Availability of Low-Level APIs

This thread has been locked by a moderator.

231

I regularly see folks confused by this point so I decided to write it up in a single place.

If you have questions or comments about this, start a new thread here on DevForums. Tag your thread with something appropriate for the API you're trying to use. If nothing leaps out, *Kernel* is a good option [1].

IMPORTANT I don't work for App Review and can't make definitive statements on their behalf. All of the following is about whether an API is available for you to use, not whether App Review will approve your specific usage.

Share and Enjoy
—
Quinn "The Eskimo!" @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

[1] Because of the KPI aspect of this, discussed below.

Availability of Low-Level APIs

Every now and again I see questions like this:

The developer documentation has no entry for `getpid`*. How can that not be API?*

Or this:

I want to call `mach_absolute_time` *on iOS. Its documentation says that it's only available on macOS. But it's in the iOS SDK and it works just fine. Is it OK for me to use it in my iOS app?*

These questions arise because:

- Apple Developer Documentation focuses on Apple's frameworks.
- Most low-level APIs, specifically those with a BSD heritage, are documented in man pages. These man pages aren't available in Apple Developer Documentation (r. 16512537). For information about how to access them, see Reading UNIX Manual Pages.
- Some low-level APIs are documented in both man pages and Apple Developer Documentation. A classic example of this is Dispatch, which has comprehensive man pages (start at `dispatch 3`) and good documentation in Apple Developer Documentation.
- Some low-level APIs, like Compression, are documented in Apple Developer Documentation but have no man pages.
- Some low-level APIs aren't documented in either Apple Developer Documentation or the man pages. A classic example of this is SQLite. In such cases the best documentation may be the doc comments in the APIs headers, or on a third-party website.
- For Mach APIs, you often find that the best documentation is in the Darwin open source!
- Some low-level APIs have equivalent KPIs (Kernel Programming Interfaces). For example, `mach_absolute_time` is both an API and a KPI. Many KPIs are documented in Apple Developer Documentation, in a special area known as the Kernel framework [1]. Kernel development is only supported on macOS, so those KPIs are flagged as being macOS-only. However, their equivalent APIs are typically available on Apple's other platforms.

These questions most often crop up in the context of obscure low-level APIs, but many obviously valid low-level APIs have the same issue and no one worries about those. For example, Apple Developer Documentation has no documentation for the `printf` API [2], but no one asks whether it's OK to use `printf`!

Given the above, it's clear that you can't infer anything about the availability of an API based on its state in Apple Developer Documentation. Rather, the acid test for availability is:

- Is it declared in a platform SDK header?
- Is it not marked as unavailable on that platform? [3]
- Does the platform SDK have a stub library [4] that makes its symbol available to the linker?

If the answer to all four questions is "yes", the API is available on that platform.

[1] The kernel does not support frameworks but bundling these KPIs into `Kernel.framework` within the macOS SDK makes some degree of sense from a tooling perspective, and that logic flows through to the documentation.

[2] There's a documentation page for the KPI, but that's not the same thing.

[3] Sorry for the double negative but it's the only way to capture an important subtlety. If the header contains a declaration with no availability markings, that API is available on all platforms. A classic example of this is `printf`.

[4] If you're not familiar with the term *stub library*, see An Apple Library Primer.

Hints and Tips

In general, prefer high-level APIs over low-level ones. For example, prefer `Date`, or even `gettimeofday` or `clock_gettime`, over `mach_absolute_time`. However, that's only a general guideline. In some cases the low-level API really is the right one to use.

Just because something is an API doesn't mean that there aren't any restrictions on it. `mach_absolute_time` is a perfect example of this. Using it for highly accurate performance analysis of your code is fine, but using it for fingerprinting is not. See Describing use of required reason API.

If you can't find adequate documentation for an API you're using, always look in the headers for doc comments. In some cases that's the only source of documentation. However, even if the API is reasonably well documented, the headers might contain critical tidbits that slipped through the cracks.

Developer Tools Kernel

Reply

Posted 2 months ago by eskimo

Add a Comment