NAME

pthread_mutexattr_init, pthread_mutexattr_destroy, pthread_mutexattr_setpshared, pthread_mutexattr_getpshared, pthread_mutexattr_setprotocol, pthread_mutexattr_getprotocol, pthread_mutexattr_setprioceiling, pthread_mutexattr_getprioceiling – mutex initialization attributes

SYNOPSIS

```
#include <pthread.h>

int pthread_mutexattr_init(pthread_mutexattr_t *attr);
int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);
int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr, int process-shared);
int pthread_mutexattr_getpshared(const pthread_mutexattr_t *attr, int *process-shared);
```

MT-LEVEL

MT-Safe

DESCRIPTION

Initialize

pthread_mutexattr_init( ) initializes a mutex attributes object, *attr*, with the default value for its attribute, which is **PTHREAD_PROCESS_PRIVATE**. If the *process-shared* attribute is **PTHREAD_PROCESS_PRIVATE**, only threads created within the same process as the thread that initialized the mutex can access the mutex. If threads of differing processes attempt to access the mutex, the behavior is unpredictable.

Attempts to initialize an already initialized mutex variable attributes object will leave the storage allocated by the previous initialization unallocated.

Once a mutex attributes object is used to initialize one or more mutexes, any function that affects the attributes object (including destruction) will not affect any previously initialized mutexes.

Destroy

pthread_mutexattr_destroy( ) destroys a mutex attributes object; the object will then become uninitialized. A destroyed mutex attributes object can be reinitialized using **pthread_mutexattr_init( )**. The results of referencing the object after it has been destroyed are undefined.

Set/Get Scope

pthread_mutexattr_setpshared( ) and **pthread_mutexattr_getpshared( )** sets the *process-shared* attribute in an initialized attributes object pointed to by *attr*, and gets the value of the *process-shared* attribute from the attributes object pointed to by *attr*, respectively.

At present, only the attribute *process-shared* is defined.

Unsupported Interfaces

Currently, the following interfaces, which are optional under POSIX, are not supported:

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

RETURN VALUES

Upon successful completion, **pthread_mutexattr_init( )**, **pthread_mutexattr_destroy( )**, **pthread_mutexattr_setprotocol( )**, **pthread_mutexattr_getprotocol( )**, **pthread_mutexattr_setprioceiling( )**, **pthread_mutexattr_getprioceiling( ) ,** and **pthread_mutexattr_setpshared( )** return **0**; otherwise, an error number is returned.

Upon successful completion, **pthread_mutexattr_getpshared( )** returns **0** and stores the value of the *process-shared* attribute of *attr* in the object pointed to by the *process-shared* parameter; otherwise, an error number is returned.

ERRORS

The function **pthread_mutexattr_init( )** returns an error number if the following condition is detected:

ENOMEM Insufficient memory exists to initialize the mutex attributes object.

The functions **pthread_mutexattr_destroy( )**, **pthread_mutexattr_getpshared( )**, and **pthread_mutexattr_setpshared( )** return an error number if the following condition is detected:

EINVAL The value specified by *attr* is invalid.

The function **pthread_mutexattr_setpshared( )** returns an error number if the following condition is detected:

EINVAL The new value specified for the attribute is outside the range of legal values for that attribute.

Currently, the functions **pthread_mutexattr_setprotocol( )**, **pthread_mutexattr_getprotocol( )**, **pthread_mutexattr_setprioceiling( )**, and **pthread_mutexattr_getprioceiling( )** always return the following error code:

ENOSYS These optional interfaces are not supported.

SEE ALSO

pthread_cond_init(3T), **pthread_create**(3T), **pthread_mutex_init**(3T)

NOTES

The functions **pthread_mutexattr_setprotocol( )**, **pthread_mutexattr_getprotocol( )**, **pthread_mutexattr_setprioceiling( )**, and **pthread_mutexattr_getprioceiling( )** return **ENOSYS** in the current implementation, i.e., this function is not currently implemented.