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.

Interpretations are issued to explain and clarify the intent of a standard and **do not** constitute an alteration to the original standard. In addition, interpretations are not intended to supply consulting information. Permission is hereby granted to download and print one copy of this document. Individuals seeking permission to reproduce and/or distribute this document in its entirety or portions of this document must contact the IEEE Standards Department for the appropriate license. Use of the information contained in this document is at your own risk.

IEEE Standards Department, Copyrights and Permissions, 445 Hoes Lane, Piscataway, New Jersey 08855-1331, USA

Interpretation Request #8
Topic: pthread_key_create() **Relevant Clauses:** 17.1.1.2

There seems to be an editorial error in IEEE Std 1003.1c-1995 regarding pthread_key_create() The pthread_key_create section, 17.1.1.2 of IEEE Std 1003.1c-1995, is very unclear. As written today it easily leads one to believe that it is required that the destructor function be called until the thread-specific value becomes NULL, or until PTHREAD_DESTRUCTOR_ITERATIONS iterations have occured, whichever comes first. If you read the draft POSIX 1003.1c standard revisions, it's clear what the intent was. Between draft standards a sentence was lost which said "Before each destructor is called, the thread's value for the corresponding key is set to NULL". The re-iteration exists so that if the destructor uses keys, they will be destroyed in a re-call to the destructor. Was this omission an editorial error?

Interpretation Response
The standard states that the destructor is called with the current associated value as its only argument and it is repeatedly called until PTHREAD_DESTRUCTOR_ITERATIONS or the value associated with the key becomes null. Conforming implementations must conform to this. However, concerns have been raised that the intent of the working and balloting groups was somewhat different and that the associated value should automatically set to NULL before the destructor is called in order to prevent infinite loops. This is needed since the destructor function does not have a way to determine which key that caused it to be invoked and thus does not have a way to null the value. The committee feels that this is a problem and it is being referred to the sponsor for consideration.

Rationale for Interpretation
None.