CHAPTER 14

Crash Dump Analysis

Almost every Windows user has heard of, if not experienced, the infamous "blue screen of death." This ominous term refers to the blue screen that is displayed when Windows crashes, or stops executing, because of a catastrophic fault or an internal condition that prevents the system from continuing to run.

In this chapter, we'll cover the basic problems that cause Windows to crash, describe the information presented on the blue screen, and explain the various configuration options available to create a *crash dump*, a record of system memory at the time of a crash that can help you figure out which component caused the crash and why. This section is *not* intended to provide detailed troubleshooting information on how to analyze a Windows system crash. This section will also show you how to analyze a crash dump to identify a faulty driver or component. The effort required to perform basic crash dump analysis is minimal and takes a few minutes. Even if an analysis ascertains the problematic driver for only one out of every five or ten crash dumps, it's still worth doing: one successful analysis can avoid future data loss, system downtime, and frustration.

Why Does Windows Crash?

Windows crashes (stops execution and displays the blue screen) for many possible reasons. A common source is a reference to a memory address that causes an access violation, either a write operation to read-only memory or a read operation on an address that is not mapped. Another common cause is an unexpected exception or trap. Crashes also occur when a kernel subsystem (such as the memory manager or power manager) or a driver (such as a USB or display driver) detect inconsistencies in their operation.

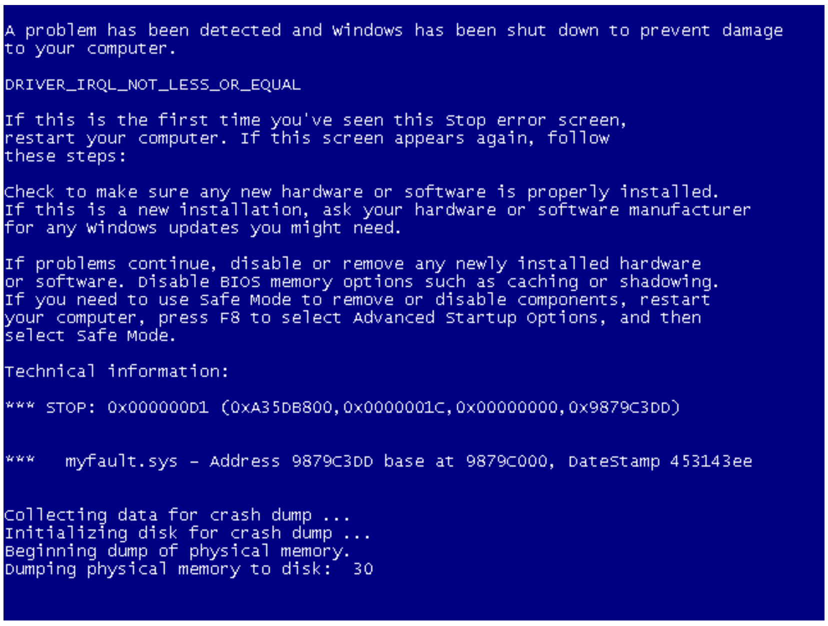
When a kernel-mode device driver or subsystem causes an illegal exception, Windows faces a difficult dilemma. It has detected that a part of the operating system with the ability to access any hardware device and any valid memory has done something it wasn't supposed to do.

But why does that mean Windows has to crash? Couldn't it just ignore the exception and let the device driver or subsystem continue as if nothing had happened? The possibility exists that the error was isolated and that the component will somehow recover. But what's more likely is that the detected exception resulted from deeper problems—for example, from a general corruption of memory or from a hardware device that's not functioning properly. Permitting the system to continue operating would probably result in more exceptions, and data stored on disk or other peripherals could become corrupt—a risk that's too high to take. So Windows adopts a *fail fast* policy in attempting to prevent the corruption in RAM from spreading to disk.

The Blue Screen

Regardless of the reason for a system crash, the function that actually performs the crash is *KeBug-CheckEx*, documented in the Windows Driver Kit (WDK). This function takes a *stop code* (sometimes called a *bugcheck code*) and four parameters that are interpreted on a per–stop code basis. After *KeBugCheckEx* masks out all interrupts on all processors of the system, it switches the display into a low-resolution VGA graphics mode (one implemented by all Windows-supported video cards), paints a blue background, and then displays the stop code, followed by some text suggesting what the user can do. Finally, *KeBugCheckEx* calls any registered device driver bugcheck callbacks (registered by calling the *KeRegisterBugCheckCallback* function), allowing drivers an opportunity to stop their devices. It then calls registered reason callbacks (registered with *KeRegisterBugCheckReasonCallback)*, which allow drivers to append data to the crash dump or write crash dump information to alternate devices.

The first line in the Technical information section in the sample Windows blue screen shown in Figure 14-1 lists the stop code and the four additional parameters passed to *KeBugCheckEx*. A text line near the top of the screen provides the text equivalent of the stop code's numeric identifier. According to the example in Figure 14-1, the stop code 0x000000D1 is a DRIVER_IRQL_NOT_LESS_OR_EQUAL crash. When a parameter contains an address of a piece of operating system or device driver code (as in Figure 14-1), Windows displays the base address of the module the address falls in, the date stamp, and the file name of the device driver. This information alone might help you pinpoint the faulty component.



```
A problem has been detected and Windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0xA35DB800,0x0000001C,0x00000000,0x9879C3DD)


***   myfault.sys - Address 9879C3DD base at 9879C000, DateStamp 453143ee


Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk:  30
```

FIGURE 14-1 Example of a blue screen

Although there are more than 300 unique stop codes, most are rarely, if ever, seen on production systems. Instead, just a few common stop codes represent the majority of Windows system crashes. Also, the meaning of the four additional parameters depends on the stop code (and not all stop codes have extended parameter information). Nevertheless, looking up the stop code and the meaning of the parameters (if applicable) might at least assist you in diagnosing the component that is failing (or the hardware device that is causing the crash).

You can find stop code information in the section "Bug Checks (Blue Screens)" in the Debugging Tools for Windows help file. (For information on the Debugging Tools for Windows, see Chapter 1, "Concepts and Tools," in Part 1.) You can also search Microsoft's Knowledge Base (*http://support.microsoft.com*) for the stop code and the name of the suspect hardware or driver. You might find information about a workaround, an update, or a service pack that fixes the problem you're having. The Bugcodes.h file in the WDK contains a complete list of the 300 or so stop codes, with some additional details on the reasons for some of them. Last but not least, these stop codes are listed and documented at *http://msdn.microsoft.com/en-us/library/windows/hardware/hh406232(v=vs.85).aspx*.

Causes of Windows Crashes

Based on data collected from the release of Windows 7 through the release of Windows 7 SP1, the top 20 stop codes account for 91 percent of crashes and can be grouped into the following categories:

- **Page fault** A page fault on memory backed by data in a paging file or a memory-mapped file occurs at an IRQL of DPC/dispatch level or above, which would require the memory manager to have to wait for an I/O operation to occur. The kernel cannot wait or reschedule threads at an IRQL of DPC/dispatch level or higher. (See Chapter 3, "System Mechanisms," in Part 1 for details on IRQLs.) The common stop codes are:
  - 0xA - IRQL_NOT_LESS_OR_EQUAL
  - 0xD1 - DRIVER_IRQL_NOT_LESS_OR_EQUAL
- **Power management** A device driver or an operating system function running in kernel mode is in an inconsistent or invalid power state. Most frequently, some component has failed to complete a power management I/O request operation within the default period of 10 minutes. The common stop code is:
  - 0x9F - DRIVER_POWER_STATE_FAILURE
- **Exceptions and traps** A device driver or an operating system function running in kernel mode incurs an unexpected exception or trap. The common stop codes are:
  - 0x1E - KMODE_EXCEPTION_NOT_HANDLED
  - 0x3B - SYSTEM_SERVICE_EXCEPTION
  - 0x7E - SYSTEM_THREAD_EXCEPTION_NOT_HANDLED
  - 0x7F - UNEXPECTED_KERNEL_MODE_TRAP

    - 0x8E - KERNEL_MODE_EXCEPTION_NOT_HANDLED with P1 != 0xC0000005 STATUS_ACCESS_VIOLATION

- **Access violations** A device driver or an operating system function running in kernel mode incurs a memory access violation, which is caused either by attempting to write to a read-only page or by attempting to read an address that isn't currently mapped and therefore is not a valid memory location. The common stop codes are:
    - 0x50 - PAGE_FAULT_IN_NONPAGED_AREA
    - 0x8E - KERNEL_MODE_EXCEPTION_NOT_HANDLED with P1 = 0xC0000005 STATUS_ACCESS_VIOLATION
- **Display** The display device driver detects that it can no longer control the graphics processing unit. This indicates that an attempt to reset the display driver failed. The common stop code is:
    - 0x116 - VIDEO_TDR_FAILURE
- **Pool** The kernel pool manager detects a corrupt pool header or an improper pool reference. The common stop codes are:
    - 0x19 - BAD_POOL_HEADER
    - 0xC2 - BAD_POOL_CALLER
    - 0xC5 - DRIVER_CORRUPTED_EXPOOL
- **Memory management** The kernel memory manager detects a corruption of memory management data structures or an improper memory management request. The common stop codes are:
    - 0x1A - MEMORY_MANAGEMENT
    - 0x4E - PFN_LIST_CORRUPT
- **Hardware** A hardware error, such as a machine check or a nonmaskable interrupt (NMI), occurs. This category also includes disk failures when the memory manager is attempting to read data to satisfy page faults. The common stop codes are:
    - 0x7A - KERNEL_DATA_INPAGE_ERROR
    - 0x124 - WHEA_UNCORRECTABLE_ERROR
- **USB** An unrecoverable error occurs in a universal serial bus operation. The common stop code is:
    - 0xFE - BUGCODE_USB_DRIVER
- **Critical object** A fatal error occurs in a critical object without which Windows cannot continue to run. The common stop code is:

    - 0xF4 - CRITICAL_OBJECT_TERMINATION

- **NTFS file system** A fatal error is detected by the NTFS file system. The common stop code is:
    - 0x24 - NTFS_FILE_SYSTEM

Figure 14-2 shows the distribution of these categories for Windows 7 and Windows 7 SP1 in May 2012:

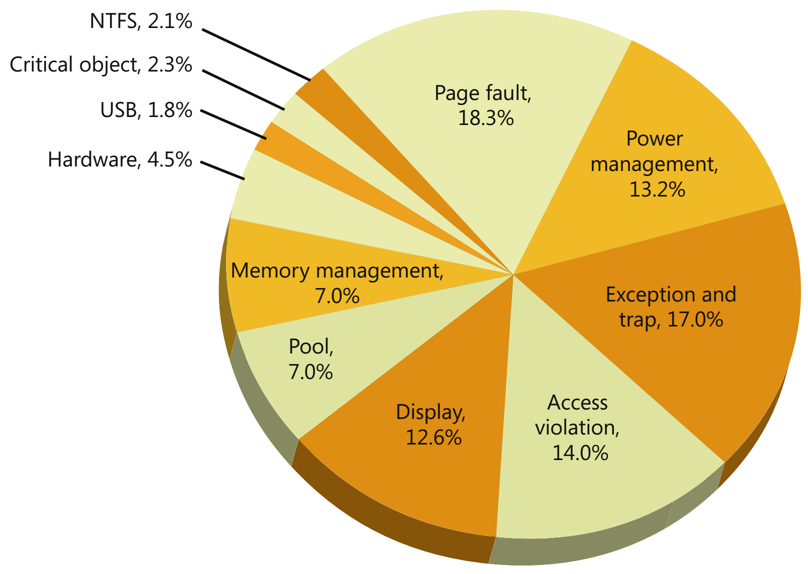


FIGURE 14-2 Distribution of top 20 stop codes by category for Windows 7 and Windows 7 SP1 in May 2012.

Troubleshooting Crashes

You often begin seeing blue screens after you install a new software product or piece of hardware. If you've just added a driver, rebooted, and gotten a blue screen early in system initialization, you can reset the machine, press the F8 key when instructed, and then select Last Known Good Configuration. Enabling last known good causes Windows to revert to a copy of the registry's device driver registration key (HKLM\SYSTEM\CurrentControlSet\Services) from the last successful boot (before you installed the driver). From the perspective of last known good, a successful boot is one in which all services and drivers have finished loading and at least one logon has succeeded. (Last known good is further described in Chapter 13, "Startup and Shutdown.")

During the reboot after a crash, the Boot Manager (Bootmgr) will automatically detect that Windows did not shut down properly and display a Windows Error Recovery message similar to the one shown in Figure 14-3. This screen gives you the option to attempt booting into safe mode so that you can disable or uninstall the software component that might be broken.