WG 14 Standing Document 3

Partial list of proposals that did not fit into the former DR process for C11

Doc No:	WG 14 N 2556
Date:	2020-08-02
Project:	JTC 1.22.20
Reply To:	David Keaton Convener, WG 14 Keaton Consulting 848 N. Rainbow Blvd. #4499 Las Vegas, NV 89107
E-mail:	dmk@dmk.com
Phone:	+1 702 518 8192
Fax:	+1 702 852 0248

This document contains a partial list of proposals that could not be dealt with by a former C11 Defect Report. Most originated as potential Defect Reports but could not be handled via that process. Each should be reviewed when the time comes to start the process to republish the C Standard.

Except where noted, this document is not intended to indicate the direction WG 14 will take for the development of the C Standard. This is just a list of proposals that were not addressed by the WG 14 Defect Report process. It is also not meant to be a complete list of items to be considered for the next revision of the C standard.

1. N 1730, DR 440

   To do as suggested in this defect report, distinguish whether the **`float`**, **`double`**, and **`long double`** types are IEC compliant or not, requires the addition of new feature test macros. This is a new feature, and is not allowed by the mechanism of the defect report process.

2. N 1817, DR 463

   This is a request to align C with C++14, C++14 has defined the shifting of a bit into the sign bit, again this is a feature that can not be addressed by the defect report process. The following change was proposed in the defect report.

   6.5.7p4 should be modified to read:

   > The result of `E1 << E2` is `E1` left-shifted `E2` bit positions; vacated bits are filled with zeros. If `E1` has an unsigned type, the value of the result is $E1 \times 2^{E2}$, reduced modulo one more than the maximum value representable in the result type. If `E1` has a signed type and nonnegative value, and $E1 \times 2^{E2}$ is representable in the corresponding unsigned type of the result type, then that value, converted to the result type, is the resulting value; otherwise, the behavior is undefined.

3. N 1793, N 1818, DR 451

   The issue about indeterminate values (*wobbly bits*) will possibly need to be clarified. *Wobbly bits* were not considered during the development of C11.

4. N 1812, N 1874, DR 461

   The portion of this defect report that requests allowing `const` qualified objects in signal handlers is considered a new feature. The implications of the following suggested change from the defect report should be studied carefully before adopting any change.

   In section 7.14.1.1, modify the first sentence of paragraph 5 as indicated below:

   > If the signal occurs other than as the result of calling the `abort` or `raise` function, the behavior is undefined if the signal handler ~~refers to~~accesses any non-`const` object with static or thread storage duration, or any non-`const` object with automatic storage duration whose lifetime started before the signal handler has been entered, that is not a lock-free atomic object other than by...

   In addition, make the corresponding change to section **J.2 Undefined behavior**.

5. N 1736, DR 446

   This defect report requests a local change to a globally-used term. A comprehensive review of all uses would be required, which is outside the scope of this defect report.

6. N 1865, DR 466

   This defect report requests a harmonization with C++, which is often grounds for making a change. However, this particular change would invalidate existing code and should not be done using the defect report process. The following change was proposed in the defect report.

   Align the C rules with those of C++ by adding a new paragraph to section **6.2.1 Scopes of identifiers** as follows.

   > Names declared in *clause-1* of the `for` statement are local to the `for` statement and shall not be redeclared in a subsequent condition of that statement nor in the outermost block of the controlled statement.

7. N 1899

   This proposal for integer width macros should be considered for adoption with the caveat that the widths suggested for 7.20.2.1 should be equal to N, not <= N.

8. N 1910

   This proposal requests a new feature, `_Alignof` on incomplete array types, for harmonization with C++.

9. N 1911

   The committee decided that the preprocessor behavior addressed in this document should be considered for potential unspecified behavior, rather than undefined behavior as suggested in the document.

10. N 1923

    This is a proposal for a new feature, allowing qualified doubly-dimensioned array parameters to be compatible with unqualified arguments.

11. N 1870, DR 467

    New feature request: consider adding to the standard the macros `FLT_TRUE_MAX`, `DBL_TRUE_MAX`, and `LDBL_TRUE_MAX` from this DR.

12. N 1962, N 1969, N 2225

    TR 24731-1, which eventually became Annex K, was written before threads existed in the C language. Then Annex K and threads were separately added to the C standard. Their effects on each other were overlooked, leading to runtime constraint handlers not integrating well with threads. This urgently needs to be addressed, and requires changes beyond what a technical corrigendum can do.

13. DR 482

    Feature removal request: consider removing from the standard the ability for macro invocations to span an include file boundary.

    In Ithaca (minutes N 2509) the committee reviewed this topic in the form of a proposal in N 2324, and decided not to make this change.

14. N 2008

    New feature request: consider allowing the programmer to specify the type used to represent an enum.

15. N 2034

    New feature request: consider adding a mechanism to prevent trailing commas in the expansion of variadic macros.
    Track what C++ does with this issue (WG 21 P0306R0).

16. N 2017

    New feature request: consider adding parallelism TS 21938-1 into the C standard.

17. N 2043

    New feature request: consider changing the definition of out-of-bounds store in Annex L.

18. N 2269, N 2265, N 2266, N 2267, N 2268

    New feature request: consider adding attributes to C.

19. N 2074

    New feature request: consider adding a constraint to make `[static n]` declarations of the same array parameter match.

20. N 2078, N 2079, N 2095

    Add floating-point TS 18661 parts 1 and 2 into the C standard. The committee has decided to pursue this for C2x.

21. N 2083

New feature request: consider allowing a structure with a flexible array member to be nested inside another structure.

22. N 2089, N 2090, N 2091

    Technical change request: consider changing the specification of unspecified values, pointer provenance, and trap representations.

23. N 2098

    Technical change request: consider changing the specification of the number of fractional digits in `printf %a` output. The committee prefers an option that is similar to #3 in the paper, or that goes beyond the first two options.

24. N 2101

    New feature request: consider adding the C++ `__has_include` feature to C.

25. N 2117, N 2118, N 2119, N 2120, N 2121, N 2122, N 2124

    Consider adding the following floating-point items to the C standard. Committee deliberations revealed a medium level of committment to pursue these (more committed than most items, though less committed than TS 18661 parts 1 and 2, which are already decided).
    - TS 18661-3 - entire part
    - TS 18661-4 - mathematical functions
    - TS 18661-4 - reduction functions
    - TS 18661-5 - evaluation format pragmas
    - TS 18661-5 - optimization control pragmas
    - TS 18661-5 - reproducible results
    - TS 18661-5 - rounding direction macro

26. N 2123

    Consider adding something along the lines of *TS 18661-5 - alternate exception handling pragma* to the C standard. The committee may be interested in the general idea but is not sure of the specifics.

27. Additional C++ harmonization

    Consider whether the following C++ changes should be made in C. Note that these items need proposals submitted as WG 14 papers to progress further.
    - Drop trigraphs (C++17)
    - Add u8'x' literals (C++17)
    - Expand `static_assert` (details needed)

28. Future Directions for C2x

    Consider removing features. Note that this item needs proposals submitted as WG 14 papers to progress further.

29. N 2186

    Consider an alternate approach to defining evaluation formats.

30. N 2197

Consider making changes to harmonize `static_assert` with C++.