# Unit 1: Overview of Operating Systems

## 1.3. Windows Operating System Family - Concepts & Tools

# Roadmap for Section 1.3.

High-level Overview on Windows Concepts

- Processes, Threads
- Virtual Memory, Protection
- Objects and Handles

Windows is thoroughly instrumented

- Key monitoring tools
- Extra resources at  www.sysinternals.com

# Requirements and Design Goals
## for the original Windows NT project

- Provide a true 32-bit, preemptive, reentrant, virtual memory operating system
- Run on multiple hardware architectures and platforms
- Run and scale well on symmetric multiprocessing systems
- Be a great distributed computing platform (Client & Server)
- Run most existing 16-bit MS-DOS and Microsoft Windows 3.1 applications
- Meet government requirements for POSIX 1003.1 compliance
- Meet government and industry requirements for operating system security
- Be easily adaptable to the global market by supporting Unicode

# Goals (contd.)

- **Extensibility**
  - Code must be able to grow and change as market requirements change.
- **Portability**
  - The system must be able to run on multiple hardware architectures and must be able to move with relative ease to new ones as market demands dictate.
- **Reliability and Robustness**
  - Protection against internal malfunction and external tampering.
  - Applications should not be able to harm the OS or other running applications.
- **Compatibility**
  - User interface and APIs should be compatible with older versions of Windows as well as older operating systems such as MS-DOS.
  - It should also interoperate well with UNIX, OS/2, and NetWare.
- **Performance**
  - Within the constraints of the other design goals, the system should be as fast and responsive as possible on each hardware platform.

# Portability

- HAL (Hardware Abstraction Layer):
  - support for x86 (initial), MIPS (initial), Alpha AXP, PowerPC (NT 3.51), Itanium (Windows XP/2003)
  - Machine-specific functions located in HAL
- Layered design:
  - architecture-specific functions located in kernel
- Windows kernel components are primarily written in C:
  - OS executive, utilities, drivers
  - UI and graphics subsystem - written in C++
  - HW-specific/performance-sensitive parts - written in assembly language: interrupt trap handler, context switching

# Windows API & Subsystems

- Windows API (application programming interface):
  - Common programming interface to three different Windows OSes: Windows NT, Windows 9x and Windows CE (Embedded Compact)
  - OSes implement (different) subsets of the API
  - MSDN: http://msdn.microsoft.com
- Windows supports multiple subsystems (APIs):
  - Windows (primary), POSIX, OS/2
  - User space application access OS functionality via subsystems
- Subsystems define APIs, process, and file system semantics
  - OS/2 used to be primary subsystem for Windows NT 3.x

# 64-bit vs. 32-bit Windows APIs

- Pointers and types derived from pointer, e.g. handles, are 64-bit long
  - A few others go 64, e.g. WPARAM, LPARAM, LRESULT, SIZE_T
  - Rest are the same, e.g., 32-bit INT, DWORD, LONG

Win32 and Win64 are consistently named the Windows API

| API | Data Model | `int` | `long` | pointer |
|---|---|---|---|---|
| Win32 | ILP32 | 32 | 32 | 32 |
| Win64 | LLP64 | 32 | 32 | 64 |
| UNIXes | LP64 | 32 | 64 | 64 |

# Services, Functions, and Routines

- Windows API functions:
  - Documented, callable subroutines
  - *CreateProcess, CreateFile, GetMessage*
- Windows system services:
  - Undocumented functions, callable from user space
  - *NtCreateProcess* is used by Windows *CreateProcess* and POSIX fork() as an internal service
- Windows internal routines:
  - Subroutines inside the Windows executive, kernel, or HAL
  - Callable from kernel mode only (device driver, NT OS components)
  - *ExAllocatePool* allocates memory on Windows system heap

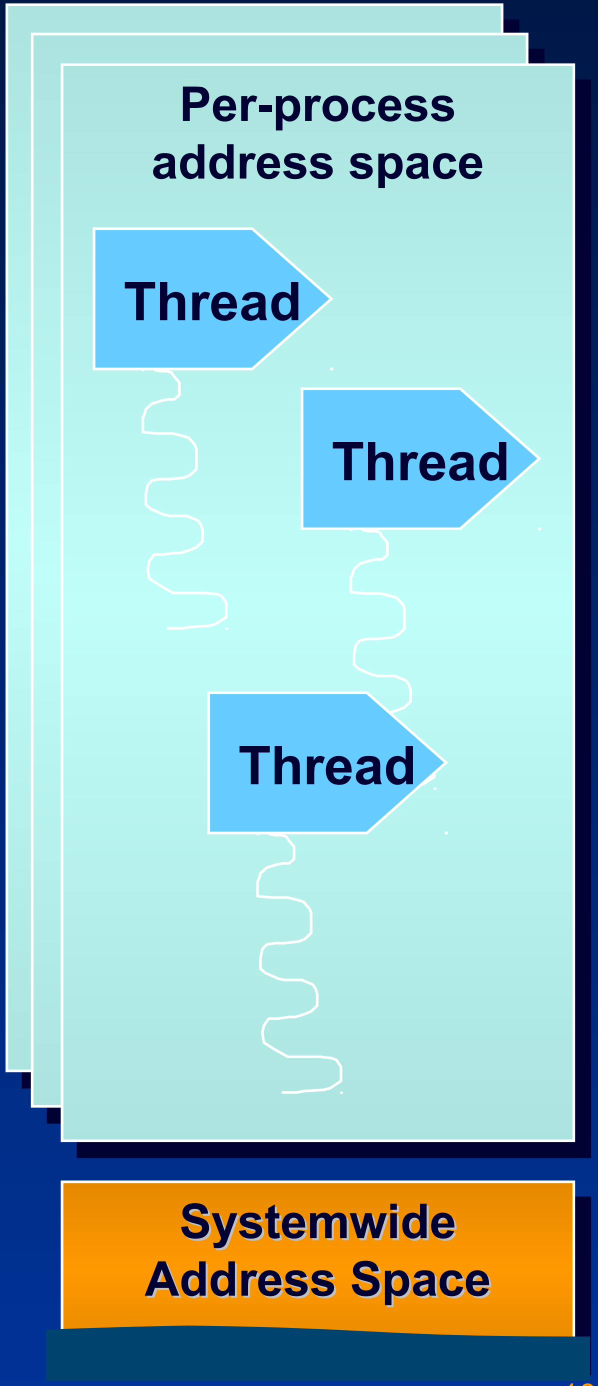
# Services, Functions, and Routines (contd.)

- Windows services:
  - Processes which are started by the Service Control Manager
  - Example: The *Schedule* service supports the at-command
- DLL (dynamic link library)
  - Subroutines in binary format contained in dynamically loadable files
  - Examples: MSVCRT.DLL – MS Visual C++ run-time library

    KERNEL32.DLL – one of the Windows API libraries

# Processes and Threads

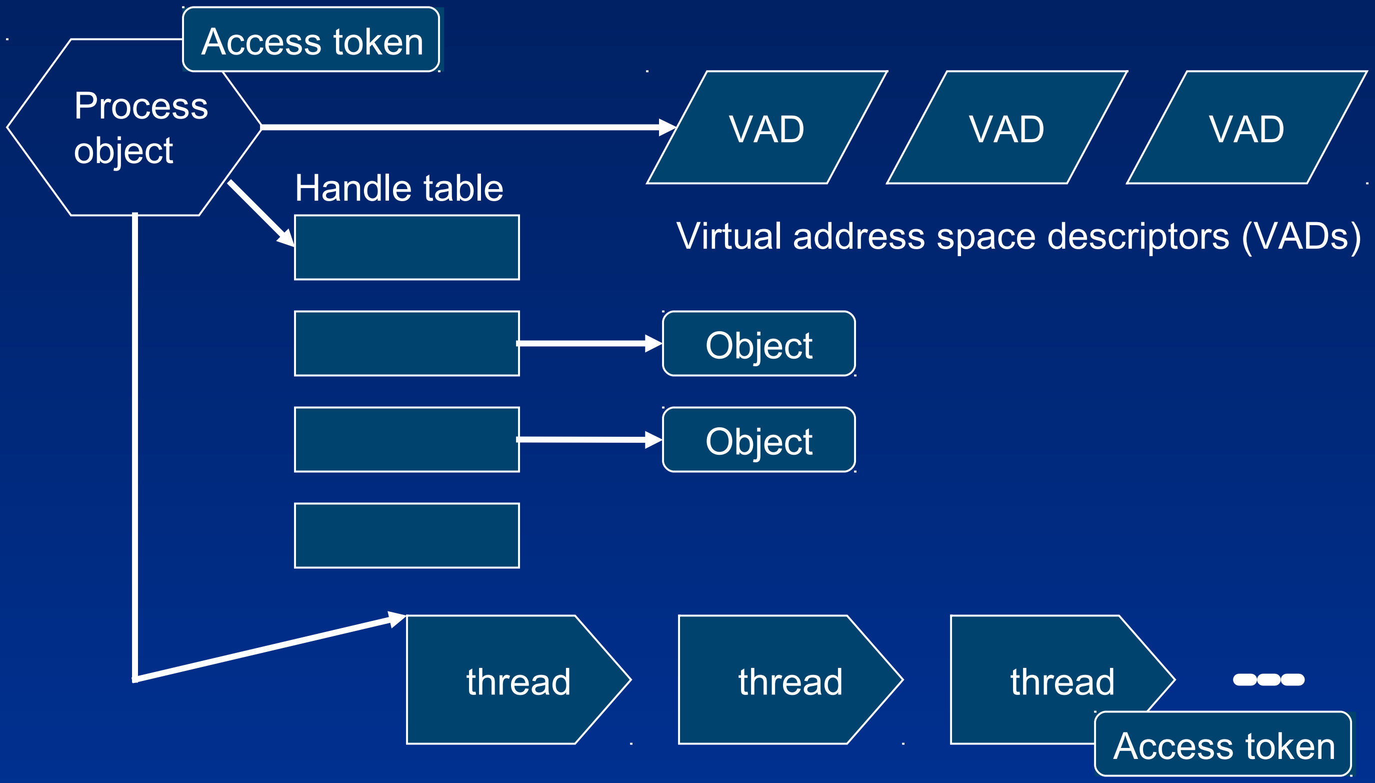
- **What is a process?**
  - Represents an instance of a running program
    - you create a process to run a program
    - starting an application creates a process
  - Process defined by:
    - Address space
    - Resources (e.g. open handles)
    - Security profile (token)
- **What is a thread?**
  - 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
- **System calls**
  - Primary argument to CreateProcess is image file name (or command line)
  - Primary argument to CreateThread is a function entry point address



# Processes & Threads

- Every process starts with one thread
  - First thread executes the program's "main" function
    - Can create other threads in the same process
    - Can create additional processes
- Why divide an application into multiple threads?
  - Perceived user responsiveness, parallel/background execution
    - Examples: Word background print – can continue to edit during print
  - Take advantage of multiple processors
    - On an MP system with n CPUs, n threads can literally run at the same time
    - Question: given a single threaded application, will adding a 2nd processor make it run faster?
  - Does add complexity
    - Synchronization
    - Scalability well is a different question…
      - # of multiple runnable threads vs # CPUs
      - Having too many runnable threads causes excess context switching

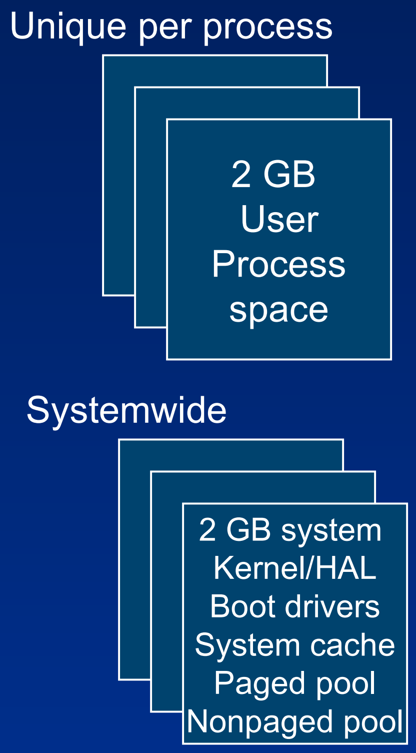
# A Process and its Resources



# Virtual Memory

- 32-bit address space (4 GB)
  - 2 GB user space (per process)
  - 2 GB operating system
- 64-bit address space
  - 7192 GB user space (Itanium)
  - 8192 GB user space (x64)
  - ~6000 GB operating system
- Memory manager maps virtual onto physical memory

**Default 32-bit layout**



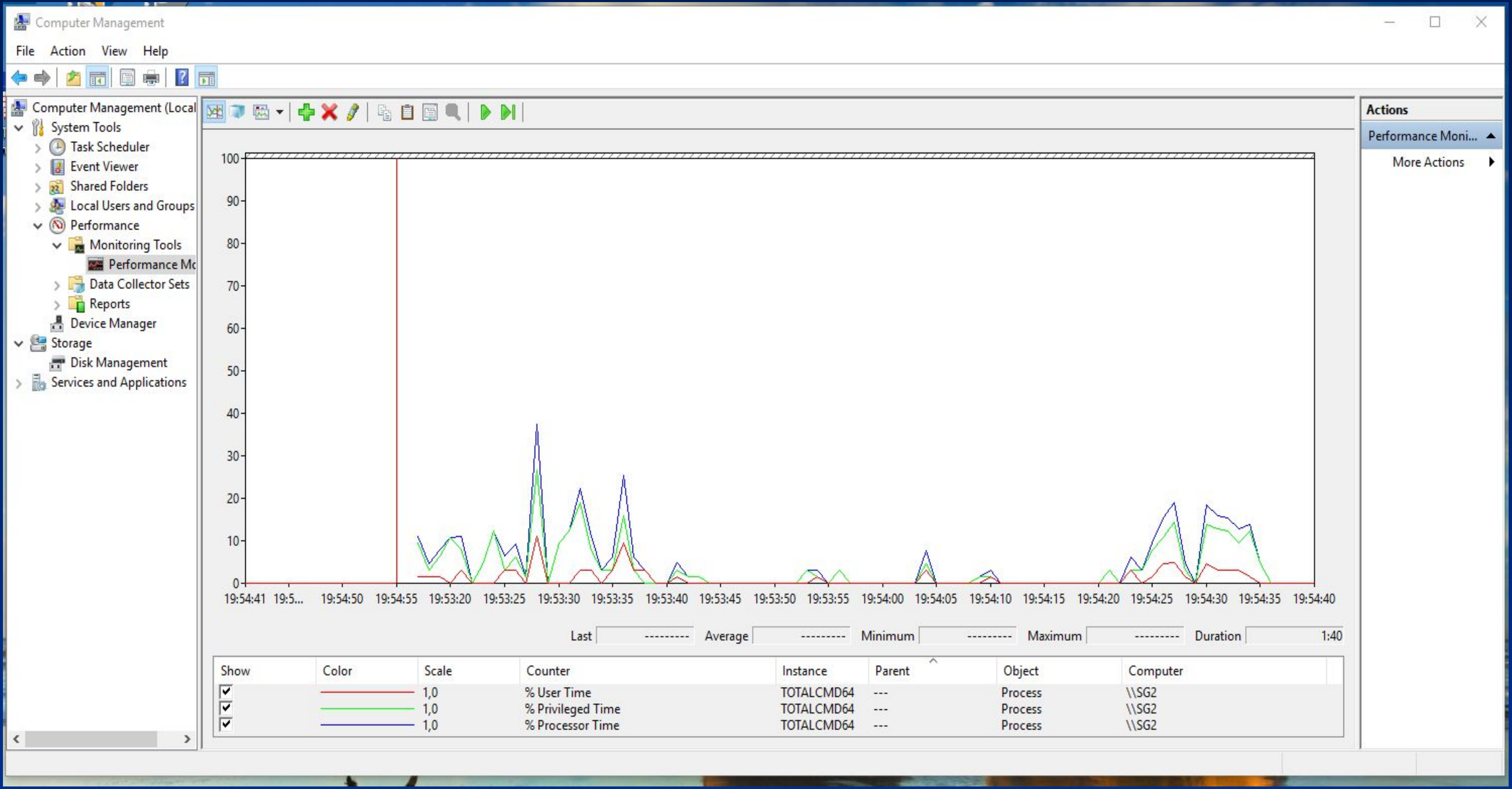
# Memory Protection Model

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

# Kernel Mode vs. User Mode

- No protection for components running in kernel mode
- Transition from user mode to kernel mode through special instruction (processor changes privilege level)
  - OS traps this instruction and validates arguments to syscalls
  - Transition from user to kernel mode does not affect thread scheduling
- Performance Counters: System/Processor/Process/ Thread – Privileged Time/User time
  - Windows kernel is thoroughly instrumented
  - Hundreds of performance counters throughout the system
- Performance Monitor – perfmon.msc – a Microsoft Management Console (MMC) snap in

# Performance Monitor



Last --------- Average --------- Minimum --------- Maximum --------- Duration 1:40

| Show | Color | Scale | Counter | Instance | Parent | Object | Computer |
|---|---|---|---|---|---|---|---|
| ☑ | 🟥 | 1,0 | % User Time | TOTALCMD64 | --- | Process | \\SG2 |
| ☑ | 🟩 | 1,0 | % Privileged Time | TOTALCMD64 | --- | Process | \\SG2 |
| ☑ | 🟦 | 1,0 | % Processor Time | TOTALCMD64 | --- | Process | \\SG2 |

# Objects and Handles

- Process, thread, file, event objects in Windows - are mapped on NT executive objects
- Object services read/write object attributes
- Objects:
  - Human-readable names for system resources
  - Resource sharing among processes
  - Resource protection against unauthorized access
- Security/Protection based on NT executive objects
- 2 forms of access control:
  - Discretionary control: read/write/access rights
  - Privileged access: administrator may take ownership of files

# Networking

- Integral, application-transparent networking services
  - Basic file and print sharing and using services
- A platform for distributed applications
  - Application-level inter-process communication (IPC)
- Windows provides an expandable platform for other network components

# Registry

- System wide software settings: boot & configuration info
- Security database
- Per-user profile settings
- In-memory volatile data (current hardware state)
  - What devices are loaded?
  - Resources used by devices
  - Performance counters are accessed through registry functions
- Regedit.exe is the tool to view/modify registry settings
  - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control
  - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
  - HKEY_LOCAL_MACHINE\Software

# Unicode

- Most internal text strings are stored/processed as 16-bit wide Unicode strings
- Windows API string functions have 2 versions
  - Unicode (wide) version
    - L“This string uses 16-bit characters“
  - ANSI (narrow) version
    - “This string uses 8-bit characters“
  - Generic character representation in Windows API
    - _T (“This string uses generic characters“)

  (Windows 9x has Windows API but no Unicode characters,
  Windows CE has Windows API but Unicode characters only)

# Tools used to dig in

- Many tools available to dig into Windows internals
  - Helps to see internals behavior “in action”
- Several sources of tools
  - Support Tools
  - Resource Kit Tools
  - Debugging Tools
  - Sysinternals.com
- Additional tool packages with internals information
  - Platform Software Development Kit (SDK)
  - Device Driver Development Kit (DDK)

# Tools for Viewing Windows Internals

| Tool | Image Name | Origin |
|---|---|---|
| Startup Programs Viewer | AUTORUNS | www.sysinternals.com |
| Dependency Walker | DEPENDS | Support Tools, Platform SDK |
| DLL List | LISTDLLS | www.sysinternals.com |
| EFS Information Dumper | EFSDUMP | www.sysinternals.com* |
| File Monitor | FILEMON | www.sysinternals.com |
| Global Flags | GFLAGS | Support Tools |
| Handle Viewer | HANDLE | www.sysinternals.com |
| Junction tool | JUNCTION | www.sysinternals.com |
| Kernel debuggers | WINDBG, KD | Debugging tools, Platform SDK, Windows DDK |
| Live Kernel Debugging | LIVEKD | www.sysinternals.com |
| Logon Sessions | LOGINSESSIONS | www.sysinternals.com |
| Object Viewer | WINOBJ | www.sysinternals.com |
| Open Handles | OH | Resource kits |
| Page Fault Monitor | PFMON | Support Tools, Resource kits, Platform SDK |
| Pending File Moves | PENDMOVES | www.sysinternals.com |

# Tools for Viewing Windows Internals (contd.)

| Tool | Image Name | Origin |
|---|---|---|
| Performance tool | PERFMON.MSC | Windows built-in tool |
| PipeList tool | PIPELIST | www.sysinternals.com |
| Pool Monitor | POOLMON | Support Tools, Windows DDK |
| Process Explorer | PROCEXP | www.sysinternals.com |
| Get SID tool | PSGETSID | www.sysinternals.com |
| Process Statistics | PSTAT | Support Tools, Windows 2000 Resource kits, Platform SDK, www.reskit.com |
| Process Viewer | PVIEWER (in the Support Tools) or PVIEW (in the Platform SDK) | Platform SDK |
| Quick Slice | QSLICE | Windows 2000 resource kits |
| Registry Monitor | REGMON | www.sysinternals.com |
| Service Control | SC | Windows XP, Platform SDK, Windows 2000 resource kits |
| Task (Process) List | TLIST | Debugging tools |
| Task Manager | TASKMGR | Windows built-in tool |
| TDImon | TDIMON | www.sysinternals.com |

# Windows Debugging Tools

- Separate package of advanced debugging tools
- Download latest version from:
  - http://www.microsoft.com/whdc/ddk/debugging
- Tools
  - User-mode and kernel-mode debuggers
    - Kd – command line interface
    - WinDbg – GUI interface (kernel debugging still mostly “command line”)
    - Allow exploring internal system state & data structures
  - Ntsd, Cdb – command line user-mode debugger (newer versions than what ships with OS)
  - Misc other tools (some are also in Support Tools):
    - kill, remote, tlist, logger/logview (API logging tool), Autodump

# Live Kernel Debugging

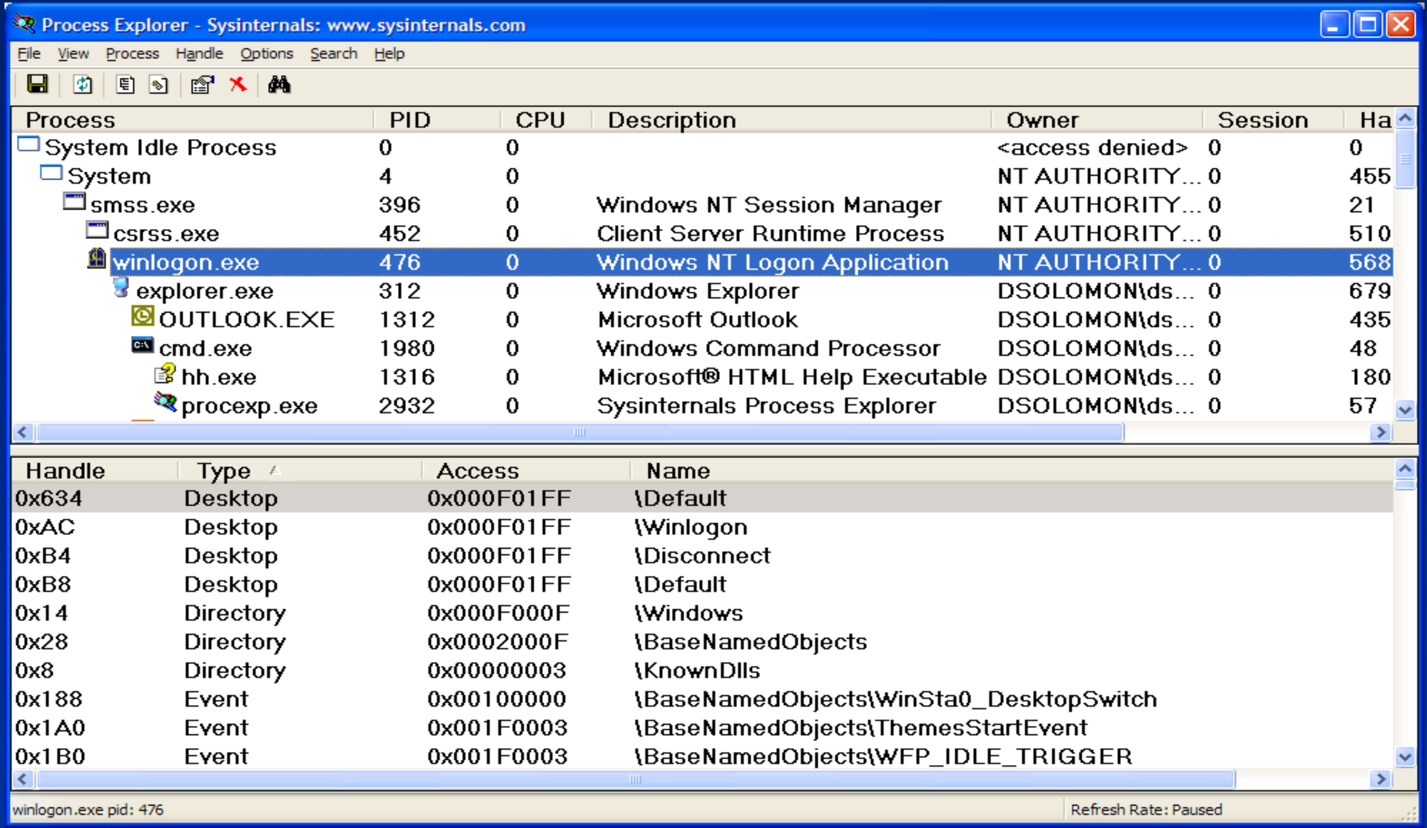
- Useful for investigating internal system state not available from other tools
  - Previously, required 2 computers (host and target)
  - Target would be halted while host debugger in use
- Starting from version XP & Srv2003, Windows NT supports live local kernel debugging
  - Technically requires system to be booted /DEBUG to work correctly
  - You can edit kernel memory on the live system (!)
  - But, not all commands work
- LiveKd (www.sysinternals.com)
  - Tricks standard Microsoft kernel debuggers into thinking they are looking at a crash dump
  - Commands that fail in local kernel debugging work in LiveKD:
    - Kernel stacks (!process, !thread)
    - Lm (list modules)
    - Can snapshot a live system (.dump)
  - Does not guarantee consistent view of system memory
    - Thus can loop or fail with access violation
    - Just quit and restart

# Sysinternals Tools

- **Freeware Windows internals tools from www.sysinternals.com**
  - Written by Mark Russinovich & Bryce Cogswell (cofounders of Winternals)
- **Useful for developers, system administrators, and power users**
  - Most popular: Filemon, Regmon, Process Explorer
- **Require no installation – run them directly after downloading and unzipping**
- **Many tools require administrative privileges**
  - Some load a device driver
- **Tools regularly updated, so make sure to check for updated versions**

# Process Explorer (Sysinternals)

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


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

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

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

winlogon.exe pid: 476

Refresh Rate: Paused

# Platform SDK
## (Software Development Kit)

- **Contains header files, libraries, documentation, & sample code for entire Windows "platform" API**
    - 14 separate SDKs
    - "Core SDK" contains core services, COM, messaging, active directory, management, etc.
- **Freely downloadable from www.microsoft.com/msdownload/platformsdk/sdkupdate**
    - Part of MSDN Professional (or higher) subscription
- **Always matches operating system revision**
    - E.g. Platform SDK revised with new release (or beta) as new APIs are added
- **Not absolutely required for Win32 development (because VC++ comes with the Win32 API header files), but…**
    - VC++ headers, libs, doc won't reflect APIs added after VC++ was mastered
- **Also provides a few tools (e.g. WinObj, Working Set Tuner) not available elsewhere**

# Further Reading

- Pavel Yosifovich, Alex Ionescu, et al., “Windows Internals”, 7th Edition, Microsoft Press, 2017.
  - Concepts and Tools (from chapter 1)
  - Digging into Windows Internals (from chapter 1)