

ISO/IEC 9899 editor report March 2019

Larry Jones, Jens Gustedt

▶ **To cite this version:**

Larry Jones, Jens Gustedt. ISO/IEC 9899 editor report March 2019. [Technical Report] N2348, ISO JCT1/SC22/WG14. 2019. hal-02089676

HAL Id: hal-02089676
https://inria.hal.science/hal-02089676

Submitted on 4 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ISO/IEC JTC 1/SC 22/WG14 **N 2348**

March 18, 2019 **v 1**

ISO/IEC 9899 editor report March 2019

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

The working draft for the April 2019 meeting of WG14 in London is available:

N2346 – ISO/IEC 9899 working draft March 2019

The following document contains change marks to C17:

N2347 – ISO/IEC 9899 working draft March 2019, diffmarks

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.

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 497. "white-space character" defined in two places
DR 499. Anonymous structure in union behavior
DR 500. Ambiguous specification for `FLT_EVAL_METHOD`
N2186. Alternative to N2166
N2260. Clarifying the `restrict` Keyword v2
N2265. Harmonizing `static_assert` with C++
N2271. CR for `pow` divide-by-zero case
N2314. TS 18661-1 plus CR/DRs for C2X

The integration of *TS 18661-1* has raised several questions, some of which were purely editorial, and some of which already gave rise to discussion on the reflector and with the FP study group. There will most certainly arise other changes that concern in particular the use of "`WANT`" macros in regular clauses, and more generally the testability of additions in new editions of the document.

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:

TS 18661-2. Decimal floating-point arithmetic (last working draft **N2341**)
N2124. rounding direction macro `FE_TONEARESTFROMZERO`

1.3. Held back

The following changes have been voted into C2x with a 6 month period for possible comments or objections and we expect to integrate them after the meeting:

N2293. Alignment requirements for memory management functions
N2302. `nextafterl(1.L,2.L)`

© 2019 Larry Jones and Jens Gustedt. Distributed under a Creative Commons Attribution 4.0 International License

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 miss-interpretation of `--` (decrement operator) as – (long hyphen) has been chased down.

Other editorial changes are a bit more involved.

2.1. 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 6 (Portability) that categorizes identifiers used by the document. It consists of a list of 29 regular expressions that systematically reserve 462 identifiers. Not matched by these regular expressions are 808 identifiers. The lists and corresponding counts are generated automatically such that their maintenance should not add work for future editors.

2.2. 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.

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).