

## **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)**

Copyright © 1996 by the Institute of Electrical and Electronics Engineers, Inc. 345 East 47th Street New York, New York 10017 USA All Rights Reserved.

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 #22**

**Topic:** pthread\_setspecific **Relevant Clauses:** 17.1.2.2, page 166 D10, lines 192-195

This paragraph states that calling pthread\_setspecific() from within a destructor function may result in lost storage or infinite loops. This can be a real problem. POSIX.1c has just stated that it is impossible to use thread-specific data in an application. A portable and correctly behaving application cannot rely on calling pthread\_setspecific() from within a destructor function. What makes thread-specific data impossible to use is the fact that in pthread\_key\_create() it is stated that when a thread terminates, any non-NULL TSD values which have destructors will have the destructor called for them.

The problem here is that the destructor function is called in a loop (potentially forever -- so a portable application has to assume forever) until the key value is NULL. However, according to pthread\_setspecific() a portable and correctly behaving application has no way possible to change the key value to a NULL value. Portable application have to rely on "worst-case" guaranteed behavior.

According to the definitions of pthread\_setspecific() and pthread\_key\_create() an application cannot reliably make use of thread-specific data with destructor functions or its thread will end up in an infinite loop. Could you please clarify the intended behavior of these functions? Obviously there must be something missing here or this looping behavior for destructor functions would not have been added since an application must assume it results in an infinite loop.

### **Interpretation Response**

This is a duplicate. See Interpretation #3, part 6.

**Rationale for Interpretation**

None.