

Re: Executable entry points incorrectly documented

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2009-01/msg00103.htm>

- *From:* "m" <m@xxx>
 - *Date:* Sat, 17 Jan 2009 22:28:57 -0500
-

You are correct: this page contains erroneous and confusing information.
The function signature is there though

<snip>

When creating a managed image, the function specified with /ENTRY must have a signature of (LPVOID var1, DWORD var2, LPVOID var3).

</snip>

though it should really be listed like this

```
int __stdcall Entry(LPVOID var1, DWORD var2, LPVOID var3)
```

with a note that a call to ExitProcess is mandatory on some OS versions when using certain APIs (ie gethostbyname on XP & server 2003) because they create threads that never die.

This is where the C language, Win32, as a C API, and implementation details of the platform (ie PE and OS loader) and CRT cross paths. The documentation is sparse and often confusing; but since hardly any programs need to care about the details it isn't likely high on Microsoft's list of things to fix.

"Kornél Pál" <kornelpal@xxxxxxxxxxxxxxxxxxxx> wrote in message <news:e6h5rOKeJHA.3708@xxxxxxxxxxxxxxxxxxxxxxxx>

Hi,

I was speaking about Windows SDK not MSDN Library entirely, but even if Windows SDK contains C/C++ documentation I believe that main/WinMain

Re: Executable entry points incorrectly documented

belong to C/C++ documentation rather than Windows API documentation. (So we agree on this.)

As for the entry point signature:

I am referring to <http://msdn.microsoft.com/en-us/library/f9t8842e.aspx>

If there is a better version please let me know.

It's correct about the default entry point names used by the linker and also correct about what those entry points call and because this is the C/C++ documentation it's entirely correct that it describes the C/C++ runtime behavior this way.

But this is incorrect:

"The parameters and return value must be defined as documented in the Win32 API for WinMain (for an .exe file) or DllEntryPoint (for a DLL)."

Because this is the real entry and because it's possible to use your own entry point form C/C++ (I actually do this whenever I want to avoid CRT dependency) here should be the signature of the function expected by the /ENTRY argument rather than the signature expected by the CRT entry point wrappers. Neither WinMain nor main (not referenced here) qualifies as a real entry point.

DllEntryPoint on the other hand is incorrect because its name is documented and used as DllMain so using DllEntryPoint just confuses the user and makes finding documentation more difficult.

This page does not contain the signature definition of either the EXE entry point or the DLL entry point.

Kornél

m wrote:

Look at /Entry in MSDN. The signature of the entry point is defined and many caveats mentioned.

Also, you are absolutely correct in that main / WinMain have nothing to do with the core of the OS. They are, however, the entry points used by most C/C++ programs and MSDN contains CRT and other documentation besides

the core OS too. In my MSDN anyway, they are found under the Windows Management section and not in Win32 at all.

Re: Executable entry points incorrectly documented

"Kornél Pál" <kornelpal@xxxxxxxxxxxxxxxxxxxx> wrote in message
news:49707D7D.7060801@xxxxxxxxxxxxxxxxxxxx

Hi,

Thanks for your reply.

You wrote that WinMain is not a starting function in CRT,
but an
user-defined entry point for a Windows application.

This is true but this actually should read entry point for a
Windows
application written in C/C++.

I believe that the fact that this function is called by the
C/C++
runtime clearly backs my opinion that WinMain has nothing
to do with the
operating system.

Even if WinMain documentation remains in Windows SDK
it would be wise to
explicitly state that support for this entry point is provided by
the
C/C++ runtime rather than Windows itself.

My bigger problem however is that the real entry point
signature is not
documented anywhere in Windows SDK.

That is the entry point called by Windows in a PE executable
and the
entry point signature expected by the linker. For this reason
documentation for the real entry point is definitely missing.

Some comments to Jeroen's opinion:
The fact that WinMain is supported both by C/C++ and some
Pascal variants
(or any other languages) does not mean that it is an operating
system
feature. And the entry point signature of a PE executable is
definitely
part of the programming interface (call it API or ABI) rather
than an OS
internal.

I agree with him that the Windows SDK's primary target is
C/C++ and all

Re: Executable entry points incorrectly documented

the header files were written for C/C++. On the other hand as long as the application binary interface (ABI) is well defined (and it is) anyone has the chance to write his own compiler or use any compiler (and language) he/she wants. This means that the function signatures specified in C are only a representation of the API.

Jeroen also has the point that macros for example are C/C++ specific but they just provide extra functionality to C/C++ users. Ideally everything that cannot be expressed using a C function or structure signature should be documented using texts. And fortunately this is the case for most of the things.

And I would like to add one more documentation request related to PE files:

HMODULE and HINSTANCE were two different things in Win16 but in Win32 (and Win64) they both are the same and some functions use HMODULE while others HINSTANCE that causes confusion. (See <http://blogs.msdn.com/oldnewthing/archive/2004/06/14/155107.aspx> for a short explanation.)

This HMODULE is also the base address of the module (EXE or DLL) that means the address of the MZ header of the memory mapped image.

These facts are well known and widely used by the developer community there are several articles about them but is not documented by the Windows SDK.

Although – unlike PE entry point signature – this qualifies for being an OS internal, it is so widely known fact and so many programs take advantage of it that I can't imagine that it would change without introducing a new OS API that would break all the

Re: Executable entry points incorrectly documented

applications anyway so

I believe that this should be documented in Windows SDK as well.

Kornél

Jialiang Ge [MSFT] wrote:

Hello

This is an interesting topic. I basically agree with Jeroen. Here is the addition of my opinion.

(w)mainCRTStartup, (w)WinMainCRTStartup are the starting addresses from the C runtime library. They have the same signature of int XXXXX(void). Only one of them will be compiled:

(The following code is quoted from C:\Program Files\Microsoft Visual Studio 9.0\VC\src\crt\crtexe.c)

```
#ifdef _WINMAIN_

#ifdef WPRFLAG
int wWinMainCRTStartup(
#else /* WPRFLAG */
int WinMainCRTStartup(
#endif /* WPRFLAG */

#else /* _WINMAIN_ */

#ifdef WPRFLAG
int wmainCRTStartup(
#else /* WPRFLAG */
int mainCRTStartup(
#endif /* WPRFLAG */
```

The starting function calls to the user-defined entry point (main or WinMain) based on the app type. WinMain is not a starting function in CRT, but an user-defined entry point for a Windows application. MSVC runtime actually passes some Windows arguments to WinMain (see the Windows API GetStartupInfo. GetStartupInfo is not a part of CRT):

Re: Executable entry points incorrectly documented

(The following code is quoted from
C:\Program Files\Microsoft Visual
Studio 9.0\VC\src\crtexe.c)

```
int wWinMainCRTStartup(  
_TCHAR *lpszCommandLine;  
STARTUPINFO StartupInfo;  
BOOL inDoubleQuote=FALSE;  
  
.....  
__try {  
/*  
Note: MSDN specifically notes that  
GetStartupInfo returns no error, and throws  
unspecified SEH if it  
fails, so  
the very general exception handler below is  
appropriate  
*/  
GetStartupInfo( &StartupInfo );  
}  
__except(EXCEPTION_EXECUTE_HANDLER)  
{  
return 255;  
}  
  
.....  
  
mainret = WinMain(  
#endif /* WPRFLAG */  
(HINSTANCE)&__ImageBase,  
NULL,  
lpszCommandLine,  
StartupInfo.dwFlags &  
STARTF_USESHOWWINDOW  
? StartupInfo.wShowWindow  
: SW_SHOWDEFAULT  
);  
  
.....
```

Therefore, I feel that it is reasonable to place
WinMain in Win32
documentation, instead of C/C++
documentation. This is my opinion.
Please feel free to further discuss it with me.

Regards,
Jialiang Ge

Re: Executable entry points incorrectly documented

(jialge@xxxxxxxxxxxxxxxxxxxxxx, remove
'online.')

Microsoft Online Community Support

Delighting our customers is our #1 priority.
We welcome your comments
and suggestions about how we can improve
the support we provide to you.
Please feel free to let my manager know
what you think of the level of
service provided. You can send feedback
directly to my manager at:
msdnmg@xxxxxxxxxxxxxxxxxx

=====
Get notification to my posts through email?

Please refer to

<http://msdn.microsoft.com/en-us/subscriptions/aa948868.aspx#notifications>.

MSDN Managed Newsgroup support
offering is for non-urgent issues where
an initial response from the community or a
Microsoft Support Engineer
within 2 business day is acceptable. Please
note that each follow up
response may take approximately 2 business
days as the support
professional working with you may need
further investigation to reach
the most efficient resolution. The offering is
not appropriate for
situations that require urgent, real-time or
phone-based interactions.

Issues of this nature are best handled
working with a dedicated
Microsoft Support Engineer by contacting
Microsoft Customer Support
Services (CSS) at

<http://msdn.microsoft.com/en-us/subscriptions/aa948874.aspx>

=====
This posting is provided "AS IS" with no
warranties, and confers no
rights.

Re: Executable entry points incorrectly documented