IEEE Standards Interpretations for IEEE Std 1003.1c-1995 IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) - System Application Program Interface (API) Amendment 2: Threads Extension (C Language)

These are interpretations of IEEE Std 1003.1c-1995.

Interpretation Request #2
Topic: pthread_key_delete **Relevant Clauses:** 2.8.4, 17.1.1.4, 17.1.3

Regarding sections 2.8.4, 17.1.1.4, and 17.1.3, may/must pthread_key_delete make the key value available for re-use? Section 2.8.4 defines PTHREAD_KEYS_MAX as being the number of data keys that can created per process. 17.1.1.4 says that pthread_key_create shall return EAGAIN if "the system- imposed limit on the total number of keys per process {PTHREAD_KEYS_MAX} has been exceeded." 17.1.3.2 the rationale for pthread_key_delete, says "A thread-specific data key deletion function has been included in order to allow the resources associated with an unused thread-specific data key to be freed."

Consider the following example. Assume that this application does nothing else with any thread-specific data.

pthread_key_t keys[PTHREAD_KEYS_MAX], key_A, key_B; int i, status_A, status_B; struct foobar data[PTHREAD_KEYS_MAX]; for (i = 0; i < PTHREAD_KEYS_MAX; i++) { /* these calls all succeed */ (void) pthread_key_create (&(keys[i]), NULL); pthread_setspecific (keys[i], (void *) &(data[i])); } pthread_key_delete(keys[5]); /* this should succeed */ status_A = pthread_key_create (&key_A, NULL); /* Unclear about this */ pthread_setspecific (keys[6], (void *) NULL); /* this should succeed */ pthread_key_delete(keys[6]); /* this should succeed */ status_B = pthread_key_create (&key_B, NULL);/* Unclear about this */ Questions: 1) May status_A be 0? (I recommend "yes") 1.1) If so, may key_A equal keys[5]? (I recommend "yes") 2) Must status_A be 0? (I recommend "no") 3) May status_A be EAGAIN? (I recommend

"yes") 4) Must status_A be EAGAIN? (I recommend "no") 5) May status_B be 0? (I recommend "yes") 5.1) If so, may key_B equal keys[6]? (I recommend "yes") 6) Must status_B be 0? (I recommend "no") 7) May status_A be EAGAIN? (I recommend "yes") 8) Must status_A be EAGAIN? (I recommend "no") I expect that the answers to questions 1-4 will be identical to the answers to 5-8, but I wanted to point out the distinction of having a non-NULL thread-specific data value at the time of the pthread_key_delete call.

The answers I have recommended above coincide with my reading of the normative text of the standard, but the wording in the 17.1.3.2 rationale for pthread_key_delete leaves me uneasy.

While I have your attention, I'll add another question that is somewhat redundant with the above. 9) In 2.8.4, should the definition for PTHREAD_KEYS_MAX, which reads "Maximum number of data keys that can be created per process", be interpreted to not include in the count those data keys that have been deleted? If so, I would suggest changing the definition to "Maximum number of data keys that can exist per process".

Interpretation Response
The standard states that once the system defined limit: PTHREADS_KEYS_MAX keys have been created, the call shall return the indicated error, and conforming implementations must conform to this. However, concerns have been raised about this behavior which are being forwarded to the sponsor. In the view of the interpretations committee, this is not what the balloting group intended. The provision of a Delete function and the rationale for this function make the intent very clear that keys should be reusable with a limit on the number of "simultaneous" keys. The wording in the definition should possibly have been similar to POSIX_OPEN_MAX or POSIX_CHILD_MAX. This matter is being referred to the sponsor for consideration as a technical correction. Additionally, the interpretation committee noted that PTHREADS_THREADS_MAX may have a similar problem and have referred this to the sponsor as well. Specific answers: 1. - No 2. - No 3. - Yes, must be EAGIN 4. - Yes 5. - No 6. - No 7./8. same as 3/4

Rationale for Interpretation
None.