ISO/IEC JTC 1/SC 22/WG14 **N2435**

September 25, 2019 **v 3**

ISO/IEC 9899 editor report September 2019

Larry Jones and Jens Gustedt
Siemens, USA, and INRIA and ICube, Université de Strasbourg, France

The September 2019 working draft is now available. It contains most of the changes to ISO 9899 that have been voted in the April 2019 meeting of WG14 in London:

> **N2433** – ISO/IEC 9899 working draft September 2019

The following document contains change marks to C17:

> **N2434** – ISO/IEC 9899 working draft September 2019, diffmarks

These diffmarks are not perfect (and cannot be) as they also show differences that are only markup, and as some changes in code snippets (examples, synopsis) are difficult to capture as textual differences. Please verify in the full document before reporting any oddities. Beware that, although these documents only present editorial changes or changes that have been voted into C2x, subsequent changes to the same parts of the document may still occur in later stages of the process. In particular, the naming of the new floating point functions should not yet be considered stable.

1. PROGRESS ON VOTED AND PROPOSED CHANGES

1.1. Integrated changes

The following resolutions to clarification requests (DR) and new features (N-documents) have been integrated in these documents:

DR 476. volatile semantics for lvalues
DR 488. `c16rtomb()` on wide characters encoded as multiple `char16_t`
DR 494. Part 1: Alignment specifier expression evaluation
DR 496. `offsetof` and subobjects (with editorial modification)
DR 497. ”white-space character” defined in two places
DR 499. Anonymous structure in union behavior
DR 500. Ambiguous specification for `FLT_EVAL_METHOD`
DR 501. make `DECIMAL_DIG` obsolescent
FP DR 20. changes for obsolescing `DECIMAL_DIG`
FP DR 21. `printf` of one-digit character string
FP DR 23. `llquantexp` invalid case
FP DR 24. `remainder` NaN case
FP DR 25. `totalorder` parameters
N2124 and N2319. rounding direction macro `FE_TONEARESTFROMZERO`
N2186. Alternative to N2166
N2212. type generic `cbrt` (with editorial changes)
N2260. Clarifying the `restrict` Keyword v2
N2265. Harmonizing `static_assert` with C++
N2267. `nodiscard` attribute
N2270. `maybe_unused` attribute
N2271. CR for `pow` divide-by-zero case
N2293. Alignment requirements for memory management functions
N2314. TS 18661-1 plus CR/DRs for C2X
N2322. preprocessor line numbers unspecified
N2325. `DBL_NORM_MAX` etc

N2326. floating-point zero and other normalization
N2334. **deprecated** attribute
N2335. attributes
N2337. **strftime**, with 'b' and 'B' swapped
N2338. error indicator for encoding errors in **fgetwc**
N2341. TS 18661-2 plus CR/DRs for C2X
N2345. editors, resolve ambiguity of a semicolon
N2349. the **memccpy** function
N2350. defining new types in **offsetof**
N2353. the **strdup** and **strndup** functions
N2356. update for payload functions
N2358. no internal state for **mblen**
N2359. part 2 (remove WANT macros from numbered clauses) and part 3 (version macros for changed library clauses)
N2401. TS 18661-4a for C2X

The integration of *TS 18661-1* and *TS 18661-2* (and even more *TS 18661-2*) has serious issues concerning the invention of about 500 (1700) new identifiers, many of which had not previously been reserved. Thus they intrude the user name space and as a whole may have a non-negligible risk of name collision. This risk has not yet been evaluated. Several ideas have been discussed to resolve these issue, but none has yet resulted in a proposal that would find consensus. See papers **N2409** and **N2426** for discussions of these issues.

1.2. Voted but still missing changes

Because of lack of work force the following voted changes have not yet been integrated into the documents:

1.2.1. Spring 2019 meeting

Direct changes to ISO 9899:

N2342. TS 18661-3 as annex (see also **N2426**)
N2112. change bullet points in annex J to referable labels or numbers (see also **N2427**)

The following resolutions to CR/DR for the floating-point specifications TS18661-x have been adopted. Since these specifications are now integrated in large parts into C2x, this also induces changes, there.

FP DR 13. Type-generic macros for functions that round result to narrower type
FP DR 26. **rootn** case differs from IEEE 754

1.3. Held back

The following changes have been voted into C2x but held back for different reasons;

— there is a 6 month period for possible comments or objections, or
— because a presented paper that has to be wrapped up to represent the committee consensus.

We expect to integrate them after the next meeting:

N2328. Introduce the term storage instance, see N2388 for an updated version.
N2329. part one, non-normative changes to atomics, see N2389 for an updated version.
N2330. Consensus has been reached to restrict sign representations to two's complement, with a minimum value that has the form of -2^{N-1}, and such that the bounds in

`<limits.h>` and similar can be expressed as math formulas only using the width N. See N2412 for an updated version.

N2368. Consensus has be reached to use the methods described in this paper to integrate a set of lowercase keywords that had been provided by special headers as macros more closely into C. The keywords that have been approved so far are **`alignas`**, **`alignof`**, `bool`, **`false`**, **`static_assert`**, **`thread_local`**, and **`true`**.
See N2392 for an updated version.

2. EDITORIAL IMPROVEMENTS

The document has undergone some editorial changes, namely:

— Improvement of the "diffmark" procedure to produce the document with diffmarks. In particular the page layout should be stabilized.
— Improvements for the categorization of identifiers that occur both as functions and as macros.
— Correction of some LaTeX coding. In particular the identification of terminal punctuators in the syntax and the miss-interpretation of `--` (decrement operator) as – (long hyphen).

Other editorial changes are a bit more involved.

2.1. Library function synopsis

A lot of function synopsis in the library clause have been reworked editorially. Annex B is now mostly produced automatically, such that we may avoid inconsistencies in the future. The dependency of the availability of certain interfaces is systematically classified by means of feature test macros.

2.2. Lists of reserved identifiers

The discussion of the integration of *TS 18661-1* has brought to light that there might be large difference in perception among the committee and the public about the impact of changes to the document in terms of reserved identifiers. This impact may be important, in particular if short abbreviated identifiers or common English words are added.
This has lead us to the addition of a new non-normative clause J.6 to Annex J (Portability) that categorizes identifiers used by the document. It consists of a list of 38 regular expressions that systematically reserve 629 identifiers. Not matched by these regular expressions are 1188 identifiers. The lists and corresponding counts are generated automatically such that their maintenance should not add work for future editors.

2.3. Member declarations

The discussion about possible future integration of the *attribute* construct with its author Aaron Ballmann have brought to light that there has been a set of misnomers of syntax terms that were particularly annoying, namely four terms that describe different syntax levels for **`struct`** and **`union`** members. As historical artifacts these bare all the non-word *struct* in their names, leading to the confusion that they might actually talk about **`struct`** themselves, and that they would not apply to **`union`**.
Thus, we proceeded to a systematic text replacement of the syntax terms "*struct declaration*", "*struct declaration list*" "*struct declarator*", and "*struct declarator list*" to the more appropriate "*member declaration*", "*member declaration list*", "*member declarator*" and "*member declarator list*", respectively.

2.4. `cbrt` example in N2212

N2212 and FP DR 16 were still not completely correct in the way they expressed the dealing with the rounding modes. Namely, the identifiers inside the **`_Generic`** are evaluated much

later than preprocessing, so refering to macros that would magically implement the rounding properties correctly made no sense.
For the final patch:

— We replace references to macro properties by function calls with the required rounding properties
— We added crossreferences to and from the example in the clause for **`_Generic`**

3. SUBMISSION OF CHANGE SETS

3.1. Contents of the changes

Please, provide titles for your documents that clearly indicate the subject of the change, and *not* something that refers to our process of changes. “New version of NXYZ0” is *not* a good title, “A second attempt to specify the toto feature” is much better and comprehensive for everybody.
Please, try to adapt your language to the normative context where it is applied. Don’t use unspecified terms but use the terms as introduced by the document, even if it sounds repetitive.
Clearly distinguish normative and non-normative parts. Non-normative text belongs in footnotes, “Notes” and “Examples”.
Usage of the verb forms “must”, “shall” and “may” is quite regulated in ISO text and should be applied with care. The preferred form to express a normative requirement is “shall”. If found inside a “Constraint” section it is a normative constraint and requires a diagnostic, if applied in other places it makes a non conforming program or execution undefined.
Avoid to add requirements for which the violation “only” results in undefined behavior. In particular, any erroneous construct that can always be detected at translation time, should be a constraint violation. Undefined behavior (implicit or explicit) should only be introduced to fulfill one of the following:

— A violation (runtime or translation) is not detectable in all cases, because the sought property is not constructive or has a high verification complexity.
— Implementations may use their own definitions to extend the C standard.

Remember that “undefined behavior” is not a thing but the absence of a thing, so there is no “undefined behavior” but only behavior that is undefined.

3.2. Submission format for changes

Authors of papers that are supposed to change the document and that have access to the source repository (all members of WG14 should), should apply their changes there and submit patches to the editors. Such a procedure ensures a seamless integration into the text and often reveals dependencies to other parts of the document that need to be explored. Such patches need not to be perfect, usually simple text level integration is a good start for the editors. The LaTeX sources can be seen as pure text for which it should be easy to apply modifications for any person with programming skills. Please contact the editors if you have any doubt or question about such a procedure.
If for some reason that procedure does not seem feasible for you, please submit your changes with enough care, such that the editors may easily track modifications down to the word level: it is very tedious and error prone to scan entire paragraphs for single words that might have changed.

— Use a color scheme to mark deleted words and a different one for added words.
— Have your change sets left adjusted, ideally in a mono-space font, such that it is easy to spot changes.
— Mark all special words (identifiers, keywords) with some easily identifiable markup

- Footnotes that change, are deleted or added, should immediately follow the paragraph to which they apply. The spot of insertion should be clearly marked.
- If possible, give indications where your text introduces new definitions of terms, and when the renewed usage of a term should be listed in the index.
- If you are changing the document structure, by deleting or inserting new elements (such as paragraphs, sections, clauses) invent a numbering scheme that helps to uniquely identify old and new elements (e.g p8a for a paragraph inserted after old paragraph 8).