C2X Proposal: WG14 N2498

Title:            intmax_t, again
Author:           Martin Uecker, University Medical Center Göttingen
Date:             2020-02-29

Synopsis

(u)intmax_t is preserved in its current form and keeping its promise. Language feature which
make it difficult to change (u)intmax_t to a larger type because this requires ABI breaking
changes are deprecated and replaced by future-proof alternatives.

Introduction

(u)intmax_t is intended to be the largest standard type so that it can be safely used as a generic
integer type that can represent the values of all other standard integer types of same signedness.
Unfortunately, the C standard introduced several APIs where (u)intmax_t is used in a way
which makes it impossible to change it to a wider type without breaking binary compatibility.
Because of this problem, some implementations introduced new large integer types bigger than
(u)intmax_t and in this way broke the promise that it is the largest integer type. The APIs in
question are the conversion specifier j for input/output functions and several functions in the
standard library. This topic was discussed in N2425 and N2465.

Conversion Specifier

We simply declare the j length specifier an obsolescent feature. The corresponding PRI and
SCN macros are retained unmodified and implementation can later redefine these macros as
needed when (u)intmax_t needs to be changed. Here, we also adopt the new length modifier w
as proposed in N2465 that defines modifiers for all exact-width types which can then be
generated by the macros.

Library Functions

Existing functions affected:

imaxabs, imaxdiv, strtoimax, strtoumax, wcstoimax, wcstoumax

New functions affected:

compoundn, pown, rootn, fromfp, ufromfp, fromfpx, ufromfpx

For compoundn, pown, rootn, it seems desirable that they use a largest standard integer
type, but not necessarily a much larger implementation-defined extended integer type. Therefore,
the type of these function is changed to use long long int.
The functions \texttt{imaxabs}, \texttt{imaxdiv}, \texttt{strtoimax}, \texttt{strtoumax}, \texttt{wcstoimax}, \texttt{wcestoumax}, \texttt{romfrp}, \texttt{uframfp}, \texttt{fromfpx}, \texttt{uromfpx} are deprecated as library functions and will be redefined as function-like macros. In particular, in the future it will not be required anymore that there exists an underlying library function of the same name. An implementation can then redefine these macros as needed if the definition of \texttt{(u)intmax_t} changes.

**Rationale**

This proposal tries to keep \texttt{(u)intmax_t} as intended while deprecating those parts which make it difficult for implementations to honor its promise. It avoids major changes, but it sets the foundations for fixing things later. After a transitional period implementations will be able to introduce extended integer types by changing \texttt{(u)intmax_t} without breaking binary compatibility. The intentions and expectations are made clear in the standard.

**Future-Proof Interfaces**

Functions which use \texttt{(u)intmax_t} should be specified as function-like macro or as functions taking a size parameter and a pointer to the integer. The size and pointer could also be combined into a struct type. It is possible to wrap such interfaces into type-safe macro wrappers. Implementations could also perform their own type checking similar to how this is done for format specifiers. Future introduction of a bignum type, or expanded use of generic functions could also make it easier to develop generic and type-safe interfaces. It may be wise to restrict the use of \texttt{(u)intmax_t} even further as proposed here. To facilitate a future transition to a very big integer type which may have some usage restriction or to a proper bignum type of unbounded width, we could consider current restrictions on VLA types and disallow the use of \texttt{(u)intmax_t} in structs and at file scope, maybe also of assignment (allowing only initialization, arithmetic, passing as argument, and conversion to other integer and pointer types).

**Suggested Wording Changes**

7.8.2 Functions for greatest-width integer types*\)

7.8.2.1 The \texttt{imaxabs} function
7.8.2.2 The \texttt{imaxdiv} function
7.8.2.3 The \texttt{strtoimax} and \texttt{strtoumax} functions
7.8.2.4 The \texttt{wcstoimax} and \texttt{wcstoumax} functions

*) These functions are obsolescent as library functions and will be changed to function-like macros that will be provided under the same
names. This is intended to make it possible to change the (u)intmax_t types in the future without breaking binary interfaces.

7.20.1.5 Greatest-width integer types

... 

NOTE Because any introduction of larger standard or extended integer type means that the greatest-width types change, extra care is required when these types are to be used in interfaces that require a stable application binary interface.

7.21.6.1 The fprintf function
7.29.2.1 The fwprintf function

...

7 The length modifiers and their meanings are:*)

...

wN Specifies that a following d, i, o, u, x, or X conversion specifier applies to an exact-width integer type argument of exactly N bits; or that a following n conversion specifier applies to a pointer to an exact-width integer type argument of exactly N bits.

...

*) The length modifier j for input/output functions is an obsolescent feature.

7.21.6.2 The fscanf function
7.29.2.2 The fwscanf function

...

11 The length modifiers and their meanings are:*)

...

wN Specifies that a following d, i, o, u, x, or X, or n conversion specifier applies to a pointer to an exact-width integer type argument of exactly N bits.

...
The length modifier \( j \) for input/output functions is an obsolescent feature.

7.31.4 Format conversion of integer types <inttypes.h>

Functions for greatest-width integer types are an obsolescent features as library functions and this functionality will be provided as function-like macros under the same name.

7.31 Future library directions

7.31.11 Input/output <stdio.h>

The length modifier \( j \) for input/output functions is an obsolescent feature.

Additional Suggested Wording Changes

4 ...  
   – An optional length modifier that specifies the size of the argument. In some cases, the length modifier may be followed by an asterisk.

5 As noted above, a field width, or precision, or length, or both any combination, may be indicated by an asterisk. In this case, an int argument supplies the field width, or precision, or length. The arguments specifying field width, or precision, or length, or both any combination, shall appear (in that order) before the argument (if any) to be converted. A negative field width argument is taken as a - flag followed by a positive field width. A negative precision argument is taken as if the precision were omitted.

7.21.6.1 The fprintf function
7.29.2.1 The fwprintf function
7.21.6.2 The fscanf function
7.29.2.2 The fwscanf function

\( \text{wN} \) ..... to an exact-width integer type argument of exactly \( N \) bits. If \( N \) is given as an asterisk, the pointer argument is preceded by an integer argument indicating the length as described above.

© 2020 by the author(s). Distributed under a Creative Commons Attribution 4.0 International License.