NAME

pthread.h - threads

SYNOPSIS

```
#include <pthread.h>
```

DESCRIPTION

The *<pthread.h>* header shall define the following symbolic constants:

PTHREAD_BARRIER_SERIAL_THREAD
PTHREAD_CANCEL_ASYNCHRONOUS
PTHREAD_CANCEL_ENABLE
PTHREAD_CANCEL_DEFERRED
PTHREAD_CANCEL_DISABLE
PTHREAD_CANCELED
PTHREAD_CREATE_DETACHED
PTHREAD_CREATE_JOINABLE
[TPS]
PTHREAD_EXPLICIT_SCHED
PTHREAD_INHERIT_SCHED

PTHREAD_MUTEX_DEFAULT
PTHREAD_MUTEX_ERRORCHECK
PTHREAD_MUTEX_NORMAL
PTHREAD_MUTEX_RECURSIVE
PTHREAD_MUTEX_ROBUST
PTHREAD_MUTEX_STALLED
PTHREAD_ONCE_INIT
[RPI|TPI]
PTHREAD_PRIO_INHERIT

[MC1]
PTHREAD_PRIO_NONE

[RPP|TPP]
PTHREAD_PRIO_PROTECT

PTHREAD_PROCESS_SHARED
PTHREAD_PROCESS_PRIVATE
[TPS]
PTHREAD_SCOPE_PROCESS
PTHREAD_SCOPE_SYSTEM

The *<pthread.h>* header shall define the following compile-time constant expressions valid as initializers for the following types:

Name	Initializer for Type
PTHREAD_COND_INITIALIZER	**pthread_cond_t**
PTHREAD_MUTEX_INITIALIZER	**pthread_mutex_t**
PTHREAD_RWLOCK_INITIALIZER	**pthread_rwlock_t**

The *<pthread.h>* header shall define the **pthread_attr_t**, **pthread_barrier_t**, **pthread_barrierattr_t**, **pthread_cond_t**, **pthread_condattr_t**, **pthread_key_t**, **pthread_mutex_t**, **pthread_mutexattr_t**, **pthread_once_t**, **pthread_rwlock_t**, **pthread_rwlockattr_t**, **pthread_spinlock_t**, and **pthread_t** types as described in *<sys/types.h>*.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int   pthread_atfork(void (*)(void), void (*)(void),
          void(*)(void));
int   pthread_attr_destroy(pthread_attr_t *);
int   pthread_attr_getdetachstate(const pthread_attr_t *, int *);
int   pthread_attr_getguardsize(const pthread_attr_t *restrict,
          size_t *restrict);
[TPS]
int   pthread_attr_getinheritsched(const pthread_attr_t *restrict,
          int *restrict);

int   pthread_attr_getschedparam(const pthread_attr_t *restrict,
          struct sched_param *restrict);
[TPS]
int   pthread_attr_getschedpolicy(const pthread_attr_t *restrict,
          int *restrict);
int   pthread_attr_getscope(const pthread_attr_t *restrict,
          int *restrict);

[TSA TSS]
int   pthread_attr_getstack(const pthread_attr_t *restrict,
          void **restrict, size_t *restrict);

[TSS]
int   pthread_attr_getstacksize(const pthread_attr_t *restrict,
          size_t *restrict);

int   pthread_attr_init(pthread_attr_t *);
int   pthread_attr_setdetachstate(pthread_attr_t *, int);
int   pthread_attr_setguardsize(pthread_attr_t *, size_t);
[TPS]
int   pthread_attr_setinheritsched(pthread_attr_t *, int);

int   pthread_attr_setschedparam(pthread_attr_t *restrict,
          const struct sched_param *restrict);
[TPS]
int   pthread_attr_setschedpolicy(pthread_attr_t *, int);
int   pthread_attr_setscope(pthread_attr_t *, int);

[TSA TSS]
int   pthread_attr_setstack(pthread_attr_t *, void *, size_t);

[TSS]
int   pthread_attr_setstacksize(pthread_attr_t *, size_t);

int   pthread_barrier_destroy(pthread_barrier_t *);
int   pthread_barrier_init(pthread_barrier_t *restrict,
          const pthread_barrierattr_t *restrict, unsigned);
```

```
int   pthread_barrier_wait(pthread_barrier_t *);
int   pthread_barrierattr_destroy(pthread_barrierattr_t *);
[TSH]
int   pthread_barrierattr_getpshared(
          const pthread_barrierattr_t *restrict, int *restrict);

int   pthread_barrierattr_init(pthread_barrierattr_t *);
[TSH]
int   pthread_barrierattr_setpshared(pthread_barrierattr_t *, int);

int   pthread_cancel(pthread_t);
int   pthread_cond_broadcast(pthread_cond_t *);
int   pthread_cond_destroy(pthread_cond_t *);
int   pthread_cond_init(pthread_cond_t *restrict,
          const pthread_condattr_t *restrict);
int   pthread_cond_signal(pthread_cond_t *);
int   pthread_cond_timedwait(pthread_cond_t *restrict,
          pthread_mutex_t *restrict, const struct timespec *restrict);
int   pthread_cond_wait(pthread_cond_t *restrict,
          pthread_mutex_t *restrict);
int   pthread_condattr_destroy(pthread_condattr_t *);
int   pthread_condattr_getclock(const pthread_condattr_t *restrict,
          clockid_t *restrict);
[TSH]
int   pthread_condattr_getpshared(const pthread_condattr_t *restrict,
          int *restrict);

int   pthread_condattr_init(pthread_condattr_t *);
int   pthread_condattr_setclock(pthread_condattr_t *, clockid_t);
[TSH]
int   pthread_condattr_setpshared(pthread_condattr_t *, int);

int   pthread_create(pthread_t *restrict, const pthread_attr_t *restrict,
          void *(*)(void*), void *restrict);
int   pthread_detach(pthread_t);
int   pthread_equal(pthread_t, pthread_t);
void  pthread_exit(void *);
[OB XSI]
int   pthread_getconcurrency(void);

[TCT]
int   pthread_getcpuclockid(pthread_t, clockid_t *);

[TPS]
int   pthread_getschedparam(pthread_t, int *restrict,
          struct sched_param *restrict);

void *pthread_getspecific(pthread_key_t);
int   pthread_join(pthread_t, void **);
int   pthread_key_create(pthread_key_t *, void (*)(void*));
int   pthread_key_delete(pthread_key_t);
int   pthread_mutex_consistent(pthread_mutex_t *);
int   pthread_mutex_destroy(pthread_mutex_t *);
```

```
[RPP|TPP]
int   pthread_mutex_getprioceiling(const pthread_mutex_t *restrict,
          int *restrict);
int   pthread_mutex_init(pthread_mutex_t *restrict,
          const pthread_mutexattr_t *restrict);
int   pthread_mutex_lock(pthread_mutex_t *);
[RPP|TPP]
int   pthread_mutex_setprioceiling(pthread_mutex_t *restrict, int,
          int *restrict);
int   pthread_mutex_timedlock(pthread_mutex_t *restrict,
          const struct timespec *restrict);
int   pthread_mutex_trylock(pthread_mutex_t *);
int   pthread_mutex_unlock(pthread_mutex_t *);
int   pthread_mutexattr_destroy(pthread_mutexattr_t *);
[RPP|TPP]
int   pthread_mutexattr_getprioceiling(
          const pthread_mutexattr_t *restrict, int *restrict);
[MC1]
int   pthread_mutexattr_getprotocol(const pthread_mutexattr_t *restrict,
          int *restrict);
[TSH]
int   pthread_mutexattr_getpshared(const pthread_mutexattr_t *restrict,
          int *restrict);
int   pthread_mutexattr_getrobust(const pthread_mutexattr_t *restrict,
          int *restrict);
int   pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict,
          int *restrict);
int   pthread_mutexattr_init(pthread_mutexattr_t *);
[RPP|TPP]
int   pthread_mutexattr_setprioceiling(pthread_mutexattr_t *, int);
[MC1]
int   pthread_mutexattr_setprotocol(pthread_mutexattr_t *, int);
[TSH]
int   pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
int   pthread_mutexattr_setrobust(pthread_mutexattr_t *, int);
int   pthread_mutexattr_settype(pthread_mutexattr_t *, int);
int   pthread_once(pthread_once_t *, void (*)(void));
int   pthread_rwlock_destroy(pthread_rwlock_t *);
int   pthread_rwlock_init(pthread_rwlock_t *restrict,
          const pthread_rwlockattr_t *restrict);
int   pthread_rwlock_rdlock(pthread_rwlock_t *);
int   pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict,
          const struct timespec *restrict);
int   pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict,
          const struct timespec *restrict);
```

```
int   pthread_rwlock_tryrdlock(pthread_rwlock_t *);
int   pthread_rwlock_trywrlock(pthread_rwlock_t *);
int   pthread_rwlock_unlock(pthread_rwlock_t *);
int   pthread_rwlock_wrlock(pthread_rwlock_t *);
int   pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
[TSH]
int   pthread_rwlockattr_getpshared(
          const pthread_rwlockattr_t *restrict, int *restrict);

int   pthread_rwlockattr_init(pthread_rwlockattr_t *);
[TSH]
int   pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);

pthread_t
      pthread_self(void);
int   pthread_setcancelstate(int, int *);
int   pthread_setcanceltype(int, int *);
[OB XSI]
int   pthread_setconcurrency(int);

[TPS]
int   pthread_setschedparam(pthread_t, int,
          const struct sched_param *);
int   pthread_setschedprio(pthread_t, int);

int   pthread_setspecific(pthread_key_t, const void *);
int   pthread_spin_destroy(pthread_spinlock_t *);
int   pthread_spin_init(pthread_spinlock_t *, int);
int   pthread_spin_lock(pthread_spinlock_t *);
int   pthread_spin_trylock(pthread_spinlock_t *);
int   pthread_spin_unlock(pthread_spinlock_t *);
void  pthread_testcancel(void);
```

The following may be declared as functions, or defined as macros, or both. If functions are declared, function prototypes shall be provided.

pthread_cleanup_pop()
pthread_cleanup_push()

Inclusion of the *<pthread.h>* header shall make symbols defined in the headers *<sched.h>* and *<time.h>* visible.

The following sections are informative.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<sched.h>, <sys/types.h>, <time.h>

XSH pthread_atfork, pthread_attr_destroy, pthread_attr_getdetachstate, pthread_attr_getguardsize, pthread_attr_getinheritsched, pthread_attr_getschedparam, pthread_attr_getschedpolicy, pthread_attr_getscope, pthread_attr_getstack, pthread_attr_getstacksize, pthread_barrier_destroy, pthread_barrier_wait, pthread_barrierattr_destroy, pthread_barrierattr_getpshared, pthread_cancel, pthread_cleanup_pop, pthread_cond_broadcast, pthread_cond_destroy, pthread_cond_timedwait, pthread_condattr_destroy, pthread_condattr_getclock, pthread_condattr_getpshared, pthread_create, pthread_detach, pthread_equal , pthread_exit, pthread_getconcurrency, pthread_getcpuclockid, pthread_getschedparam, pthread_getspecific, pthread_join, pthread_key_create, pthread_key_delete, pthread_mutex_consistent, pthread_mutex_destroy, pthread_mutex_getprioceiling, pthread_mutex_lock, pthread_mutex_timedlock, pthread_mutexattr_destroy, pthread_mutexattr_getprioceiling, pthread_mutexattr_getprotocol, pthread_mutexattr_getpshared, pthread_mutexattr_getrobust, pthread_mutexattr_gettype, pthread_once, pthread_rwlock_destroy, pthread_rwlock_rdlock, pthread_rwlock_timedrdlock, pthread_rwlock_timedwrlock, pthread_rwlock_trywrlock, pthread_rwlock_unlock, pthread_rwlockattr_destroy, pthread_rwlockattr_getpshared, pthread_self, pthread_setcancelstate, pthread_setschedprio, pthread_spin_destroy, pthread_spin_lock, pthread_spin_unlock

CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6

The RTT margin markers are broken out into their POSIX options.

The Open Group Corrigendum U021/9 is applied, correcting the prototype for the pthread_cond_wait() function.

The Open Group Corrigendum U026/2 is applied, correcting the prototype for the pthread_setschedparam() function so that its second argument is of type **int**.

The pthread_getcpuclockid() and pthread_mutex_timedlock() functions are added for alignment with IEEE Std 1003.1d-1999.

The following functions are added for alignment with IEEE Std 1003.1j-2000: pthread_barrier_destroy(), pthread_barrier_init(), pthread_barrier_wait(), pthread_barrierattr_destroy(), pthread_barrierattr_getpshared(), pthread_barrierattr_init(), pthread_barrierattr_setpshared(), pthread_condattr_getclock(), pthread_condattr_setclock(), pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock(), pthread_spin_destroy(), pthread_spin_init(), pthread_spin_lock(), pthread_spin_trylock(), and pthread_spin_unlock().

PTHREAD_RWLOCK_INITIALIZER is removed for alignment with IEEE Std 1003.1j-2000.

Functions previously marked as part of the Read-Write Locks option are now moved to the Threads option.

The **restrict** keyword is added to the prototypes for pthread_attr_getguardsize(), pthread_attr_getinheritsched(), pthread_attr_getschedparam(), pthread_attr_getschedpolicy(), pthread_attr_getscope(), *pthread_attr_getstackaddr,()* pthread_attr_getstacksize(), pthread_attr_setschedparam(), pthread_barrier_init(), pthread_barrierattr_getpshared(), pthread_cond_init(), pthread_cond_signal(), pthread_cond_timedwait(), pthread_cond_wait(), pthread_condattr_getclock(), pthread_condattr_getpshared(), pthread_create(), pthread_getschedparam(), pthread_mutex_getprioceiling(), pthread_mutex_init(), pthread_mutex_setprioceiling(), pthread_mutexattr_getprioceiling(), pthread_mutexattr_getprotocol(), pthread_mutexattr_getpshared(), pthread_mutexattr_gettype(),

pthread_rwlock_init(), *pthread_rwlock_timedrdlock()*, *pthread_rwlock_timedwrlock()*, *pthread_rwlockattr_getpshared()*, and *pthread_sigmask()*.

IEEE PASC Interpretation 1003.1 #86 is applied, allowing the symbols from *<sched.h>* and *<time.h>* to be made visible when *<pthread.h>* is included. Previously this was an XSI option.

IEEE PASC Interpretation 1003.1c #42 is applied, removing the requirement for prototypes for the *pthread_kill()* and *pthread_sigmask()* functions. These are required to be in the *<signal.h>* header. They are allowed here through the name space rules.

IEEE PASC Interpretation 1003.1 #96 is applied, adding the *pthread_setschedprio()* function.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/13 is applied, correcting shading errors that were in contradiction with the System Interfaces volume of POSIX.1-2008.

Issue 7

SD5-XBD-ERN-55 is applied, adding the **restrict** keyword to the *pthread_mutex_timedlock()* function prototype.

SD5-XBD-ERN-62 is applied.

Austin Group Interpretation 1003.1-2001 #048 is applied, reinstating the PTHREAD_RWLOCK_INITIALIZER symbol.

The *<pthread.h>* header is moved from the Threads option to the Base.

The following extended mutex types are moved from the XSI option to the Base:

PTHREAD_MUTEX_NORMAL
PTHREAD_MUTEX_ERRORCHECK
PTHREAD_MUTEX_RECURSIVE
PTHREAD_MUTEX_DEFAULT

The PTHREAD_MUTEX_ROBUST and PTHREAD_MUTEX_STALLED symbols and the *pthread_mutex_consistent()*, *pthread_mutexattr_getrobust()*, and *pthread_mutexattr_setrobust()* functions are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to the Thread Priority Protection and Thread Priority Inheritance options is changed to be Non-Robust Mutex or Robust Mutex Priority Protection and Non-Robust Mutex or Robust Mutex Priority Inheritance, respectively.

This reference page is clarified with respect to macros and symbolic constants.

POSIX.1-2008, Technical Corrigendum 2, XBD/TC2-2008/0069 [624] is applied.

End of informative text.

return to top of page

[Main Index | XBD | XSH | XCU | XRAT]

<<< Previous Home Next >>>