

Finding Crash Information Using the MAP File

Wouter Dhondt

Rate me:

★★★★★

 4.94/5 (138 votes)

5 Jan 2003

CPOL

5 min read

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

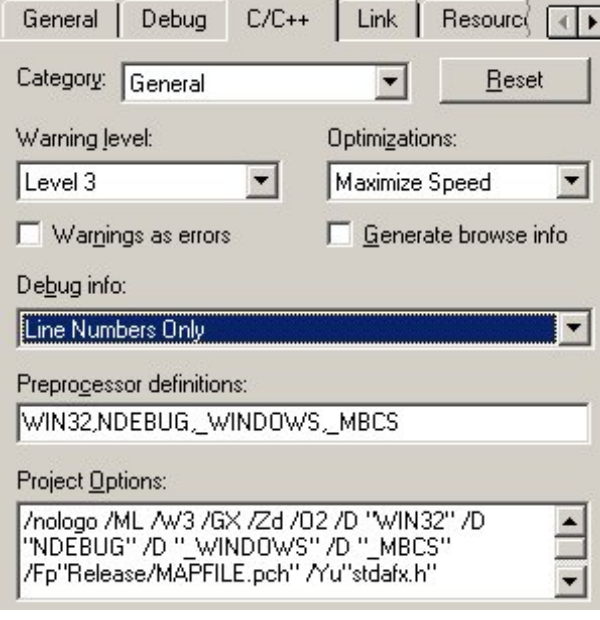
Introduction

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?

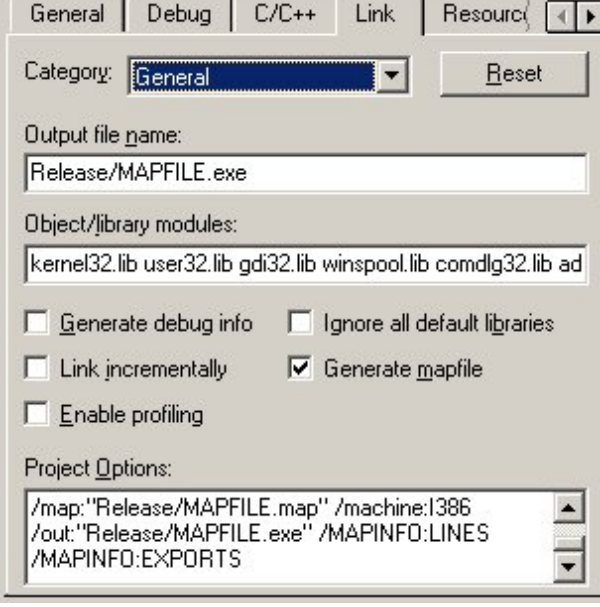
Creating a MAP File

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

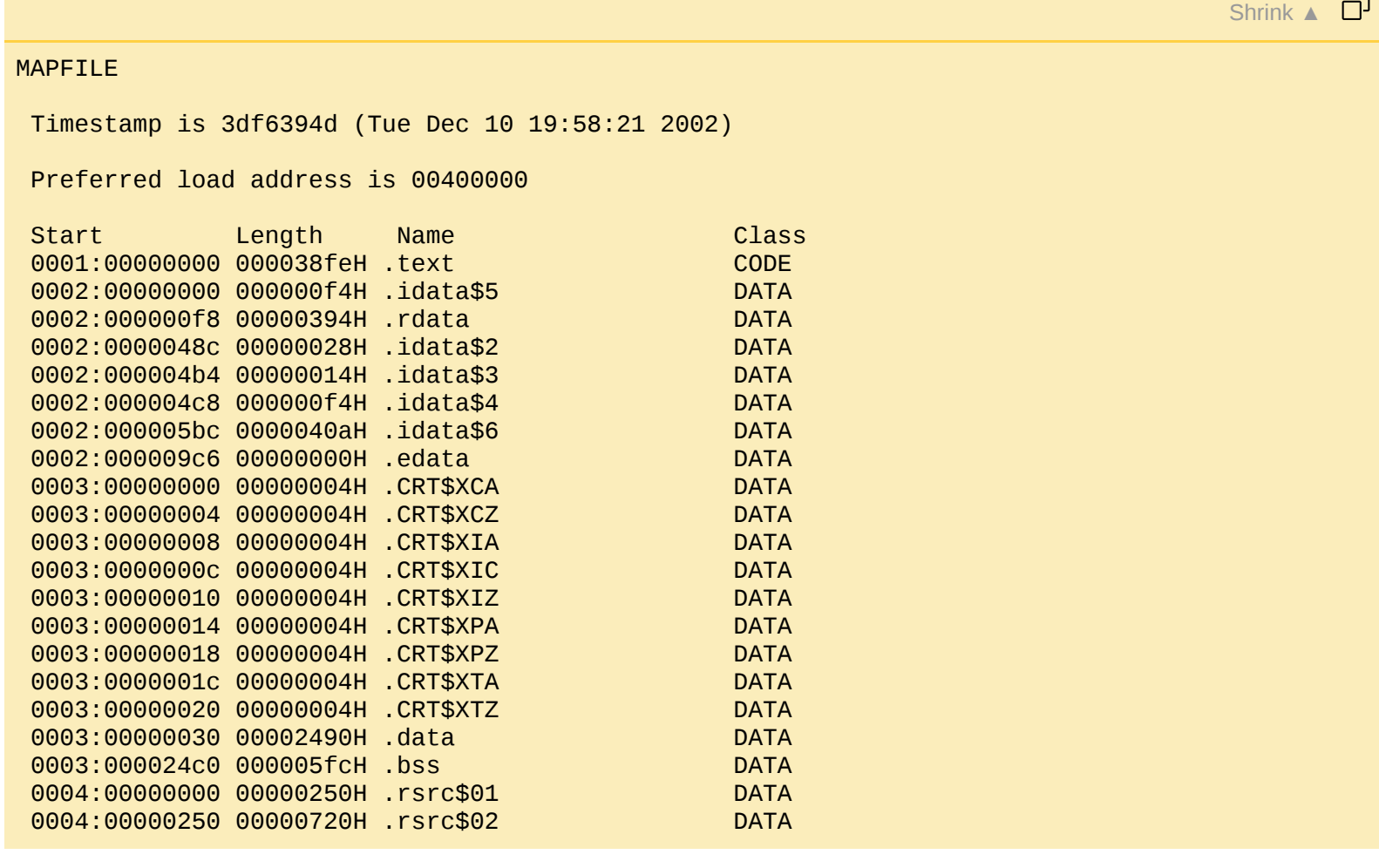
After all this dull work, now comes the neat part: how to read the MAP file. We'll do this by using a crash example. So first: how to crash your application. I did this by adding these two lines at the end of the `InitInstance()` function:

```
C++char* pEmpty = NULL;
pEmpty = 'x'; // 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 "0x004011a1" 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:

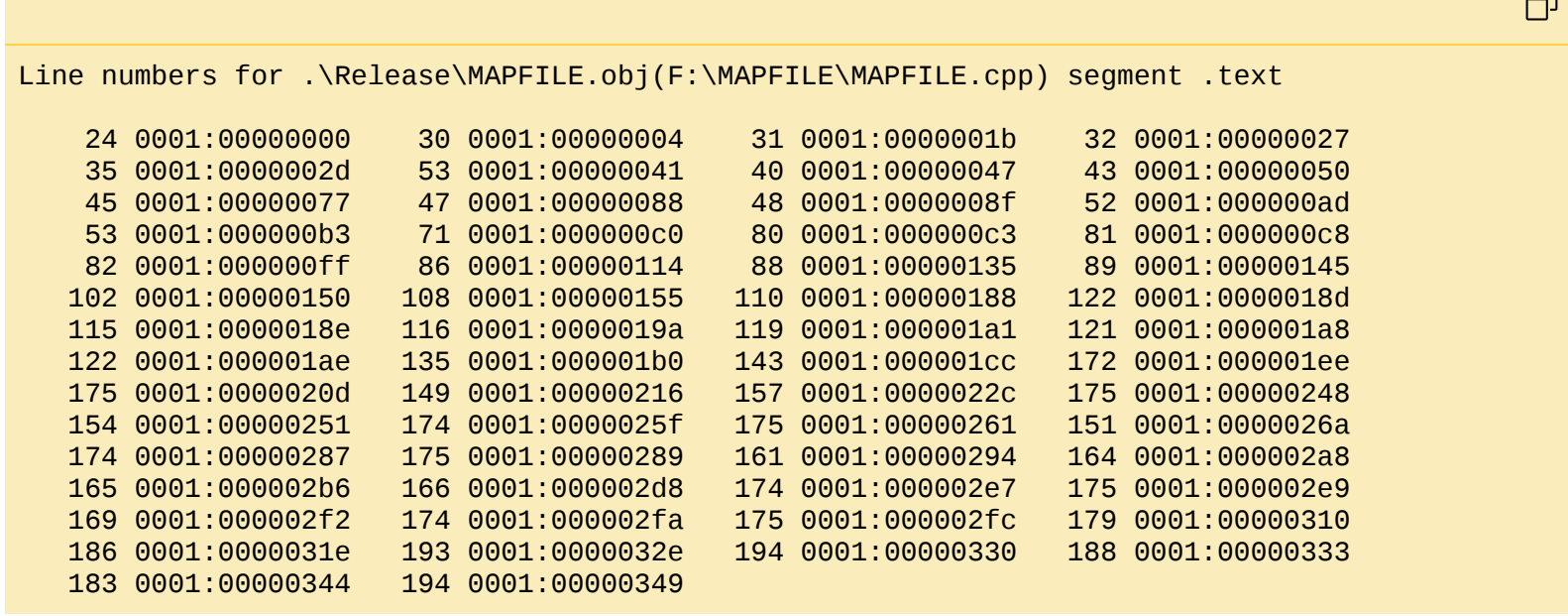
The top of the MAP file contains the module name, the timestamp indicating the link of the project, and the preferred load address (which will probably be 0x00400000 unless you're using a DLL). After the header comes the section information that shows which sections the linker brought in from the various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



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.



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.



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 0x004011a1. This is between 0x00401150 and 0x004011b0 so we know the crash function is ?InitInstance@YAHPAUHINSTANCE_@@@Z. 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 subtract 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: `0x004011a1 - 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?). A user 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

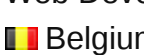
License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

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 Pping as an alternative to the windows ping program (just for his own amusement). Amazed by the response / interest, he founded [kwakkelap.com](#) to ensure a better distribution for the tool. Several other applications have been released since...

Comments and Discussions



You must Sign In to use this message board.

Search Comments

🔍

Spacing

Relaxed

Layout

Normal

Per page

25

Update

First

Prev

Next

Crash map from Open Watcom IDE

Member 14812645

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 win7 ?

ralfsch

4-Nov-09 22:02

Re: Find crash address with win7 ?

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 get exact line number 119

shumonshumon

11-Jan-08 0:44

Re: Couldn't understand how to get exact line number 119

Priya_Sundar

3-Mar-09 18:45

How to add line numbers in visual studio 2005?

moithadar

27-Dec-07 3:21

Re: How to add line numbers in visual studio 2005?

Matk 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

GSCHARYA

19-Oct-05 2:08

Application crashed error

Anonymous

28-May-05 0:17

Last Visit: 31-Dec-99 18:00

Last Update: 30-Aug-23 2:38

Refresh

1 2 3 4

Next >

General

News

Suggestion

Question

Bug

Answer

Joke

Praise

Rant

Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.