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_mutex_timedlock.3p' command

$ man pthread_mutex_timedlock.3p

PTHREAD_MUTEX_TIMEDLOCK(3P)POSIX Programmer's ManuaPTHREAD_MUTEX_TIMEDLOCK(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_mutex_timedlock ? lock a mutex

SYNOPSIS

```
#include <pthread.h>
#include <time.h>
int pthread_mutex_timedlock(pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abstime);
```

DESCRIPTION

The pthread_mutex_timedlock() function shall lock the mutex object ref? erenced by mutex. If the mutex is already locked, the calling thread shall block until the mutex becomes available as in the pthread_mu? tex_lock() function. If the mutex cannot be locked without waiting for another thread to unlock the mutex, this wait shall be terminated when the specified timeout expires.

The timeout shall expire when the absolute time specified by abstime passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds abstime), or if the ab?

solute time specified by abstime has already been passed at the time of the call.

The timeout shall be based on the CLOCK_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The timespec data type is defined in the <time.h> header.

Under no circumstance shall the function fail with a timeout if the mu? tex can be locked immediately. The validity of the abstime parameter need not be checked if the mutex can be locked immediately.

As a consequence of the priority inheritance rules (for mutexes ini? tialized with the PRIO_INHERIT protocol), if a timed mutex wait is ter? minated because its timeout expires, the priority of the owner of the mutex shall be adjusted as necessary to reflect the fact that this thread is no longer among the threads waiting for the mutex.

If mutex is a robust mutex and the process containing the owning thread terminated while holding the mutex lock, a call to pthread_mutex_timed? lock() shall return the error value [EOWNERDEAD]. If mutex is a robust mutex and the owning thread terminated while holding the mutex lock, a call to pthread_mutex_timedlock() may return the error value [EOWN? ERDEAD] even if the process in which the owning thread resides has not terminated. In these cases, the mutex is locked by the thread but the state it protects is marked as inconsistent. The application should en? sure that the state is made consistent for reuse and when that is com? plete call pthread_mutex_consistent(). If the application is unable to recover the state, it should unlock the mutex without a prior call to pthread_mutex_consistent(), after which the mutex is marked permanently unusable.

If mutex does not refer to an initialized mutex object, the behavior is undefined.

RETURN VALUE

If successful, the pthread_mutex_timedlock() function shall return zero; otherwise, an error number shall be returned to indicate the er? ror.

ERRORS

The pthread_mutex_timedlock() function shall fail if:

EAGAIN The mutex could not be acquired because the maximum number of recursive locks for mutex has been exceeded.

EDEADLK

The mutex type is PTHREAD_MUTEX_ERRORCHECK and the current thread already owns the mutex.

EINVAL The mutex was created with the protocol attribute having the value PTHREAD_PRIO_PROTECT and the calling thread's priority is higher than the mutex' current priority ceiling.

EINVAL The process or thread would have blocked, and the abstime parameter specified a nanoseconds field value less than zero or greater than or equal to 1000 million.

ENOTRECOVERABLE

The state protected by the mutex is not recoverable.

EOWNERDEAD

The mutex is a robust mutex and the process containing the previous owning thread terminated while holding the mutex lock. The mutex lock shall be acquired by the calling thread and it is up to the new owner to make the state consistent.

ETIMEDOUT

The mutex could not be locked before the specified timeout expired.

The pthread_mutex_timedlock() function may fail if:

EDEADLK

A deadlock condition was detected.

EOWNERDEAD

The mutex is a robust mutex and the previous owning thread terminated while holding the mutex lock. The mutex lock shall be acquired by the calling thread and it is up to the new owner to make the state consistent.

This function shall not return an error code of [EINTR].

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Applications that have assumed that non-zero return values are errors will need updating for use with robust mutexes, since a valid return for a thread acquiring a mutex which is protecting a currently inconsistent state is [EOWNERDEAD]. Applications that do not check the error returns, due to ruling out the possibility of such errors arising, should not use robust mutexes. If an application is supposed to work with normal and robust mutexes, it should check all return values for error conditions and if necessary take appropriate action.

RATIONALE

Refer to pthread_mutex_lock().

FUTURE DIRECTIONS

None.

SEE ALSO

pthread_mutex_destroy(), pthread_mutex_lock(), time()

The Base Definitions volume of POSIX.1?2017, Section 4.12, Memory Synchronization, <pthread.h>, <time.h>

COPYRIGHT