

Kick-Start Startup with Boot.ini

Kathy Ivens

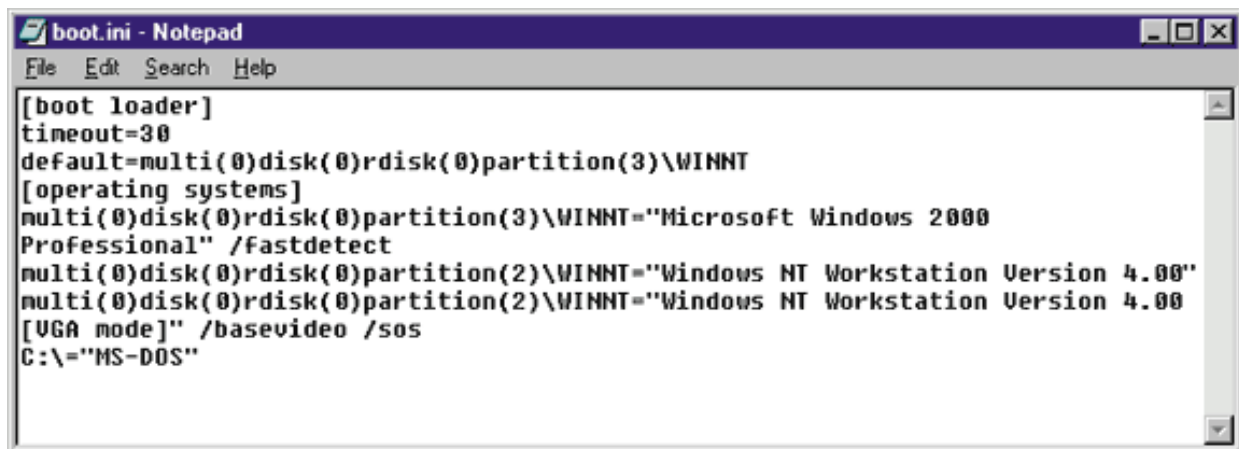
(Reprinted from WindowsItPro Magazine)

After you install Windows 2000, a small, hidden, read-only text file—boot.ini—exists in the boot partition's root directory. This file is an important component in the machinery that controls the OS startup processes. The installation process creates the file's contents, so boot.ini is specific to the computer. Understanding boot.ini's format, its likely content, the conventions that its content follows, and your options for manipulating the file gives you two important elements of control over a system. First, you can change the file's contents to modify the startup process. Second, you can create a boot.ini file to repair a computer that won't boot.

You can edit boot.ini in any text editor. Before you do so, I suggest you copy the original file to a 3.5" disk in case your changes wreak havoc. Boot.ini is read-only, so you must change that attribute before you can save your edits. (Of course, don't forget to restore the read-only attribute when you've finished editing.)

Boot.ini Sections

All .ini files follow the same format rules. The contents are arranged in sections, and each section has a title enclosed in square brackets. As Figure 1, page 110, shows, boot.ini has two sections: [boot loader] and [operating systems].



```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(3)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(3)\WINNT="Microsoft Windows 2000
Professional" /fastdetect
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Windows NT Workstation Version 4.00"
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Windows NT Workstation Version 4.00
[VGA mode]" /basevideo /sos
C:\="MS-DOS"
```

The [boot loader] section contains a timeout specification and a pointer to the default OS's location. The timeout specification is the amount of time, in seconds, during which users can make a selection from the onscreen menu that appears when users have a choice of startup options. By default, the timeout duration is 30 seconds, and the default OS loads if users fail to make a choice within that time. Typically, a choice of startup options occurs when more than one OS exists on the system (e.g., you updated a previous OS to Win2K and kept the previous OS, you installed two versions of Windows).

A choice also exists when you've installed the Recovery Console (RC), which automatically adds the Microsoft Windows 2000 Recovery Console option to the onscreen menu.

When users have no choices, no onscreen menu appears. The system ignores the timeout specification and immediately begins the OS startup.

Boot.ini's [operating systems] section contains the path or paths to the OS or OSs on the computer. As Figure 1 shows, text strings enclosed in quotation marks represent the text lines that the onscreen menu displays. You can edit the onscreen text to provide special instructions. For example, if you install a beta version of the next Windows version, you can edit the text to say Not for production.

Kick-Start Startup with Boot.ini

Kathy Ivens

(Reprinted from WindowsItPro Magazine)

ARC Path Statements

The [boot loader] section's OS location information and the [operating systems] section's OS path information both follow the conventions in the Advanced RISC Computing (ARC) specifications. Win2K recognizes three ARC path structures: multi syntax, SCSI syntax, and signature syntax.

Multi syntax. Systems that utilize IDE hard disks commonly use the multi syntax in boot.ini. Using the multi syntax tells Win2K to depend on the computer BIOS to load the system files. The OS uses INT 13 BIOS calls to find the disk that holds ntoskrnl.exe and the other files the system needs to boot the OS. You can also use the multi syntax for SCSI drives if the SCSI device is configured to use INT 13 calls instead of the device's BIOS settings.

In theory, the multi syntax can identify any drive that the standard INT 13 interface identifies. In reality, most system BIOS structures can identify only one disk controller through INT 13, so you can usually use the multi syntax to launch Win2K from one of only the first two drives attached to the primary disk controller. (If you have a cooperative BIOS, you can use the multi syntax for as many as four drives across two controllers.) A multi syntax line reads as follows:

```
multi(<A>)disk(<B>)rdisk(<C>)  
partition(<D>)
```

A is the ordinal number for the boot adapter. The first adapter, which is usually the boot adapter, is 0. B provides disk-parameter information. In a multi syntax line, this variable is always 0 because the multi syntax uses the INT 13 call instead of a self-discovery method.

C is an ordinal number that specifies the disk attached to the adapter; the number can range from 0 to 3, depending on the number of drives on the adapter. D specifies the partition number; the first possible number is 1 (as opposed to adapters and drives, which begin numbering with 0).

SCSI syntax. Your computer probably uses the SCSI syntax if you boot Win2K from a SCSI device. The SCSI syntax tells Win2K to use a device driver for the controller, rather than depend on the system BIOS and INT 13 calls, to access the boot partition. The device driver is always named ntbootdd.sys and resides on the system partition's root. To create ntbootdd.sys, Win2K Setup copies the specific driver for the SCSI drive to the hard disk. Win2K then renames the file ntbootdd.sys.

Usually, Win2K copies the driver from the Win2K CD-ROM (which contains drivers for the vast majority of SCSI adapters), but if you use a manufacturer-supplied driver, Win2K copies and renames that file. The SCSI syntax line is as follows:

```
scsi(<A>)disk(<B>)rdisk(<C>)  
partition(<D>)
```

A is the ordinal number for the adapter linked to the ntbootdd.sys driver. B is the SCSI ID for the target disk on that adapter. C is the SCSI LUN that contains the boot partition. (Although this LUN could be a separate disk, most SCSI setups have only one LUN per SCSI ID.) D specifies the partition number.

If you have multiple SCSI controllers, each of which uses a different device driver, A specifies the controller linked to ntbootdd.sys. During setup, Win2K determines—usually as the result of your specification for the installation partition—which controller to use. Even if your SCSI drive can work

Kick-Start Startup with Boot.ini

Kathy Ivens

(Reprinted from WindowsItPro Magazine)

with INT 13 calls, using the SCSI syntax is preferable (and usually less error-prone) because it forces the OS to use the ntbootdd.sys data during startup.

Signature Syntax

The signature syntax is technically the same as the SCSI syntax, but the installation program uses the signature syntax to support Win2K's Plug and Play (PnP) architecture. The signature syntax line is as follows:

```
signature(<A>)disk(<B>) rdisk(<C>)partition(<D>)
```

A is the disk signature (e.g., 6c156c97); the other variables are the same as for the SCSI syntax. A is a unique hexadecimal number that Win2K Setup writes to the Master Boot Record (MBR) during the Setup process's text-mode portion.

Using the signature syntax forces NT Loader (NTLDR)—the first file that Win2K launches to start the OS—to locate whichever drive has a disk signature that matches the A value during OS startup. Keep in mind that this drive might connect to a different SCSI controller number than when you first installed Win2K if you've added SCSI controllers to the machine. As with the SCSI syntax, the signature syntax requires a copy of the specific SCSI driver, renamed ntbootdd.sys, on the drive's root.

Win2K Setup determines that it should use the signature syntax under certain conditions. The most common conditions are when the drive has more than 1024 cylinders (which creates a problem if any cylinder number greater than 1024 is in the system partition) or when the SCSI controller's BIOS is disabled.

Tweaking Boot.ini

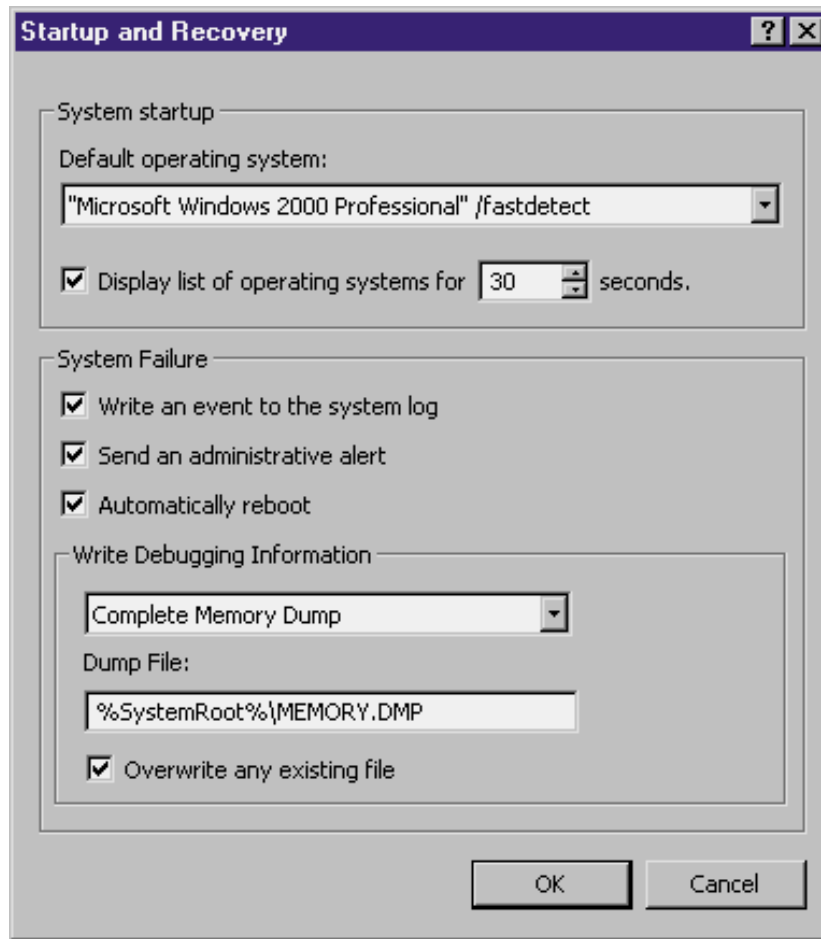
You can edit boot.ini to change or enhance the startup process. You can also edit the file to troubleshoot a computer that's having problems.

Changing the default OS. You can change a system's default OS and the length of the timeout that boot.ini offers users. You can make these modifications without editing the boot.ini file; simply use the System Properties dialog box. (To quickly access this dialog box, right-click My Computer and choose Properties from the shortcut menu.) Move to the Advanced tab, and click Startup and Recovery to open the Startup and Recovery dialog box, which Figure 2 shows. Select the desired OS from the Default operating system drop-down list in the System startup section.

Kick-Start Startup with Boot.ini

Kathy Ivens

(Reprinted from WindowsItPro Magazine)



Use the section's Display list of operating systems for x seconds option to change the timeout specification. Don't shorten the time to less than 10 seconds or users won't have time to read the options and select one. (Although you can clear the box to prevent display of the onscreen menu, I recommend against it. If you have to perform some administrative task that involves another OS, you'll need to go through all these steps again to redisplay the menu and gain access.)

In Win2K, you can't specify 1 as the timeout duration to leave the menu on the screen until the user makes a choice, as you can in Windows NT. Win2K's Startup and Recovery dialog box doesn't accept a negative number. If you manually edit boot.ini and change the timeout duration to 1, Win2K ignores the entry and resets the timeout to the previous specification during the next boot.

To keep the menu onscreen, a user can press any key except the Enter key (e.g., users can click an arrow key to highlight a different choice). Of course, this trick requires user intervention, so it is of no help unless the user is available to press a key. The user can't turn on the computer and head for the coffee machine and expect the menu to still be onscreen when he or she returns.

Troubleshooting

Boot.ini takes a substantial number of parameters, most of which are useful under specific conditions—usually when you're troubleshooting a serious problem. The file also requires certain switches to support specific hardware configurations. Table 1 lists several common boot.ini switches and their functions.

Kick-Start Startup with Boot.ini

Kathy Ivens

(Reprinted from WindowsItPro Magazine)

TABLE 1: Common Boot.ini Parameters	
Parameter	Purpose
/basevideo	Forces the system into 640 * 480, 16-color VGA mode.
/baudrate=nnnn	Sets the baud rate of the debug port. The default baud rate for the port is 19,200, but 9600 is the preferable rate for remote debugging through a modem. Using the /baudrate switch automatically activates the /debug switch.
/bootlog	Writes a log of the boot process to \%systemroot%\ntbtlog.txt. The log file contains a list of drivers that load and a list of drivers that fail to load during startup.
/crashdebug	Enables the debug COM port for debugging if Win2K crashes (i.e., has a STOP error) but lets you continue to use the COM port for regular serial operations if no crash occurs.
/debug	Enables the kernel debugger to perform live remote debugging through the COM port.
/debugport=comx	Chooses a COM port for the debug port. By default, the debug port is COM2; if COM2 doesn't exist, the default is COM1.
/fastdetect	Tells nt detect.com not to check parallel and serial ports; lets the PnP drivers perform that task during startup. Microsoft says that when you dual boot with any version of NT, Win2K automatically adds this switch to the boot.ini line that references Win2K. However, I've found this switch on every one of my six Win2K computers—even the systems that don't contain a previous version of any OS and that don't dual boot.
/nocodebug	Disables the kernel debugger (i.e., overrides all other debug settings). This switch speeds up the boot process but results in a blue screen when you run code that has a hard-coded debug breakpoint.
/noguiboot	Stops initialization of the VGA video driver that's responsible for presenting bitmapped graphics during the boot process. This driver is necessary to display progress information during bootup as well as the blue screen of death, so disabling the driver also prevents Win2K from displaying those screens.
/noserialmice:comx	Disables the mouse port check for the specified COM port. This switch is useful when you have a UPS on COM1 and don't want the OS to check the port for a mouse (the UPS sometimes interprets the communication signal as a warning and starts shutting down). If you use this switch without specifying a port, serial mouse detection is disabled for all ports. As you might guess, disabling serial mouse detection on all ports will cause a problem after the OS loads if you use a serial mouse.
/sos	Tells NTLDR to display the names of drivers as the drivers are loaded.

Creating an Emergency Boot Disk

If one of the Win2K startup files is missing or corrupt and the Windows File Protection (WFP) feature fails to correct the problem automatically, you can usually boot with a Win2K boot disk to replace the file. Because OS startup is totally dependent on the information in boot.ini, you must provide a boot.ini file on the boot disk, even if boot.ini isn't the corrupted or missing file.

Kick-Start Startup with Boot.ini

Kathy Ivens

(Reprinted from WindowsItPro Magazine)

To create a boot disk, format a 3.5" disk on another Win2K computer. Copy NTLDR and ntdetect.com from that computer's root to the 3.5" disk. These files aren't version specific, so the source computer can be running any version of Win2K Server or Win2K Professional.

If the source computer has exactly the same hard disk setup (i.e., same drive type, same drive, and same partition for the OS installation) as the computer you're troubleshooting, you can copy and use the source computer's boot.ini file as is. Otherwise, copy the boot.ini file onto the 3.5" disk, gather the necessary information about the destination computer's physical drive type, and edit the file's [operating systems] section information to create a viable boot.ini. Use the boot disk to boot the destination computer, and overwrite the corrupted file or files from the 3.5" disk.

Understanding what boot.ini does, and how it does it, gives you the power to control the OS startup process. This capability is especially important if you need to troubleshoot a computer that's producing blue screens of death because Microsoft Product Support Services (PSS) might ask you to edit boot.ini and add switches that help the debugging effort.