SC22/WG14 N804

Editor's Report

January 30, 1998

This is a revised version of N799 incorporating corrections and additions. The diff marks indicate changes from the original version.

1. Status

As you know, an editorial committee consisting primarily of Benito, Farance, Jaeschke, Jones, MacDonald, and Walls (with the assistance of Thomas, Gwyn, and others) met in Santa Cruz November 17–20 to prepare the draft for CD ballot. After much work at the meeting and some subsequent formatting clean-up, a final document was prepared on November 24 and forwarded to SC22. That document has since been distributed to the members of SC22 and made available on SC22's web page. Various member bodies have announced its availability and solicited public comments. A preliminary version of the draft was distributed (via the committee's web page) as N794; this document has also been circulated outside the committee as well as within it. Both documents have generated a fair amount of discussion on the comp.std.c newsgroup and the committee reflector as well as some private e-mail pointing out errors and omissions. The remainder of this report is based on these comments as well as my own review.

2. CD1 Errata

General

All of the headings are too small. When the size of the body text was increased, the sizes of the headings were not increased to match.

None of the tables are properly formatted in the text version due to a production error. (The text version has not been reviewed and undoubtedly contains many other formatting problems.)

There are many typesetting issues still to be addressed. For example, there are many equations typeset as code and vice versa, particularly in annexes F and G. Some of these issues primarily affect the typeset version and some primarily affect the text version.

Clause 2, Paragraph 1, Page 2

Although one of the references is a technical report, the introductory text refers only to standards; it should be revised to include TRs.

The hyphens in the standards' designations should be dashes. This is also true for other references in the text, particularly annex A.

The correct date for ISO 4217 is 1995, the correct date for ISO/IEC TR 10176 is 1998 (also annex A).

Clause 3, Paragraph 2, Page 3

The reference to ISO 2382−1 should be to ISO/IEC 2382−1.

3.18, Paragraph 3, Page 6

An implementation need not successfully translate a given program if it exceeds a translation limit.

5.1.1.2, Paragraph 1, Page 9

The term "universal-character-name" should not be hyphenated. This is also true for all other occurrences (other than in the syntax).

Since we now allow concatenation of character and wide string literals, phase 6 should be rewritten to simply:

> Adjacent string literal tokens are concatenated.

5.1.1.2, Forward References, Page 10

There should be a forward reference to 5.2.1 for universal character names.

5.1.2.3, Example 6, Page 16

"However" (4th line from the end) should be followed by a comma.

5.2.1.1, Footnote 12, page 19

The reference to ISO/IEC 646:1991 should be to ISO 646.

5.2.4.2.2, Paragraph 6, Page 26

The list of values should be indented like the one in paragraph 5.

6.1 and 6.2, Pages 31−66

The reorganization specified in N672 has not been applied.

6.1.2, Paragraph 2, Page 34

The description of nondigit characters should mention other characters specified by universal character names.

The reference to annex H should be to annex I.

6.1.2.5, Footnote 29, Page 40

In the second line, "alternate" should be "alternative" and "designed" should be "designate". I suggest rewording the first two sentences to:

> An implementation may define new keywords that provide alternative ways to designate a basic (or any other) type; this does not violate the requirement that all basic types be different.

6.1.2.5, Footnote 30, Page 40

In the last line, there should not be a comma after "two" and "it" should be "is".

6.1.2.6, Paragraph 1, Page 42

Near the end of the fifth line, the period after "following requirements" should be a colon.

6.1.4, Paragraph 2, Page 54

The terms "character string literal" and "wide string literal" should be in italics.

6.1.4, Paragraph 6, Page 55

Although this paragraph doesn't contain the magic word "unspecified", whether string literals are distinct or not is unspecified and should be added to K.1.

6.1.5, Paragraph 1, Page 55

The definition of operator is missing `{` and `}`, although it does have the alternative spellings `<:` and `:>`. They should also appear in the constraint. (Also annex B.)

6.1.6, Paragraph 1, Page 56

The definition of punctuator is missing `..`. (Also annex B.)

6.2.1.1, Paragraph 2, Page 60

The period after ''may be used'' should be a colon.

6.2.1.3, Footnote 48, Page 61

In the last line, there should be a space after the comma.

6.3.1.1, Page 69

In the heading, there should be a thin space between the underscores.

6.3.2.2, Paragraphs 5−6, Page 72

These paragraphs are almost impossible to read correctly. At the very least, paragraph 6 should be a continuation of paragraph 5, not a new paragraph. My suggestion is to move paragraph 6 into paragraph 5 between the third and fourth sentences. I also suggest moving paragraph 7 after paragraph 8.

6.3.2.5, Example 5, Page 78

In the last line of code, there should be a space before the square brackets. Likewise for example 6.

6.3.3.3, Paragraphs 2−4, Page 81

In each paragraph, ''the integer promotion is'' should be ''the integer promotions are''.

6.3.6, Paragraph 12, Page 87

In line 2, ''know constant size'' should be ''known constant size''.

6.3.15, Paragraphs 4−6, Page 93

The last clause of paragraph 4 should read:

> the result of the operator is the value of the second or third operand (whichever is evaluated), converted to the type described below.

The first sentence of paragraph 5 should read:

> If both the second and third operands have arithmetic type, the type that the usual arithmetic conversions would yield if applied to those two operands is the type of the result.

Paragraph 6, beginning with the second sentence, should read:

> Furthermore, if both operands are pointers to compatible types or differently qualified versions of compatible types, the result type is a pointer to an appropriately qualified version of the composite type; if one operand is a null pointer constant, the result has the type of the other operand; otherwise, one operand is a pointer to **`void`** or a qualified version of **`void`**, in which case the result type is a pointer to an appropriately qualified version of **`void`**.

6.5, Paragraph 1, Page 100

The last production for *declaration-specifiers* should read:

> *function-specifier declaration-specifiers*$_{opt}$

6.5.2, Paragraph 4, Page 103

In the middle of the second line, ''specified'' should be ''specifier'' and ''is the same type'' should be ''designates the same type''.

6.5.2.1, Footnote 91, Page 105

The remnants of implicit int should be removed; the body should read:

> As specified in 6.5.2 above, if the actual type specifier used is **`int`** or a typedef-name defined as **`int`**, then it is implementation-defined whether the bit-field is signed or unsigned.

6.5.2.3, Paragraph 3, Page 108

In the second line, ‘‘complete’’ should be ‘‘incomplete’’.

6.5.2.3, Paragraph 6, Page 109

The declaration is missing the trailing semicolon and, in the last line, ‘‘structure of union’’ should be ‘‘structure or union’’.

6.6.4.1, Paragraph 3, Page 142

In the first line, ‘‘preceeding’’ should be ‘‘preceding’’. (And I’d be grateful if anyone could explain to me why ispell doesn’t complain about it.)

6.6.6.1, Example 2, Page 146

In the middle of the block of code, the line ‘‘**`goto lab 4;`**’’ should be ‘‘**`goto lab4;`**’’.

6.6.6.2, Paragraph 2, Page 147

The statements in the code blocks are not properly aligned.

6.8.8, Paragraph 2, Page 171

The second line should not appear and the remainder of the paragraph should be formatted as definitions as in the previous paragraph.

6.8.9, Paragraph 1, Page 172

In the third line, the period after ‘‘follows’’ should be a colon and ‘‘string literal’’ should not be italicized.

6.8.9, Examples, Paragraph 2, Page 172

In the **`#pragma`**, ‘‘**`list`**’’ should be ‘‘**`listing`**’’.

7.1.1, Paragraph 1, Page 174

The quoted terms should all be italicized (cf. paragraph 6).

7.1.2, Paragraph 1, Page 175

In the last line, ‘‘explicity’’ should be ‘‘explicitly’’.

7.1.7, Page 179

Still need final words for a real boolean type (N738 and part of N743 were adopted in principle, but no final words were ever produced).

7.1.7, Paragraph 2, Page 179

In the last line, ‘‘for’’ should be ‘‘of’’.

7.1.7, Footnote 138, Page 179

In the second paragraph, the quotes around ‘‘masked’’ should be real quotes, not double-quotes.

7.1.8, Paragraph 3, Page 181

"return" should be "returns". (Also annex C.)

7.2.1.1, Paragraph 2, Page 183

In line 4, "name of the source file, and the source line number" should be "name of the source file, the source line number, and the name of the enclosing function".

7.3.1.3, Page 186

The isblank function was not approved by the committee and should not have appeared in the draft.

7.4, Paragraph 1, Page 190

There should be a footnote pointing to future library directions (7.20.3).

7.4, Paragraph 3, Page 190

Starting at the end of the first line, "character constants" should be "constants".

7.4.5, Paragraph 1, Page 199

The reference to footnote 151 should be to 149 instead. Since footnote 149 is so far away, I suggest replicating it here or moving this subclause to immediately after 7.4.2.

7.5, Paragraph 3, Page 202

The reference to footnote 153 should be moved to the end of the sentence.

7.5.1.1, Paragraph 2, Page 203

In the very last line, there should be a reference to **`strfxtime`** as well as **`strftime`**. Also a forward reference.

7.7.8.3, Paragraph 2, Page 236

In the last line, "too large" should be "too large or too small".

7.7.9.3, Paragraph 2, Page 238

The parenthetical sentence at the end of the paragraph should be included in the previous sentence and the hyphen should be the word "and".

7.8, Paragraph 1, Page 249

There should be a footnote pointing to future library directions (7.20.8).

7.8.2.23, Paragraph 2, Page 258

There should be a reference back to footnote 188.

7.10.2.1, Paragraph 2, Page 264

The comma near the end of the second line should be deleted. It implies that you can only longjmp to the most recent setjmp, which is clearly not correct.

7.11, Paragraph 3, Page 266

The portion of the paragraph between the two lists should read:

> which expand to constant expressions with distinct values that have type compatible with the second argument to, and the return value of, the **`signal`** function, and whose values compare unequal to the address of any declarable function; and the following, which expand to positive integer constant expressions with type **`int`** and distinct values that are the signal numbers...

7.11.1.1, Paragraph 3, Page 267

In the fourth line, ‘‘occuring’’ should be ‘‘occurring’’.

7.13.2, Paragraph 5, Page 277

In the first line, there should not be a semicolon before ‘‘and’’.

In the third line, the phrase ‘‘and are not affected by’’ should be set off by commas.

7.13.2, Forward references, Page 278

The entries should be the actual section titles, not just the names of the functions.

7.13.3, Paragraph 11, Page 279

‘‘**getwc**’’ should be ‘‘**fgetwc**’’.

7.13.3, Forward references, Page 280

The reference for ‘‘conversion state’’ should be 7.19.6, for **mbrtowc** should be 7.19.6.3.2, and for **wcrtomb** should be 7.19.6.3.3.

7.13.6.1, Paragraph 3, Page 288

In the list item beginning ‘‘An optional **hh**...’’, the words for **hhn** have been omitted.

7.13.6.1, Paragraph 3, Page 291

In the description of the **c** conversion specifier, at the end of the second line, ‘‘Otherwise’’ should be ‘‘If an **l** qualifier is present’’ and should start a new paragraph (cf., the **s** conversion specifier).

7.13.6.1, Footnote 213, Page 292

The reference should be to 7.20.6.

7.13.6.1, Paragraph 14, Page 293

The heading should read "Environmental limits".

7.13.6.2, Paragraph 11, Pages 297−298

In the descriptions of the **s**, **[**, and **c** conversion specifiers, there should be a paragraph break before the sentence beginning ‘‘If no **l** qualifier is present...’’.

The reference to footnote 216 should be replicated in the descriptions of the **[** and **c** conversion specifiers.

7.13.6.2, Footnote 216, Page 297

There should be an ‘‘and’’ before **c**.

7.13.6.2, Footnote 217, Page 299

The reference should be to 7.20.6.

7.13.6.2, Example 3, Page 301

The last block of code is too wide; it should be reformatted to fit within the margin.

7.13.7.11, Paragraph 5, Page 314

There should be a footnote at the end of the paragraph pointing to ‘‘future library directions’’ (7.20.6).

7.14, Footnote 221, Page 321

The reference should be to 7.20.7.

7.14.2.1, Paragraph 5, Page 330

The heading should read "Environmental limits".

7.14.5, Footnote 227, Page 336

The last line of the footnote is missing. It should read:

```
(char *)p < (char *)base + nmemb * size
```

7.15.1, Footnote 232, Page 345

The reference should be to 7.20.9.

7.16.1, Paragraph 1, Page 357

In the first line, ‘‘four types and several functions’’ should be ‘‘several types and functions’’.

7.16.1, Paragraph 3, Page 357

The wording for **struct tm** and **struct tmx** should be revised to clarify that both hold broken-down times.

7.16.2.3, Paragraphs 5–6, Page 360

The last sentence of paragraph 5 and all of paragraph 6 should be deleted.

7.16.2.4, Paragraph 2, Page 361

At the end of the line, ‘‘takes into account of the additional members’’ should be ‘‘takes into account the values of the additional members’’.

7.16.2.5, Paragraph 2, Page 362

‘‘L’’ should be italicized throughout this paragraph.

7.16.2.6, Paragraph 3, Page 363

At the end of the block of **//** comments, there should be one more:

> **//** *if the offset cannot be determined.*

7.16.3.6, Paragraphs 2–6, Pages 367–369

The typography throughout this section is inconsistent, particularly the use of program font and double-quotes; the use of **==** in paragraph 3 is also unconventional.

The hyphens in the range specifications should be dashes.

7.18.1, Paragraph 2, Page 371

At the end of the description of **wint_t**, there should not be a period before the parenthetical remark, the parenthetical remark should not start with an upper case letter and should not end with a period, and a semicolon should follow the parenthetical remark.

At the end of the description of **wctrans_t**, the comma should be a semicolon.

7.18.1, Footnote 241, Page 371

The reference should be to 7.20.10.

7.18.2.1, Paragraph 3, Page 372

This should be a normal Forward References section — it should not have a paragraph number and the heading should be in bold font.

7.18.2.1.1, Paragraph 2, Page 373

‘‘Th’’ should be ‘‘The’’.

7.18.2.1.3, Page 373

The iswblank function was not approved by the committee and should not have appeared in the draft.

7.18.2.2, Paragraph 1, Page 376

The reference to subclause 4.5.2.1 should be to 7.18.2.1.

7.19.1, Footnote 247, Page 380

The reference should be to 7.20.11.

7.19.1, Paragraph 2, Page 380

The comma at the end of the first line should be a semicolon.

7.19.1, Paragraph 3, Page 380

The comma near the end of the third line should be a semicolon and there should not be a paragraph number.

7.19.1, Paragraph 4, Page 380

The period at the end of the line should be a semicolon followed by ‘‘and’’ and there should not be a paragraph number.

7.19.1, Paragraph 5, Page 380

There should not be a paragraph number and this should read:

```
struct tm
```

and

```
struct tmx
```

which are declared as incomplete structure types, the contents of which are described in subclause 7.16.1.

7.19.1, Paragraph 6, Page 380

The comma at the end of the first line should be a semicolon.

7.19.1, Paragraph 7, Page 380

The comma at the end of the first line should be a semicolon and there should not be a paragraph number.

7.19.1, Paragraph 8, Page 380

The comma near the end of the first line should be a semicolon and should not be in program font, and there should not be a paragraph number.

7.19.1, Paragraph 9, Page 380

There should not be a paragraph number.

7.19.2.1, Paragraph 2, Page 381

The last sentence is duplicated.

7.19.2.1, Paragraph 2, Page 382

In the list item beginning “An optional **hh**...”, the words for **hhn** have been omitted.

7.19.2.1, Paragraph 7, Page 386

In the description of the **c** conversion specifier, near the beginning of the third line, “Otherwise” should be “If an **l** qualifier is present” and should start a new paragraph (cf., the **s** conversion specifier).

7.19.2.1, Footnote 255, Page 387

The reference should be to 7.20.11.

7.19.2.1, Paragraph 14, Page 388

The heading should read "Environmental limits".

7.19.2.1, Paragraph 16, Page 388

This should be a normal Forward References section — it should not have a paragraph number and the heading should be in bold font.

7.19.2.2, Paragraph 11, Pages 391–392

In the descriptions of the **s**, **[**, and **c** conversion specifiers, there should be a paragraph break before the sentence beginning “If no **l** qualifier is present”.

In the description of the **s** conversion specifier, the paragraph beginning “Otherwise” should begin “If an **l** qualifier is present” (cf., the **[** conversion specifier).

In the second line of the description of the **[** format specifier, “f” should be “If”.

7.19.2.2, Footnote 258, Page 392

The reference should be to 7.20.11.

7.19.2.2, Examples, Page 393

There should not be numbered paragraphs in the examples.

7.19.2.2, Paragraph 22, Page 394

This should be a normal Forward References section — it should not have a paragraph number and the heading should be in bold font.

7.19.4.6.2, Paragraph 1, Page 421

The prototype is not parallel to **memcmp**; neither **s1** nor **s2** should be restrict qualified, and **s1** should be a pointer to **const**.

7.19.5 and 7.19.6, Pages 422–423

These should be sub-subclauses of a new subclause titled “Wide-character time conversion functions”.

7.19.7.3.1, Paragraph 3, Page 426

“a value between” should be “or a value between”.

7.19.7.3.1, Paragraph 4, Page 426

This should be a normal Forward References section — it should not have a paragraph number and the heading should be in bold font.

7.20.11, Paragraph 2, Page 433

Should note that other characters may be used in extensions (cf., 7.20.6, paragraph 1, page 432).

Annexes

The annexes and the index are all formatted wider than the main text.

Annex C, Paragraph 2, Page 451

There should also be entries for the sequence points within library functions specified in 7.13.6, 7.19.2, and 7.14.5.

F.1, Paragraph 1, Page 485

The standards' designations should not be in italics, only the titles.

F.1, Footnote 275, Page 488

Both occurrences of "IEEE" should be "ANSI/IEEE".

F.9.3.11, Paragraph 1, Page 501

In the second bullet item, there should be a space between **`)`** and "returns".

G.5, Paragraph 5, Page 514

The reference to subclause 7.9.2 should be to 7.8.2.

Annex H, Pages 522–527

The hyphen in "LIA−1" should be a dash.

H.2.2, Paragraphs 1 and 3, Page 522

The double-quotes should be real quotes.

H.2.4, Paragraph 1, Page 525

The ugly arrows should be replaced by real arrows.

There should not be spaces between the casts and the variable names.

K.2, Page 539, Item 3

The reference should be to 6.5.5.3 (and the item should be moved in the list appropriately).

K.2, Page 544, Item 5

`strfxtime`, **`wcsftime`**, and **`wcsfxtime`** should be listed in addition to **`strftime`**. (Similar changes to page 545 items 1 and 7, and page 547 items 1 and 2; and K.3.12 page 554 item 6.)

K.4, Page 555

In item 8, **`isblank`** (7.3.1.3) should not appear in the list.

In item 18, **`iswblank`** (7.18.2.1.3) should not appear in the list.

Index, Pages 559–570

The index, which was produced manually, is both incomplete and inaccurate. I have since reviewed and corrected it and used that information to continue the process of adding index macros to the text so as to generate the index automatically as part of the formatting process. I have also added the automatic index generation to the formatting process and have produced an enhanced and corrected index to CD1 (N800) which can be used to aid in the review process.

Note that there is still a lot of work to be done on the index; I would appreciate any suggestions for additional entries, additional cross references, missing references, or excessive references.

3. Suggested Editorial Changes

General

References to other standards should not include specific versions except in the References and Bibliography.

3.17, Paragraph 1, Page 5

I think this would read better if ‘‘specification that is’’ was changed to ‘‘specifications that are’’. (Also in other places, particularly annexes F and G.)

5.1.1.2, Paragraph 2, Page 10

This is a peculiar place for a constraint since there is no syntax for it to constrain. It should be moved to where the syntax is, currently 5.2.1 on page 19.

5.2.1 Paragraph 4, Page 19

This is a peculiar place to have syntax, since the syntax isn’t even described until clause 6. I suggest leaving the first sentence here but moving the syntax and all of paragraph 5 (along with the related constraint from 5.1.1.2) into 6.1.2 (Identifiers).

I also suggest changing the description to:

> The universal character name **`\U`***nnnnnnnn* designates the character whose character short identifier (as specified by ISO/IEC 10646-1) is *nnnnnnnn*. Similarly, the universal character name **`\u`***nnnn* designates the character whose character short identifier is 0000*nnnn*.

5.2.4.2.2, Paragraph 7, Page 26

A better wording is:

> The values given in the following list shall be replaced by implementation-defined expressions with values that are greater or equal in magnitude (absolute value) to those shown, with the same sign:

5.2.4.2.2, Paragraph 8, Page 27

A better wording is:

> The values given in the following list shall be replaced by implementation-defined expressions with values that are greater than or equal to those shown:

5.2.4.2.2, Paragraph 9, Page 28

A better wording is:

> The values given in the following list shall be replaced by implementation-defined expressions with [positive?] values that are less than or equal to those shown:

Note that ‘‘positive’’ is implied by the float-point model, but it might be worth repeating.

6.1.1, Paragraph 2, Page 33

When discussing **`imaginary`**, I think it would be clearer to say:

> ... the token **`imaginary`** is reserved in translation units where the header **`<complex.h>`** is included and defines the macro **`_Imaginary_I;`**

The current wording implies that the program needs to define **`_Imaginary_I`** rather than the header.

6.1.3.1, Paragraph 1, Page 47

The definition of hexadecimal digit really belongs in 6.1.3.2 (Integer constants) since there is text there explaining it. It would perhaps be best to switch these two sections.

6.1.3.4, Paragraph 5, Page 50

The lists of types for constants would be much easier to read as a table.

6.3.5, Paragraph 7, Page 84

This paragraph should be merged with paragraph 3.

6.3.6, Paragraph 10, Page 86

This paragraph should be merged with paragraph 4.

6.4, Footnote 84, Page 98

Since the footnote only talks about integer constant expressions, it would be better to have the reference on that term in paragraph 6 rather than where it currently is.

6.5.2.2, Paragraph 5, Page 108

It would be better to say:

> The type is incomplete until after the **}** that terminates the list.

since that's what we say for structure and union specifiers (cf., 6.5.2.1, paragraph 6, page 104).

6.8.8, Paragraph 1, Page 170

It might be clearer to specify that **`__LINE__`** is the line number *within the current source file* and that **`__FILE__`** is the name of the *current* source file.

Clause 7, Pages 174–433

Many of the subclauses (and sub-subclauses) should be rearranged to put them into alphabetical order. I suggest moving the headers that are currently under 7.1 out to the top level (even though that means splitting up **<float.h>** and **<limits.h>**, neither of which say much); alternatively, **<iso646.h>** (and, perhaps, **<stdbool.h>**) would need to be moved into 7.1. I also suggest alphabetizing the individual functions within each category.

7.7, Pages 218–248

A number of subclauses in this section write out equations using words such as ‘‘x raised to the power y’’. I suggest using equations in all cases.

7.7.6.1, Paragraphs 2–3, Page 229

Since we now have additional exponential functions, this should be reworded parallel to 7.7.6.7 (The exp2 function) to make the base explicit.

7.7.6.4, Paragraphs 2–3, Page 230

Since we now have additional log functions, this should be reworded parallel to 7.7.6.10 (The log2 function) to make the base explicit. The note that the base-*e* logarithm is the natural logarithm should be kept as a parenthetical note.

7.7.6.5, Paragraphs 2–3, Page 230

This subclause uses ‘‘base-ten’’, other subclauses use ‘‘base-2’’; we should be consistent. I suggest using numbers everywhere. I also suggest a parenthetical note that the base-10 logarithm is the common logarithm since we note that the base-*e* logarithm is the natural logarithm.

7.13.3, Paragraph 7, Page 279

Since this paragraph is talking about the standard streams, it might be better to move it to subclause 7.13.2 (Streams).

7.15.1, Page 345

There doesn't seem to be any good reason for this being a subclause; I suggest eliminating the heading.

7.16.1, Page 357

There doesn't seem to be any good reason for this being a subclause; I suggest eliminating the heading.

7.16.3.6, Paragraph 3, Page 368

I suggest rewording the second sentence as:

> In this system, weeks begin on a Monday and week 1 of the year is the week that includes January 4th, which is also the week that includes the first Thursday of the year.

7.16.3.6, Paragraph 5, Pages 369

I think it would be clearer if paragraph 5 were merged into the end of paragraph 2.

7.18.1, Page 371

There doesn't seem to be any good reason for this being a subclause; I suggest eliminating the heading.

7.18.2.1, Paragraph 2, Page 372

In the second line, it would be better to not mention the exact number of functions (it's easy to overlook if the list ever changes). Similarly for 7.18.2.2.1, paragraph 3, page 376; 7.18.2.2.2, paragraph 3, page 377; 7.18.3.2.1, paragraph 3, page 379; and 7.18.3.2.2, paragraph 3, page 379.

7.19.1, Page 380

There doesn't seem to be any good reason for this being a subclause; I suggest eliminating the heading.

K.2, Page 537, Item 3

Similarly, converting between a pointer to an object or incomplete type and a pointer to a function causes undefined behavior. Subclause 6.2 is also relevant.

K.2, Page 537, Item 12

It should be clarified that this only applies to relational comparisons, not equality comparisons.

K.2, Page 538, Item 14

I think what this is supposed to say is "An attempt is made to access an object through both a restrict-qualified pointer and another pointer not based on it."

4. Suggested Possibly-substantive Changes

6.1.3.1, Paragraph 1, Page 47

In the first two productions for hexadecimal floating constant, the binary exponent part should be optional (parallel to decimal floating constant).

5. Open Issues

General

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 many typesetting issues still to be addressed. For example, there are many equations typeset as code and vice versa, particularly in annexes F and G. Some of these issues primarily affect the typeset version and some primarily affect the text version.

Still need final words for a real boolean type (N738 and part of N743 were adopted in principle, but no final words were ever produced).

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.

Clause 4, Paragraph 2, Page 7

Should **`<stdbool.h>`** be added to the list of required headers for freestanding implementations?

6.1, Paragraph 3, Page 31

Are all the italicized terms really appropriate?

6.1.1, Paragraph 2, Page 33

It has been suggested that a freestanding implementation need not support complex types since **`complex`** is not a keyword unless **`<complex.h>`** has been included and a freestanding implementation need not provide **`<complex.h>`**. However, the actual conformance specification in clause 4 only exempts freestanding implementations from features specified in the library clause and making **`complex`** a keyword is specified here in the language clause. We need to decide what we really want to require and make the wording clear one way or the other.

6.1.3.1, Paragraph 1, Page 47

The **`0x`** and **`0X`** prefixes are used in multiple places, it might simplify the grammar to factor them out into their own nonterminal.

Clause 7

There are no synopses for the float and long double versions of the math routines and there are a number of synopses that basically duplicate or refer to other synopses. We should consider allowing multiple synopses for a family of functions with a single description like the Unix man pages have traditionally done.

7.4

Much of **`<inttypes.h>`** is appropriate for freestanding implementations, but not all of it. My suggestion is to move the conversion functions to **`<stdlib.h>`** and **`<wchar.h>`** with the other conversion functions, move the **`printf`**/**`scanf`** macros to **`<stdio.h>`** (yes, I know that's adding a bunch of names to a very popular header, but I don't like any of the alternatives I've heard so far any better), and move the rest to **`<stdint.h>`** and require freestanding implementations to support it. (Changing the name allows implementations that currently support **`<inttypes.h>`** to continue to provide it with the existing semantics.)

7.7.11.2, Paragraph 2, Page 242

It is not clear what the last line (“a call to the **nan** function is unspecified”) is supposed to mean; is it just the returned value that’s unspecified or is the behavior completely unspecified? Can such an implementation not provide the function at all? Can calling the function crash the program?

7.11.2.1, Paragraph 3, Page 268

It is not clear when the **raise** function is permitted to fail; is it only for an invalid signal number or are there other conditions?

7.13.5.2, Paragraph 4, Page 283

It is not clear what happens if **stream** is a null pointer and a write error occurs on one of the streams being flushed.

7.13.5.3, Page 283

It has been pointed out that a call like:

```
fopen("bar", "r")
```

results in undefined behavior since the string literals might not be distinct and thus do not satisfy the requirements for restrict-qualified pointers. There doesn’t seem to be much benefit in making the **filename** and **mode** parameters restrict-qualified, so I suggest removing the qualification. (Also **freopen**.)

Alternatively (or additionally), we may want to try to revise the formal definition of **restrict** to allow aliasing under limited circumstances (such as read-only usage), but I’m afraid that’s starting down the slippery slope toward **noalias**.

7.14.3.4, Paragraph 3, Page 334

The comparison described in the last sentence results in undefined behavior if the object has moved.

7.3.1 and 7.18.2.1, Pages 185–189 and 372–377

Many of these functions provide very little guidance as to their intended semantics, particularly with respect to the additional locale-specific characters they accept. It seems like we should say a bit more than we currently do.

Annex F

There are no entries for **ilogb**, **scalbln**, **llrint**, **llround**, or **nextafterx**. We should say something about them, just so it doesn’t look like an oversight.

Annex I

There is a new version of ISO TR 10176; the table needs to be updated (and I suggest removing the leading dashes). The new specification includes extended digits and special characters; while it seems reasonable to allow these in identifiers, it seems like a good idea to disallow an extended digit as the initial character (that would allow an implementation to extend the syntax for numeric constants to include the extended digit characters). I suggest changing the last sentence of 6.1.2, paragraph 2, page 34 to:

> The first character must be a nondigit character; it shall not be a universal character name designating a digit.

(also in K.2).

K.2, Page 535, Item 4

The first character of an identifier can’t be a digit, the syntax doesn’t allow it.

K.2, Page 540, Item 10

This isn't undefined behavior. I think the situation that this is supposed to be describing is the one in DR #017, Question 23: When a macro expansion ends with the name of a function-like macro and the first token of the remaining source-file tokens is a `(`, and that macro expansion ends with the name of the first macro and the first token of the remaining source-file tokens is again a `(`, it is not clear whether that is a nested replacement or not and thus it might be replaced and it might not. It is undefined behavior to depend on one or the other.

I have no idea how to express this succinctly.

-- Larry Jones