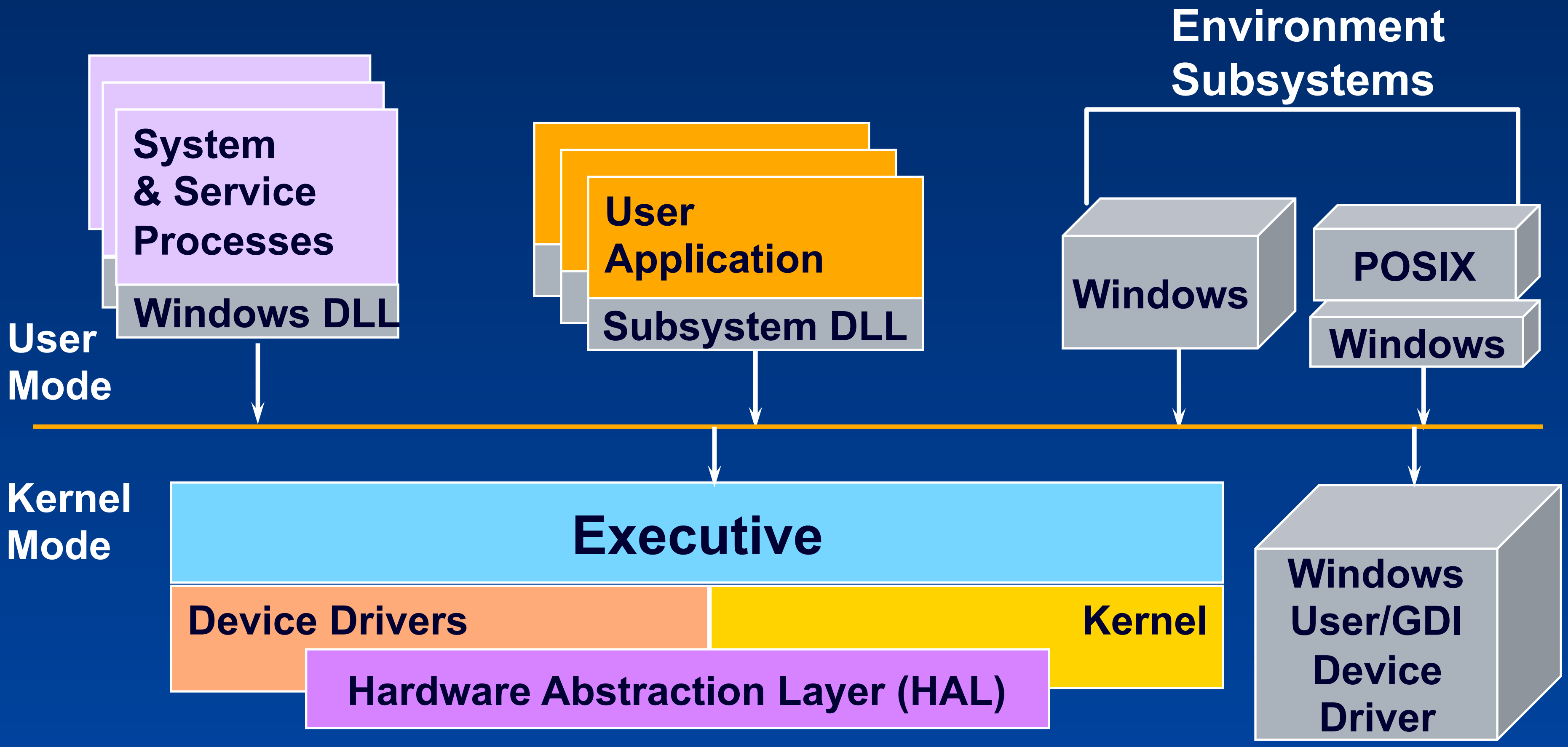
# Debug Version (“Checked Build”)

- **Special debug version of system called “Checked Build”**
  - Multiprocessor versions only (runs on UP systems)
    - helps catch synchronization bugs that are more visible on MP systems
  - Primarily for driver testing, but can be useful for catching timing bugs in multithreaded applications
- **Built from same source files as “free build” (aka “retail build”)**
  - But with “DBG” compile-time symbol defined
  - This enables:
    - error tests for “can’t happen” conditions in kernel mode (ASSERTs)
    - validity checks on arguments passed from one kernel mode routine to another

```
#ifdef DBG
        if (something that should never happen has happened)
                KeBugCheckEx(…)
#endif
```

- **Since no checked Windows 2000 Server provided, can copy checked NTOSKRNL, HAL, to a normal Server system**
  - Select debug kernel & HAL with Boot.ini /KERNEL=, /HAL= switches
- **Windows Server 2003 has its own checked build**
- **See Knowledge base article 314743 (HOWTO: Enable Verbose Debug Tracing in Various Drivers and Subsystems)**

# Key System Components



## Multiple OS Personalities

- Windows was designed to support multiple “personalities”, called environment subsystems
  - Programming interface
  - File system syntax
  - Process semantics
- Environment subsystems provide exposed, documented interface between application and Windows native API
  - Each subsystem defines a different set of APIs and semantics
  - Subsystems implement these by invoking native APIs
    - Example:  Windows CreateFile in Kernel32.Dll calls native NtCreateFile
- .exes and .dlls you write are associated with a subsystem
  - Specified by LINK /SUBSYSTEM option
  - Cannot mix calls between subsystems

## Environment Subsystems

- Three environment subsystems originally provided with NT:
  - Windows –Windows API (originally 32-bit, now also 64-bit)
  - OS/2 - 1.x character-mode apps only
    - Removed in Windows 2000
  - Posix - only Posix 1003.1 (bare minimum Unix services - no networking, windowing, threads, etc.)
    - Removed in XP/Server 2003 – enhanced version ships with Services For Unix 3.0
- Of the three, the Windows subsystem provides access to the majority of native OS functions
- Of the three, Windows is required to be running
  - System crashes if Windows subsystem process exits
  - POSIX and OS/2 subsystems are actually Windows applications
  - POSIX & OS/2 start on demand (first time an app is run)
    - Stay running until system shutdown

# App calls Subsystem

- Function entirely implemented in user mode
  - No message sent to environment subsystem process
  - No Windows executive system service called
  - Examples: *PtInRect(), IsRectEmpty()*
- Function requires one/more calls to Windows executive
  - Examples: Windows *ReadFile() / WriteFile()* implemented using I/O system services *NtReadFile() / NtWriteFile()*
- Function requires some work in environment subsystem process (maintain state of client app)
  - Client/server request (message) to env. Subsystem (LPC facility)
  - Subsystem DLL waits for reply before returning to caller
- Combinations of 2/3: *CreateProcess() / CreateThread()*

# Windows Subsystem

- Environment subsystem process (CSRSS.EXE):
  - Console (text) windows
  - Creating and deleting processes and threads
  - Portions of the support for 16-bit virtual DOS machine (VDM)
  - Other func: *GetTempFile, DefineDosDevice, ExitWindowsEx*
- Kernel-mode device driver (WIN32K.SYS):
  - Window manager: manages screen output;
  - input from keyboard, mouse, and other devices
  - user messages to applications.
  - Graphical Device Interface (GDI)

## Windows Subsystem (contd.)

- Subsystem DLLs (such as USER32.DLL, ADVAPI32.DLL, GDI32.DLL, and KERNEL32.DLL)
  - Translate Windows API functions into calls to NTOSKRNL.EXE and WIN32K.SYS.
- Graphics device drivers
  - graphics display drivers, printer drivers, video miniport drivers

Prior to Windows NT 4.0, the window manager and graphics services were part of the user-mode Win32 subsystem process.

Is Windows Less Stable with Win32 USER and GDI in Kernel Mode?

# Executive

- Upper layer of the operating system
- Provides “generic operating system” functions (“services”)
  - Process Manager
  - Object Manager
  - Cache Manager
  - LPC (local procedure call) Facility
  - Configuration Manager
  - Memory Manager
  - Security Reference Monitor
  - I/O Manager
  - Power Manager
  - Plug-and-Play Manager
- Almost completely portable C code
- Runs in kernel (“privileged”, ring 0) mode
- Most interfaces to executive services not documented

# Kernel

- Lower layers of the operating system
  - Implements processor-dependent functions (x86 vs. Itanium etc.)
  - Also implements many processor-independent functions that are closely associated with processor-dependent functions
- Main services
  - Thread waiting, scheduling & context switching
  - Exception and interrupt dispatching
  - Operating system synchronization primitives (different for MP vs. UP)
  - A few of these are exposed to user mode
- Not a classic “microkernel”
  - shares address space with rest of kernel-mode components

# HAL - Hardware Abstraction Layer

- Responsible for a small part of “hardware abstraction”
  - Components on the motherboard not handled by drivers
    - System timers, Cache coherency, and flushing
    - SMP support, Hardware interrupt priorities
- Subroutine library for the kernel & device drivers
  - Isolates Kernel and Executive from platform-specific details
  - Presents uniform model of I/O hardware interface to drivers
- Reduced role as of Windows 2000
  - Bus support moved to bus drivers
  - Majority of HALs are vendor-independent

# HAL - Hardware Abstraction Layer (Cont'd)

- HAL also implements some functions that appear to be in the Executive and Kernel
- Selected at installation time
  - See \windows\repair\setup.log to find out which one
  - Can select manually at boot time with /HAL= in boot.ini
- HAL kit
  - Special kit only for vendors that must write custom HALs (requires approval from Microsoft)
  - see http://www.microsoft.com/whdc/ddk/HALkit/default.mspx

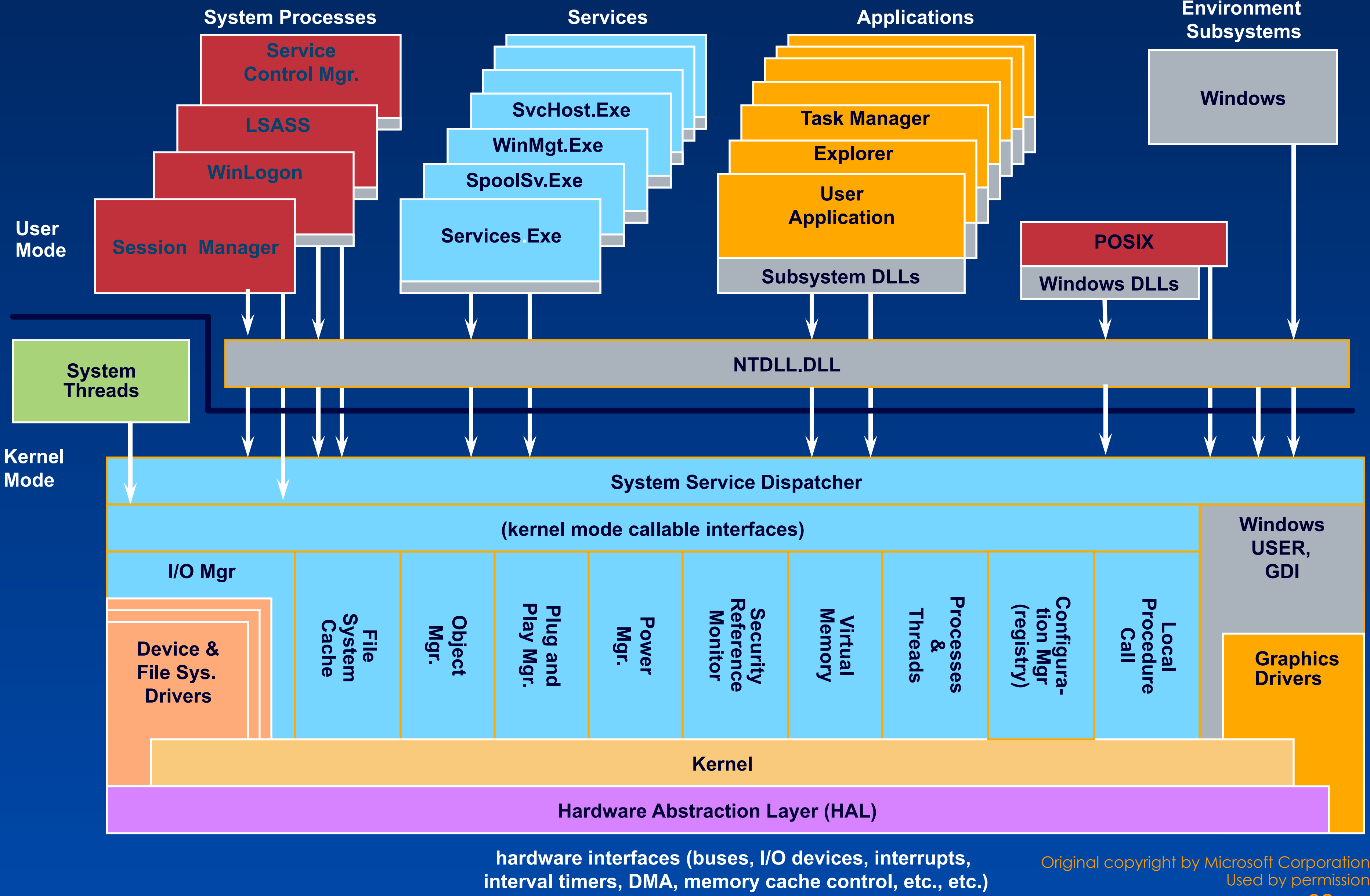
**Sample HAL routines:**

```
HalGetInterruptVector
HalGetAdapter
WRITE_PORT_UCHAR
```

## Kernel-Mode Device Drivers

- Separate loadable modules (drivername.SYS)
  - Linked like .EXEs
  - Typically linked against NTOSKRNL.EXE and HAL.DLL
  - Only one version of each driver binary for both uniprocessor (UP) and multiprocessor (MP) systems…
  - … but drivers call routines in the kernel that behave differently for UP vs. MP Versions
- Defined in registry
  - Same area as Windows services (t.b.d.) - differentiated by Type value
- Several types:
  - "ordinary", file system, NDIS miniport, SCSI miniport (linked against port drivers), bus drivers
  - More information in I/O subsystem section
- To view loaded drivers, run drivers.exe
  - Also see list at end of output from pstat.exe – includes addresses of each driver
- To update & control:
  - System properties→Hardware Tab→Device Manager
  - Computer Management→Software Environment→Drivers

# Windows Architecture



## Background System Processes

- *Core system processes,*
  - logon process, the session manager, etc.
  - not started by the service control manager
- *Service processes*
  - Host Windows services
  - i.e.; Task Scheduler and Spooler services
  - Many Windows server applications, such as Microsoft SQL Server and Microsoft Exchange Server, also include components that run as services.

## System Threads

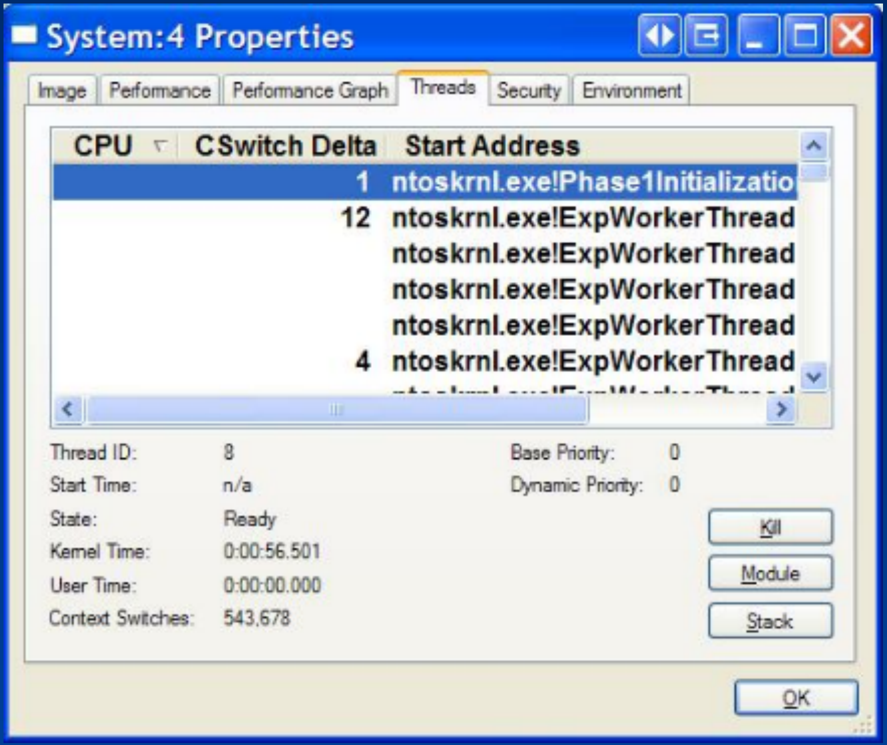
- Functions in OS and some drivers that need to run as real threads
  - E.g., need to run concurrently with other system activity, wait on timers, perform background "housekeeping" work
  - Always run in kernel mode
  - *Preemptible* (unless they raise IRQL to 2 or above)
  - For details, see DDK documentation on PsCreateSystemThread
- What process do they appear in?
  - "System" process (NT4: PID 2, W2K: PID 8, XP: PID 4)
  - In Windows 2000 & later, windowing system threads (from Win32k.sys) appear in "csrss.exe" (Windows subsystem process)

# Examples of System Threads

- Memory Manager
  - Modified Page Writer for mapped files
  - Modified Page Writer for paging files
  - Balance Set Manager
  - Swapper (kernel stack, working sets)
  - Zero page thread (thread 0, priority 0)
- Security Reference Monitor
  - Command Server Thread
- Network
  - Redirector and Server Worker Threads
- Threads created by drivers for their exclusive use
  - Examples: Floppy driver, parallel port driver
- Pool of Executive Worker Threads
  - Used by drivers, file systems, …
  - Work queued using ExQueueWorkItem
  - System thread (ExpWorkerThreadBalanceManager) manages pool

# Identifying System Threads: Process Explorer

- With Process Explorer:
  - Double click on System process
  - Go to Threads tab – sort by CPU time
- As explained before, threads run between clock ticks (or at high IRQL) and thus don't appear to run
  - Sort by context switch delta column
  - Cswitch Delta: number of context switches per Process Explorer refresh interval



| CPU | CSwitch Delta | Start Address |
|---|---|---|
| | 1 | ntoskrnl.exe!Phase1Initializatio |
| | 12 | ntoskrnl.exe!ExpWorkerThread |
| | | ntoskrnl.exe!ExpWorkerThread |
| | | ntoskrnl.exe!ExpWorkerThread |
| | | ntoskrnl.exe!ExpWorkerThread |
| | 4 | ntoskrnl.exe!ExpWorkerThread |

| | | | |
|---|---|---|---|
| Thread ID: | 8 | Base Priority: | 0 |
| Start Time: | n/a | Dynamic Priority: | 0 |
| State: | Ready | | |
| Kernel Time: | 0:00:56.501 | | |
| User Time: | 0:00:00.000 | | |
| Context Switches: | 543,678 | | |

# Process-Based Code

- OS components that run in separate executables (.exe's), in their own processes
  - Started by system
  - Not tied to a user logon
- Three types:
  - Environment Subsystems (already described)
  - System startup processes
    - note, "system startup processes" is not an official MS-defined name
  - Windows Services
- Let's examine the system process "tree"
  - Use Tlist /T or Process Explorer

## System Startup Processes

- First two processes aren't real processes
  - not running a user mode .EXE
  - no user-mode address space
  - different utilities report them with different names
  - data structures for these processes (and their initial threads) are "pre-created" in NtosKrnl.Exe and loaded along with the code

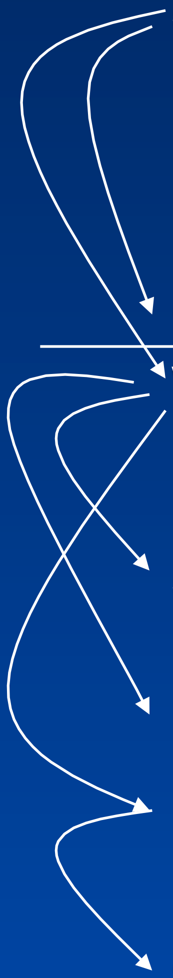
(Idle) Process id 0

Part of the loaded system image
Home for idle thread(s) (not a real process nor real threads)
Called "System Process" in many displays

(System) Process id 2 *(8 in Windows 2000; 4 in XP)*

Part of the loaded system image
Home for kernel-defined threads (not a real process)
Thread 0 (routine name Phase1 Initialization)
launches the first "real" process, running smss.exe...
...and then becomes the zero page thread

# System Startup Processes (cont.)



| | |
|---|---|
| smss.exe | Session Manager<br>The first “created” process<br>Takes parameters from \HKEY_LOCAL_MACHINE\System \CurrentControlSet<br>\Control\Session Manager<br>Launches required subsystems (csrss) and then winlogon |
| csrss.exe | Windows subsystem |
| winlogon.exe | Logon process: Launches services.exe & lsass.exe; presents first login prompt<br>When someone logs in, launches apps in \Software\Microsoft\Windows NT\WinLogon\Userinit |
| services.exe | Service Controller; also, home for many Windows-supplied services<br>Starts processes for services not part of services.exe (driven by \Registry \Machine\System\CurrentControlSet\Services ) |
| lsass.exe | Local Security Authentication Server |
| userinit.exe | Started after logon; starts Explorer.exe (see \Software\Microsoft\Windows NT\CurrentVersion\WinLogon\Shell) and exits (hence Explorer appears to be an orphan) |
| explorer.exe | and its children are the creators of all interactive apps |

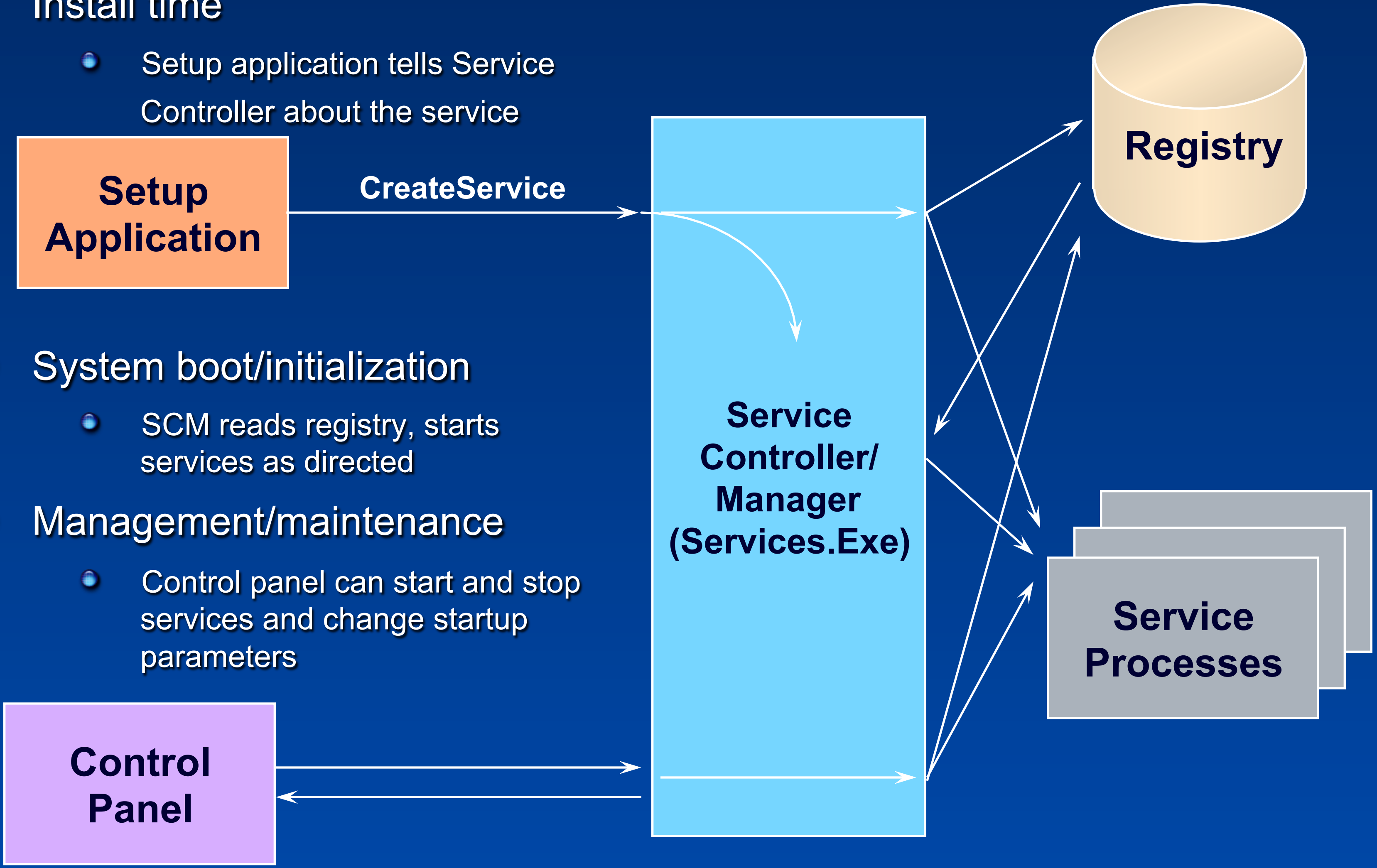
## Where are Services Defined?

- Defined in the registry:

  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
  - one key per installed service
- Mandatory information kept on each service:
  - Type of service (Windows, Driver, ...)
  - Imagename of service .EXE
    - Note: some .EXEs contain more than one service
  - Start type (automatic, manual, or disabled)
- Optional information:
  - Display Name
  - New in W2K: Description
  - Dependencies
  - Account & password to run under
- Can store application-specific configuration parameters
  - “Parameters” subkey under service key

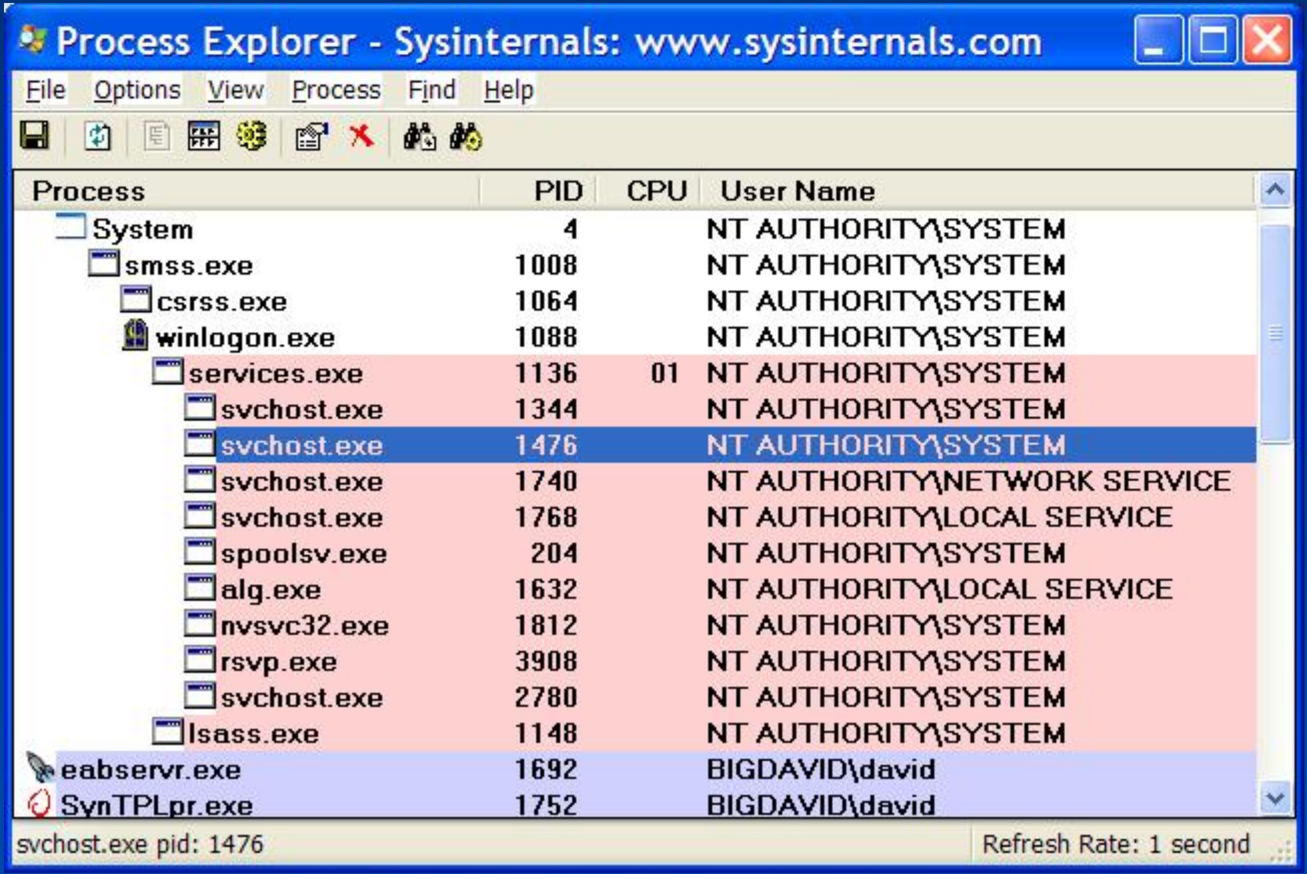
# Life of a Service

- Install time
  - Setup application tells Service Controller about the service
- System boot/initialization
  - SCM reads registry, starts services as directed
- Management/maintenance
  - Control panel can start and stop services and change startup parameters



# Process Explorer: Service Information

- Process Explorer identifies Service Processes
  - Click on Options->Highlight Services


Process Explorer - Sysinternals: www.sysinternals.com
File Options View Process Find Help

| Process | PID | CPU | User Name |
|---|---|---|---|
| System | 4 | | NT AUTHORITY\SYSTEM |
| smss.exe | 1008 | | NT AUTHORITY\SYSTEM |
| csrss.exe | 1064 | | NT AUTHORITY\SYSTEM |
| winlogon.exe | 1088 | | NT AUTHORITY\SYSTEM |
| services.exe | 1136 | 01 | NT AUTHORITY\SYSTEM |
| svchost.exe | 1344 | | NT AUTHORITY\SYSTEM |
| svchost.exe | 1476 | | NT AUTHORITY\SYSTEM |
| svchost.exe | 1740 | | NT AUTHORITY\NETWORK SERVICE |
| svchost.exe | 1768 | | NT AUTHORITY\LOCAL SERVICE |
| spoolsv.exe | 204 | | NT AUTHORITY\SYSTEM |
| alg.exe | 1632 | | NT AUTHORITY\LOCAL SERVICE |
| nvsvc32.exe | 1812 | | NT AUTHORITY\SYSTEM |
| rsvp.exe | 3908 | | NT AUTHORITY\SYSTEM |
| svchost.exe | 2780 | | NT AUTHORITY\SYSTEM |
| lsass.exe | 1148 | | NT AUTHORITY\SYSTEM |
| eabservr.exe | 1692 | | BIGDAVID\david |
| SynTPLpr.exe | 1752 | | BIGDAVID\david |

svchost.exe pid: 1476 Refresh Rate: 1 second

## Service Processes

- A process created & managed by the Service Control Manager (Services.exe)
  - Similar in concept to Unix daemon processes
  - Typically configured to start at boot time (if started while logged on, survive logoff)
  - Typically do not interact with the desktop
- Note:  Prior to Windows 2000 this was the only way to start a process on a remote machine
  - Now you can do it with WMI (Win Management Instrumentation)

# Service Control Tools

- Net start/stop – local system only
- Sc.exe (built in to XP/2003; also in Win2000 Resource Kit)
  - Command line interface to all service control/configuration functions
  - Works on local or remote systems
- Psservice (Sysinternals) – similar to SC
- Other tools in Resource Kit
  - Instsrv.exe – install/remove services (command line)
  - Srvinstw.exe – install/remove services (GUI)
  - Why are service creation tools included in Reskit?
    - Because Reskit comes with several services that are not installed as services when you install the Reskit

# Services Infrastructure

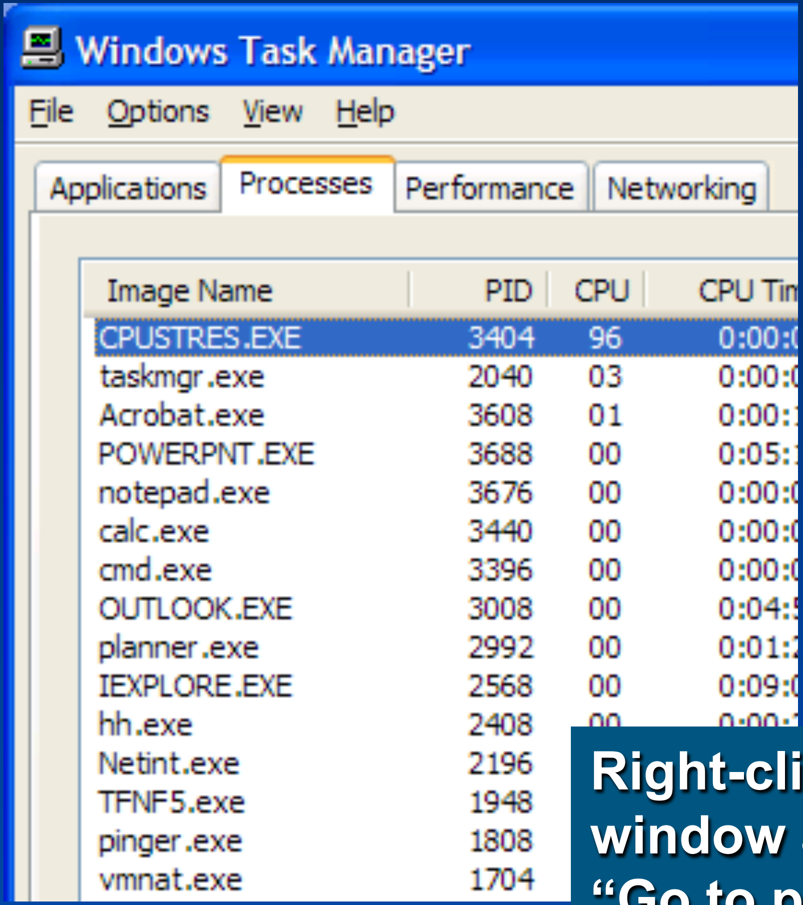
- Windows 2000 introduced generic Svchost.exe
  - Groups services into fewer processes
    - Improves system startup time
    - Conserves system virtual memory
  - Not user-configurable as to which services go in which processes
  - 3rd parties cannot add services to Svchost.exe processes
- Windows XP/2003 have more Svchost processes due to two new less privileged accounts for built-in services
  - LOCAL SERVICE, NETWORK SERVICE
  - Less rights than SYSTEM account
    - Reduces possibility of damage if system compromised
- On XP/2003, four Svchost processes (at least):
  - SYSTEM
  - SYSTEM (2nd instance – for RPC)
  - LOCAL SERVICE
  - NETWORK SERVICE

## Lab: Which version of NTOSKRNL?

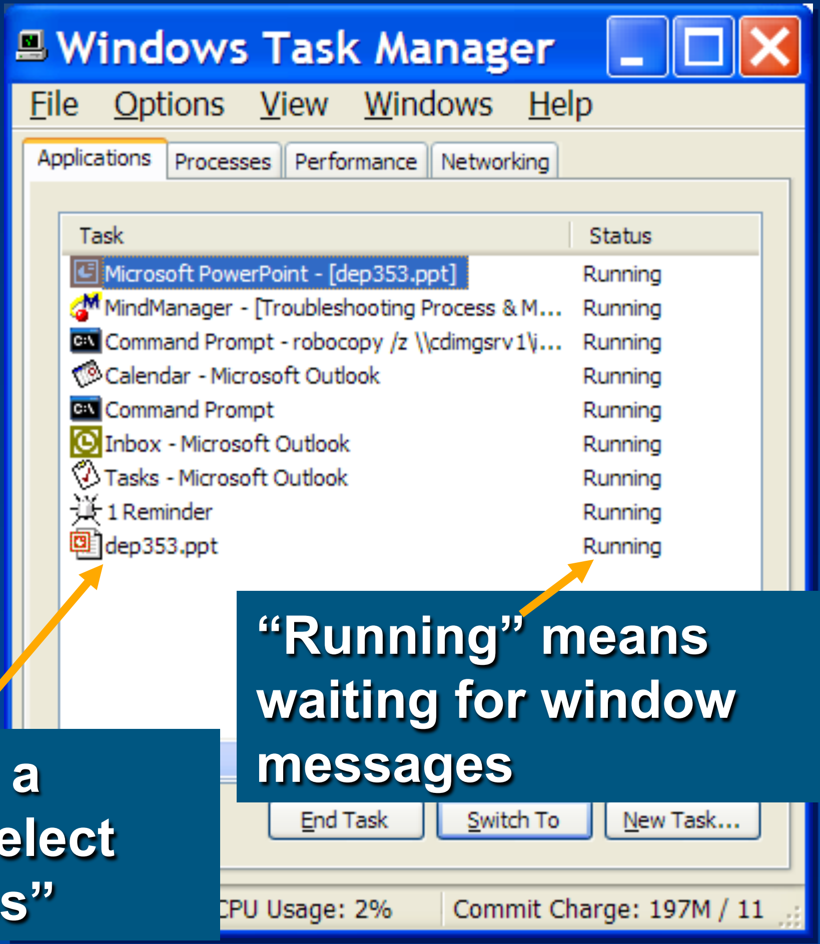
- Programs/ Administrative Tools/Event Viewer
  - Select System Log
  - Double-click an Event Log  entry with an Event ID of 6009
- Check if it’s booted with PAE version(win2000 or winXp):
  - Registry entry: HKLM\SYSTEM\CurrentControlSet\Control \SystemStartOptions
- Uniprocess or multiprocessor:
  - right-click on  Ntoskrnl.exe in your \Windows\System32 folder, and select Properties

# Lab: Task Manager (Set Priority, etc. )

- Processes tab: List of processes
- Applications tab: List of top level visible windows



| Image Name | PID | CPU | CPU Tim |
|---|---|---|---|
| CPUSTRES.EXE | 3404 | 96 | 0:00:0 |
| taskmgr.exe | 2040 | 03 | 0:00:0 |
| Acrobat.exe | 3608 | 01 | 0:00:1 |
| POWERPNT.EXE | 3688 | 00 | 0:05:1 |
| notepad.exe | 3676 | 00 | 0:00:0 |
| calc.exe | 3440 | 00 | 0:00:0 |
| cmd.exe | 3396 | 00 | 0:00:0 |
| OUTLOOK.EXE | 3008 | 00 | 0:04:5 |
| planner.exe | 2992 | 00 | 0:01:2 |
| IEXPLORE.EXE | 2568 | 00 | 0:09:0 |
| hh.exe | 2408 | 00 | 0:00:1 |
| Netint.exe | 2196 | | |
| TFNF5.exe | 1948 | | |
| pinger.exe | 1808 | | |
| vmnat.exe | 1704 | | |


Windows Task Manager
File Options View Windows Help
Applications Processes Performance Networking
End Task
Switch To
New Task...

| Task | Status |
|---|---|
| Microsoft PowerPoint - [dep353.ppt] | Running |
| MindManager - [Troubleshooting Process & M... | Running |
| Command Prompt - robocopy /z \\cdimgsrv1\i... | Running |
| Calendar - Microsoft Outlook | Running |
| Command Prompt | Running |
| Inbox - Microsoft Outlook | Running |
| Tasks - Microsoft Outlook | Running |
| 1 Reminder | Running |
| dep353.ppt | Running |

CPU Usage: 2% Commit Charge: 197M / 11

**"Running" means waiting for window messages**

**Right-click on a window and select "Go to process"**

# Lab: Identifying System Threads in the System Process (Process Explorer)

- You can see that the threads inside the System process must be kernel-mode system threads because the start address for each thread is greater than the start address of system space (which by default begins at 0x80000000, unless the system was booted with the /3GB Boot.ini switch).
- Also, if you look at the CPU time for these threads, you'll see that those that have accumulated any CPU time have run only in kernel mode. To find out which driver created the system thread, look up the start address of the thread (which you can display with Pviewer.exe) and look for the driver whose base address is closest to (but before) the start address of the thread. Both the Pstat utility (at the end of its output) as well as the !drivers kernel debugger command list the base address of each loaded device driver.

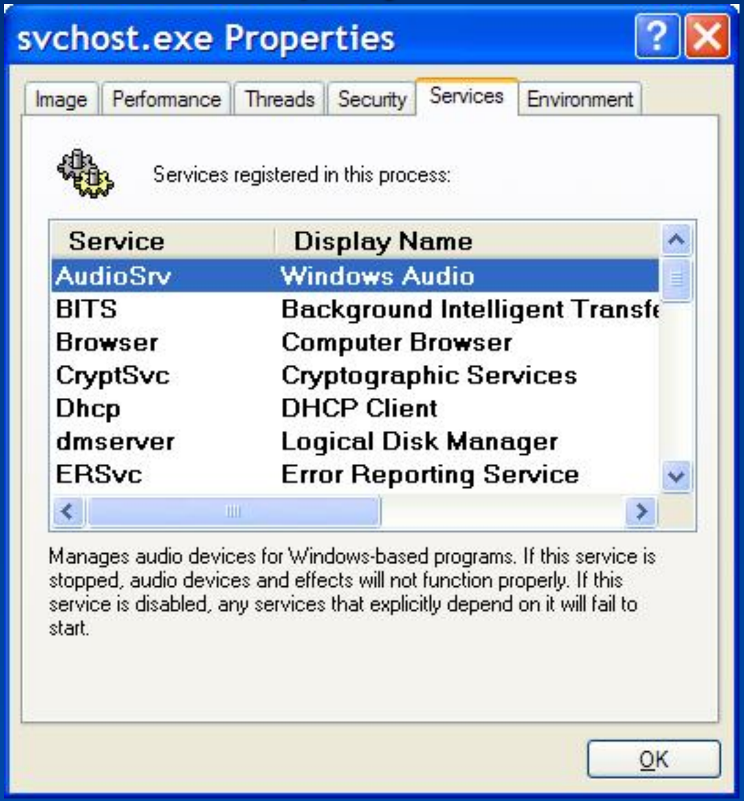
# Lab: Determining If You Are Running the Checked(debug) Build

- There is no built-in tool.
- However, this information is available through the "Debug" property of the Windows Management Instrumentation (WMI) Win32_OperatingSystem class. (wbemtest.exe)
- The following sample Visual Basic script displays this property:

```
strComputer= "."
Set objWMIService = GetObject("winmgmts:" &
"{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set colOperatingSystems = objWMIService.ExecQuery ("SELECT * FROM
Win32_OperatingSystem")
For Each objOperatingSystem in colOperatingSystems
  wscript.Echo "Caption: " & objOperatingSystem.Caption
  wscript.Echo "Debug: " & objOperatingSystem.Debug
  wscript.Echo "Version: " & objOperatingSystem.Version
  NEXT
```

# Lab: Mapping Services to Service Processes

- Tlist /S (Debugging Tools) or Tasklist /svc (XP/2003) list internal name of services inside service processes
- Process Explorer shows more: external display name and description



Services registered in this process:

| Service | Display Name |
|---|---|
| AudioSrv | Windows Audio |
| BITS | Background Intelligent Transfe |
| Browser | Computer Browser |
| CryptSvc | Cryptographic Services |
| Dhcp | DHCP Client |
| dmserver | Logical Disk Manager |
| ERSvc | Error Reporting Service |

Manages audio devices for Windows-based programs. If this service is stopped, audio devices and effects will not function properly. If this service is disabled, any services that explicitly depend on it will fail to start.

# Source Code References

- Windows Research Kernel sources
  - Processes
  - Threads
  - LPC
  - Virtual memory
  - Scheduler
  - Object manager
  - I/O manager
  - Synchronization
  - Worker threads
  - Kernel heap manager
  - Core components of Ntoskrnl.exe

# Source Code References (Cont'd)

- Does not include the code for these modules:
  - Windowing system driver (Csrss.exe, Win32k.sys)
  - Windows API DLLs (Kernel32, User32, Gdi32, etc)
  - Core system processes (Smss, Winlogon, Services, Lsass)

## Further Reading

- Mark E. Russinovich , *et al.*, Windows Internals, 5th Edition, Microsoft Press, 2009.
- Chapter 2 - System Architecture
    - Operating System Model (from pp. 34)
    - Architecture Overview (from pp. 35)
    - Key System Components (from pp. 49)