SC22/WG14 N852

Editor’s Report

September 24, 1998

1. Status

CD2 was completed and delivered to SC22 as a final CD the first week of August along with the disposition of comments document. In addition to the substantive changes made in response to public comments, there were also many small editorial changes. The changes are too numerous to list here, please see the version of CD2 with diff marks from the previous draft (available from the committee’s web site at http://wwwold.dkuug.dk/JTC1/SC22/WG14/) for details.

The SC22 FCD ballot runs from 10 August 1998 to 8 January 1999; for details, see http://wwwold.dkuug.dk/JTC1/SC22/open/n2794.htm. SC22 published only the PDF and text versions of the draft, but the PostScript version is available from the committee’s web site.

The US public review runs from 11 September 1998 to 10 November 1998; details are at http://www.ncits.org/prsweb.htm. As far as I know, the US is still willing to accept public comments from anyone, not just citizens or residents.

2. CD2 Errata

General

Footnote 74 is missing from the text version of the draft.

There is extra whitespace following each example.

Foreword

The foreword should note that this standard is a revision of ISO/IEC 9899:1990 and includes AM1, TC1, and TC2. It should also contain a list of major technical changes.

6.2.5 Types

Paragraph 22: ‘‘*type*domain’’ should be ‘‘*type domain*’’.

6.3.2.3 Pointers

Paragraph 6: There needs to be special dispensation for **`_Bool`** since the result in that case is not implementation-defined.

6.4.4.4 Character constants

Paragraph 8: UCNs are not limited to graphic characters, they may also be used for various kinds of space characters.

6.5.9 Equality operators

Paragraph 6: ‘‘if’’ should be ‘‘if and only if’’.

6.7.8 Initialization

Paragraph 20 doesn’t take designated initializers into account. In the first line, ‘‘aggregate’’ should be ‘‘aggregate or union’’ and the phrase ‘‘or if the first member of a union is an aggregate or union’’ should be deleted. In the fifth line, the phrase ‘‘the first member of the’’ should be deleted.

6.10.8 Predefined macro names

Footnote 137: "ISO/IEC 9899:AMD1:1995" should be "ISO/IEC 9899/AMD1:1995".

7.11.2.1 The `localeconv` function

In the example, the **`currency_symbol`** for the Netherlands should not have a trailing space and all of the **`int_*_sep_by_space`** entries should be **`0`**.

7.16 Boolean type and values `<stdbool.h>`

Paragraph 3: "decimal constant" should be "integer constant".

7.19.6.1 The `fprintf` function

Paragraph 6: The descriptions of the **`#`** and **`0`** flags should note that, for floating-point formats, they only apply to finite values and not to infinities or NaNs.

This also applies to 7.24.2.1 The **`fwprintf`** function.

7.23.2.6 Normalization of broken-down times

Paragraph 3: The computation of **`D`** is incorrect. The first line should be:

```
D = Y*365 + QUOT(Z, 400)*97 + REM(Z, 400)/4 - MOD(Z, 400)/100 +
```

A.2.1 Expressions

(6.5.2) *postfix-expression*: The last two lines have punctuation characters in the wrong font.

Annex D Formal model of sequence points

This annex uses the term "field" to refer to structure and union members; it should use the term "member" instead.

3. Open Issues

The index still needs more work.

The text version of the draft has not been reviewed and contains numerous formatting problems such as superscripts overprinting information from the previous line and unintelligible expressions.

There are still some typesetting issues to be addressed, particularly in annex F.

All hyphenated terms should be examined to determine where the hyphenation is appropriate and where it isn't. All italicized terms should also be examined.

All of the cross references need to be checked for relevance and accuracy.

The draft still needs some work to fully comply with Part 3 of the ISO Directives. The major issues remaining are:

— the formatting of tables

— we have many subclauses that contain text in addition to subordinate subclauses; ISO strongly discourages this

— all references to a specific version of a standard must include the date; it is not sufficient to include the date only in the Normative references or the Bibliography as we currently do

— annexes are required to appear in the order in which they are cited in the text (and thus there has to be a citation for each annex); if anyone has any strong feelings about ordering, please let me know

— ISO requires numbers to be formatted with a comma for the decimal-point character and digits before and after the decimal-point character to be in space-separated groups of three; I believe we have more than adequate justification for ignoring this requirement

4. My comments

5.2.4.1 Translation limits

How should a **`for`** loop be counted against the nesting level limit? It is one iteration statement, but 6.8.5.3 says it's equivalent to a sequence of statements that contains two blocks and a different iteration statement. So, does it count as one, three, four, or something else?

6.2.1 Scopes of identifiers

Paragraph 3 says that label names shall be unique within a function; why isn't this a constraint?

6.3.1.5 Real floating types

Should we add words to paragraph 2 noting that the same thing happens when a value with excess range or precision is converted to its overt type?

6.5.16.1 Simple assignment

Pointers can be converted to **`_Bool`** but they're not assignment compatible, so it requires an explicit cast. Given that pointers are frequently used in boolean contexts, I believe we should make them assignment compatible.

6.7.5.3 Function declarators (including prototypes)

Describing how an identifier in a parameter declarator that could be taken either as a typedef name or as a parameter name is interpreted has been a long-standing problem. C90 specified the interpretation in some specific contexts but left the interpretation in other contexts unspecified. In trying to better specify this for C9X, we now say that if the identifier ''can'' be treated as a typedef name, it is, but it's not clear how ''can'' is to be interpreted in this context. Consider the following fragment:

```
typedef int what;

int f(int (what)(int));
```

In the declaration of **`f`**, **`what`** ''can't'' be be treated as a typedef name because doing so would violate the constraint that a function not return a function. But determining that requires more than one token of look-ahead so very few, if any, compilers actually work that way and I don't think there's any value in requiring them to. I don't have any suggestions for resolving this issue, it's something we need to think about.

7.1.4 Use of library functions

What, exactly, are the constraints on library macros and on macro versions of library functions? Footnote 144 mentions that they do not have to have the same sequence points that an actual function call would, but what about parameter type checking and conversion? I believe that both should be required — both are an integral part of the interface — but the response to DR 107 says differently. The standard should be clarified one way or the other.

7.4.1.8 The `ispunct` macro

In C89, for every character for which **`isprint`** was true, either the character was the space character (`' '`), **`isalnum`** was true, or **`ispunct`** was true. This is no longer true in C9X (for good reason, in the general case), but it is a somewhat gratuitous change for the C locale. I believe we should restore the C89 behavior, but just for the C locale.

7.11.2.1 The `localeconv` function

The description of the `*_sep_by_space` members of `struct lconv` still does not accurately reflect the intent as illustrated by the Posix table. Unfortunately, the table appears to be incorrect as well since there are two entries that disagree with the implied rules. One is not particularly important, but the other shows a serious oversight: while it is possible to specify the formats `$1.25+`, `$ 1.25+`, `$1.25 +`, `+1.25$`, and `+1.25 $`, it is not possible to specify the format `+ 1.25$`.

I believe the correct description of `*_sep_by_space` is:

`0` No space separates the currency symbol and value.

`1` If the currency symbol and sign string are adjacent, a space separates them from the value; otherwise, a space separates the currency symbol from the value.

`2` If the currency symbol and sign string are adjacent, a space separates them; otherwise, a space separates the sign string from the value.

I have come to believe the table should also be included in the standard as an example, since it makes it easier to understand the specifications and has been widely published (both as part of Posix and as part of the Single Unix Specification). Based on the above, the correct table is (the two entries which are different than Posix are highlighted):

		p_sep_by_space		
p_cs_precedes	p_sign_posn	0	1	2
0	0	(1.25$)	(1.25 $)	> (1.25$) <
	1	+1.25$	+1.25 $	> + 1.25$ <
	2	1.25$+	1.25 $+	1.25$ +
	3	1.25+$	1.25 +$	1.25+ $
	4	1.25$+	1.25 $+	1.25$ +
1	0	($1.25)	($ 1.25)	($1.25)
	1	+$1.25	+$ 1.25	+ $1.25
	2	$1.25+	$ 1.25+	$1.25 +
	3	+$1.25	+$ 1.25	+ $1.25
	4	$+1.25	$+ 1.25	$ +1.25

Also, I question whether any of the new `int_*` members are necessary. While I can certainly believe that many countries use different local and international formats, it seems that everyone uses the same international format. If that is true, there is no reason to parameterize it.

7.20 General utilities <stdlib.h>

Paragraph 3: There was a suggestion in UK0067 that `EXIT_FAILURE` and `EXIT_SUCCESS` be integer constant expressions and that `MB_CUR_MAX` be of type `size_t`. We should consider these changes.

7.20.1.3 The `strtod`, `strtof`, and `strtold` functions

The expected form for a hexadecimal floating-point number insists that either a decimal point or a binary exponent part appear, but we have never required the same for decimal floating-point. While this was presumably done for backwards compatibility, it is not necessary as a hexadecimal integer starting with `0X` would be taken as a decimal `0` and the remaining character would not be a valid hexadecimal integer. We should remove the restriction.

This also applies to 7.24.4.1.1 The `wcstod`, `wcstof`, and `wcstold` functions.

7.20.5 Searching and sorting utilities

Is it valid to call **`bsearch`** or **`qsort`** with **`nmemb`** equal to zero? The current wording implies that it is not since **`nmemb`** is described as the number of elements in an array, which cannot be zero in C.

If it is valid, is **`base`** allowed to be a null pointer in that case? I presume not.

-- Larry Jones