When you get a stop error (Blue Screen of Death), your system writes a small file called a minidump. This is a small write up on how to debug memory dumps. This becomes extremely useful when you are trying to figure out what caused a particular stop error, and no filename was mentioned and/or it is undocumented.

You could always let Microsoft do it for you, but there is no gurantee they will answer, and it takes a very long time (over a month in my case).

Your first step is to make certain your computer is setup to record memory dumps. The small dumps are most desirable, because they aren't the size of your amount of ram!

Right click My Computer, choose properties. Click on the advanced tab, and then choose startup and recovery 'settings.' From the screenshot attached at the bottom you will see the settings you want. By default, this is largely how it is already setup; I only unchecked automatically restart for XP. For Vista, there is an extra step involved, you must click start, right click computer. Then from the next screen, click Advanced system settings. Then, its in the same location as XP. I have attached a Vista screenshot, as the options are a little different.

Note: Make certain that your pagefile still resides on the system partition, otherwise WIndows will not be able to save the debug files.

Your second step is to download and install the Microsoft Debugging Tools found here:
http://www.microsoft.com/whdc/devtoo...nstallx86.mspx

Once you have downloaded and installed these tools, go to start, all programs, Debugging Tools For Windows, Windbg. Once you open Windbg, you will presented with a blank screen. Click on File, Symbol File Path. Here you will enter the symbols path. Symbols are needed to effectively debug.

The path will be:

SRV*c:\symbols*http://msdl.microsoft.com/download/symbols

Enter in this path and click OK. Now, go to File, Save Workspace so that your symbols path is saved for future use. Now what you want to do is locate your memory dumps. They are usually located in %systemroot%/minidump (in my case C:/windows/minidump).

If you notice, they are usually named the date, and then a -*number* to indicate the order of minidumps that day. My example is called Mini061904-01.dmp (it happened today).

Inside of Windbg, go to File, Open Crash Dump and load the file. You will get a message to save base workspace information. Choose no.

Now you will get a debugging screen. Now it takes a little bit to run it, as the symbols have to be downloaded as they are needed. Then you will see information such as:

```
Symbol search path is: SRV*c:\symbols*
http://msdl.microsoft.com/download/symbols

Microsoft (R) Windows Debugger Version 6.3.0017.0
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\WINDOWS\Minidump\Mini061904-01.dmp]
Mini Kernel Dump File: Only registers and stack trace are available

Symbol search path is: SRV*c:\symbols*
http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows XP Kernel Version 2600 (Service Pack 1) UP Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 2600.xpsp2.030422-1633
Kernel base = 0x804d4000 PsLoadedModuleList = 0x80543530
Debug session time: Sat Jun 19 19:06:57 2004
System Uptime: 0 days 1:03:36.951
Loading Kernel Symbols
......................................................................
..............................................................
Loading unloaded module list
..........
Loading User Symbols
**********************************************************************
*********
* *
* Bugcheck Analysis *
* *
**********************************************************************
*********

Use !analyze -v to get detailed debugging information.

BugCheck 86427532,  <--This is your stop code
```

```
Unable to load image pavdrv51.sys, Win32 error 2
*** WARNING: Unable to verify timestamp for pavdrv51.sys
*** ERROR: Module load completed but symbols could not be loaded for
pavdrv51.sys
Probably caused by : pavdrv51.sys ( pavdrv51+7fc0 )

Followup: MachineOwner
---------
```

Now, we can already see what it was most likely caused by, in my case it was pavdrv51.sys, which is a Panda AV file.

If we want to get further in depth, we can use the command, !analyze -v at the kd> prompt to delve more info about the error:

```
kd> !analyze -v
**********************************************************************
*********
* *
* Bugcheck Analysis *
* *
**********************************************************************
*********

Unknown bugcheck code (86427532)
Unknown bugcheck description <--Its unknown, and not listed on the MS
KB at all.
Arguments:
Arg1: 000001db
Arg2: 00000002
Arg3: 00000003
Arg4: 0000000b

Debugging Details:
------------------


CUSTOMER_CRASH_COUNT: 1

DEFAULT_BUCKET_ID: DRIVER_FAULT

BUGCHECK_STR: 0x86427532

LAST_CONTROL_TRANSFER: from f4198fc0 to 804f4103

STACK_TEXT:
f41f0964 f4198fc0 86427532 000001db 00000002 nt!KeBugCheckEx+0x19
```

```
WARNING: Stack unwind information not available. Following frames may
be wrong.
f41f0ba0 f419920b 864db520 f419ccf0 00000000 pavdrv51+0x7fc0
f41f0c34 804ea221 865b8910 864a52c0 806ad190 pavdrv51+0x820b
f41f0c44 8055d0fe 864a5330 86305028 864a52c0 nt!IopfCallDriver+0x31
f41f0c58 8055de46 865b8910 864a52c0 86305028
nt!IopSynchronousServiceTail+0x5e
f41f0d00 80556cea 000000a4 00000000 00000000
nt!IopXxxControlFile+0x5c2
f41f0d34 8052d571 000000a4 00000000 00000000
nt!NtDeviceIoControlFile+0x28
f41f0d34 7ffe0304 000000a4 00000000 00000000 nt!KiSystemService+0xc4
00cdff70 00000000 00000000 00000000 00000000
SharedUserData!SystemCallStub+0x4


FOLLOWUP_IP:
pavdrv51+7fc0
f4198fc0 ?? ???

SYMBOL_STACK_INDEX: 1

FOLLOWUP_NAME: MachineOwner

SYMBOL_NAME: pavdrv51+7fc0

MODULE_NAME: pavdrv51

IMAGE_NAME: pavdrv51.sys

DEBUG_FLR_IMAGE_TIMESTAMP: 3e8c072b

STACK_COMMAND: kb

BUCKET_ID: 0x86427532_pavdrv51+7fc0

Followup: MachineOwner
---------
```

Update: After the intial run of the debug process, you can use the command !analyze -v to gather more information.

Now that may be more info than you need. This tutorial only covers minidumps, however, if needed, you could change your memory dump options to do a complete dump. This is useful, however, very cumbersome, as the file generated will be the same size as your amount of ram.

Note: Make absolutely sure that your symbol path is correct. If it isn't, then you will get symbol errors and not likely be able to debug the dump to get the info you desire.

I hope this info is useful, I find it invaluable to finding out what is causing random, sporadic, and/or obscure stop errors.