Editor's Report

October 4, 2010

1. Status

N1516 is a revised draft incorporating the changes from the editorial review meetings. It contains diff marks for all of the changes from N1494 except for minor formatting changes and subclause number changes.

2. Notable editorial changes from N1494

1. Finally populated the list of major changes in the Foreword.
2. Replaced ''pointed-to type'' with ''referenced type'', a defined term from 6.2.5p20 (last item). Also replaced ''pointed-to'' with ''referenced'' in a couple of related contexts (e.g., ''pointed-to object'') for symmetry.
3. Clarified NOTE 2 in the definition of memory location (3.14) to take atomic bit-fields into account.
4. Clarified footnote 3 in the definition of a strictly conforming program (4) to note that the use of conditional features must be protected by an appropriate preprocessing directive, which might not be **`#ifdef`**.
5. Clarified the description of signal handling in 5.1.2.3p5 to take atomic objects into account.
6. Clarified 6.2.4p2 to include a pointer just past the end of an object becomming indeterminate when the object's lifetime ends.
7. Clarified the description of alignment in 6.2.8 to only apply to complete types.
8. Clarified the description of integer promotions in 6.3.1.1p2 to not apply to **`int`** or **`unsigned int`**.
9. Reformatted the constraint in 6.4.4.4p9 to be easier to read.
10. Added introductory text to the generic selection example in 6.5.1.1p5
11. Moved **`_Atomic()`** from Declarations (6.5) to Type specifiers (6.7.2), where it belongs.
12. Revised the definition of contracted expressions in 6.5p8 to avoid confusion with operations on atomic objects.
13. Added a footnote in 6.5.16 to clarify the behavior of assigning to a volatile variable.
14. Added a constraint in 6.7.2p3 to forbid using **`_Atomic()`** when atomic types are not supported.
15. Clarified the definition of full expression in 6.8p4 (and Annex C) to exclude initializers that are part of compound literals.
16. Split 6.10.8 into separate subclauses for macros that are always defined, macros that specify something about the environment, and macros that specify which conditional features are supported.
17. Examined the library synopses and removed extraneous headers (only **`btowc`** and **`wctob`** were affected).
18. Corrected the introductory text in a number of the library subclauses to avoid conflating definition (of macros) and declaration (of types and functions).
19. Corrected the note about the **`CMPLX`** macros in 7.3.9.3p5 to cast the results to the correct types.

20. Clarified the description of the **frexp** functions in 7.12.6.4 to note that the result is unspecified if the exponent is too big to fit in an **int**.
21. Clarified the description of the non-**_explicit** functions in 7.17.1p5.
22. Clarified the description of overflow behavior for **strtod** and friends in 7.22.1.3p10 and 7.28.4.1.1p10 to take rounding into account.
23. Added **#include <threads.h>** line to the synopses of all of the threads functions in 7.25.
24. Clarified the specification of allowable mutex types for **mtx_timedlock** and **mtx_trylock** in 7.25.4.4p2 and 7.25.4.5p2 to avoid confusing use of the word ''type''.
25. Updated Annex B to include new items that were overlooked as well as all of the bounds-checking stuff from Annex K.
26. Updated the introductions in Annex F and G to note that they specify conditional features.
27. Clarified the description of floating to integer conversions in F.4p1 to exclude out-of-range integral values from the ''no floating-point exceptions'' provision.
28. Corrected the description of relational operator optimization in F.9.3 to refer to expressions rather than statements.
29. Corrected the description of comparison macros in F.10.11 to refer to relational operators rather than inequality operators.
30. Corrected footnote 362 in K.3.1.1.3 to indicate that the **__STDC_WANT_LIB_EXT1__** macro must be defined as **0** to avoid the additional names, not just left undefined.
31. Added TR 19769 (new character data types) to the Bibliography.

3. Previously Applied Content

1. N1252 *A finer-grained specification of sequencing*
2. N1282 *Clarification of Expressions*
3. N1285 *Extending the lifetime of temporary objects* (factored approach)
4. N1300 *Draft minutes for April 2008* 9.19 *N1271*
5. N1310 *Requiring* **signed char** *to have no padding bits*
6. N1311 *Initializing static or external variables*
7. N1316 *Conversion between pointers and floating types*
8. N1319 *Adding* **EPOLE** *to math library functions* (modulo change in minutes N1346)
9. N1320 *Integrating C89 Defect Report 25 into C1x* (modulo N1346)
10. N1321 *Split* **FLT_EVAL_METHOD** *into operations and constants* (modulo N1346) [subsequently removed by N1361]
11. N1326 *Adding TR 19769 to the C Standard Library*
12. N1327 *Abandoning a Process (adding* **quick_exit** *and* **at_quick_exit***)* (modulo N1346)
13. N1330 *Static Assertions* (modulo N1346)
14. N1338 *More Thoughts on Implementing* **errno** *as a Macro*
15. N1346 *Draft Minutes for September 2008* (3.2 Report of the Project Editor) [Update Annex C (Sequence Points) to match revised text]
16. N1349 *Parallel memory sequencing model proposal*
17. N1350 *Analyzability* (#1, #4 - conditionally normative)
18. N1353 **FLT_EVAL_METHOD** *issues* (first change only)
19. N1356 **_Bool** *bit-fields*
20. N1357 **tgamma** *range error* (first way)
21. N1358 *Extensions to the C1X Library* (#1, #2, #3 along the lines)
22. N1359 *Technical corrigendum for C1X*
23. N1360 *Benign typedef redefinition*
24. N1361 **FLT_EVAL_METHOD** *and constants* (back out N1321)

25. N1363 *Editor's Report* (move unicode feature test macros, make **`wchar_t`** encoding imp-def if not 10646, and remove **`quick_exit`** footnote)
26. N1364 *Thread-local storage* (plus make auto access imp-def just like thread local)
27. N1365 *Constant expressions* (second bullet except ''are covered'' rather than ''are not covered'')
28. N1367 *Contractions and expression evaluation methods*
29. N1371 *Thread Unsafe Standard Functions* (except for 21.5)
30. N1372 *Threads for the C Standard Library*
31. N1373 *Wording improvements for* **`mblen`**, **`mbtowc`**, *and* **`c16rtomb`**
32. N1376 *Adding pole error to math library functions*
33. N1377 *xxx*`_DECIMAL_DIG` macros for **`<float.h>`**
34. N1380 *Minutes for Markham, March/April 2009* (4.29 WG14 Mail 11572)
35. N1381 **`memset_s()`** *to clear memory, without fear of removal*
36. N1382 **`FLT_EVAL_METHOD`** *and return*
37. N1383 *LIA annex correction*
38. N1384 *xxx*`_TRUE_MIN` *macros for* **`<float.h>`**
39. N1387 *Requested clarifications for thread-local storage* (initialization only)
40. N1391 *Floating-point to* **`int`**/**`_Bool`** *conversions*
41. N1394 *Analyzability* (along the lines)
42. N1396 *Wide function return values* (alternate proposal)
43. N1397 *Adding Alignment Support to C* (use **`_Align`** keyword rather than **`[[ ]]`** syntax)
44. N1398 *Treatment of math error conditions*
45. N1400 *Headers not idempotent*
46. N1406 *Anonymous Member-Structures and -Unions* (modulo ''name lookup'')
47. N1420 *On The Removal of* **`gets()`**
48. N1371 *Thread Unsafe Standard Functions* (21.5 changes).
49. N1439 *Completeness of types*
50. N1441 *Generic macro facility*
51. N1444 *Dependency Ordering for C Memory Model*
52. N1447 *C and C++ Alignment Compatibility*
53. N1459 *Comparison Macros* (option 2)
54. N1460 *Subsetting the Standard*
55. N1462 **`errno`** *and threads*
56. N1464 *Creation of complex value* (move static init to Recommended Practice and make "as if" a Note)
57. N1468 *Assumed types in F.9.2* (along the lines - change font)
58. N1472 *Comparison Macro Argument Conversion* (option 3)
59. N1478 *Supporting the 'noreturn' property in C1x*
60. N1480 *Updates to C++ Memory Model Based on Formalization*
61. N1482 *Explicit Initializers for Atomics* (use data race words from N.1 for N.2, remove ''where it applies'' from N.1)
62. N1485 *Atomic proposal* (minus ternary op)
63. N1486 **`fabs`** (F.3 change only)
64. N1487 *Comparison Macro Argument Conversion*
65. N1488 *UTF-8 string literals*
66. *E-mail 12207* (fix to 6.7p2 for static_assert declarations)
67. *dr269fix*

4. Non-editorial Issues

1. The type **`char`** is not included in the standard integer types in 6.2.5, although both **`signed char`** and **`unsigned char`** are (p7). It is included in integer types (p17), but it seems like it should be a standard integer type.

2. Footnote 57 in 6.2.8p3 notes that every over-aligned type is, or contains, a structure or union type with a member that has an extended alignment. It was not clear to the editorial review committee where, if anywhere, this is stated normatively.

3. The prohibition in 6.5.2.3p5 against accessing members of an _Atomic-qualified structure or union would seem to make them rather difficult to use. Should this say that undefined behavior *may* occur rather than that it *does* occur?

4. The ''editorial'' change made to 6.7.6.2p2 in response to DR 320 and incorporated as part of TC3 had an unintended normative effect that needs to be reversed.

   The original words were:

   > Only an ordinary identifier (as defined in 6.2.3) with both block scope or function prototype scope and no linkage shall have a variably modified type.

   This clearly restricts the kinds of things that can be declared with a variably modified type. In particular, it forbids declaring structure and union members as is noted in footnote 121 in 6.7.2.1. The revised wording from DR 320 no longer contains this restriction:

   > An ordinary identifier (as defined in 6.2.3) that has a variably modified type shall have either block scope and no linkage or function prototype scope.

   I propose the following rewording to restore the restriction:

   > If an identifier is declared as having a variably modified type, it shall be an ordinary identifier (as defined in 6.2.3) and have both block scope or function prototype scope and no linkage.

5. The description of initialization in 6.7.9 was not updated to take the new sequencing model into account. I propose updating 6.7.9p23 to:

   > The evaluations of the initialization list expressions are indeterminately sequenced with respect to one another and thus the order in which any side effects occur is unspecified.

   Does this need to be tweaked for compound literals?

6. The description of signal handling in 7.14.1.1p5 should be harmonized with the description in 5.1.2.3p5.

   The description of a signal handler in p2 should probably note that functions called indirectly via standard library functions like **`abort`** (when **`SIGABRT`** is being caught) are also considered to be part of the handler.

7. ISO/IEC TR 10176 *Information technology — Guidelines for the preparation of programming language standards* (the basis for Annex D) has been updated (twice!) since the 1998 edition that we reference. We should probably update our reference and Annex D to the current edition.

8. ISO 4217, *Codes for the representation of currencies and funds*, ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*, ISO/IEC 9945, *Information technology — Portable Operating System Interface (POSIX)*, and ISO/IEC 10646, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)* have also been updated. We should investigate referencing the newer editions.

9. The index probably needs work, particularly for the newly added material.

--

Larry Jones