

## **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.

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 #12**

**Topic:** thread-specific data **Relevant Clauses:** 3.1.3.2, page 27 D10, lines 84-94

This function allows an application to essentially install cancellation type handlers to guarantee that the proper state is maintained in the child process after a fork. What is supposed to happen with allocated thread-specific data in the child process? These functions are, in effect, executed by the thread calling fork. If the other threads have allocated a lot of thread-specific data, there is no way for the process to release that memory. The child has immediately inherited a memory leak when using TSD. Is this intended? There is no way of using `pthread_atfork()` to release other threads' thread-specific data.

### **Interpretation Response**

This is a duplicate. See Interpretation #3, part 11.

### **Rationale for Interpretation**

None.