Windows Internals

PDC05
DEVELOPER POWERED

David Solomon (daves@solsem.com)
David Solomon Expert Seminars
www.solsem.com

Mark Russinovich (mark@sysinternals.com)
Winternals
www.winternals.com, www.sysinternals.com

About the Speaker: David Solomon

- 1982-1992: VMS operating systems development at Digital
- 1992-present: Researching, writing, and teaching Windows operating system internals
- Frequent speaker at technical conferences (Microsoft TechEd, IT Forum, PDCs, …)
- Microsoft Most Valuable Professional (1993, 2005)
- Books
  - *Windows Internals, 4th edition*
    - PDF version ships with Server 2003 Resource Kit
  - *Inside Windows 2000, 3rd edition*
  - *Inside Windows NT, 2nd edition*
  - *Windows NT for OpenVMS Professionals*
- Live Classes
  - 2-5 day classes on Windows Internals, Advanced Troubleshooting
- Video Training
  - 12 hour interactive internals tutorial
  - Licensed by MS for internal use

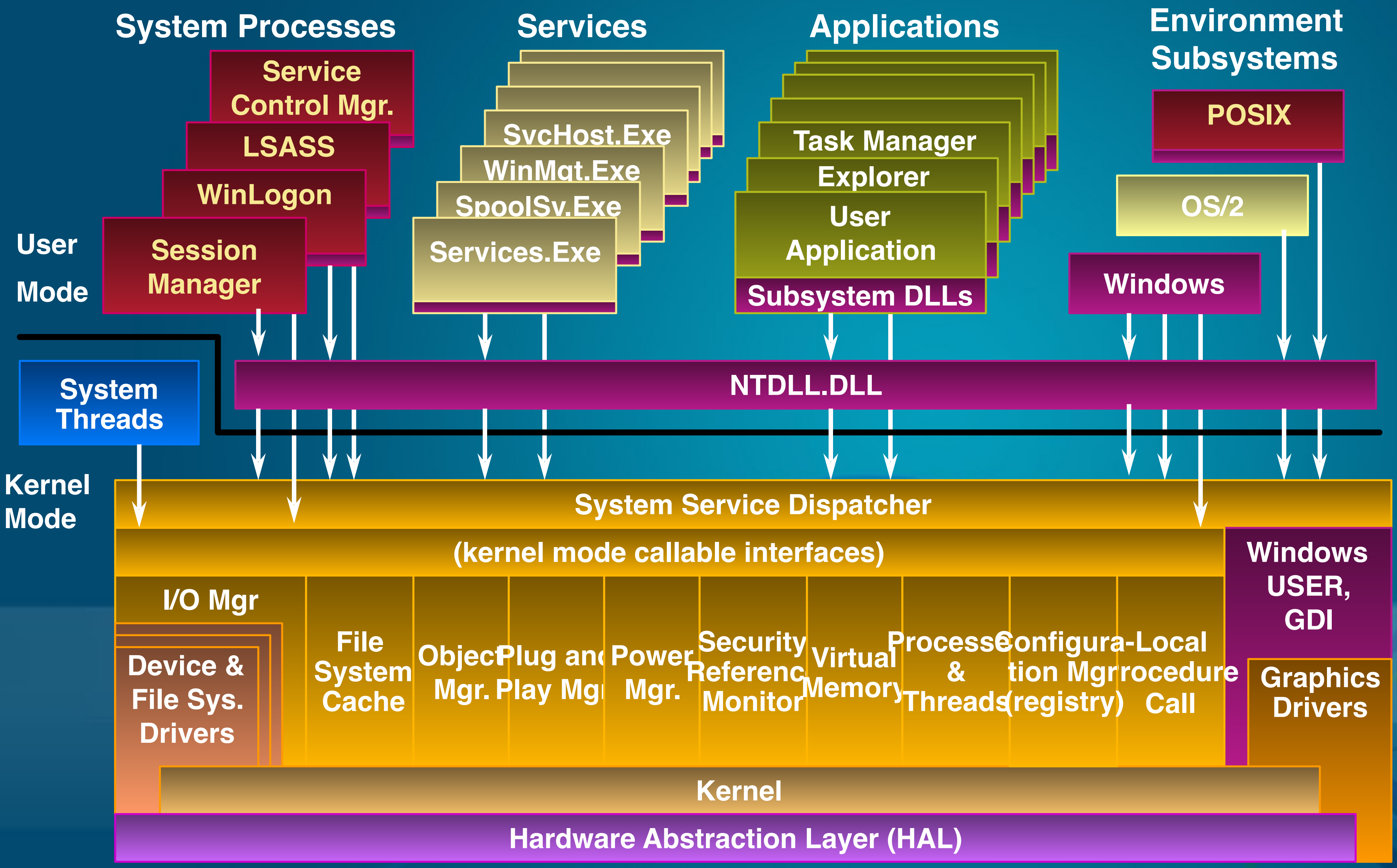
About the Speaker: Mark Russinovich

- Co-author of *Inside Windows 2000, 3rd Edition* and *Windows Internals, 4th edition* with David Solomon
- Senior Contributing Editor to Windows IT Pro Magazine
  - Co-authors Windows Power Tools column
- Author of tools on www.sysinternals.com
- Microsoft Most Valuable Professional (MVP)
- Co-founder and chief software architect of Winternals Software (www.winternals.com)
- Ph.D. in Computer Engineering

Purpose of Tutorial

- Give Windows developers a foundation understanding of the system’s kernel architecture
  - Design better for performance & scalability
  - Debug problems more effectively
  - Understand system performance issues
- We’re covering a small, but important set of core topics:
  - The “plumbing in the boiler room”

System Architecture



System Processes

Service Control Mgr.

LSASS

WinLogon

Session Manager

Services

SvcHost.Exe

WinMgt.Exe

SpoolSv.Exe

Services.Exe

Applications

Task Manager

Explorer

User Application

Subsystem DLLs

Environment Subsystems

POSIX

OS/2

Windows

User Mode

System Threads

NTDLL.DLL

Kernel Mode

System Service Dispatcher

(kernel mode callable interfaces)

I/O Mgr

Device & File Sys. Drivers

File System Cache

Object Mgr.

Plug and Play Mgr

Power Mgr.

Security Reference Monitor

Virtual Memory

Processes & Threads

Configura-tion Mgr (registry)

Local Procedure Call

Windows USER, GDI

Graphics Drivers

Kernel

Hardware Abstraction Layer (HAL)

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

Tools Used To Dig In

- Many tools available to dig into Windows OS internals without requiring source code
  - Helps to see internals behavior “in action”
  - Many of these tools are used in labs in the video and the book
- Several sources of tools
  - Support Tools (on Windows OS CD-ROM in \support\tools)
  - Resource Kit Tools
  - Sysinternals tools (www.sysinternals.com)
  - Windows Debugging Tools

Live Kernel Debugging

- Useful for investigating internal system state not available from other tools
  - Previously, required 2 computers (host and target)
  - Target would be halted while host debugger in use
- XP & later supports live local kernel debugging
  - Technically requires system to be booted / DEBUG to work correctly
  - But, not all commands work

LiveKD

- LiveKd makes more commands work on a live system
  - Works on NT4, Windows 2000, Windows XP, Server 2003, and Vista
  - Was originally shipped on *Inside Windows 2000* book CD-ROM – now is free on Sysinternals
  - Tricks standard Microsoft kernel debuggers into thinking they are looking at a crash dump
  - Does not guarantee consistent view of system memory
    - Thus can loop or fail with access violation
    - Just quit and restart

Outline

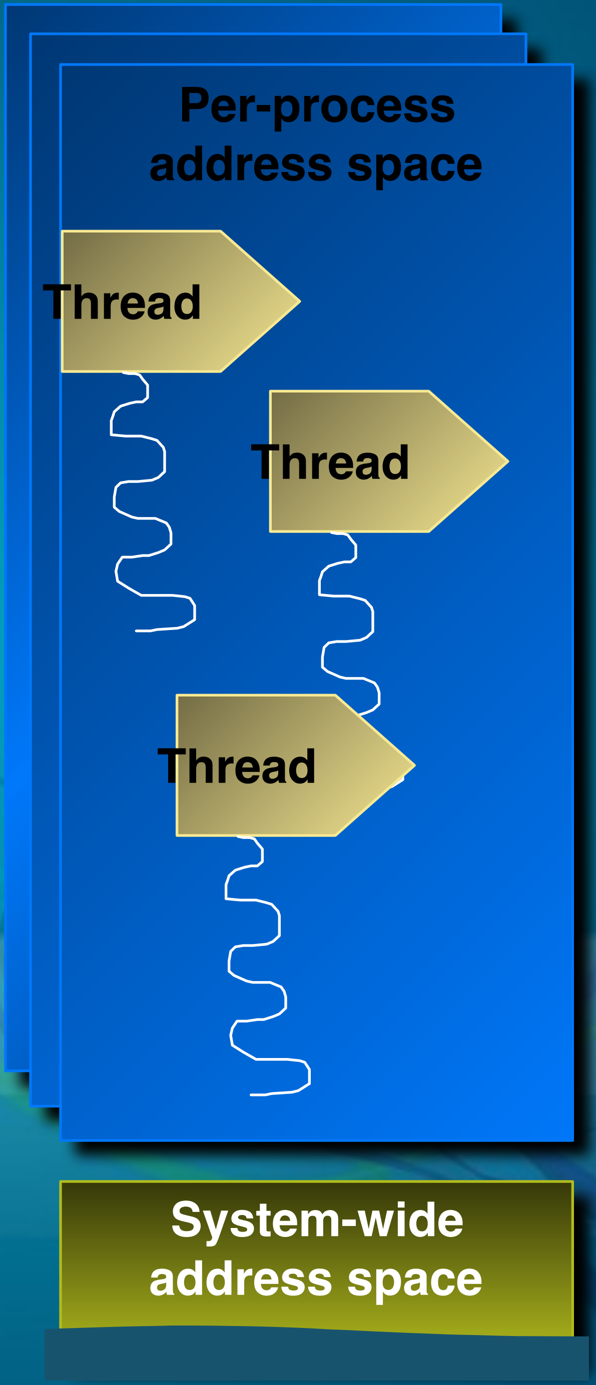
1. System Architecture
2. Processes and Thread Internals
3. Memory Management Internals
4. Security Internals

System Architecture

- Process Execution Environment
- Kernel Architecture
- Interrupt Handling
- Object Manager
- System Threads
- Process-based code
- Summary

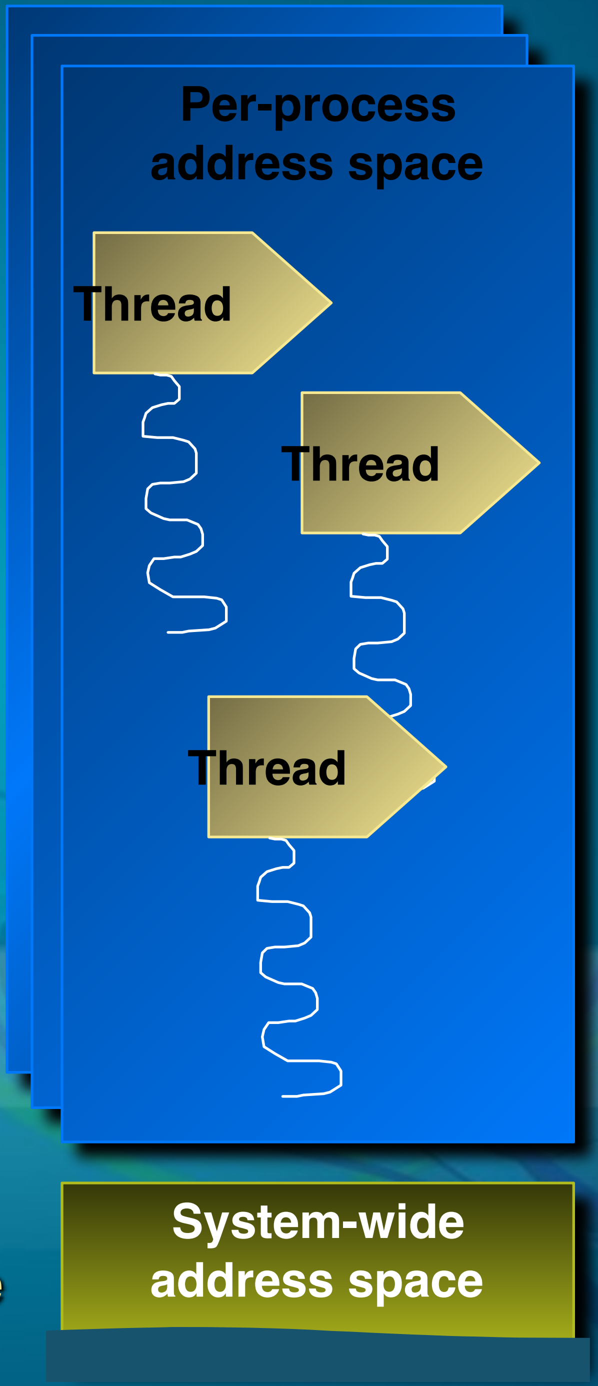
Processes And Threads

- What is a process?
  - Represents an instance of a running program
    - You create a process to run a program
    - Starting an application creates a process
  - Process defined by
    - Address space
    - Resources (e.g., open handles)
    - Security profile (token)
- System call
  - Primary argument to CreateProcess is image file name (or command line)



Processes And Threads

- What is a thread?
  - An execution context within a process
  - Unit of scheduling (threads run, processes don't run)
  - All threads in a process share the same per-process address space
    - Services provided so that threads can synchronize access to shared resources (critical sections, mutexes, events, semaphores)
  - All threads in the system are scheduled as peers to all others, without regard to their "parent" process
- System call:
  - Primary argument to CreateThread is a function entry point address
- Linux:
  - No threads per-se
  - Tasks can act like Windows threads by sharing handle table, PID and address space

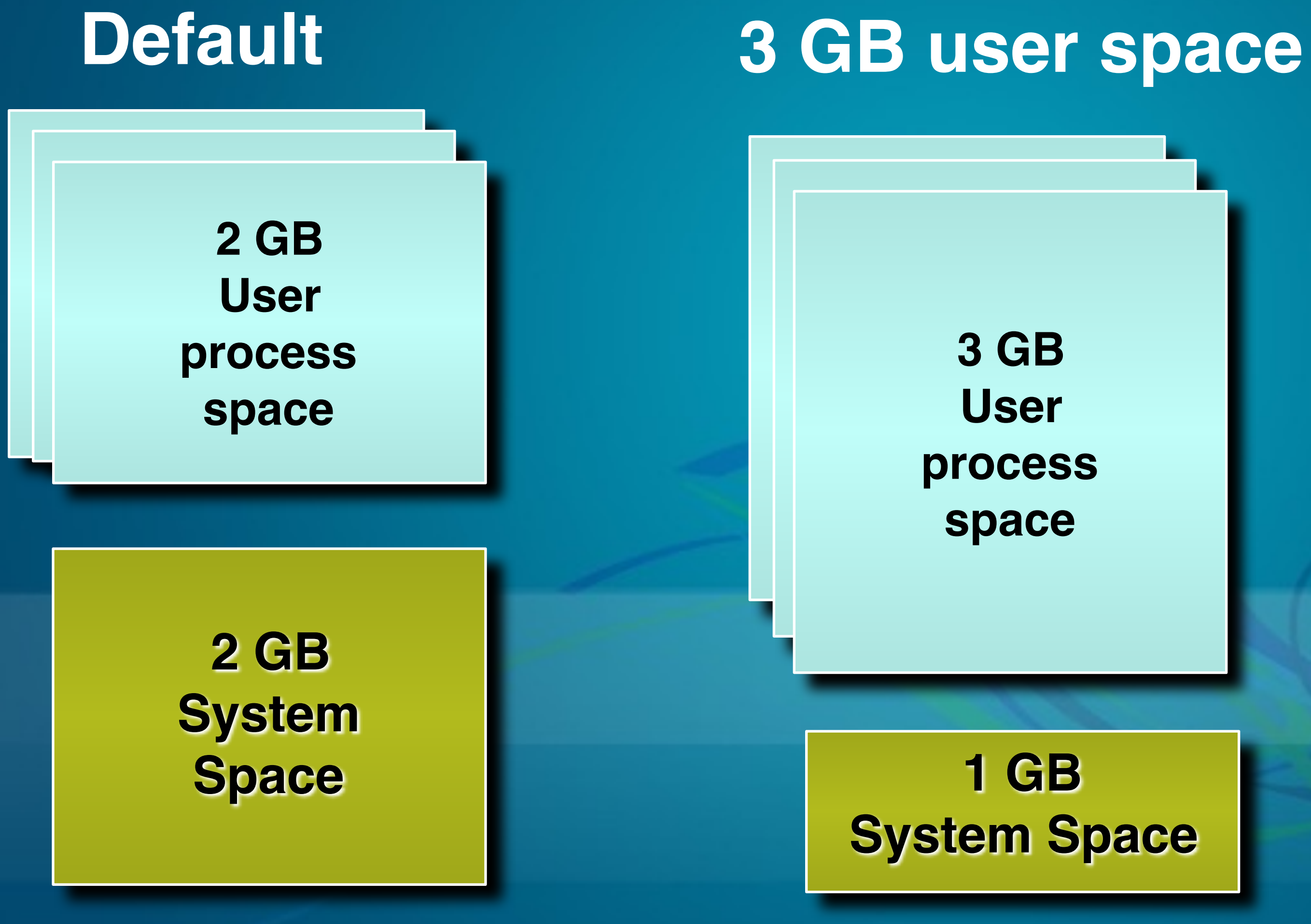


Processes And Threads

- Every process starts with one thread
  - First thread executes the program’s “main” function
    - Can create other threads in the same process
    - Can create additional processes
- Why divide an application into multiple threads?
  - Perceived user responsiveness, parallel/background execution
    - Examples: Word background print – can continue to edit during print
  - Take advantage of multiple processors
    - On an MP system with n CPUs, n threads can literally run at the same time
    - Question: Given a single threaded application, will adding a second processor make it run faster?
  - Does add complexity
    - Synchronization
    - Scalability well is a different question…
      - Number of multiple runnable threads versus number CPUs
      - Having too many runnable threads causes excess context switching

32-bit x86 Address Space

- 32-bits = 4 GB



Default

2 GB
User
process
space

2 GB
System
Space

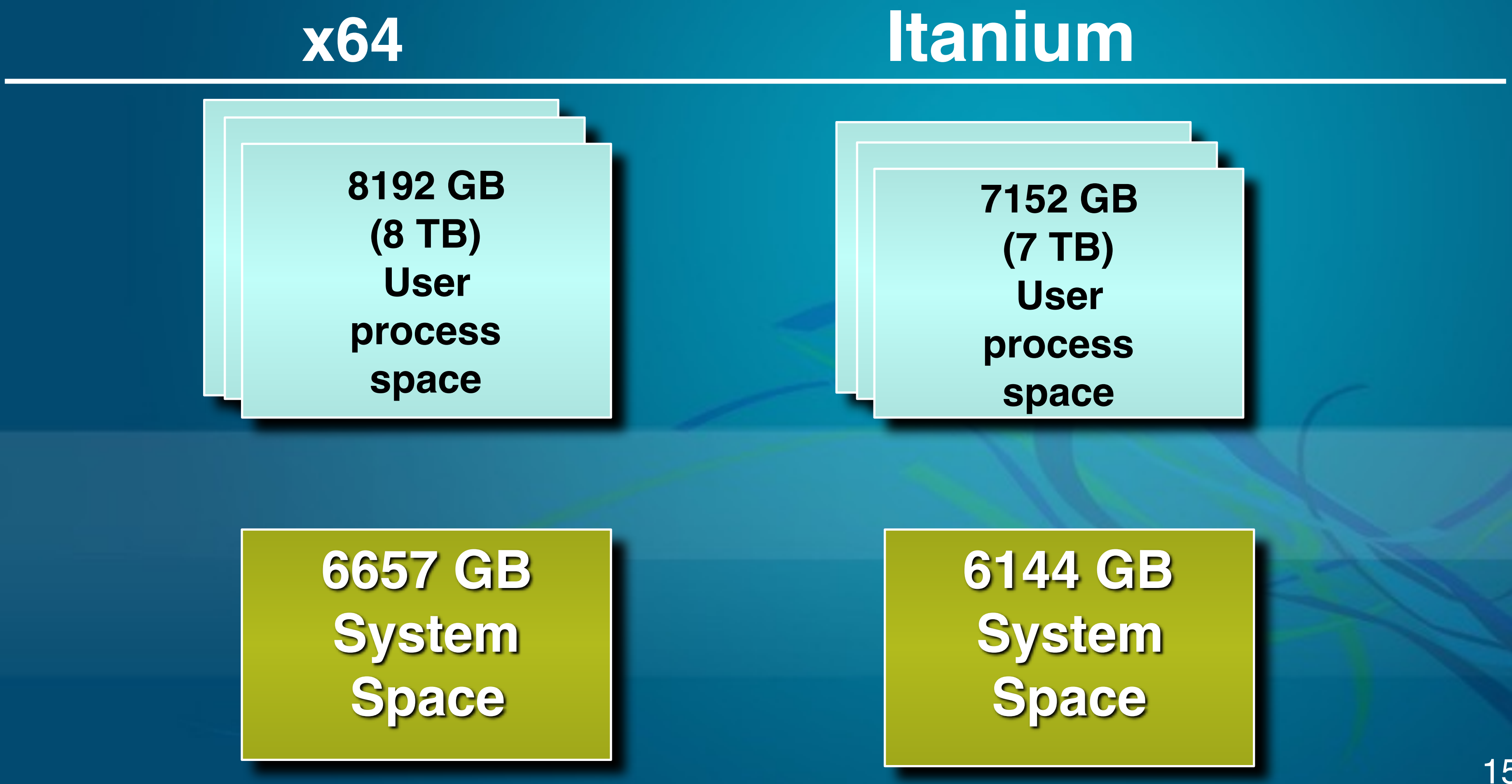
3 GB user space

3 GB
User
process
space

1 GB
System Space

64-bit Address Spaces

- 64-bits = 17,179,869,184 GB
  - x64 today supports 48 bits virtual = 262,144 GB
  - IA-64 today support 50 bits virtual = 1,048,576 GB



x64

8192 GB
(8 TB)
User
process
space

6657 GB
System
Space

Itanium

7152 GB
(7 TB)
User
process
space

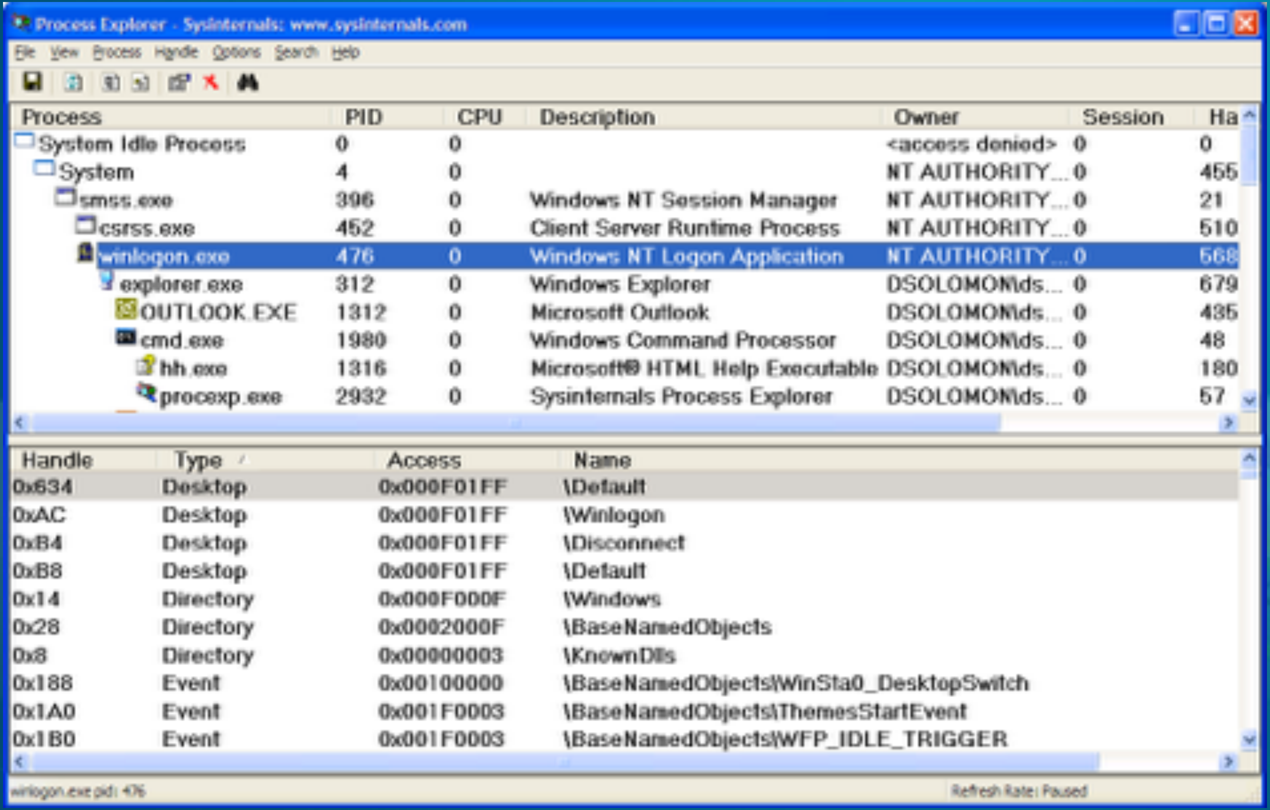
6144 GB
System
Space

Memory Protection Model

- No user process can touch another user process address space (without first opening a handle to the process, which means passing through NT security)
  - Separate process page tables prevent this
  - “Current” page table changed on context switch from a thread in 1 process to a thread in another process
- No user process can touch kernel memory
  - Page protection in process page tables prevent this
  - OS pages only accessible from “kernel mode”
    - x86: Ring 0, Itanium: Privilege Level 0
  - Threads change from user to kernel mode and back (via a secure interface) to execute kernel code
    - Does not affect scheduling (not a context switch)

Process Explorer (Sysinternals)

- "Super Task Manager"
  - Shows full image path, command line, environment variables, parent process, thread details, security access token, open handles, loaded DLLs & mapped files


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

Process	PID	CPU	Description	Owner	Session	Ha
System Idle Process	0	0		<access denied>	0	0
System	4	0		NT AUTHORITY...	0	455
smss.exe	396	0	Windows NT Session Manager	NT AUTHORITY...	0	21
csrss.exe	452	0	Client Server Runtime Process	NT AUTHORITY...	0	510
winlogon.exe	476	0	Windows NT Logon Application	NT AUTHORITY...	0	568
explorer.exe	312	0	Windows Explorer	DSOLOMON\ds...	0	679
OUTLOOK.EXE	1312	0	Microsoft Outlook	DSOLOMON\ds...	0	435
cmd.exe	1980	0	Windows Command Processor	DSOLOMON\ds...	0	48
hh.exe	1316	0	Microsoft® HTML Help Executable	DSOLOMON\ds...	0	180
procexp.exe	2932	0	Sysinternals Process Explorer	DSOLOMON\ds...	0	57

Handle	Type	Access	Name
0x634	Desktop	0x000F01FF	\Default
0xAC	Desktop	0x000F01FF	\Winlogon
0xB4	Desktop	0x000F01FF	\Disconnect
0xB8	Desktop	0x000F01FF	\Default
0x14	Directory	0x000F000F	\Windows
0x28	Directory	0x0002000F	\BaseNamedObjects
0x8	Directory	0x00000003	\KnownDlls
0x188	Event	0x00100000	\BaseNamedObjects\WinSta0_DesktopSwitch
0x1A0	Event	0x001F0003	\BaseNamedObjects\ThemesStartEvent
0x1B0	Event	0x001F0003	\BaseNamedObjects\WFP_IDLE_TRIGGER

winlogon.exe pid: 476

Refresh Rate: Paused

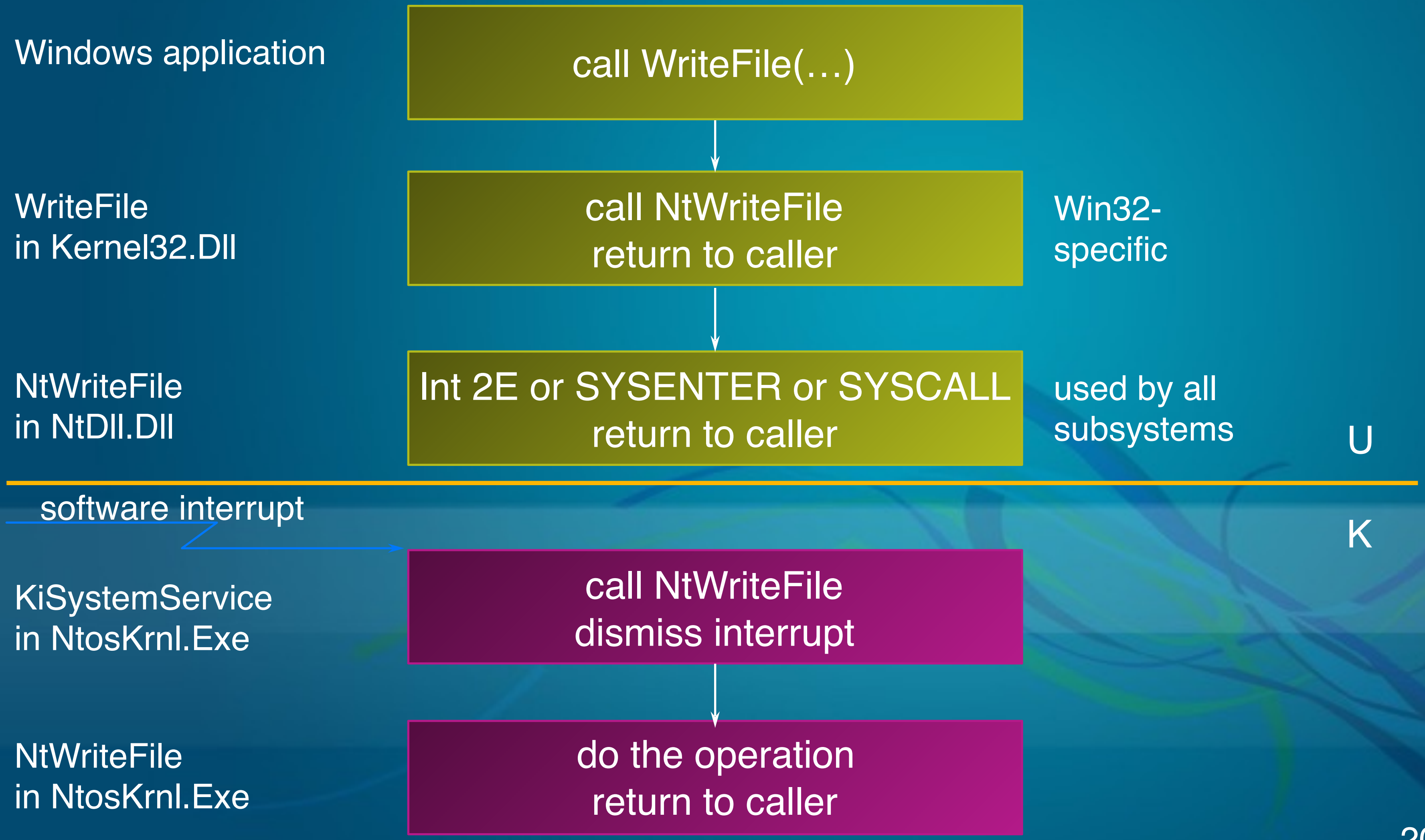
System Architecture

- Process Execution Environment
- Kernel Architecture
- Interrupt Handling
- Object Manager
- System Threads
- Process-based code
- Summary

Windows Kernel Evolution

- Basic kernel architecture has remained stable while system has evolved
  - Windows 2000: major changes in I/O subsystem (plug & play, power management, WDM), but rest similar to NT4
  - Windows XP & Server 2003: modest upgrades as compared to the changes from NT4 to Windows 2000
- Internal version numbers confirm this:
  - Windows 2000 was 5.0
  - Windows XP is 5.1
  - Windows Server 2003 is 5.2
  - Windows Vista is 6.0

Example
Invoking a Win32 Kernel API



Windows application

call WriteFile(…)

WriteFile
in Kernel32.Dll

call NtWriteFile
return to caller

Win32-
specific

NtWriteFile
in NtDll.Dll

Int 2E or SYSENTER or SYSCALL
return to caller

used by all
subsystems

U

software interrupt

K

KiSystemService
in NtosKrnl.Exe

call NtWriteFile
dismiss interrupt

NtWriteFile
in NtosKrnl.Exe

do the operation
return to caller

NTOSKRNL.EXE

- Core operating system image
  - Contains Executive and Kernel
- Four retail variations:

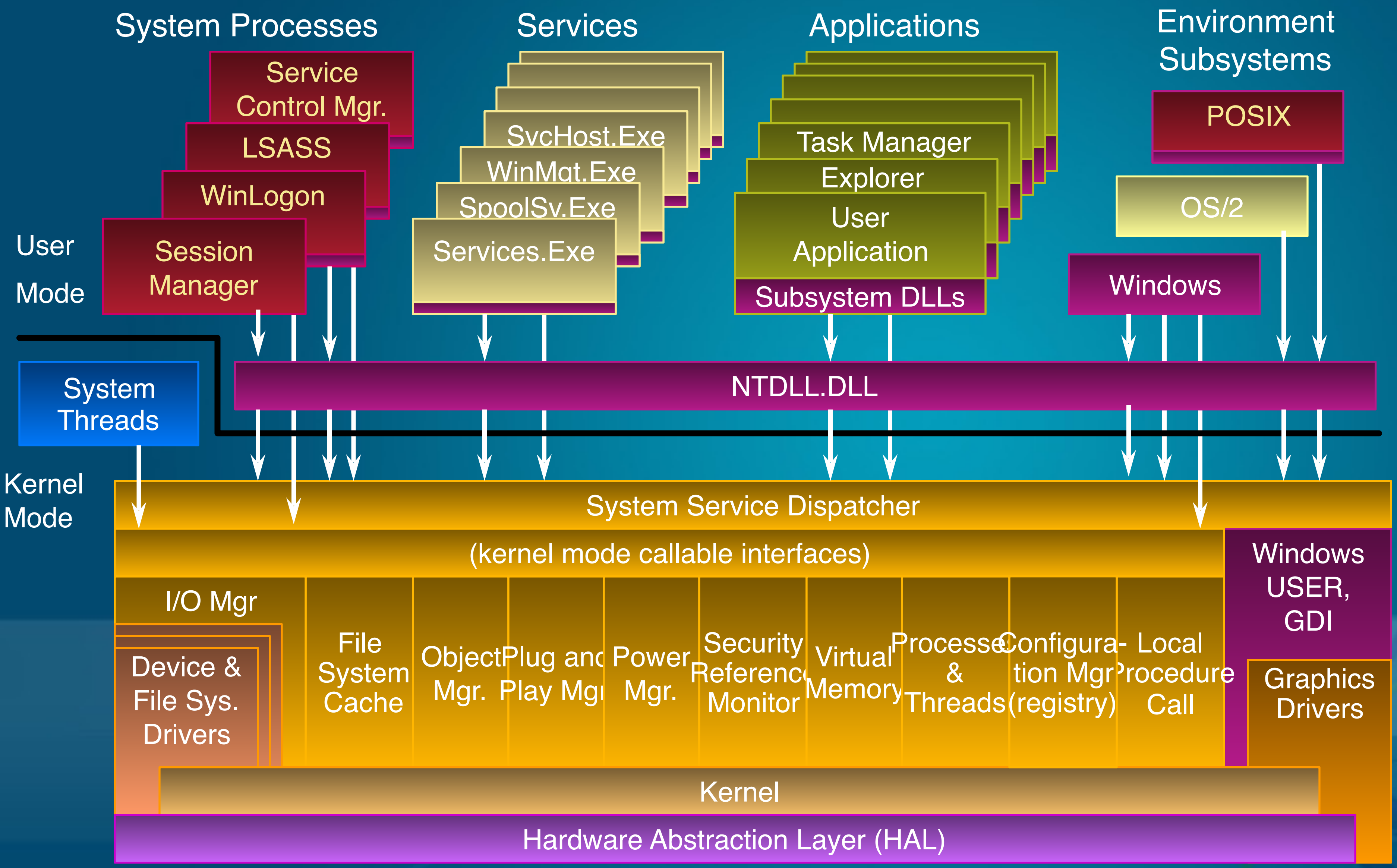
NTOSKRNL.EXE	Uniprocessor
NTKRNLMP.EXE	Multiprocessor

32-bit Windows PAE versions (for DEP & >4GB RAM):

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

- Vista: no uniprocessor kernel

System Architecture



System Processes

Service Control Mgr.

LSASS

WinLogon

Session Manager

Services

SvcHost.Exe

WinMgt.Exe

SpoolSv.Exe

Services.Exe

Applications

Task Manager

Explorer

User Application

Subsystem DLLs

Environment Subsystems

POSIX

OS/2

Windows

User Mode

System Threads

NTDLL.DLL

Kernel Mode

System Service Dispatcher

(kernel mode callable interfaces)

I/O Mgr

Device & File Sys. Drivers

File System Cache

Object Mgr.

Plug and Play Mgr

Power Mgr.

Security Reference Monitor

Virtual Memory

Processes & Threads

Configura- tion Mgr (registry)

Local Procedure Call

Windows USER, GDI

Graphics Drivers

Kernel

Hardware Abstraction Layer (HAL)

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

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

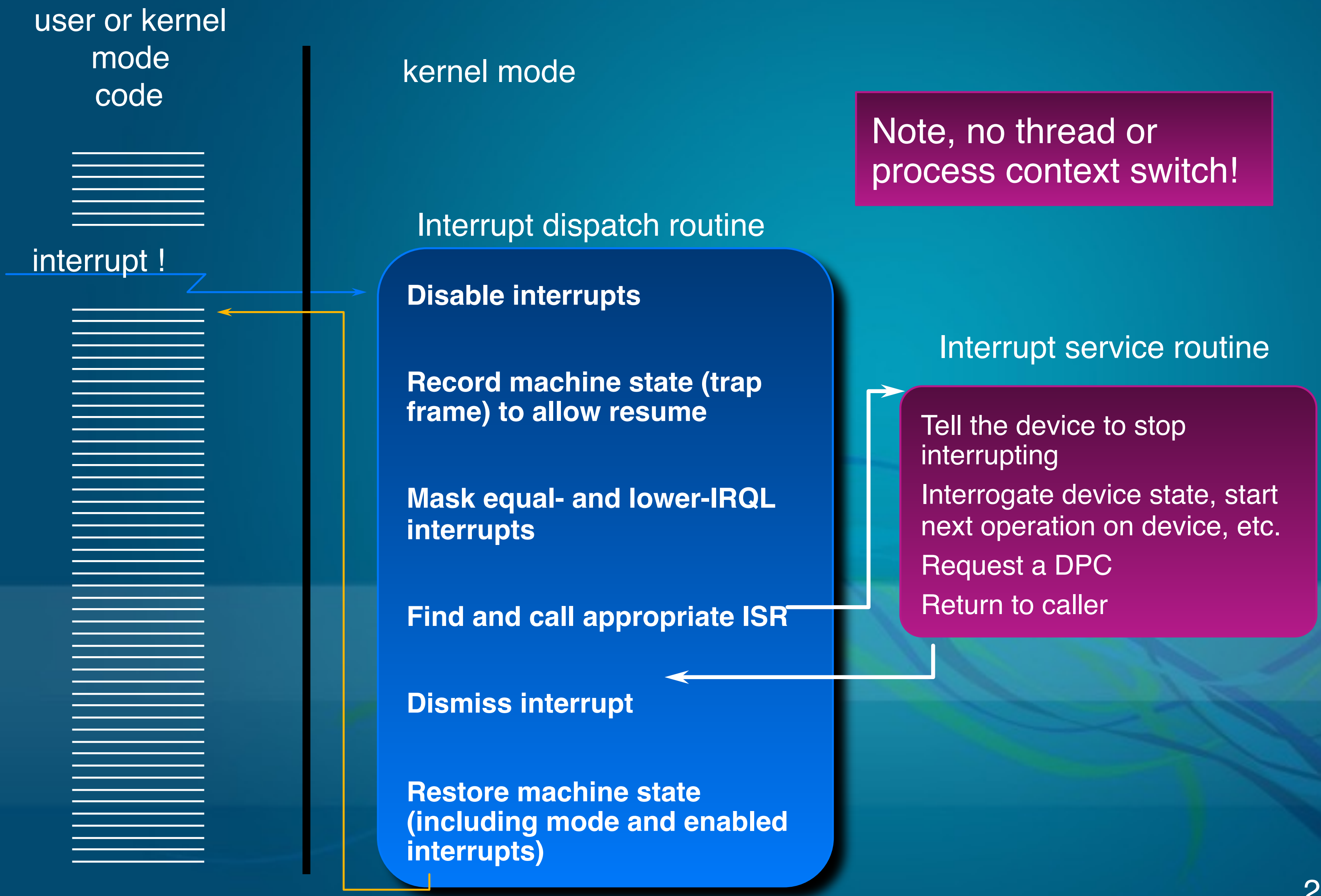
HAL – Hardware Abstraction Layer

- 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 and device drivers
  - Isolates OS & drivers from platform-specific details
  - Presents uniform model of I/O hardware interface to drivers
- Reduced role in Windows 2000
  - Bus support moved to bus drivers
  - Majority of HALs are vendor-independent

System Architecture

- Process Execution Environment
- Kernel Architecture
- Interrupt Handling
- Object Manager
- System Threads
- Process-based code
- Summary

Interrupt Dispatching



user or kernel mode code

interrupt !

kernel mode

Interrupt dispatch routine

Disable interrupts

Record machine state (trap frame) to allow resume

Mask equal- and lower-IRQL interrupts

Find and call appropriate ISR

Dismiss interrupt

Restore machine state (including mode and enabled interrupts)

Interrupt service routine

Tell the device to stop interrupting

Interrogate device state, start next operation on device, etc.

Request a DPC

Return to caller

Note, no thread or process context switch!

System Architecture

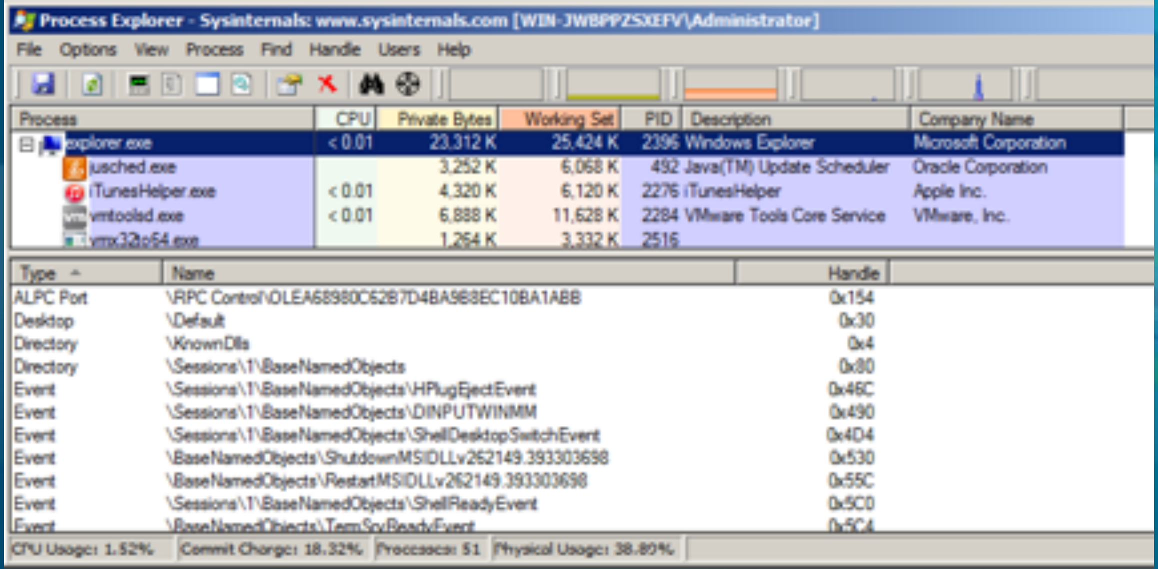
- Process Execution Environment
- Kernel Architecture
- Interrupt Handling
- Object Manager
- System Threads
- Process-based code
- Summary

Handles And Security

- Process handle table
  - Is unique for each process
  - But is in system address space, hence cannot be modified from user mode
  - Hence, is trusted
- Security checks are made when handle table entry is created
  - i.e. at CreateXxx time
  - Handle table entry indicates the “validated” access rights to the object
    - Read, Write, Delete, Terminate, etc.
  - No need to revalidate on each request

Examining Open Handles: Sysinternals Tools

- Process Explorer
- View, Lower Pane View, Handles
- Right-click column header, select column "Handle Value"


Process Explorer - Sysinternals: www.sysinternals.com [WIN-JWBPPZSXEFV\Administrator]
File
Options
View
Process
Find
Handle
Users
Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
explorer.exe	< 0.01	23,312 K	25,424 K	2396	Windows Explorer	Microsoft Corporation
jusched.exe		3,252 K	6,068 K	492	Java(TM) Update Scheduler	Oracle Corporation
iTunesHelper.exe	< 0.01	4,320 K	6,120 K	2276	iTunesHelper	Apple Inc.
vmtoolsd.exe	< 0.01	6,888 K	11,628 K	2284	VMware Tools Core Service	VMware, Inc.
vmx32to64.exe		1,264 K	3,332 K	2516		

Type	Name	Handle
ALPC Port	\RPC Control\OLEA68980C62B7D4BA9B3EC10BA1ABB	0x154
Desktop	\Default	0x30
Directory	\KnownDlls	0x4
Directory	\Sessions\1\BaseNamedObjects	0x80
Event	\Sessions\1\BaseNamedObjects\HPlugEjectEvent	0x46C
Event	\Sessions\1\BaseNamedObjects\DINPUTWINMM	0x490
Event	\Sessions\1\BaseNamedObjects\ShellDesktopSwitchEvent	0x4D4
Event	\BaseNamedObjects\ShutdownMSIDLLv262149.393303698	0x530
Event	\BaseNamedObjects\RestartMSIDLLv262149.393303698	0x55C
Event	\Sessions\1\BaseNamedObjects\ShellReadyEvent	0x5C0
Event	\BaseNamedObjects\TermSrvReadyEvent	0x5C4

CPU Usage: 1.52%   Commit Charge: 18.32%   Processes: 51   Physical Usage: 38.80%

Viewing Open Handles

- Handle View
  - By default, shows named objects
    - Click on Options->Show Unnamed Objects
- Uses:
  - Solve file locked errors
    - Can search to determine what process is holding a file or directory open
    - Can even close an open files (be careful!)
  - Understand resources used by an application
  - Detect handle leaks using refresh difference highlighting
  - View the state of synchronization objects (mutexes, semaphores, events)

System Architecture

- Process Execution Environment
- Kernel Architecture
- Interrupt Handling
- Object Manager
- System Threads
- Process-based code
- Summary

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
  - Not non-preemptible (unless they raise IRQL to 2 or above)
  - For details, see DDK documentation on PsCreateSystemThread
- What process do they appear in?
  - “System” process (Windows NT 4.0: PID 2,
    Windows 2000: PID 8, Windows XP: PID 4)
  - In Windows 2000 and later, windowing system threads (from Win32k.sys) appear in “csrss.exe”
    (Windows subsystem process)

Examples Of System Threads

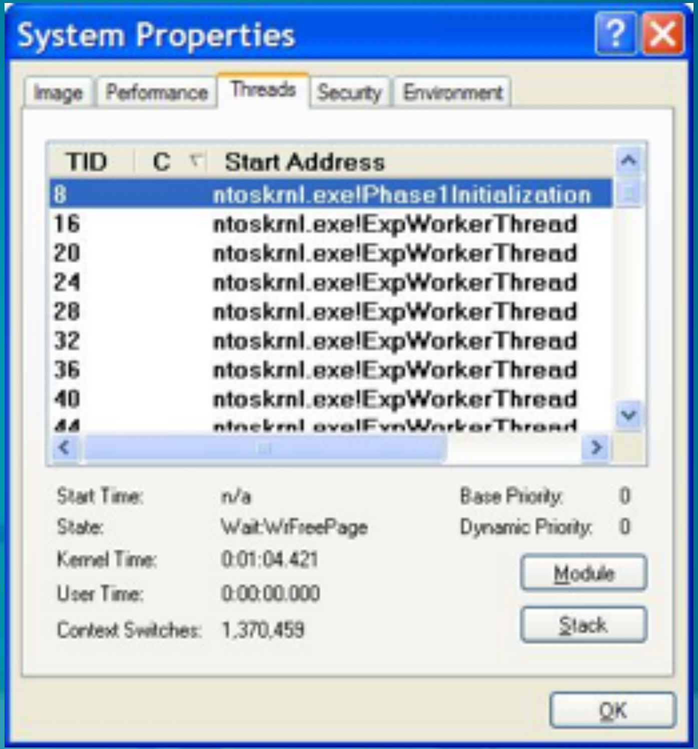
- Memory Manager
  - Modified Page Writer for mapped files
  - Modified Page Writer for paging files
  - Balance Set Manager
  - Swapper (kernel stack, working sets)
  - Zero page thread (thread 0, priority 0)
- Security Reference Monitor
  - Command Server Thread
- Network
  - Redirector and Server 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, …
  - Accessed via ExQueueWorkItem

Identifying System Threads

- If System threads are consuming CPU time, need to find out what code is running, since it could be any one of a variety of components
  - Pieces of OS (Ntoskrnl.exe)
  - File server worker threads (Srv.sys)
  - Other drivers
- To really understand what’s going on, must find which driver a thread “belongs to”

Identifiying System Threads

- Process Explorer:
  - Double click on System process
  - Go to Threads tab and sort by CPU
    - To view call stack, must use kernel debugger
- Note: several threads run between clock ticks (or at high IRQL) and thus don't appear to run
  - Watch context switch count



TID	C	Start Address
8		ntoskrnl.exe!Phase1Initialization
16		ntoskrnl.exe!ExpWorkerThread
20		ntoskrnl.exe!ExpWorkerThread
24		ntoskrnl.exe!ExpWorkerThread
28		ntoskrnl.exe!ExpWorkerThread
32		ntoskrnl.exe!ExpWorkerThread
36		ntoskrnl.exe!ExpWorkerThread
40		ntoskrnl.exe!ExpWorkerThread
44		ntoskrnl.exe!ExpWorkerThread

Start Time: n/a
State: Wait:WrFreePage
Kernel Time: 0:01:04.421
User Time: 0:00:00.000
Context Switches: 1,370,459
Base Priority: 0
Dynamic Priority: 0

System Architecture

- Process Execution Environment
- Kernel Architecture
- Interrupt Handling
- Object Manager
- System Threads
- Process-based code
- Summary

Process-Based Code

- OS components that run in separate executables (.exes), 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 Microsoft defined name
  - Windows Services
- Let's examine the system process "tree"
  - Use Tlist /T or Process Explorer

Process-Based NT Code

System Startup Processes

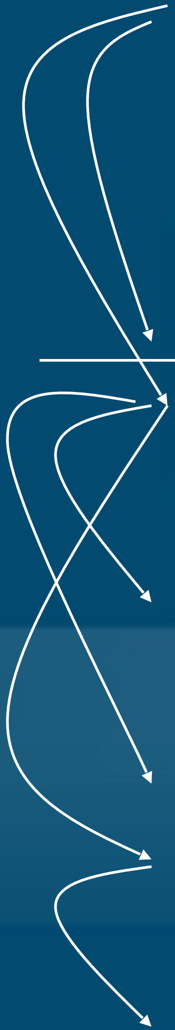
- First two processes aren't real processes
  - Not running a user mode .EXE
  - No user-mode address space
  - Different utilities report them with different names
  - Data structures for these processes (and their initial threads) are "pre-created" in NtosKrnl.Exe and loaded along with the code

(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 Phase1Initialization) launches the first "real" process, running smss.exe...
...and then becomes the zero page thread

Process-Based NT Code

System Startup Processes



smss.exe Session Manager

The first "created" process
Takes parameters from \HKEY_LOCAL_MACHINE\System
\CurrentControlSet
\Control\Session Manager
Launches required subsystems (csrss) and then winlogon

csrss.exe Windows subsystem

winlogon.exe Logon process: Launches services.exe & lsass.exe; presents first login prompt
When someone logs in, launches apps in \Software\Microsoft
\Windows NT\WinLogon\Userinit

services.exe Service Controller; also, home for many NT-supplied services
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

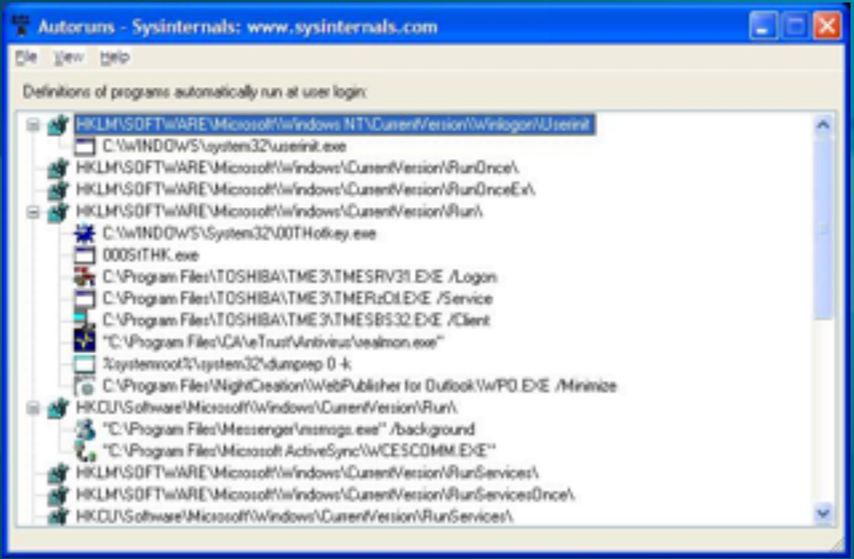
Logon Process

- Winlogon sends username/password to Lsass
  - Either on local system for local logon, or to Netlogon service on a domain
  - Windows XP enhancement: Winlogon doesn’t wait for Workstation service to start if
    - Account doesn't depend on a roaming profile
    - Domain policy that affects logon hasn't changed since last logon
    - Controller for a network logon
- Creates a process to run
  HKLM\Software\Microsoft\Windows NT
  \CurrentVersion\WinLogon\Userinit
  - By default: Userinit.exe
  - Runs logon script, restores drive-letter mappings, starts shell
- Userinit creates a process to run
  HKLM\Software\Microsoft\Windows NT
  \CurrentVersion\WinLogon\Shell
  - By default: Explorer.exe
- There are other places in the Registry that control programs that start at logon

Processes Started at Logon

- Displays order of processes configured to start at log on time
- Also can use new XP built-in tool called "System Configuration Utility"
  - To run, click on Start->Help, then "Use Tools…", then System Configuration Utility
  - Only shows what's defined to start vs Autoruns which shows all places things CAN be defined to start

Autoruns (Sysinternals)

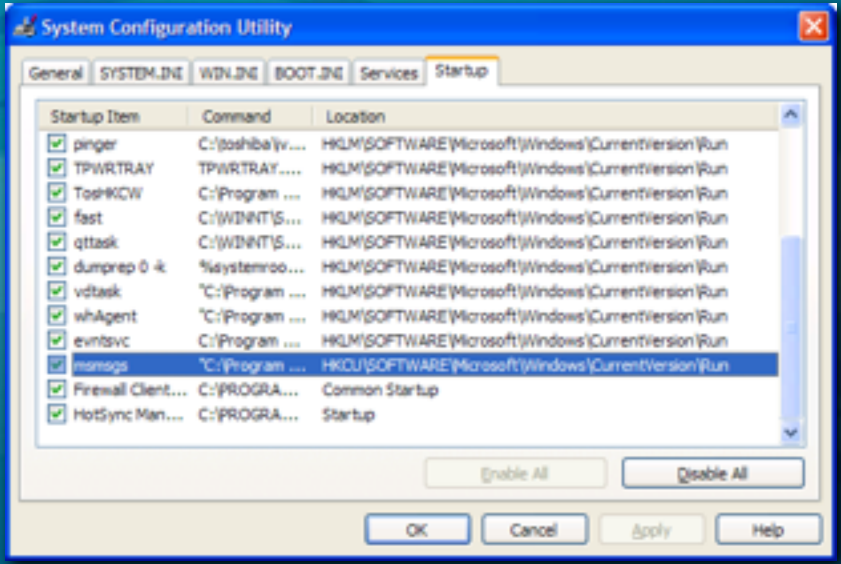


Definitions of programs automatically run at user login:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
    C:\WINDOWS\system32\userinit.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce\
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
    C:\WINDOWS\System32\00THotkey.exe
    000StTHK.exe
    C:\Program Files\TOSHIBA\TME3\TMESRV31.EXE /Logon
    C:\Program Files\TOSHIBA\TME3\TMERzCtl.EXE /Service
    C:\Program Files\TOSHIBA\TME3\TMESBS32.EXE /Client
    "C:\Program Files\CA\eTrust\Antivirus\realmon.exe"
    %systemroot%\system32\dumprep 0 -k
    C:\Program Files\NightCreation\WebPublisher for Outlook\WPO.EXE /Minimize
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
    "C:\Program Files\Messenger\msmsgs.exe" /background
    "C:\Program Files\Microsoft ActiveSync\WCESCOMM.EXE"
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices\
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce\
HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices\
```

Msconfig

(in \Windows\pchealth\helpctr\binaries)



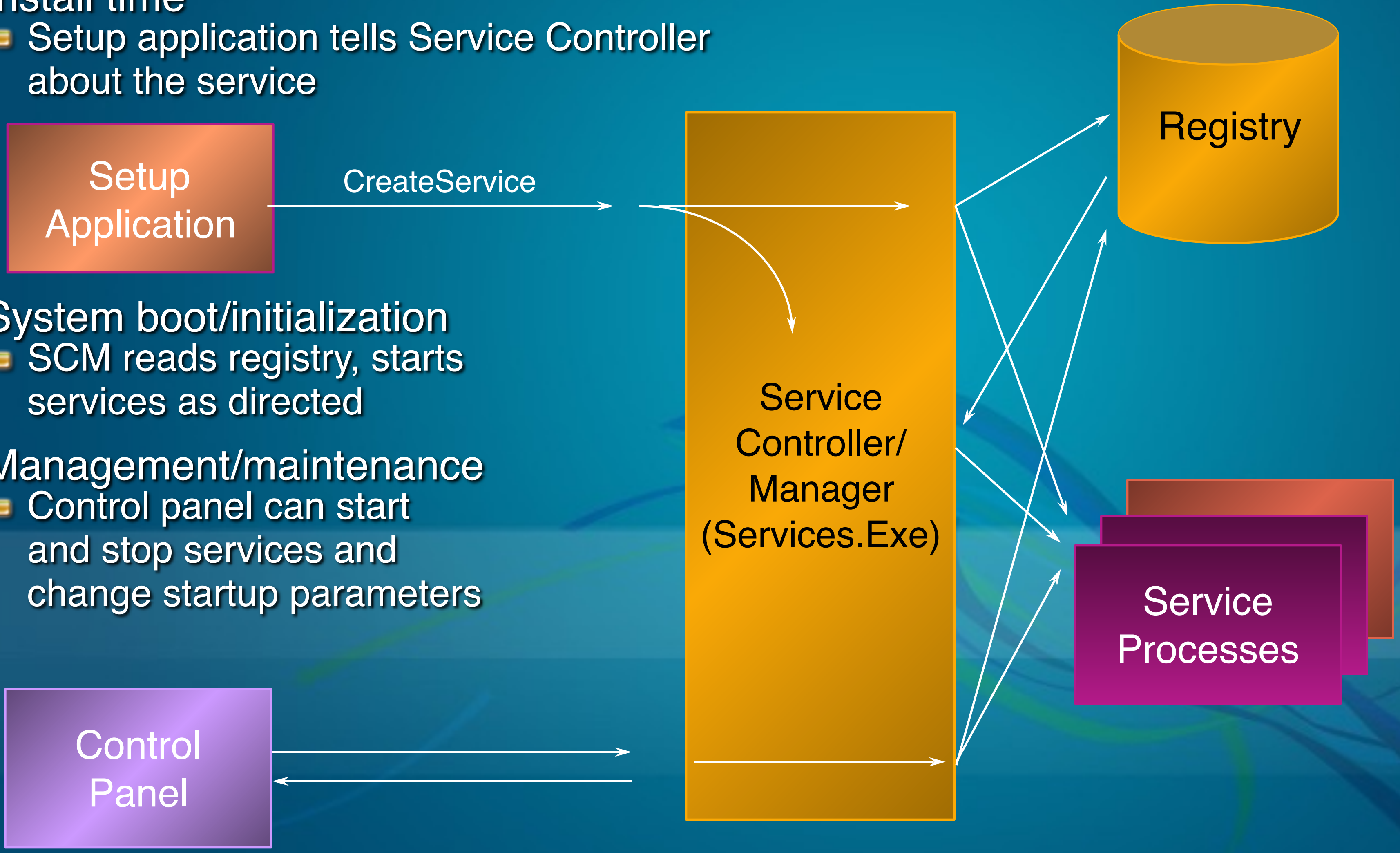
Startup Item	Command	Location
☑ pinger	C:\toshiba\iv....	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ TPWRTRAY	TPWRTRAY.....	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ TosHKCW	C:\Program	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ fast	C:\WINNT\S....	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ qttask	C:\WINNT\S....	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ dumprep 0 -k	%systemroo....	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ vdtask	"C:\Program	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ whAgent	"C:\Program	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ evntsvc	C:\Program	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ msmsgs	"C:\Program	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
☑ Firewall Client....	C:\PROGRA....	Common Startup
☑ HotSync Man....	C:\PROGRA....	Startup

Windows Services

- An overloaded generic term
- A process created and managed by the Service Control Manager (Services.exe)
  - E.g. Solitaire can be configured as a service, but is killed shortly after starting
- 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 is one way to start a process on a remote machine (now you can do it with WMI)

Life Of A Service

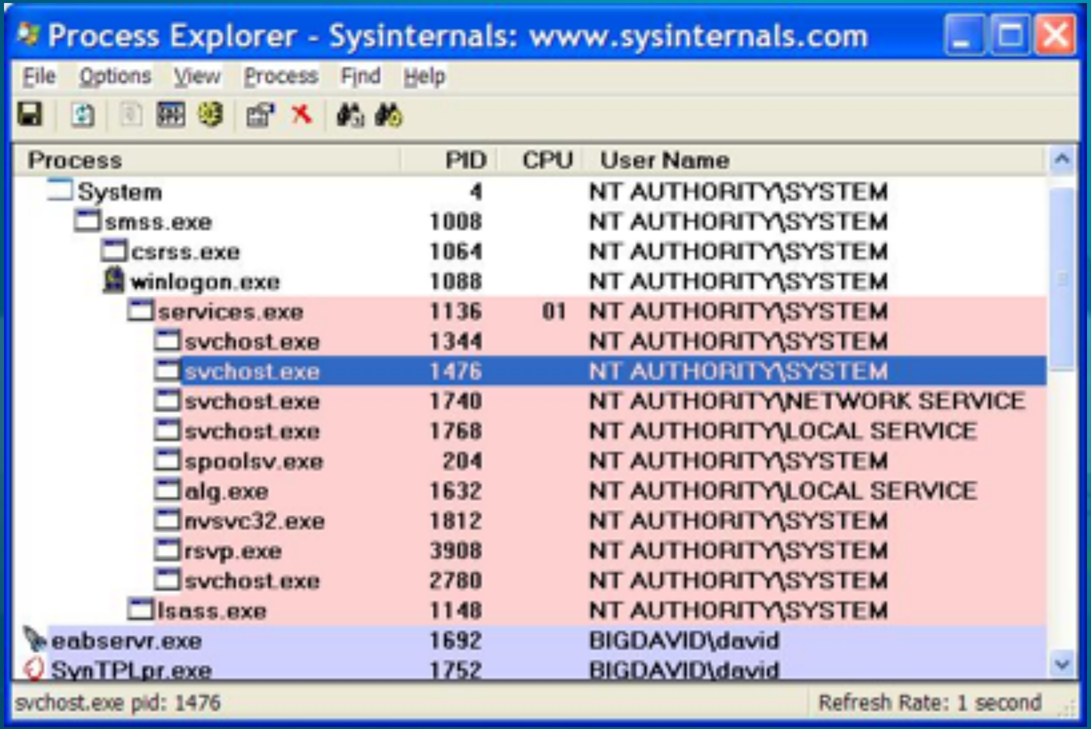
- Install time
  - Setup application tells Service Controller about the service



- System boot/initialization
  - SCM reads registry, starts services as directed
- Management/maintenance
  - Control panel can start and stop services and change startup parameters

Viewing Service Processes

- Process Explorer colors Services pink by default


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

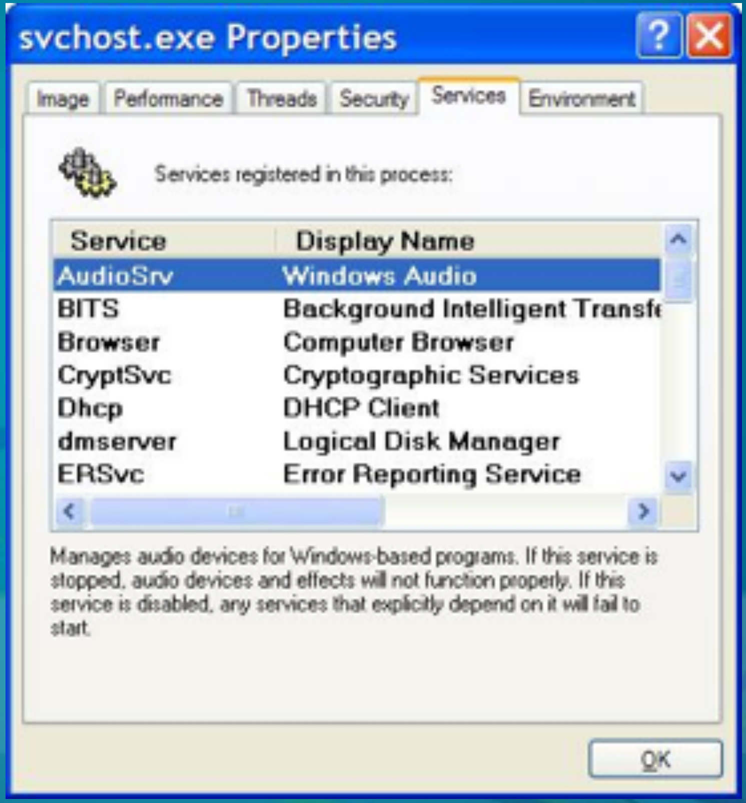
Refresh Rate: 1 second

Svchost Mechanism

- 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

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.

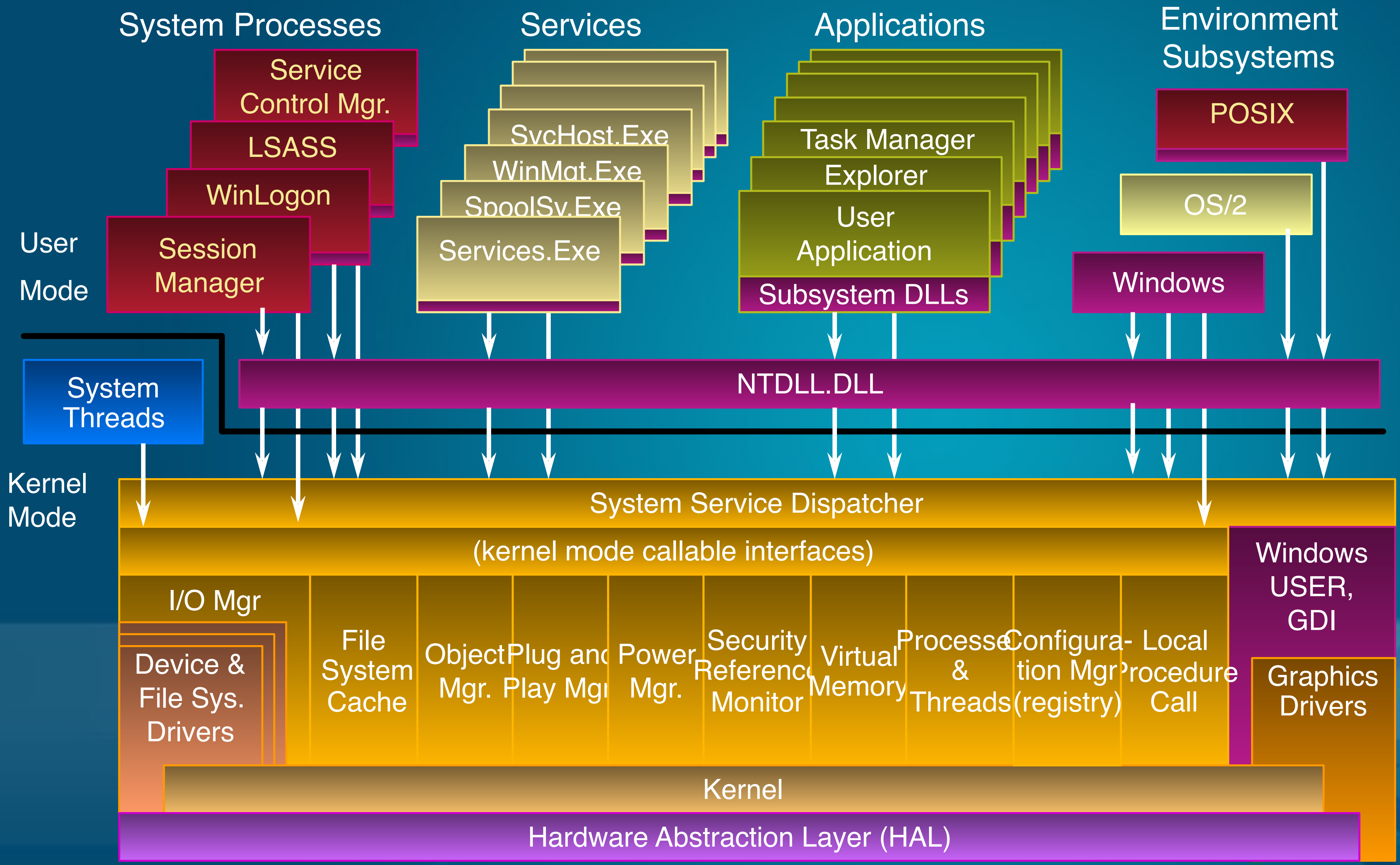
System Architecture

- Process Execution Environment
- Kernel Architecture
- Interrupt Handling
- Object Manager
- System Threads
- Process-based code
- Summary

Four Contexts For Executing Code

- Full process and thread context
  - User applications
  - Windows Services
  - Environment subsystem processes
  - System startup processes
- Have thread context but no “real” process
  - Threads in “System” process
- Routines called by other threads/processes
  - Subsystem DLLs
  - Executive system services (NtReadFile, etc.)
  - GDI32 and User32 APIs implemented in Win32K.Sys (and graphics drivers)
- No process or thread context (“arbitrary thread context”)
  - Interrupt dispatching
  - Device drivers

System Architecture



System Processes

Service Control Mgr.

LSASS

WinLogon

Session Manager

Services

SvcHost.Exe

WinMgt.Exe

SpoolSv.Exe

Services.Exe

Applications

Task Manager

Explorer

User Application

Subsystem DLLs

Environment Subsystems

POSIX

OS/2

Windows

User Mode

System Threads

NTDLL.DLL

Kernel Mode

System Service Dispatcher

(kernel mode callable interfaces)

I/O Mgr

Device & File Sys. Drivers

File System Cache

Object Mgr.

Plug and Play Mgr

Power Mgr.

Security Reference Monitor

Virtual Memory

Processes & Threads

Configuration Mgr (registry)

Local Procedure Call

Windows USER, GDI

Graphics Drivers

Kernel

Hardware Abstraction Layer (HAL)

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

Outline

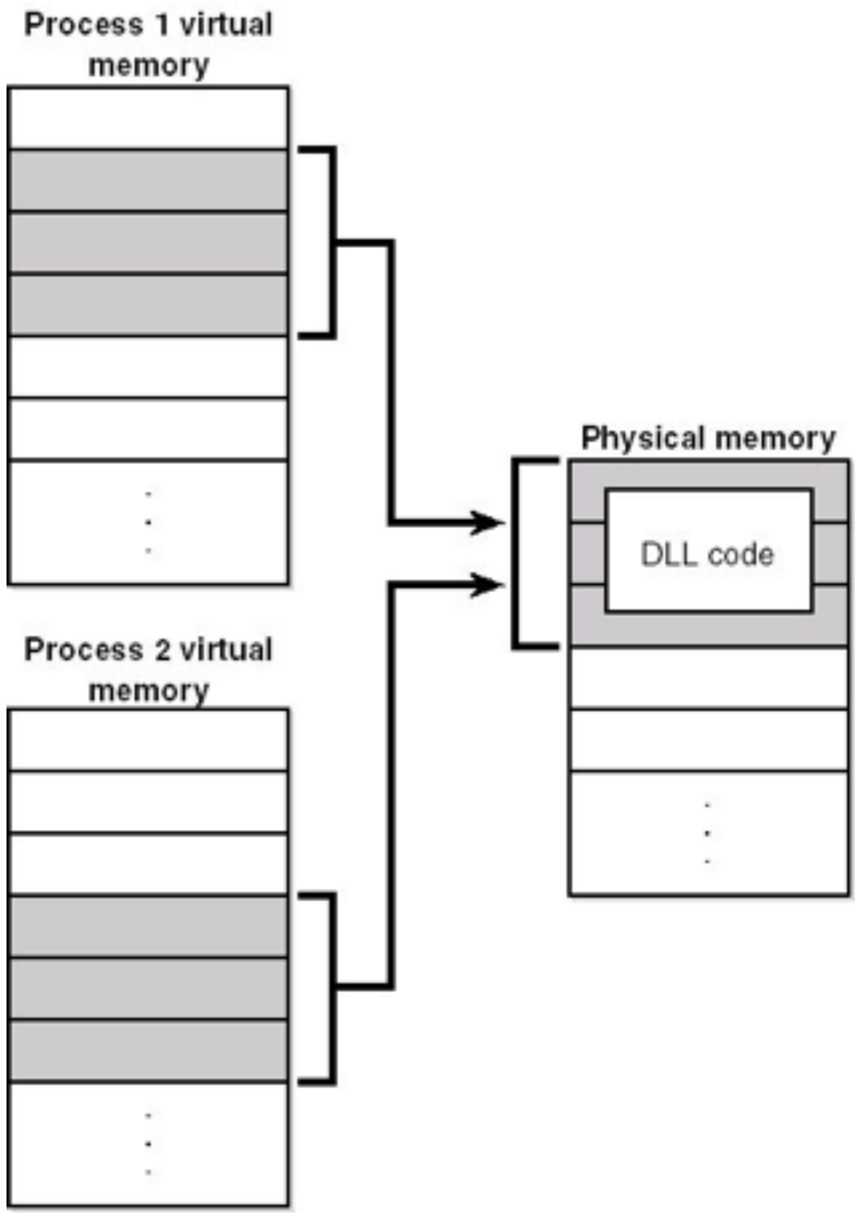
1. System Architecture
2. Processes and Thread Internals
3. Memory Management Internals
4. Security Internals

Memory Management

- Core Memory Management Services
- Working Set Management
- Unassigned Memory
- Page Files

Shared Memory

- Like most modern OSs, Windows provides a way for processes to share memory
  - High speed IPC (used by LPC, which is used by RPC)
  - Threads share address space, but applications may be divided into multiple processes for stability reasons
- Processes can also create shared memory sections
  - Called page file backed file mapping objects
  - Full Windows security
- It does this automatically for shareable pages
  - E.g., code pages in an .EXE



Memory Management

- Core Memory Management Services
- Working Set Management
- Unassigned Memory
- Page Files

Prefetch Mechanism

- File activity is traced and used to prefetch data the next time
  - First 10 seconds are monitored
    - Pages referenced & directories opened
  - Prefetch “trace file” stored in \Window\Prefetch
    - Name of .EXE-<hash of full path>.pf
- Also applies to system boot
  - First 2 minutes of boot process logged
    - Stops 30 seconds after the user starts the shell or 60 seconds after all services
      are started
  - Boot trace file: NTOSBOOT-B00DFAAD.pf

Prefetch Mechanism

- When application run again, system automatically
  - Reads in directories referenced
  - Reads in code and file data
    - Reads are asynchronous
    - But waits for all prefetch to complete
- In addition, every 3 days, system automatically defrags files involved in each application startup
- Bottom line: Reduces disk head seeks
  - This was seen to be the major factor in slow application/system startup

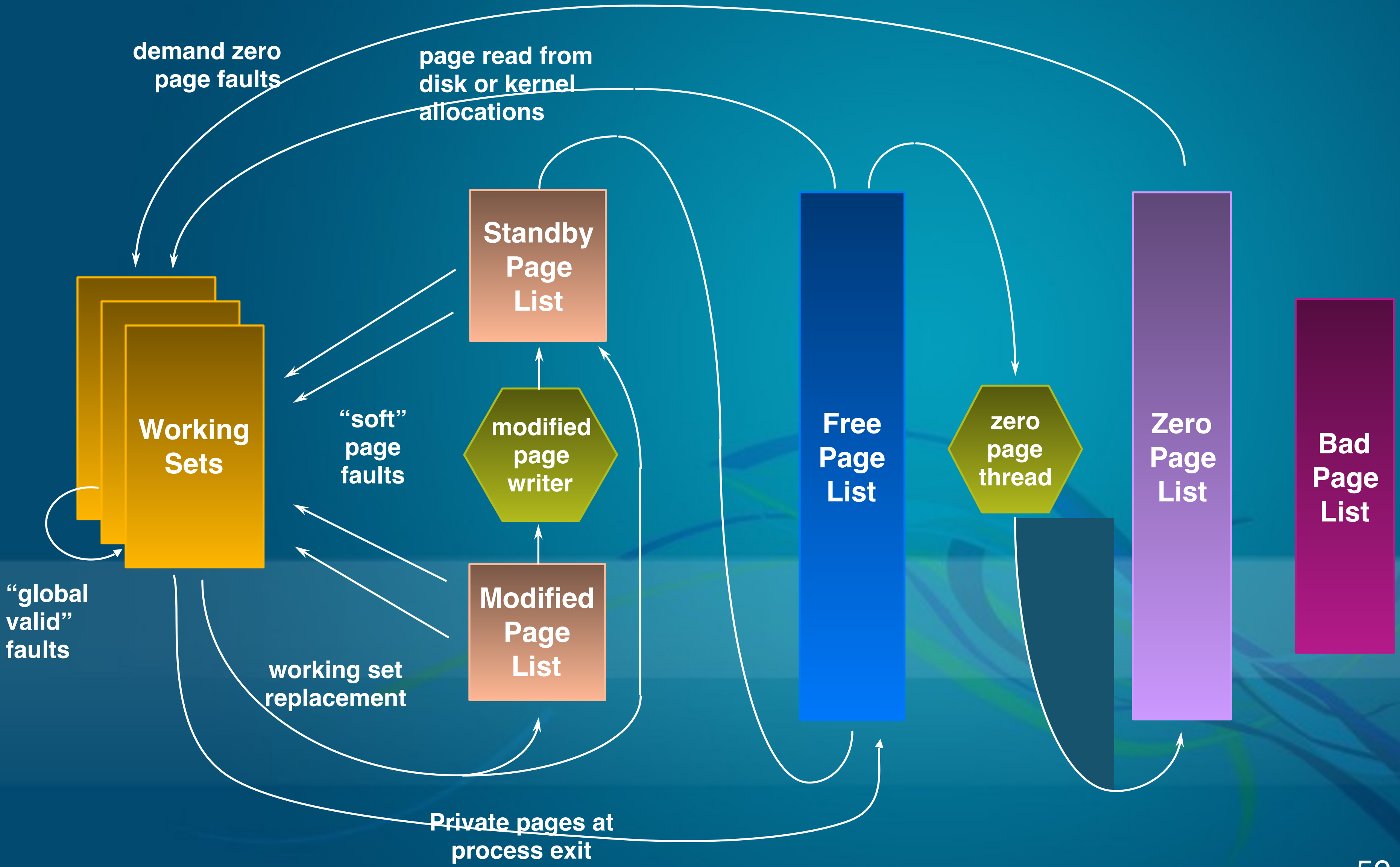
Memory Management

- Core Memory Management Services
- Working Set Management
- Unassigned Memory
- Page Files

Managing Physical Memory

- System keeps unassigned physical pages on one of several lists
  - Free page list
  - Modified page list
  - Standby page list
  - Zero page list
  - Bad page list – pages that failed memory test at system startup
- Lists are implemented by entries in the “PFN database”
  - Maintained as FIFO lists or queues

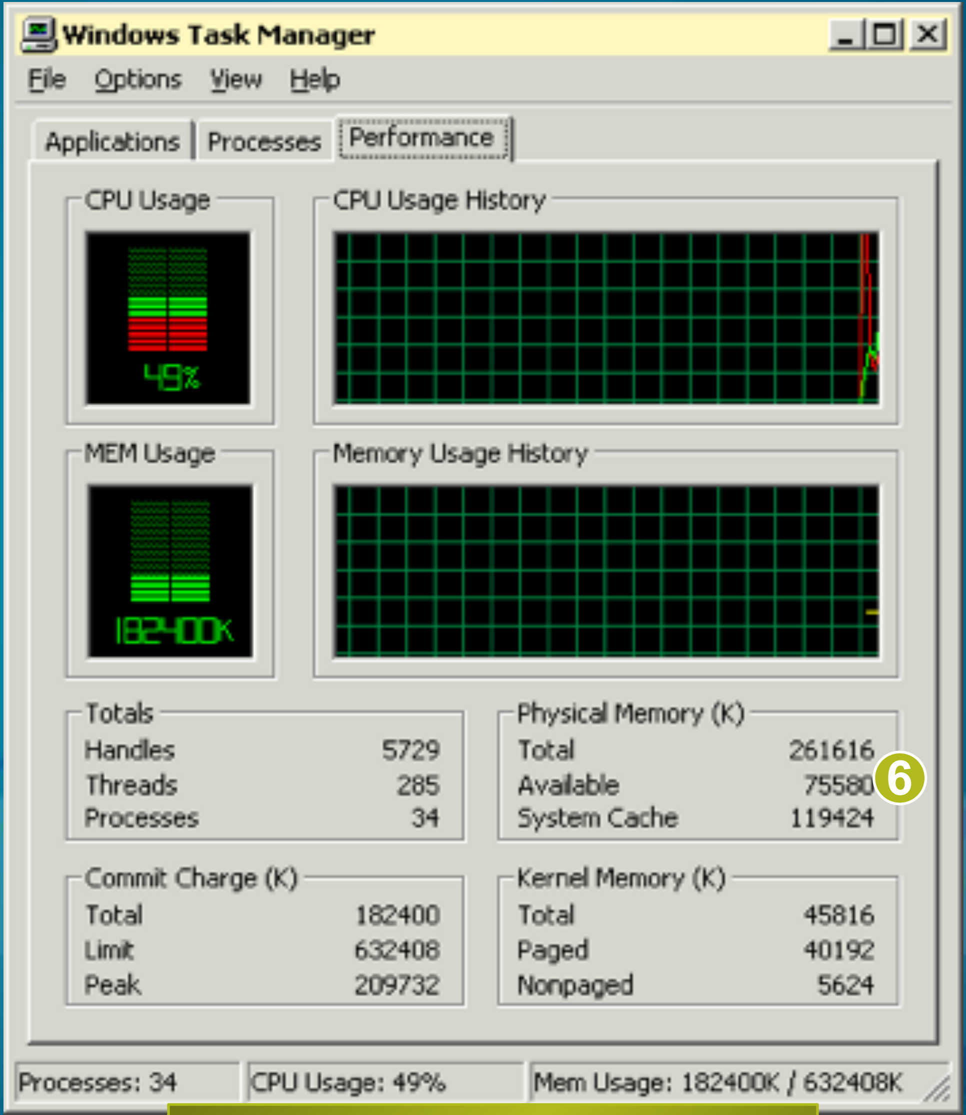
Paging Dynamics



Memory Management Information

Task Manager
Performance tab

- 6 “Available” = sum of free, standby, and zero page lists (physical)
- Majority are likely standby pages
- “System Cache” = size of standby list + size of system working set (file cache, paged pool, pageable OS/driver code & data)



49%

182400K

Totals	
Handles	5729
Threads	285
Processes	34

Physical Memory (K)	
Total	261616
Available	75580
System Cache	119424

6

Commit Charge (K)	
Total	182400
Limit	632408
Peak	209732

Kernel Memory (K)	
Total	45816
Paged	40192
Nonpaged	5624

Processes: 34 | CPU Usage: 49% | Mem Usage: 182400K / 632408K

Screen snapshot from:
Task Manager | Performance tab

Outline

1. System Architecture
2. Processes and Thread Internals
3. Memory Management Internals
4. Security Internals

Security

- Introduction
- Components
- Logon
- Protecting Objects
- Privileges

Windows Security Support

- Microsoft’s goal was to achieve C2, which requires:
  - Secure Logon: NT provides this by requiring user name and password
  - Discretionary Access Control: fine grained protection over resources by user/group
  - Security Auditing: ability to save a trail of important security events, such as access or attempted access of a resource
  - Object reuse protection: must initialize physical resources that are reused e.g. memory, files
- Certifications achieved:
  - Windows NT 3.5 (workstation and server) with SP3 earned C2 in July 1995
  - In March 1999 Windows NT 4 with SP3 earned e3 rating from UK’s Information Technology Security (ITSEC) – equivalent to C2
  - In November 1999 NT4 with SP6a earned C2 in stand-alone and networked environments

Windows Security Support

- Windows meets two B-level requirements:
  - Trusted Path Functionality: way to prevent trojan horses with “secure attention sequence” (SAS) - Ctrl-Alt-Del
  - Trusted Facility Management: ability to assign different roles to different accounts
    - Windows does this through account privileges (TBD later)

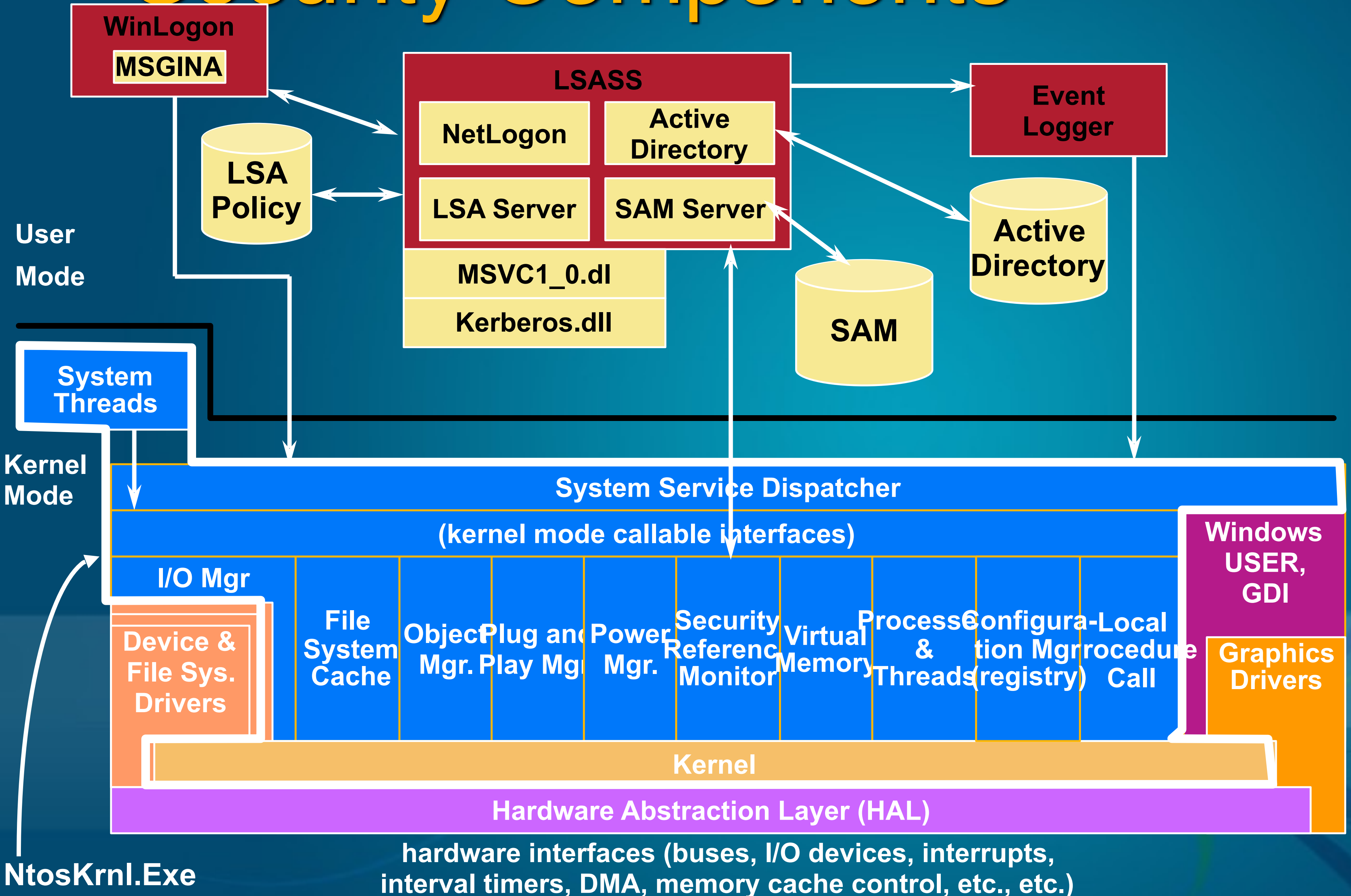
Common Criteria

- New standard, called Common Criteria (CC), is the new standard for software and OS ratings
  - Consortium of US, UK, Germany, France, Canada, and the Netherlands in 1996
  - Became ISO standard 15408 in 1999
  - For more information, see http://www.commoncriteriaportal.org/ and *http://csrc.nist.gov/cc*
- CC is more flexible than TCSEC trust ratings
  - Protection Profile collects security requirements
  - Security Target (ST) are security requirements that can be made by reference to a PP
- Windows 2000 was certified as compliant with the CC Controlled Access Protection Profile (CAPP) in October 2002
  - Windows XP and Server 2003 are undergoing evaluation

Security

- Introduction
- Components
- Logon
- Protecting Objects
- Privileges

Security Components



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

Security Reference Monitor

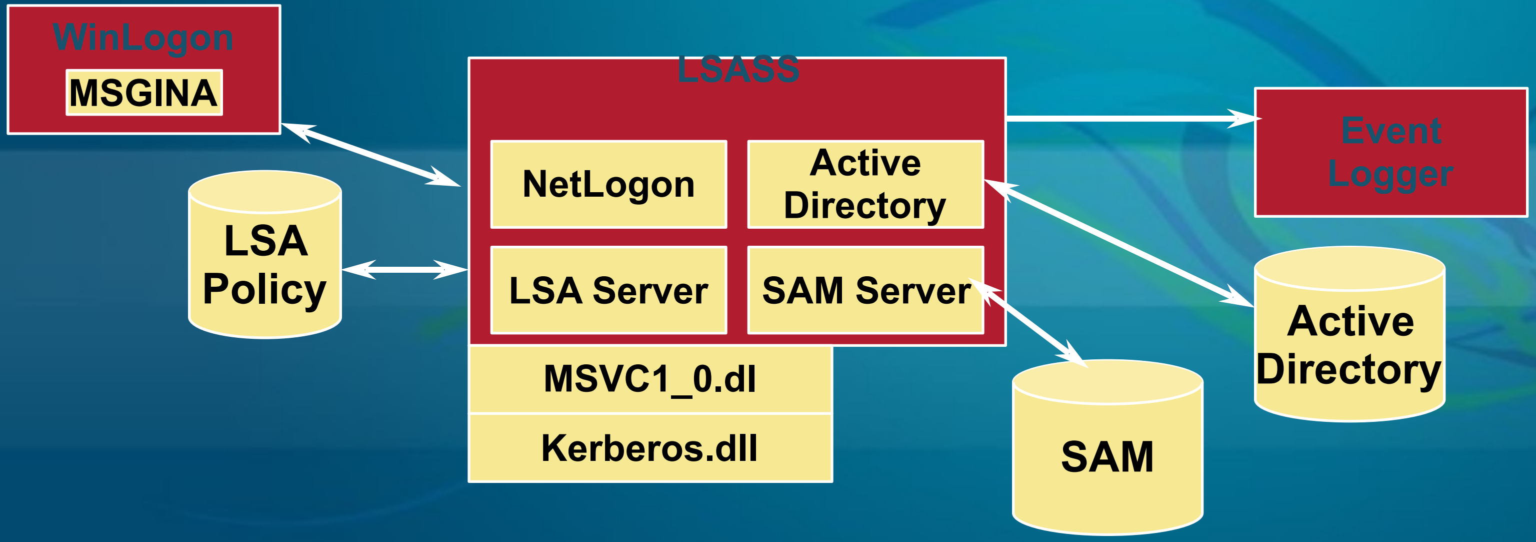
- Performs object access checks, manipulates privileges, and generates audit messages
- Group of functions in Ntoskrnl.exe
  - Some documented in DDK
  - Exposed to user mode by Windows API calls
- Demo: Open Ntoskrnl.exe with Dependency Walker and view functions starting with “Se”

Demo: Viewing Security Processes

- Run Process Explorer
- Collapse Explorer process tree and focus on upper half (system processes)

Security Components

- Local Security Authority
  - User-mode process (\Windows\System32\Lsass.exe) that implements policies (e.g. password, logon), authentication, and sending audit records to the security event log
  - LSASS policy database: registry key HKLM\SECURITY



WinLogon

MSGINA

LSA Policy

LSASS

NetLogon

Active Directory

LSA Server

SAM Server

MSVC1_0.dl

Kerberos.dll

Event Logger

Active Directory

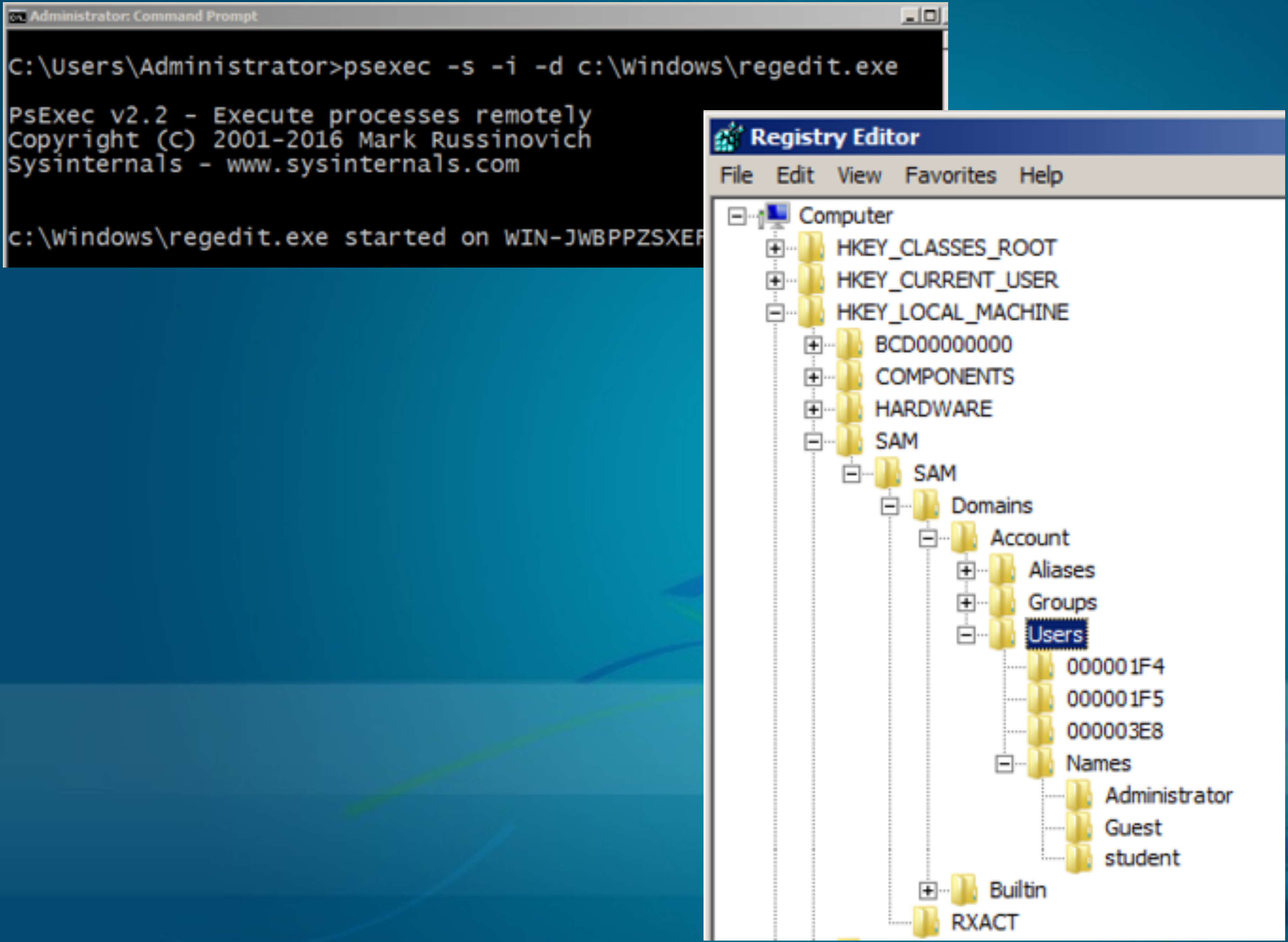
SAM

Demo: Looking at the SAM

- Look at HKLM\SAM permissions
  - SAM security allows only the local system account to access it
  - Run Regedit
  - Look at HKLM\SAM - nothing there?
  - Check permissions (right click->Permissions)
  - Close Regedit
- Look in HKLM\SAM
  - Running Regedit in the local system account allows you to view the SAM:

```
psexec –s –i –d c:\windows\regedit.exe
or
sc create cmdassystem type= own type= interact
          binpath= "cmd /c start cmd /k“
sc start cmdassystem
```

  - View local usernames under HKLM\SAM\SAM\Domains\Account\Users\Names
  - Passwords are under Users key above Names



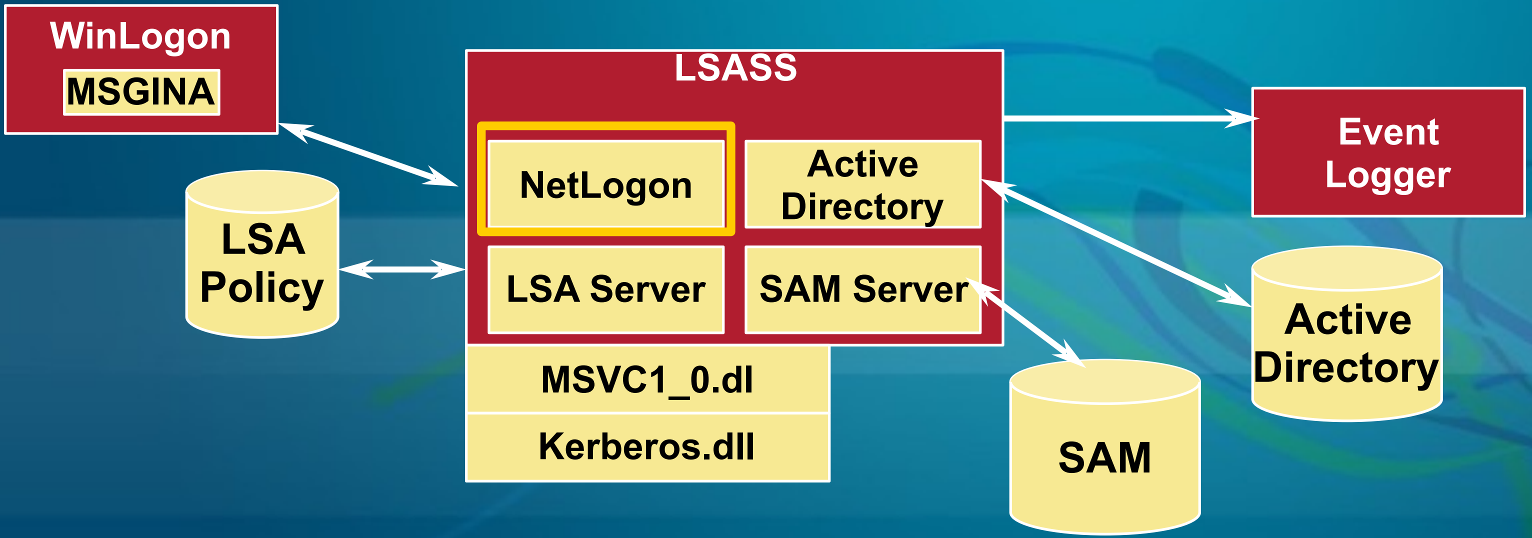
```
C:\Users\Administrator>psexec -s -i -d c:\Windows\regedit.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


c:\Windows\regedit.exe started on WIN-JWBPPZSXEF
```

LSASS Components

- Active Directory
  - A directory service that contains a database that stores information about objects in a domain
  - A *domain* is a collection of computers and their associated security groups that are managed as a single entity
  - The Active Directory server, implemented as a service, \Windows \System32\Ntdsa.dll, that runs in the Lsass process
- Authentication packages
  - DLLs that run in the context of the Lsass process and that implement Windows authentication policy:
    - LanMan: \Windows\System32\Msv1_0.dll
    - Kerberos: \Windows\System32\Kerberos.dll
    - Negotiate: uses LanMan or Kerberos, depending on which is most appropriate

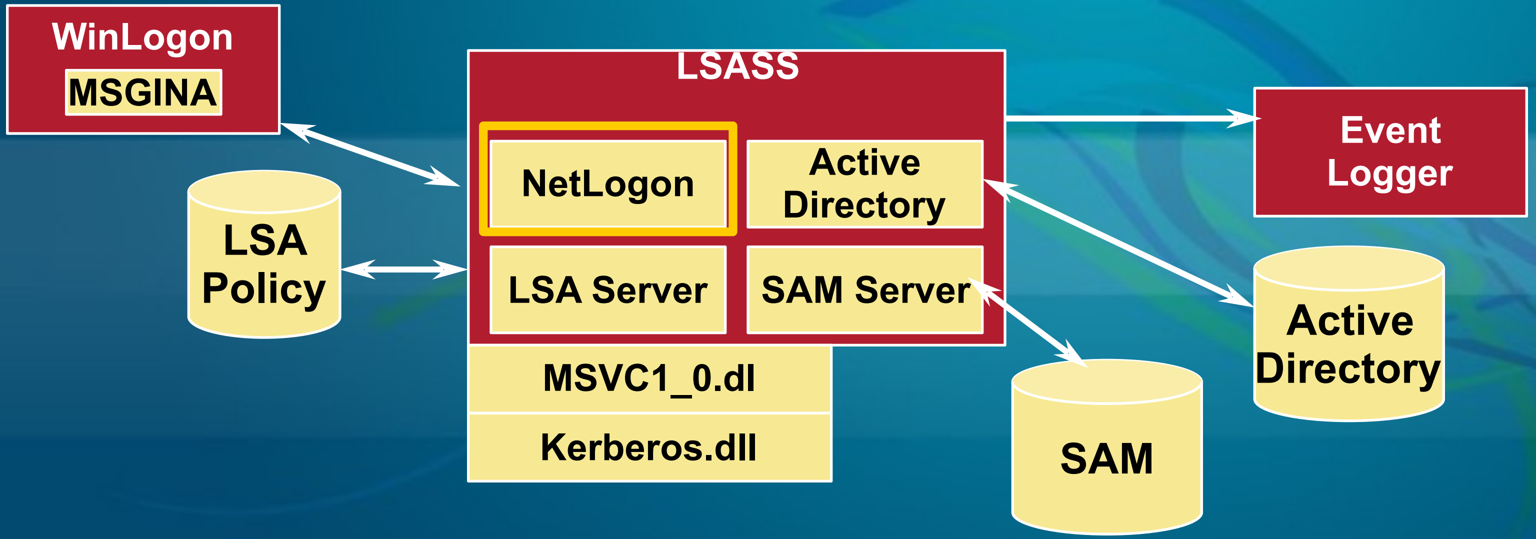
LSASS Components

- Net Logon service (Netlogon)
  - A Windows service (\Windows\System32\Netlogon.dll) that runs inside Lsass and responds to Microsoft LAN Manager 2 Windows NT (pre-Windows 2000) network logon requests
  - Authentication is handled as local logons are, by sending them to Lsass for verification
  - Netlogon also has a locator service built into it for locating domain controllers



Winlogon

- Logon process (Winlogon)
  - A user-mode process running \Windows\System32\Winlogon.exe that is responsible for responding to the SAS and for managing interactive logon sessions
- Graphical Identification and Authentication (GINA)
  - A user-mode DLL that runs in the Winlogon process and that Winlogon uses to obtain a user's name and password or smart card PIN
    - Default is \Windows\System32\Msgina.dll



Security

- Introduction
- Components
- Logon
- Protecting Objects
- Privileges

What Makes Logon Secure?

- Before anyone logs on, the visible desktop is Winlogon’s
- Winlogon registers CTRL+ALT+DEL, the Secure Attention Sequence (SAS), as a standard hotkey sequence
- SAS takes you to the Winlogon desktop
- No application can deregister it because only the thread that registers a hotkey can deregister it
- When Windows’ keyboard input processing code sees SAS it disables keyboard hooks so that no one can intercept it

Logon

- After getting security identification (account name, password), the GINA sends it to the Local Security Authority Sub System (LSASS)
- LSASS calls an authentication package to verify the logon
  - If the logon is local or to a legacy domain, MSV1_0 is the authenticator. User name and password are encrypted and compared against the Security Accounts Manager (SAM) database
    - Cached domain logons are also handled by MSV1_0
  - If the logon is to a AD domain the authenticator is Kerberos, which communicates with the AD service on a domain controller
- If there is a match, the SIDs of the corresponding user account and its groups are retrieved
- Finally, LSASS retrieves account privileges from the Security database or from AD

Logon

- LSASS creates a token for your logon session and Winlogon attaches it to the first process of your session
  - Tokens are created with the NtCreateToken API
  - Every process gets a copy of its parent's token
- SIDs and privileges cannot be added to a token
- A logon session is active as long as there is at least one token associated with the session
- Lab
  - Run "LogonSessions –p" (from Sysinternals) to view the active logon sessions on your system

Security

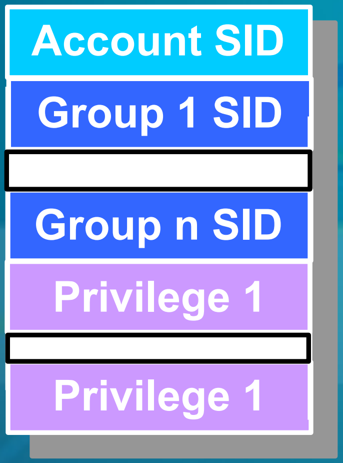
- Introduction
- Components
- Logon
- Protecting Objects
- Privileges

The Access Validation Algorithm

- Access validation is a security equation that takes three inputs:
  - Desired Access
  - Process Token
    - Or Thread’s token if the thread is “impersonating”
  - The object’s Security Descriptor, which contains a Discretionary Access Control List (DACL)
- The output is access allowed or access denied

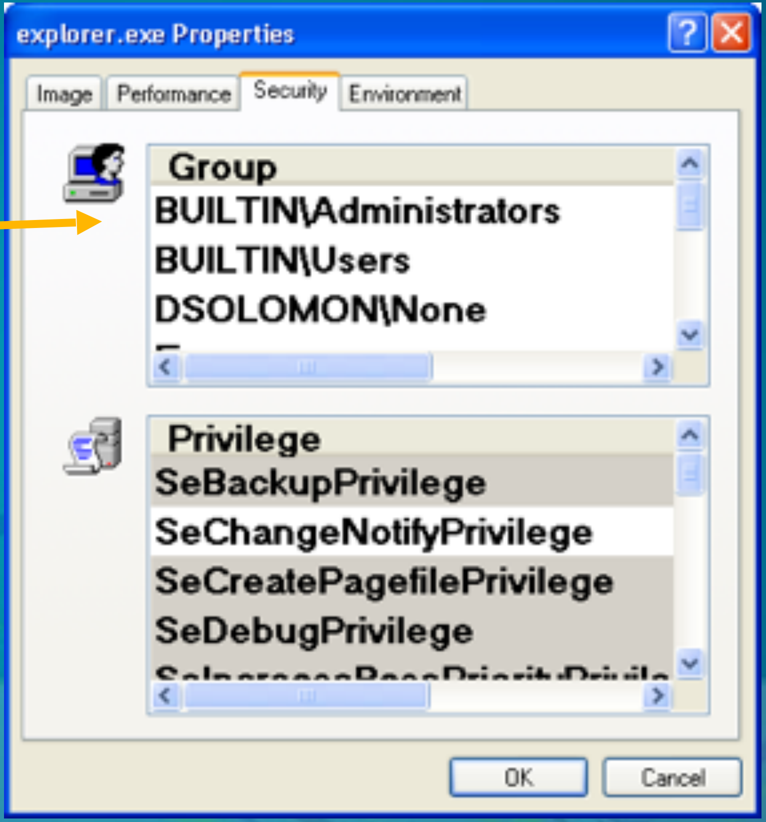
Tokens

- The main components of a token are:
  - SID of the user
  - SIDs of groups the user account belongs to
  - Privileges assigned to the user (described in next section)



Labs: Viewing Access Tokens

- Process Explorer: double click on a process and go to Security tab
  - Examine groups list
- Use RUNAS to create a CMD process running under another account (e.g. your domain account)
  - Examine groups list
- Viewing tokens with the Kernel Debugger
  - Run !process 0 0 to find a process
  - Run !process <PID> 1 to dump the process
  - Get the token address and type !token –n <token address>
  - Type dt _token <token address> to see all fields defined in a token



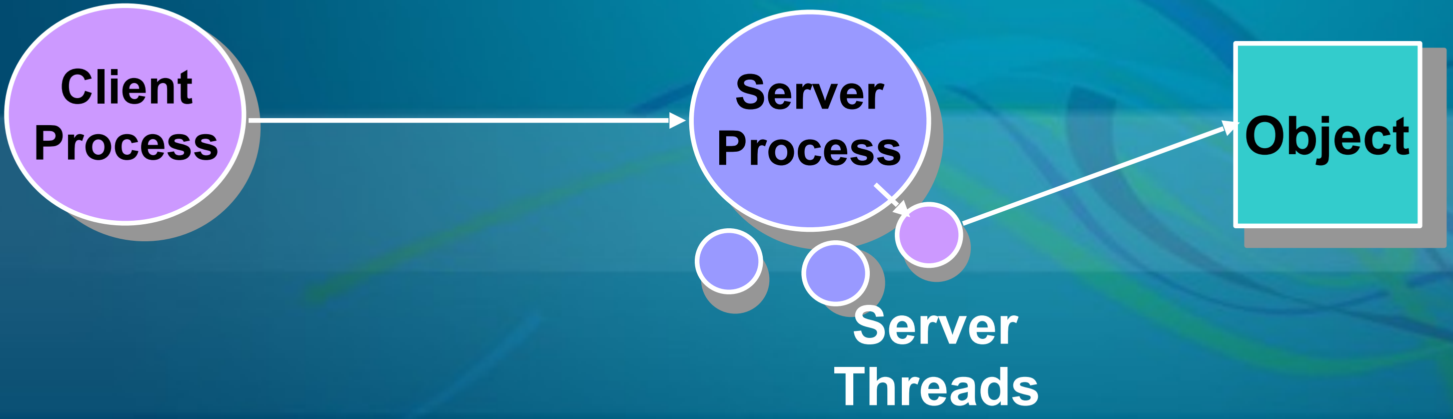
explorer.exe Properties

Group
BUILTIN\Administrators
BUILTIN\Users
DSOLOMON\None

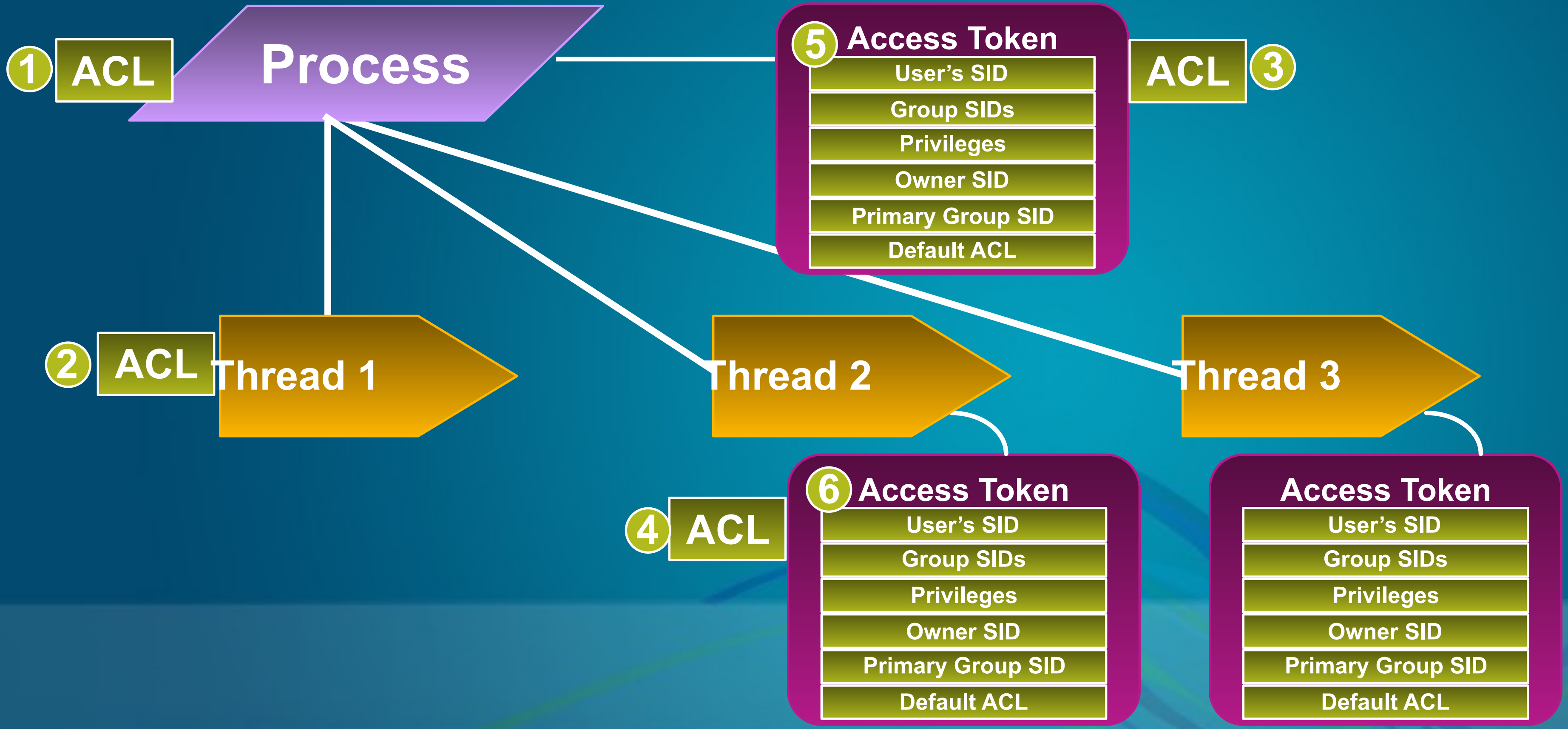
Privilege
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreatePagefilePrivilege
SeDebugPrivilege
SeIncreaseBasePriorityPrivile

Impersonation

- Lets an application adopt the security profile another user
  - Used by server applications
  - Impersonation is implemented at the thread level
    - The process token is the “primary token” and is always accessible
    - Each thread can be impersonating a different client
- Can impersonate with a number of client/server networking APIs – named pipes, RPC, DCOM



Process And Thread Security Structures



- Thread tokens (where present) completely supersede process token (basis for "security impersonation")

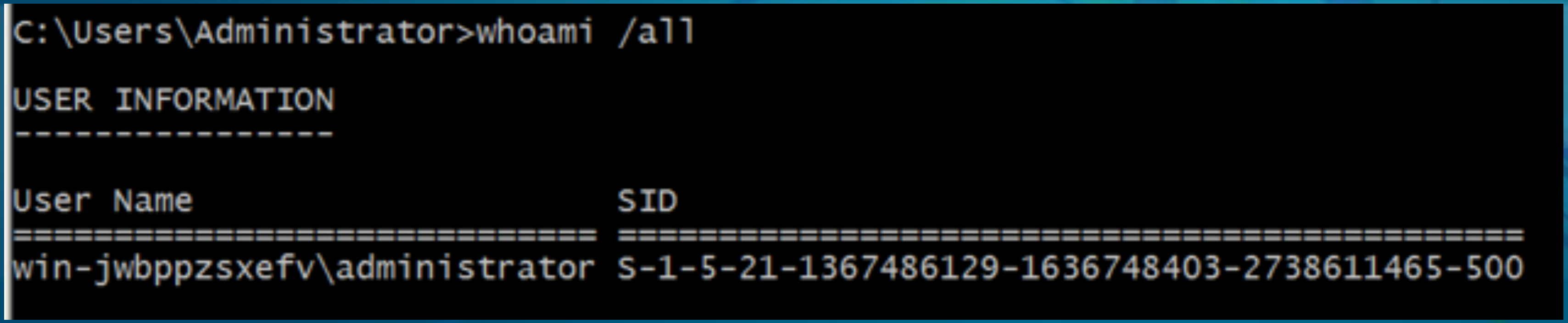
SIDs

- Windows uses Security Identifers (SIDs) to identify security principles:
  - Users, Groups of users, Computers, Domains
- SIDs consist of:
  - A revision level e.g. 1
  - An identifier-authority value e.g. 5 (SECURITY_NT_AUTHORITY)
  - One or more subauthority values
- Who assigns SIDs?
  - Setup assigns a computer a SID
  - Dcpromo assigns a domain a SID
  - Users and groups on the local machine are assigned SIDs that are rooted with the computer SID, with a Relative Identifier (RID) at the end
    - RIDs start at 1000 (built-in account RIDs are pre-defined)
- Some local users and groups have pre-defined SIDs (eg. World = S-1-1-0)

Demo: SIDs

- Example SIDs

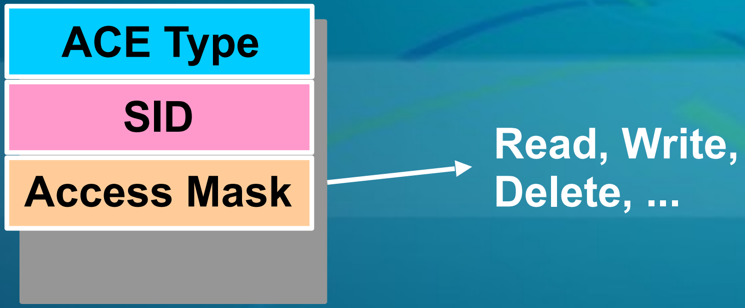
Domain SID:	**S-1-5-21-34125455-5125555-1251255**
First account:	**S-1-5-21-34125455-5125555-1251255-1000**
Admin account:	**S-1-5-21-34125455-5125555-1251255-500**
System account:	**S-1-5-18**



```
C:\Users\Administrator>whoami /all

USER INFORMATION
----------------

User Name                     SID
============================= =============================================
win-jwbppzsxefv\administrator S-1-5-21-1367486129-1636748403-2738611465-500
```

DACLs

- DACLs consist of zero or more Access Control Entries
  - A security descriptor with no DACL allows all access
  - A security descriptor with an empty (0-entry) DACL denies everybody all access
- An ACE is either “allow” or “deny”

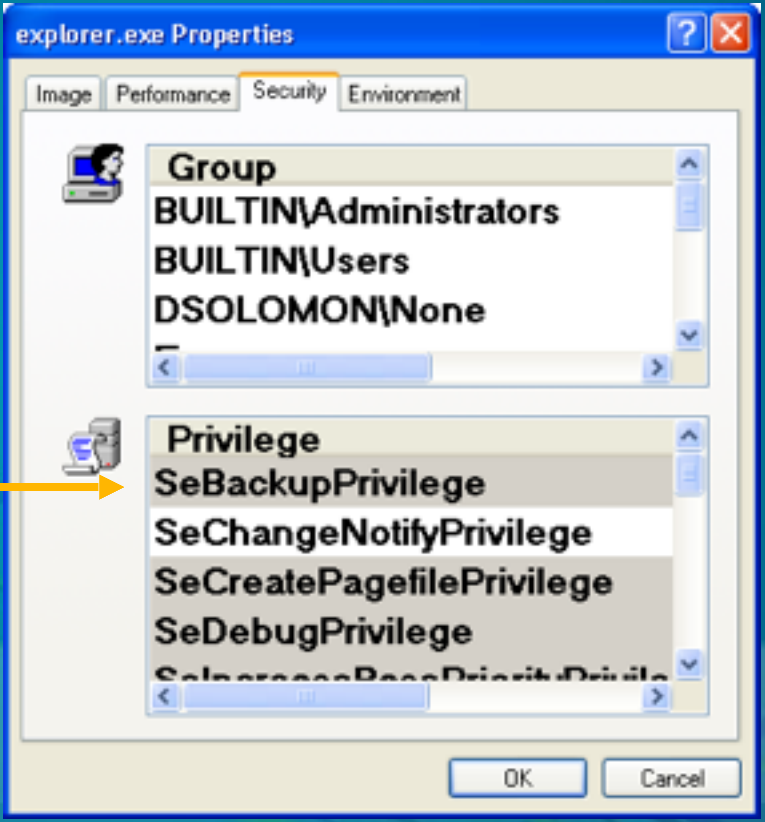


Security

- Introduction
- Components
- Logon
- Protecting Objects
- Privileges

Privileges

- Specify which system actions a process (or thread) can perform
- Privileges are associated with groups and user accounts
  - There are sets of pre-defined privileges associated with built-in groups (e.g. System, Administrators)
- Examples include:
  - Backup/Restore
  - Shutdown
  - Debug
  - Take ownership
- Privileges are disabled by default and must be programmatically turned on with a system call



explorer.exe Properties

Group
BUILTIN\Administrators
BUILTIN\Users
DSOLOMON\None

Privilege
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreatePagefilePrivilege
SeDebugPrivilege
SeIncreaseBasePriorityPrivile

Demo: Privileges

- Run Secpol.msc and examine full list
  - Click on Local Policies->User Rights assignment
- Process Explorer: double click on a process, go to security tab, and examine privileges list
- Watch changes to privilege list:
  1. Run Process Explorer – put in paused mode
  2. Open Control Panel applet to change system time
  3. Go back to Process Explorer & press F5
  4. Examine privilege list in new process that was created
  5. Notice in privilege list that system time privilege is enabled

Powerful Privileges

- There are several privileges that gives an account that has them full control of a computer:
  - Debug: can open any process, including System processes to
    - Inject code
    - Modify code
    - Read sensitive data
  - Take Ownership: can access any object on the system
    - Replace system files
    - Change security
  - Restore: can replace any file
  - Load Driver
    - Drivers bypass all security
  - Create Token
    - Can spoof any user (locally)
    - Requires use of undocumented NT API
  - Trusted Computer Base (Act as Part of Operating System)
    - Can create a new logon session with arbitrary SIDs in the token

Demo: Powerful Privileges

- View the use of the backup privilege:
  - Make a directory
  - Create a file in the directory
  - Use the security editor to remove inherited security and give Everyone full access to the file
  - Remove all access to the directory (do not propagate)
  - Start a command-prompt and do a “dir” of the directory
  - Run \Sysint\Solomon\PView and enable the Backup privilege for the command prompt
  - Do another “dir” and note the different behavior
- View the use of the Bypass-Traverse Checking privilege (internally called “Change Notify”)
  - From the same command prompt run notepad to open the file (give the full path) in the inaccessible directory
  - Extra credit: disable Bypass-Traverse Checking so that you get access denied trying to open the file (hint: requires use of secpol.msc and then RUNAS)