Windows Internals & Advanced Troubleshooting

David Solomon

David Solomon Expert Seminars, Inc.

http://www.solsem.com

Amsterdam – March 23, 2005

About the Speaker

- 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, …)
- Books:
  - *Windows Internals, 4th edition*
  - *Inside Windows 2000, 3rd edition*
  - *Inside Windows NT, 2nd edition*
  - *Windows NT for OpenVMS Professionals*
- Video Training:
  - Interactive Windows 2000/XP/2003 interna tutorial
  - Used by Microsoft for their internal training
- Live Classes:
  - 2-5 day classes on Windows Internals, Advanced Troubleshooting

1-2

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. Process & Thread Troubleshooting
2. Understanding & Troubleshooting Memory Problems
3. Troubleshooting with Filemon & Regmon
4. Crash Dump Analysis
5. Boot & Startup Troubleshooting

1-4

Part 1: Process & Thread Troubleshooting

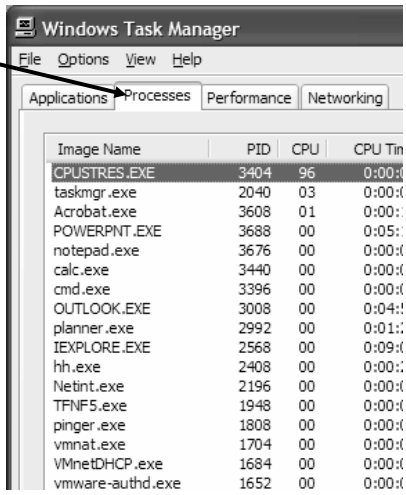
- Introduction
- Investigating Processes & Threads
- Accounting for CPU Usage
- Process Handle Table
- DLLs and Memory Mapped Files
- Investigating Unknown Processes

Analyzing CPU Activity

- Scenario: your system is slow
  - What is running and why?
- What do you do?
  - Most bring up Task Manager, which by default lists the “running applications”
  - This doesn’t tell you anything

Which Process Is Consuming CPU?

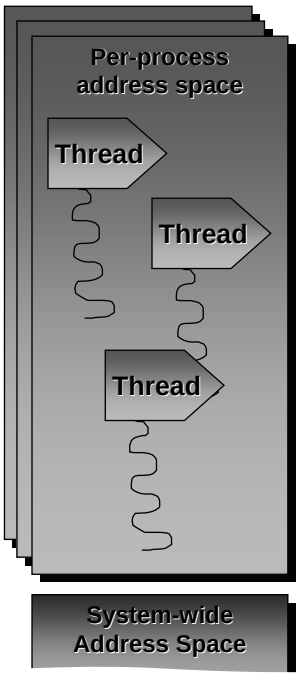
- The Processes tab does indicate what is using the CPU (sort of)
  - Note that CPU Time column is mostly 0 (explained later)
- To determine which process is consuming the most CPU time, sort processes by “CPU” usage column



1-7

Processes And Threads

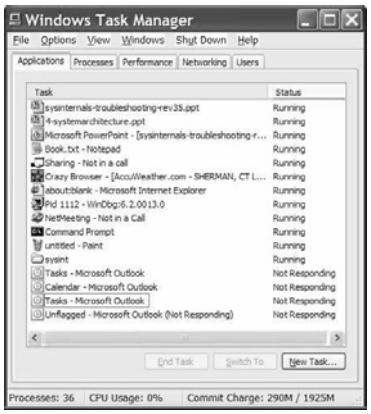
- What is a process?
  - Represents an instance of a running program
    - You create a process to run a program
    - Starting an application creates a process
  - Process defined by
    - Address space
    - Resources (e.g., open handles)
    - Security profile (token)
- What is a thread?
  - An execution 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



1-8

What Are Task Manager's "Applications"?

- A meaningless term at the OS level
  - Not a list of processes
  - Not a list of "tasks" (another meaningless term)
  - It's a list of top level visible windows in your session that meet certain criteria
- What does the status column mean?
  - Running:
    - Windows don't run—threads do
    - Running displayed only when owning thread is waiting for a window message (e.g. not running!)
  - Not Responding: not waiting for window messages
- To map a window to a process, right-click on a window and select "Go to 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
  - Sometimes is a "multi-purpose" process like Svchost.exe or Inetinfo.exe or Dllhost.exe or System
- We need more information!

1-10

Process Explorer

- "Super Task Manager"
  - Illuminates many details about processes and threads that are otherwise hard or impossible to obtain

1-11

How Process Explorer Works

- Uses undocumented functions for:
  - Enumerating loaded modules with full path names
  - Enumerating processes and handles
- Obtains handle names using the aid of a driver
- Runs on Win95, 98, ME, NT4, 2000, XP, 2003
- Related Tools:
  - Handle – command-line handle viewer
  - Listdlls – command-line DLL viewer

1-12

Agenda

- Introduction
- Investigating Processes & Threads
- Accounting for CPU Usage
- Process Handle Table
- DLLs and Memory Mapped Files
- Investigating Unknown Processes

1-13

Lab: The Process List

1. Run Process Explorer & maximize window
2. Run Task Manager – click on Processes tab
3. Arrange windows so you can see both
4. Notice process tree vs flat list in Task Manager
   - If parent has exited, process is left justified
5. Sort on first column (“Process”) and note tree view disappears
6. Click on View->Show Process Tree (or CTRL+T) to bring it back
7. Notice description and company name columns
8. Hover mouse over image to see full path of image

1-14

Lab: Refresh Highlighting

1. Change update speed to paused by pressing space bar
2. Run Notepad
3. In ProcExp, hit F5 and notice new process
4. Exit Notepad
5. In ProcExp, hit F5 and notice Notepad in red

- Uses
  - Understanding process startup sequences
  - Detecting appearance of processes coming and going

1-15

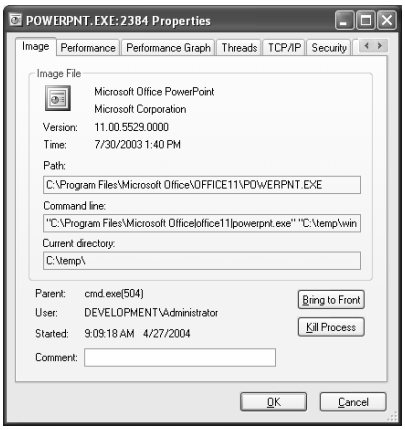
Lab: Column Selection And Username

1. Click on View->Select Columns
   - Add username column
2. Compare username column in Task Manager with Process Explorer – what is the difference?
3. Deselect View->Show Processes From All Users

1-16

Lab: Image Information

- Open Process Properties
- Items of note on Image tab
  - Description, company name, version (from .EXE)
  - Image verification status
  - Full image path
  - Command line used to start process
  - Current directory
  - Parent process
  - User name
  - Start time
  - Bring to Front
    - Useful if multiple windows with identical titles



1-17

Lab: The Command Line

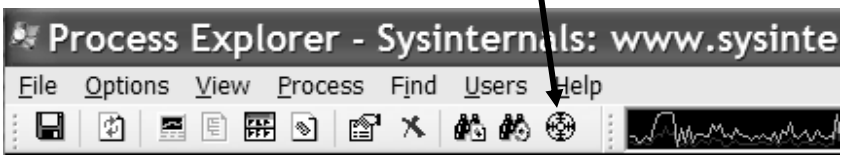
1. Double click on date/time in task bar (lower right of screen)
2. In Process Explorer, hit F5 to refresh
3. Find new process created (RUNDLL32.EXE)
4. Examine command line arguments

- Example: cmd.exe process was consuming lots of CPU time
  - Command line argument showed which .BAT file was running

1-18

Associating Windows with Processes

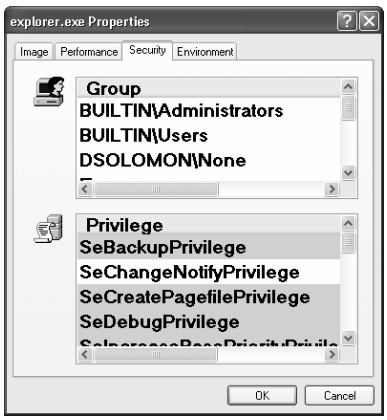
- Task Manager can associate a window in its list with a process
  - But sometimes windows appear that are not in its "Applications" list
- Process Explorer has a "window finder" tool
  - On tool bar, drag window finder icon over window and release
  - Process that owns thread that owns window is highlighted
- Visual Studio Spy++ tool shows which thread owns a window…



1-19

Security

- Click on Security tab of process properties
- Shows rest of access token (username is on image tab)
  - Groups list
    - Includes OS-assigned groups
  - Privileges (user rights)
    - Disabled by default
    - Programs turn these on when needed
- This is really a "Resultant Set of Groups" and "Resultant Set of Privileges" page



1-23

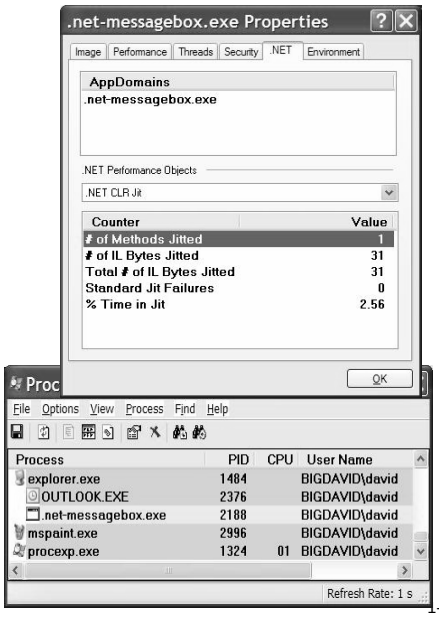
Lab: Privileges

1. Bring up process properties for Explorer.exe
2. Go to Security tab and notice that SeSystemTime privilege is disabled
3. Double-click on Date/Time in the tray
4. Bring up the process properties for the new Rundll32.exe process
5. Click on the Security tab
6. Notice SeSystemTime privilege is enabled

1-24

.NET Processes

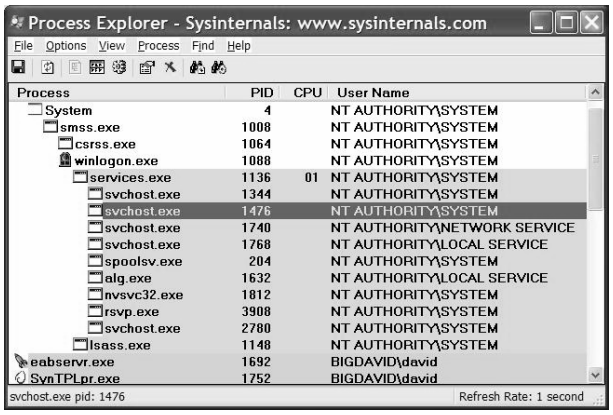
- Process Explorer is aware of .NET processes
  - Can highlight with Options->Highlight .NET Processes
- Process properties have .NET tab
  - Shows details about .NET process (CLR, Appdomains)
- Can also add .NET-specific columns to process list



1-28

Service Information

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



Examining Service Processes

- Services tab on process properties lists the active services inside
- Displays the internal name, the display name, and description (if available)
  - Tlist /s (Debugging Tools) & Tasklist /svc (new as of XP) only show internal name

Lab: Services

1. Open a command prompt
2. Type “tasklist /svc”
3. Find the Svchost.exe process with the most services inside it
4. In Process Explorer, double click on that Svchost.exe process
5. Click on Services tab
6. Notice extra details about each service displayed by Process Explorer

1-31

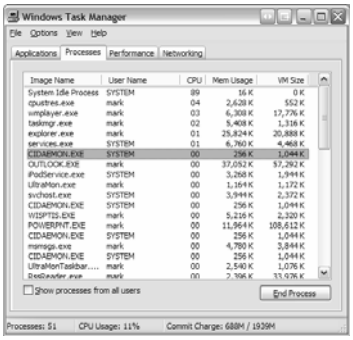
Agenda

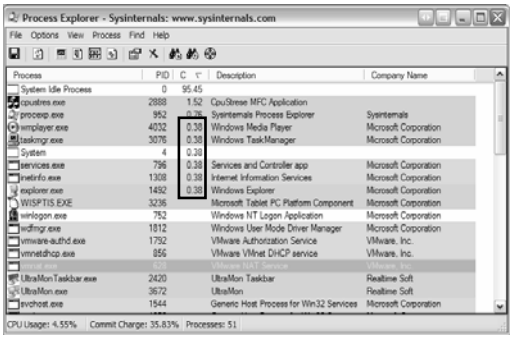
- Introduction
- Investigating Processes & Threads
- Accounting for CPU Usage
- Process Handle Table
- DLLs and Memory Mapped Files
- Investigating Unknown Processes

1-33

Examining CPU Time

- Looking at total CPU time for each process may not reveal where system has spent its time
  - Task Manager rounds CPU
  - In Process Explorer, click on Options->Show Fractional CPU time





Time Accounting

- CPU time accounting is driven by programmable interrupt timer
  - Normally 10 msec
  - Run Clockres (Sysinternals tool) to check
- 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 run but never get charged
- Process Explorer can show context switch activity

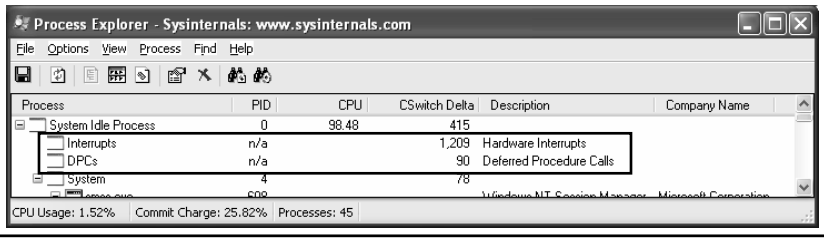
Context Switch Delta

- Whenever a thread begins running, its “context switch” counter increments
  - The number of context switches doesn’t reflect length of execution, just number of times it was scheduled
- Process Explorer has a context switch delta column
  - Sum of thread context switches in last refresh interval
  - Notice all of the thread activity in processes that show 0% CPU

1-36

Interrupt Time Accounting

- Interrupt activity is not charged to any thread or process
  - Two types: interrupts and DPCs
  - Task Manager includes interrupt and DPC time with the Idle process time
- Process Explorer shows these as separate processes (not really processes)
  - Context switches for these are really number of interrupts and DPCs



1-37

Identifying ISR and DPC's by Driver

- Since time spent at DPC level and above is not accounted by driver type, one way to determine where time has been spent in kernel mode is by using a *profiling/sampling* tool
- Kernrate is a such a tool
  - Free download from http://www.microsoft.com/whdc/system/sysperf/krview.mspx
  - Can be used both for kernel time and user mode processes
  - Can show where time is being spent down to the function level
  - May miss short lived events or events close to the sampling interval
- XP SP2 and Server 2003 SP1 and later support tracing ISRs and DPCs
  - Documented in Windows Internals

1-38

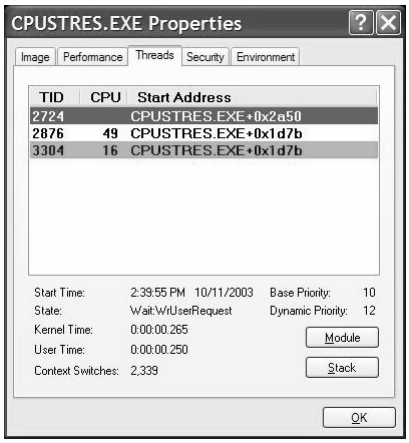
Understanding Svchost.exe CPU Time Consumption

- If a Svchost or other multi-component process such Inetinfo.exe (IIS) or Dllhost.exe (COM) is consuming CPU time, how do you determine which service is responsible?
  - Need to drill down to thread granularity
- Two approaches:
  - Invasive: attach with a debugger to examine threads
    - Threads are suspended while looking
  - Noninvasive: view threads with Process Explorer
    - Threads are not affected while looking

1-39

Viewing Threads with Process Explorer

- Process Explorer "Threads" tab shows which thread(s) are running
  - Start address represents where the thread began running (not where it is now)
  - Click Module to get details on module containing thread start address



Thread Start Functions

- Process Explorer can map the addresses within a module to the names of functions
  - This can help identify which component within a process is responsible for CPU usage
- Requires access to:
  - Symbol file for that module
  - Proper version of Dbghelp.dll

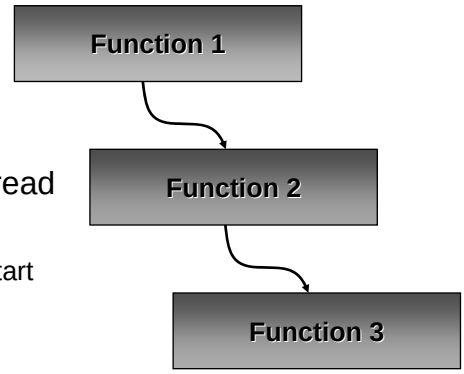
Lab: Threads

- Double click on the Svchost process running under Local Service (if Win2000, pick any Svchost)
  - Go to Threads tab
  - Find thread with the most number of context switches
  - Look at the start address to determine which service DLL it's running

1-42

Call Stacks

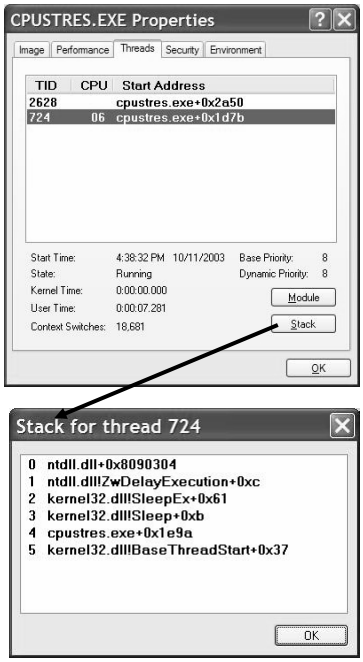
- Process Explorer can also show the thread call stack
  - Represents sequence of functions called
- Important if start address doesn't indicate what the thread is doing
  - E.g. if it's a generic library start routine



1-43

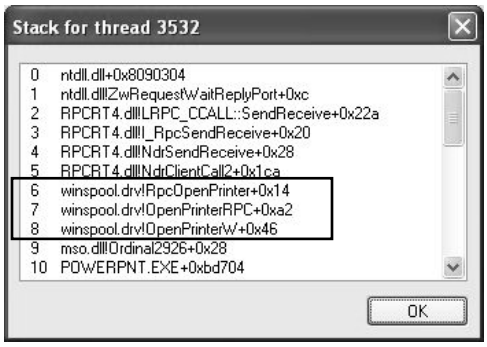
Call Stacks

- Click Stack to view call stack
  - Lists functions in reverse chronological order
- Note that start address on Threads tab is different than first function shown in stack
  - This is because all threads created by Windows programs start in a library function in Kernel32.dll which calls the programmed start address



Example: Solving Hung Processes

- Problem: Powerpoint was hanging for 1 minute on startup
- Thread stack shows waiting on a printer driver



Hung Processes

- If thread stack doesn't solve it, get a memory dump of the process address space and send to developer
  - Use Adplus in Debugging Tools
    - Can snapshot a group of processes at the same time
  - Can also run Drwtsn32.exe
    - Drwtsn32.exe -p <processid>
      - Creates User.dmp file
      - Run Drwtsn32.exe with no switches to see crash dump path
        - Note: in XP and later, by default does not create a full dump

1-46

Understanding System Process CPU Time Consumption

- The "System" process is another "multi-service process"
  - Windows NT 4.0: PID 2, Windows 2000: PID 8, Windows XP: PID 4
- Contains kernel-mode 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
- System threads can be in other processes
  - Look for threads with 100% kernel mode time
  - Example: CSRSS contains threads running pieces of Win32k.sys

1-47

Examining 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

1-48

Lab: System Threads

1. Generate network file access activity, for example, “dir \\computername\c$ /s”
   - System process should be consuming CPU time
2. Open System process process properties
3. Go to Threads tab
4. Sort by CPU time and find thread(s) running
5. Determine what driver these are in

1-50

Agenda

- Introduction
- Investigating Processes & Threads
- Accounting for CPU Usage
- Process Handle Table
- DLLs and Memory Mapped Files
- Investigating Unknown Processes

1-51

Handle View

- Each process has a list of open “objects”
  - Files, Registry keys, synchronization objects, TCP/UDP ports…
  - May be useful to query this list
- Microsoft tools:
  - Oh.exe in Resource Kit
  - XP/2003 have new “Openfiles /query” command
    - Only shows handles to open files – not other non-file objects
  - Both require setting a global flag and rebooting (see Gflags.exe in Support Tools)
- Process Explorer can show open handles without this flag
  - Uses a device driver

1-52

Uses Of Handle View

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

1-53

Lab: Handle Difference Highlighting

1. Open a command prompt (Cmd.exe)
2. Open Process Explorer and pause it
3. View the command-prompt's handles in Process Explorer and identify the handle to the current directory
4. Change directories in the command prompt
5. Refresh in Process Explorer to see the old directory handle close and new one open

1-54

Agenda

- Introduction
- The Process View
- Accounting for CPU Usage
- The Handle View
- The DLL View
- Investigating Unknown Processes
- System Information

1-56

DLL View

- Click on View->DLL View
  - Shows more than just loaded DLLs
  - Includes .EXE and any “memory mapped files”
    - High speed file access mechanism
    - Makes file appear as virtual memory
- Uses:
  - Detect DLL versioning problems
    - Compare the output from a working process with that of a failing one (use File->Save As)
  - Find which processes are using a specific DLL (search for it)

1-57

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
- If this occurs, and it's not due to permissions problems, simply search DLL list with Process Explorer for file
- Lab
  - Run Notepad.exe and then try to delete \windows\notepad.exe
  - Do a search for Notepad.exe in DLL view

1-58

Agenda

- Introduction
- Investigating Processes & Threads
- Accounting for CPU Usage
- Process Handle Table
- DLLs and Memory Mapped Files
- Investigating Unknown Processes

1-59

Investigating Unknown Processes

- Some processes have no identifiable information
  - No version information, are in the System32 directory, etc.
- Right click->Google or MSN Search
- Check the parent process
- Look at the Handle table for file or Registry key handles

1-60

Suspend it While You Look

- Process Explorer can suspend a process
  - This suspends the threads
- Why would you want to do this?
  - Gives you time to investigate possible malware without having to kill it on sight
- Other applications:
  - You’ve started a long running job but want to pause it to do something else
    - Lowering the priority still leaves it running…
  - You’ve started a long download but want to have your network bandwidth temporarily
  - Some multi-service system process activity is due to other processes calling upon their services
    - Suspend a process that is consuming CPU time to see what that does to the system process in question

1-61

Examine Open TCP/IP Endpoints

- Open connections might reveal spyware
- Netstat (part of XP) command lists all endpoints
  - TCPView from Sysinternals also lists this
- Or, use Process Explorer TCP/IP tab
  - Can display thread stack of thread that opened endpoint

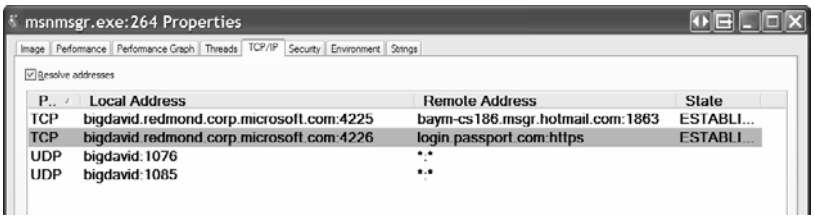


Image Strings

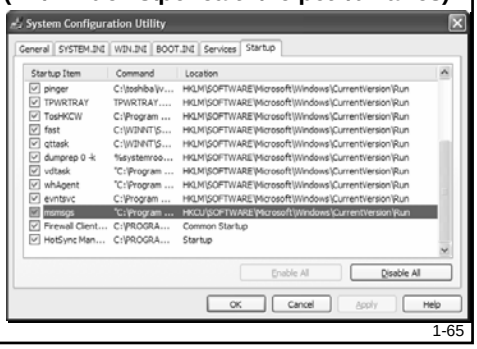
- Strings tab on process properties shows printable Unicode and Ascii strings in .EXE
  - Might find clues from registry keys, URLs or error messages
  - Need also to check strings in loaded DLLs
    - Choose DLL view in Process Explorer
      - Select DLL and right click->Strings
- Also can use command line Strings.exe tool from Sysinternals

Understanding Who Started Processes

- New XP built-in “System Configuration” tool shows startup processes
  - To run, click on Start->Help, then “Use Tools…”, then System Configuration Utility (or run Msconfig.exe)
- Autoruns (from Sysinternals) shows many more places things CAN be defined to start

Autoruns (Sysinternals)

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



End of Part 1

Next: Troubleshooting Memory Problems

Outline

1. Process & Thread Troubleshooting
2. Understanding & Troubleshooting Memory Problems
3. Troubleshooting with Filemon & Regmon
4. Crash Dump Analysis
5. Boot & Startup Troubleshooting

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…

2-2

Windows Memory Management

- Demand paged virtual memory
  - Unit of protection and usage is one page
    - x86, EM64T, AMD64: 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)
- Intelligent, automatic sharing of memory
- Supports up to 128 GB (32-bit systems) or 1024 GB (64-bit systems) physical memory

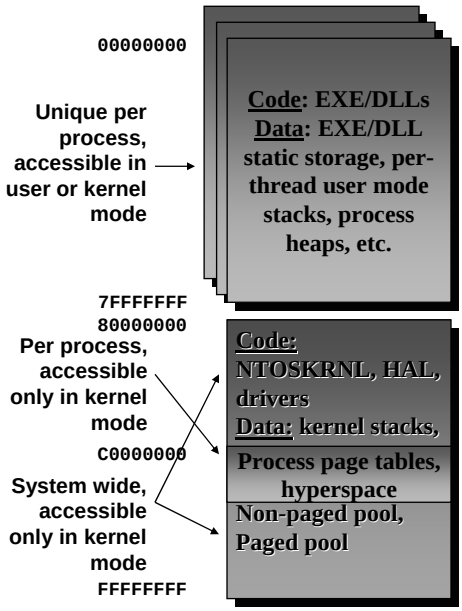
2-3

Physical Memory Limits (in GB)

	x86	x64 32-bit	x64 64-bit	IA-64 64-bit
XP Home	4	4	n/a	n/a
XP Professional	4	4	16	16
Server 2003 Web Edition	2	2	n/a	n/a
Server 2003 Standard	4	4	16	n/a
Server 2003 Enterprise	32	32	64	64
Server 2003 Datacenter	64	128	1024	1024

2-4

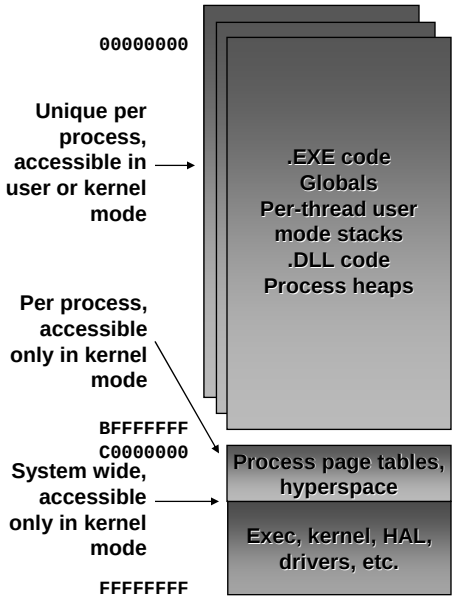
Process Memory Usage



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")

3GB Process Space Option

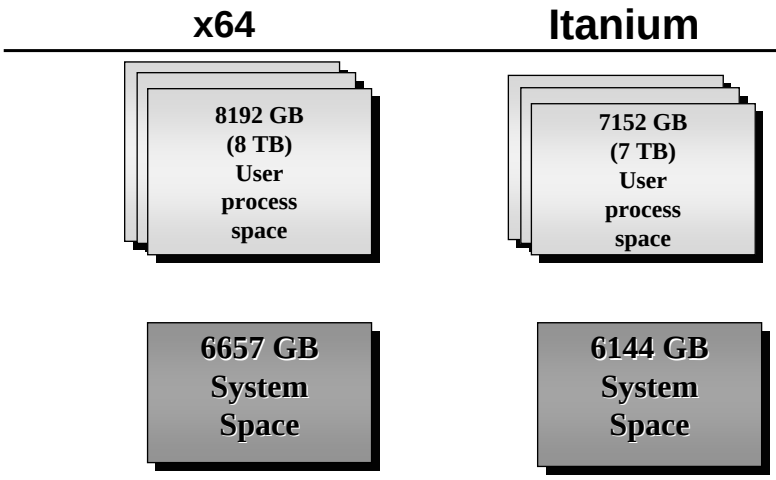


- /3GB option in BOOT.INI
  - Provides up to 3 GB per-process address space
  - Windows Server 2003 & XP SP2 supports variations from 2GB to 3GB (/USERVA=)
- Restrictions to use:
  - Only available on Windows 2000 Advanced Server & Server 2003 & XP SP2
    - Limits memory to 16 GB
  - .EXE must have "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)

2-7

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



2-8

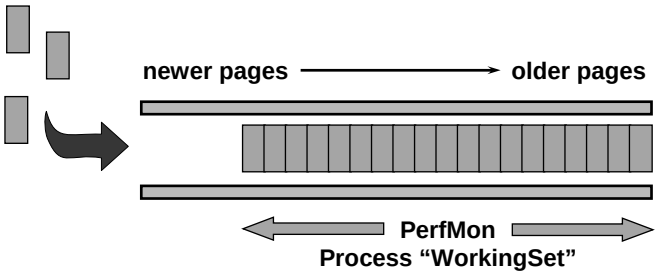
Process Memory Usage

- What limits total process private 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

2-9

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



2-10

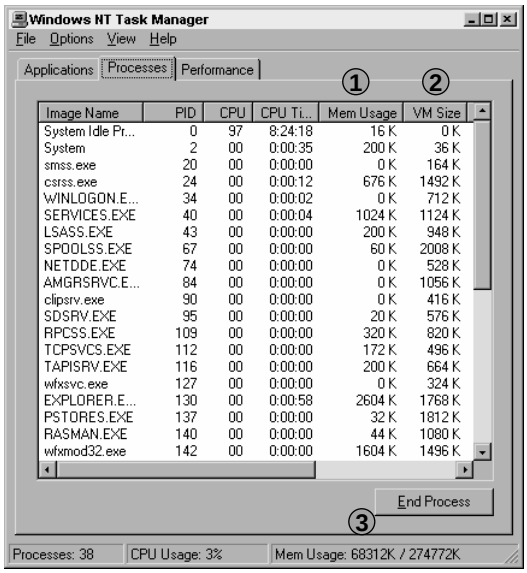
Process Memory Information

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)c**



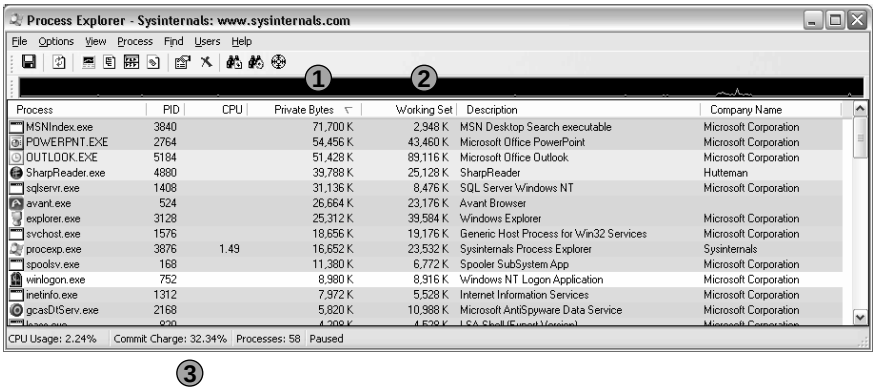
Screen snapshot from: Task Manager | Processes tab

Process Explorer

① **"Private Bytes" = Task Manager "VM Size"**

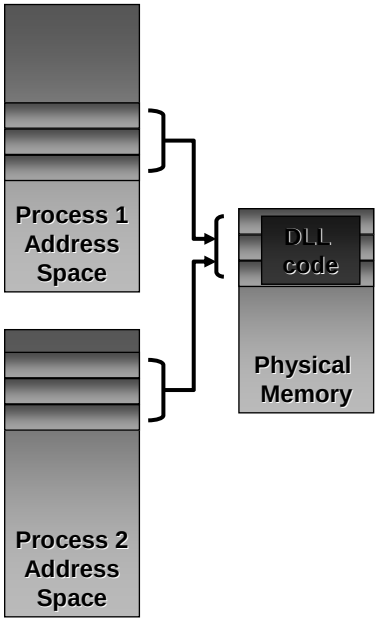
② **"Working Set" = Task Manager "Mem Usage"**

③ **Commit Charge shown as percentage in status bar**



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



2-13

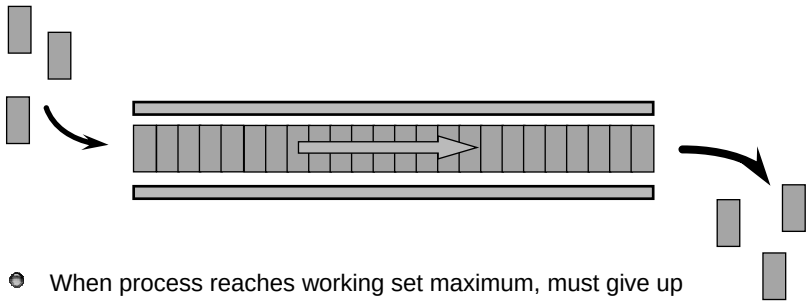
Viewing the Working Set

- 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
```

2-14

Working Set Replacement



- When process reaches working set maximum, must give up 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 Server 2003: All systems

Kept in memory on standby or modified page list

2-15

Paging Lists

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

Standby And Modified Page Lists

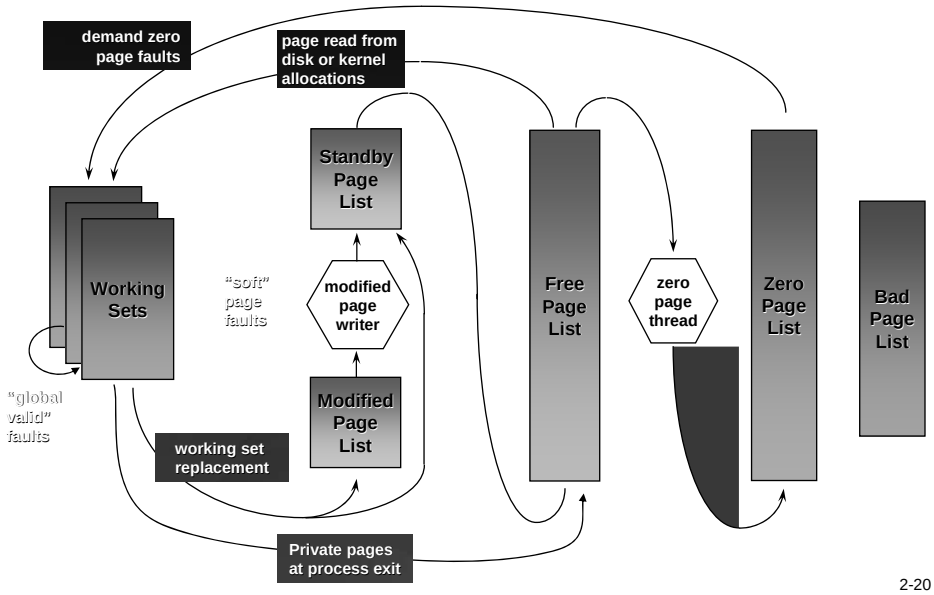
- Modified pages go to modified (dirty) list
  - Avoids writing pages back to disk too soon
- Unmodified pages go to standby (clean) list
- They form a system-wide cache of “pages likely to be needed again”
  - Pages can be faulted back into a process from the standby and modified page list
  - These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

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

2-19

Paging Dynamics

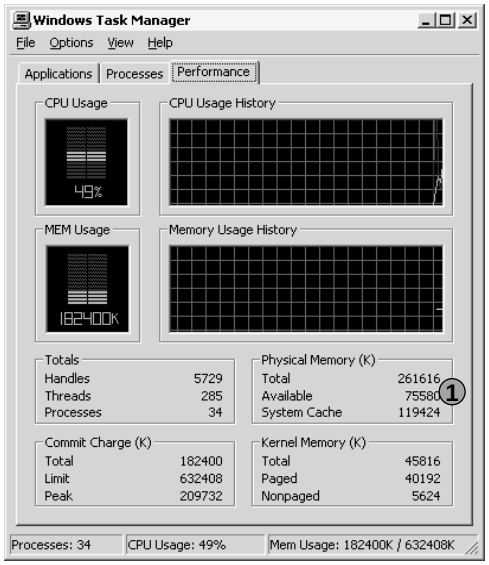


2-20

Memory Management Information

Task Manager
Performance tab

1. "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)
```

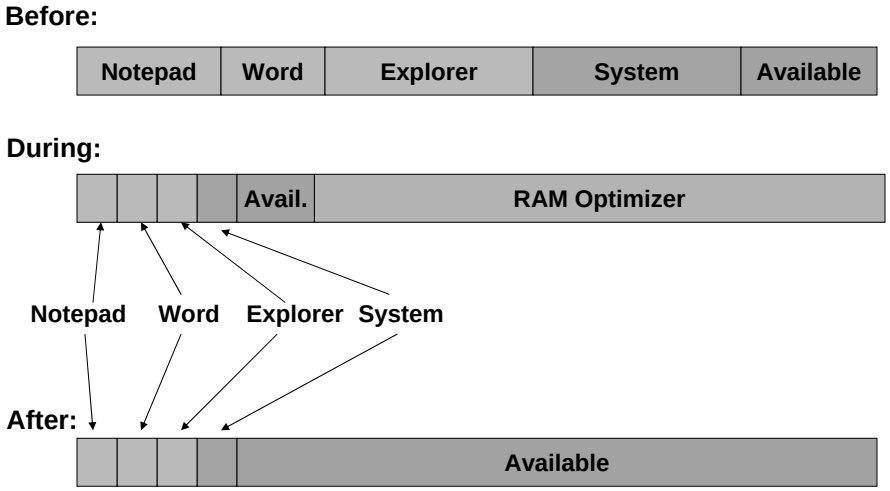
Screen snapshot from:kernel debugger
!memusage command

Do I Need More Memory?

- No single performance counter can answer this
- Page fault activity doesn't necessarily show it
  - First, don't forget to monitor page reads, not page faults (soft faults don't indicate a need for more memory)
  - But, some hard page faults are unavoidable!
    - Process startup
    - Normal file I/O done via paging
- The Solomon/Russinovich "Simple" formula:
  - Available Memory is small most of the time
- To be any more sure, must trace I/Os and analyze
  - If I/Os occur frequently to the same part of the same files OR reads from Pagefile.sys, memory might help
    - Note: Filemon does not show reads from Pagefile.sys if in basic mode

2-23

Why "Memory Optimizers" are Fraudware



See article on this topic at
http://www.winnetmag.com/Windows/Article/ArticleID/41095/41095.html

2-24

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

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)

2-27

Sizing The Page File

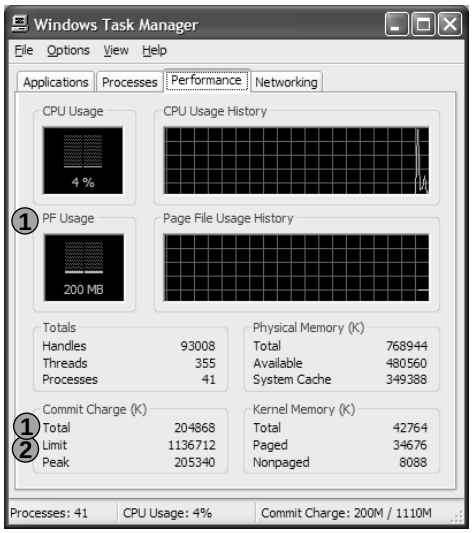
- Worst case: Windows has to page all private data out to make room for code pages
  - To handle, minimum size should be the maximum of VM usage (“Commit Charge Peak”)
    - Hard disk space is cheap, so why not double this
  - Normally, make maximum size same as minimum
  - But, max size could be much 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

2-28

Memory Management Information

Task Manager Performance tab

1. 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”
1. “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

Memory Leaks

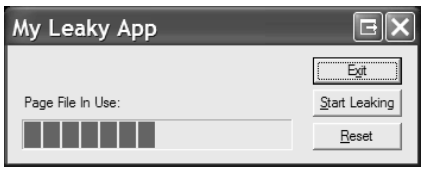
Process Memory Leaks

- 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

2-32

Lab: Process Memory Leak

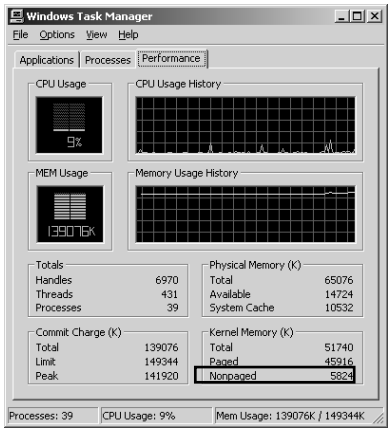
- Leakyapp.exe is in the 2000 Resource Kit
  - Continuously allocates private, nonshareable virtual memory
  - When there is no more, it just keeps trying..
- To cause a memory leak, run it and press Start Leaking
  - Run several copies to fill pagefile more quickly

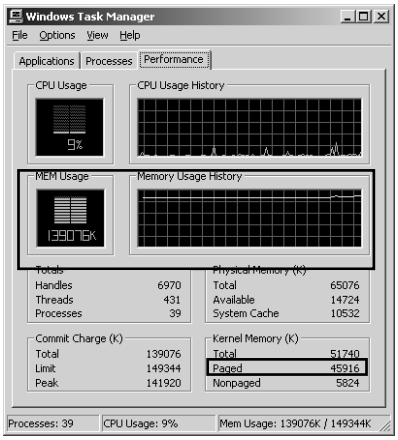


2-33

Kernel Memory Leaks

- A driver leaking nonpaged pool shows up as large and growing Nonpaged pool usage
- Or, a growing Memory Usage and Paged pool usage





2-34

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
- Sysinternals Testlimit with –h option leaks handles
  - http://www.sysinternals.com/files/testlimit.zip

2-35

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

2-36

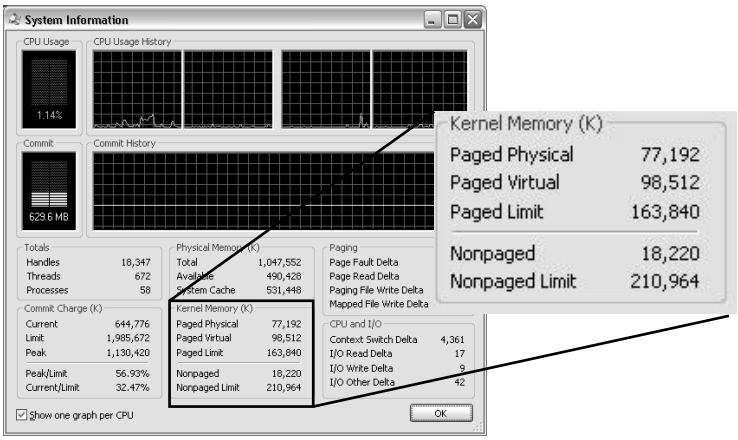
Kernel Memory Pools

- 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

2-37

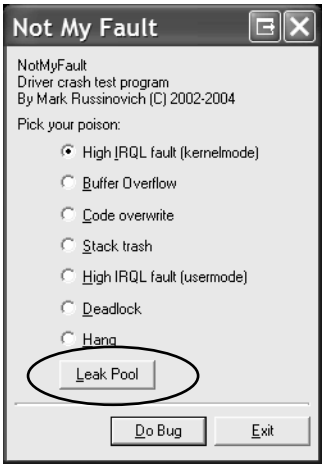
Process Explorer

- Or, just use Process Explorer's System Information dialog



Lab: Causing a Pool Leak

- Run NotMyFault and select "Leak Pool"
  - http://www.sysinternals.com/files/notmyfault.zip
  - Allocates paged pool buffers and doesn't free them
  - Stops leaking when you select "Stop Leaking"

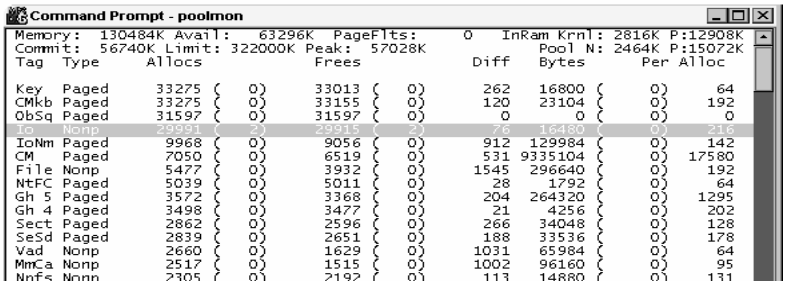


Debugging Pool Leaks

- 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

Monitoring Pool Usage

- 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 (not in XP or Win2000)



```
Command Prompt - poolmon
Memory:  130484K Avail:   63296K  PageFlts:     0   InRam Krnl: 2816K P:12908K
Commit:  56740K Limit: 322000K Peak:  57028K            Pool N: 2464K P:15072K
Tag  Type     Allocs            Frees             Diff   Bytes      Per Alloc

Key  Paged     33275 (   0)     33013 (   0)      262   16800 (     0)     64
CMkb Paged     33275 (   0)     33155 (   0)      120   23104 (     0)    192
ObSq Paged     31597 (   0)     31597 (   0)        0       0 (     0)      0
[illegible]    [illegible]      [illegible]        [illegible]  [illegible]
IoNm Paged      9968 (   0)      9056 (   0)      912  129984 (     0)    142
CM   Paged      7050 (   0)      6519 (   0)      531 9335104 (     0)  17580
File Nonp       5477 (   0)      3932 (   0)     1545  296640 (     0)    192
NtFC Paged      5039 (   0)      5011 (   0)       28    1792 (     0)     64
Gh 5 Paged      3572 (   0)      3368 (   0)      204  264320 (     0)   1295
Gh 4 Paged      3498 (   0)      3477 (   0)       21    4256 (     0)    202
Sect Paged      2862 (   0)      2596 (   0)      266   34048 (     0)    128
SeSd Paged      2839 (   0)      2651 (   0)      188   33536 (     0)    178
Vad  Nonp       2660 (   0)      1629 (   0)     1031   65984 (     0)     64
MmCa Nonp       2517 (   0)      1515 (   0)     1002   96160 (     0)     95
Nnfs Nonp       2305 (   0)      2192 (   0)      113   14880 (     0)    131
```

Controls: “p” to toggle between nonpaged, paged pool, or both
“b” to sort by total # of bytes, “a” to sort by # of allocations, “t” to sort by structure tag
“?” displays help

Troubleshooting with Poolmon

- Once you find pool tag that is leaking, need to find which driver is creating the tag
  1. Try looking up in Windows Debugging Tools subfolder \triage\pooltag.txt
     - May not be there if 3rd party driver
  2. If not, run Strings (from Sysinternals) on all drivers:

     ```
     strings \windows\system32\drivers\*.sys | findstr
       Xyzz
     ```
- Or, use new Poolmon in 2003 DDK to generate local pool tags
  - Poolmon -c will create a "localtag.txt" (if not present)
  - Scans binaries of loaded drivers for pool tags
    - Still missed drivers that are loaded but deleted

2-42

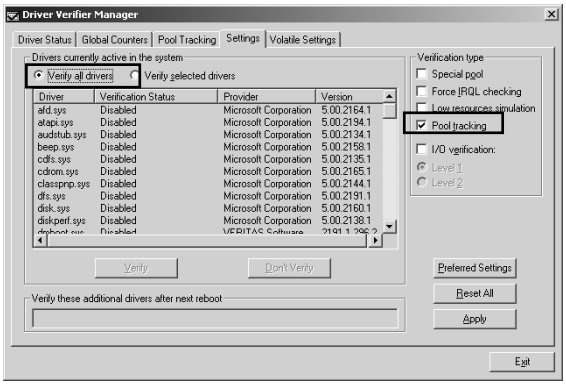
Finding All the Drivers

- Note that while most drivers are in \Windows\System32\Drivers, they can be loaded from anywhere
- To check the location of all drivers:
  - Run Msinfo32.exe, click on Software Environment->System Drivers, sort by Path
  - Or, type "Driverquery /v" (XP & 2003)
- However, some drivers are deleted after they are loaded
  - Binary file and registry key can be deleted after load
    - Examples: Process Explorer, Filemon, Regmon
- Instead, list system loaded kernel mode module list
  - Drivers.exe or Pstat.exe
  - "lm k" in Kernel Debugger
  - Process Explorer DLL view of System process

2-43

Troubleshooting with Driver Verifier

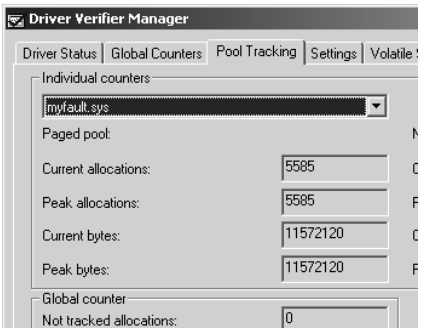
- Use Driver Verifier to enable pool tracking for all drivers (or ones of interest)
  - System tracks pool usage by driver
    - Poolmon looks at pool usage by structure tag



2-44

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



2-45

End of Part 2

Next: Troubleshooting with Filemon & Regmon

Outline

1. Process & Thread Troubleshooting
2. Understanding & Troubleshooting Memory Problems
3. Troubleshooting with Filemon & Regmon
4. Crash Dump Analysis
5. Boot & Startup Troubleshooting

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

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

3-3

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)

3-4

Example: Word Crash

- While typing in the document Word XP closes without any prompts
- See filemon-lab2.log and try and determine why
  - Go to end of Filemon log and search backwards for Winword.exe
  - Stop at first unexplainable activity

Solution: Word Crash

- Working backwards, the first “strange” or unexplainable behavior are the constant reads past end of file to MSSP3ES.LEX

	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

DLL Problems

- Process Explorer may solve a DLL versioning issue, but may not if:
  - A DLL is missing
  - The order of DLL loads is relevant
- So, use Filemon!
  - Look at the last DLL opened before the application died
  - Compare the startup of a working with a failing application
    - Missing or inaccessible DLLs often not reported correctly
      - Look for “NOTFOUND” or “ACCESS DENIED”
    - May be opening wrong versions due to wrong versions being in folders in PATH

3-9

Example: 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

3-10

Example: 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
- Compare filemon-lab4-good.log with filemon-lab4-bad.log

3-13

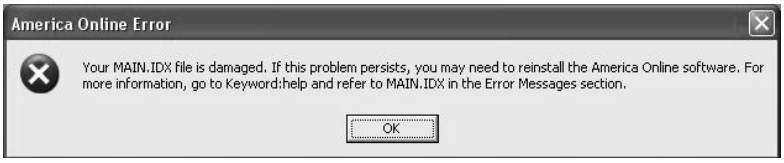
Solution: Access Hangs

- Failing system was loading an old Access DLL from \winnt\system32 due to having installed Access 97 previously
- First unexplainable difference was that Accwiz.dll was being loaded from two different directories
- Removed DLL in \winnt\system32 and problem went away

3-14

Example: Access Denied

- AOL reported this error:



- Filemon showed this:

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

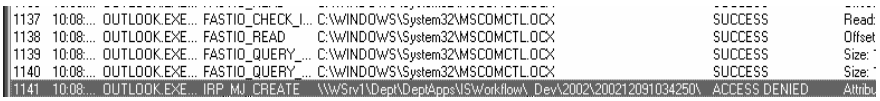
- User did not have admin rights to AOL directory

3-15

Example: Outlook Application Error

- For example, an Outlook application failed with this error:

- Ran Filemon and found it was getting Access Denied



1137	10:08:...	OUTLOOK.EXE...	FASTIO_CHECK_I...	C:\WINDOWS\System32\MSCOMCTL.OCX	SUCCESS	Read:
1138	10:08:...	OUTLOOK.EXE...	FASTIO_READ	C:\WINDOWS\System32\MSCOMCTL.OCX	SUCCESS	Offset
1139	10:08:...	OUTLOOK.EXE...	FASTIO_QUERY_...	C:\WINDOWS\System32\MSCOMCTL.OCX	SUCCESS	Size:
1140	10:08:...	OUTLOOK.EXE...	FASTIO_QUERY_...	C:\WINDOWS\System32\MSCOMCTL.OCX	SUCCESS	Size:
1141	10:08:...	OUTLOOK.EXE...	IRP_MJ_CREATE	\\WSrv1\Dept\DeptApps\ISWorkflow_Dev\2002\200212091034250\	ACCESS DENIED	Attribu

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

3-16

File I/O Activity

- Sometimes, applications perform needless file I/O
  - Run Filemon to see how “quiet” your system is when you think nothing is going on
- Some applications perform inefficient file I/O
  - May be due to libraries used in application that indirectly cause needless I/O

3-18

Lab: Notepad File Save

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

3-19

Understanding Disk Activity

- Performance counters show which disks are being hit, but not which files
- Filemon pinpoints which file(s) are being accessed and how frequently
- Example: used Filemon on a server to determine which file(s) were being accessed most frequently
  - Imported into Excel and make a pie chart by file name or operation type
  - Moved these files to a different disk on a different controller

3-20

Agenda

- Troubleshooting with Filemon
- Troubleshooting with Regmon

3-21

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…

3-22

Registry Activity

- Normally, registry activity should be only at application/system startup and exit
  - But, sadly, lots of processes perform needless registry querying…
- Try running Regmon to see how "quiet" your registry is

3-23

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

3-24

Example: Missing Word Toolbar

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

3-25

Example: IE Error

- Internet Explorer failed to start with this error:

- See c:\lab\regmon-lab2.log

3-26

Solution: IE Error

- 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

3-27

Example: IE Hangs

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

3-28

Solution: IE Hangs

- 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

3-29

Using Regmon

- Sometimes queries to what is not there is more interesting than what is there
  - Identify missing Registry keys
    - Search for status “NOTFOUND”
  - May reveal hidden capabilities

3-30

Filemon/Regmon as a Service

- Sometimes need to capture I/O or registry activity during the logon or logoff process
  - E.g. errors occurring during logon/logoff
- Solutions:
  - Install and run Filemon/Regmon as a service using Srvany tool in Resource Kit
    - Can configure to start at system boot
  - For a quick, one-time execution, run Filemon/Regmon with “psexec –s –i”
- In either case, but tools remain running after logoff

3-31

End of Part 3

Next: Crash Dump Analysis

Outline

1. Process & Thread Troubleshooting
2. Understanding & Troubleshooting Memory Problems
3. Troubleshooting with Filemon & Regmon
4. Crash Dump Analysis
5. Boot & Startup Troubleshooting

Outline

- Crash dumps and tools
- Analysis basics
  - IRQLs
  - Stacks
- Analyzing an “easy” crash
- Un-Analyzable crashes
  - Crash transformation
  - Buffer overrun
  - Code overwrite
- Manual analysis
  - Hung Systems
  - When there is no crash dump

Introduction

- Many systems administrators never attempt crash dump analysis
  - "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?
- More advanced crash dump analysis much harder
  - Not well documented
  - Requires advanced internals knowledge
  - Often difficult to pinpoint cause
    - More often than not, victim is often not the culprit
    - e.g. a driver corrupts an operating system structure; NT crashes later

4-3

Why Does Windows Crash?

- Top 100 Reported Crashing Issues (reported at WinHEC 2004 conference)
  - ~70% caused by 3rd party driver code
  - ~15% caused by unknown (memory is too corrupted to tell)
  - ~10% caused by hardware issues
  - ~5% caused by Microsoft code
- There are lots of third party drivers!
  - From online crash analysis database:
    - 28,000 unique drivers on XP (9 new / day)
    - 130,000 total drivers seen on XP (88 revised / day)
  - Many Devices
    - Over 680,000 distinct Plug and Play (PnP) IDs
    - 1,600 PnP IDs added every day

4-4

Blue Screens

- The kernel and all drivers are “trusted” and share the system address space
  - Any driver or the kernel can cause corruption or an illegal operation
  - Many illegal actions can go undetected for long periods of time
- What causes crashes?
  - Something in kernel-mode detects a problem and calls KeBugCheckEx
  - User-mode applications cannot directly cause a crash

4-5

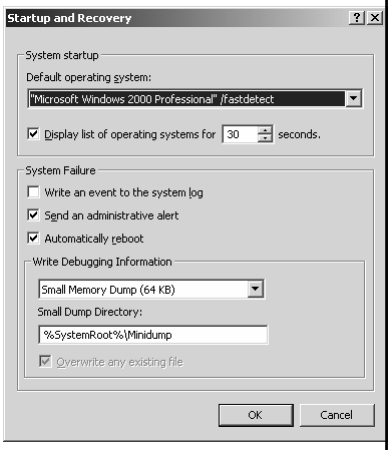
What Happens At The Crash

- When a condition is detected that requires a crash, KeBugCheckEx is called
  - Takes five arguments:
    - Stop code (also called bugcheck 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 (and it is safe to do so), writes dump to disk

4-6

Crash Dumps Options

- Small Memory Dump (aka minidump)
  - Default for Windows 2000/XP Professional/Home
  - Only 64kb (128kb on 64-bit systems)
  - Contains minimal crash information
  - Creates a unique file name in \Windows\Minidump after reboot
- Kernel
  - Writes OS memory and not processes
    - most crash debugging doesn't involve looking at process memory anyway
  - Useful for large memory systems
  - Overwrites every time
- Full
  - Default for Server
  - Writes all of RAM
  - Overwrites every time



Minidumps

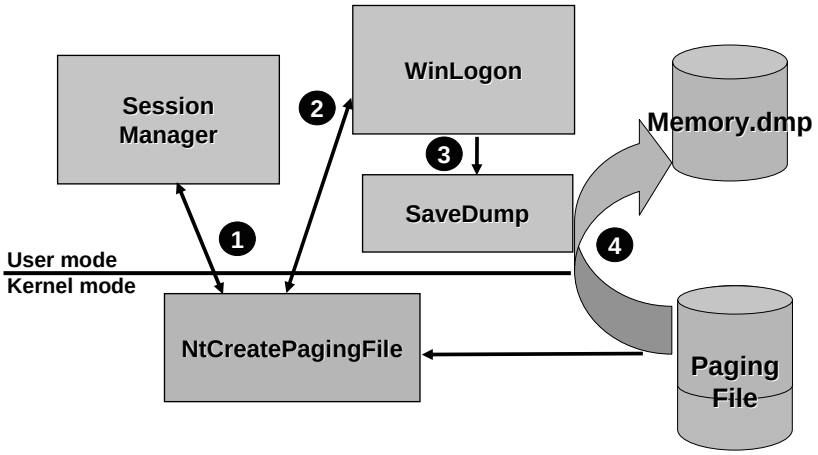
- On Windows XP and Windows Server 2003, minidump is always created, even if system set to full or kernel dump
- Can extract a minidump from a kernel or full dump using the debugger ".dump /m" command
- To analyze, requires access to the images on the system that crashed
  - At least must have have access to the Ntoskrnl.exe
  - Microsoft Symbol Server now has images for Windows XP and later
    - Set image path to same as symbol path (covered later)

Writing a Crash Dump

- Crash dumps are written to the paging file
  - Too risky to try and create a new file (no guarantee you will get a dump anyway)
- How is even this protected?
  - When the system boots it checks HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\CrashControl
  - The *boot disk* paging file's on-disk mapping is obtained
  - Relevant components are checksummed:
    - Boot disk miniport driver
    - Crash I/O functions
    - Page file map
  - On crash, if checksum doesn't match, dump is not written

4-10

At The Reboot



4-11

At The Reboot

- Session Manager process (\Windows\system32\smss.exe) initializes paging file ❶
  - NtCreatePagingFile
- NtCreatePagingFile determines if the dump has a crash header ❷
  - Protects the dump from use
  - Note: crash dump portion of paging file is in use during the copy, so virtual memory can run low while the copy is in progress
- WinLogon calls NtQuerySystemInformation to tell if there's a dump to extract ❸
- If there's a dump, Winlogon executes SaveDump ❹ (\Windows\system32\savedump.exe)
  - Writes an event to the System event log
  - SaveDump writes contents to appropriate file
  - On XP or later, checks to see if Windows Error Reporting should be invoked

4-12

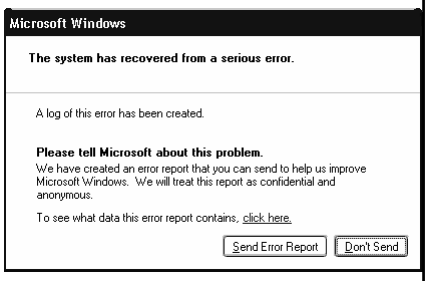
Why Would You Not Get A Dump?

- Most common reasons:
  - Crash occurred before paging file was open
    - E.g. a crash during driver initialization
  - 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 beyond storage drivers must implement dump I/O to get a Microsoft® signature
- How do you troubleshoot a system that is crashing with no resulting dump?
  - Covered later…

4-13

Online Crash Analysis (OCA)

- By Default, after a reboot XP/Server 2003 prompts you to send information to http://watson.microsoft.com
  - Can be configured with Computer Properties->Advanced->Error Reporting
  - Can be customized with Group Policies
    - Do/do not show UI
    - Send dump to an internal error reporting server



4-14

Windows Error Reporting Internals

- Savedump checks if kernel error reporting is enabled
  - Checks two values under HKLM\Software\Microsoft\PCHealth\ErrorReporting: IncludeKernelFaults and DoReport
- If crash reporting is enabled, Savedump:
  - Extracts a minidump from the dump file (if system set to full or kernel dumps)
  - Writes the name of the minidump under HKLM\Software\Microsoft\PCHealth\ErrorReporting\KernelFaults
  - Adds a command to execute Dumprep.exe to HKLM\Software\Microsoft\Windows\CurrentVersion\Run
    - This will cause it to run at the first user log on
- Dumpprep then:
  - Generates an XML description of system version, drivers present, loaded plug and play drivers and depending on the configuration
  - Displays the message box (if enabled) to send the dump
  - Submits to dump for automatic analysis

4-15

What Gets Sent

1. XML description of system version, drivers present, loaded plug and play drivers
2. Minidump file (Loaded and recently unloaded drivers, basic information on current process, kernel-mode call stack for the interrupted thread, hardware devices installed)


```
<?xml version="1.0" encoding="Unicode" ?>
<SYSTEMINFO>
<SYSTEM>
        <OSNAME>Microsoft Windows XP Professional</OSNAME>
        <OSVER>5.1.2477 0.0</OSVER>
        <OSLANGUAGE>1033</OSLANGUAGE>
</SYSTEM>
<DRIVERS>
        <DRIVER>
        <FILENAME>ac97intc.sys</FILENAME>
                <FILESIZE>98112</FILESIZE>
                <CREATIONDATE>05-17-2001 06:31:52</CREATIONDATE>
        <VERSION>5.10.0.3518</VERSION>
                <MANUFACTURER>Intel Corporation</MANUFACTURER>
        <PRODUCTNAME>
Intel(r) Integrated Controller Hub Audio Driver</PRODUCTNAME>
        </DRIVER>
```


4-16

What Does OCA Do?

- Server analyzes crash information submitted
  - If there is a response, displays a link to get more information
    - E.g. KB article, hot fix, updated 3rd party driver, etc.
  - If not, you can check back later at oca.microsoft.com
- If no solution, you should try and analyze dump to tell you:
  - What driver to disable, update, or replace with different hardware
  - What OEM to send the dump to

4-17

Manual Crash Analysis

- There are two kernel-level debuggers that can open crash dump files:
  - WinDbg –Windows program
  - Kd – command-line program
  - Both provide same kernel debugger analysis commands
- Get the latest from: http://www.microsoft.com/whdc/DevTools/Debugging/default.mspx
  - Supports Windows NT 4, Windows 2000, Windows XP, Windows Server 2003 (32-bit and 64-bit)
  - Check for updates frequently

4-18

Manual Crash Analysis

- Open the crash dump with Windbg
- 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
  - !analyze uses heuristics to determine what the likely cause of the crash is
- Debugger tells you to type “!analyze –v” to get detailed debugging information
  - Explains the bugcheck code
  - Interprets the bugcheck parameters

4-19

Symbol Files

- In order to analyze a crash dump, Debugger needs access to (at least) the symbol file for Ntoskrnl.exe
  - Symbol files contain global function and variable names
- Easiest to use Microsoft Symbol Server for symbol access
  - Windbg: click on File->Symbol File Path
  - Enter “srv*c:\symbols*http://msdl.microsoft.com/download/symbols”
- If a minidump, must also configure image path to point to location of images (File->Image File Path)
  - Use same string as for symbol server (XP and beyond)

4-20

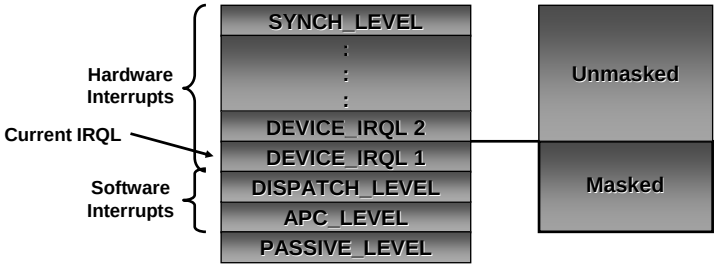
Outline

- Crash dumps and tools
- Analysis basics
  - IRQLs
  - Stacks
- Analyzing an “easy” crash
- Un-Analyzable crashes
  - Crash transformation
  - Buffer overrun
  - Code overwrite
- Manual analysis
  - Hung Systems
  - When there is no crash dump

4-21

IRQLs

- IRQL stands for Interrupt Request Level
  - Each CPU maintains IRQL independently
  - Software and hardware interrupts map to IRQLs
  - When a CPU raises its IRQL to a level all interrupts at that level and below are masked for that CPU



4-22

Key IRQLs

- PASSIVE_LEVEL:
  - No interrupts are masked
  - User mode code always executes at PASSIVE_LEVEL
  - Kernel-mode code executes at PASSIVE_LEVEL most of the time
- DISPATCH_LEVEL:
  - Highest software interrupt level
  - Scheduler is off
  - Page faults cannot be handled and are illegal operations

4-23

Stacks

- The stack is the #1 analysis resource
  - It requires that a driver get “caught in the act”
  - Sometimes that’s not possible without the Driver Verifier’s help
- Each thread has a user-mode and kernel-mode stack
  - The user-mode stack is usually 1 MB on x86
  - The kernel-mode stack is typically 12 KB on x86 systems
- Stacks allow for nested function invocation
  - Parameters can be passed on the stack
  - Stores return address
  - Serves as storage for local variables

4-24

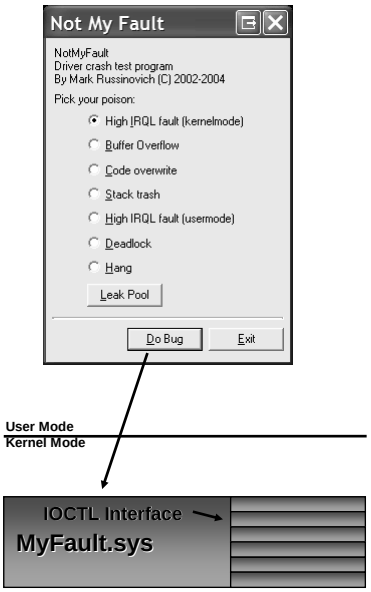
Outline

- Crash dumps and tools
- Analysis basics
  - IRQLs
  - Stacks
- Analyzing an “easy” crash
- Un-Analyzable crashes
  - Crash transformation
  - Buffer overrun
  - Code overwrite
- Manual analysis
  - Hung Systems
  - When there is no crash dump

4-27

NotMyFault.exe

- Test program and driver to demonstrate common crash scenarios
  - It loads MyFault.sys
  - MyFault.Sys has an IOCTL interface that implements different bugs
- Get it at: http://www.sysinternals.com /files/notmyfault.zip



Generating a Straight-Forward Crash

- Run NotMyFault and select “High IRQL fault (kernel mode)”
  - Allocates paged pool buffer
  - Frees the buffer
  - Raises IRQL ≥ DISPATCH_LEVEL
  - Touches the buffer
- Paged buffers that are marked “not present” but are touched when IRQL ≥ DISPATCH_LEVEL result in the DRIVER_IRQL_NOT_LESS_OR_EQUAL bug check
  - Page fault handler calls KeBugCheckEx from page fault handler
  - The IRQL is not less than or equal to the maximum IRQL at which the operation is legal (which is < DISPATCH_LEVEL)

```
*** STOP: 0x000000D1 (0xE1D6F008,0x0000001C,0x00000001,0xEC20635A)
DRIVER_IRQL_NOT_LESS_OR_EQUAL
```

High IRQL Fault Analysis

- !analyze says “memory corruption”
- !analyze –v easily identifies MyFault.sys by looking at the KeBugCheckEx parameters
  - Displays the faulting IP in Myfault.sys

4-30

Outline

- Crash dumps and tools
- Analysis basics
  - IRQLs
  - Stacks
- Analyzing an “easy” crash
- Un-Analyzable crashes
  - Crash transformation
  - Buffer overrun
  - Code overwrite
- Manual analysis
  - Hung Systems
  - When there is no crash dump

4-31

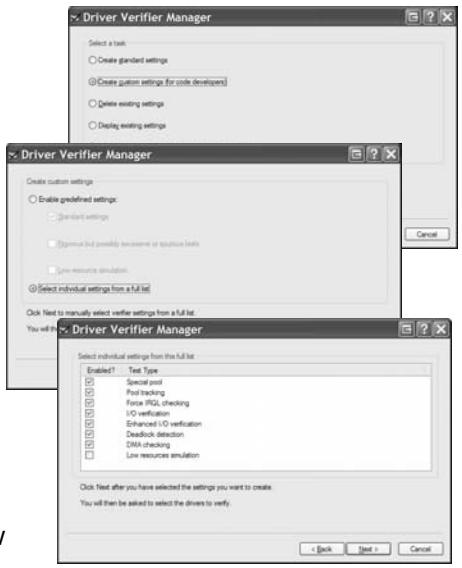
Crash Transformation

- Many crashes can't be analyzed
  - The "victim" crashed the system, not the criminal
  - The analyzer may point at Ntoskrnl.exe or Win32K.sys or other Windows components
  - Or, you may get many different crash dumps all pointing at different causes
- You're goal isn't to analyze impossible crashes…
  - *Its to try to make an "unanalyzable" crash into one that can be analyzed*

4-32

Crash Transformation Recipe

- The tool for crash transformation is the Driver Verifier (Verifier.exe – not in Start menu)
  - Introduced in Windows 2000
  - Helps developers test their drivers and systems administrators identify faulty drivers
- Run Verifier.exe
  - Choose "Create Custom Settings"
  - Choose "Select Individual Settings from a List"
  - Enable all options except Low Resource Simulation



4-33

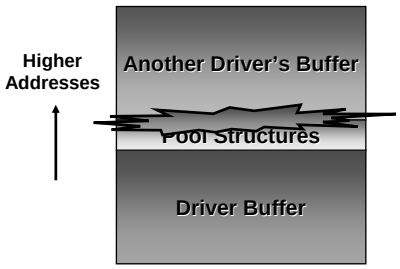
Using Driver Verifier

- Then, need to decide which drivers to verifier
  1. First, try any “suspicious” drivers (recently updated, known to be problematic, etc)
  2. If no crash, try enabling verification on all 3rd party drivers and/or all unsigned drivers
  3. If still no crash, enable verification on all drivers
     - May significantly slow down your system

4-34

Un-Analyzable Crash: Buffer Overrun

- Result when a driver goes past the end (overrun) or the beginning (underrun) of a buffer
- Usually detected when overwritten data is referenced
  - Another driver or the kernel makes the reference
  - There can be a long delay between corruption and detection



4-35

Causing a Buffer Overrun

- Run NotMyFault and select “Buffer Overrun”
  - Allocates a nonpaged pool buffer
  - Writes a string past the end
- Note that you might have to run several times since a crash will occur only if:
  - The kernel references the corrupted pool structures
  - A driver references the corrupted buffer
- The crash tells you what happened, but not why

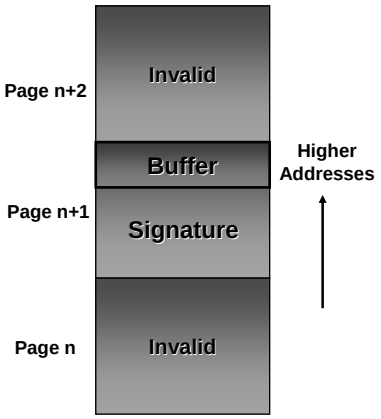
A Buffer Overrun Bluescreen

- In this example, where the crash was the result of the kernel tripping on corrupt pool tracking structures, the Bluescreen tells you what to do:

```
*** STOP: 0x000000C5 (0x4F4F4F4F,0x00000002,0x00000001,0x80467137)

A device driver has corrupted the executive memory pool.

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows 2000 updates you might need.

Run the driver verifier against any new (or suspect) drivers.
If that doesn't reveal the corrupting driver, try enabling special pool.
Both of these features are intended to catch the corruption at an earlier
point where the offending driver can be identified.

If you need to use Safe Mode to remove or disable components,
restart your computer, press F8 to select Advanced Startup Options,
and then select Safe Mode.

Refer to your Getting Started manual for more information on
troubleshooting Stop errors.

*** Address 80467137 base at 80400000, DateStamp 384d9b17 - ntoskrnl.exe
```

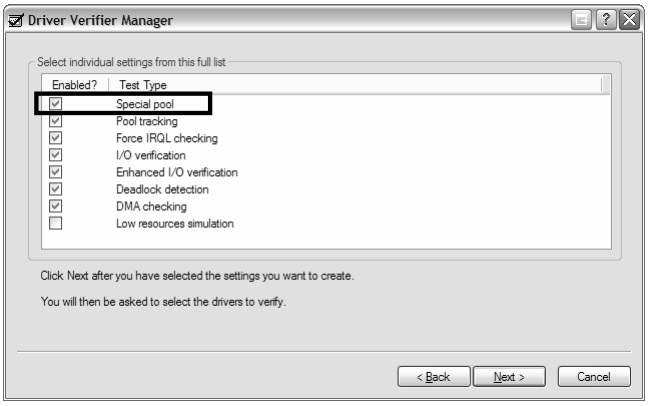
What is Special Pool?

- 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)



Turning on Special Pool

- You get Special Pool when you enable the Verifier



The Verifier Catching Buffer Overrun

- The Driver Verifier catches the overrun when it occurs
  - The blue screen tells you who's fault it is
  - !analyze explains the crash and also tells you the buggy driver name
  - The stack shows where the driver bug is

```
*** STOP: 0x000000D6 (0xBAA08000,0x00000001,0xEC24C3F7,0x00000000)
The driver is attempting to access memory beyond the end of the allocation.
This driver may be at fault :myfault.sys
*** Address EC24C3F7 base at EC24C000, DateStamp 3dd03737 - myfault.sys
```


Un-Analyzable Crash: Code Overwrite

- Caused when a bug results in a wild pointer
  - A wild pointer that points at invalid memory is easily detected
  - A wild pointer that points at data is similar to buffer overrun
    - Might not cause a problem for a long time
    - Crash makes it look like its something else's fault
- System code write protection catches code overwrite, but it's not on if:
  - It's a Windows 2000 system with > 127 MB memory
  - It's a Windows XP or Server 2003 system with > 255 MB

Causing a Code Overwrite

- Run NotMyFault and select “Code Overwrite”
  - Overwrites first bytes of nt!ntreadfile
  - Function is most common entry to I/O system so a random thread will cause the crash

```
*** STOP: 0x0000001E (0xC000001D,0x00000003,0x00000000,0xBFE33B18)
KMODE_EXCEPTION_NOT_HANDLED
```

- The crash hints that the fault occurred in NtReadFile
  - The last user-mode address is ZwReadFile
  - The ebx register in the exception frame points at NtReadFile
  - NtReadFile’s start location looks scrambled (u ntreadfile)

System Code Write Protection

- If at least one driver is being verified then system code write protection is enabled
  - You can also enable it manually:
    - Set HKLM\System\CurrentControlSet\Control
      \Session Manager\Memory Management

      LargePageMinimum REG_DWORD 0xFFFFFFFF
      EnforceWriteProtection REG_DWORD 1
    - Reboot to take effect
- Rerun NotMyFault
  - Crash occurs immediately and even the blue screen points at MyFault.sys:

```
*** STOP: 0x000000BE (0x804BB7FD,0x004BB121,0xBE570B60,0x0000000B)
An attempt was made to write to read-only memory.
This driver may be at fault :myfault.sys

*** Address 804BB7FD base at 80400000, DateStamp 384d9b17 - ntoskrnl.exe
```

  - !analyze shows the address of the write and the target (NtReadFile)

Outline

- Crash dumps and tools
- Analysis basics
  - IRQLs
  - Stacks
- Analyzing an “easy” crash
- Un-Analyzable crashes
  - Crash transformation
  - Buffer overrun
  - Code overwrite
- Manual analysis
  - Hung Systems
  - When there is no crash dump

4-44

Manual Analysis

- Sometimes !analyze –v isn’t enough
  - Doesn’t tell you anything useful
  - You want to know what was happening at the time of the crash
- Useful commands:
  - List loaded drivers: lm kv
    - Make sure drivers are all recognized and up to date
  - Look at memory usage: !vm
    - Make sure memory pools are not full
      - If full, use !poolused (requires pool tagging to be on)
  - Examine current thread: !thread
    - May or may not be related to the crash
  - List all processes: !process 0 0
    - Make sure you understand what was running on the system
  - If a Verifier detected deadlock: !deadlock
  - Additional commands: !help

4-45

Analyzing a “Sick” System

- Sometimes a system is still responsive, but you know that something is wrong with it
  - You want to look at its kernel state, but…
  - You don’t want to take it off line by crashing it or connecting a debugger to it
- You can get a “dump” of a live system with LiveKd (free download from Sysinternals.com)
  - Use it to run Windbg or Kd
  - Use .dump to snapshot live system

4-46

Hung Systems

- Scenerio: “hard hangs” (e.g. no keyboard or mouse response)
- Two techniques: both require prior setup and a reboot
  1. Manually crash the hung system and hope you get a dump to analyze offline
  2. Boot the system in debugging mode and when it hangs, break in with the kernel debugger and analyze system

4-47

Crashing Hung Systems

- Set the keystroke crash option
  - Set HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\i8042prt\Parameters\CrashOnCtrlScroll to 1
  - Reboot
- Rerun NotMyFault
  - Enter right-ctrl+[scroll-lock, scroll-lock] to crash the system

```
*** STOP: 0x000000E2 (0x00000000,0x00000000,0x00000000,0x00000000)
The end-user manually generated the crashdump.
```

  - !analyze won't help
- Use !thread to see what's running
  - Use ~ command to look at each CPU
  - One CPU is executing MyFault code grabbing a spinlock
- Spinlock hangs are much easier to debug than dispatcher object (events, semaphores, mutexes) and resource hangs

4-48

Booting in Debugging Mode

- If you have hangs or crashes with no resulting dump (or other "spontaneous reboots")
  - …you need to get into the kernel debugger at the time of the crash
- Boot in Debugging Mode
  - ◆ Two options to boot in debugging mode:
  1. Press F8 during the boot and choose "Debugging Mode"
  2. Or, edit the target's boot.ini file to configure:
     - /debugport=comX /baudrate=XXX (note: default baud rate in Debugging Mode is 19200)
     - Windows XP and 2003 support 1394
- In either case, this loads the kernel debugger at boot time
  - Does not affect performance
  - But, system will not auto reboot after a crash, even if configured to do so!

4-49

Debugging System in Debugging Mode

- When system hangs or crashes, attach a kernel debugger and analyze
  - In Windbg, choose File->Kernel Debug
    - Configure baud rate and COM port
    - Click OK
- If a crash, Debugger should connect and display the bugcheck code
  - If a hung system, must break in with “Debug->Break”
- Type !analyze –v and/or perform manual analysis commands described earlier
  - To save complete memory dump for offline analysis, use “.dump” (or “.dump /f” to capture a full dump)
    - Note: this will be slow over a serial cable

4-50

End of Part 3

- For more information:
  - Windows Internals: Chapter 14 is on crash dump analysis
  - The help file which is installed with Debugging Tools for Windows
- Next: Boot & Startup Troubleshooting

4-51

Outline

1. Process & Thread Troubleshooting
2. Understanding & Troubleshooting Memory Problems
3. Troubleshooting with Filemon & Regmon
4. Crash Dump Analysis
5. Boot & Startup Troubleshooting

Introduction

- Kinds of problems we're addressing:
  - Error messages during boot
  - Crashes and hangs during boot
  - Errors messages during the logon process
- Typical causes:
  - 3rd party drivers and applications
  - System file corruption due to hardware problems or blue screens (from 3rd party drivers)
  - Malware, viruses…
- Common response: "Reinstall Windows"
  - You can do better than that by understanding the boot and startup process and the tools available to track down and repair problems

Agenda

- The boot process
- MBR corruption
- Boot sector corruption
- Boot.ini misconfiguration
- System file corruption
- Crashes or hangs

Boot Process Terminology

- Boot begins during installation when Setup writes various things to disk
- System volume:
  - Master Boot Record (MBR)
  - Boot sector
  - NTLDR – NT Boot Loader
  - NTDETECT.COM
  - BOOT.INI
  - SCSI driver – Ntbootdd.sys
- Boot volume:
  - System files – %SystemRoot%: Ntoskrnl.exe, Hal.dll, etc.

The Boot Process

1. MBR
   - Contains small amount of code that scans partition table
     - 4 entries
     - First partition marked active is selected as the system volume
   - Loads boot sector of system volume

2. Boot sector (NT-specific code)
   - Reads root directory of volume and loads NTLDR

5-5

x86 and x64 Boot Process

3. NTLDR (screen is black)
   - Moves system from 16-bit to 32-bit mode and enables paging
   - Reads and uses Ntbootdd.sys to perform disk I/O if the boot volume is on a SCSI disk
     - Uses BIOS to read from system volume's disk
     - This is a copy of the SCSI miniport driver used when the OS is booted
   - Reads Boot.ini
     - Boot.ini selections point to boot drive
     - Specifies OS boot selections and optional switches (most for debugging/troubleshooting) that passed to kernel during boot
   - If more than one selection, NTLDR displays boot menu (with timeout)
   - If you select a 64-bit installation, NTLDR moves the CPU into 64-bit mode

5-6

The Boot Process (cont)

3. NTLDR (cont)
   - Once boot selection made, user can type F8 to get to special boot menu
     - Last Known Good, Safe modes, hardware profile, Debugging mode
   - NTLDR executes Ntdetect.com to perform BIOS hardware detection (x86 and x64 only)
     - Later saved into HKLM\Hardware\Description
   - NTLDR loads the SYSTEM hive (HKLM\System), boot drivers, Ntoskrnl.exe, Hal.dll and transfers control to main entry point of Ntoskrnl.exe
     - Boot driver: critical to boot process (e.g. boot file system driver)

5-7

The Boot Process (cont)

4. Ntoskrnl (splash screen appears)
   - Initializes kernel subsystems in two phases:
     - First phase is object definition (process, thread, driver, etc)
     - Second builds on the base that the objects provide
     - This is done in the context of a kernel-mode system thread that becomes the idle thread
   - I/O Manager starts boot-start drivers and then loads and starts system-start drivers
   - Finally, Ntoskrnl creates the Session Manager process (\Windows\System32\Smss.exe), the first user-mode process

5-8

The Boot Process (cont)

5. Smss.exe:
   - Runs programs specified in BootExecute e.g. autochk, the native API version of chkdsk
   - Processes “Delayed move/rename” commands
     - Used to replace in-use system files by hotfixes, service packs, etc.
     - Get Pendmoves from Sysinternals to see registered commands
   - Initializes the paging files and rest of Registry (hives or files)
   - Loads and initializes kernel-mode part of Win32 subsystem (Win32k.sys)
   - Starts Csrss.exe (user-mode part of Win32 subsystem)
   - Starts Winlogon.exe

5-10

The Boot Process (cont)

6. Winlogon.exe:
   - Starts LSASS (Local Security Authority)
   - Loads GINA (Graphical Identification and Authentication) to wait for logon
     - default is Msgina.dll
   - Starts Services.exe (the service controller)
7. Services.exe starts Win32 services marked as “automatic” start
   - Also includes any drivers marked Automatic start (Start value is 2)
   - Service startup continues asynchronous to logons

End of normal boot process

5-11

The Recovery Console

- Description:
  - Simple repair-oriented command-line environment
  - Built on a minimal NT kernel
  - Bootable from Win2K/XP/Server 2003 Setup CD
    - Type “r” to repair and then select the installation
  - Installable onto hard disk (winnt32.exe /cmdcons)

```
Microsoft Windows XP(TM) Recovery Console.

The Recovery Console provides system repair and recovery functionality.

Type EXIT to quit the Recovery Console and restart the computer.

1: C:\WINDOWS

Which Windows installation would you like to log onto
(To cancel, press ENTER)? 1
Type the Administrator password: *********
C:\WINDOWS>
```


The Recovery Console

- Capabilities:
  - File commands: rename, move, delete, copy
  - Service/Driver commands: listsvc, enable, disable
  - MBR/Boot sector commands: fixmbr, fixboot
- Limitations:
  - Must “log into” the system with the Administrator password
  - Limits on what you can access:
    - Only access system directory, \System Volume Information directories, and root of non-removable media
    - Can only copy files onto system, not off
    - You can override these in the Local Security Policy editor (secpol.msc) on the installation when its running
  - No networking, file editing, or registry editing

Agenda

- The boot process
- MBR corruption
- Boot sector corruption
- Boot.ini misconfiguration
- System file corruption
- Crashes or hangs

MBR Corruption

- Symptoms:
  - Hang at a black screen after BIOS executes
  - “Invalid Partition Table”, “Error loading operating system” or “Missing operating system” message on black screen
- Cause:
  - MBR is corrupt
- Resolution:
  - Boot into Recovery Console
  - Execute the RC’s “fixmbr” command
    - If the partition table is corrupt you have to rely on restoring a backup MBR or use 3rd-party disk repair tools

Agenda

- The boot process
- MBR corruption
- Boot sector corruption
- Boot.ini misconfiguration
- System file corruption
- Crashes or hangs

Boot Sector Corruption

- Symptoms:
  - Black screen hang
  - “A disk read error occurred”, “NTLDR is missing” or “NTLDR is compressed” error message on black screen
- Cause:
  - Boot sector corruption
- Troubleshooting:
  - Boot into RC
  - Execute “fixboot” command

Agenda

- The boot process
- MBR corruption
- Boot sector corruption
- Boot.ini misconfiguration
- System file corruption
- Crashes or hangs

Boot.ini Problems

- Symptom:
  - NTOSKRNL complains that boot device is inaccessible

```
Windows could not start because of a computer disk hardware
configuration problem.
Could not read from the selected boot disk.  Check boot path
and disk hardware.
Please check the Windows documentation about hardware disk
configuration and your hardware reference manuals for
additional information.
```

- Cause:
  - Boot.ini is missing or corrupt
  - Boot.ini is out-of-date because a partition has been added

Boot.ini Problems

- Troubleshooting:
  - Boot into RC
  - Run Bootcfg /rebuild

```
C:\>bootcfg /rebuild

Scanning all disks for Windows installations.

Please wait, since this may take a while...

The Windows installation scan was successful.

Note: These results are stored statically for this session.
      If the disk configuration changes during this session,
      in order to get an updated scan, you must first reboot
      the machine and then rescan the disks.

Total identified Windows installs: 1

[1]: C:\WINDOWS
Add installation to boot list? (Yes/No/All): y
Enter Load Identifier: Windows XP
Enter OS Load Options:

C:\>
```


Agenda

- The boot process
- MBR corruption
- Boot sector corruption
- System file corruption
- Boot.ini misconfiguration
- Crashes or hangs

Registry SYSTEM Hive Corruption

- Symptom:
  - NTLDR reports that System hive is corrupt

```
Windows could not start because the following file is missing
or corrupt:
\WINDOWS\SYSTEM32\CONFIG\SYSTEM

You can attempt to repair this file by starting Windows Setup
using the original Setup CD-ROM.
Select 'r' at the first screen to start repair.
```

- Causes:
  - Disk is corrupt
  - System hive is corrupted or deleted

5-22

Registry SYSTEM Hive Corruption

- Troubleshooting:
  - Boot into RC
  - Run Chkdsk
- If no errors, need to get a backup of the SYSTEM hive
  - On XP with System Restore enable, look in \System Volume Information
    - Go to _restore{xxx}\RPnnn\snapshot
    - Copy _REGISTRY_MACHINE_SOFTWARE to \Windows\System32\Config\System
  - Otherwise, copy from \Windows\Repair
    - This backup is created by Windows Setup
    - Backing up “System State” with Windows Backup update the Repair directory

5-23

System File Corruption

- Symptom:
  - Error message indicating that NTLDR, NTOSKRNL.EXE, HAL.DLL or other system file is missing or corrupt

```
Windows could not start because the following file is missing
or corrupt:
<Windows root>\system32\hal.dll.
Please re-install a copy of the above file.
```

  - Blue screen with corruption message

```
STOP: c0000135 {Unable To Locate Component}
This application has failed to start because KERNEL32.dll was not found. Re-inst
alling the application may fix this problem.
```

5-24

System File Corruption

- Causes:
  - Disk is corrupt
  - File is missing or corrupt
- Troubleshooting:
  - Boot into RC
    - Run Chkdsk
  - If no chkdsk errors obtain clean copy of file and replace file
    - Check in \Windows\System32\DLLCache for backup
    - If not there, get a replacement from an identical system i.e. from same hotfix or service pack
  - If can't find replacement use Automated System Recovery (ASR) or some other disk image restore utility

5-25

Agenda

- The boot process
- MBR corruption
- Boot sector corruption
- Boot.ini misconfiguration
- System file corruption
- Crashes or hangs

5-27

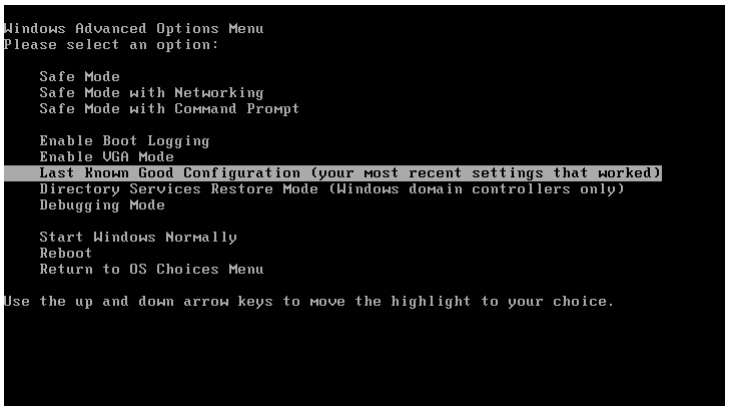
Post-Splash Screen Crash or Hang

- Symptoms:
  - System blue screens on boot
  - Hang before logon prompt appears
  - NOTE: If system auto-reboots on crash you won't see the blue screen!
- Causes:
  - Buggy driver
  - Registry corruption of non-System hive
- Troubleshooting:
  - Last Known Good
    or
  - Safe Mode
    or
  - RC

5-28

Accessing Last Known Good

- Enable it by pressing F8 and selecting it in the Advanced Options boot menu



5-29

LKG Description

- Last Known Good (LKG) Uses backup of registry control set last used to boot successfully
- A Control Set is core startup configuration
  - HKLM\System\Control00n
  - Control set only includes core OS and driver configuration
  - Control set does *not* include Software, SAM, Security, or Users
  - HKLM\System\Select\Current points at active Control Set

5-30

LKG Description

- Boot control makes a copy of the control set that booted the system
  - Copy is ControlSet00n, where 00n is the next available number
- After a successful boot:
  - 1. LastKnownGood is set to the copy
  - 2.The previous LastKnownGood is deleted
- By default, “Successful boot” is determined when
  - All the auto-start services have started successfully
  - A successful interactive log in
    - Can be overridden programmatically

5-31

LKG Capabilities

- Restores bootable configuration when:
  - A new driver was installed since the last successful boot
  - A driver’s settings were modified since the last successful boot
  - System settings were modified since the last successful boot

5-32

LKG Limitations

- Doesn't work if:
  - An existing driver was updated
  - A latent driver bug for some reason becomes active
  - Files or registry hives are missing or corrupt

5-33

Leveraging the Failed Control Set

- When you use LKG the control set you avoid is saved as the Failed control set
  1. Look at the Failed value in the Select key – this is the control set that you aborted
  2. Export the current control set and failed control set to .reg files
  3. Massage the text so that there are no differences in the control set name
  4. Windiff or Fc to see what's different

5-34

Safe Mode Description

- Try Safe Mode if LKG doesn't work
  - Accessible from same boot menu as LKG
- Idea is to only include core set of drivers/services
  - Modeled after Safe Mode in Windows 95
  - Avoids third-party and unnecessary drivers, which hopefully are what's causing the boot problem

5-35

Safe Mode Description

- HKLM\System\CurrentControlSet\Safeboot guides safe mode by specifying names and groups of drivers
  - Normal, Network, Command-Prompt
    - No networking in Normal
    - Networking includes networking services
    - Command-Prompt is same as Normal except launches Command Prompt instead of Explorer as shell for when Explorer shell extensions cause logon problems
  - Directory Services Restore Mode: not for boot troubleshooting (for repairing or restoring Active Directory database from backup)

5-36

Safe Mode Internals

- Registry keys guide what's in safe modes:
  - HKLM\System\CurrentControlSet\SafeBoot\Minimal is for Normal and Command-Prompt
    - HKLM\System\CurrentControlSet\SafeBoot\AlternateShell specifies shell for Command-Prompt boot
  - HKLM\System\CurrentControlSet\SafeBoot\Network is for Network
  - Drivers and services must be listed by name or by group to be loaded
- Exception: all boot-start drivers load regardless!
  - System assumes they are necessary to boot

5-37

Using Safe Mode

- If Safe Mode works determine what's wrong:
  - Compare boot logs
  - Analyze a crash dump (covered in the next section)
- Boot logging:
  - Select it from same menu as LKG and Safe Mode and boot to the failure
    - Saves log in \Windows\Ntbtlog.txt
  - Reboot in Safe Mode
    - Safe Mode appends to the boot log
  - Extract failed boot and Safe Mode entries to separate files, strip "Did not load driver" lines and compare e.g. Windiff, fc

5-38

Resolving the Faulty Driver Issue

- If you can determine what driver is causing the problem:
  - Roll back to a previous version if one is available and known to be stable
    or
  - Disable it with Device Manager
    - Note: can't do this for non-PnP drivers: use the registry editor

5-39

When Safe Mode Fails

- Symptom:
  - Safe mode crashes the same as a normal boot
- Causes:
  - The driver causing the crash also loads in safe mode
- Troubleshooting:
  - Determine the problematic driver:
    - Boot into RC and look at the last line in the boot log
    - Or, boot into debugging mode and analyze crash
  - Disable it with the RC's "disable" command

5-47

The End!

- Thank you for coming
- For more information, see Windows Internals, 4th edition
- Questions: send to daves@solsem.com

5-56