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_setcanceltype.3' command

$ man pthread_setcanceltype.3

PTHREAD_SETCANCELSTATE(3) Linux Programmer's Manual PTHREAD_SETCANCELSTATE(3)

NAME

pthread_setcancelstate, pthread_setcanceltype - set cancelability state and type

SYNOPSIS

#include <pthread.h>

int pthread_setcancelstate(int state, int *oldstate);

int pthread_setcanceltype(int type, int *oldtype);

Compile and link with -pthread.

DESCRIPTION

The pthread_setcancelstate() sets the cancelability state of the call? ing thread to the value given in state. The previous cancelability state of the thread is returned in the buffer pointed to by oldstate.

The state argument must have one of the following values:

PTHREAD_CANCEL_ENABLE

The thread is cancelable. This is the default cancelability state in all new threads, including the initial thread. The thread's cancelability type determines when a cancelable thread will respond to a cancellation request.

PTHREAD_CANCEL_DISABLE

The thread is not cancelable. If a cancellation request is re? ceived, it is blocked until cancelability is enabled.

The pthread_setcanceltype() sets the cancelability type of the calling

thread to the value given in type. The previous cancelability type of the thread is returned in the buffer pointed to by oldtype. The type argument must have one of the following values:

PTHREAD_CANCEL_DEFERRED

A cancellation request is deferred until the thread next calls a function that is a cancellation point (see pthreads(7)). This is the default cancelability type in all new threads, including the initial thread.

Even with deferred cancellation, a cancellation point in an asynchronous signal handler may still be acted upon and the ef? fect is as if it was an asynchronous cancellation.

PTHREAD_CANCEL_ASYNCHRONOUS

The thread can be canceled at any time. (Typically, it will be canceled immediately upon receiving a cancellation request, but the system doesn't guarantee this.)

The set-and-get operation performed by each of these functions is atomic with respect to other threads in the process calling the same function.

RETURN VALUE

On success, these functions return 0; on error, they return a nonzero error number.

ERRORS

The pthread_setcancelstate() can fail with the following error:

EINVAL Invalid value for state.

The pthread_setcanceltype() can fail with the following error:

EINVAL Invalid value for type.

ATTRIBUTES

For an explanation of the terms used in this section, see at? tributes(7).

Interface	Attribute	Value
pthread_setcancelstate(),	Thread safety	MT-Safe

?pthread_setcanceltype() ? ? ?

??

?pthread_setcancelstate(), ? Async-cancel-safety ? AC-Safe ?

?pthread_setcanceltype() ? ? ?

??

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

NOTES

For details of what happens when a thread is canceled, see pthread_can?
cel(3).

Briefly disabling cancelability is useful if a thread performs some
critical action that must not be interrupted by a cancellation request.
Beware of disabling cancelability for long periods, or around opera?
tions that may block for long periods, since that will render the
thread unresponsive to cancellation requests.

Asynchronous cancelability

Setting the cancelability type to PTHREAD_CANCEL_ASYNCHRONOUS is rarely
useful. Since the thread could be canceled at any time, it cannot
safely reserve resources (e.g., allocating memory with malloc(3)), ac?
quire mutexes, semaphores, or locks, and so on. Reserving resources is
unsafe because the application has no way of knowing what the state of
these resources is when the thread is canceled; that is, did cancella?
tion occur before the resources were reserved, while they were re?
served, or after they were released? Furthermore, some internal data
structures (e.g., the linked list of free blocks managed by the mal?
loc(3) family of functions) may be left in an inconsistent state if
cancellation occurs in the middle of the function call. Consequently,
clean-up handlers cease to be useful.

Functions that can be safely asynchronously canceled are called async-
cancel-safe functions. POSIX.1-2001 and POSIX.1-2008 require only that
pthread_cancel(3), pthread_setcancelstate(), and pthread_setcancel?
type() be async-cancel-safe. In general, other library functions can't
be safely called from an asynchronously cancelable thread.

One of the few circumstances in which asynchronous cancelability is useful is for cancellation of a thread that is in a pure compute-bound loop.

Portability notes

The Linux threading implementations permit the oldstate argument of pthread_setcancelstate() to be NULL, in which case the information about the previous cancelability state is not returned to the caller. Many other implementations also permit a NULL oldstat argument, but POSIX.1 does not specify this point, so portable applications should always specify a non-NULL value in oldstate. A precisely analogous set of statements applies for the oldtype argument of pthread_setcancel?type().

EXAMPLES

See pthread_cancel(3).

SEE ALSO

pthread_cancel(3), pthread_cleanup_push(3), pthread_testcancel(3), pthreads(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.