THE WINDOWS OPERATING SYSTEM

Zoltán Micskei
http://mit.bme.hu/~micskeiz
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).


© 2000-2005 David A. Solomon and Mark Russinovich
The history of Windows NT

- **New operating system in 1988**
  - Originally: OS/2 3.0
  - Change: Successor of Windows 3.0

- **Creator:**
  - Dave Cutler (creator of VMS at Digital)

- **Windows NT name**
  - NT = New Technology
  - Windows NT = WNT = ?
### Releases

- **Product name** ↔ **Build number**
- **Increment with every build (weekly 5-6 times)**

<table>
<thead>
<tr>
<th>Build#</th>
<th>Version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>297</td>
<td>PDC developer release</td>
<td>Jul 1992</td>
</tr>
<tr>
<td>511</td>
<td>NT 3.1</td>
<td>Jul 1993</td>
</tr>
<tr>
<td>807</td>
<td>NT 3.5</td>
<td>May 1995</td>
</tr>
<tr>
<td>1057</td>
<td>NT 3.51</td>
<td>Sep 1994</td>
</tr>
<tr>
<td>1381</td>
<td>NT 4.0</td>
<td>Jul 1996</td>
</tr>
<tr>
<td>2195</td>
<td>Windows 2000 (NT 5.0)</td>
<td>Dec 1999</td>
</tr>
<tr>
<td>2600</td>
<td>Windows XP (NT 5.1)</td>
<td>Aug 2001</td>
</tr>
<tr>
<td>3790</td>
<td>Windows Server 2003 (NT 5.2)</td>
<td>Jul 1999</td>
</tr>
<tr>
<td>6000</td>
<td>Windows Vista RTM</td>
<td>Nov 2003</td>
</tr>
<tr>
<td>9600</td>
<td>Windows 8.1 RTM</td>
<td>Aug 2013</td>
</tr>
</tbody>
</table>

- **Features**
  - 6 developers at the beginning
  - 200 dev, 140 testers at end
  - 6M LOC
  - Whole compiling: 5 hours
  - 1400 dev, 1700 testers
  - 29M LOC
  - 50 GB source code
  - Compiling: 8 hours
  - Stress test: 1000 machines
Windows and Linux

- **1970**: VMS v1.0
- **1970**: UNIX born
- **1980**: Windows NT 3.1
- **1980**: UNIX public
- **1990**: Windows 2000
- **1990**: UNIX V6
- **1990**: NT 4.0
- **2000**: Windows XP
- **2000**: Linux v1.0
- **2000**: v2.0
- **2000**: v2.2
- **2000**: v2.3
- **2000**: v2.4
- **2000**: v2.6
THE STRUCTURE OF WINDOWS
What does smss.exe do on my machine??

What is WoW?
Design goals

- **Portability**
  - Multiple processor architectures:
    - Originally: Intel x86, MIPS, Alpha, PowerPC
    - Windows XP: Intel x86
    - Windows Server 2003: x86, x64, IA64 (Itanium)
    - Windows 8: x86, x64, ARM (?)
  - HW specific part separated
  - Kernel: written in C
Design goals

- Portability
- Extensibility
  - Modular design
  - Well-defined interfaces
  - Unicode (even in kernel)
Design goals

- Portability
- Extensibility
- Reliability
  - Windows 3.0: shared address space
  - Security standards
Design goals

- Portability
- Extensibility
- Reliability
- Performance
  - 32 bit, preemptive, *multi-threaded*, *reentrant*
  - Symmetric Multiprocessing (SMP)
  - Asynchronous I/O
  - Optimized for client-server applications
Design goals

- Portability
- Extensibility
- Reliability
- Performance
- Compatibility, support for
  - DOS and 16 bit Windows API
  - POSIX
  - OS/2
Multiple personalities

- How to support Win32, POSIX and OS/2 API?
- Solution: environment subsystem

Application 1

- Windows API
- Windows subsystem

Application 2

- POSIX API
- POSIX subsystem

NT API
- NT Kernel
Which subsystem do they belong?
- cmd.exe
- notepad.exe
- smss.exe
Simplified architecture

- **System processes**
- **Services**
- **Applications**
- **Subsystem processes**

**User Mode**

**Subsystem DLLs**

**Kernel Mode**

- **Executive**
  - **Kernel**
  - **Device drivers**
  - **Hardware Abstraction Layer (HAL)**

**Windowing, graphics**
Simplified architecture

System processes
Services
Applications
Subsystem processes

Subsystem DLLs

User Mode
Kernel Mode

Executive

Kernel
Device drivers

Hardware Abstraction Layer (HAL)

Windowing, graphics

- Hiding HW details
- Unified interface
- hal.dll
Simplified architecture

- System processes
- Services
- Applications
- Subsystem processes

User Mode

Kernel Mode

Executive

- Kernel modules
- Layered structure
- Network, file system, I/O devices
- *.sys

Windowing, graphics

Kernel

Device drivers

Hardware Abstraction Layer (HAL)
Operating Systems

Simplified architecture

- System processes
- Services
- Applications
- Subsystem processes

User Mode
- DLLs
- Executive
- Basic services
- Interrupts, scheduling
- Synchronization
- ntoskrnl.exe

Kernel Mode
- Windowing, graphics
- Hardware Abstraction Layer (HAL)
- Kernel
- Device drivers
Simplified architecture

System processes

Services
- Complex OS services
- Memory, process handling
- Object oriented
- Security, I/O
- Also in ntoskrnl.exe

Applications

Subsystem processes

User Mode

Kernel Mode

Executive

Kernel

Device drivers

Hardware Abstraction Layer (HAL)

Windowing, graphics

Operating Systems
Simplified architecture

- Processor Access Mode
- CPU support
- Protecting
  - kernel from the user processes
  - user processes from each other

User Mode

Kernel Mode

Executive

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Device drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Abstraction Layer (HAL)</td>
<td></td>
</tr>
</tbody>
</table>

Windowing, graphics
Task Manager

Performance counters

Time spent in user and kernel mode
Simplified architecture

- Basic system functionality
- Initialization, authentication, logon
- These are started first

System processes

Subsystem processes

Subsystem DLLs

User Mode

Kernel Mode

Executive

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Device drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Abstraction Layer (HAL)</td>
<td></td>
</tr>
</tbody>
</table>

Windowing, graphics
Simplified architecture

- **System processes**
- **Services**
  - Processes running in the background
  - Like UNIX daemons
  - E.g.: DNS client, Indexing, RPC...
- **Subsystem DLLs**
- **Executive**
  - Kernel
  - Device drivers
  - Hardware Abstraction Layer (HAL)
- **Windowing, graphics**

User Mode

Kernel Mode
Simplified architecture

System processes

• Managing user mode processes
• Offering APIs
• Windows: csrss.exe

Subsytem processes

Subsystem DLLs

User Mode

Kernel Mode

Executive

Kernel

Device drivers

Hardware Abstraction Layer (HAL)

Windowing, graphics
Simplified architecture

System processes

Applications

Subsystem processes

Subsystem DLLs

User Mode

Kernel Mode

Executive

Kernel

Device drivers

Hardware Abstraction Layer (HAL)

Windowing, graphics

• Translating subsystem API calls to calls to the Executive
  • E.g. Windows: kernel32.dll
  • ReadFile() -> NtReadFile()
Simplified architecture

- **System processes**
- **Services**
- **Applications**
- **Subsystem**

**User Mode**

**Kernel Mode**

**Executive**

**Device drivers**

**Hardware Abstraction Layer (HAL)**

**Windowing, graphics**

- Is it good?
- Is it faster?
- Is it more instable?

- GUI running in kernel mode (performance)
- Windowing, drawing
- Graphics drivers
- Win32k.sys
- Documented kernel calls in the Windows DDK
- Documented Windows API calls in the Windows SDK
- List of services
Calling a Windows Kernel functions

Own application
- call ReadFile(...)
  - return to caller

ReadFile in Kernel32.Dll
- call NtReadFile
  - return to caller
  - Windows subsystem specific

NtReadFile in Ntdll.Dll
- Int 2E or SYSCALL or SYSENTER
  - return to caller
  - All subsystems

Software interrupt
- KiSystemService in NtosKrnl.Exe
  - call NtReadFile
    - dismiss interrupt
    - System Service Dispatcher

NtReadFile in NtosKrnl.Exe
- do the operation
  - return to caller

Operating Systems
DEMO

Tracing calls:
application → kernel32.dll → ntdll.dll

- Dependency walker
- WinDbg debugger

- Windows API function
  - E.g. ReadFile
  - documented in the SDK

- System services
  - Functions of the Executive callable from user mode

- Windows internal functions
  - Callable only from kernel mode
The Windows kernel

- Monolithic or microkernel?
- Shows mikrokernel-like properties
  - Only minimal functionality in the kernel
  - Kernel only callable on well-defined interfaces
  - Part of the OS runs in user mode
- However
  - Protected components run in one address space
- (other names)
(Windows 8: WinRT)

- One more layer
- Support for Metro / Immersive apps
Windows on ARM (WOA)

- Separate product
  - ~ consumer device

- WOA (and Windows 8): new design goals
  - thin and light design
  - long battery life
  - integrated quality
Not so simplified architecture

Operating Systems

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

Original copyright by Microsoft Corporation.
Not so simplified architecture

- Session Manager: handling sessions
- Wininit: starting system processes
- Winlogon: login, Ctrl+Alt+Del
- LSASS: Local Security Authentication SubSystem, security
- Service Control Manager: starting, stopping services

System processes:
- Service Control Mgr.
- LSASS
- WinLogon
- Session Manager
- Service
- Control Mgr.
- LSASS
- WinMgt.Exe
- SpoolSv.Exe
- SvcHost.Exe
- Explorer

Original copyright by Microsoft Corporation.
Not so simplified architecture

- >50
- Run without user logon
- SvcHost: general service host process

hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)
Not so simplified architecture

- Most programs use the OS through subsystem DLLs

System Service Dispatcher

I/O Mgr

Device & File Sys. Drivers

File Sys. Cache

System Threads

WinLogon

Session Manager

SpoolSv.exe

SvcHost.exe

System processes

Applications

Task Manager

Explorer

User Application

Subsystem DLLs

WinMgt.exe

SpoolSv.exe

POSIX (SUA)

Subsystems

Windows

Windows DLLs

POSIX (SUA)

Windows DLLs

Subsystem DLLs

System Service Dispatcher

(kernel mode callable interfaces)

I/O Mgr

File Sys. Cache

Device & File Sys. Drivers

Hardware Abstraction Layer (HAL)

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

Operating Systems

Original copyright by Microsoft Corporation.
Not so simplified architecture

- Stubs for the functions in Executive
- Internal functions for the subsystems

Operating Systems

Hardware Abstraction Layer (HAL)

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

Original copyright by Microsoft Corporation.
Not so simplified architecture

- Trapping system calls
- Checking parameters
- Calling to the specific component

System processes

Services

Applications

Task Manager

User Application

Subsystem DLLs

POSIX (SUA)

Windows DLLs

Windows

POSIX (SUA)

Windows DLLs

System Service Dispatcher

NTDLL.DLL

User Mode

System Threads

Kernel Mode

I/O Mgr

Device & File Sys. Drivers

File System Cache

Object

Plug and Play Mgr.

Power Mgr.

Security Reference Monitor

Virtual Memory

Processes & Threads

Configuration Mgr (registry)

Local Procedure Call

Graphics Drivers

Windows USER, GDI

Kernel

Hardware Abstraction Layer (HAL)

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

Original copyright by Microsoft Corporation.
Not so simplified architecture

- Threads for kernel and device drivers
- Task which wait
- These also can be preempted!
- “System” process in Task Manager
- E.g. zero page thread
Not so simplified architecture

Components of the Executive

I/O Mgr
- File System Cache
- Device & File Sys. Drivers
- Device & File Sys. Drivers
- Plug and Play Mgr.
- Object Mgr.
- Power Mgr.

Power Mgr.

Security Reference Monitor

Virtual Memory

Processes & Threads

Configuration Mgr (registry)

Local Procedure Call

Windows USER, GDI

Graphics Drivers

Kernel

Hardware Abstraction Layer (HAL)

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

Original copyright by Microsoft Corporation.
Base OS components:
- NTOSKRNL.EXE: Executive and kernel
- HAL.DLL: Hardware abstraction layer
- NTDLL.DLL: Stubs for the Executive

System processes:
- SMSS.EXE: Session manager process
- WINLOGON.EXE: Logon process
- SERVICES.EXE: Service controller process
- LSASS.EXE: Local Security Authority Subsystem

Windows subsystem, GUI:
- CSRSS.EXE: Windows subsystem process
- WIN32K.SYS: USER and GDI kernel-mode components
- KERNEL32/USER32/GDI32.DLL: Windows subsystem DLLs
Windows versions

- The same source scales from
  - 1 CPU, 1 GB memory (Windows Vista Starter)
  - 64 CPU, 2 TB memory (Windows Server 2008 Datacenter Edition)

- Depending on settings in the registry:
  - Server or client
  - Type of server

- Differences
  - Defaults values for scheduling, memory mgmt
  - Licensing limits
Tools to dig in..

- **Windows SDK**
  - Successor of the Platform SDK, .NET Framework SDK
  - C/C++ headers, API description, compiles

- **Windows Driver Kit**
  - Successor of the Windows DDK
  - C/C++ headers, documentation, static verifiers

- **Windows Debugging Tools**
  - User and kernel mode debugger

- **Sysinternals**
  - Company of Mark Russinovich (MS bought it)
  - Process Explorer, Filemon, liveKd...

- **Windows Support Tools, Windows Resource Kit**

- ...
To read

  - Everything about Windows