Make implicit undefined behaviour in mtx_destroy() explicit

Reply-to: dbanham at qnx.com
Document number: N3655
Date: 2025-07-21

Summary of Changes

- Initial proposal

Problem Statement

In ISO 9899:2024 subsection §7.28.4.2 The description of the `mtx_destroy()` function states "No threads can be blocked waiting for the mutex pointed to by `mtx`." This is because a mutex should not be destroyed when it is in the locked state as it could leave other threads in a permanently blocked-waiting state. In C-language standard terms, this possibility should be declared as an undefined behaviour (UB). This is not clearly indicated by the standard and the use of "can" to introduce the condition as a possibility is unhelpful when an erroneous program state could arise.

Analysis

The POSIX v8 standard for `pthread_mutex_destroy()` function states:
"Attempting to destroy a locked mutex, or a mutex that another thread is attempting to lock, or a mutex that is being used in a `pthread_cond_clockwait()`, `pthread_cond_timedwait()`, or `pthread_cond_wait()` call by another thread, results in undefined behavior."

This text is clear and explicit about the potential for a UB.

Nevertheless, other implementations of the C Mutex could define a behaviour for a destroyed mutex when it is locked, perhaps by unblocking the affected threads and returning an error to them. Such an implementation would also have to deal with the race condition occurring when another thread is attempting to lock a mutex when it is destroyed, perhaps by not destroying the mutex until no threads can refer to it (e.g., when the process terminates). However, as such behaviours are not defined by the C Standard they are either *undefined* or at best *implementation defined*. In the general case though, the more accurate classification would be that of an *undefined behaviour* because erroneous program behaviour could result.

The two prominent public domain C coding standards treat the behaviour as *undefined* and have rules to guard against it. These are CERT rule CON31-C (and for the POSIX equivalent, rule POS48-C), and MISRA C:2025 rule 22.15.

Proposed wording

There are (at least) two possible ways to make the undefined behaviour in mtx_destroy() explicit when a locked mutex is destroyed:

Option 1:
The simple fix is to change “can be” to “shall be” so that it becomes a requirement violation that is inherently a UB, per clause 4.
Amend the description of `mtx_destroy()` as follows:

> No threads ~~can~~ shall be blocked waiting for the mutex pointed to by `mtx`.

Option 2:
A more direct solution is to state the UB explicitly, along similar lines to the POSIX text quoted in the above analysis.
Amend the description of `mtx_destroy()` as follows:

> The `mtx_destroy` function releases any resources used by the mutex pointed to by `mtx`. ~~No threads can be blocked waiting for the mutex pointed to by `mtx`.~~ Attempting to destroy a locked mutex, or a mutex that another thread is attempting to lock, or a mutex that is being used in a `cnd_timedwait()`, or `cnd_wait()` call by another thread, results in undefined behavior.