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_setcancelstate.3p' command

$ man pthread_setcancelstate.3p

PTHREAD_SETCANCELSTATE(3P) POSIX Programmer's ManualPTHREAD_SETCANCELSTATE(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_setcancelstate, pthread_setcanceltype, pthread_testcancel ? set cancelability state

SYNOPSIS

```
#include <pthread.h>
int pthread_setcancelstate(int state, int *oldstate);
int pthread_setcanceltype(int type, int *oldtype);
void pthread_testcancel(void);
```

DESCRIPTION

The pthread_setcancelstate() function shall atomically both set the calling thread's cancelability state to the indicated state and return the previous cancelability state at the location referenced by old? state. Legal values for state are PTHREAD_CANCEL_ENABLE and PTHREAD_CANCEL_DISABLE.

The pthread_setcanceltype() function shall atomically both set the calling thread's cancelability type to the indicated type and return the previous cancelability type at the location referenced by oldtype.

Legal values for type are PTHREAD_CANCEL_DEFERRED and PTHREAD_CAN?
CEL_ASYNCHRONOUS.

The cancelability state and type of any newly created threads, includ?
ing the thread in which main() was first invoked, shall be PTHREAD_CAN?
CEL_ENABLE and PTHREAD_CANCEL_DEFERRED respectively.

The pthread_testcancel() function shall create a cancellation point in
the calling thread. The pthread_testcancel() function shall have no ef?
fect if cancelability is disabled.

RETURN VALUE

If successful, the pthread_setcancelstate() and pthread_setcanceltype()
functions shall return zero; otherwise, an error number shall be re?
turned to indicate the error.

ERRORS

The pthread_setcancelstate() function may fail if:

EINVAL The specified state is not PTHREAD_CANCEL_ENABLE or PTHREAD_CAN?
CEL_DISABLE.

The pthread_setcanceltype() function may fail if:

EINVAL The specified type is not PTHREAD_CANCEL_DEFERRED or
PTHREAD_CANCEL_ASYNCHRONOUS.

These functions shall not return an error code of [EINTR].

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

In order to write a signal handler for an asynchronous signal which can
run safely in a cancellable thread, pthread_setcancelstate() must be
used to disable cancellation for the duration of any calls that the
signal handler makes which are cancellation points. However, the stan?
dard does not permit strictly conforming applications to call
pthread_setcancelstate() from a signal handler since it is not cur?
rently required to be async-signal-safe. On implementations where
pthread_setcancelstate() is not async-signal-safe, alternatives are to
ensure either that the corresponding signals are blocked during execu?

tion of functions that are not async-cancel-safe or that cancellation is disabled during times when those signals could be delivered. Imple? mentations are strongly encouraged to make pthread_setcancelstate() async-signal-safe.

RATIONALE

The pthread_setcancelstate() and pthread_setcanceltype() functions con? trol the points at which a thread may be asynchronously canceled. For cancellation control to be usable in modular fashion, some rules need to be followed.

An object can be considered to be a generalization of a procedure. It is a set of procedures and global variables written as a unit and called by clients not known by the object. Objects may depend on other objects.

First, cancelability should only be disabled on entry to an object, never explicitly enabled. On exit from an object, the cancelability state should always be restored to its value on entry to the object.

This follows from a modularity argument: if the client of an object (or the client of an object that uses that object) has disabled cancelabil? ity, it is because the client does not want to be concerned about cleaning up if the thread is canceled while executing some sequence of actions. If an object is called in such a state and it enables cancela? bility and a cancellation request is pending for that thread, then the thread is canceled, contrary to the wish of the client that disabled.

Second, the cancelability type may be explicitly set to either deferred or asynchronous upon entry to an object. But as with the cancelability state, on exit from an object the cancelability type should always be restored to its value on entry to the object.

Finally, only functions that are cancel-safe may be called from a thread that is asynchronously cancelable.

FUTURE DIRECTIONS

The pthread_setcancelstate() function may be added to the table of async-signal-safe functions in Section 2.4.3, Signal Actions.

SEE ALSO

pthread_cancel()

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

COPYRIGHT