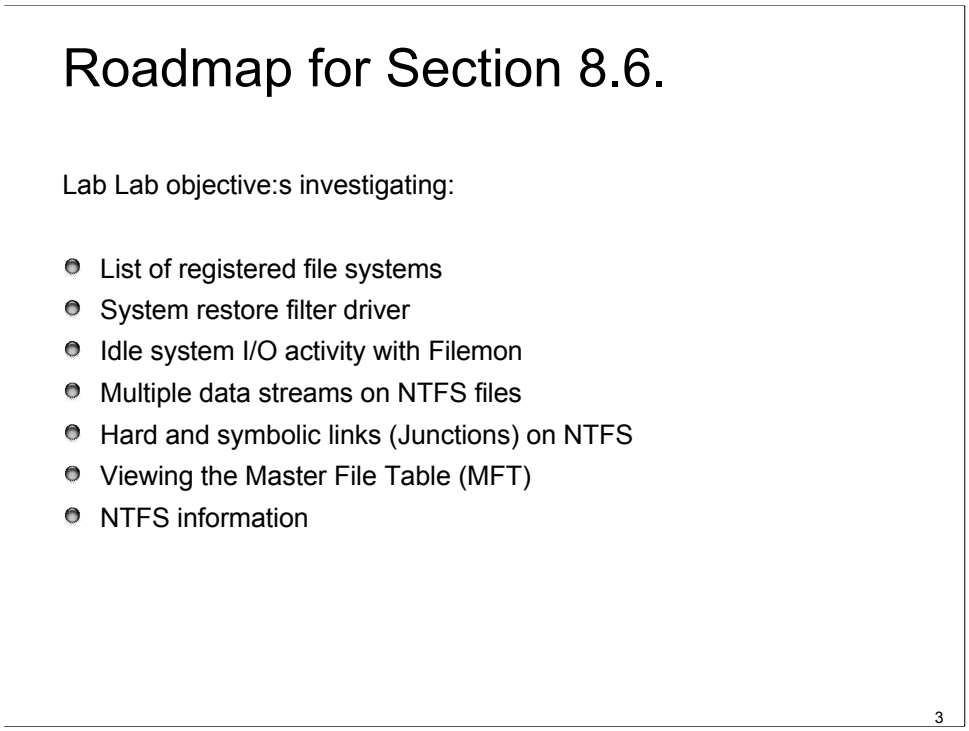
Unit OS8: File System

8.6. Lab Manual

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)

2



This LabManual includes Lab objective:s investigating the the file system mechanisms and concepts implemented inside the Windows operating system. Students are expected to carry out Labs in addition to studying the learning materials in Unit OS8.

A thorough understanding of the concepts presented in Unit OS8: File System is a prerequisite for these Labs.

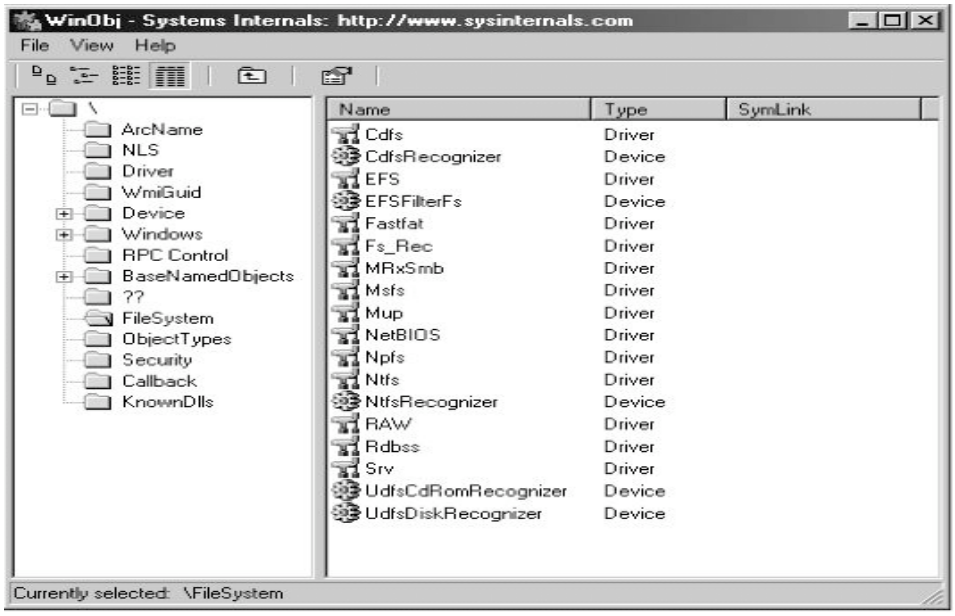
List of Registered File Systems Lab

- When I/O manager loads driver, it typically names driver object according to file system
- Not all driver objects of type file system driver represent local/remote file systems
  - I.e.; Npfs (Named Pipe File System) is a network API driver
- WinObj and the System Information viewer reveal list of registered file systems
  - (MMC snap-in on W2K, Msinfo32 on Server 2003)

4

Lab objective: Viewing the List of Registered File Systems

When the I/O manager loads a device driver into memory, it typically names the driver object it creates to represent the driver so that it's placed in the \Drivers object manager directory. The driver objects for any driver the I/O manager loads that have a Type attribute value of SERVICE_FILE_SYSTEM_DRIVER (2) are placed in the \FileSystem directory by the I/O manager. Thus, using a tool such as Winobj (from www.sysinternals.com), you can see the file systems that have registered on a system, as shown in the following screen shot. (Note that some file system drivers also place device objects in the \FileSystem directory.)



System Restore Lab

- System Restore provides a way to restore a Windows XP system to a previously known point
  - Not available on Windows 2000 or Server 2003
- XP-compatible Setup may create a “restore point” before installation begins
  - Restore works on per-volume basis
- System restore filter driver attaches filter device objects to FAT and NTFS objects (volumes)
  - Platform SDK provides SRSetRestorePoint and SRRemoveRestorePoint APIs for installation programs
  - Lab investigates restore filter driver objects using kernel debugger

5

Lab objective: Looking at System Restore Filter Device Objects

To monitor changes to files and directories, the System Restore filter driver must attach filter device objects to the FAT and NTFS device objects representing volumes. In addition, it attaches a filter device object to the device objects representing file system drivers so that it can become aware of new volumes as they are mounted by the file system and then subsequently attach filter device objects to them. You can see System Restore’s device objects with a kernel debugger:

```
lkd> !drvobj\filesystem\sr
Driverobject(81543850) is for:
  \FileSystem\sr
DriverExtensionList: (id ,addr)
DeviceObjectlist:
 814ee370 81542dd0 81543728
```

In this sample output, the System Restore driver has three device objects. The first and second objects are attached to NTFS file system device objects. The last one in the list is named SystemRestore, so it serves as the interface to which the user-mode components of System Restore direct commands. Let us further examine the first device object:

```
lkd>!devobj814ee370
Device object (814ee370) is for:
  \FileSystem\sr DriverObject 81543850
Current Irp 00000000 RefCount 0 Type 00000008 Flags 00000000
DevExt 814ee428 DevObjExt 814ee570
ExtensionFlags (0x80000000) DOE_DESIGNATED_FDO
AttachedTo (Lower) 81532020 \FileSystem\Ntfs
Device queue is not busy.
```

Filemon Idle System Lab

- Filemon shows all file activity as it occurs
  - ideal tool for troubleshooting file system–related system and application failures
  - Filemon requires Load Driver and Debug privileges
- Basic mode vs. advanced mode
  - I/O operations (IRPs) are tagged with friendly names
  - Access to NTFS metadata, paging I/O, System and filemon process activity, fast I/O failures are reported only in advanced mode
- Lab uses filemon to examine file system activity on idle system

6

Lab objective: Viewing File System Activity on an Idle System

Windows file system drivers implement support for file change notification, which enables applications to request notification of file system changes without polling for them. The Windows functions for doing so include ReadDirectoryChangesW() and the FindFirstChangeNotification(), FindNextChangeNofification() pair.

When you run Filemon on a system that's idle, you should therefore not see the repeated accesses to files or directories because that activity unnecessarily negatively affects a system's overall performance. Run Filemon, and after several seconds examine the output log to see whether you can spot polling behavior. Right-click on an output line associated with polling and choose Process Properties from the context menu to view details of the process performing the activity.

Filemon App Error Lab

- Applications sometimes present error messages in response to an error condition that do not reveal the root cause of the error.
  - These error messages can be frustrating because they might lead you to spend time diagnosing or resolving problems that do not exist.
  - If the error message is related to a file system issue, Filemon will show what underlying errors might have occurred prior to the appearance of an error message.

7

Lab objective: Seeing an Error's Root Cause with Filemon

In this Lab objective:, you'll set permission on a directory and then perform a file save operation in Notepad that results in a misleading error message. Filemon's trace shows the actual error and the source of the message displayed in Notepad's error dialog box.

1.Run Filemon, and set the include filter to "notepad.exe".

2.Open Explorer, and create a directory named "Test" in a directory on an NTFS volume. (In this example, the root directory was used.)

3. Edit the security permissions on the Test directory to remove all access. This might require you to open the Advanced Security Settings dialog box and use the settings on the Permissions tab to remove inherited security. When you apply the modified security, Explorer should warn you that no one will have access to the folder.

4.Run Notepad, and enter some text into its window. Then select the Save entry in the File menu. In the File Name field of the Save dialog box, enter c:\test\test.txt (assuming the folder you created is on the volume C:).

5.Notepad will display the error message "c:\\test\test.txt - Path does not exist", implying that C:\\test does not exist.

Filemon shows that Notepad tried to open C:\Test and got an access-denied error. Immediately, it tried to open the C:\Test\Test.txt directory and received a file-not-found error because the directory does not exist.

The error message Notepad displays, "Path does not exist", is consistent with a file-not-found error, not an access-denied error. So it appears that Notepad first tried to open the directory, and when that failed it assumed for some reason that the name C:\Test\Test.txt was the name of a directory instead of a file. When it couldn't open that directory, Notepad presented the error message, but the root cause, which Filemon reveals, is the access denied error.

NTFS Streams Lab

- An NTFS has a default, unnamed data stream
- Applications can create additional streams
  - Each stream has different allocation size, actual size, and valid data length
  - Windows Explorer uses streams to store summary information for files (right-click -> properties)
  - Server for Macintosh stores resource fork in a separate stream
- Streams are named <file>":"<stream>

8

Lab objective: Looking at Streams

Most Windows applications aren't designed to work with alternate named streams, but both the echo and more commands are. Thus, a simple way to view streams in action is to create a named stream using echo and then display it using more. The following command sequence creates a file named test with a stream named stream:

```
C:\>echo hello > test:stream
C:\>more <  test:stream
hello
C:\>
```

If you perform a directory listing, test's file size doesn't reflect the data stored in the alternate stream because NTFS returns the size of only the unnamed data stream for file query operations, including directory listings.

```
C:\>dir test
  Volume in drive C is WINDOWS
  Volume Serial Number is3991-3040

    Directory of C:\    08/01/00 02:37p 0 test
          1 File(s) 0 bytes  112,558,080 bytes free
```

You can determine what files and directories on your system have alternate data streams with the Streams utility from www.sysinternals.com.

Hard links and Junctions - Lab

- A hard link allows multiple paths to refer to the same file
  - Created via CreateHardLink() or ln() functions
  - **`ln file file1`** creates a new name for file
- NTFS also supports Junctions (symbolic links)
  - Redirect file/pathname translation to another dir
  - Based on NTFS reparse points
  - No API functions to create reparse points (must use DeviceIoControl() or Linkd.exe / Junction.exe)
  - **`Linkd \etc C:\Windows\system32`** creates a new name for the Windows system32 directory

9

Lab objective: Creating a Junction

Windows doesn't include any tools for creating junctions, but you can create a junction with either the Junction tool from www.sysinternals.com (which includes source code) or the Windows resource kits tool Linkd. The Linkd tool also lets you view the definition of existing junctions, and Junction lets you view information about junctions and other reparse point tags.

```
C:> Linkd.exe \etc C:\Windows\system32
```

With the Posix subsystem installed, ln.exe becomes available that allows for creation of hard links:

```
C:> echo "hello again" > file
C:> ln.exe file file1
C:> more < file1
hello again
C:>
```

Viewing the MFT

- In NTFS, all data on a volume is stored in files,
  - data structures used to locate and retrieve files,
  - bootstrap data,
  - the bitmap that records the allocation state of the entire volume (the NTFS metadata).
- The MFT is the heart of an NTFS volume
  - implemented as an array of file records.
  - The size of each file record is fixed at 1 KB, regardless of cluster size.
  - Logically, the MFT contains one record for each file on the volume, including a record for the MFT itself.
- MFT can be inspected - it is only a file
  - Nfi.exe utility from OEM Support Tools

10

Lab objective: Viewing the MFT

The Nfi utility included in the OEM Support Tools (part of the Windows debugging tools and available for download at support.microsoft.com/support/kb/articles/Q253/0/66.asp) allows you to dump the contents of an NTFS volume's MFT as well as to translate a volume cluster number or physical-disk sector number (on non-RAID volumes only) to the file that contains it, if it's part of a file. The first 16 entries of the MFT are reserved for metadata files, but optional metadata files (which are present only if a volume uses an associated feature) fall outside this area: \$Extend\$Quota, \$Extend\$ObjId, \$Extend\$UsnJrnl, and \$Extend\$Reparse.

```
C:\>nfi G:\
NTFS File Sector Information Utility.
Copyright(C) Microsoft Corporation 1999.All rightsreserved.

File 0  Master File Table ($Mft)
  $STANDARD_INFORMATION (resident)
  $FILE_NAME (resident)
  $DATA(nonresident)
            logical sectors32-52447 (0x20-0xccdf)
  $BITMAP (nonresident)
            logical sectors16-23 (0x10-0x17)

File 1  MasterFile Table Mirror ($MftMirr)
  $STANDARD_INFORMATION (resident)
  $FILE_NAME (resident)
  $DATA(nonresident)
            logical sectors2048728-2048735 (0x1f42d8-0x1f42df)

File 2  LogFile ($LogFile)
  $STANDARD_INFORMATION (resident)
  $FILE_NAME (resident)
  $DATA(nonresident)
            logical sectors2048736-2073343 (0x1f42e0-0x1fa2ff)
```

View NTFS Information

- When it first accesses a volume, NTFS must mount it
  - read metadata from the disk
  - construct internal data structures so that it can process application file system accesses.
- To mount the volume, NTFS looks in the boot sector to find the physical disk address of the MFT.
  - The MFT's own file record is the first entry in the table;
  - The second file record points to a file located in the middle of the disk called the MFT mirror (filename $MftMirr) that contains a copy of the first few rows of the MFT.
- NTFSInfo.exe and Fsutil.exe tools reveal crucial information about MFT placement

11

Lab objective: Viewing NTFS Information

In Windows 2000, you can use the NTFSInfo tool from www.sysinternals.com to view information about an NTFS volume, including the placement and size of the MFT and MFT zone; and in Windows XP and Windows Server 2003, you can use the built-in Fsutil.exe command-line program:

```
C:\Windows\System32>fsutil fsinfo ntfsinfoc:
NTFS Volume Serial Number :         0xe82828e72828b68a
Version :                           3.1
NumberSectors :                     0x0000000001e461b7
TotalClusters :                     0x00000000003c8c36
Free Clusters :                     0x00000000000164c8
TotalReserved :                     0x00000000000001b0
BytesPerSector :                    512
BytesPerCluster:                    4096
BytesPerFileRecordSegment :         1024
Clusters PerFileRecordSegment:      0
MftValidData Length :               0x0000000006413800
MftStartLcn :                       0x00000000000c5294
Mft2 StartLcn :                     0x000000000002f427
MftZone Start :                     0x00000000003bf7e0
MftZone End :                       0x00000000003bf800
```