Kernel Basics and Registry: Section 1 Transcript

Introduction

Welcome to the Kernel Basics and Registry module. During this module, we'll focus on the fundamental concepts of the Windows Kernel, the basic structure of the Windows Registry, and the Command-Line Registry Editor. Throughout this module, you'll be presented with opportunities to assess and apply what you've learned.

At the end of this module, you will be able to:

- Define the various functions of the Windows Kernel,
- Describe the differences between User-mode vs. Kernel-mode,
- Describe the structure and contents of the Windows Registry,
- Describe how LastKnownGood is updated,
- Demonstrate the use of the Command-Line Registry Editor to view, analyze, modify, and create Registry entries, and
- Identify when changes to the Registry are expected to take effect.

Bypass Exam Introduction

If you are already familiar with the subject matter presented in this module, you can choose to take a Bypass Exam to skip this module.

The Bypass Exam option provides a single opportunity to successfully demonstrate your competence with the material presented within the module. If you pass, you'll receive credit for completing the module, unlocking the content within, and you will be free to proceed to the next module. If you do not pass, you will need to successfully complete the module, including all exercises and the Module Exam, to receive credit.

Click the Next Section button to continue.

Kernel Basics and Registry: Section 2 Transcript

The Windows Kernel 1/5

Let's begin this module by discussing the Windows Kernel. The kernel's role in operating systems, such as UNIX and Windows, is to control various elements of the machine. The kernel sits between individual running programs and the hardware itself performing various critical functions for the operating system and acting as a liaison between user-level programs and the hardware.

The Windows Kernel 2/5

The kernel is a set of functions, usually contained in a file written as an embedded computer program. It manages system resources, communicating between hardware and software by translating input/output (I/O) requests between applications and I/O devices, such as the Central Processing Unit (CPU) and memory. The kernel is also responsible for initializing device drivers at boot-up.

Kernel mode drivers exist in three levels:

- Highest-level drivers,
- Intermediate-level drivers, and
- Lowest-level drivers.

Click each driver level to learn more.

Highest-level drivers

Highest-level drivers include File System Drivers (FSDs) that support file systems, such as:

- New Technology File System (NTFS)
- File Allocation Table (FAT)
- CD-ROM file system (CDFS)

Highest-level drivers always depend on support from underlying lower-level drivers, such as intermediate-level function drivers and lowest-level hardware bus drivers.

Intermediate-level drivers

Intermediate-level drivers include virtual disk, mirror, Windows Driver Model, or device-type-specific class and depend on support from underlying lower-level drivers. Intermediate-level drivers are subdivided further as follows:

- Function drivers control specific peripheral devices on an I/O bus
- Filter drivers insert themselves above or below function drivers
- Software bus drivers present a set of child devices to which still higher-level class, function, or filter drivers can attach themselves

Lowest-level drivers

Lowest-level drivers control an I/O bus to which peripheral devices are connected. Lowest-level drivers do not depend on lower-level drivers. Hardware bus drivers are system-supplied and usually control dynamically configurable I/O buses.

Hardware bus drivers work with the Plug and Play manager to configure and reconfigure system hardware resources, for all child devices that are connected to the I/O buses that the driver controls. These hardware resources include mappings for device memory and interrupt requests (IRQs).

Legacy drivers that directly control a physical device are lowest-level drivers.

Kernel Functions

The kernel is a fundamental part of the computer's operating system that connects the application software to the hardware of a computer.

Typically, the kernel is responsible for:

- Process and task management,
- Memory management, and
- Device management.

Click each kernel function tab to learn more.

Process and Task Management:The kernel manages the computer's hardware and resources and allows other programs to run and use these resources. One of these resources is the CPU. This is the most central part of a computer system, responsible for running or executing programs. The kernel takes responsibility for deciding, at any time, which of the many running programs should be allocated to the processor or processors, each of which can usually run only one program at a time.

Kernels also provide methods for synchronization and communication between processes called inter-process communication (IPC). A kernel may implement these features itself, or rely on some of the processes it runs to provide the facilities to other processes, although in this case it must provide some means of IPC to allow processes to access the facilities provided by each other. Finally, a kernel must provide running programs with a method to make requests to access these facilities.

Memory Management: The kernel code is usually loaded into a protected area of memory to prevent it from being overwritten by programs or other parts of the operating system. Memory is used to store both program instructions and data. Typically, both need to be present in memory in order for a program to execute. Often, multiple programs will want access to memory, frequently demanding more memory than the computer has available. The kernel is responsible for deciding which memory each process can use, and determining what to do when not enough is available.

For example, if a program needs data which is not currently in RAM, the CPU signals to the kernel that this has happened, and the kernel responds by writing the contents of an inactive memory block to disk and replacing it with the data requested by the program. The program can then be resumed from the point where it was stopped.

Virtual addressing also allows creation of virtual partitions of memory in two disjointed areas, one being reserved for the kernel space and the other for the applications or user space. The applications are not permitted by the processor to address kernel memory, thus preventing an application from damaging the running kernel.

Device Management: To perform useful functions, processes need access to the peripherals connected to the computer, which are controlled by the kernel through device drivers. These peripherals can include the keyboard, mouse, disk drives, USB devices, printers, displays, and network adapters.

For example, to show the user something on the screen, an application would make a request to the kernel, which would forward the request to its display driver, which is then responsible for actually plotting the character or pixel. A kernel must maintain a list of available devices.

This list may be known in advance, configured by the user, or detected by the operating system at run time, normally called plug and play. In a plug and play system, the device manager first scans the different hardware buses, such as the Peripheral Component Interconnect (PCI) or Universal Serial Bus (USB), to detect installed devices, and then searches for the appropriate drivers.

User-mode and Kernel-mode

The NT operating system protects system data from modification and/or corruption from user applications by implementing two processor access modes known as User-mode and Kernel-mode.

User-mode is a less privileged processor than Kernel-mode. Applications, environment subsystems, and integral subsystems execute in User-mode. Code executing in User-mode cannot damage the integrity of the operating system.

Kernel-mode is a higher privileged processor than User-mode and has direct access to all hardware and memory. I/O operations and other system services run in Kernel-mode.

Components that run in Kernel-mode include:

- Executive subsystems,
- Kernel,
- Device drivers,
- Hardware Abstraction Layer (HAL), and
- Windows USER and graphics display interface (GDI) functions.

Click Resources for more information about the Windows kernel.

Section Completed

Kernel Basics and Registry: Section 3 Transcript

The Registry

The Registry data is read during the:

- Initial boot process,
- Kernel boot process,
- Logon process, and
- Application startup.

During the initial boot process, the boot loader reads the list of boot drivers to load into the memory before initializing the kernel. Then, during the kernel boot process, the kernel reads settings that specify which device drivers to load and how various subsystems configure themselves and tune system performance. During the logon process, the Explorer and other Windows components read per-user preferences.

Finally, during the application startup, applications read system-wide settings as well as per-user preference settings which include menu and toolbar placement, fonts, and most-recently accessed documents.

The Registry Structure

The registry is a hierarchical database that contains keys and values. When compared to a file system, registry keys act like folders and values act like files.

The top-level key is referred to as a root key. There are five major root keys in the registry database. Each root key contains one or more subkeys and subkeys can have one or more subkeys - again, similar to a file system. Finally, values contain the actual data and are not considered organizational units.

Values contain three information types:

- Value Name
- Value Data Type
- Value Data

The final term related to the registry is a hive. A hive is a discreet set of keys, subkeys, and values contained in the registry. Each hive is stored in a single file in the %SystemRoot%\system32\config directory, along with an associated .LOG file.

Registry Organization

The five root keys that make up the registry are:

- \HKEY_CLASSES_ROOT, which stores file association and Component Object Model (COM) object registration information,
- \HKEY_LOCAL_MACHINE, which stores system-related information,
- \HKEY_CURRENT_USER, which stores data associated with the currently logged-on user,
- \HKEY_USERS, which stores information about all the accounts on the machine, and
- \HKEY_CURRENT_CONFIG, which stores some information about the current hardware profile.

That said, there are really only two root keys since the others are subsets or pointers for:

- \HKEY_LOCAL_MACHINE, and
- \HKEY_USERS

Within the root key, the 'H' stands for handle. Registry keys are accessed from a known root key handle mapped to the content of a registry key that is preloaded by the kernel from a stored hive.

Click Resources for information related to Windows Registry root Keys.

Registry Organization

The image shown here provides a visual representation of the linked hives and where they reside.

Here are some examples:

- HKLM\System\CurrentControlSet is actually an alias to one of the other control sets: ControlSet001, ControlSet002, etc.,
- HKCR is an alias to HKLM\Software\Classes and HKU\<SID>\Classes,
- HKCU is an alias to the currently logged-on user's profile under HKU, and
- HKCC is an alias to HKLM\System\CurrentControlSet\Hardware Profiles\Current.

Hives are links and not copies. The information needs to appear in two different locations but rather than having two copies to maintain, links that point to the original location are used-a link is also known as an alias. Users cannot create an alias in the registry - this is Microsoft-only created.

Registry Organization

Hives have the following size limits:

- On 32-bit systems, the limit is 50 percent of physical memory, up to 400 megabytes (MB),
- On 64-bit systems, the limit is 50 percent of physical memory, up to 1.5 gigabytes (GB), and
- On Itanium systems, the limit is 50 percent of physical memory, up to 1 gigabyte (GB).

Hive limits are imposed since the hive is loaded during the boot process before paging is enabled.

Click Resources for more information on Registry storage space.

Registry Organization

HKLM is the root key that contains all system-wide configuration subkeys.

- HKLM\BCD (Vista +),

- HKLM\COMPONENTS (Vista +),
- HKLM\HARDWARE,
- HKLM\SAM,
- HKLM\SECURITY,
- HKLM\SOFTWARE, and
- HKLM\SYSTEM.

Click Resources for additional information related to the HKLM root key.

Registry Nomenclature

One of the subkeys in the HKEY_LOCAL_MACHINE root key is HKEY_LOCAL_MACHINE\SYSTEM\Select.

When Windows boots normally, it looks at HKEY_LOCAL_MACHINE\SYSTEM\Select.

Under this key are four REG_DWORD values: Current, Default, Failed, and LastKnownGood.

Each of these numbers reference a ControlSet00x key under HKEY_LOCAL_MACHINE\SYSTEM:

- Current is the one used to boot the system this time unless it differs from Default, indicating a system change was made before the last restart.
- Default is the one to use for the next boot when LastKnownGood needs updating.
- Failed is the last one which was Current when "LastKnownGood" was selected at boot.
- LastKnownGood indicates the one which was known to let the system boot correctly.

LastKnownGood is a Windows startup option that uses the most recent system settings that worked correctly. Every time you turn your computer off and Windows shuts down successfully, important system settings are saved in the registry. If a problem occurs, you can restart your computer using those settings. For example, if a new driver for your video card is causing problems, or an incorrect registry setting is preventing Windows from starting correctly, you can restart your computer using LastKnownGood Configuration.

Registry Data Types

Every registry entry has a data type which represents the specific type of data the entry can store. Knowing the various data types provides an indicator of the kind of data to expect such as Binary, DWORD, or a String, which is another term for text, and how long those strings will be; fixed, variable, 32-bit, or 64-bit.

The most common data types seen in the registry are REG_SZ and REG_DWORD.

Click Resources for additional information related to Registry Data Types.

Section Completed

Kernel Basics and Registry: Section 4 Transcript

Console Registry Tools

There are several methods for modifying the registry but we'll discuss Regedit, which is the graphical user interface tool, or GUI, and reg.exe, also known as Reg, which is the command-line interface tool (CLI).

Regedit is good for learning the names of the registry keys and the proper syntax to be used in the CLI tool. Regfind is a CLI tool developed by Windows that can be used to easily search for and find registry keys. The source of Regfind is from the Windows 2000 resource kit but it can be found online as well.

Regedit and Reg.exe are capable of modifying local and remote machine registries; however, only two registry keys are available for remote changes: HKLM and HKU.

Remote registry changes are made possible by the existence of the Remote Registry Service as well as proper administrative credentials.

Remote Registry Service is ON by default in Win NT and 2000, but is OFF by default in XP and in later versions.

Console Registry Tools

The Reg.exe command allows you to:

- Add or delete data or keys,
- Import or export hives and keys,
- Load and unload keys to troubleshoot or conduct analysis, and
- Compare keys for differences.

Let's review the proper syntax for the CLI:

- reg add keyname[/v Valuename][/t Type][/d Data]

Using that syntax, here's an example:

- reg add HKLM\Software\MyCo /v Data /t REG_BINARY /d fe340ead

In the example, within the registry, the valuename "Data" is created under the subkey "HKLM\Software\MyCo", and the binary data "fe340ead" is stored in this value. The use of this data is dependent upon the application that reads it.

Registry Changes

When changes are made to the registry, they take effect at different times depending on the changes

that are made.

Changes to user settings take effect the next time a user logs on, changes to application settings take effect the next time an application starts, and changes to system settings take effect during the next system start or boot.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Kernel Basics and Registry: Section 5 Transcript

Summary

You have completed the Kernel Basics and Registry module.

During this module, we discussed the Windows kernel and structure of the Windows Registry. Additionally, we discussed and provided opportunities to interface and modify the registry using command-line interface tools.

You should now be able to:

- Define the various functions of the Windows Kernel,
- Describe the differences between User-mode vs. Kernel-mode,
- Describe the structure and contents of the Windows Registry,
- Describe how LastKnownGood is updated,
- Demonstrate the use of the Command-Line Registry editor to view, analyze, modify, and create Registry entries, and
- Identify when changes to the Registry are expected to take effect.

To receive credit and advance to the next module, you must achieve a passing score on the Module Exam. Click the Next Section button to begin the Module Exam.