

# Inside NT Utilities

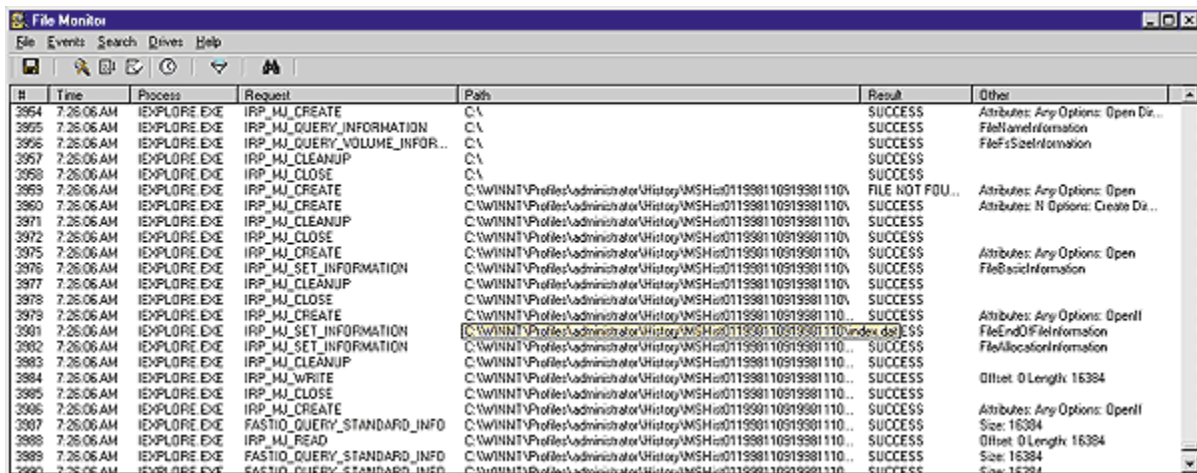
Mark Russinovich

(Reprinted from WindowsItPro Magazine)

Since I began exploring the internal structure and functioning of Windows NT, I've discovered ways to improve the OS's performance and to fill holes in its built-in functionality. As I've delved ever deeper into NT's internals, I've developed monitoring tools, performance-related tools, and recovery tools. This month, I'll acquaint you with a few of these powerful utilities, which you can use to learn more about NT's inner workings and perhaps even to troubleshoot problems you encounter. I'll describe what these utilities do and how you can use them to solve problems, and I'll briefly discuss how they work internally. The tools are Filemon, a file-system monitor; Regmon, a Registry monitor; HandleEx, a DLL and open-handle viewer; NTFSDOS, an NTFS file system driver; and NewSID, a security ID (SID)-changing tool. All the tools are available for free download from Systems Internals (<http://www.sysinternals.com>), the Web site I developed in collaboration with Bryce Cogswell. Many of these tools include full source code.

## Filemon

Have you ever wondered what files applications are accessing when you hear your disk drive start up during otherwise idle periods? Maybe you've tried to run an application and received a vague error message that the application didn't install properly or is unable to find a required file. Filemon for NT lets you see precisely which files and directories applications are accessing or trying to access. Filemon monitors file system activity on logical drives (e.g., C, D) you specify, including 3.5" disk drives, hard disk drives, CD-ROM drives, and even network drives. Monitoring begins when you launch Filemon, and each file system access that takes place while Filemon is monitoring displays on a separate line in Filemon's output window, as Screen 1, page 56 shows. You can use the scroll bars to navigate forward and backward through Filemon. You can also use toolbar buttons or menu commands to save a trace to a file, start and stop monitoring, select drives that you want Filemon to monitor, and cancel the selection of drives.



#	Time	Process	Request	Path	Result	Other
3864	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CREATE	C:\	SUCCESS	Attributes: Any Options: Open Dir...
3865	7:26:06 AM	EXPLORE.EXE	IRP_MJ_QUERY_INFORMATION	C:\	SUCCESS	FileEndOfFileInformation
3866	7:26:06 AM	EXPLORE.EXE	IRP_MJ_QUERY_VOLUME_INFOR...	C:\	SUCCESS	FileFsSetInformation
3867	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLEANUP	C:\	SUCCESS	
3868	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLOSE	C:\	SUCCESS	
3869	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CREATE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	FILE NOT FOU...	Attributes: Any Options: Open
3869	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CREATE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Attributes: N Options: Create Da...
3870	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLEANUP	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3871	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLEANUP	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3872	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLOSE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3873	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CREATE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Attributes: Any Options: Open
3874	7:26:06 AM	EXPLORE.EXE	IRP_MJ_SET_INFORMATION	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	FileBasicInformation
3875	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLEANUP	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3876	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLEANUP	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3877	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLOSE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3878	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLOSE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3879	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CREATE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Attributes: Any Options: Operil
3880	7:26:06 AM	EXPLORE.EXE	IRP_MJ_SET_INFORMATION	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	FileEndOfFileInformation
3881	7:26:06 AM	EXPLORE.EXE	IRP_MJ_SET_INFORMATION	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	FileAllocationInformation
3882	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLEANUP	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3883	7:26:06 AM	EXPLORE.EXE	IRP_MJ_WRITE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Offset: 0 Length: 16384
3884	7:26:06 AM	EXPLORE.EXE	IRP_MJ_WRITE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3885	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CLOSE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	
3886	7:26:06 AM	EXPLORE.EXE	IRP_MJ_CREATE	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Attributes: Any Options: Operil
3887	7:26:06 AM	EXPLORE.EXE	FASTIO_QUERY_STANDARD_INFO	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Size: 16384
3888	7:26:06 AM	EXPLORE.EXE	IRP_MJ_READ	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Offset: 0 Length: 16384
3889	7:26:06 AM	EXPLORE.EXE	FASTIO_QUERY_STANDARD_INFO	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Size: 16384
3890	7:26:06 AM	EXPLORE.EXE	FASTIO_QUERY_STANDARD_INFO	C:\WINNT\Profiles\Administrator\History\MSHed011998110919981110...	SUCCESS	Size: 16384

The Filemon display divides each access record into fields. Filemon's fields include the sequence number assigned to a record, timing information related to the access, name of the process that performed the access, full path of the accessed file or directory, result of the access, and supplementary information related to the access. As accesses take place, Filemon assigns them unique sequence numbers. These sequence numbers help you to navigate through records when you scroll up and down in the display and let you know when the rate of file system activity causes Filemon's internal buffers to overflow. Filemon drops records when its buffers are full (a rare occurrence); a gap in sequence numbers in the output signals such an event.

## Inside NT Utilities

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

Filemon's timing features let you specify whether you want to see the time at which file accesses occur or the duration of an access. Measuring duration is helpful if you're developing a program for which file system performance is an important factor in overall performance. For example, to improve Excel's performance, Microsoft used Filemon to compare the time Excel 2000 spent executing file I/O across versions with different optimizations.

Filemon's Result field can provide insight when you're trying to determine why an application won't start or is misbehaving. Most requests will display a status of SUCCESS, which means the file system driver successfully carried out the operation. FILE NOT FOUND and NO SUCH FILE status codes can be red flags that identify which files and directories an application expects to find but doesn't. However, many times, what appear to be errors in Filemon's output are expected behavior. For example, when you type the name of a program at a command prompt, the system will search through the directories in your PATH environment variable for the program's file. If your command prompt isn't in the directory that contains the file, typing the program's name at the prompt will result in NT trying and failing to open the file in various PATH directories until it finds the directory in which the file resides.

Filemon's Other field displays information specific to different file system requests. For read and write operations, Filemon informs you of the offset into the file that an application read from or wrote to and the length of the access. Lock requests show you the range of the file an application locks, and file attributes print for requests that query or set a file's attributes.

Filemon has powerful advanced filtering capabilities. You can configure filters that result in the logging of accesses particular processes make or of accesses that refer to only specific files or directories. You can use Filemon filtering to isolate file system activity that is related to files or applications you're troubleshooting or learning about. When you use Filemon, you might be surprised to discover the presence of applications on your system that constantly perform file I/O. One example is antivirus programs that check their virus definition files one or more times per second for new information. Filemon remembers filters across executions, so you can command Filemon to ignore monotonous background activity, such as antivirus-program polling.

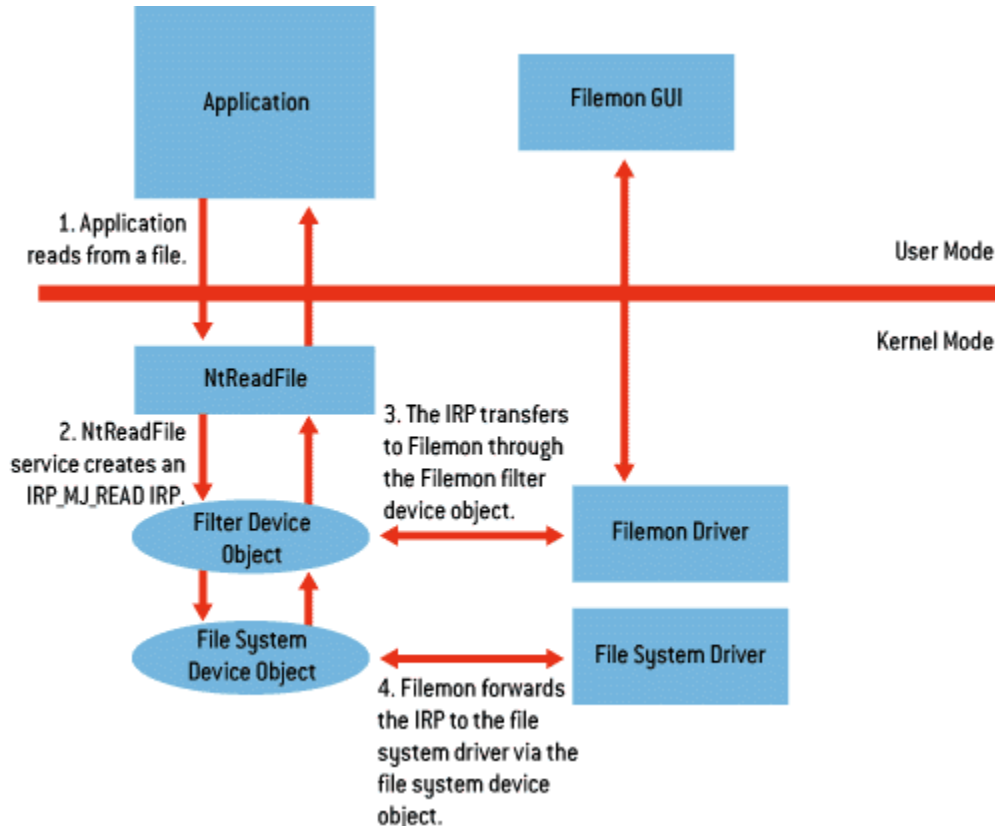
I based Filemon on a type of device driver called a filter driver, which I described in a previous column (see "Inside On-Access Virus Scanners," September 1997). NT describes file I/O with discrete command packets called I/O request packets (IRPs). NT's file-related services, which have names such as NtReadFile and NtCreateFile, create IRPs. When an application performs a file access, the application indirectly invokes one or more file-related services, and the I/O Manager delivers the resulting IRPs to the file system driver responsible for managing the target file. The I/O Manager locates the file system driver through one of the driver's device objects that serves as a logical representation of the file on which the driver resides.

NT's I/O model lets filter drivers create device objects that attach to other device objects. The I/O Manager can route IRPs (which the I/O Manager would otherwise send to the device driver associated with the underlying object) to the filter driver that owns the object that has attached to the target device object. Thus, the I/O Manager hands to the filter driver IRPs aimed at the object the driver is filtering. When you direct Filemon to watch a certain drive, Filemon creates a filter device object and attaches it to the drive's file system device object. From that point, Filemon receives and inspects all the IRPs directed at the drive, records information about the IRPs in memory buffers, then passes the IRPs on to the file system driver. For each IRP it sees, Filemon registers with the I/O Manager a request to see the IRP again after the file system driver has finished servicing it. Filemon can then record what time an operation completes and the operation's status. The Filemon user

# Inside NT Utilities

Mark Russinovich  
(Reprinted from WindowsItPro Magazine)

interface (UI) periodically requests the latest batch of activity buffers from the driver and displays the records the buffers contain in the Filemon output window. Figure 1 shows the relationships between applications, device objects, IRPs, Filemon's filter driver, and file system drivers.



Because Filemon intercepts IRPs, you see commands such as IRP\_MJ\_READ and IRP\_MJ\_CREATE in Filemon's output window, rather than more familiar commands, such as Read and Open. In Filemon's output windows, you'll also see non-IRP command names such as FASTIO\_READ and FASTIO\_QUERY\_BASIC\_INFORMATION. These commands represent the Fast I/O shortcut path that NT uses to speak with file system drivers. This path is a shortcut because it bypasses the creation of IRPs, and with the path, the I/O Manager directly invokes functions within file system drivers to carry out requests. You can see both IRP-related and Fast I/O commands because file system drivers can service Fast I/O requests only if the accessed file data is in the file system cache and several other conditions hold. If a file system driver returns a Fast I/O command status of FALSE, the I/O Manager falls back on the IRP-based request path.

A pathname you'll frequently see is DASD. This pathname stands for Direct Access Storage Device and is the name Microsoft uses to describe I/O that bypasses the file system data structures. This pathname occurs so frequently because Filemon shows the requests that file systems make to update their on-disk data structures, such as FAT's File Allocation Table or NTFS's Master File Table (MFT). Because file systems aim these requests at files with no explicit name, Filemon uses DASD.

Describing the meaning of every command type and status code is outside the scope of this article, but the important command types and status codes are self-explanatory. Full source code to Filemon is on the Systems Internals Web site.

# Inside NT Utilities

Mark Russinovich  
(Reprinted from WindowsItPro Magazine)

## Regmon

Regmon is a Registry monitoring tool with a UI, which Screen 2 shows, that is similar to that of Filemon. When you run Regmon, the tool immediately begins monitoring and logging all Registry activity on your system. As with Filemon, Regmon's output window is divided into different fields. Regmon assigns sequence numbers to access records, and these numbers serve as navigation aids and dropped-record indicators, just as sequence numbers serve in Filemon. Regmon displays the name of each process that performs a Registry access, the full path name of the key or value the process accessed, the result code of the status, and supplemental information specific to the type of access. To save screen space, Regmon relies on the common abbreviations for root keys that Table 1 lists. Regmon includes the same filtering capabilities Filemon includes, letting you zoom in on the behavior of a particular application or on accesses to particular Registry keys or values.

#	Process	Request	Path	Result	Other
3248	explorer.exe	OpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	Key: 0xE1EE1A40
3250	explorer.exe	QueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\EnforceShellExtensionSecurity	NOTFOUND	
3251	explorer.exe	CloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	Key: 0xE1EE1A40
3252	explorer.exe	QueryValue	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32\ThreadingModel	SUCCESS	"Apartment"
3253	explorer.exe	CloseKey	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32	SUCCESS	Key: 0xE11AD540
3254	explorer.exe	OpenKey	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32	SUCCESS	Key: 0xE11AD540
3255	explorer.exe	QueryValue	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32(Default)	SUCCESS	"Shell32.dll"
3256	explorer.exe	OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	NOTFOUND	
3257	explorer.exe	OpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	Key: 0xE1EE1A40
3258	explorer.exe	QueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\EnforceShellExtensionSecurity	NOTFOUND	
3259	explorer.exe	CloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	Key: 0xE1EE1A40
3260	explorer.exe	QueryValue	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32\ThreadingModel	SUCCESS	"Apartment"
3261	explorer.exe	CloseKey	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32	SUCCESS	Key: 0xE11AD540
3262	explorer.exe	OpenKey	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32	SUCCESS	Key: 0xE11AD540
3263	explorer.exe	QueryValue	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32(Default)	SUCCESS	"Shell32.dll"
3264	explorer.exe	OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	NOTFOUND	
3265	explorer.exe	OpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	NOTFOUND	Key: 0xE1EE1A40
3266	explorer.exe	QueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\EnforceShellExtensionSecurity	NOTFOUND	
3267	explorer.exe	CloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	Key: 0xE1EE1A40
3268	explorer.exe	QueryValue	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32\ThreadingModel	SUCCESS	"Apartment"
3269	explorer.exe	CloseKey	HKCR\CLSID\{20004FED-3AEA-1069-A2D8-08002B30309D}\InProcServer32	SUCCESS	Key: 0xE11AD540

In its Other column, Regmon prints the data values that programs read from and write to the Registry. If a value is a string data type, you'll see the entire string in the Other column; otherwise, you'll see up to 8 bytes of the value, presented as hexadecimal digits.

Because the Registry is the nerve center for NT and applications, Regmon provides an inside look at how NT and applications use configuration parameters. You can start an application while Regmon is watching and see precisely what settings the application obtains from the Registry and where those settings are located. This view is helpful for determining why an application might fail to start or might behave in unexpected ways, and the view can also be useful for discovering undocumented application parameters.

Regmon aids in tracking down problems in production software, and it is a powerful tool for debugging software under development. Component object model (COM) applications rely heavily on Registry COM class-id settings under HKEY\_CLASSES\_ROOT, in which Regmon can identify problems. Microsoft found Regmon useful as a debugging aid during the development of Word 2000.

As with file system status codes, Registry status codes that appear to be errors might instead be expected results. Status codes you'll commonly see in Regmon traces are NOTFOUND and BUFOVRFLOW. Applications, particularly NT Explorer, often check for the existence of optional configuration settings. If the settings aren't present, attempted accesses will yield an error signaling that the application didn't find the value or key, but such an error doesn't prevent the application from behaving correctly. When an application reads a Registry value, the application often dynamically allocates buffers that are just big enough to hold the value. To determine how large a buffer must be, the application passes a buffer of length zero in its first access of the value. The result of the

## Inside NT Utilities

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

operation (if the value exists) is an error specifying that the buffer is too small (the buffer has overflowed), and additional information will tell the application how large a buffer must be to hold the value. The application can then allocate a buffer of the necessary size and reattempt the access.

At some point, you might want to edit a value or key you see referenced in Regmon's output. Regmon makes this task easy—simply double-click on the line with the name of the value or key you're interested in. Regmon launches the regedit Registry Editor, opening it to the specific value or key. You can achieve the same result by using the Jump to Regedit toolbar button.

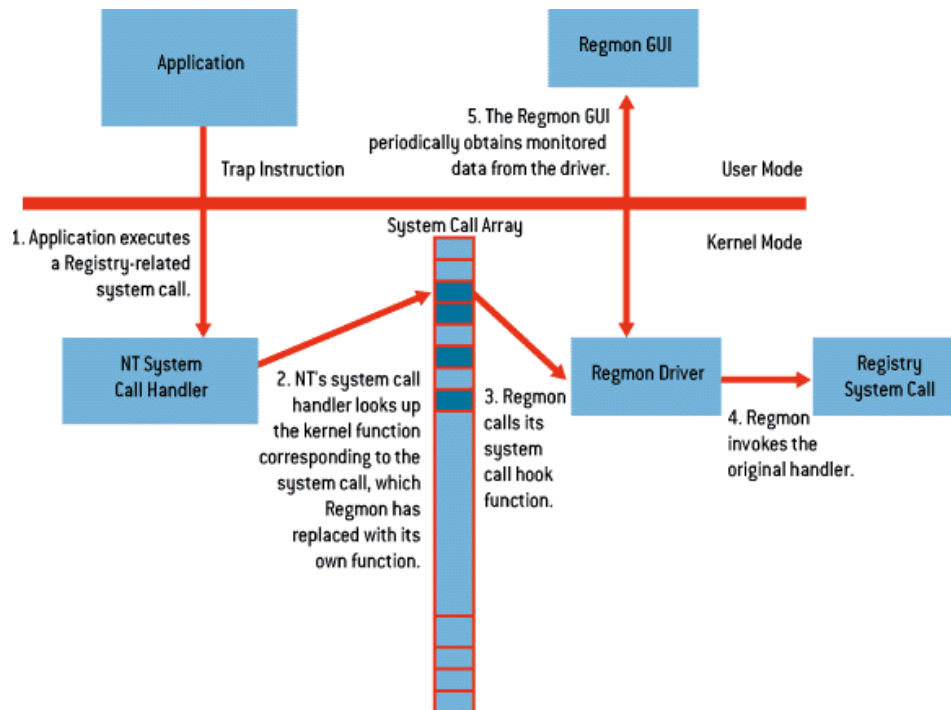
In addition to displaying monitored accesses in realtime in its GUI, Regmon has a mode you can configure to begin logging accesses to a file from early in the system boot. This capability lets you obtain a complete log of all Registry accesses that take place on a system during an entire boot-to-shutdown cycle. Logs that Regmon generates reveal NT and driver misconfiguration problems and undocumented Registry parameters.

Although the similarity between Regmon's and Filemon's interfaces and operations might imply that Regmon relies on a filter driver similar to Filemon's, internally, Regmon's device driver is different from that of Filemon. Regmon has a device driver component, but Regmon's driver uses an unsupported technique I developed to monitor Registry accesses. I call this technique system call hooking. Understanding system call hooking requires some background in NT's system call interface. When an application performs a Registry-related operation, the application indirectly or directly invokes Registry system calls. When an NT application invokes a system call, the application loads a CPU register with the index number of the call, then carries out a special instruction that causes the CPU to switch from user mode to supervisor mode. During the mode change, the CPU transfers control to a function in the NT kernel that reads the system call index and finds the address of the corresponding system call routine by reading the index's entry in a system call array. The kernel then carries out the system call, after which control returns to the application.

Regmon can have its functions run in lieu of the original system call functions simply by inserting pointers to its functions in NT's system call array. After Regmon examines the parameters an application passes to a system call, Regmon invokes the system call it replaced. When control returns to Regmon from the system call, Regmon can examine the return status and return control to the application. Figure 2 demonstrates this process. System call indexes can change between different releases of NT, but the method Regmon uses to determine indexes is version-independent. To learn more about system call hooking, examine Regmon's source code on the Systems Internals Web site.

# Inside NT Utilities

Mark Russinovich  
(Reprinted from WindowsItPro Magazine)



## HandleEx

Have you ever tried to delete a directory and received an Access denied error message for no apparent reason? Perhaps you've experienced a situation in which a particular application runs well on one system but crashes on another, and you're sure that the failure is the result of a DLL versioning problem. You can easily investigate both these problems, in addition to other problems, by using HandleEx. HandleEx is a utility that gives you a list of all the processes executing on a system. Screen 3 shows HandleEx's double-paned UI.

Process	PID	Owner	Priority	Handles
System	2	<unable to open token>	8	280
smss.exe	1A	NT AUTHORITY:SYSTEM	11	30
csrss.exe	22	NT AUTHORITY:SYSTEM	13	312
winlogon.exe	28	NT AUTHORITY:SYSTEM	13	54
services.exe	2E	NT AUTHORITY:SYSTEM	9	229
lsass.exe	31	NT AUTHORITY:SYSTEM	9	96
nddeagnt.exe	4E	DUAL\Administrator	8	16
explorer.exe	52	DUAL\Administrator	8	189
spoolss.exe	58	NT AUTHORITY:SYSTEM	8	74
jsdaemon.exe	5F	NT AUTHORITY:SYSTEM	8	109
llssrv.exe	6B	NT AUTHORITY:SYSTEM	9	73
mgasc.exe	6E	NT AUTHORITY:SYSTEM	8	16
RpcSs.exe	70	NT AUTHORITY:SYSTEM	8	135
mgactl.exe	72	NT AUTHORITY:SYSTEM	8	13
tapisrv.exe	7D	NT AUTHORITY:SYSTEM	8	90
rasman.exe	85	NT AUTHORITY:SYSTEM	8	165

Handle	Type	Name
4	Section	C:\WINNT\system32\spoolss.exe
C	ObjDirectory	\KnownDlls
10	ObjDirectory	\Windows
14	File	C:\WINNT\system32\
24	WinStation	\Windows\WindowStations\WinSta0
2C	WinStation	\Windows\WindowStations\WinSta0
3C	Key	HKLM
54	File	\Device\NamedPipe\nt\ControlPipe
64	File	\Device\NamedPipe\spoolss
68	File	\Device\NamedPipe\spoolss
70	File	\Device\NamedPipe\spoolss
74	File	\Device\NamedPipe\svcdl
80	Port	\RPC Control\spoolss
94	Key	HKLM\System\ControlSet001\Control\Print\Printers
9C	Key	HKU\Default
A4	File	\Device\NamedPipe\EVENTLOG

spoolss.exe pid: 58

# Inside NT Utilities

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

When you click a process in HandleEx's upper pane, the information about the process displays in the lower pane depending on HandleEx's mode of operation. You can use a toolbar button or menu command to choose HandleEx's mode—handle or DLL. When HandleEx is in handle mode, you'll see a list of handles the executing process has open. When a process opens a file, Registry key, synchronization object, or other system resource, NT creates a reference to the resource in an array, called its handle table, that is associated with the process. NT returns the resource entry's index number, called its handle, to the process, so that the process can use the handle to refer to the resource. You can set HandleEx to list handles that specify resources that have names, or to list all of a process' handles, named and unnamed.

HandleEx in DLL mode displays the list of DLLs that a process has loaded. Information HandleEx displays about the DLLs includes the address at which the DLL is loaded in the process' address map, the size of the loaded DLL image, the version number of the DLL, and the full pathname of the DLL's file.

Regmon Abbreviation	Registry Key
HKCC	HKEY_CURRENT_CONFIG
HKCR	HKEY_CLASSES_ROOT
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKU	<>HKEY_USERS

When you receive an Access denied error message when trying to delete a directory, the error might have occurred because the directory contains one or more files that are open. Such a deletion can also fail because you have a command prompt—either on the local machine or a remote system—for which the directory you want to delete is the current directory. HandleEx has a search feature that lets you quickly see whether either of these cases are the cause of an Access denied message. To use the feature, you enter part of the directory's name in HandleEx's Find dialog box, and HandleEx will return the names of all files and directories that are open and that match the name you typed in, in addition to the name of the process that opened the file or directory. DLL version mismatches are similarly easy to track. Save the list of DLLs that a properly working application loads on a system, and compare that list to the list from a system on which the application is misbehaving. You can look for version number differences between the lists to find the version mismatches.

HandleEx can also reveal instances of another common application bug: the handle leak. Handle leaks occur when a program opens a resource but fails to close the resource when the program is finished referencing it. Leaks are visible in HandleEx as an abnormally high number of handles in the process that is leaking handles. I used HandleEx to find and report a handle leak in one of NT's device drivers, a problem Microsoft corrected in Service Pack 4 (SP4).

HandleEx uses undocumented functions both to obtain the list of DLLs a process loads and to query the handles the process opens. (HandleEx's source code isn't posted on the Systems Internals Web site.) However, an application can determine a great deal of process- and thread-related information by using documented Performance APIs such as the ones Performance Monitor uses. Microsoft's

# Inside NT Utilities

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

Win32 software development kit (SDK) includes the full source code to PerfMon. In addition, the ToolHelp32 API, documents functions for enumerating loaded DLLs, is making its appearance in Windows 2000 (Win2K—formerly NT 5.0). Command-line equivalents of HandleEx, ListDLLs, and Handle are available for download at the Systems Internals Web site.

## NTFSDOS

The most popular utility on the Systems Internals Web site is NTFSDOS. NTFSDOS is a file system driver that provides read-only access to NTFS drives from DOS, Windows 9x, and Windows 3.x. NTFSDOS is popular for two reasons. The first is that NTFSDOS provides a means for accessing files on the NTFS drives of systems that won't boot. Booting a system from a standard DOS boot disk will give you access to files on a system's FAT drives, but NTFS drives are inaccessible from outside NT without a third-party driver. After you can access the NTFS files, you can salvage the files by copying them from the nonbootable system.

The second reason for NTFSDOS's popularity is that the utility allows for the sharing of files and applications between NT and Windows 9x or Windows 3.x in a dual-boot environment. NTFS drives are invisible to Windows 9x and Windows 3.x, but NTFSDOS makes the NTFS drives appear to these systems as standard—although read-only—drives.

NTFSDOS has prompted Microsoft to add an encryption facility to NTFS in Win2K. Because NTFSDOS doesn't honor NTFS file security, you can boot a system off an NTFSDOS boot disk and access otherwise secure NTFS files on the system's hard disk (see "NTFSDOS Poses Little Security Risk," September 1996, to learn why NTFSDOS doesn't exploit holes in NT security). Microsoft is introducing Encrypting File System (EFS), which prevents NTFSDOS from reading sensitive files.

NTFSDOS's connection with NT internals occurs only through the tool's interpretation of NTFS on-disk data structures. Otherwise, NTFSDOS is a DOS terminate-and-stay-resident (TSR) program that hooks DOS's network file system callouts to interface DOS (and Windows) to NTFS volumes. Source code to NTFSDOS isn't available, but you can learn about NTFS's on-disk structures from my January 1998 column, "Inside NTFS," or by studying the source code to a read-only NTFS file system driver for Linux that is available through <http://www.informatik.hu-berlin.de/~loewis>.

## NewSID

In my June 1998 article, "NT Rollout Options," I describe a security problem that cloning NT installations can cause. Cloning is a popular technique for quickly rolling out identical NT system configurations to multiple workstations throughout an enterprise. The security problem arises from the fact that SIDs of local accounts derive from the SID of the computer on which the accounts reside. The NT setup process assigns a computer's SID, and if you clone a system from a computer whose SID is already assigned, the clone will have the original computer's SID. Therefore, local accounts on cloned systems will have identical SIDs, resulting in the inability of NT's security mechanism to distinguish between the users of two cloned computers. Users associated with accounts on cloned computers can access files and other resources belonging to other users. The solution to this problem is to change the SID of a cloned computer, which also changes the SIDs of the computer's accounts.

Several companies have developed SID changers to accompany their disk-cloning software, and Bryce and I have developed a SID-changing tool, NewSID, which QuarterDeck Software distributes as part of its system cloning utility. We supply NewSID on the Systems Internals Web site with full source code, which reveals how the utility finds and updates a computer's SID, and the way it updates all references to a particular system's SID. References that refer to a local account exist in any NTFS file

## Inside NT Utilities

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

and in Registry security settings; therefore, NewSID must traverse every Registry key and NTFS file on a computer to assign a new SID.

Also, you can use NewSID to move NT 4.0 Backup Domain Controllers (BDCs) between domains. Domain controllers from the same domain share the same computer SID, so to move a controller to a new domain, you simply give the BDC the SID of the new domain and let the domain synchronize the BDC. You can tell NewSID to copy a SID from another computer instead of randomly generating a SID, so that it matches that of a PDC or BDC.

### Useful Tools

I hope you'll find the tools I've described valuable additions to your NT toolkit, helping you solve problems and learn more about the way NT works. You'll find additional similar tools at the Systems Internals Web site.