Finding Crash Information Using the MAP File

Wouter Dhondt
5 Jan 2003 CPOL 5 min read

Finding crash information using the MAP file: how to create and read the file

Introduction

Programming neat applications is one thing. But when a user informs you your software has crashed, you know it's best to fix this before adding other features. If you're lucky enough, the user will have a crash address. This will go a long way in solving the problem. But how can you determine what went wrong, using this crash address?

Creating a MAP File

Well first of all, you'll need a MAP file. If you don't have one, it will be nearly impossible to find where your application crashed using the crash address. So first, I'll show you how to create a good MAP file. For this, I will create a new project (`MAPFILE`). You can do the same, or adjust your own project. I create a new project using the Win32 Application option in VC++ 6.0, selecting the typical "`Hello World!`" application to keep the size of the MAP file reasonable for explanation.

Once created, we need to adjust the project settings for the release version. In the C/C++ tab, select "Line Numbers Only" for Debug info.

Many people forget this, but you'll need this option if you want a good MAP file. This will not affect your release in any way. Next is the Link tab. Here, you need to select the "Generate mapfile" option. Also, type the switches `/MAPINFO:LINES` and `/MAPINFO:EXPORTS` in the Project Options edit box.

Now, you're ready to compile and link your project. After linking, you will find a *.map* file in your intermediate directory (together with your EXE).

Reading the MAP File

After all this dull work, now comes the neat part: how to read the MAP file. We'll do this by using a crash example. So first: how to crash your application. I did this by adding these two lines at the end of the `InitInstance()` function:

C++

```
char* pEmpty = NULL;
*pEmpty = '*.*'; // This is line 119
```

I'm sure you can find other instructions which will crash your application. Now recompile and link. If you start the application, it will crash and you'll get a message like this: '*The instruction at "0x00401a1a" referenced memory at "0x00000000". The memory could not be "written".*'.

Now, it's time to open the MAP file with notepad or something similar. Your MAP file will look like this:

The top of the MAP file contains the module name, the timestamp indicating the link of the project, and the preferred load address (which will probably be `0x00400000` unless you're using a DLL). After the header comes the section information that shows which sections the linker brought in from the various OBJ and LIB files.

Show ▲

```
MAPFILE

 Timestamp is 3df6394d (Tue Dec 10 19:58:21 2002)

 Preferred load address is 00400000

 Start         Length     Name                   Class
 0001:00000000 0000f3f8H .text$mn                 CODE
 0002:00000000 00000074H .idata$2                 DATA
 0002:00000078 00000394H .rdata                   DATA
 0002:0000040c 00000028H .idata$3                 DATA
 0002:00000434 00000014H .idata$3                 DATA
 0002:0000044c 00000074H .idata$4                 DATA
 0002:000005bc 0000040aH .idata$6                 DATA
 0002:00000bc6 00000000H .edata                   DATA
 0003:00000000 0000004cH .CRTSXCA                 DATA
 0003:00000050 00000000H .CRTSXCZ                 DATA
 0003:00000008 00000004H .CRTSXIA                 DATA
 0003:0000000c 00000004H .CRTSXIZ                 DATA
 0003:00000010 00000004H .CRTSXPA                 DATA
 0003:00000014 00000004H .CRTSXPZ                 DATA
 0003:00000018 00000004H .CRTSXTA                 DATA
 0003:0000001c 00000004H .CRTSXTZ                 DATA
 0003:00000020 00001400H .data                    DATA
 0003:00002420 00000050H .bss                     DATA
 0004:00000000 0000025dH .rsrc$01                 DATA
 0004:00000250 00000720H .rsrc$02                 DATA
```

After the section information, you get the `public` function information. Notice the "`public`" part. If you have `static`-declared C functions, they won't show up in the MAP file. Fortunately, the line numbers will still reflect the `static` functions. The important parts of the `public` function information are the function names and the information in the `Rva+Base` column, which is the starting address of the function.

Show ▲

```
 Address         Publics by Value              Rva+Base     Lib:Object

 0001:00000000       _WinMain@16               00401000 f   MAPFILE.obj
 0001:00000c0        ?MyRegisterClass@YAPAPAUHINSTANCE__@@DZ 00401dc0 f   MAPFILE.obj
 0001:00000150       ?InitInstance@YAPAPAUHINSTANCE__@@HDZ 00401150 f   MAPFILE.obj
 0001:00000200       ?WndProc@YAJPAUHWND__@@IJJZ 00401200 f   MAPFILE.obj
 0001:00000310       ?About@YAJPAUHWND__@@IJJZ 00401310 f   MAPFILE.obj
 0001:00000350       __WinMainCRTStartup       00401350 f   LIBC:wincrt0.obj
 0001:00000446       __amsg_exit               00401446 f   LIBC:wincrt0.obj
 0001:0000048f       __cinit                   00401487 f   LIBC:crt0dat.obj
 0001:0000050c       __exit                    0040140c f   LIBC:crt0dat.obj
 0001:0000040d       __exit                    0040140d f   LIBC:crt0dat.obj
 0001:00000581       __XcptFilter              00401591 f   LIBC:winxfltr.obj
 0001:0000097c       __wincmdln                00401715 f   LIBC:wincmdln.obj
 //UNIFIED FOR BETTER READING
 0003:00002ab4       __FPinit                  00408ab4     <common>
 0003:00002ab8       __acmdln                  00408ab8     <common>

entry point at        0001:00000350

 Static symbols

 0001:00003500       LeadUp1                   004045d0 f   LIBC:memmove.obj
 0001:00003fcc       LeadUp2                   004045fc f   LIBC:memmove.obj
 //UNIFIED FOR BETTER READING
 0001:00000577       __initterm                00401577 f   LIBC:crt0dat.obj
 0001:0000048b       __fast_error_exit         0040148b f   LIBC:wincrt0.obj
```

The `public` function part is followed by the line information (you got this if you used the `/MAPINFO:LINES` in the Link tab and selected the "Line numbers" in the C/C++ tab). After this, you will get the export information if your project contains exported functions and you included `/MAPINFO:EXPORTS` in the link tab.

```
Line numbers for .\Release\MAPFILE.obj(F:\MAPFILE\MAPFILE.cpp) segment .text

    24 0001:00000000   30 0001:00000004    31 0001:0000001b   32 0001:00000027
    33 0001:00000029   34 0001:0000002b    35 0001:0000002d   36 0001:0000003a
    45 0001:00000077   47 0001:00000080    48 0001:0000008f   52 0001:0000009d
    53 0001:000000b3   71 0001:000000c0    80 0001:000000c3   81 0001:000000c8
    82 0001:000000f7   86 0001:00000114    88 0001:00000135   89 0001:00000145
   102 0001:000000150  108 0001:00000155   110 0001:00000188  122 0001:000001ffd
   115 0001:000001bf   116 0001:000001ca   119 0001:000001a1  121 0001:000001a8
   122 0001:000001ae   135 0001:000001b0   143 0001:000001cc  172 0001:000001ee
   175 0001:000002fc   177 0001:00000316   179 0001:00000324  185 0001:00000326
   154 0001:00000251   164 0001:00000258   167 0001:00000263  171 0001:00000264
   165 0001:00000267   175 0001:00000269   161 0001:00000294  164 0001:000002a8
   165 0001:000002b6   166 0001:000002d8   174 0001:000002e7  175 0001:000002e9
   169 0001:000002f2   174 0001:000002fa   175 0001:000002fc  179 0001:00000310
   186 0001:0000031e   193 0001:0000032e   194 0001:00000330  188 0001:00000333
   183 0001:000000344  194 0001:00000349
```

Now we will look up where the crash occurred. First, we'll determine which function contains the crash address. Look in the `Rva+Base` column and search the first function with an address bigger than the crash address. The preceding entry in the MAP file is the function that had the crash. In our example our crash address is `0x00401a1a`. This is between `0x00401150` and `0x00401100` so we know the crash function is `?InitInstance@YAPAPAUHINSTANCE__@@HDZ`. Any function name that starts with a question mark is a C++ decorated name. To translate the name, pass it as a command-line parameter to the Platform SDK program UNDNAME.EXE (in the bin dir). You won't need to do this most of the time as you might figure it out just by looking at it (here: `InitInstance()` in MAPFILE.obj).

This is a big step for bug tracking. But it gets even better: we can find out on which line the crash occurred! We need to do some basic hexadecimal mathematics, so people who can't do this without a calculator: now is the time to use it. The first step is the following calculation: `crash_address - preferred_load_address - 0x1000`.

Addresses are offsets from the beginning of the first code section, so we need to do this calculation. Subtracting the preferred load address is logical, but why do we need to substract another `0x1000`? The crash address is an offset from the beginning of the code section, but the first part of the binary isn't the code section! The first part of the binary is the Portable Executable (PE), which is `0x1000` bytes long. Mystery solved. In our example, this is: `0x00401a1a - 0x00400000 - 0x1000 = 0x1a1`

Now it's time to look in the line information section of the MAP file. The lines are shown like this: `30 0001:00000004`. The first number is the line number, the second number is the offset from the beginning of the code section in which this line occurred. If we want to look for our line number, we just have to do the same thing we did for the function: determine the first occurrence of a bigger offset than the one we just calculated. The crash occurred in the preceding entry. In our example: `0x1a1` is before `0x1a8`. So our crash occurred on `line 119` in *MAPFILE.CPP*.

Keeping Track of MAP Files

Each release had its own MAP file. It's not a bad idea to include the MAP file with the EXE distribution. This way, you can be certain you have the correct MAP file for this EXE. You could keep every MAP file with every EXE on your system, but we all know this might give some troubles later on. The MAP file doesn't contain any information you wouldn't want the user to see (unless maybe class and function names?). Auser would have no use with it, but at least you can ask for the MAP file if you don't have a copy yourself.

Acknowledgements

- John Robbins for his "**Debugging Applications**" book

License

Written By
Wouter Dhondt
Web Developer
Belgium

Wouter got interested in computers and programming at the age of 12 (using a 286 and basic). Several years and an electronics degree later, he started working as a software engineer. In the summer of 2001, Wouter created Fping as an alternative to the windows ping program (just for his own amusement). Amazed by the response / interest, he founded kwakkelflap.com to ensure a better distribution for the tool. Several other applications have been released since...

Comments and Discussions

Crash map from Open Watcom IDE	Member 14813645	15-May-20 7:27
Bug Finder	Antonio Petricca	7-Jun-13 1:47
My vote of 5	LeoBro	30-May-13 20:48
My vote of 5	jlwarlow	18-Apr-12 0:06
Find crash address with win? ?	ralfsch	4-Nov-09 22:02
Re: Find crash address with win? ?	leproza	29-Mar-12 23:15
Nice article..!	Kumar.Prabhu	26-Nov-08 20:55
Can a Map File Be Created for a C# in VS.Net 2005 Express?	RBinSFla	13-Feb-08 8:13
Partial Answer: Differences Between C# Compiler and C++ Compiler Output	RBinSFla	13-Feb-08 10:49
Couldn't understand how to got exact line number 119	shumonshumon	11-Jan-08 0:44
Re: Couldn't understand how to got exact line number 119	Priya_Sundar	3-Mar-09 18:45
How to add line numbers in visual studio 2005?	motihadar	27-Dec-07 3:21
Re: How to add line numbers in visual studio 2005?	Mark Colvin	20-May-20 10:17
How to make sense of a crash in a DLL	DeepT	12-Dec-07 6:36
How to use map file for finding the crash in DLL?	login0001	13-Feb-07 17:19
Re: How to use map file for finding the crash in DLL?	indujce	19-Sep-11 23:59
If crash is in a DLL ?	Aravind Kumar K	15-Nov-06 16:23
What about crash in static lib code ?	ana_v123	10-Jul-06 7:10
Re: What about crash in static lib code ?	Rodrick	13-Sep-06 8:36
Re: What about crash in static lib code ?	ekodeveloper	29-Sep-06 5:09
How to do this in VS2005	Sreekanth Muralidhar	30-Dec-05 18:23
Re: How to do this in VS2005	luzhiyang1221	8-Feb-10 16:08
How to gen .map lines info in VS.NET IDE	timhaung	22-Dec-05 22:04
application crash	GSCHARIYA	19-Oct-05 2:08
Application crashed error	Anonymous	28-May-05 0:17