N1907
Proposal for DR469

Blaine Garst
2014-11-10

While searching **7.26.4 Mutex functions** for an appropriate place to place the Suggested TC of DR469 revealed that additional repairs are needed.

Issue 1:

C11 Section 7.26.4.2 **`mtx_init`** function p2 states "must have one of the six values" and then enumerates only four, this is a typo from when we removed additional proposed functionality.

Proposed Technical Corrigendum

In 7.26.4.2p2 replace the phrase

> which must have one of the six values:

with

> which must have one of the four values:

Issue 2:

Recursive locks have thread specific behaviors, namely ongoing success on locking an already locked mutex, as specified in 7.26.4.3p2. Each successive lock operation must also be matched with a call to **`mtx_unlock`**, yet this is not stated.

Proposed Technical Corrigendum

In 7.26.4.6 The **`mtx_unlock`** function p2

replace

> The **`mtx_unlock`** function unlocks the mutex pointed to by **`mtx`**. The mutex pointed to by **`mtx`** shall be locked by the calling thread.

with

> The mutex pointed to by **`mtx`** shall be locked by a prior call to **`mtx_lock`**, **`mtx_trylock`**, or **`mtx_timedlock`** in the calling thread. If the mutex is non-recursive, it is unlocked. If the mutex is recursive, **`mtx_unlock`** shall succeed without

unlocking the mutex for every successful prior locking call while the lock was held, and shall unlock the mutex when there have been no further successful locking calls.*

*Every successful locking call to a mutex must be matched by a call to **`mtx_unlock`**.

Issue 3:

From DR469, we wish to state explicitly that an operation on a mutex that remains locked after thread termination results in undefined behavior.

Proposed Technical Corrigendum

To 7.26.4.6 The **`mtx_unlock`** function, add a new paragraph after paragraph 2:

The behavior of a program is undefined if a thread terminates without unlocking a mutex that it has locked.