Unit OS1:
Overview of Operating Systems

1.3. Windows Operating System Family - Concepts & Tools

Copyright Notice

- These materials are part of the *Windows Operating System Internals Curriculum Development Kit,* developed by David A. Solomon and Mark E. Russinovich with Andreas Polze
- Microsoft has licensed these materials from David Solomon Expert Seminars, Inc. for distribution to academic organizations solely for use in academic environments (and not for commercial use)

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

- 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 2000)
  - Machine-specific functions located in HAL
- Layered design:
  - architecture-specific functions located in kernel
- Windows NT/2000/XP/2003 is written in C
  - (OS executive, utilities, drivers)
  - UI and graphics subsystem - written in C++
  - HW-specific/performance-sensitive parts:
    written in assembly lang: int trap handler, context switching

Windows API & Subsystems

- Windows API (application programming interface):
  - Common programming interface to Windows NT/2000/XP, Windows 95/98/ME and Windows CE
  - OS implement (different) subsets of the API
  - MSDN: www.microsoft.com/msdn
- Windows supports multiple subsystems (APIs):
  - Win32 (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

Here we introduce key operating system concepts found in Windows 2000, such as the Win32 API, processes, threads, virtual memory, kernel mode and user mode, objects, handles, and security.

The Win32 application programming interface (API) is the primary programming interface to the Microsoft Windows operating system family, including Windows 2000/NT/XP, Windows 95/98/ME, and Windows CE. Each operating system implements a different subset of Win32. For the most part, Windows 2000 is a superset of all Win32 implementations. The Win32 API refers to the base set of functions that cover areas such as processes, threads, memory management, security, I/O, windowing and graphics.

The specifics of which services are implemented on which platform are included in the reference documentation for the Win32 API. This documentation is available at msdn.microsoft.com and on the MSDN Library CD-ROMs (MSDN stands for Microsoft Developer Network).

Although Windows 2000 was designed to support multiple programming interfaces, Win32 is the primary, or preferred to the operating system. Win32 has this position because, of the three environment subsystems (Win32, POSIX, and OS/2), it provides the greatest access to the underlying Windows 20000 system services.

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
- Only five replacement APIs!
  - Four for Window/Class Data
    - Replaced by Polymorphic (_ptr) versions
    - Updated constants used by these APIs
  - One (_ptr) version for flat scroll bars properties

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

Every application and every operating system has an abstract data model. Many applications do not explicitly expose this data model, but the model guides the way in which the application's code is written. In the 32-bit programming model (known as the ILP32 model), integer, long, and pointer data types are 32 bits in length. Most developers have used this model without realizing it. For the history of the Win32® API, this has been a valid (although not necessarily safe) assumption to make.

In 64-bit Microsoft® Windows®, this assumption of parity in data type sizes is invalid. Making all data types 64 bits in length would waste space, because most applications do not need the increased size. However, applications do need pointers to 64-bit data, and they need the ability to have 64-bit data types in selected cases. These considerations led the team to select an abstract data model called LLP64 (or P64). In the LLP64 data model, only pointers expand to 64 bits; all other basic data types (integer and long) remain 32 bits in length.

Services, Functions, and Routines

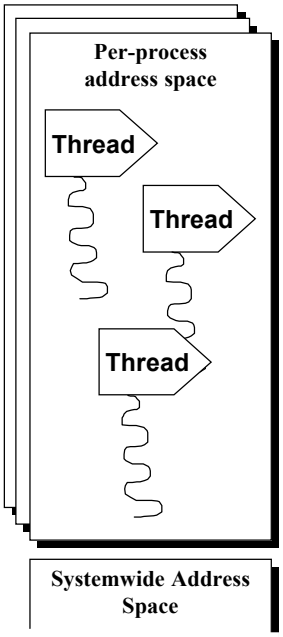
- Windows API functions:
  - Documented, callable subroutinges
  - *CreateProcess, CreateFile, GetMessage*
- Windows system services:
  - Undocumented functions, callable from user space
  - *NtCreateProcess* is used by *CreateProcess* as an internal service
- Windows internal routines:
  - Subroutines 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

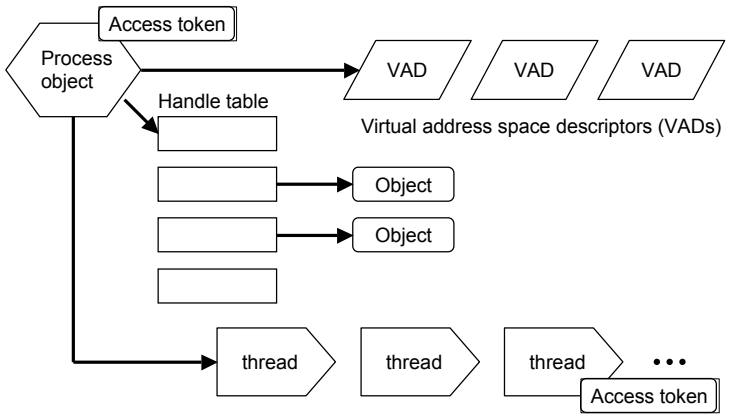


LAB: Pview

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

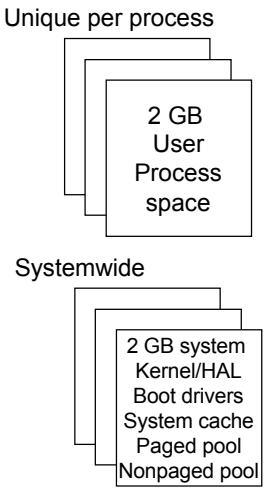
A Process and its Resources



Virtual Memory

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

Default 32-bit layout



Although 3 GB is better than 2 GB, it‘s still not enough virtual address space to map very large (multigigabyte) databases. To address this need, Windows 2000 has a new mechanism called *Address Windowing Extension* (AWE), which allows a 32-bit application to allocate up to 64 GB of physical memory and then map views, or windows, into its 2-GB virtual address space.

Using AWE puts the burden of mapping virtual to physical memory on the programmer (as in the days of MS-DOS overlays). The ultimate solution to this address space limitation is 64-bit Windows.

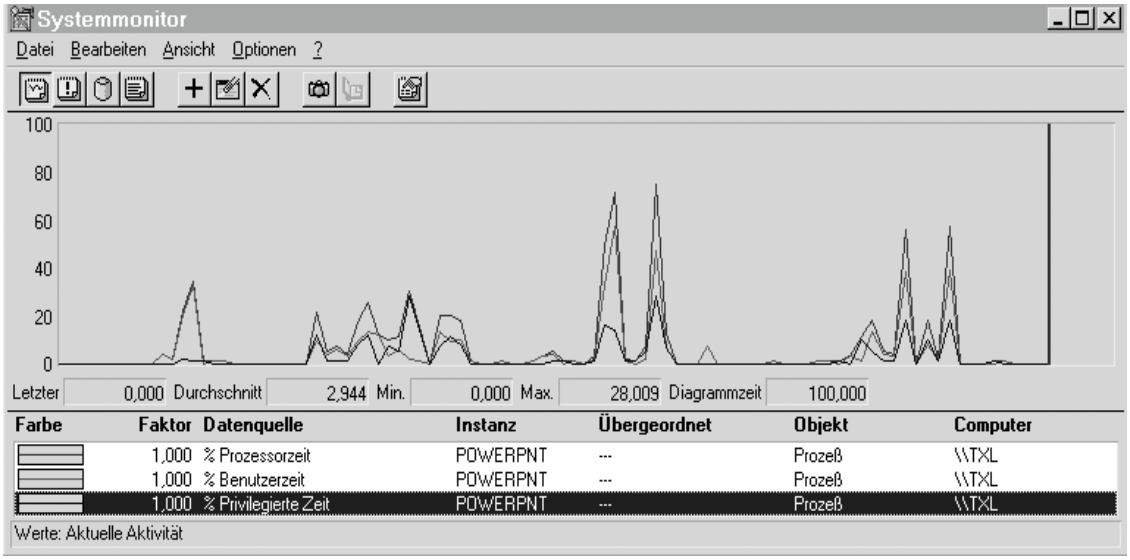
Memory Protection Model

- 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.exe

Performance Monitor



Lab:

- Run perfmon.exe, add performance counters for privileged time/user time on the thread/process/processor levels
- Run TaskManager.exe, display kernel times

Objects and Handles

- Process, thread, file, event objects in Win32 - 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

Security

- Windows 2000 supports C2-level security (DoD 5200.23-STD, December 1985)
  - Discretionary protection (need-to-know) for shareable system objects (files, directories, processes, threads)
  - Security auditing (accountability of subjects and their actions)
  - Password authentication at logon
  - Prevention of access to un-initialized resources (memory, disk space)
- Windows NT 3.51 was formally evaluated for C2
- Windows NT 4.0 SP 6a passed C2 in December 1999
  - Networked workstation configuration
- European IT Security Criteria FC2/E3 security level

Lab:

•Use Explorer to view NTFS Access rights/ownerships, ACLs

•Passwd change: CTRL-ALT-DEL (secure login sequence)

Currently, one has to consider three major versions of the Windows 2000 file system (NTFS versions NT 4.0, NT 4.0 SP 4, Win2000). This distinction becomes important in multi-boot scenarios or when using NTFS on removable media. The NTFS version used on Windows NT prior to service pack 4 cannot access media formatted with NTFS under Windows 2000. Windows 2000, on the other hand, converts older NTFS-formatted volumes into its own NTFS format when mounting a volume the first time.

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
    - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control
    - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
    - HKEY_LOCAL_MACHINE\Software
- Regedit.exe is the tool to view/modify registry settings

Besides the Windows 2000 registry editor (regedt32.exe), there exists another commonly used registry editor called regedit.exe which was originally developed for Windows 9X. Regedit.exe implements some interesting search capabilities, however, it does not handle properly the Windows 2000-specific security database contained in the registry.

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 Win32
    - _T (“This string uses generic characters“)

(Windows 95/98/ME have Win32 but no Unicode characters,
Windows CE has Win32 but only Unicode characters)

Tools for Viewing Windows Internals

Tool	Executable	Origin
Performance Monitor	PerfMon	Windows 2000
Registry Editor	RegEdt32	Windows 2000
Windows 2000 Diagnostics	WinMSD	Windows 2000
Kernel Debugger	i386kd, KD, WINDBG	Platform SDK, Windows 2000 DDK
Pool Monitor	poolmon	Windows 2000 CD \Support\Tools
Global Flags	gflags	Windows 2000 CD \Support\Tools
Open Handles	oh	Windows 2000 Resource Kits
QuickSlice	qslice	Windows 2000 Resource Kits
Process Viewer	pviewer, pview	Windows 2000 CD \Support\Tools Platform SDK
Process Explode	pview	www.reskit.com
Process Statistics	pstat	Platform SDK, www.reskit.com
Pool Monitor	poolmon	Windows 2000 CD \Support\Tools, DDK
Object Viewer	WinObj	Platform SDK, www.sysinternals.com
Page Fault Monitor	PFMon	Windows 2000 Resource Kits, Platform SDK
Service Control Tool	sc	Windows 2000 Resource Kits
Task (Process) List	tlist	Windows 2000 CD \Support\Tools

Tools used to dig in

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

Support Tools

- Tools that used to be in the NT4 Resource Kit
  - Win2K: 40+ tools, WinXP: 70+ tools
- Located on Windows OS CD in \support\tools
- Not a subset of the Resource Kit
  - So, you have to install this and the Resource Kit
    - In NT4, the NT4 Server Resource Kit included the NT4 Resource Kit Support Tools

TODO: last two bullets are misleading. Better to describe the two "support tools" packages separately.

Windows Resource Kit Tools

- Windows 2000 Server Resource Kit Tools (Supplement 1 is latest)
  - Not freely downloadable
    - Comes with MSDN & TechNet, so most sites have it
  - May be legally installed on as many PCs as you want at one site
  - Installs fine on 2000/XP Professional (superset of 2000 Professional Resource Kit)
- Windows XP Resource Kit: no tools, just documentation
- Windows Server 2003 Resource Kit Tools
  - Free download – visit http://www.microsoft.com/windows/reskits/default.asp
  - Tool updates are at http://www.microsoft.com/windowsserver2003/techinfo/reskit/tools/default.mspx
- NOTE: Windows 2000 Server Resource Kit has more tools than 2003 Resource Kit (225 vs 115 .EXEs)
  - Many tools dropped due to lack of support
  - Tools are still officially unsupported
    - But, can send bug reports to ntreskit@microsoft.com

15: revised & updated

13h: fixed what is shipped with what

15a: removed “ignore Workstation Resource Kit” point (I like hardcopy doc) -- see next slide

rev17: added FTP location of fixes

Windows Debugging Tools

- Separate package of advanced debugging tools
  - Installs on NT4, Win2000, XP, 2003
- 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

17e: added notes about Customer Diagnostics CD being downloadable

17g5: added note about new Debugger tools release (rewritten KD/WinDbg engine, new kernel debugger extensions, etc)

Live Kernel Debugging

- Useful for investigating internal system state not available from other tools
  - Previously, required 2 computers (host and target)
  - Target would be halted while host debugger in use
- XP & Server 2003 support 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
    - Works on NT4, Windows 2000, Windows XP, & Server 2003
    - Was originally shipped on Inside Windows 2000 book CD-ROM—now is free on Sysinternals
  - 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

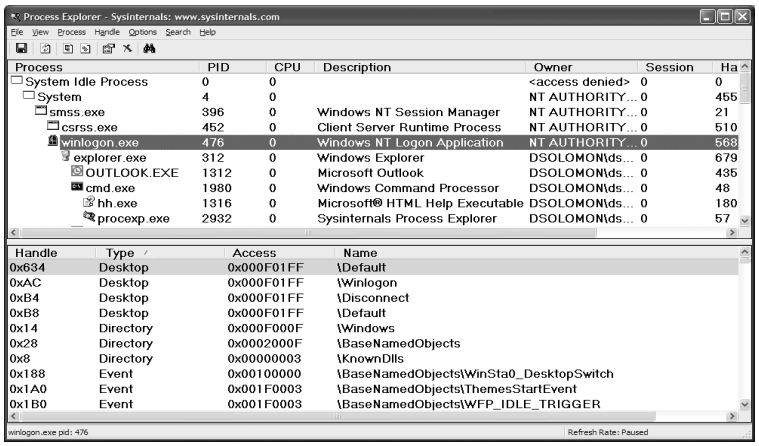
Sysinternals Tools

- **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
- **Generated via reverse engineering (no source access)**
- **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**
  - Copy of Sysinternals web site on book CD is already out of date
  - Subscribe to free Sysinternals newsletter

rev17g4: these are integral to our presentation, so removed “i”

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



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**

15a: new slide

15b: sdk tools available on the web

15e: added note that it includes latest header files that match OS (VC++ is always just a snapshot of whatever was current at the time)

15f: expanded on the above point in the slide text

rev17: SDK no longer available for free download (only to subscribers)

rev17jeh: we should put winobj, process walker, etc., on the CD we pass around.

Rev17a: agree (you and I have yet to agree on the CD contents)

17g: NOTE: for example, vc6's headers and libs definitetly do not include APIs new with win2k!

17g5: noted that SDK is freely downloadable (again)

Sources of Information

- Windows NT Resource Kits
- Platform SDK and Windows NT DDK
  - MSDN Development Platform
- Knowledge Base at www.microsoft.com
- TechNet CD-ROM edition
- Free Builds and Checked Builds
  - Kernel Debuggers
  - I386KD.EXE (command line)
  - WINDBG.EXE (GUI) with platform SDK