How Windows Starts Up (Part the second)

ntdebug 28 Jun 2007 9:13 AM 7

Howdy folks, David here again with part two of **How Windows Starts Up.** Today we'll be covering the **Boot Loader Phase**. Let's take a moment to recap where we are at this point. So far, the computer has completed POST, executed the MBR code, located the active partition, executed the Boot Sector code in that partition, found NTLDR and loaded it into memory. So what happens next? During this phase, we'll be discussing the following files located in the root of your system volume: **NTLDR**, **ntdetect.com** and **boot.ini**.

The Boot Loader Phase begins when NTLDR executes:

1. NTLDR switches the processor from real mode to 32-bit flat memory mode.
   How does it do this you ask? NTLDR contains two parts – a 16-bit component which handles switching the processor to Protected Mode and a 32-bit component which handles everything else.

   When NTLDR first begins execution, the processor is in an x86-operating mode called Real-Mode. In Real-Mode, no virtual to physical translation of memory addresses occurs, which means that all memory addresses use the 16-bit Segment:Offset format, and only the first 1MB of the computer's physical memory is accessible. However, we don't have to concern ourselves with this too much because the first action NTLDR takes is to switch the CPU to Protected-Mode.

   Trivia Alert *before we enter Protected-Mode, we enable the A20 Line – you IBM guys know what I'm talking about…*

   Still no virtual-to-physical translation occurs at this point in the boot process, but a full 32 bits of memory addressing becomes accessible, and now NTLDR can access up to 4GB of physical memory.

2. NTLDR creates enough page tables to make memory below 16MB accessible with paging turned on. This number has been increased with Windows Server 2003.

3. NTLDR enables paging. "Protected-Mode with Paging Enabled" is the mode in which Windows executes during normal operation.

4. NTLDR then starts the FAT and NTFS mini file system drivers contained within its own code. Unlike the Boot Sector's file system code, NTLDR's code is sophisticated enough to read subdirectories.

   Note: All disk access from this point until the Kernel Phase begins uses BIOS Interrupt 13h, Extended Interrupt 13h or NTBOOTDD.SYS

5. NTLDR locates and parses Hiberfil.sys to determine if it contains a valid hibernator file. If a valid hibernator file is found, then that file is loaded into memory and execution continues from the point the computer was placed into hibernation.

   Hibernation files are interesting. The file either contains a full memory image, or it contains a pointer (ARC path) to the boot partition of the last operating system that entered hibernation. That boot partition will contain another Hiberfil.sys file, which contains a full memory image of the hibernating operating system. If the Hiberfil.sys file is a pointer, the file will contain an ARC path using one of the following syntaxes:

   - linkmulti()disk()rdisk()partition()
   - linksignature()disk()rdisk()partition()
   - linkscsi()disk()rdisk()partition()

6. If Hiberfil.sys is not found, NTLDR parses the Boot.ini file. If the Boot.ini contains a reference to more than one operating system or boot option, then the boot loader screen is displayed.

   This is a sample of a default Boot.ini file from a Windows 2000 Server-based computer:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows 2000 Server" /fastdetect
```

Let's talk a bit about what's in the Boot.ini file. Boot.ini contains a list of ARC paths. Ok, quick – five bonus points to the first person who can tell me what 'ARC' stands for… come on… Yes, you there in the red shirt… correct! It stands for 'Advanced RISC Computing'; now beam down with the captain. The rest of you come with me…

ARC paths may use the "Multi", "SCSI" or "Signature" syntax.

The "Multi" syntax instructs Windows to rely on the system BIOS to load system files. This means that NTLDR will be using INT13 BIOS calls to find and load NTOSKRNL and any other files on the boot partition.

The "SCSI" syntax tells Windows to load a SCSI device driver to access the boot partition. A copy of the driver must be renamed to NTBOOTDD.SYS and placed in the root of the system partition.

The "Signature" syntax is equivalent to the SCSI syntax, but is used instead to support the Plug and Play architecture in Windows 2000 or later. Because Windows 2000 is a PnP OS, the SCSI controller number instance might vary each time you start Windows 2000, especially if you add new SCSI controller hardware after setup.

The Signature syntax instructs NTLDR to locate the drive with a disk signature that matches the value in the parentheses, regardless of which SCSI controller number that the drive is connected to. Keep in mind that the Signature syntax also requires the presence of an Ntbootdd.sys file in the root of the system partition. If you recall from the previous post, the disk signature value is located in the MBR.

Example:

```
signature(8b467c12)disk(1)rdisk(0)partition(2)\winnt="Windows 2000"
```

Signature() syntax is used only if one of the following conditions exists:

- The System BIOS or controller hosting the boot partition on which Windows is installed, does not support INT13 Extensions, or has the INT13 Extensions option disabled, and the partition on which you installed Windows is either larger than ~7.8 gigabytes (GB) in size, or the ending cylinder number for that partition is higher than 1024.
- The drive on which you installed Windows is connected to a SCSI controller with SCSI BIOS disabled, so INT13 BIOS calls cannot be used during the boot process.

227704 Windows May Use Signature() Syntax in the Boot.ini File
http://support.microsoft.com/default.aspx?scid=kb;EN-US;227704

7. If Boot.ini is not found, NTLDR attempts to continue the boot process by assuming that Windows was installed in the default location on the same partition (c:\windows). If the c:\windows directory is not found, NTLDR displays this error message: "**Windows could not start because the following file is missing or corrupt: \winnt root\system32\ntoskrnl.exe**"

8. If the boot loader screen is displayed, the user must select an operating system to boot.
   a. If the user chooses to run an operating system other than Windows 2003/XP such as DOS or Windows 95/98, NTLDR loads and executes the Bootsect.dos file located in the root of the boot drive.
      (1) If the Bootsect.dos file is found, NTLDR loads it into RAM and then passes control to it.
      (2) Bootsect.dos then loads IO.sys, MSDOS.sys, Command.com, etc., as necessary and proceeds to boot the downlevel operating system.
      (3) If the file is not found in the root, NTLDR displays:
      "**I/O Error accessing boot sector file multi(0)disk(0)rdisk(0)partition(1): \bootsect.dos**"
   b. If the user selects a Windows NT-based operating system, NTLDR executes Ntdetect.com.
      (1) If the ARC path points to a hard drive or partition that is not on the system, the following is displayed:
      Windows could not start because of a computer disk hardware configuration problem. Could not read from the selected boot disk. Check boot path and disk hardware. Please check the Windows documentation about hardware disk configuration and your hardware reference manuals for additional information.
      (2) The above message may also be displayed if you are using Signature() ARC path notation and the disk signature in the MBR is overwritten (for example, by a virus) or changed (corrupted) so that it no longer matches the Signature() value in the Boot.ini file. The following error message may also be displayed when there are problems with the Signature () value:
      Windows could not start because of the following ARC firmware boot configuration problem: did not properly generate ARC name for HAL and system paths. Please check the Windows documentation about ARC configuration options and your hardware reference manuals for additional information.

9. Ntdetect.com initializes and detects all supported hardware that it can find. (**Troubleshooting tip:** There is an Ntdetect.chk on the Windows Resource Kit CD in the Debug.cab file, which can be renamed and used in place of Ntdetect.com to view what Ntdetect.com is detecting during startup. You know you want to – go ahead, I'll wait…)

   Ntdetect.com detects the following components:

   a) Computer ID
   b) Bus/adapter type

c) Video adapter
d) Keyboard
e) Communications ports
f) Floppy disks
g) Mouse or other pointing devices
h) Parallel ports

10. Ntdetect.com passes its list of hardware components to NTLDR.

11. If the /SOS switch was used in the Boot.ini for the OS that is loading, NTLDR will display the files it loads into memory from this point on. Keep in mind – none of these files are being initialized. They are just being loaded into RAM at this point. I can't tell you how many times someone has incorrectly concluded that there is a problem with MUP.SYS because it's the last file displayed on this screen before their computer hangs. Poor MUP – always getting a bad rap. This list can be useful, however; it can help you identify potential malware that may be loading.

12. NTLDR loads Ntoskrnl.exe and Hal.dll into memory. Again, *loading*, not initializing.

13. NTLDR loads the System registry hive into memory. If Windows cannot load the registry hive, it may be necessary to restore the System hive from the ERD or by using the Windows Recovery Console to copy the files from the Windows\Repair or Windows\Repair\RegBack folder.

14. NTLDR reads the HKLM\System\Select registry key.
    a. If the Last Known Good option was not selected, NTLDR looks for the "Default" value in the SELECT key to select which ControlSet to use. (0x1 = ControlSet001, 0x2 = ContolSet002, 0x3 = ControlSet003)
    b. If the Last Known Good option was selected, NTLDR looks for the "LastKnownGood" value in the SELECT key to select the ControlSet to use.

15. NTLDR sets the "Current" value in the HKLM\System\Select key to the ControlSet being used.

    Note: At this point, NTLDR has determined which ControlSet it needs to use, but we don't actually create the "CurrentControlSet" symbolic link until the kernel initializes (more on that in our next installment). Therefore, for the purposes of this discussion, I'll use *ActiveControlSet* to refer to whichever registry ControlSet has been identified as the one NTLDR needs to use.

16. NTLDR scans HKLM\System*ActiveControlSet*\Control\Nls\Codepage for the "Acp", "Oemcp", and "Oemhal" values. These entries are pointers to the table of codepage information stored in the same key. The table has the filenames that correspond to the codepage numbers. NTLDR loads these files into memory. The defaults are C_1252.nls, C_437.nls, and Vgaoem.fon.

    A code page is a character set specific to an operating system or hardware platform. It maps character codes to individual characters. Different code pages include different special characters, typically customized for a language or a group of languages. The system uses code pages to translate keyboard input into character values for non-Unicode based applications, and to translate character values into characters for non-Unicode based output displays.

17. NTLDR scans HKLM\System*ActiveControlSet*\Control\Nls\Language for the "Default" value. This entry is a pointer to the table of language information stored in the same key. NTLDR loads this file into memory. The default is L_intl.nls.

18. NTLDR scans HKLM\System*ActiveControlSet*\Services for device drivers with a "Start" value of 0x0. It finds their "Group" and "Tag" values also.

19. NTLDR scans HKLM\System*ActiveControlSet*\Control\ServiceGroupOrder for the order that service groups should be loaded.

20. NTLDR sorts the drivers by "Group" and then by their "Tag" value. Each service or driver has a particular group to which it belongs. All drivers within a specific group are ordered by tag.

21. NTLDR loads the 0x0 device drivers into memory. It does not initialize or start these drivers.

22. The board is set, and the pieces are moving. NTLDR executes Ntoskrnl.exe.

Troubleshooting the Boot Loader Phase

Okay, now that we've seen what is supposed to happen, what could possibly go wrong? Well, several things…

First, all file-system activity during this phase uses BIOS Int13, Extended Int13 or Ntbootdd.sys.

1. If your computer uses Extended Int 13, verify that Ext. Int13 is enabled in your system's BIOS or SCSI BIOS as appropriate.

2. If you are using a SCSI driver, verify that you are using the correct SCSI driver (Ntbootdd.sys). If you are using "SCSI()" syntax in the Boot.ini file, copy the correct device driver for the SCSI controller to the root and then rename it to Ntbootdd.sys. This is the driver that is stored in the Winnt.sbs\System32\Drivers folder (for example, the Aic78xx.sys file). If you use "MULTI()" syntax in the Boot.ini file, you do not need to do this. Use the Windows Recovery Console or a Windows NT boot floppy to complete this step.

   301680 HOW TO: Create a Boot Disk for an NTFS or FAT Partition
   http://support.microsoft.com/default.aspx?scid=kb;EN-US;301680

   307654 How to install and use the Recovery Console in Windows XP

http://support.microsoft.com/default.aspx?scid=kb;EN-US;307654

Significant Errors during this phase:

- **Windows could not start because of a computer disk hardware configuration problem. Could not read from the selected boot disk. Check boot path and disk hardware. Please check the Windows documentation about hardware disk configuration and your hardware reference manuals for additional information.**

This error indicates that the ARC path is pointing to a hard drive or partition that is not on the system. This message may also be displayed if you are using Signature() ARC path notation and the disk signature in the MBR is overwritten (for example, by a virus) or changed (corrupted) so that it no longer matches the Signature() value in the Boot.ini file.

- **Windows could not start because of the following ARC firmware boot configuration problem: did not properly generate ARC name for HAL and system paths. Please check the Windows documentation about ARC configuration options and your hardware reference manuals for additional information.**

This error message may be displayed when there are problems with the Signature () ARC path syntax.

Ntoskrnl.exe missing or corrupt. This error could be the result of an incorrect ARC path in the Boot.ini file. Use the information in the following knowledge base articles to correct the Boot.ini information.

102873 BOOT.INI and ARC Path Naming Conventions and Usage
http://support.microsoft.com/default.aspx?scid=kb;EN-US;102873

155222 How to Determine the ARC Path
http://support.microsoft.com/default.aspx?scid=kb;EN-US;155222

130921 Creating an FT Boot Disk With Scsi( ) and Multi( ) Identifiers
http://support.microsoft.com/default.aspx?scid=kb;EN-US;130921

It is also possible to use the Windows XP Recovery Console to create a new Boot.ini for a Windows NT, Windows 2000 or .NET installation. Please see the following knowledge base article for a description of this process:

291980 A Discussion About the Bootcfg Command and Its Uses
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q291980

Specify the /SOS switch in the Boot.ini file and monitor the output to the screen. If the system stops responding before anything is output to the screen, then the problem may be with hardware detection. Replace Ntdetect.com with Ntdetect.chk to view the hardware detected by the system during this phase.

103659 Setup Hangs While Inspecting Hardware; How to Use Ntdetect.com
http://support.microsoft.com/default.aspx?scid=kb;EN-US;103659

320040 HOW TO: Use Ntdetect.chk to Identify Hardware Problems
http://support.microsoft.com/default.aspx?scid=kb;EN-US;320040

Incorrect HAL
If after specifying the /SOS option, you see that an incorrect version of the HAL.DLL is loading, use the following KB articles to replace the HAL or troubleshoot other HAL-related issues:

314477 "Hardware Configuration Problem" Err Msg Starting Windows
http://support.microsoft.com/default.aspx?scid=kb;EN-US;314477

237556 Troubleshooting Windows 2000 Hardware Abstraction Layer Issues
http://support.microsoft.com/default.aspx?scid=kb;EN-US;237556

330184 "Invalid Boot.ini" or "Windows could not start" error messages
http://support.microsoft.com/default.aspx?scid=kb;EN-US;330184

Unable to load System hive
Several problems could happen here. First, consider that the amount of memory we're dealing with is only 16MB. We have to squeeze NTLDR, the System Registry Hive, the kernel, the HAL, all the boot drivers and the PFN database into a relatively small area. So, with Windows Server 2003, we've given ourselves a bit more room.

302594 The System hive memory limitation is improved in Windows Server 2003
http://support.microsoft.com/default.aspx?scid=kb;EN-US;302594

If Windows is unable to load the System registry hive into memory, then it may be necessary to restore the System hive from the ERD or use the Windows Recovery Console to copy the files from the Winnt\Repair or Winnt\Repair\RegBack folder whichever version is most recent.

307545 How to Recover from a Corrupted Registry
http://support.microsoft.com/default.aspx?scid=kb;EN-US;307545

Table of Startup Type Values

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services	
0x0 = Boot	Loaded into RAM by the Boot loader (NTLDR) during Boot Loader Phase, then initialized by the Kernel (NTOSKRNL) during Kernel Phase
0x1 = System	Loaded and initialized by the Kernel during Kernel Phase
0x2 = Auto	Loaded or started automatically at system startup
0x3 = Manual	Driver is manually started by the user or another process
0x4 = Disabled	Driver is not to be started (There are exceptions, such as NTFS)

Conclusion
Well folks, we certainly have covered a lot of ground so far. Any questions? Yes, you there in the front… Ah, why have we not discussed any blue screens? Very good question. We haven't seen any BSOD's up to this point in the boot sequence because the kernel isn't running yet and STOP errors are generated by the kernel. No kernel, no blue screens.

That's all the time we have this week boys & girls. Join us next time for **How Windows Starts Up, The Kernel Phase**.

Comments

ryan
29 Jun 2007 12:32 PM

How windows starts up, the one and only part:

-

That is all. :P

Wayne
22 Aug 2007 3:36 PM

Is the kernel phase part coming out soon? Anxiously waiting.

Thanks

Mike Diack
15 Oct 2007 7:29 AM

Is part 3 - the kernel phase coming anytime soon - it's now mid October 2007....

[Sorry folks - apparently there was a temporary dip in the neutron saturation levels of the planck generator - this has now been corrected and turbines are up to speed. How Windows Starts Up - Part The Next will be posted this week.]

Reinier
6 Nov 2007 12:23 PM

it's november already and not part 3? :(

come on guys, you can do it! :)

wmbrae
29 Jun 2008 11:59 PM

I want to thank-you for the great care you took to make your explanation understandable. You were successful in communicating because you looked out for the reader. You are clearly not the engineer who wrote the error message

INACCESSIBLE_BOOT_DEVICE.

My friend's Win2000 PC failed out of old age (7 years used 10 hours a day). The hard drive was seemed to spin ok so we tried installing in on another box.

We got to step 11 /SOS switch was used in the Boot.ini...

NTLDR will display the files it loads into memory ...

I can't tell you how many times someone has incorrectly concluded that there is a problem with MUP.SYS because it's the last file displayed on this screen before their

computer hangs. Poor MUP – always getting a bad rap.

We hung exactly at this spot.

In our case, the bios had the ACPI switched off and so the load hung.

We set ACPI switch to on and Ntoskrnl.exe executed just fine.

So now you can continue your story and tell us what Ntoskrnl.exe does.

daniel j aguiar
10 Feb 2009 1:10 PM

Guys... Wonderfull post....

I cant wait the 3rd part

Lindsey
26 Sep 2011 8:36 PM

Hello, Im loving these articles... did you ever finish the 3rd and 4th part? I would KILL to read them.. please PLEASEEE!!!

[Sorry, we haven't put together any additional articles on this subject. The individuals who wrote the original articles have moved on from the team. We will consider this subject for future content.]