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_mutexattr_getrobust.3p' command

$ man pthread_mutexattr_getrobust.3p

PTHREAD_MUTEXATTR_GETROBUSTPOSIX Programmer's MPTHREAD_MUTEXATTR_GETROBUST(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_mutexattr_getrobust, pthread_mutexattr_setrobust ? get and set the mutex robust attribute

SYNOPSIS

```
#include <pthread.h>
int pthread_mutexattr_getrobust(const pthread_mutexattr_t *restrict
    attr, int *restrict robust);
int pthread_mutexattr_setrobust(pthread_mutexattr_t *attr,
    int robust);
```

DESCRIPTION

The pthread_mutexattr_getrobust() and pthread_mutexattr_setrobust() functions, respectively, shall get and set the mutex robust attribute. This attribute is set in the robust parameter. Valid values for robust include:

PTHREAD_MUTEX_STALLED

No special actions are taken if the owner of the mutex is termi? nated while holding the mutex lock. This can lead to deadlocks if

no other thread can unlock the mutex.

This is the default value.

PTHREAD_MUTEX_ROBUST

If the process containing the owning thread of a robust mutex terminates while holding the mutex lock, the next thread that acquires the mutex shall be notified about the termination by the return value [EOWNERDEAD] from the locking function. If the owning thread of a robust mutex terminates while holding the mutex lock, the next thread that attempts to acquire the mutex may be notified about the termination by the return value [EOWNERDEAD]. The notified thread can then attempt to make the state protected by the mutex consistent again, and if successful can mark the mutex state as consistent by calling pthread_mutex_consistent(). After a subsequent successful call to pthread_mutex_unlock(), the mutex lock shall be released and can be used normally by other threads. If the mutex is unlocked without a call to pthread_mutex_consistent(), it shall be in a permanently unusable state and all attempts to lock the mutex shall fail with the error [ENOTRECOVERABLE]. The only permissible operation on such a mutex is pthread_mutex_destroy().

The behavior is undefined if the value specified by the attr argument to pthread_mutexattr_getrobust() or pthread_mutexattr_setrobust() does not refer to an initialized mutex attributes object.

RETURN VALUE

Upon successful completion, the pthread_mutexattr_getrobust() function shall return zero and store the value of the robust attribute of attr into the object referenced by the robust parameter. Otherwise, an error value shall be returned to indicate the error. If successful, the pthread_mutexattr_setrobust() function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The pthread_mutexattr_setrobust() function shall fail if:

EINVAL The value of robust is invalid.

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

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The actions required to make the state protected by the mutex consis? tent again are solely dependent on the application. If it is not possi? ble to make the state of a mutex consistent, robust mutexes can be used to notify this situation by calling pthread_mutex_unlock() without a prior call to pthread_mutex_consistent().

If the state is declared inconsistent by calling pthread_mutex_unlock() without a prior call to pthread_mutex_consistent(), a possible approach could be to destroy the mutex and then reinitialize it. However, it should be noted that this is possible only in certain situations where the state protected by the mutex has to be reinitialized and coordina? tion achieved with other threads blocked on the mutex, because other? wise a call to a locking function with a reference to a mutex object invalidated by a call to pthread_mutex_destroy() results in undefined behavior.

RATIONALE

If an implementation detects that the value specified by the attr argu? ment to pthread_mutexattr_getrobust() or pthread_mutexattr_setrobust() does not refer to an initialized mutex attributes object, it is recom? mended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_mutex_consistent(), pthread_mutex_destroy(), pthread_mu? tex_lock()

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

COPYRIGHT

table Operating System Interface (POSIX), The Open Group Base Specifi? cations Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker? nel.org/doc/man-pages/reporting_bugs.html .