Document Number: N3400

Submitter: Aaron Peter Bachmann

Submission Date: 2024-11-22

Reducing undefined behavior for printf-style functions

Summary

If a single format specifier produces more than 4095 characters the behavior is undefined. Lift this limit. Make this limit a define queryable at compile time. Make this limit large enough to print all values of all floating-point types supported, provided that field with and precision are not explicitly specified. This is a follow up for N3184, reduced in scope to focus on the reduction of undefined behavior.

Problem description

`printf("%Lf",-LBL_MAX);` produces 4942 characters for both IEEE754 2008 128 bit binary floating point numbers as well as the 80 bit floating point format still widely used (e. g. i386). It is unfortunate that that brings us into the realm of undefined behavior.

Printing `-DEC128_MAX` via `%DDf` requires 6146 characters.

Printing the smallest negative subnormal binary128 number exactly requires 16450 characters (including the leading "`-0.`").

Printing more than 4095 characters with a single "`%s`" is not unusual.

The problem is well known and has been partly addressed by C99. Back then the committee lifted the limit from 509 characters to 4095 characters.

C provides `strfromf()`, `strfromd()`, `strfromencfN()`, `strfromfNx()`, `strtofN()`, `strtofNx()`. All these functions refer to `printf()`. Thus, the same 4095 character limit applies.

The return value of the printf-style functions is `int`. `INT_MAX` can be as small as 32767. Most implementations support producing more than 2e9 characters for a single format specifier. So “one value fits all” is problematic.

Proposed solution:

- Make the number of characters an implementation can safely produce implementation defined.
- Require it to be suitable to print any single primitive type.
- Introduce a new define in `<stdio.h>` giving that value. A value only buried somewhere in the documentation is less helpful.
- Lift the minimal value to 16383[1].
- Make the changes to the wide character functions as well.

[1] That is an arbitrary value larger than 4095, larger than 6145 and representable in a plain 16 bit integer not expected to impose undue burdens onto implementations.

Not proposed:

No distinction between different format specifiers is proposed. Anyone can submit a separate proposal introducing an additional (larger) value for e. g. the `%s` format specifier.

Wording changes relative to n3301:

In 7.23.1 before

_PRINT_NAN_LEN_MAX

insert:

_PRINTF_SINGLE_CONVERSION_LEN_MAX

which expands to an integer constant expression (suitable for use in conditional expression inclusion directive) that is the maximum number of characters output from any single printf-format specifier.

_PRINTF_SINGLE_CONVERSION_LEN_MAX shall be no less than 16383.

_PRINTF_SINGLE_CONVERSION_LEN_MAX shall be suitable for the output of every fundamental type provided no precision and field lengths are explicitly specified.

In 7.23.6.1 replace paragraph 16

~~The number of characters that can be produced by any single conversion shall be at least 4095.~~

with

The number of characters that can be produced by any single conversion shall be at least _PRINTF_SINGLE_CONVERSION_LEN_MAX.

In 7.31.2.1 replace paragraph 16

~~The number of characters that can be produced by any single conversion shall be at least 4095.~~

with

The number of characters that can be produced by any single conversion shall be at least _PRINTF_SINGLE_CONVERSION_LEN_MAX.

Possible straw polls:

Does the committee wish to adopt N3400?

Acknowledgements:

I want to thank Thomas Kemmer for proofreading.

References:

[1] https://www.open-std.org/jtc1/sc22/wg14/www/docs/n3184.pdf

[2] https://www.open-std.org/jtc1/sc22/wg14/www/docs/n3301.pdf