WG 14, n2517

Clarification request for C17 s3.4.3p3

David Svoboda
svoboda@cert.org
Date: 2020-04-27

This clarification request originated from document n2466 (Towards Integer Safety).

In C17, ‘overflow’ is a condition where the result of an operation cannot be represented in the associated type of the operation result. Both signed and unsigned integer operations may overflow. Silent wrap-around is a behavior that can occur as a result of overflow.

Confusingly, 3.4.3 p3 states:

> EXAMPLE An example of undefined behavior is the behavior on integer overflow.

(In the C18 draft (n2479), this is now p4.)

However, 6.2.5 p9 clarifies unsigned integer behavior:

> A computation involving unsigned operands can never overflow, because a result that cannot be represented by the resulting unsigned integer type is reduced modulo the number that is one greater than the largest value that can be represented by the resulting type.

Conventionally, signed integer overflow is considered undefined in C but unsigned integer overflow is defined to silently wrap.

The C17 standard does not define the term "overflow". But the term is used in enough contexts that its specific definition can be inferred as the computation of a mathematical value (of integer or floating-point type) that cannot be represented in the type associated with the computation expression.

It might be worthwhile to add this definition of "overflow" to C17. However, this might require clarifications on most uses of the term in the document. An easier solution is to use a different example to showcase undefined behavior.

RECOMMENDATION: 3.4.3p4 should use a different example of undefined behavior, such as:

> EXAMPLE An example of undefined behavior is the behavior on dereferencing a null pointer.