Proposal for C2x
WG14 N2762

Title:	Fixes for potentially reserved identifiers
Author, affiliation:	Aaron Ballman, Intel
	Hubert Tong, IBM
Date:	2021-06-29
Proposal category:	Wording fixes
Target audience:	Implementers and readers of the C standard

Abstract: The C committee adopted WG14 N2625, which changes the way in which reserved identifiers are specified by the standard. After adoption, there were some clarification requests made on the WG14 reflectors. This document clarifies those parts of N2625 that were not sufficiently clear.

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2762
Date: 2021-06-29

Summary of Changes

N2762

- Original proposal

Introduction and Rationale

At the Dec 2020 meeting, WG14 adopted N2625, which split reserved identifiers into two categories: reserved (UB for a user to use the identifier) and potentially reserved (identifiers that may become potentially reserved by an implementation or a future revision of the standard, but are otherwise not currently reserved). After adoption, some concerns were raised on the reflectors whether the wording matched the intent of the original proposal. Specifically, there was concern that the wording suggests that implementations diagnose declarations of potentially reserved identifiers even when said declarations are known to be safe for accessing functionality provided by the implementation.

Consider a hypothetical implementation that provides the function `char *strawesome(char *)` in `<string.h>`. For a user who wants to avoid including `<string.h>` while still wishing to use the `strawesome()` functionality, they may provide a declaration like: `extern char *strawesome(char*)` in their source code. The current wording in C23 suggests that the extern declaration be diagnosed as using a potentially-reserved identifier, but a quality implementation should recognize that this is not an incorrect use because it is a forward-compatible declaration to access the functionality provided by the hypothetical implementation.

This applies not only to identifiers with external linkage provided by an implementation that may be standardized in the future, but also to macros that an implementation may introduce like new `errno` error values or new locales.

Proposal

This paper adds more precision to the wording about what is a potentially reserved identifier and under what circumstances to diagnose the identifiers. As part of refining the circumstances under which potentially reserved macro names are diagnosed, the association of the macros with the headers that are anticipated to provide them is strengthened.

It clarifies that a potentially reserved identifier becomes reserved by an implementation or a new revision of the standard providing it, as this connection was previously inferred rather than explicitly stated. The identifier remains “potentially reserved”. It is noted that any UB associated with the “reserved” status conferred by the implementation would obviate any *requirement* to produce a diagnostic; however, a *recommendation* is not so affected.

It also clarifies that potentially reserved identifiers provided by the implementation as a macro are only reserved when the header file actually providing the macro is included, whereas identifiers provided by the implementation with external linkage are reserved for any use.

It adds the notion of an implementation-compatible use of an identifier, which is a use intended to declare something provided by the implementation. In other words, it's fine for a user program to provide a declaration of a reserved identifier that resolves to a compatible external definition provided by the implementation. For functions and objects, the declaration of the reserved identifier must have a compatible type with the definition provided by the implementation. No macro definitions are deemed implementation-compatible, because UB occurs when a user defines a macro in contexts where it is reserved.

Proposed Wording

The wording proposed is a diff from the committee draft of WG14 N2596. Green text is new text, while ~~red~~ text is deleted text.

Modify 6.4.2.1p8:

Some identifiers may be potentially reserved. A *potentially reserved identifier* is an identifier which is not reserved unless made so by an implementation providing the identifier (7.1.3) but is anticipated to become reserved by an implementation or a future version of this document.

Modify 6.4.2.1p9:

Implementations are encouraged to issue a diagnostic message when a potentially reserved identifier is declared or defined for any use that is not implementation-compatible (see below) in a context where the potentially reserved identifier may be reserved under a conforming implementation. This brings ~~to bring~~ attention to a potential conflict when porting a program to a future revision of this document.

Add 6.4.2.1p10 to the Recommended Practice section:

An implementation-compatible use of a potentially reserved identifier is a declaration of an external name where the name is provided by the implementation as an external name and where the declaration declares an object or function with a type that is compatible with the type of the object or function provided by the implementation under that name.

Modify 7.1.3p1: (*Drafting note: the changes for identifiers with external linkage are meant to prevent an implementation from reserving, for any use and without header inclusion, identifiers that are potentially reserved for use as typedefs, etc.*)

Each header declares or defines all identifiers listed in its associated subclause, and optionally declares or defines identifiers listed in its associated future library directions subclause and identifiers which are always reserved either for any use or for use as file scope identifiers.

— All potentially reserved identifiers (including ones listed in the future library directions) that are provided by an implementation with an external definition are reserved for any use. An implementation shall not provide an external definition of a potentially reserved identifier unless that identifier is reserved for a use where it would have external linkage.[x)] All other potentially reserved identifiers that are provided by an implementation (including in the form of a macro) are reserved for any use when the associated header is included. No other potentially reserved identifiers are reserved.[199)]

Add a new foonote for [x)]:

All library functions have external linkage.

Modify 7.31p1:

~~The following names are~~ Although grouped under individual headers, ~~for convenience. All~~ all of the external names ~~described~~ identified as reserved identifiers or potentially reserved identifiers in this subclause remain so regardless of which ~~below are reserved no matter what~~ headers are included by the program.

Acknowledgements

The authors would like to recognize the following people for their help in this work: Jens Gustedt, Joseph Myers, and Robert Seacord.