

Re: CRT and Win32 SDK

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2008-05/msg00434.html>

- *From:* "Alf P. Steinbach" <alfps@xxxxxxxx>
 - *Date:* Thu, 22 May 2008 11:15:17 +0200
-

* Norman Bullen:

Alexander Nickolov wrote:

Actually, an implementation of the C runtime library doesn't need to "use the OS".

I think that comment derived from some simple misunderstanding, perhaps by ignoring context.

Nobody's (so far) claimed that a runtime library needs to use an OS.

Consider embedded devices without an OS. It's simply an implementation of the standard-prescribed functions. In reality most such implementations rely on an OS to provide some of the functionality of course.

Ignoring the last sentence, which doesn't make sense (rely on an OS on a platform without an OS?), the above paragraph is true, to some extent.

However, it might give a reader an impression that when doing embedded programming one typically has a full-fledged language at hand, where in effect the runtime library implements what the OS would provide on e.g. a PC.

The C and C++ standards differentiate between hosted and free-standing implementation to deal with the special constraints of embedded programming, where free-standing doesn't need to provide all of the language's standard library, but even that distinction does in practice not go far enough: it might be (as with case of doing "C++" programming for old versions of a popular mobile phone OS) that fundamental language features such as static variables and exceptions are not available on the embedded platform's language implementation.

OP's confusion is that the C runtime library is part of the language and one can't develop without it. It is part of the language standard,

Re: CRT and Win32 SDK

true, but one can happily develop code without using it. It is `_not_` part of the language. Both the user-mode Win32 API and the kernel are indeed written mostly in C and to the best of my knowledge both don't use a single function of the C runtime as defined by the C99 standard.

And here's a "sort of" exception to that statement.

`MoveMemory()`, `CopyMemory()`, `FillMemory()`, and `ZeroMemory()` are documented as Win32 API functions. (A note in the remarks for each equates the names to the corresponding name with "Rtl" prepended.

The documentation remarks are very important parts of that documentation and form the main part of it. These functions are documented as equates, not as API functions. E.g. "This function is defined as the `RtlMoveMemory` function. For more information, see `Winbase.h` and `Winnt.h`."

However, SEH exception handling does suffer the documentation problem you mention, namely misleading documentation, with the documentation claiming that some (extended) C language interface is really part of the API. Not to mention GDI+, where much of the "API" is inline Visual C++ code -- making it at best a compiler-specific (not just a language-specific) "API"!

That is a /documentation/ problem.

```
Sure enough, in WinBase.h from the SDK we find
#define MoveMemory RtlMoveMemory
#define CopyMemory RtlCopyMemory
#define FillMemory RtlFillMemory
#define ZeroMemory RtlZeroMemory
```

```
But now let's look in WinNT.h from the SDK. It contains
#define RtlMoveMemory(Destination,Source,Length) \
  memmove((Destination),(Source),(Length))
#define RtlCopyMemory(Destination,Source,Length) \
  memcpy((Destination),(Source),(Length))
#define RtlFillMemory(Destination,Length,Fill) \
  memset((Destination),(Fill),(Length))
#define RtlZeroMemory(Destination,Length) \
  memset((Destination),0,(Length))
```

The implementation depends on the platform. The above is an optimization for most common platforms. I find

```
#if defined(_M_AMD64)
```

```
NTSYSAPI
VOID
```

Re: CRT and Win32 SDK

```
NTAPI
RtlCopyMemory (
VOID UNALIGNED *Destination,
CONST VOID UNALIGNED *Source,
SIZE_T Length
);

#else

#define RtlMoveMemory(Destination,Source,Length) \
memmove((Destination),(Source),(Length))
```

One needs to keep in mind that the true API is language independent, and can be used from languages such as Pascal that do not have preprocessors and certainly not any C runtime library.

This means that the supposed API functions are actually implemented by C-RTL functions which are linked into the user's program or compiled in line, depending on compiler options.

Yes, it's just a documentation or Microsoft terminology problem.

Presumably, these functions were also used by Microsoft programmers when writing components of the OS

Could be, but then necessarily with either inline code generation or binding to OS implementation (such as indicated above for `_M_AMD64`).

therefore these C-RTL functions are used in the OS.

Nope.

Except in some meaningless rhetorical sense, of course. :-)

Cheers,

- Alf

--

A: Because it messes up the order in which people normally read text.

Q: Why is it such a bad thing?

A: Top-posting.

Re: CRT and Win32 SDK

Re: CRT and Win32 SDK

Q: What is the most annoying thing on usenet and in e-mail?

.