**Title:** Bit-precise I/O  
**Author, affiliation:** Aaron Ballman, Intel  
Elizabeth Andrews, Intel  
Melanie Blower (author emeritus)  
**Date:** 2021-10-08  
**Proposal category:** New features  
**Target audience:** C application programmers  
**Abstract:** C23 will have bit-precise integer types. These types would be strengthened by having facilities to input and output bit-precise values directly rather than going through an intermediary type.
Bit-precise I/O

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2824
Date: 2021-10-08

Summary of Changes
N2824
- Original proposal, split off from N2590

Introduction and Rationale
C23 will have a new bit-precise integer type that can represent signed or unsigned arbitrary-precision integer values. However, it currently lacks a facility for writing such values to an output stream or reading them from an input stream.

The `strfrom*` family of functions are used to convert a value into a string but would be inappropriate to use for bit-precise integer objects. Instead, the function signature would have to accept a `void *` which points to the `_BitInt(N)`, an integer to specify the bit-width `N`, and information about the sign, which would be a novel signature for the family of functions.

In some cases, the user can judiciously use explicit casts when calling an I/O function like `printf`, but there are two major pitfalls with this approach: forgetting to add the explicit cast will always result in undefined behavior due to type mismatches (integer promotion does not provide a safety net), and the cast will only work if the destination type can represent the bit-precise value (and so is not viable for bit-precise integer types wider than `intmax_t/uintmax_t`).

Based on the work done for the specific-width length modifier that was adopted in C23 from N2680, we propose a new length modifier, `wb`, which must be followed by an integer `N` to describe that the corresponding argument is a `_BitInt` of width `N`. The signedness of the argument’s type is determined by the conversion specifier. A new format specifier for the `_BitInt` type is required because `_BitInt` is not converted during default argument promotion, so the exact type and width are required when calling `va_arg` to interpret the `_BitInt(N)` value. It would not be appropriate to reuse the `wN` specifier because `_BitInt(N)` and `intN_t` are distinct types.

Proposed Straw Polls
We would like bit-precise integer types to be supported by existing I/O facilities in C23. To that end, we would like to poll the following:

*Does WG14 wish to adopt NXXXX into C23?*

Proposed Wording
The wording proposed is a diff from WG14 N2596 with WG14 N2763 and WG14 N2680 applied. Green text is new text, while red text is deleted text.

Modify 7.21.6.1p7 to add a new bullet after the `wfN` modifier:
wbN Specifies that a following d, i, o, u, x, or X conversion specifier applies to a bit-precise integer argument with a width N where N is a positive decimal integer with no leading zeros; or that a following n conversion specifier applies to a pointer to a signed bit-precise integer argument with a width of N bits. All values of N less than or equal to BITINT_MAXWIDTH (5.2.4.2.1) shall be supported. It is implementation-defined if values greater than BITINT_MAXWIDTH are supported.

Modify 7.21.6.1p8: Drafting note: this resolves confusion over whether int is meant syntactically or not; given that long long int is a distinct type from int, yet is already supported via %lld, we assume that this an editorial change.

\[ \begin{align*}
\text{d, i} & \quad \text{The int integer argument is converted…} \\
\text{o, u, x, X} & \quad \text{The unsigned int unsigned integer argument is converted…}
\end{align*} \]

Modify 7.22.6.2p11 to add a new bullet after the wbN modifier:

wbN Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument which is a pointer to a bit-precise integer with a width N where N is a positive decimal integer with no leading zeros. All values of N less than or equal to BITINT_MAXWIDTH (5.2.4.2.1) shall be supported. It is implementation-defined if values greater than BITINT_MAXWIDTH are supported.

Modify 7.29.2.1p7 to add a new bullet after the wbN modifier:

wbN Specifies that a following d, i, o, u, x, or X conversion specifier applies to a bit-precise integer argument with a width N where N is a positive decimal integer with no leading zeros; or that a following n conversion specifier applies to a pointer to a signed bit-precise integer argument with a width of N bits. All values of N less than or equal to BITINT_MAXWIDTH (5.2.4.2.1) shall be supported. It is implementation-defined if values greater than BITINT_MAXWIDTH are supported.

Modify 7.29.2.1p8: Drafting note: this resolves confusion over whether int is meant syntactically or not; given that long long int is a distinct type from int, yet is already supported via %lld, we assume that this an editorial change.

\[ \begin{align*}
\text{d, i} & \quad \text{The int integer argument is converted…} \\
\text{o, u, x, X} & \quad \text{The unsigned int unsigned integer argument is converted…}
\end{align*} \]

Modify 7.29.2.2p11 to add a new bullet after the wbN modifier:

wbN Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument which is a pointer to a bit-precise integer with a width N where N is a positive decimal integer with no leading zeros. All values of N less than or equal to BITINT_MAXWIDTH (5.2.4.2.1) shall be supported. It is implementation-defined if values greater than BITINT_MAXWIDTH are supported.
Acknowledgements
I would like to recognize the following people for their help with this work: Robert Seacord.

References
[N2763]

[N2680]