Proposal for C2x
WG14 N2587

Title:	Specific bit-width length modifier
Author, affiliation:	Robert C. Seacord, NCC Group
Date:	2020-10-19
Proposal category:	Feature
Target audience:	Implementers supporting fixed-width and extended integer types
Abstract:	Add specific bit-width length modifier to formatted IO functions
Prior art:	C

Specific bit-width length modifier

Reply-to: Robert C. Seacord (rcseacord@gmail.com)
Document No: **N2587**
Reference Document: **N2511, N2465**
Date: 2020-

Proposal N2465 **"**intmax_t, a way forward" was presented at the Spring 2020 meeting but failed to gain support. However, there was support for specific bit-width length modifier that was a small component of the broader proposal. N2511 was presented at the October 2020 virtual meeting, but incorrectly only referenced exact-width types.

Change Log

2020-10-14:

- removed "exact-width" from proposed text
- clarified applicability to N-bit integers

1. PROBLEM DESCRIPTION

The C Standard allows for implementation-defined *extended signed integer types* (6.2.5, p4) and corresponding *extended unsigned integer types* (6.2.5, p6). There is no portable mechanism for specifying the width of extended integer types when passing them as arguments to formatted input and output functions.

Section 7.20.1 of the C Standard defines integer types including minimum-width integer types (7.20.1.2) and exact-width integer types (7.20.1.1). For example, the typedef name **`intN_t`** designates a signed integer type with width N, no padding bits, and a two's complement representation. Consequently, **`int8_t`** denotes such a signed integer type with a width of exactly 8 bits. Specific bit-width modifiers provide a more usable mechanism for passing minimum-width and exact-width integer types as arguments to formatted input and output functions than the **`PRI`** and **`SCN`** macros for format specifiers specified in Section 7.8.1.

The fastest types defined in Section 7.20.1.3, "Fastest minimum-width integer types" cannot be differentiated from other types and are not supported by the specific bit-width length modifier.

2. SUGGESTED CHANGES

The width of a type can be specified using a **specific bit-width length modifier** in a manner that can be understood by both the implementation and the library. If the library doesn't support the specified width, the formatted input or output function can return an error.

The length modifier uses a lowercase letter because uppercase letters are reserved for implementation extensions. Avoiding the letters used in the standard and various TRs leaves bqvw. In this case, we decided to use **`'w'`** to denote the *width* of the value and to preserve **`'b'`** to support *binary* output in the future. 128-bit integers, for example, will appear as follows:

```
uint128_t all = -1;
printf("the largest set is %w128d\n", all);
```

A **`w`** followed by a positive decimal integer followed by a **`d`**, **`i`**, **`o`**, **`u`**, **`x`**, or **`X`** conversion specifier applies to an integer argument with a specific bit-width where *N* is a decimal integer (the argument will have been promoted according to the integer promotions, but its value shall be converted to *N* bits before printing); or that a following n conversion specifier applies to a pointer to an integer type argument with a width of *N* bits. All values of *N* for which corresponding minimum-width integer types (7.20.1.2) and exact-width integer types (7.20.1.1) are defined in the header **`<stdint.h>`** shall be supported. Other supported values of *N* are implementation-defined.

There is some relevant implementation experience. Microsoft **printf** has **I32** and **I64** for this purpose. The use of **I** is in the space reserved for implementation extensions and other implementations use **I** for other things, but it's still relevant experience should we wish to support w<width> for that purpose. Microsoft also uses **'w'** in extensions, but only with string and character formats so that wouldn't conflict in any way with a standard use of **'w'**. Existing **PRI** and **SCN** macros in header **<inttypes.h>** also provides implementation experience.

Small types that are subject to integer promotions will work correctly. Consider the following code fragment:

```
uint8_t i = 1;
printf("%w8d", i);
```

The argument **i** is promoted to an **int** when passed to the formatted output function. The implementation must anticipate this and correctly process promoted arguments.

In the following code fragment, both arguments are hexadecimal literals of type **int**:

```
printf("%w8d %w8d", 0xFF, 0x1FF);
```

Only the low order 8 bits are considered when determining the sign and magnitude of the values. Consequently, this code prints the following to the standard output stream:

```
-1 -1
```

7.21.6.1 The fprintf function

Add the following to paragraph 7:

…

t — Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **ptrdiff_t** or the corresponding unsigned integer type argument; or that a following n conversion specifier applies to a pointer to a **ptrdiff_t** argument.

wN — Specifies that a following **d**, **i**, **o**, **u**, **x**, or **X** conversion specifier applies to an integer argument with a specific bit-width where *N* is a positive decimal integer (the argument will have been promoted according to the integer promotions, but its value shall be converted to *N* bits before printing); or that a following **n** conversion specifier applies to a pointer to an integer type argument with a width of *N* bits. All minimum-width integer types (7.20.1.2) and exact-width integer types (7.20.1.1) defined in the header **<stdint.h>** shall be supported. Other supported values of *N* are implementation-defined.

L — Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to a **long double** argument.

…

Modify paragraph 14 as follows:

The **fprintf** function returns the number of characters transmitted, or a negative value if an output or encoding error occurred or if the implementation does not support a specified width length modifier.

7.21.6.2 The fscanf function

Add the following to paragraph 11:

…

t — Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **ptrdiff_t** or the corresponding unsigned integer type.

wN Specifies that a following **d**, **i**, **o**, **u**, **x**, or **X**, or **n** conversion specifier applies to an argument which is a pointer to an integer with a specific bit-width where *N* is a positive decimal integer. All minimum-width integer types (7.20.1.2) and exact-width integer types (7.20.1.1) defined in the header **<stdint.h>** shall be supported. Other supported values of *N* are implementation-defined.

L Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to **long double**.

…

Modify paragraph 16 as follows:

The **fscanf** function returns the value of the macro **EOF** if an input failure occurs before the first conversion (if any) has completed. Otherwise, the function returns the number of input items assigned, which can be fewer than provided for, or even zero, in the event of an early matching failure or if the implementation does not support a specific bit-width length modifier.

7.29.2.1 The fwprintf function

Add the following to paragraph 7:

…

t Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **ptrdiff_t** or the corresponding unsigned integer type argument; or that a following n conversion specifier applies to a pointer to a **ptrdiff_t** argument.

wN Specifies that a following **d**, **i**, **o**, **u**, **x**, or **X** conversion specifier applies to an integer argument with a specific bit-width where *N* is a positive decimal integer (the argument will have been promoted according to the integer promotions, but its value shall be converted to *N* bits before printing); or that a following **n** conversion specifier applies to a pointer to an integer type argument with a width of *N* bits. All minimum-width integer types (7.20.1.2) and exact-width integer types (7.20.1.1) defined in the header **<stdint.h>** shall be supported. Other supported values of *N* are implementation-defined.

L Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to a **long double** argument.

…

Modify paragraph 14 as follows:

The **fwprintf** function returns the number of wide characters transmitted, or a negative value if an output or encoding error occurred or if the implementation does not support a specific bit-width length modifier.

7.29.2.2 The fwscanf function

Add the following to paragraph 11:

…

t Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **ptrdiff_t** or the corresponding unsigned integer type.

wN Specifies that a following **d**, **i**, **o**, **u**, **x**, or **X**, or **n** conversion specifier applies to an argument which is a pointer to an integer with a specific bit-width where *N* is a positive decimal integer. All minimum-width integer types (7.20.1.2) and exact-width integer types (7.20.1.1) defined in the header **<stdint.h>** shall be supported. Other supported values of *N* are implementation-defined.

L Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to **`long double`**.

…

The **`fwscanf`** function returns the value of the macro **`EOF`** if an input failure occurs before the first conversion (if any) has completed. Otherwise, the function returns the number of input items assigned, which can be fewer than provided for, or even zero, in the event of an early matching failure or if the implementation does not support a specific bit-width length modifier.

4.0 Acknowledgements

I would like to recognize the following people for their help with this work: Joseph S. Myers, Aaron Ballman, and Melanie Blower.

5.0 References

N2465 Seacord, intmax_t, a way forward
http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2465.pdf