Windows Internals and Advanced Troubleshooting

Part 1: Kernel Architecture

Mark Russinovich
Winternals Software
David Solomon
David Solomon Expert Seminars

1-1

Purpose of Tutorial

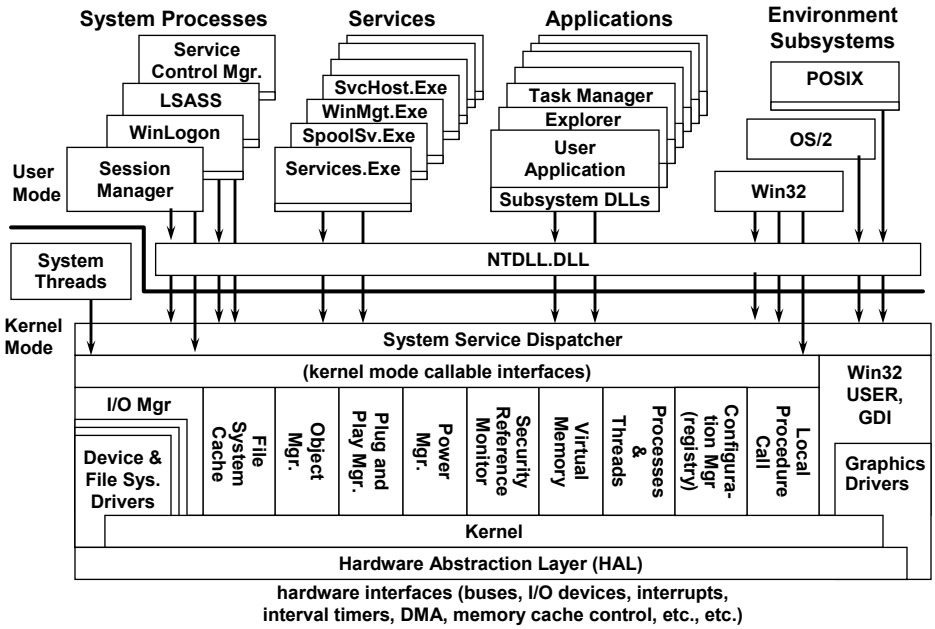
- Give IT Professionals a foundation understanding of the Windows OS kernel architecture
  - Note: this is a small, but important part of Windows
    - The “plumbing in the boiler room”
  - Condensed from a 5 day internals class
- Benefits:
  - Able to troubleshoot problems more effectively
  - Understand system performance issues
- Applies to NT4, Windows 2000, Windows XP, and Windows Server 2003

1-3

Outline

1. Kernel Architecture
2. Troubleshooting Processes and Threads
3. Troubleshooting Memory Problems
4. Crash Dump Analysis

Kernel Architecture



Tools used to dig in

- Many tools available to dig into Windows 2000/XP internals
  - Helps to see internals behavior “in action”
- We’ll use these tools to explore the internals
  - Many of these tools are also used in the labs that you can do after each module
- Several sources of tools
  - Support Tools
  - Resource Kit Tools
  - Debugging Tools
  - Sysinternals.com
  - Inside Windows 2000, 3rd edition book CD
- Additional tool packages with internals information
  - Platform Software Development Kit (SDK)
  - Device Driver Development Kit (DDK)

1-6

Windows XP

- Six variants:

1. Windows XP Professional: replaces Windows 2000 Professional
2. Windows XP Home Edition (new)
   - First consumer focused release of NT
   - Replaces Windows ME (Millenium Edition)
   - Has slightly less features than Windows XP Professional
3. Windows XP Professional 64-bit Edition (new)
   - First 64-bit version of NT - 64-bit pointers, much larger address space
   - Runs on Intel Itanium & Itanium 2 (later: AMD Opteron)
4. Windows XP Embedded
   - Same kernel as regular 32-bit XP
   - Configurable to remove unnecessary components
   - Boot and execute from ROM (OS runs from RAM, apps from ROM)
5. Windows XP Media Center Edition
6. Windows XP Tablet PC Edition

1-7

Windows Server 2003

- Replacement for Windows 2000 Server family
- Name changes for flavors

Windows Server 2003, Web Edition	(new package)
Windows Server 2003, Standard Edition	(was Server)
Windows Server 2003, Enterprise Edition	(was Advanced Server)
Windows Server 2003, Datacenter Edition	(no change)

- New features:
  - More scalable: 64 processor systems, 8 node clusters, larger memory maximums
  - IIS 6.0 (HTTP in the kernel, Connection failover)
  - Active Directory enhancements
  - Many new group policies
  - Remote Installation Support (RIS)
  - Bundles .NET Framework

1-8

Level Of Kernel Change

- Windows .NET Server 2003 & Windows XP are modest upgrades as compared to the changes from Windows NT 4.0 to Windows 2000
- Kernel architecture is basically unchanged
  - No new subsystems
  - No new API sets
- Internal version numbers confirm this
  - Windows 2000 is 5.0
  - Windows XP is 5.1 (not 6.0)
  - Windows .NET Server is 5.2
    - Not the same kernel as XP (a superset)
- But, nonetheless, still lots of interesting kernel changes…

1-9

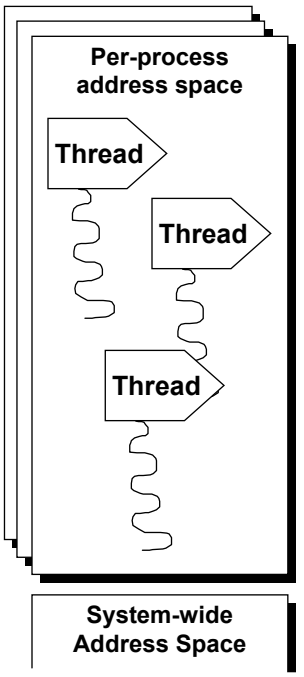
Kernel Architecture

- Process Execution Environment
- Architecture Overview
- Interrupt Handling & Time Accounting
- System Threads
- Process-based code
- Summary

1-10

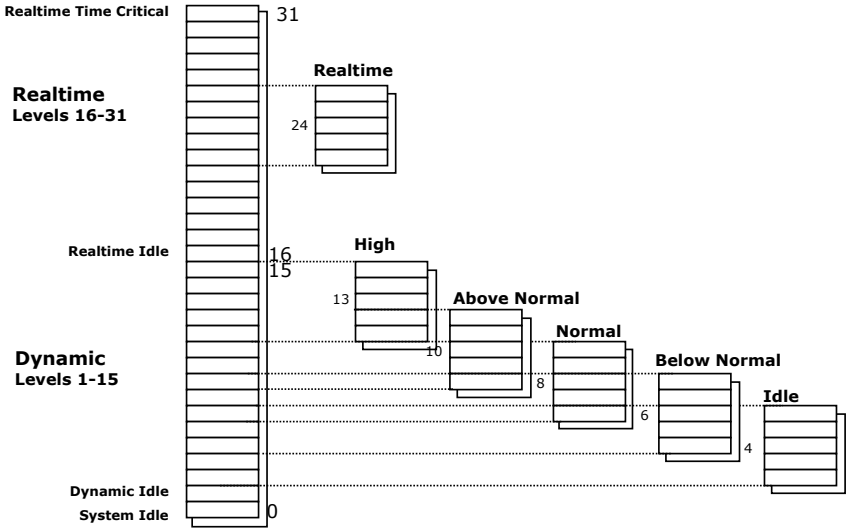
Processes And Threads

- 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)
- 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



1-11

Scheduling Priorites

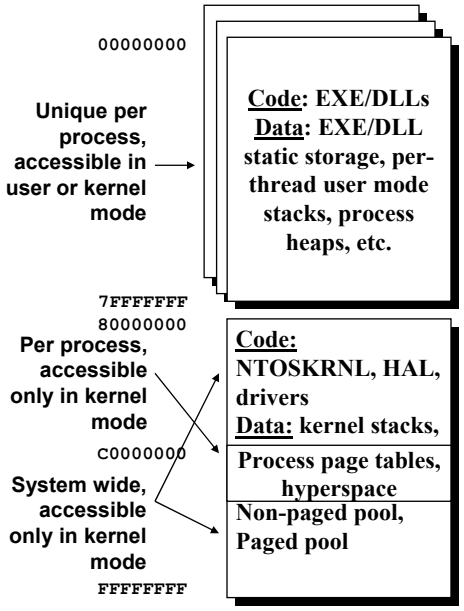


1-12

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
- Questions
  - Given a single threaded application, will adding a second processor make it run faster?
  - Will a multithreaded application run faster on an MP system?
    - Depends if application internal synchronization permits this
    - Having too many runnable threads causes excess context switching

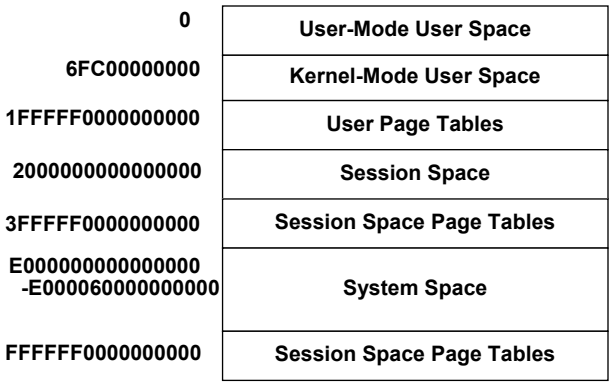
1-13



32-Bit Virtual Address Space (x86)

- 2 GB per-process
  - Address space of one process is not directly reachable from other processes
- 2 GB system-wide
  - The operating system is loaded here, and appears in every process's address space
  - The operating system is not a process (though there are processes that do things for the OS, more or less in "background")
- 3 GB user space and Address Windowing Extensions (AWE) t.b.d.

1-14



64-Bit Virtual Address Space (Itanium)

	64-bit Windows	32-bit Windows
User Address Space	7152 GB	2 or 3 GB
System PTE Space	128 GB	2 GB
System Cache	1 TB	960 MB
Paged pool	128 GB	650 MB
Non-paged pool	128 GB	256 MB

1-15

Memory Protection Model

- No user process can touch another user process' address space
  - Without first opening the process (means passing through NT security)
- All kernel components share a single address space
  - This is how driver bugs can cause 'blue screens'
  - Most other commercial OSs (Unix, Linix, VMS etc.) have the same design

1-16

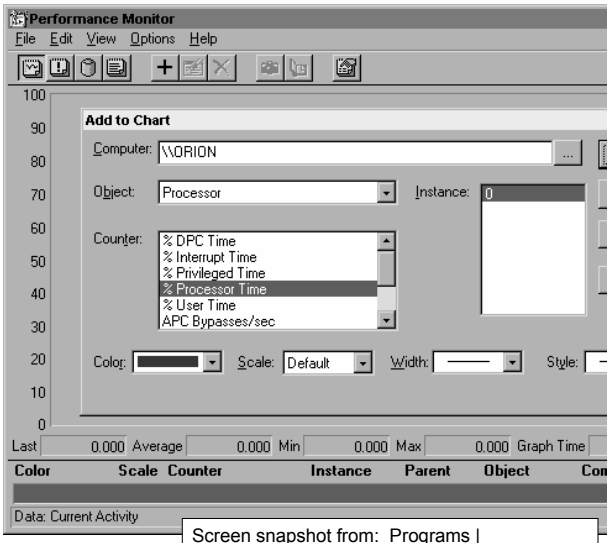
Memory Protection Model

- Controlled by using two hardware access modes: user and kernel
  - X86: Ring 0, Ring 3
  - Itanium: Privilege Level 0 & 3
  - Each memory page is tagged to show the required mode for access
- Associated with threads
  - Threads can change from user to kernel mode and back (via a secure interface)
  - Part of saved context, along with registers, etc.
  - Does not affect scheduling

1-17

Accounting for Kernel-Mode Time

- "Processor Time" = total busy time of processor (equal to elapsed real time - idle time)
- "Processor Time" = "User Time" + "Privileged Time"
- "Privileged Time" = time spent in kernel mode
- "Privileged Time" includes:
  - Interrupt Time
  - DPC Time
  - Explained later…

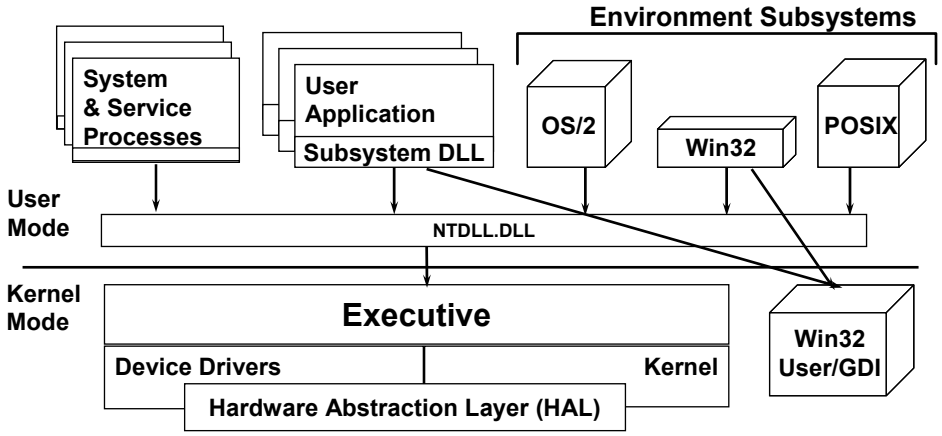


Screen snapshot from: Programs | Administrative Tools | Performance Monitor click on "+" button, or select Edit | Add to chart...

Kernel Architecture

- Process Execution Environment
- Architecture Overview
- Interrupt Handling & Time Accounting
- System Threads
- Process-based code
- Summary

Multiple OS Personality Design



1-20

Environment Subsystems

- Windows NT 4.0 shipped with three environment subsystems
  - Win32 – 32-bit Windows API
  - 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 Windows XP/Server 2003 – enhanced version ships with Services For Unix 3.0
- Of the three, Win32 provides access to the majority of the native functions
- Of the three, Win32 is required to be running
  - System crashes if Win32 subsystem process exits
  - POSIX and OS/2 subsystems are Win32 programs
  - POSIX and OS/2 start on demand (first time an app is run)
    - Stay running until system shutdown

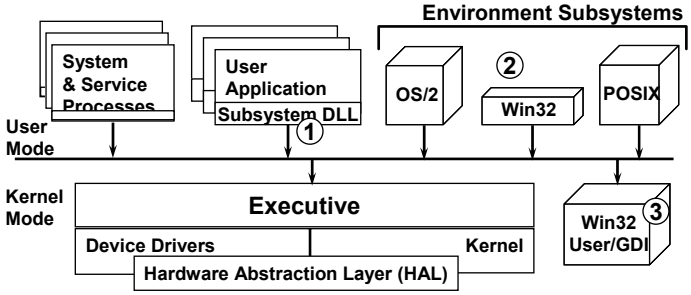
1-21

Subsystem Components

①API DLLs
- For Win32: Kernel32.DLL, Gdi32.DLL, User32.DLL, etc.

②Subsystem process
- For Win32: CSRSS.EXE (Client Server Runtime SubSystem)

③For Win32 only: kernel-mode GDI code
- Win32K.SYS – (this code was formerly part of CSRSS)



1-22

Role Of Subsystem Components

① API DLLs
- Export the APIs defined by the subsystem
- Implement them by calling Windows “native” services, or by asking the subsystem process to do the work

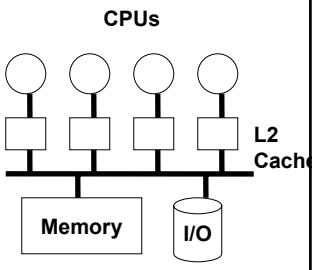
② Subsystem process
- Maintains global state of subsystem
- Implements a few APIs that require subsystem-wide state changes
  - Processes and threads created under a subsystem
  - Drive letters
  - Window management for apps with no window code of their own (character-mode apps)
  - Handle and object tables for subsystem-specific objects

③ Win32K.Sys
- Implements Win32 User and GDI functions; calls routines in GDI drivers
- Also used by Posix and OS/2 subsystems to access the display

1-23

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
- Current implementation supports up to 32 CPUs (64-bit edition is 64 internally)
  - Not an architectural limit—just implementation
  - Maximum # of CPUs stored in registry HKLM\System\CurrentControlSet \Control\Session Manager \LicensedProcessors



SMP

1-24

SMP Scalability

- Scalability is a function of parallelization and resource contention
  - Can't make a general statement
  - Different for different applications (e.g., file server versus SQL versus Exchange)
- Windows kernel provides a scalable foundation
  - Multiple threads of execution within a single process, each of which can execute simultaneously on different processors
  - Ability to run operating system code 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
  - Multiple programming mechanisms that facilitate scalable server applications (e.g. I/O completion ports)

1-25

SMP Scalability

- More efficient locking mechanism (pushlocks)
- Minimized lock contention for hot locks
  - E.g., PFN (Page Frame Database) lock
- Some locks completely eliminated
  - Charging nonpaged/paged pool quotas, allocating and mapping system page table entries, charging commitment of pages, allocating/mapping physical memory through AWE functions
- Even better in Server 2003:
  - Further reduction of use of spinlocks & length they are held
  - Dispatcher (scheduling) database locking now per-CPU

1-26

New MP Configurations

- NUMA (non uniform memory architecture) systems
  - Groups of physical processors (called “nodes”) that have “local memory”
  - 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
- Hyperthreading support
  - CPU fools OS into thinking there are multiple CPUs
    - Example: dual Xeon with hyperthreading can support 2 logical processors
  - Windows Server 2003 is hyperthreading aware
    - Logical processors don’t count against physical processor limits
    - Scheduling algorithms take into account logical vs physical processors

1-27

Many Packages…

1. Windows XP Home Edition
   - 1 CPU, 4GB RAM
2. Windows 2000 & XP Professional
   - Desktop version (but also is a fully functional server system)
   - 2 CPUs, 4GB RAM
3. Windows Server 2003, Web Edition (new)
   - Reduced functionality Standard Server (no domain controller)
   - 2 CPUs, 2GB RAM
4. Windows 2000 Server/Windows Server 2003, Standard Edition
   - Adds server and networking features (active directory-based domains, host-based mirroring and RAID 5, NetWare gateway, DHCP server, WINS, DNS, …)
   - Also is a fully capable desktop system
   - 4 CPUs (2 in Server 2003), 4GB RAM
5. Windows 2000 Advanced Server/Windows Server 2003, Enterprise Edition
   - 3GB per-process address space option, Clusters (8 nodes)
   - 8 CPUs, 8GB RAM (32GB in Server 2003 32-bit; 64GB on 64-bit)
6. Windows 2000/Server 2003 Datacenter Edition
   - Process Control Manager
   - Licensed for 32 CPUs, 64GB RAM (128GB on 64-bit edition)

1-28

…But one OS

- Through Windows 2000, core operating system executables are identical
  - NTOSKRNL.EXE, HAL.DLL, xxxDRIVER.SYS, etc.
  - XP & Server 2003 have different kernel versions, but not substantially different
- 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 Windows NT 4.0: Enterprise Edition, Terminal Server, …)
- Code in the operating system tests these values and behaves slightly differently in a few places
  - Licensing limits (number of processors, number of inbound network connections, etc.)
  - Boot-time calculations (mostly in the memory manager)
  - Default length of time slice

1-29

NTOSKRNL.EXE

- Core operating system image
  - Contains Executive and Kernel
- Kernel versions
  - Windows NT 4.0 is 4.0 (client and server)
  - Windows 2000 is 5.0 (client and server)
  - Windows XP is 5.1 (client only)
  - Windows Server 2003 is 5.2 (server only)
- Kernel evolution
  - NT4->Windows 2000 – significant change
  - Windows 2000->Windows XP – modest change
  - Windows XP->Server 2003 – minimal change

1-30

NTOSKRNL Variants

- Four variations:
  - 4GB or less

    NTOSKRNL.EXE Uniprocessor
    NTKRNLMP.EXE Multiprocessor
  - >4GB (new as of Windows 2000)

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

1-31

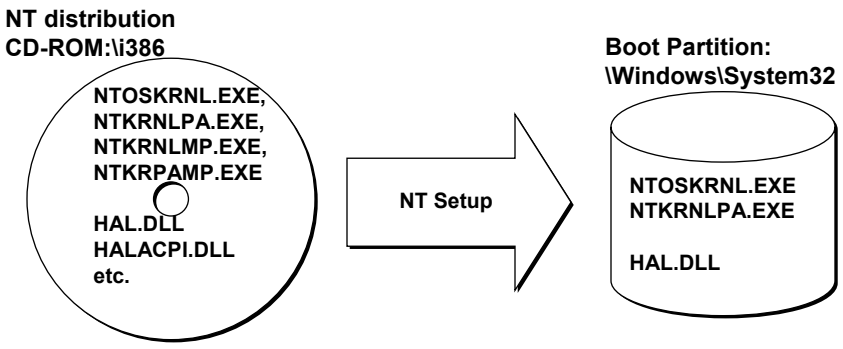
HAL – Hardware Abstraction Layer

- 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

1-32

NTOSKRNL And HAL Selection

- 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

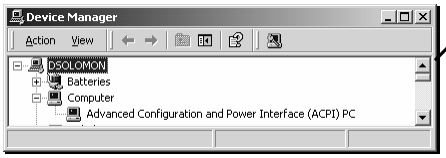


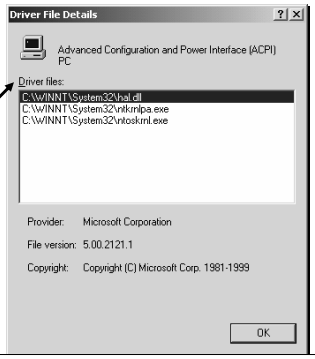
(see \windows\repair\setup.log)

1-33

NTOSKRNL And HAL Selection

- NTOSKRNL & HAL considered to be the “device drivers” for the “computer”
  - Go to Control Panel->System – Hardware tab
  - Click on “Device Manager”
  - Click on “Computer”
  - Right click/Properties on “driver” for PC





Screen snapshot from:
Control Panel | System | Hardware | Device Manager | Computer properties | Driver Details

Debug Version
“Checked Build”

- Special debug version of system called “Checked Build”
  - Provided with MSDN
  - Primarily for driver testing, but can be useful for catching timing bugs in multithreaded applications
- Built from same source files as “free build” (a.k.a., “retail build”)
  - “DBG” compile-time symbol defined which 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
```

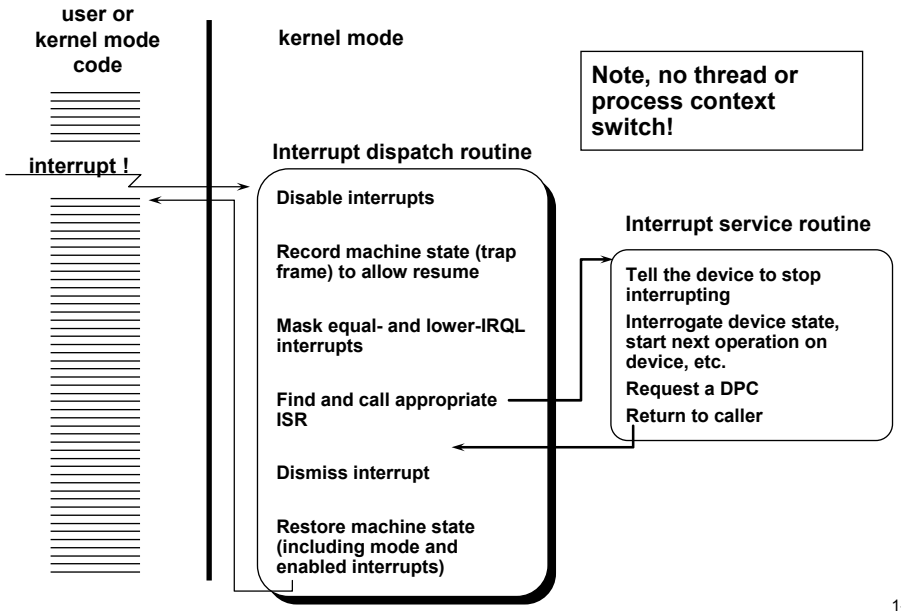
  - Multiprocessor kernel (of course, runs on UP systems)
- Since no checked Server CD provided, can copy checked NTOSKRNL, HAL, to a normal Server system
  - Select debug kernel and HAL with Boot.ini /KERNEL=, /HAL= switches
- See Knowledge base article 314743 (HOWTO: Enable Verbose Debug Tracing in Various Drivers and Subsystems)

Kernel Architecture

- Process Execution Environment
- Architecture Overview
- Interrupt Handling & Time Accounting
- System Threads
- Process-based code
- Summary

1-36

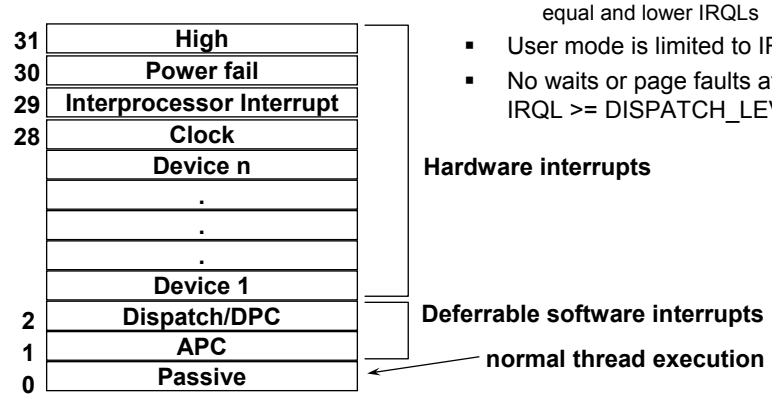
Interrupt Dispatching



1-37

Interrupt Precedence Via IRQLs

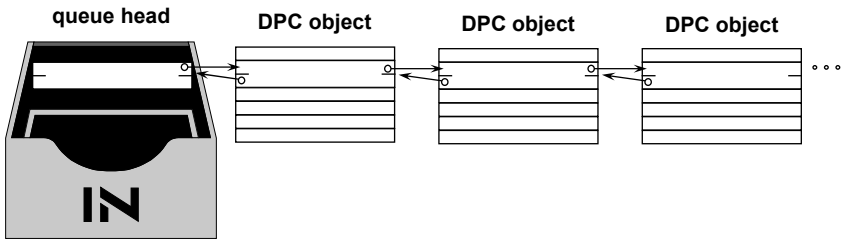
- IRQL = Interrupt Request Level
  - The “precedence” of the interrupt with respect to other interrupts
  - Different interrupt sources have different IRQLs
  - Not the same as IRQ
- IRQL is also a state of the processor
- Servicing an interrupt raises processor IRQL to that interrupt’s IRQL
  - This masks subsequent interrupts at equal and lower IRQLs
- User mode is limited to IRQL 0
- No waits or page faults at IRQL >= DISPATCH_LEVEL



1-38

Deferred Procedure Calls (DPCs)

- Used to defer processing from higher (device) interrupt level to a lower (dispatch) level
  - Driver (usually ISR) queues request
  - One queue per CPU; DPCs are normally queued to the current processor, but can be targetted to other CPUs
  - Executes specified procedure at dispatch IRQL (or “dispatch level”, also “DPC level”) when all higher-IRQL work (interrupts) completed
- Used heavily for driver “after interrupt” functions
  - Also used for quantum end and timer expiration



1-39

Interrupt Time Accounting

- Time servicing interrupts are NOT charged to interrupted thread
  - Time spent at IRQL 2 appears as “% DPC time”
  - Time spent at IRQL >2 appears as “% interrupt time”
  - Hence no process appears to be running
- What if system is not idle, but no process appears to be running?
  - Must be due to interrupt-related activity
- Performance counters (Processor object):
  - % Interrupt time – time spent processing hardware interrupts
  - % DPC time – software generated interrupts
  - Can also look at Interrupts/sec & DPCs Queued/sec

1-40

Time Accounting Quirks

- Looking at total CPU time for each process may not reveal where system has spent its time
- CPU time accounting is driven by programmable interrupt timer
  - Normally 10 msec (15 msec on some MP Pentiums)
- Thread execution and context switches between clock intervals NOT accounted
  - E.g., one or more threads run and enter a wait state before clock fires
  - Thus threads may run but never get charged

1-41

Kernel Architecture

- Process Execution Environment
- Architecture Overview
- Interrupt Handling & Time Accounting
- System Threads
- Process-based code
- Summary

1-42

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 XP, windowing system threads (from Win32k.sys) appear in “csrss.exe” (Win32 subsystem process)

1-43

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

Understanding System Threads

- Later we'll see how to understand what system thread is running when the System process is consuming CPU time…

Kernel Architecture

- Process Execution Environment
- Architecture Overview
- Interrupt Handling & Time Accounting
- System Threads
- Process-based code
- Summary

1-46

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
  - Win32 Services
- Let’s examine the system process “tree”
  - Use Tlist /T or Process Explorer

1-47

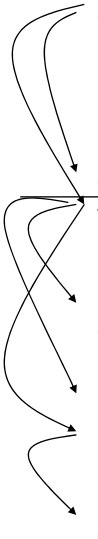
Process-Based NT Code
System Startup Processes

- 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

1-48

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	Win32 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

1-49

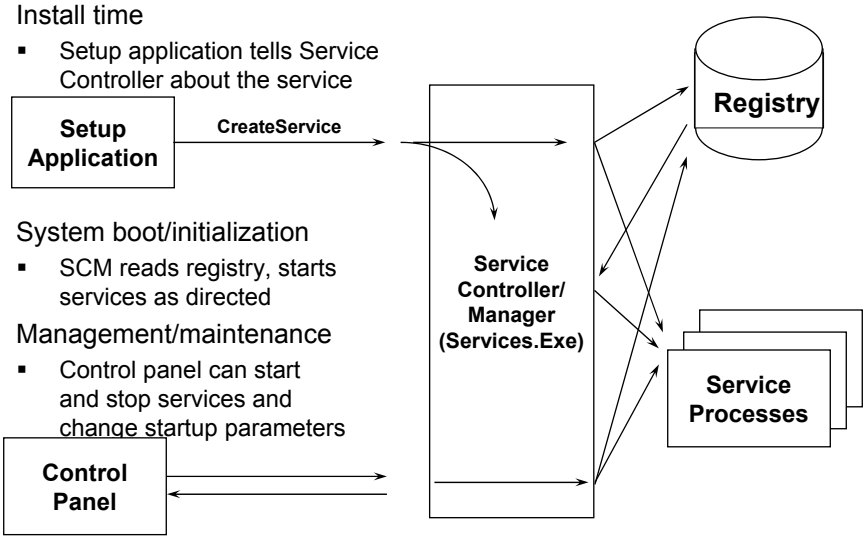
Win32 Services

- 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)

1-50

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



1-51

Mapping Services to Service Processes

- Service properties displayed through Control Panel (services.msc) show name of .EXE
  - But not which process started services are in
- Tlist /S or Tasklist /svc (new as of XP) list internal name of services inside service processes
- Process Explorer shows both internal and external name

1-52

Services Infrastructure Improvements

- Two new less privileged accounts for built-in services
  - LOCAL SERVICE, NETWORK SERVICE
  - Less rights than LocalSystem
    - Reduces possibility of damage if system compromised
- More services run in generic service host process (svchost.exe)
  - Reduces number of processes
- Four instances (at least)
  - SYSTEM
  - SYSTEM (2nd instance – for RPC)
  - LOCAL SERVICE
  - NETWORK SERVICE
- Later we'll see how to understand WHICH service is consuming CPU time when a multi-service process is running

1-53

Logon Process

1. 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
2. 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
3. Userinit creates a process to run
   HKLM\Software\Microsoft\Windows NT
   \CurrentVersion\WinLogon\Shell
   - By default: Explorer.exe

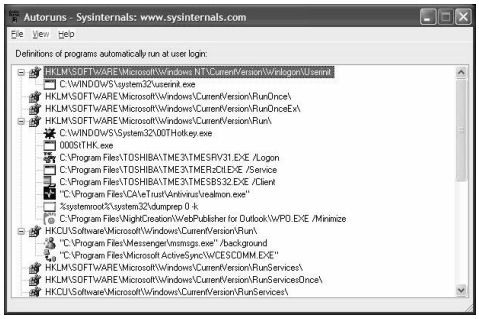
- There are other places in the Registry that control programs that start at logon

1-54

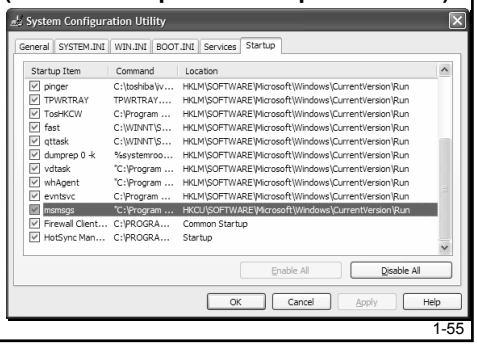
Processes Started at Logon

- 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)



Msconfig
(in \Windows\pchealth\helpctr\binaries)

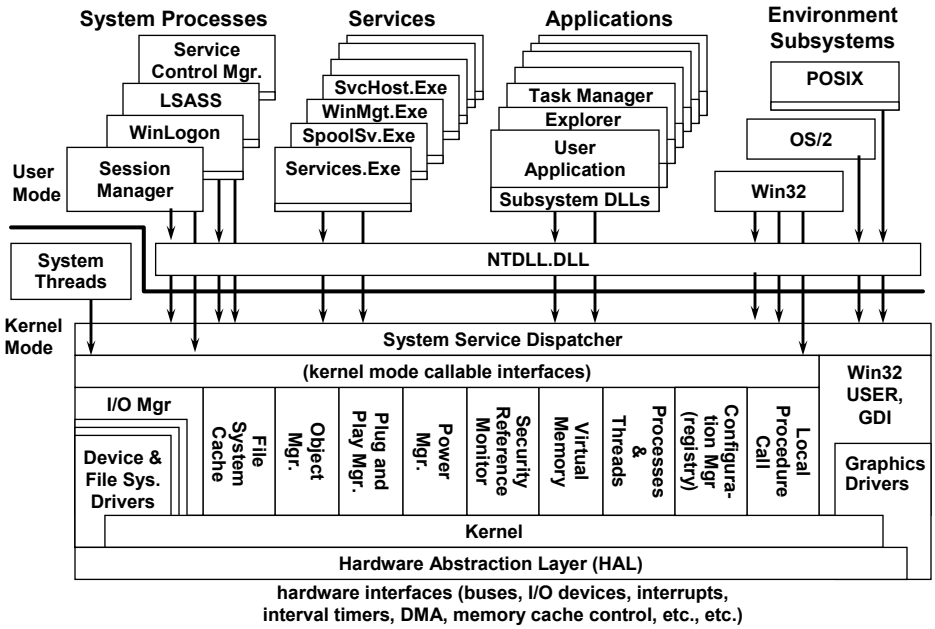


1-55

Kernel Architecture

- Process Execution Environment
- Architecture Overview
- Interrupt Handling & Time Accounting
- System Threads
- Process-based code
- Summary

Kernel Architecture



Four Contexts For Executing Code

- Full process and thread context
  - User applications
  - Win32 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

1-58

Core Kernel System Files

- Kernel32.Dll, Gdi32.Dll, User32.Dll
  - Export Win32 entry points
- NtDll.Dll
  - Provides user-mode access to system-space routines
  - Also contains heap manager, image loader, thread startup routine
- NtosKrnl.Exe (or NtkrnlMp.Exe)
  - Executive and kernel
  - Includes most routines that run as threads in “system” process
- Win32K.Sys
  - The loadable module that includes the now-kernel-mode Win32 code (formerly in csrss.exe)
- Hal.Dll
  - Hardware Abstraction Library
- drivername.Sys
  - Loadable kernel drivers

1-59

End of Kernel Architecture

Next: Process & Thread Troubleshooting

1-60

Windows Internals and Advanced Troubleshooting

Part 2: Troubleshooting Processes & Threads

1-1

Agenda

- Introduction to Tools
- Identifying the Process
- Analyzing Process/Thread Activity
- Application Failures

1-2

Tools for Obtaining Process & Thread Information

- Many overlapping tools (most show one item the others do not)
- Built-in tools in Windows 2000/XP:
  - Task Manager, Performance Tool
  - Tasklist (new in XP)
- Support Tools
  - pviewer - process and thread details (GUI)
  - pmon - process list (character cell)
  - tlist - shows process tree and thread details (character cell)
- Resource Kit tools:
  - apimon - system call and page fault monitoring (GUI)
  - oh – display open handles (character cell)
  - pviewer - processes and threads and security details (GUI)
  - ptree – display process tree and kill remote processes (GUI)
  - pulist - lists processes and usernames (character cell)
  - pstat - process/threads and driver addresses (character cell)
  - qslice - can show process-relative thread activity (GUI)
- Tools from www.sysinternals.com
  - Process Explorer – super Task Manager – shows open files, loaded DLLs, security info, etc.
  - Pslist – list processes on local or remote systems
  - Ntpmon - shows process/thread create/deletes (and context switches on MP systems only)
  - Listdlls - displays full path of EXE & DLLs loaded in each process

1-3

Tools We'll Look At

- Task Manager – see what's using CPU
- Process Explorer (Procexp) – view process details
- Filemon – monitors file I/O
- Regmon – monitors registry I/O
- Pssuspend – suspends a proces
- Strings – dumps printable strings in files

1-4

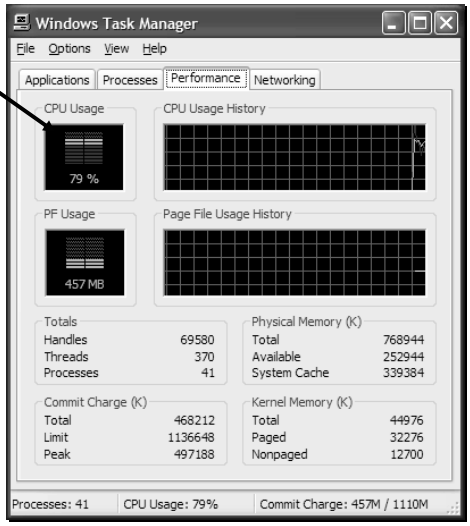
Agenda

- Introduction & Data Structures
- Identifying the Process
- Analyzing Process/Thread Activity
- Application Failures

1-5

The CPU Is Busy – Why?

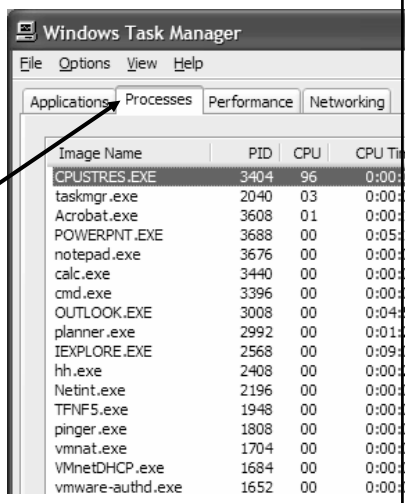
- System is busy (may be slow)
- What is running?
  - A user or system process?
  - Interrupt activity?
- What's it doing?
  - File I/O? Network I/O? Registry calls?
  - Application code?



1-6

Which Process Is Running?

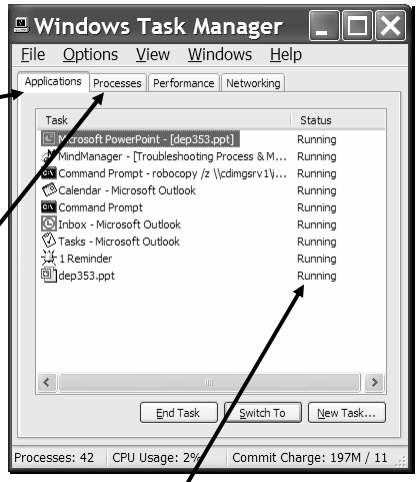
- Determine which process' threads are consuming the most CPU time
- Quick method:
  - Open Task Manager ->Processes
  - Sort processes by "CPU" usage column
- Other tools
  - Qslice.exe (Resource Kit)
  - Performance Monitor (monitor %Processor Time counter in process object for all processes)



1-7

Task Manager: Applications vs. Processes

- Applications tab: List of top level visible windows
  - Windows are owned by threads
  - Right-click on a window and select "Go to process"
- Processes tab: List of processes
  - Can configure with View->Select columns



"Running" means waiting for window messages

1-8

Dealing with a CPU Hog

- Option 1: Try and figure out what it's doing using monitoring tools explained later in this talk
- Option 2: Lower the priority
- Option 3: Suspend the process with PsSuspend
  - Another use: you've started a long running job but want to pause it to do something else
    - Lowering the priority still leaves it running…
- Option 4: Kill the process

1-9

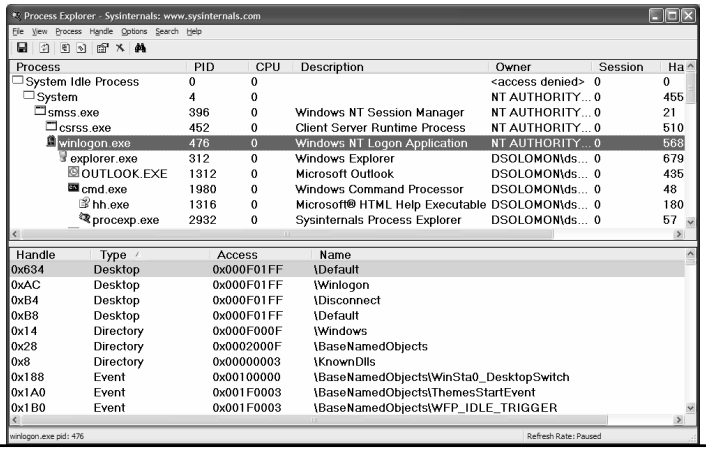
Identify The Image

- Once you've found the process of interest, what is it?
  - Sometimes name of .EXE identifies clearly (e.g., Winword.exe)
  - Often, it doesn't since Task Manager doesn't show the full path of the image
- We need more information!

1-10

Process Explorer (Sysinternals)

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



1-11

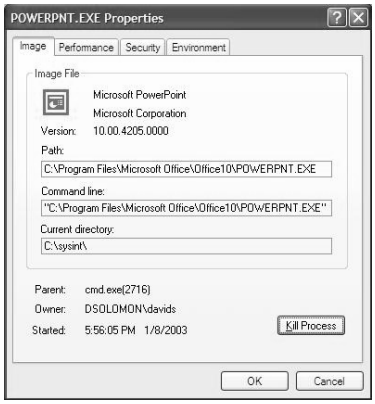
Process Explorer

- Process tree
  - If left justified, parent has exited
  - Disappears if you sort by any column
    - Bring back with View->Show Process Tree
- Additional details in process list
  - Icon and description (from .EXE)
  - User Name shows domain name
- Highlight Own, Services Processes
- Differences highighting
  - Green: new, Red: gone
  - View->Update speed->Paused

1-12

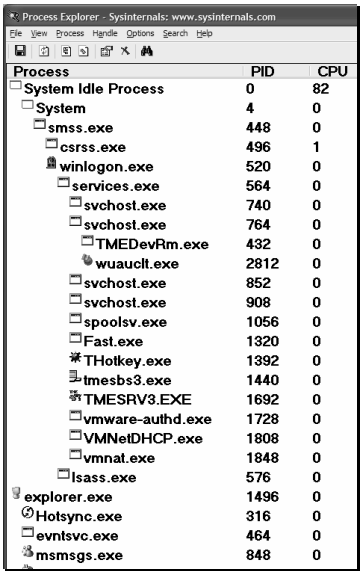
Process Properties

- Image tab:
  - Description, company name, version (from .EXE)
  - Full image path
  - Command line used to start process
  - Current directory
  - Parent process
  - User name
  - Start time
- Performance tab:
  - Basic process CPU/memory usage
- Security tab:
  - Access token (groups list, privilege list)
- Environment tab: environment variables
- Services tab (only for service processes):
  - List of services hosted by process



Process Tree

- System keeps track of parent/child relationship
- What if parent exits?
  - System only keeps track of parent PID
  - If parent exits, no way to find its ancestors (without a trace of process creations)
  - Process Explorer shows orphans left justified



Handle and DLL Views

- Lower half of display shows either:
  - Open handles
  - Loaded DLLs & mapped files
- Handle View
  - Sort by handle
  - Objects of type “File” and “Key” are most interesting for general troubleshooting
- DLL View
  - Shows loaded DLLs, .EXE, and any memory mapped files

1-15

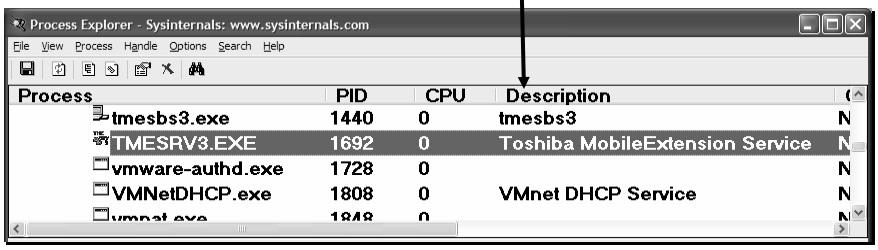
Process Explorer Lab

1. Run Process Explorer
2. Sort on first column (“Process”) and note tree view disappears
3. Click on View->Show Process Tree to bring it back
4. Change update speed to paused
5. Run Notepad
6. In ProcExp, hit F5 and notice new process
7. Find value of PATH environment variable in Notepad
8. Exit Notepad
9. In ProcExp, hit F5 and notice Notepad in red

1-16

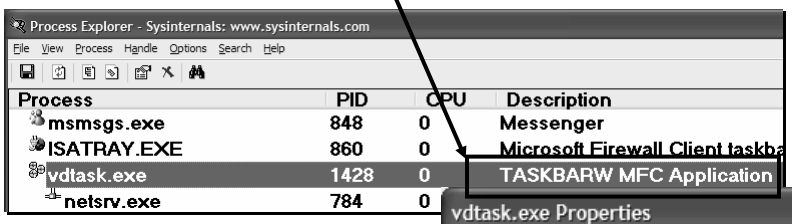
Identify The Image (Continued)

- Check “Description” column in Process Explorer
  - Taken from .EXE header

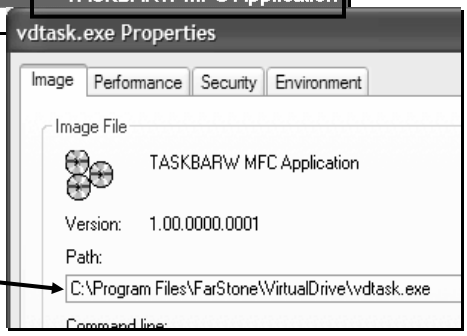


Identify The Image

- Sometimes description is not meaningful

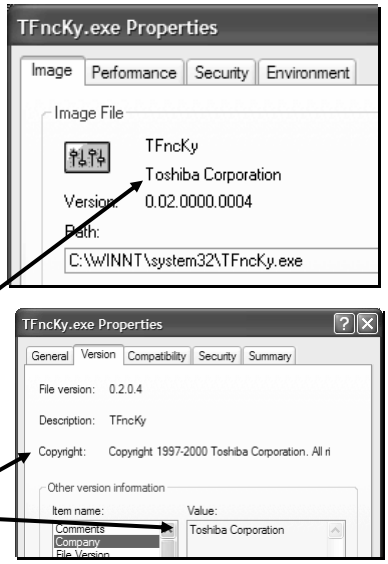


- Check full path of .EXE with Process Explorer
  - Often pinpoints which product



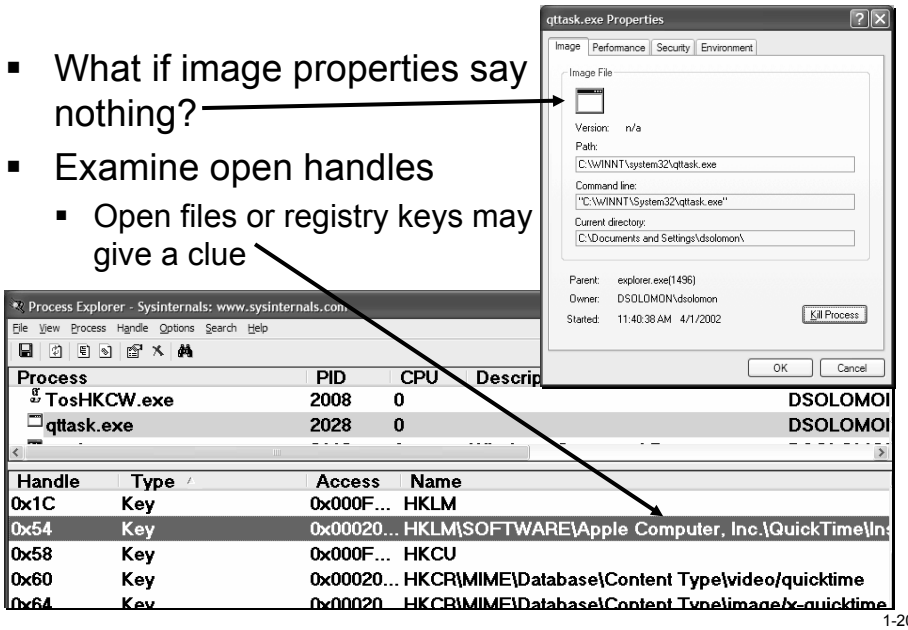
Identify The Image

- Often, applications are installed in \Windows\System32
  - Or in folders with unrecognizable names
- Check company name or copyright
  - Process Explorer: double click on process
  - Explorer->right-click, properties on .EXE



Identify The Image

- What if image properties say nothing?
- Examine open handles
  - Open files or registry keys may give a clue



Identifying Processes

- If you still don't know what the EXE is, run Strings on it
  - Dumps printable strings in binary
- Need to run twice
  - No switches dumps Unicode strings
  - "–a" switch dumps ANSI strings
- Printable strings may yield clues
  - Registry keys
  - Help/error message text

1-21

Agenda

- Introduction & Data Structures
- Identifying the Process
- Analyzing Process/Thread Activity
- Application Failures

1-22

Multi-service Processes

- Some processes host multiple services
  - E.g. Svchost.exe, Inetinfo.exe (IIS)
- If still not clear what process is doing, need to peer inside process and examine which thread(s) are running and what code they are executing
  - With Performance Monitor, monitor %Processor Time for threads inside a process
  - Find thread(s) consuming CPU time

1-23

Analyzing Thread Activity

- Then try and determine what code they are executing by finding which code module the thread started in:
  - 1. Get thread start address with Tlist
  - 2. With Process Explorer DLL view, sort by base address and find in which module the address lies
    - Can also do this with Tlist

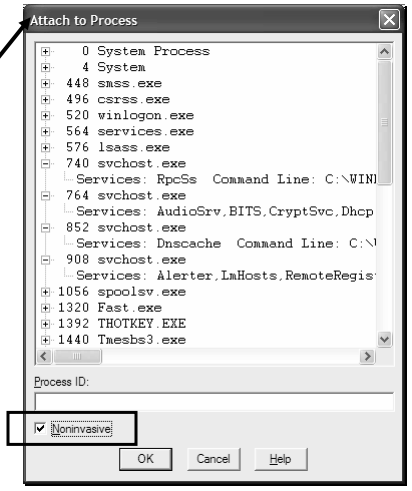
1-24

Analyzing Thread Activity

- Start address may not be enough
  - May need to look at call stack
- Can attach with Windbg or Ntsd and issue "k" command
  - Caution: pre-XP, exiting debugger kills debugee if real debugger attachment
  - Attach "noninvasive"
    - Freezes threads while connected
    - Allows viewing information in process, but not changing data

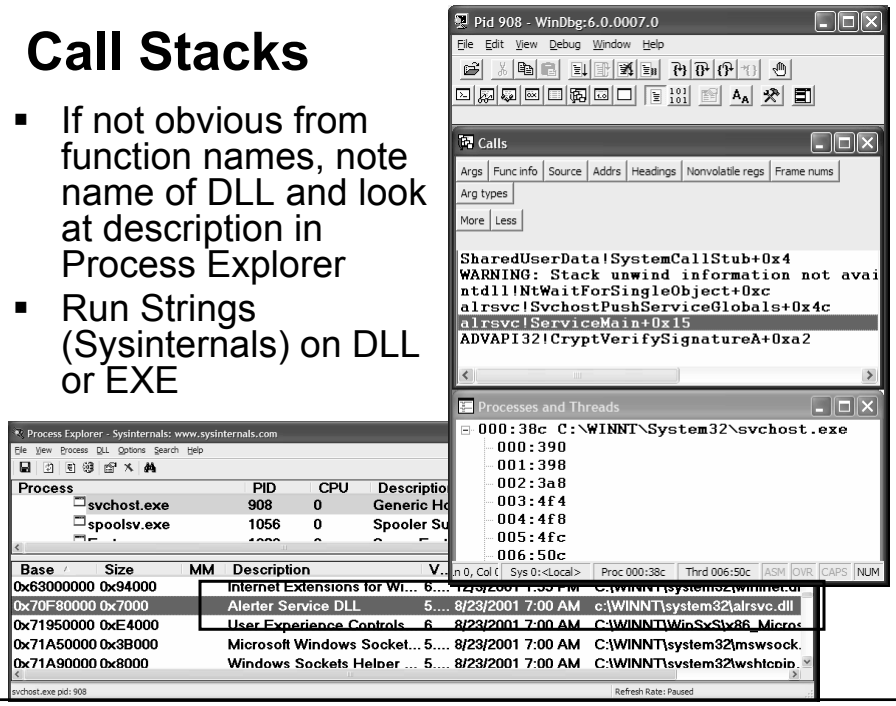
Analyzing Call Stacks

- With Windbg, click on File->Attach to Process
- Then View-> Call Stack
- Then View ->Processes and Threads
  - Select thread of interest



Call Stacks

- If not obvious from function names, note name of DLL and look at description in Process Explorer
- Run Strings (Sysinternals) on DLL or EXE



Examining System Threads

- If System threads are consuming CPU time, cannot use WinDbg to attach to process and examine user stack
  - System threads always run in kernel mode
  - No user stack
- Need to find out what code is running, since it could be any one of a variety of components
  - Memory manager modified page writer
  - Swapper
  - File server worker threads

Examining System Threads

- With user-mode tools:
  1. PerfMon: monitor %Processor time for each thread in System process
  2. Determine which thread(s) are running
  3. From this, get “Start address” (address of thread function) in Pviewer
  4. Run pstat to find which driver thread start address falls in
     - Look for what driver starts near the thread start address

Examining System Threads

- With Kernel Debugger:
  - ln (“List Near”) <startaddress> will give name of driver and function
  - Use !process or !thread to see kernel stack

```
lkd> ln 8061adb8
(8061adb8)   nt!MiModifiedPageWriter   |  (8061af38)
lkd> !process 4
            …
 THREAD 816113e0  Cid 8.50  WAIT: (Executive) KernelMode Non-Alertable
        f5c67d70  NotificationTimer
        80482540  SynchronizationEvent
     Start Address nt!KeBalanceSetManager (0x804634e0)
     Stack Init f5c68000 Current f5c67cc0 Base f5c68000 Limit f5c65000 Call 0
ChildEBP RetAddr  Args to Child
f5c67cd8 8042d5a3 ffffffff ff676980 00000000 nt!KiSwapThread+0xc5
f5c67d0c 8046355e 00000002 f5c67d98 00000001 nt!KeWaitForMultipleObjects+0x266
f5c67da8 80454faf 00000000 00000000 00000000 nt!KeBalanceSetManager+0x7e
f5c67ddc 80468ec2 804634e0 00000000 00000000 nt!PspSystemThreadStartup+0x69
00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x16
```

Agenda

- Introduction & Data Structures
- Identifying the Process
- Analyzing Process/Thread Activity
- Application Failures

1-31

Troubleshooting Application Failures

- Most applications do a poor job of reporting file-related or registry-related errors
  - E.g. permissions problems
  - Missing files
  - Missing or corrupt registry data

1-32

Troubleshooting Application Failures

- When in doubt, run Filemon and Regmon!
  - Filemon monitors File I/O; Regmon monitors registry I/O
- Ideal for troubleshooting a wide variety of application failures
- Also useful for to understand and tune file system access
  - E.g. understanding hard drive activity
- Work on all Windows® OSs
- Used extensively within Microsoft

1-33

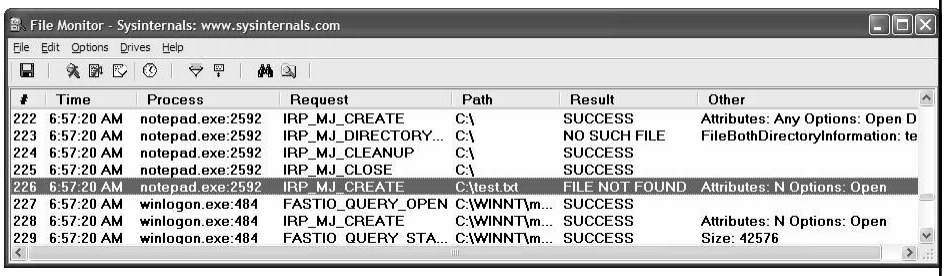
Using Regmon/Filemon

- Two basic techniques:
  - Go to end of log and look backwards to where problem occurred or is evident and focused on the last things done
  - Compare a good log with a bad log
- Often comparing the I/O and Registry activity of a failing process with one that works may point to the problem
  - Have to first massage log file to remove data that differs run to run
    - Delete first 3 columns (they are always different: line #, time, process id)
      - Easy to do with Excel by deleting columns
  - Then compare with FC (built in tool) or Windiff (Resource Kit)

1-34

Filemon

- # - operation number
- Process: image name + process id
- Request: internal I/O request code
- Result: return code from I/O operation
- Other: flags passed on I/O request

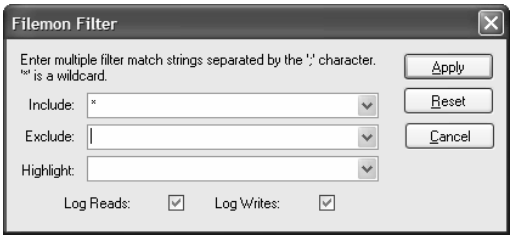


Controlling Filemon

- Start/stop logging (Control/E)
- Clear display (Control/X)
- Open Explorer window to folder containing file:
  - Double click on a line does this
- Find – finds text within window
- Save to log file
- History depth
- Advanced mode

Limiting Filemon Output

- Can set filters for including, excluding, and highlighting output



1-37

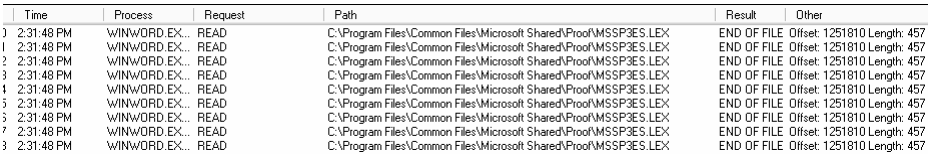
Filemon Lab 1

1. Run Filemon
2. Set filter to only include Notepad.exe
3. Run Notepad
4. Type some text
5. Save file as “test.txt”
6. Go back to Filemon
7. Stop logging
8. Set highlight to “test.txt”
9. Find line representing creation of new file
   - Hint: look for create operation

1-38

Filemon Example

- While typing in the document Word XP closes without any prompts
- Filemon log showed this:



	Time	Process	Request	Path	Result	Other
0	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
1	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
2	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
3	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
4	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
5	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
6	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
7	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457
8	2:31:48 PM	WINWORD.EX...	READ	C:\Program Files\Common Files\Microsoft Shared\Proof\MSSP3ES.LEX	END OF FILE	Offset: 1251810 Length: 457

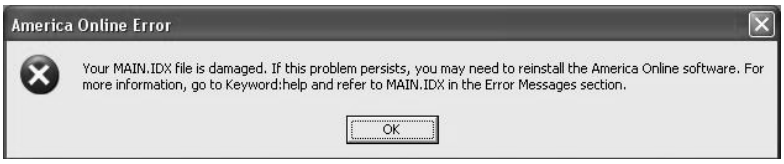
- User looked up what .LEX file was
  - Related to Word proofing tools
  - Uninstalled and reinstalled proofing tools & problem went away

Access Denied

- Many applications don't report access denied errors well
  - Example: try to save a file with Notepad to a folder you don't have access to
- Use Filemon to verify access denied errors are not occurring on file opens
  - Check Result column

Example: Access Denied

- AOL reported this error:



- Filemon showed this:

waol.exe OPEN C:\PROGRA~1\AMERIC~1.0\IDB\main.ind ACCESS DENIED

- User did not have admin rights to AOL directory

1-41

Example: Access Denied

- For example, an application failed with this error:

  Ran Filemon and found it was getting Access Denied

- Someone had misread a request to remove EDIT rights and removed all rights

1-42

Hot File Analysis

- Understand disk activity system-wide
  - Run Filemon for a period of time
  - Save output in a log file
  - Import into Excel and make a pie chart by file name or operation type
- Example: used Filemon on a server to determine which file(s) were being accessed most frequently
  - Moved these files to a different disk on a different controller

1-43

Locked Files

- Attempting to open or delete a file that is in use simply reports “file locked”
  - With Process Explorer search (in handle view) you can determine what process is holding a file or directory open
  - Can even close open files (be careful!)

1-44

Process Explorer Lab: Locked File

1. Run ProcExp
   - Click on View->Update speed – change to Paused
2. Run Microsoft Word
3. Create a file called “test.doc” and save it (but don’t close it)
4. From a command prompt try and delete “test.doc” (should get file locked)
5. In ProcExp, hit F5 and then use Search to find open handle to test.doc

1-45

Access Denied on Mapped Files

- Attempting to delete a DLL or EXE that is in use gets “access denied”, not “file locked”
  - Can be misleading
- In Process Explorer DLL View, search for file
- Example: try and delete Notepad.exe while you’re running it

1-46

DLL Problems

- DLL version mismatches can cause strange application failures
  - Most applications do a poor job of reporting DLL version problems
- Process Explorer can help detect DLL versioning problems
  - Compare the output from a working process with that of a failing one (use File->Save As)

1-47

DLL Problems

- But sometimes it's the order of DLL loads that clues you in, so use Filemon!
  - Missing DLLs often not reported correctly
    - Look for "NOTFOUND" or "ACCESS DENIED"
  - May be opening wrong versions due to files in PATH
- Look at the last DLL opened before the application died

1-48

Example Problem: Word Dies

- Word97 starts and a few seconds later gets a Dr. Watson (access violation)
  - Customer tried re-installing Office – still failed
- Solution:
  - Ran Filemon, looked at last DLL loaded before Dr. Watson
  - It was a printer DLL
  - Uninstalled printer – problem went away

1-49

Example Problem: Help Fails

- The Help command in an application failed on Win95, but worked fine on Win98/ME/NT4/Win2000/WinXP
  - Failed with meaningless error message

1-50

Solution

- Ran Filemon on failing system and working system
  - Reduced log to file opens
  - Compared logs
- At the point they diverged, looked backwards to last common thing done
  - An OLE system DLL was loaded
  - Noticed this OLE DLL was loaded from a directory in the user's PATH on Win95, but from \Windows\System on other versions
- Conclusion:
  - DLL loaded on Win95 system was not for Win95
  - Got proper version for Win95, problem went away

1-51

Example Problem: Access Hangs

- Problem: Access 2000 would hang when trying to import an Excel file
  - Worked fine on other users' workstations
  - User had Access 97 and Access 2000 installed
- Compared a Filemon log from the working and failing system
  - Failing system was loading an old Access DLL from \windows\system32 due to having installed Access 97 previously
  - Removed DLL and problem went away

1-52

Dll Version Mismatch Lab

With Word XP installed in the default folder:

1. Go to folder:
   `\Program Files\Microsoft Office\Office\1033`
2. Rename MSO9INTL.DLL to “MSO9INTL.DLL1”
3. Copy OUTLLIBR.DLL to MSO9INTL.DLL
4. Try and start Word
   - Send error report to Microsoft ☺
5. Use FileMon to confirm which DLL is likely causing the problem

1-53

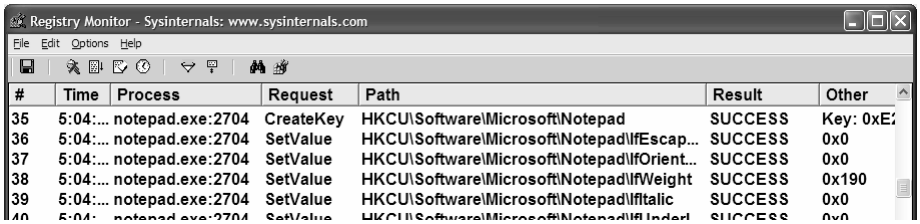
Configuration Problems

- Missing, corrupted or overly-secure Registry settings often lead to application crashes and errors
- Some applications don’t completely remove registry data at uninstall
- Regmon may yield the answer…

1-54

Regmon Output

- Request: OpenKey, CreateKey, SetValue, QueryValue, CloseKey
- Path
  - HKCU=HKEY_CURRENT_USER (per-user settings)
  - HKLM=HKEY_LOCAL_MACHINE (system wide settings)
- Result – return code from Registry operation
- Other – extended information or results

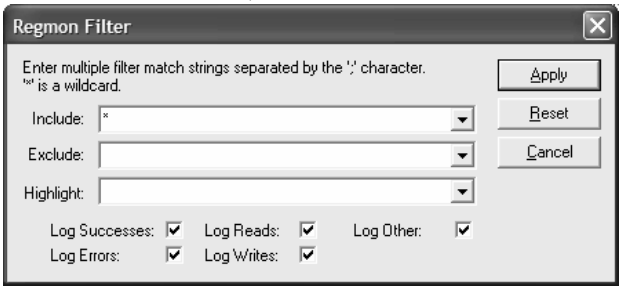


Controlling Regmon

- Start/stop logging (Control/E)
- Clear display (Control/X)
- Regedit jump (opens Registry Editor and jumps directly to key)
  - Double clicking on a line does this
- Filtering/Highlighting
- Find
- Save to log file

Regmon Filtering

- Normally, registry activity should be only at application/system startup and exit
  - But, sadly, lots of processes perform needless registry querying…
- Filtering options:
  - Process name or registry path (or partial name)
  - Success/failure, read/write



1-57

Regmon Lab 1

1. Run Regmon
2. Highlight Notepad.exe
3. Run Notepad
4. Change font to “Times New Roman”
5. Exit
6. Go back to Regmon
7. Stop logging
8. Find line showing storing of font name in registry
   - Hint: search for “times”

1-58

Using Regmon

- Identify missing Registry keys
  - Search for status “NOTFOUND”
- Troubleshoot permission problems
  - Search for status “ACCESS DENIED”
- Find incorrect or corrupt data
  - Examine values read and/or written (in Other column)

1-59

Example Problem

- Internet Explorer failed to start:

- Solution:
  - Looked backwards from end of Regmon log
  - Last queries were to:
    HKCU\Software\Microsoft\Internet Connection Wizard
  - Looked here and found a single value “Completed” set to 0
  - Compared to other users—theirs was 1
  - Set this manually to 1 and problem went away

1-60

Regmon Applications

- If you suspect registry data is causing problems, rename the key and re-run the application
  - Most applications re-create user settings when run
  - In this way, the data won't be seen by the application
    - Can always rename the key back

1-61

Regmon Lab 2

1. Run Notepad
2. Change Font and point size
3. Enable Word wrap
4. Run Regmon & filter to Notepad.exe
5. Exit Notepad
6. In Regmon log, find location of user-specific Notepad settings
7. Double click on a line to jump to Regedit
8. Delete top level Notepad user settings key
9. Re-run Notepad and confirm font and word wrap reset to default setting

1-62

Example Problem

- Internet Explorer hung when started
  - Default internet connection was set, but wasn't being dialed
- Dialing the connection first manually and then running IE worked
- Background information:
  - User had previously installed the AT&T Dialer program, but had uninstalled it and created dial up connection manually

1-63

Solution

- Ran Regmon
- Looked backwards from end (at the point IE was hung)
  - Found references to ATT under a PhoneBook key
  - Renamed ATT key and problem went away
- Conclusion: registry junk was left from uninstall

1-64

Example Problem

- User somehow disabled all toolbars and menus in Word
  - No way to open files, change settings etc.
- With Regmon, captured startup of Word
- Found location of user-specific settings for Word
- Deleted this Registry key
- Re-ran Word – menus and toolbars were back!
  - Word re-created user settings from scratch

1-65

Filemon/Regmon as a Service

- Sometimes need to capture I/O or registry activity during the logon or logoff process
  - E.g. errors occuring during logon/logoff
- Solution:
  - Run Filemon/Regmon with AT command
  - Install and run Filemon/Regmon as a service
    - Use Srvany tool from Resource Kit
- In either case, but tools remain running after logoff

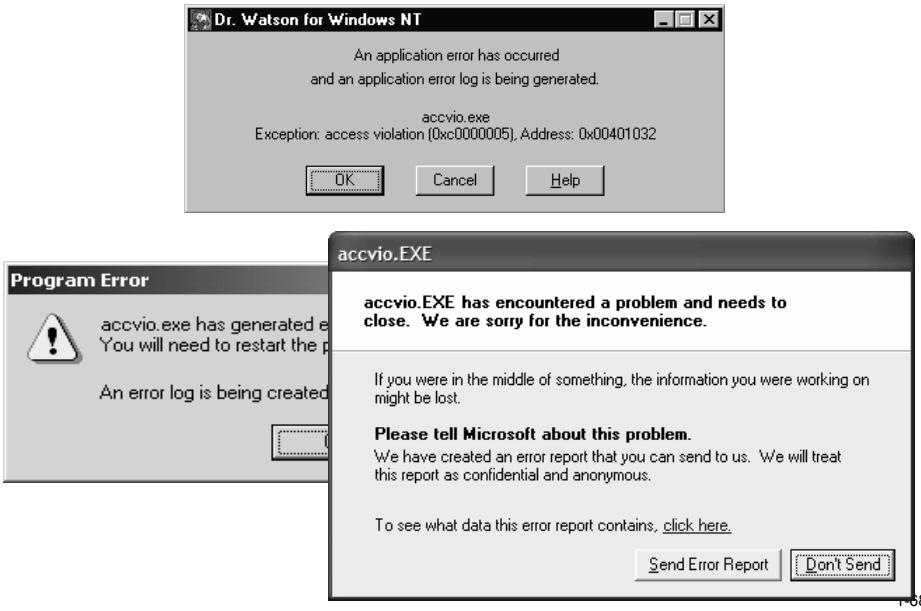
1-66

Analyzing Process Crashes

- If you still can't determine why a process is crashing, next step is to get a process dump to the developer
- But, until XP, few knew there was a process dump…

1-67

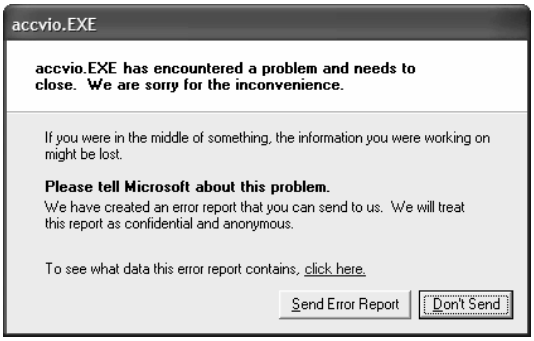
Process Crashes



1-68

Windows Error Reporting

- On XP & Server 2003, when an unhandled exception occurs:
  - System first runs DWWIN.EXE
    - DWWIN creates a process microdump and XML file and offers the option to send the error report
  - Then runs debugger (Drwtsn32.exe)



1-69

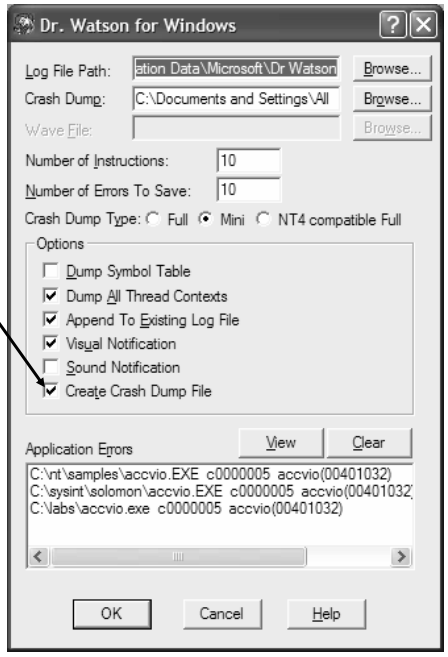
Windows Error Reporting

- Configurable with System Properties->Advanced->Error Reporting
  - HKLM\SOFTWARE\Microsoft\PCHealth\ErrorReporting
- Configurable with group policies
  - HKLM\SOFTWARE\Policies\Microsoft\PCHealth

1-70

Dr. Watson

- User message box doesn't mention most important thing:
  - A dump file was created!
- Can customize by running "DRWTSN32.EXE"
  - Note: servers default to no visual notification
- To set Dr. Watson as default debugger:
  - Drwtsn32 -i



1-71

Dumping a Running Processes

- Instead of killing a hung process (leaving no debug info), run Dr. Watson on it
  - Dr. Watson creates a crash dump file and then kills process
    - drwtsn32 –p processid
- Autodump (Debugging Tools) will snapshot a process without killing it
  - E.g. a server process that is having problems on a production system
    - Snapshot the process and debug offline
    - Determine if the process needs to be restarted or not

1-72

End of Troubleshooting Processes & Threads

Next: Troubleshooting Memory Problems

1-73

Windows Internals and Advanced Troubleshooting

Part 3: Troubleshooting Memory Problems

1-1

Troubleshooting Memory Problems

- System and process memory usage may degrade performance
  - Or eventually cause process failures
- How do you determine memory leaks?
  - Process vs. system?
- How do you know if you need more memory?
- How do you size your page file?
- What do system and process memory counters really mean?
  - Understanding process and system memory information can help answer these questions…

1-2

Windows Memory Management

- Demand paged virtual memory
  - Unit of protection and usage is one page
    - x86: 4 KB
    - Itanium 8 KB
  - Pages are read in on demand and written out when necessary (to make room for other memory needs)
- Provides illusion of flat virtual address space to each process
  - 32-bit: 4 GB, 64-bit: 16 Exabytes (theoretical)
- Supports up to 64 GB (32-bit systems) or 512 GB (64-bit systems) physical memory
- Intelligent, automatic sharing of memory

1-3

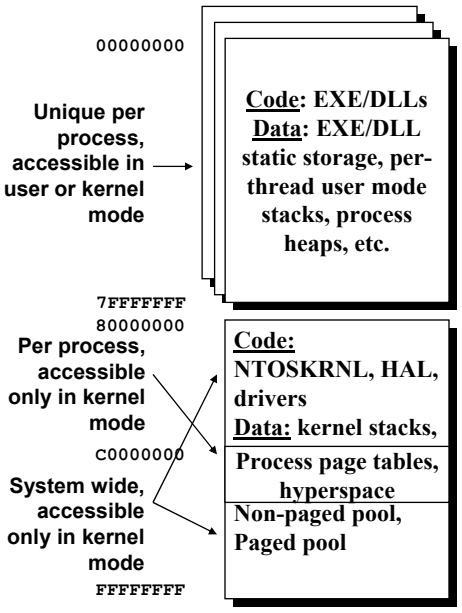
Process Memory Usage

1-4

Process Memory Usage

- Process virtual size
  - By default, 2 GB on 32-bit Windows
  - 64-bit Windows: 7152 GB
    - Up to 3 GB with Windows .NET Enterprise Server (/USERVA= or /3GB)
    - Application must be marked large address space aware
- What limits total process virtual memory?
  - Page file size + (most of) physical memory
  - Called “Commit limit”
- What limits physical size of a process?
  - Physical memory + Memory Manager policies
    - Based on memory demands and paging rates

1-5

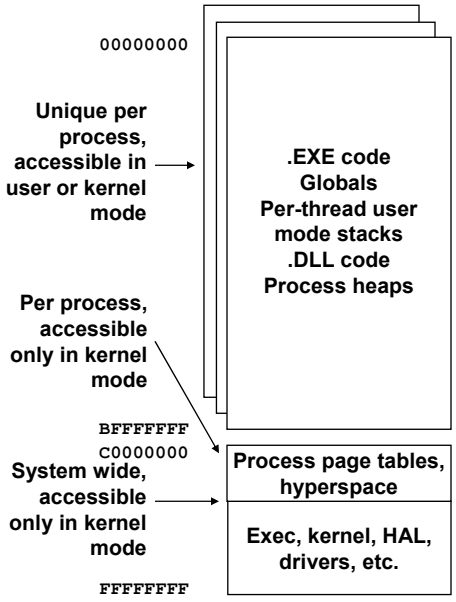


32-Bit Virtual Address Space (x86)

- 2 GB per-process
  - Address space of one process is not directly reachable from other processes
- 2 GB system-wide
  - The operating system is loaded here, and appears in every process’s address space
  - The operating system is not a process (though there are processes that do things for the OS, more or less in “background”)
- 3 GB user space and Address Windowing Extensions (AWE) t.b.d.

1-6

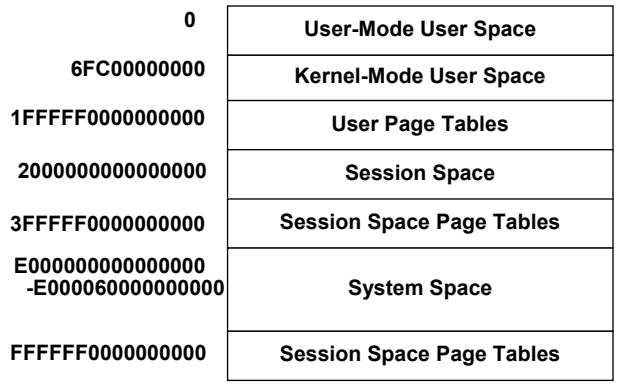
3GB Process Space Option



- /3GB option in BOOT.INI
  - Provides up to 3 GB per-process address space
  - Windows .NET Server supports variations from 2GB to 3GB (/USERVA=)
- Restrictions to use:
  - Only available on Windows 2000 Advanced Server & .NET Server Enterprise Edition
    - Limits memory to 16 GB
  - .EXE must have "large address space aware" flag in image header, or they're limited to 2 GB (specify at link time or with imagecfg.exe in Resource Kit)
  - Better solution: address windowing extensions

1-7

64-Bit Virtual Address Space (Itanium)

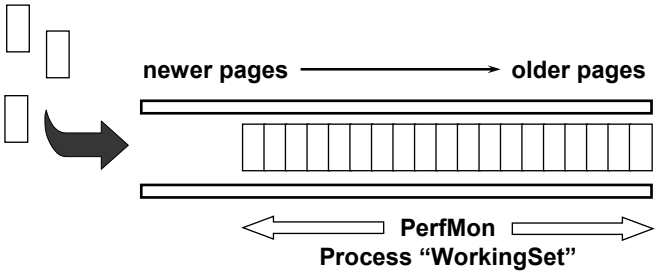


	64-bit Windows	32-bit Windows
User Address Space	7152 GB	2 or 3 GB
System PTE Space	128 GB	2 GB
System Cache	1 TB	960 MB
Paged pool	128 GB	650 MB
Non-paged pool	128 GB	256 MB

1-8

Process Memory Usage: "Working Set"

- Working set: All the physical pages "owned" by a process
  - Essentially, all the pages the process can reference without incurring a page fault
- A process always starts with an empty working set
  - Pages itself into existence
    - XP prefetches pages to speed up application startup
  - Many page faults may be resolved from memory



1-9

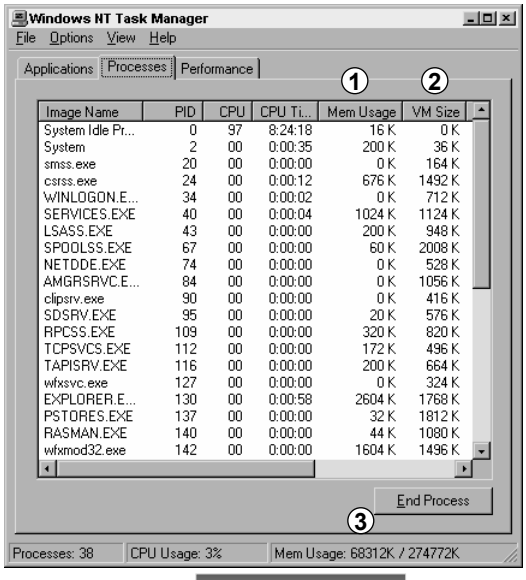
Process Memory Information

Task Manager Processes tab

① **"Mem Usage" = physical memory used by process (working set size, not working set limit)**
- **Note: Shared pages are counted in each process**

② **"VM Size" = private (not shared) committed virtual space in processes == potential pagefile usage**

③ **"Mem Usage" in status bar is <u>not</u> total of "Mem Usage" column (see later slide)**

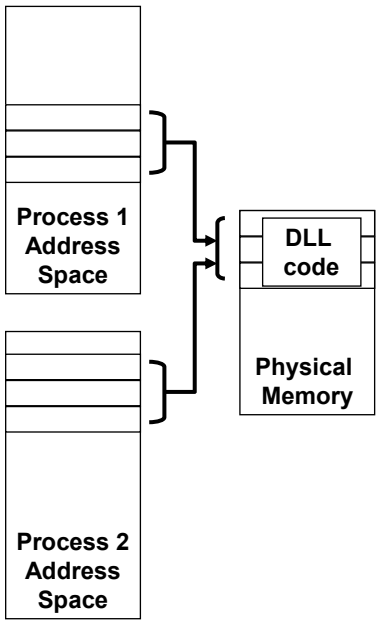


Screen snapshot from:
Task Manager | Processes tab

1-10

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 or DLL



1-11

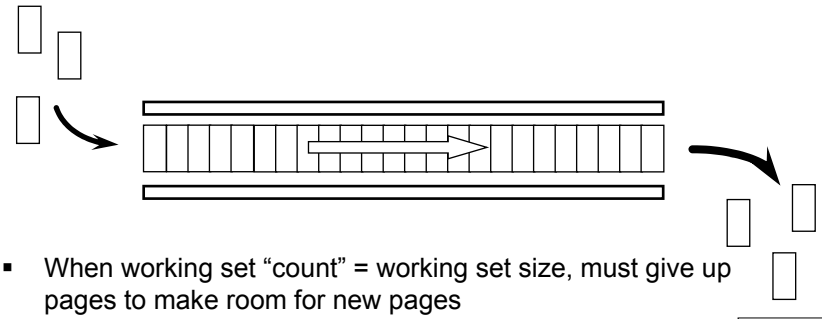
Viewing the Working Set

- Working set size counts shared pages in each working set
- Vadump (Resource Kit) can dump the breakdown of private, shareable, and shared pages

```
C:\> Vadump -o -p 3968
Module Working Set Contributions in pages
    Total   Private Shareable    Shared Module
       14         3        11         0 NOTEPAD.EXE
       46         3         0        43 ntdll.dll
       36         1         0        35 kernel32.dll
        7         2         0         5 comdlg32.dll
       17         2         0        15 SHLWAPI.dll
       44         4         0        40 msvcrt.dll
```

1-12

Working Set Replacement



- When working set “count” = working set size, must give up pages to make room for new pages
- This is called a local page replacement policy (versus a global replacement policy common on Unix)
  - Means that a single process cannot take over all of physical memory unless other processes aren’t using it
- Page replacement algorithm is least recently accessed
  - Windows 2000: only on uniprocessor; Windows XP and .NET Server: All systems

Kept in memory on standby or modified page list

1-13

Paging Lists

1-14

Managing Physical Memory

- System keeps unowned 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

1-15

Standby And Modified Page Lists

- Modified pages go to modified (dirty) list
  - Avoids writing pages back to disk too soon
- Unmodified pages go to standby (clean) list
- They form a system-wide cache of “pages likely to be needed again”
  - Pages can be faulted back into a process from the standby and modified page list
  - These are counted as page faults, but not page reads

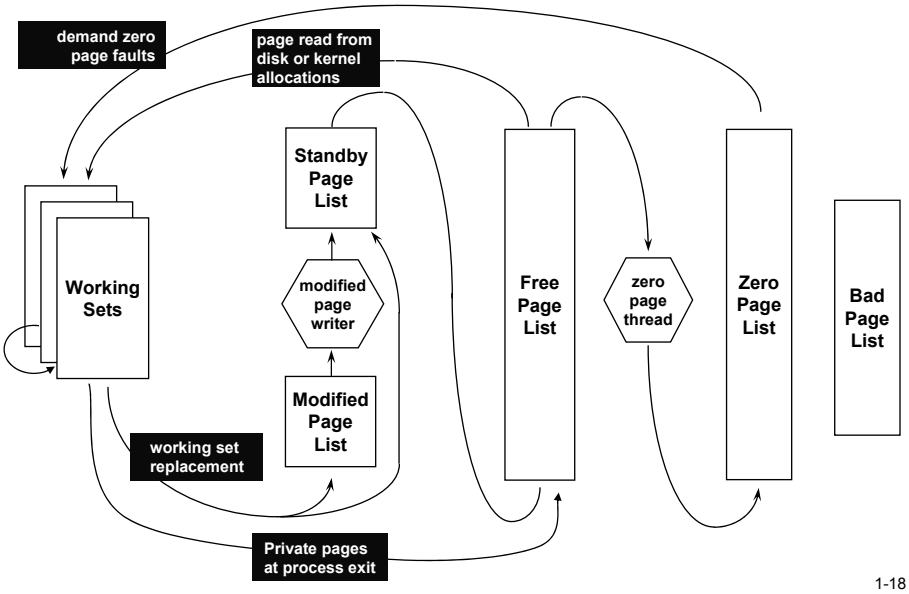
1-16

Free And Zero Page Lists

- Free Page List
  - Used for page reads
  - Private modified pages go here on process exit
  - Pages contain junk in them (e.g., not zeroed)
  - On most busy systems, this is empty
- Zero Page List
  - Used to satisfy demand zero page faults
    - References to private pages that have not been created yet
  - When free page list has 8 or more pages, a priority zero thread is awoken to zero them
  - On most busy systems, this is empty too

1-17

Paging Dynamics



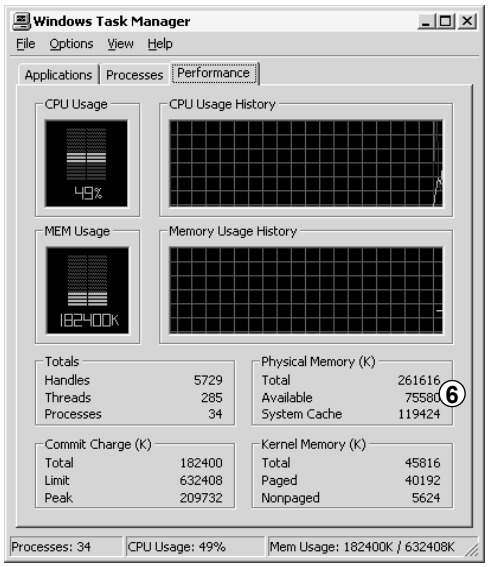
1-18

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)



Screen snapshot from:
Task Manager | Performance tab

Viewing the Paging Lists

- Only way to get actual size of physical memory lists is to use !memusage in Kernel Debugger

```
lkd> !memusage
 loading PFN database

            Zeroed:      0 (     0 kb)
              Free:      3 (    12 kb)
           Standby:  98248 (392992 kb)
          Modified:    563 (  2252 kb)
   ModifiedNoWrite:      0 (     0 kb)
      Active/Valid:  93437 (373748 kb)
        Transition:      1 (     4 kb)
           Unknown:      0 (     0 kb)
             TOTAL: 192252 (769008 kb)
```

Screen snapshot from:kernel debugger
!memusage command

Page Files

Page Files

- What gets sent to the paging file?
  - Not code – only modified data (code can be re-read from image file anytime)
- When do pages get paged out?
  - Only when necessary
  - Page file space is only reserved at the time pages are written out
  - Once a page is written to the paging file, the space is occupied until the memory is deleted (e.g., at process exit), even if the page is read back from disk
- Can run with no paging file
  - Windows NT4/Windows 2000: Zero pagefile size actually created a 20MB temporary page file

Do I Need More Memory?

- If heavy paging activity:
  - Monitor Memory->Page Reads/sec
    - Not Page Faults/sec (which includes soft faults)
  - Should not stay high for sustained period
  - Some hard page faults unavoidable
    - Process startup
    - Normal file I/O done via paging
  - To eliminate normal file I/O, subtract System->File Read Operations/sec
    - Or, use Filemon to determine what file(s) are having paging I/O (asterisk next to I/O function)

1-23

Sizing The Page File

- Given understanding of page file usage, how big should the total paging file space be?

  (Windows supports multiple paging files)
- Size should depend on total private virtual memory used by applications and drivers
  - Therefore, not related to RAM size (except for taking a full memory dump)

1-24

Sizing The Page File

- Worst case: Windows has to page all private data out to make room for code pages
  - To handle, minimum size should be the maximum of VM usage ("Commit Charge Peak")
    - Hard disk space is cheap, so why not double this
  - Normally, make maximum size same as minimum
  - But, max size could be much larger if there will be infrequent demands for large amounts of page file space
    - Performance problem: Page file extension will likely be very fragmented
    - Extension is deleted on reboot, thus returning to a contiguous page file

Memory Management Information

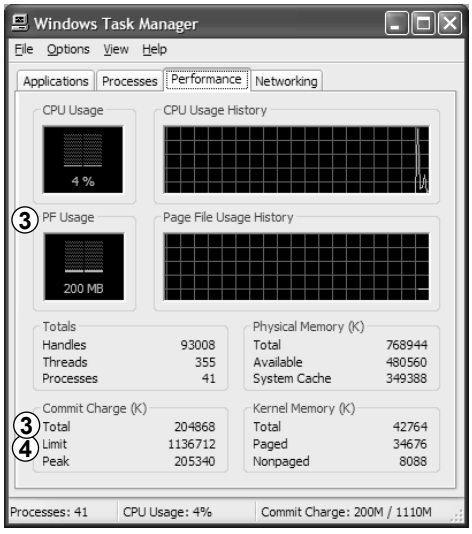
Task Manager Performance tab

③ Total committed private virtual memory (total of "VM Size" in process tab + Kernel Memory Paged)

- not all of this space has actually been used in the paging files; it is "how much would be used if it was all paged out"

④ "Commit charge limit" = sum of physical memory available for processes + current total size of paging file(s)

- does not reflect true maximum page file sizes (expansion)
- when "total" reaches "limit", further VirtualAlloc attempts by any process will fail



Screen snapshot from:
Task Manager | Performance tab

Why Page File Usage on Systems with Ample Free Memory?

- Because memory manager doesn't let process working sets grow arbitrarily
  - Processes are not allowed to expand to fill available memory (previously described)
    - Bias is to keep free pages for new or expanding processes
  - This will cause page file usage early in the system life even with ample memory free
- We talked about the standby list, but there is another list of modified pages recently removed from working sets
  - Modified private pages are held in memory in case the process asks for it back
  - When the list of modified pages reaches a certain threshold, the memory manager writes them to the paging file (or mapped file)
  - Pages are moved to the standby list, since they are still "valid" and could be requested again

Memory Leaks

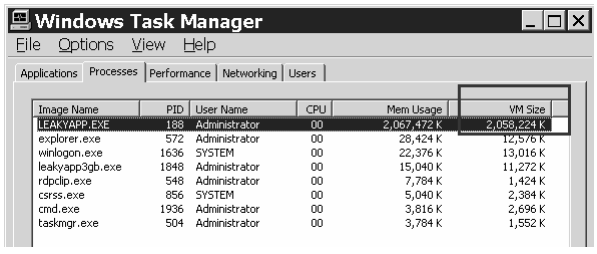
Process Memory Leaks

- System says “running low on virtual memory”
  - Before increasing size of page file, look for a process (or system) memory leak
- Look for who is consuming pagefile space
  - Process memory leak: Check Task Manager, Processes tab, VM Size column
    - Or Perfmon “private bytes”, same counter

1-29

Leakyapp Test Program

- Leakyapp.exe is in the Resource Kit
- Continuously allocates private, nonshareable virtual memory
  - When there is no more, it just keeps trying..
- Run several copies to fill pagefile more quickly



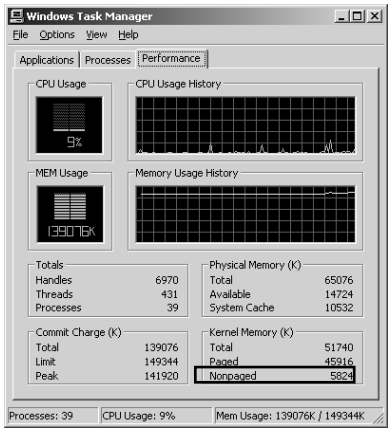
1-30

Handle Leaks

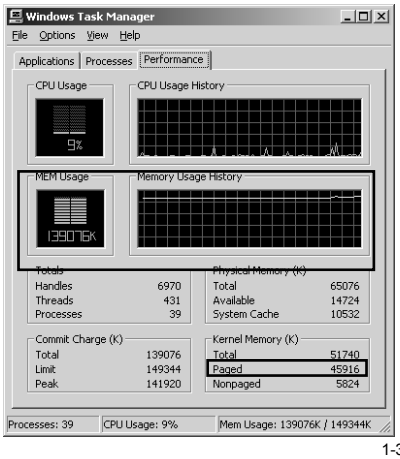
- Processes that open resources but don't close them can exhaust system memory
  - Check total handle count in Task Manager Performance tab
  - To find offending process, on Process tab add Handle Count and sort by that column
  - Using Process Explorer handle view with differences highlighting you can even find which handle(s) are not being closed

Kernel Memory Leaks

- A driver leaking nonpaged pool shows up as large and growing Nonpaged pool usage



- Or, a rowing Memory Usage and Paged pool usage



Kernel Memory Pools

- Two system memory pools
  - "Nonpaged Pool" and "Paged Pool"
  - Used for systemwide persistent data (visible from any process context)
- Pool sizes are a function of memory size & Server vs. Workstation
  - Can be overidden in Registry:

    HKLM\System\CurrentControlSet\Control\Session Manager
    \Memory Management

1-33

Kernel Memory Pools

- Nonpaged pool
  - Has initial size and upper limit (can be grown dynamically, up to the max)
  - 32-bit upper limit: 256 MB on x86 (NT4: 128MB)
    - 64-bit limit: 128 GB
- Paged pool
  - 32-bit upper limit: 650MB (Windows Server 2003), 470MB (Windows 2000), 192MB (Windows NT 4.0)
    - 64-bit limit: 128 GB
- Pool size performance counters display current size, not maximum
  - To display maximums, use "!vm" kernel debugger command

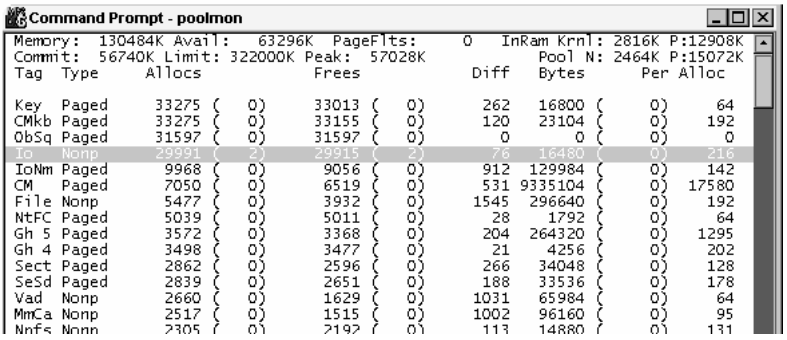
1-34

Debugging Pool Leaks

- Two options:
  - Poolmon
    - In the Support Tools and the Device Driver Kit (DDK)
    - Requires that you turn on Pool Tagging with Gflags on Windows NT and Windows 2000
  - Driver Verifier
    - Select all drivers
    - Turn on pool tracking

Troubleshooting with Poolmon

- Poolmon.exe (Support Tools)
  - Shows paged and nonpaged pool consumption by data structure “tag”
  - Must first turn on “pool tagging” with Resource Kit gflags tool & reboot
    - On by default in Windows Server 2003



Troubleshooting with Poolmon

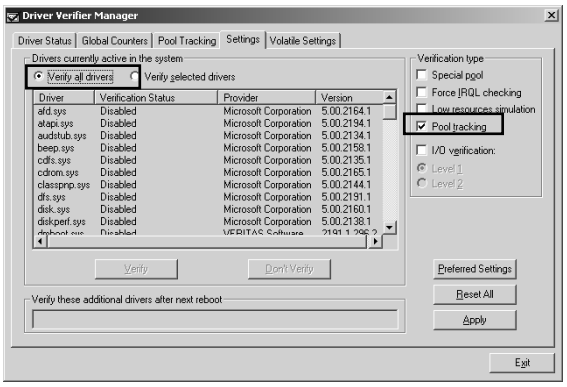
- Once you find pool tag that is leaking:
  - Look up in Windows Debugging Tools subfolder \triage\pooltag.txt
- May not be there if 3rd party driver
  - Run Strings (from Sysinternals) on all drivers:

```
strings \windows\system32\drivers\*.sys
  | findstr Xyzz
```

1-37

Troubleshooting with Driver Verifier

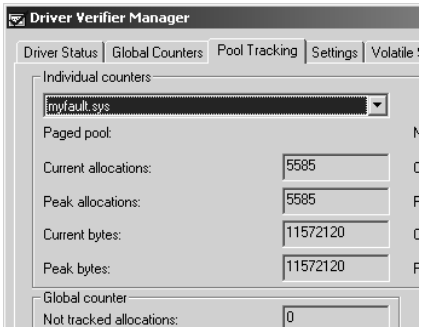
- Use Driver Verifier to enable pool tracking for all drivers (or ones of interest)
  - System tracks pool usage by driver
    - Poolmon looks at pool usage by structure tag



1-38

Looking for Leaks

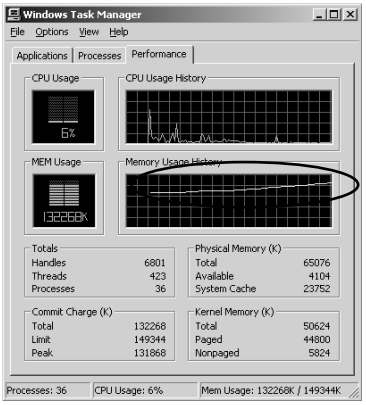
- Reboot and look at the pool usage of each driver
- A leaker exhibits the following
  - Current allocations is always close to or equal to the peak
  - The peak grows over time
  - If the leak is significant the peak allocations or bytes will be large



1-39

Causing a Pool Leak

- Run NotMyFault and select “Leak Pool”
  - Allocates paged pool buffers and doesn’t free them
  - Stops leaking when you select “Stop Leaking”



1-40

End of Troubleshooting Memory Problems

Next: Crash Dump Analysis

1-41

Windows Internals and Advanced Troubleshooting

Part 4: Crash Dump Analysis

1-1

Outline

- What causes crashes?
- Crash dump options
- Analysis with WinDbg/Kd
- Debugging hung systems
- Microsoft On-line Crash Analysis
- Using Driver Verifier
- Live kernel debugging
- Getting past a crash

1-2

Why Analyze Dumps?

- The debuggers and Microsoft Online Crash Analysis (OCA) often solve crashes
- Sometimes, however, they do not, so your analysis might tell you:
  - What driver to disable, update, or replace with different hardware
  - What OEM to send the dump to

1-3

You Can Do It!

- Many systems administrators ignore Windows NT/Windows 2000's crash dump options
  - "I don't know what to do with one"
  - "Its too hard"
  - "It won't tell me anything anyway"
- Basic crash dump analysis is actually pretty straightforward
  - Even if only 1 out of 5 or 10 dumps tells you what's wrong, isn't it worth spending a few minutes?

1-4

What Causes Crashes?

- System crashes when a fatal error prevents further execution
  - Any kernel-mode component can crash the system
- Drivers and the OS share the same memory space
  - Therefore, any driver or OS component can, due to a bug, corrupt system memory
  - Note: This is for performance reasons and is the same on Linux, most Unix's, VMS, etc…

1-5

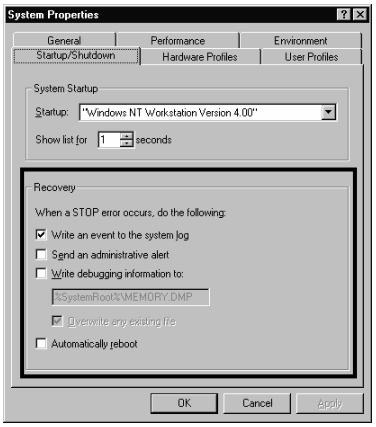
Dump Options

- Complete memory dump (Windows NT 4, Windows 2000, Windows XP)
  - Full contents of memory written to <systemroot>\memory.dmp
- Kernel memory dump (Windows 2000, Windows XP)
  - System memory written to <systemroot>\memory.dmp
- Small memory dump (Windows 2000, Windows XP)
  - Also called a minidump or triage dump
  - 64KB of summary written to <systemroot>\minidump\MiniMMDDYY-NN.dmp

1-7

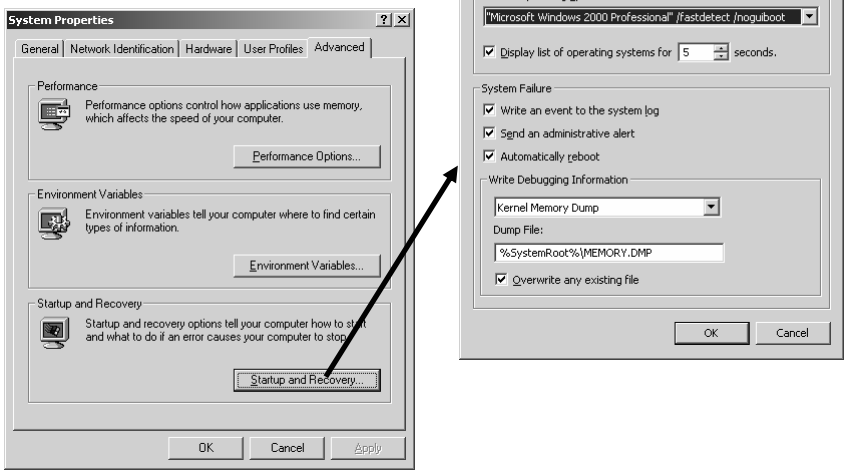
Enabling Dumps

- In Windows NT 4:



Enabling Dumps

- In Windows 2000/XP:



At The Crash

- A component calls KeBugCheckEx, which takes five arguments:
  - Stop code
  - 4 stop-code defined parameters
- KeBugCheckEx:
  - Turns off interrupts
  - Tells other CPUs to stop
  - Paints the blue screen
  - Notifies registered drivers of the crash
  - If a dump is configured:
    - Verifies checksums
    - Calls dump I/O functions

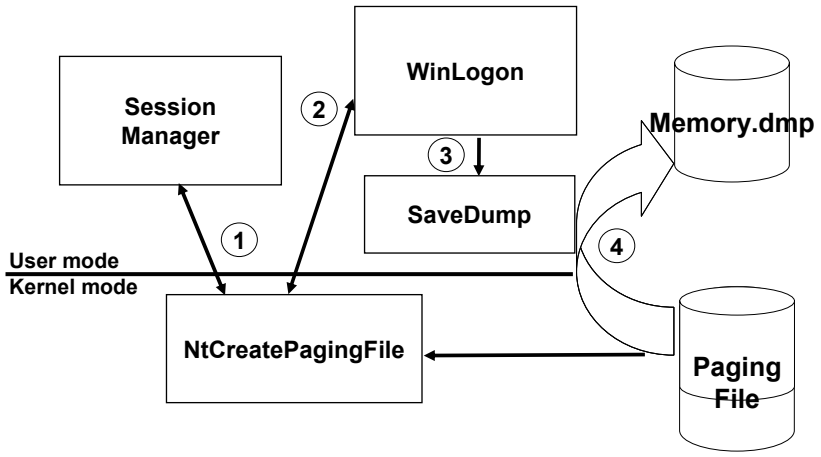
1-10

Common Stop Codes

- There are about 150 defined stop codes
- IRQL_NOT_LESS_OR_EQUAL (0x0A)
  - Usually an invalid memory access
- INVALID_KERNEL_MODE_TRAP (0x7F) and KMODE_EXCEPTION_NOT_HANDLED (0x1E)
  - Generated by executing garbage instructions
  - Its usually caused when a stack is trashed

1-11

At The Reboot



1-12

At The Reboot

- Session Manager process (\winnt\system32\smss.exe) initializes paging file ①
  - NtCreatePagingFile
- NtCreatePagingFile determines if the dump has a crash header ②
  - Protects the dump from use
- WinLogon calls NtQuerySystemInformation to tell if there's a dump

1-13

At The Reboot

- If there's a dump, Winlogon executes SaveDump ③ (\winnt\system32\savedump.exe)
  - Writes an event to the System event log
  - SaveDump writes contents to appropriate file ④
- Crash dump portion of paging file is in use during copy, so virtual memory can run low

1-14

Why Crash Dumps Fail

- Most common reasons:
  - Paging file on boot volume is too small
  - Not enough free space for extracted dump
- Less common:
  - The crash corrupted components involved in the dump process
  - Miniport driver doesn't implement dump I/O functions
    - Windows 2000 and Windows XP storage drivers must implement dump I/O to get a Microsoft® signature

1-15

Generating A Test Dump

- Get BSOD from Sysinternals:
  www.sysinternals.com/ntw2k/freeware/bluesave.shtml
- It crashes the system by:
  - Allocating kernel memory
  - Freeing the memory
  - Raising the IRQL
  - Touching the freed memory

1-16

Analyzing a Crash Dump

- There are two kernel-level debuggers:
  - WinDbg –Windows program
  - Kd – command-line program
  - Same functionality

1-17

Debugging Tools

- Get the latest from: www.microsoft.com/ddk/debugging
  - Supports Windows NT 4, Windows 2000, Windows XP, Server 2003
  - Check for updates frequently
  - *Don't* use older version on install media
- Install to c:\Debuggers
  - Easy access from command prompt

1-18

Symbol Files

- Before you can use any crash analysis tool you need symbol files
  - Symbol files contain global function and variable names
  - At the minimum, get the symbol file(s) for ntoskrnl.exe, ntkrnlmp.exe, ntkrnlpa.exe, ntkrpamp.exe
- Symbols are service pack-specific and have an installer (default directory is \winnt\symbols)
  - Windows NT 4: *.dbg
  - Windows 2000: *.dbg, *.pdb
  - Windows XP: *.pdb
  - Note: SP symbols only include updates

1-19

Microsoft Symbol Server

- WinDbg and Kd can download symbols automatically from Microsoft
- Pick a directory to install symbols and add the following to the debugger's symbol path:

  SRV**directory**http://msdl.microsoft.com/download/symbols
- The debugger automatically detects the OS version of a dump and downloads the symbols on-demand

1-20

Installing the Symbol Files

- On CDs:
  - Windows NT 4: on Windows NT 4 Setup CD under \support\debug
  - Windows 2000 SP0/Windows XP SP0 on Customer Support Diagnostics CD
  - Windows 2000 SP1 on SP1 CD
- Online:
  - Windows NT 4: All (US) service packs are at ftp.microsoft.com:\bussys\winnt\winnt-public\fixes\usa\nt40
  - Windows 2000/XP: http://www.microsoft.com/ddk/debugging/symbols.asp

1-21

Automated Analysis

- When you open a crash dump with Windbg or Kd you get a basic crash analysis:
  - Stop code and parameters
  - A guess at offending driver
- The analysis is the result of the automated execution of the !analyze debugger command

Debugger Commands

- Two types of commands
  - *Dot* commands are built-in
  - *Bang* commands are provided with extension DLLs
- Extension DLLs allow Microsoft and third-parties to dynamically add commands
  - The main extension DLL is the kernel-debugger extension DLL, kdexts.dll
  - Each OS has a subdirectory with its own kdexts.dll version as well as other, development-area specific, extension DLLs (e.g. Rpcexts.dll, ndiskd.dll, …)

Deeper Analysis

- Always execute !analyze with the –v option to get more information
  - Text description of stop code
  - Meaning (if any) of parameters
  - Stack dump
- !Analyze uses heuristics to walk up the stack and determine what driver is the likely cause of the crash
  - “Followup” is taken from optional triage.ini file

1-24

Useful Commands

- When you load a dump into the debugger it executes !analyze
  - Sometimes identifies the cause of a crash
  - Always execute !analyze –v to see more
- The next steps:

Look at the current process:	!process
List all processes:	!process 0 0
Look at a thread:	!thread <thread address or ID>
List loaded drivers:	lm kv
Look at an I/O request packet:	!irp <irp address>
Disassemble code:	u <address or function name>

1-25

Hung Systems

- You can tackle a hung system, but only if you've prepared:
  - Boot in debug mode, or
  - Set the keystroke-crash Registry value
- For debug mode you need a second system (the debugger host) connected to the target via serial cable
  - Run Windbg/Kd on the host
  - Edit the target's boot.ini file:
    - /debugport=comX /baudrate=XXX
  - When the system hangs, connect with the debugger and hit Ctrl-C

1-26

Hung Systems

- To configure keystroke-crash:
  - Set HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\i8042prt\Parameters\CrashOnCtrlScrl to 1
  - Enter right-ctrl+[scroll-lock, scroll-lock] to crash the system
- Use !thread to see what's running

1-27

Microsoft On-line Crash Analysis (OCA)

- Have Microsoft process dumps at oca.microsoft.com
  - XP asks you if you want to submit after a crash
  - You can visit OCA and manually submit a dump
- OCA accepts Win2K and XP dumps, but is focused on XP
- Currently requires a Passport account to check crash analysis status if it doesn't know right away

1-28

What Does OCA Do?

- Server farm uses !analyze, but uses Microsoft's Triage.ini file and database that includes information about known problems
- Several ways to get OCA results:
  - Via e-mail
  - At the OCA site
- Sometimes OCA will point you at KB articles that describe the problem
  - KB articles may tell you to use Windows Update to get newer drivers, a hotfix, or install a Service Pack

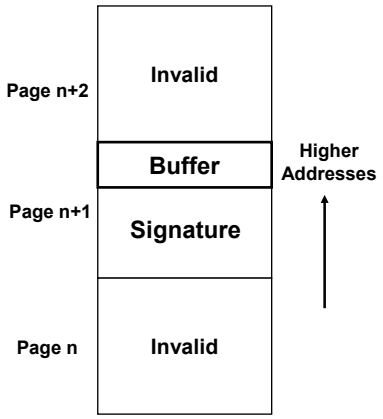
1-29

Driver Verifier

- This tool was introduced in Windows 2000 and can be useful to validate a suspicion about a driver
- The Verifier performs the following checks:
  - IRQL rule adherence
  - I/O request consistency
  - Proper memory usage

Special Pool

- Special pool is a kernel buffer area where buffers are sandwiched with invalid pages
- Conditions for a driver allocating from special pool:
  - Driver Verifier is verifying driver
  - Special pool is enabled
  - Allocation is slightly less than one page (4 KB on x86)



Driver Verifier

- If the Verifier detects a violation it crashes the system and identifies the driver
- If you find a driver in a crash dump that looks like it might be the cause of the crash, turn on verification for it
  - Use “Last Known Good” if the verifier detects a bug during the boot
  - If a bug is detected in a third-party product check for updates and/or contact the vendor’s support
- Note that the Verifier means fewer crashes on Windows XP than Windows 2000 than Windows NT 4

1-32

Getting Past a Crash

- Last-Known Good
  - Boots with driver/kernel configuration last used during a successful boot
- Safe Mode
  - Boots the system with core set of drivers and services
  - Network and non-network
- The Recovery Console
  - Manually disable offending service, replace corrupt images, update files
- ERD Commander 2002
  - Registry Editor, Explorer, Driver/Service Manager, password changer, Event Log viewer, Notepad

1-33

The Bluescreen Screen Saver

- Scare your enemies and fool your friends with the Sysinternals Bluescreen Screen Saver
  - Be careful, your job may be on the line!

1-34

More Information

- Inside Windows 2000, 3rd edition – section on System Crashes in chapter 4
- Debugging Tools help file
- Knowledge Base Articles
  - http://www.microsoft.com/ddk/debugging
- Other books:
  - http://www.microsoft.com/ddk/newbooks.asp
- The debugger team wants your feedback and bug reports

1-35

End of Tutorial

Thank you for coming!

1-36