

## Re: Anyone object to the following change in libc?

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/current/2003-10/1558.html>

---

*From:* Terry Lambert ([tlambert2\\_at\\_mindspring.com](mailto:tlambert2_at_mindspring.com))

*Date:* 10/31/03

Date: Fri, 31 Oct 2003 00:42:09 -0800

To: Harti Brandt <[brandt@fokus.fraunhofer.de](mailto:brandt@fokus.fraunhofer.de)>

Harti Brandt wrote:

> *Section 7.19.6.7 of N843 states:*

>

> *"Reaching the end of the string is equivalent to encountering end-of-file*

> *for the fscanf function."*

OK, I buy this one. 8-).

> *Unfortunately this is missing in POSIX, but obviously implied by their*

> *reference to ISO.*

I don't know if we can say the ISO C standard has precedence here; my feeling is that precedence is dictated by the validation suite for IEEE 1003.1-2001. I don't have a copy of this handy, so I can't tell you whether the answer is 0, -1, or "don't care", at this point.

One important point to note here is that a POSIX compliant OS need not be implemented in the C language, and that the library routines described are really C language bindings, and really mean their equivalents. In that case, the ISO C standard can go jump in a lake. 8-). I think that we need a ruling from someone who has a system that passes conformance testing... so you can't go wrong doing as Solaris does, but that doesn't mean it's something that's actually tested. 8-(.

> *TL>How do I distinguish a "return value is -1 as an error result" from*

> *TL>"return value is -1 as an EOF result"?*

>

> *Well, I suppose that's the intention of having scanf() setting errno*

> *when it returns -1 in POSIX. Unfortunately POSIX fails to describe*

> *the error codes. This is possibly fodder for the aardvark.*

I was afraid you were going to say that: that it's one of those terrible functions where you have to set errno equal to 0 and then check it when you get a -1 to see if it's still 0, in order to be able to distinguish the cases. 8-(.

freebsd-current: Re: Anyone object to the following change in libc?

- > TL>It says that any conflicts are unintentional, and their intent was
- > TL>to use different language for no good reason, rather than just
- > TL>copying it verbatim and removing any doubt. It does *\*NOT\** say
- > TL>that no conflicts exist.
- >
- > Yes. But I take the last sentence to mean that ISO-C takes over in the
- > case a conflict exists.

I don't think you can say exactly that without them saying exactly that. The standard is the standard; while they might regret not aligning with ISO C, I think that implementations have to adhere to the letter of the standard to be considered conformant. There's lots of stupid things in the standard that we just have to live with in order to be conformant.

This is actually recognized in conformance testing, which lets you set an environment variable to indicate to your libraries that they need to return strictly conformant results, rather than The Right Results(tm). I think that's the escape hatch for differences in details between them and other standards, so that system can claim standards compliance with multiple standards.

Yes, it's ugly.

- > TL>Also: In this context, which is IEEE 1003.1-2001, Issue 6, "the
- > TL>ISO C standard" refers to "c89", which is the version of the C
- > TL>standard that was in effect at the time that SVID IV was defined.
- >
- > Line 107 of Austin TC-1:
- >
- > "The c89 utility (which specified a compiler for the C Language specified
- > by the 108 ISO/IEC 9899: 1990 standard) has been replaced by a c99 utility
- > (which specifies a compiler for 109 the C Language specified by the
- > ISO/IEC 9899: 1999 standard)."

Again, that doesn't effect IEEE 1003.1-2001.

I can have software that conforms to RFC 821 and fails to conform to RFC 2821, and it's still RFC 821 compliant, even if the IETF wants everyone to switch over to using RFC 2821 instead, and RFC 2821 says "Obsoletes: 821, 974, 1869".

- > TL>If you need clarification on this issue, you should download the
- > TL>currently available version of the NIST/PCTS, which specifically
- > TL>requires you to compile with a c89 compiler, not one more recent.
- > TL>The same is true of The Open Group test suites which are available
- > TL>on the Internet.
- > TL>
- > TL>The version of the ISO C standard you are quoting from is *\*NOT\**
- > TL>the c89 version.
- >

Re: Anyone object to the following change in libc?

frebsd-current: Re: Anyone object to the following change in libc?

- > *Our sscanf() claims conformance to C99. So if we change the behaviour*
- > *we have to remove this claim.*

No... or we need to use inline versions in the header which call into "\_sscanf\_c89()" and "\_sscanf\_c99()" functions in the implementation namespace, variant upon which standard is in effect, e.g. via (GCC-only syntax, so it would need to be wrapped for any non-gcc compilers, as well):

```
#if (__STDC_VERSION__ - 0) > 199409
#define sscanf(str, fmt, ...) _sscanf_c99 (str, fmt, __VA_ARGS__)
#else
#define sscanf(str, fmt, ...) _sscanf_c89 (str, fmt, __VA_ARGS__)
#endif
```

This will work for all code, even code that's linked together from objects compiled with different versions of the standard in scope, since each scope will use the scope-specific definition.

- > *TL>In any case, we are practically guaranteed that returning -1, as*
- > *TL>all other UNIX-like OS's currently do, would result in less source*
- > *TL>code breaking.*
- >
- > *No coder in his right mind should have written code that depends*
- > *on this behaviour given the moot formulations in the classical books,*
- > *man pages and pre-C99 standards.*

That's what they said about all the things programmers did before the ANSI C standard changed the language to make certain ambiguous constructs illegal because programmers were doing exactly that (I guess they might have been out of their right minds ;^)).

- > *Also note, that the reason for*
- > *this change request was that configuration scripts break, not applications.*
- > *If applications break they should be fixed.*

The configuration scrips in this particular case *are* applications (albiet very small ones which aren't kept around for use by the final software).

- > *TL>In other words, conformance level has historically been dictated*
- > *TL>by what code is not broken, not what is technically permitted by*
- > *TL>the standards, if you language-lawyer them to death.*
- > *TL>*
- > *TL>To put it in IETF terms: "Be conservative in what you generate,*
- > *TL>and generous in what you accept".*
- >
- > *This does not apply here because you cannot return -1 and 0 at the same*
- > *time.*

Re: Anyone object to the following change in libc?

freebsd-current: Re: Anyone object to the following change in libc?

But you can do it conditional on the standards conformance level,  
at compile time...

- > *Adhering to a cleanly written standard and breaking a handful of*
- > *badly written autoconf scripts is clearly better than adhering to*
- > *undocumented historical behaviour. What will we do if Solaris 10*
- > *returns 0 in the above case? Change our code back?*

It depends on what the validation suite for the IEEE 1003.1-2001 tests for, if anything, I think. Solaris will not break their conformance testing and pay the fine for the variance merely to adhere with the ISO C standard (a standard with no teeth to bite their wallet).

Adhering to the IEEE 1003.1-2001, according to the validation suite, is a heck of a lot more important than adhering to the ISO C standard, when it comes to attracting customer dollars.

I think someone with access to the suite needs to post whether or not it tests this particular area for conformance, and, if it does, what it expects to see as a result (and if not, we should feel free to return an int-coerced pointer a string containing "not a penguin", or whatever).

-- Terry

---

freebsd-current@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-current>

To unsubscribe, send any mail to "freebsd-current-unsubscribe@freebsd.org"