NAME

pthread_mutexattr_init, **pthread_mutexattr_destroy**, **pthread_mutexattr_setprioceiling**, **pthread_mutexattr_getprioceiling**, **pthread_mutexattr_setprotocol**, **pthread_mutexattr_getprotocol**, **pthread_mutexattr_setpshared**, **pthread_mutexattr_getpshared**, **pthread_mutexattr_setrobust**, **pthread_mutexattr_getrobust**, **pthread_mutexattr_settype**, **pthread_mutexattr_gettype** - mutex attribute operations

LIBRARY

POSIX Threads Library (libpthread, -lpthread)

SYNOPSIS

#include <pthread.h>

int
pthread_mutexattr_init(*pthread_mutexattr_t *attr*);

int
pthread_mutexattr_destroy(*pthread_mutexattr_t *attr*);

int
pthread_mutexattr_setprioceiling(*pthread_mutexattr_t *attr*, *int prioceiling*);

int
pthread_mutexattr_getprioceiling(*const pthread_mutexattr_t *attr*, *int *prioceiling*);

int
pthread_mutexattr_setprotocol(*pthread_mutexattr_t *attr*, *int protocol*);

int
pthread_mutexattr_getprotocol(*const pthread_mutexattr_t *restrict attr*, *int *restrict protocol*);

int
pthread_mutexattr_setpshared(*pthread_mutexattr_t *attr*, *int shared*);

int
pthread_mutexattr_getpshared(*const pthread_mutexattr_t *attr*, *int *shared*);

int
pthread_mutexattr_setrobust(*pthread_mutexattr_t *attr*, *int robust*);

int
pthread_mutexattr_getrobust(*pthread_mutexattr_t *attr*, *int *robust*);

int
pthread_mutexattr_settype(*pthread_mutexattr_t *attr*, *int type*);

int
pthread_mutexattr_gettype(*const pthread_mutexattr_t *restrict attr*, *int *restrict type*);

DESCRIPTION

Mutex attributes are used to specify parameters to **pthread_mutex_init**(). One attribute object can be used in multiple calls to **pthread_mutex_init**(), with or without modifications between calls.

The **pthread_mutexattr_init**() function initializes *attr* with all the default mutex attributes.

The **pthread_mutexattr_destroy**() function destroys *attr*.

The **pthread_mutexattr_setprioceiling**() function sets the priority ceiling for the mutex, used by threads executed under the PTHREAD_PRIO_PROTECT protocol.

The **pthread_mutexattr_setprotocol**() function specifies the protocol to be followed in utilizing mutexes. The *protocol* argument can take one of the following values:

PTHREAD_PRIO_NONE   Priority and scheduling of the thread owning this mutex is not affected by its mutex ownership.

PTHREAD_PRIO_INHERIT   Request priority-inheritance protocol, where the thread owning the mutex is executed at the highest priority among priorities of all threads waiting on any mutex owned by this thread.

PTHREAD_PRIO_PROTECT   Request priority-inheritance protocol, where the thread owning the mutex is executed at highest priority among priorities or priority ceilings of all threads waiting on any mutex owned by this thread.

The **pthread_mutexattr_setpshared**() function sets the process-shared attribute of *attr* to the value specified in *pshared*. The argument *pshared* may have one of the following values:

PTHREAD_PROCESS_PRIVATE The mutex may only be used by threads in the same process as the one that created the object.

PTHREAD_PROCESS_SHARED The mutex may be used by threads in processes other than the one
                        that created the object, assuming other processes share access to the
                        memory where the mutex was allocated.
See libthr(3) for details of the implementation of the shared mutexes, and their limitations.

The **pthread_mutexattr_setrobust**() function specifies robustness attribute of the mutex. Possible values
for the *robust* argument are

PTHREAD_MUTEX_STALLED No special actions are taken if the thread owning the mutex is
                       terminated without unlocking 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, or
                      owning thread, terminates while holding the mutex lock, the next
                      thread that acquires the mutex is notified about the termination by the
                      return value EOWNERDEAD from the locking function. Then,
                      either pthread_mutex_consistent(3) can be used to repair the mutex
                      lock state, or pthread_mutex_unlock(3) can unlock the mutex lock
                      but also put it an unusable state, where all further attempts to acquire
                      it result in the ENOTRECOVERABLE error.

The **pthread_mutexattr_settype**() function sets the type of the mutex. The type affects the behavior of
calls which lock and unlock the mutex. The possible values for the *type* argument are

PTHREAD_MUTEX_NORMAL   Both recursive locking, and unlocking when the lock is not
                           owned by the current thread, cause an error to be returned from
                           the corresponding functions. This matches
                           PTHREAD_MUTEX_ERRORCHECK but somewhat
                           contradicts the behavior mandated by POSIX.

PTHREAD_MUTEX_ERRORCHECK Both recursive locking, and unlocking when the lock is not
                          owned by the current thread, cause an error returned from the
                          corresponding functions.

PTHREAD_MUTEX_RECURSIVE   Recursive locking is allowed. Attempt to unlock when current
                          thread is not an owner of the lock causes an error to be
                          returned.

PTHREAD_MUTEX_DEFAULT   The FreeBSD implementation maps this type to

PTHREAD_MUTEX_ERRORCHECK type.

The **pthread_mutexattr_get***() functions copy the value of the attribute that corresponds to each function name to the location pointed to by the second function parameter.

RETURN VALUES

If successful, these functions return 0. Otherwise, an error number is returned to indicate the error.

ERRORS

The **pthread_mutexattr_init**() function will fail if:

[ENOMEM]   Out of memory.

The **pthread_mutexattr_destroy**() function will fail if:

[EINVAL]   Invalid value for *attr*.

The **pthread_mutexattr_setprioceiling**() function will fail if:

[EINVAL]   Invalid value for *attr*, or invalid value for *prioceiling*.

The **pthread_mutexattr_getprioceiling**() function will fail if:

[EINVAL]   Invalid value for *attr*.

The **pthread_mutexattr_setprotocol**() function will fail if:

[EINVAL]   Invalid value for *attr*, or invalid value for *protocol*.

The **pthread_mutexattr_getprotocol**() function will fail if:

[EINVAL]   Invalid value for *attr*.

The **pthread_mutexattr_setpshared**() function will fail if:

[EINVAL]   Invalid value for *attr*, or invalid value for *shared*.

The **pthread_mutexattr_getpshared**() function will fail if:

[EINVAL]   Invalid value for *attr*.

The **pthread_mutexattr_settype**() function will fail if:

[EINVAL]   Invalid value for *attr*, or invalid value for *type*.

The **pthread_mutexattr_gettype**() function will fail if:

[EINVAL]   Invalid value for *attr*.

The **pthread_mutexattr_setrobust**() function will fail if:

[EINVAL]   Invalid value for *attr*, or invalid value for *robust*.

The **pthread_mutexattr_getrobust**() function will fail if:

[EINVAL]   Invalid value for *attr*.

SEE ALSO
libthr(3), pthread_mutex_init(3)

STANDARDS
The **pthread_mutexattr_init**() and **pthread_mutexattr_destroy**() functions conform to ISO/IEC 9945-1:1996 ("POSIX.1")

The **pthread_mutexattr_setprioceiling**(), **pthread_mutexattr_getprioceiling**(), **pthread_mutexattr_setprotocol**(), **pthread_mutexattr_getprotocol**(), **pthread_mutexattr_setpshared**(), **pthread_mutexattr_getpshared**(), **pthread_mutexattr_settype**(), and **pthread_mutexattr_gettype**() functions conform to Version 2 of the Single UNIX Specification ("SUSv2"). The **pthread_mutexattr_setrobust**() and **pthread_mutexattr_getrobust**() functions conform to Version 4 of the Single UNIX Specification ("SUSv4").