Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'pthread_cancel.3p' command

$ man pthread_cancel.3p

PTHREAD_CANCEL(3P) POSIX Programmer's Manual PTHREAD_CANCEL(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

pthread_cancel ? cancel execution of a thread

SYNOPSIS

#include <pthread.h>

int pthread_cancel(pthread_t thread);

DESCRIPTION

The pthread_cancel() function shall request that thread be canceled. The target thread's cancelability state and type determines when the cancellation takes effect. When the cancellation is acted on, the can? cellation cleanup handlers for thread shall be called. When the last cancellation cleanup handler returns, the thread-specific data destruc? tor functions shall be called for thread. When the last destructor function returns, thread shall be terminated.

The cancellation processing in the target thread shall run asyn? chronously with respect to the calling thread returning from pthread_cancel().

RETURN VALUE

If successful, the pthread_cancel() function shall return zero; other?
wise, an error number shall be returned to indicate the error.

ERRORS

The pthread_cancel() function shall not return an error code of [EINTR].

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Two alternative functions were considered for sending the cancellation notification to a thread. One would be to define a new SIGCANCEL signal that had the cancellation semantics when delivered; the other was to define the new pthread_cancel() function, which would trigger the can? cellation semantics.

The advantage of a new signal was that so much of the delivery criteria were identical to that used when trying to deliver a signal that making cancellation notification a signal was seen as consistent. Indeed, many implementations implement cancellation using a special signal. On the other hand, there would be no signal functions that could be used with this signal except pthread_kill(), and the behavior of the delivered cancellation signal would be unlike any previously existing defined signal.

The benefits of a special function include the recognition that this signal would be defined because of the similar delivery criteria and that this is the only common behavior between a cancellation request and a signal. In addition, the cancellation delivery mechanism does not have to be implemented as a signal. There are also strong, if not stronger, parallels with language exception mechanisms than with sig? nals that are potentially obscured if the delivery mechanism is visibly closer to signals.

In the end, it was considered that as there were so many exceptions to

the use of the new signal with existing signals functions it would be misleading. A special function has resolved this problem. This func? tion was carefully defined so that an implementation wishing to provide the cancellation functions on top of signals could do so. The special function also means that implementations are not obliged to implement cancellation with signals.

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended that the function should fail and report an [ESRCH] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_exit(), pthread_cond_timedwait(), pthread_join(), pthread_set? cancelstate()

The Base Definitions volume of POSIX.1?2017, <pthread.h>

COPYRIGHT