EFREI

2009- 2010

Annexe TD1 (système d'exploitation)

Quelques fonctions de la librairie Threads POSIX

---

pthread_cond_init Subroutine

Purpose

Initializes a condition variable and sets its attributes.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_cond_init (condition, attr)
*pthread_cond_t *condition;*
*pthread_condattr_t *attr;*

Description

The pthread_cond_init subroutine initializes a new condition variable condition, and sets its attributes according the condition attributes object attr.

After initialization of the condition variable, the condition attributes object can be reused for another condition variable initialization, or deleted.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameters

condition	Specifies the condition to be created.

attr	Specifies the condition attributes object to use for initializing the condition variable. If the value is NULL, the default attributes values are used.

Return Values

Upon successful completion, the new condition variable is returned via the condition parameter, and 0 is returned. Otherwise, an error code is returned.

Error Codes

The pthread_cond_init subroutine is unsuccessful if the following is true:

EBUSY	The condition condition is already in use: it was previously created with by calling the pthread_cond_init subroutine, and not destroyed by calling the pthread_cond_destroy subroutine.

EINVAL	The condition or attr parameters are not valid.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_condattr_init subroutine, pthread_condattr_destroy subroutine, pthread_cond_wait or pthread_cond_timedwait subroutine, pthread_cond_destroy subroutine, PTHREAD_COND_INITIALIZER macro.

Using Condition Variables and Threads Library Quick Reference.

pthread_create Subroutine

Purpose

Creates a new thread, initializes its attributes, and makes it runnable.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_create (thread, attr, start_routine, arg)
*pthread_t *thread;*
*const pthread_attr_t *attr;*
*void *(*start_routine) (void *);*
*void *arg;*

Description

The pthread_create subroutine creates a new thread and initializes its attributes using the thread attributes object specified by the attr parameter. The new thread inherits its creating thread's signal mask; but any pending signal of the creating thread will be cleared for the new thread.

Note: The number of threads per process is defined in the pthread.h file as 512.

The new thread is made runnable, and will start executing the start_routine routine, with the parameter specified by the arg parameter. The arg parameter is a void pointer; it can reference any kind of data. It is not recommended to cast this pointer into a scalar data type (int for example), because the casts may not be portable.

After thread creation, the thread attributes object can be reused to create another thread, or deleted.

The thread terminates in the following cases:

* The thread returned from its starting routine (the main routine for the initial thread)

* The thread called the pthread_exit subroutine

* The thread was canceled

* The thread received a signal that terminated it

* The entire process is terminated due to a call to either the exec or exit subroutines.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameters

thread Points to where the thread ID will be stored.

attr Specifies the thread attributes object to use in creating the thread. If the value is NULL, the default attributes values will be used.

start_routine Points to the routine to be executed by the thread.

arg Points to the single argument to be passed to the start_routine routine.

Return Values

Upon successful completion, the new thread's ID is returned via the thread parameter, and 0 is returned. Otherwise, an error code is returned.

Error Codes

The pthread_create subroutine is unsuccessful if the following is true:

EAGAIN The system does not have sufficient resources to create another thread.

EINVAL The thread or attr parameters are not valid.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_attr_init subroutine, pthread_attr_destroy subroutine, pthread_exit subroutine, pthread_cancel subroutine, pthread_kill subroutine, pthread_self subroutine, pthread_once subroutine.

Creating Threads.

Threads Library Quick Reference.

pthread_exit Subroutine

Purpose

Terminates the calling thread.

Library

Threads Library (libpthreads.a)

Syntax

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

void pthread_exit (status)
void *status;

Description

The pthread_exit subroutine terminates the calling thread safely, and stores a termination status for any thread that may join the calling thread. The termination status is always a void pointer; it can reference any kind of data. It is not recommended to cast this pointer into a scalar data type (int for example), because the casts may not be portable. This subroutine never returns.

Unlike the exit subroutine, the pthread_exit subroutine does not close files. Thus any file opened and used only by the calling thread must be closed before calling this subroutine. It is also important to note that the pthread_exit subroutine frees any thread-specific data, including the thread's stack. Any data allocated on the stack becomes invalid, since the stack is freed and the corresponding memory may be reused by another thread. Therefore, thread synchronization objects (mutexes and condition variables) allocated on a thread's stack must be destroyed before the thread calls the pthread_exit subroutine.

Returning from the initial routine of a thread implicitly calls the pthread_exit subroutine, using the return value as parameter.

If the thread is not detached, its resources, including the thread ID, the termination status, the thread-specific data, and its storage, are all maintained until the thread is detached or the process terminates.

If another thread joins the calling thread, that thread wakes up immediately, and the calling thread is automatically detached.

If the thread is detached, the cleanup routines are popped from their
stack and executed. Then the destructor routines from the thread-specific
data are executed. Finally, the storage of the thread is reclaimed
and its ID is freed for reuse.

Terminating the initial thread by calling this subroutine does not
terminate the process, it just terminates the initial thread. However,
if all the threads in the process are terminated, the process is terminated
by implicitly calling the exit subroutine with a return code of 0
if the last thread is detached, or 1 otherwise.

Note: The pthread.h header file must be the first included file of
each source file using the threads library.

Parameters

status Points to an optional termination status, used by joining threads.
If no termination status is desired, its value should be NULL.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_cleanup_push subroutine, pthread_cleanup_pop subroutine,
pthread_key_create subroutine, pthread_create subroutine, pthread_join
subroutine, pthread_cancel subroutine, exit subroutine.

Terminating Threads and Threads Library Quick Reference.

pthread_mutex_lock or pthread_mutex_trylock Subroutine

Purpose

Locks a mutex.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_mutex_lock (mutex)
*pthread_mutex_t *mutex;*

int pthread_mutex_trylock (mutex)
*pthread_mutex_t *mutex;*

Description

These subroutines lock the mutex mutex. If the mutex is already locked, their behavior differs; the pthread_mutex_lock subroutine blocks the calling thread until the mutex is unlocked, while the pthread_mutex_trylock subroutine returns an error.

The pthread_mutex_lock and pthread_mutex_trylock subroutines return an error if the calling thread has already locked the mutex using one of these subroutines, preventing deadlocks caused by recursive locking.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameter

mutex Specifies the mutex to lock.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code is returned.

Error Codes

The pthread_mutex_lock and pthread_mutex_trylock subroutines is unsuccessful if the following is true:

EDEADLK A deadlock was detected; the calling thread has already locked the mutex.

EINVAL The mutex parameter is not valid.

The pthread_mutex_trylock subroutine fails if the following is true:

EBUSY The mutex mutex is already locked.

Implementation Specifics

These subroutines are part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_mutex_unlock subroutine, pthread_mutex_init subroutine, pthread_cond_wait or pthread_cond_timedwait subroutine.

Using Mutexes and Threads Library Quick Reference.

pthread_mutex_unlock Subroutine

Purpose

Unlocks a mutex.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_mutex_unlock (mutex)
*pthread_mutex_t *mutex;*

Description

The pthread_mutex_unlock subroutine unlocks the mutex mutex. It checks the mutex owner and resets the mutex only if the calling thread is the mutex owner; otherwise it returns an error.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameter

mutex Specifies the target mutex.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code is returned.

Error Codes

The pthread_mutex_unlock subroutine is unsuccessful if the following
is true:

EINVAL	The mutex parameter is not valid.
EPERM	The calling thread does not own the mutex lock.

Implementation Specifics

These subroutines are part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_mutex_lock or pthread_mutex_trylock subroutine.

Using Mutexes.

Threads Library Quick Reference.

pthread_cond_signal or pthread_cond_broadcast Subroutine

Purpose

Unblocks one or more threads blocked on a condition.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_cond_signal (condition)
*pthread_cond_t *condition;*

int pthread_cond_broadcast (condition)
*pthread_cond_t *condition;*

Description

These subroutines unblock one or more threads blocked on the condition specified by condition. The pthread_cond_signal subroutine unblocks at least one blocked thread, while the pthread_cond_broadcast subroutine unblocks all the blocked threads.

If no thread is blocked on the condition, the subroutine succeeds, but the signalling of the condition is not held. The next thread calling pthread_cond_wait will be blocked.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameter

condition	Specifies the condition to signal.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code is returned.

Error Code

The pthread_cond_signal and pthread_cond_broadcast subroutines are unsuccessful if the following is true:

EINVAL	The condition parameter is not valid.

Implementation Specifics

These subroutines are part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_cond_wait or pthread_cond_timedwait subroutine.

Using Condition Variables.

Threads Library Quick Reference.

pthread_cond_wait or pthread_cond_timedwait Subroutine

Purpose

Blocks the calling thread on a condition.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_cond_wait (condition, mutex)
*pthread_cond_t *condition;*
*pthread_mutex_t *mutex;*

int pthread_cond_timedwait (condition, mutex, timeout)
*pthread_cond_t *condition;*
*pthread_mutex_t *mutex;*
*const struct timespec *timeout;*

Description

These subroutines block the calling thread on the condition specified by condition. The condition variable will be protected by the mutex mutex. In addition, the pthread_cond_timedwait subroutine specifies a timeout for the blocked state.

The mutex must be locked before calling the subroutine. The subroutine atomically unlocks the mutex and blocks the calling thread until the condition is signaled. The mutex is locked again before the subroutine returns.

The pthread_cond_timedwait subroutine returns an error if the timeout occurs before the condition is signaled. The timeout is specified by an absolute date, not by a duration.

The pthread_cond_wait subroutine never returns if the condition is signalled only once before the call to this subroutine.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameters

condition Specifies the condition variable to wait on.

mutex Specifies the mutex used to protect the condition variable.
The mutex must be locked when the subroutine is called.

timeout Points to the absolute time structure specifying the blocked
state timeout.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code
is returned.

Error Codes

The pthread_cond_wait and pthread_cond_timedwait subroutines are unsuccessful
if the following is true:

EINVAL The condition or mutex parameters are not valid.

EDEADLK The mutex was not owned by the calling thread.

The pthread_cond_timedwait subroutine is unsuccessful if the following
is true:

EINVAL The timeout parameter is not valid.

ETIMEDOUTThe specified timeout has elapsed.

Implementation Specifics

These subroutines are part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_mutex_lock or pthread_mutex_trylock subroutine, pthread_cond_init
subroutine, pthread_cond_signal or pthread_cond_broadcast subroutine,
pthread_join subroutine.

Using Condition Variables.

Threads Library Quick Reference.

pthread_join, or pthread_detach Subroutine

Purpose

Blocks the calling thread until the specified thread terminates.

Library

Threads Library (libpthreads.a)

Syntax

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

int pthread_join (thread, status)
pthread_t thread;
*void **status;*

*int pthread_join (pthread_t thread, void **value_ptr);*

*int pthread_detach (pthread_t thread, **value_ptr);*

Description

The pthread_join subroutine blocks the calling thread until the thread
thread terminates. The target thread's termination status is returned
in the status parameter.

If the target thread is already terminated, but not yet detached,
the subroutine returns immediately. It is impossible to join a detached
thread, even if it is not yet terminated. The target thread is automatically
detached after all joined threads have been woken up.

This subroutine does not itself cause a thread to be terminated. It
acts like the pthread_cond_wait subroutine to wait for a special condition.

Note: The pthread.h header file must be the first included file of each
source file using the threads library. Otherwise, the -D_THREAD_SAFE
compilation flag should be used, or the cc_r compiler used. In this case, the
flag is automatically set.

The pthread_detach subroutine is used to indicate to the implementation that
storage for the thread whose thread ID is in the location thread can be
reclaimed when that thread terminates. This storage shall be reclaimed on
process exit, regardless of whether the thread has been detached or not, and

may include storage for thread return value. If thread has not yet terminated, pthread_detach shall not cause it to terminate. Multiple pthread_detach calls on the same target thread causes an error.

Parameters

thread Specifies the target thread.

status   Points to where the termination status of the target thread will be stored. If the value is NULL, the termination status is not returned.

Return Values

If successful, the pthread_join function returns zero. Otherwise, an error number is returned to indicate the error.

Error Codes

The pthread_join and pthread_detach functions will fail if:

EINVAL The implementation has detected that the value specified by thread does not refer to a joinable thread.

ESRCH No thread could be found corresponding to that specified by the given thread ID.

The pthread_join function will fail if:

EDEADLK The value of thread specifies the calling thread.

The pthread_join function will not return an error code of EINTR.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_exit subroutine, pthread_create subroutine, wait subroutine, pthread_cond_wait or pthread_cond_timedwait subroutines, the pthread.h file.

Joining Threads in AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs.

Threads Library Quick Reference in AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs.

pthread_mutex_init Subroutine

Purpose

Initializes a mutex and sets its attributes.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_mutex_init (mutex, attr)
*pthread_mutex_t *mutex;*
*pthread_mutexattr_t *attr;*

Description

The pthread_mutex_init subroutine initializes a new mutex mutex, and
sets its attributes according the mutex attributes object attr. The
mutex is initially unlocked.

After initialization of the mutex, the mutex attributes object can
be reused for another mutex initialization, or deleted.

Note: The pthread.h header file must be the first included file of
each source file using the threads library.

Parameters

mutex Specifies the mutex to be created.

attr Specifes the mutex attributes object to use for initializing
the mutex. If the value is NULL, the default attributes values are
used.

Return Values

Upon successful completion, the new mutex is returned via the mutex
parameter, and 0 is returned. Otherwise, an error code is returned.

Error Codes

The pthread_mutex_init subroutine is unsuccessful if the following
is true:

EAGAIN The system does not have sufficient resources, other than memory,
to initialize a new mutex.

EBUSY The mutex mutex is already in use: it was previously created
with by calling the pthread_mutex_init subroutine, and not destroyed
by calling the pthread_mutex_destroy subroutine.

EINVAL The mutex or attr parameters are not valid.

ENOMEM There is not sufficient memory to initialize a new mutex.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_mutexattr_init subroutine, pthread_mutexattr_destroy subroutine,
pthread_mutex_lock or pthread_mutex_trylock subroutine, pthread_mutex_destroy
subroutine, PTHREAD_MUTEX_INITIALIZER macro.

Using Mutexes.

Threads Library Quick Reference.

<u>pthread_mutex_unlock Subroutine</u>

Purpose

Unlocks a mutex.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_mutex_unlock (mutex)
*pthread_mutex_t *mutex;*

Description

The pthread_mutex_unlock subroutine unlocks the mutex mutex. It checks
the mutex owner and resets the mutex only if the calling thread is
the mutex owner; otherwise it returns an error.

Note: The pthread.h header file must be the first included file of
each source file using the threads library.

Parameter

mutex Specifies the target mutex.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code
is returned.

Error Codes

The pthread_mutex_unlock subroutine is unsuccessful if the following
is true:

EINVAL	The mutex parameter is not valid.
EPERM	The calling thread does not own the mutex lock.

Implementation Specifics

These subroutines are part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_mutex_lock or pthread_mutex_trylock subroutine.

Using Mutexes.

Threads Library Quick Reference.

pthread_cond_destroy Subroutine

Purpose

Deletes a condition variable.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_cond_destroy (condition)
*pthread_cond_t *condition;*

Description

The pthread_cond_destroy subroutine deletes the condition variable condition. After deletion of the condition variable, the condition parameter is not valid until it is initialized again by a call to the pthread_cond_init subroutine.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameter

condition	Specifies the condition variable to delete.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code is returned.

Error Codes

The pthread_cond_destroy subroutine is unsuccessful if the following is true:

EBUSY	The condition variable condition is referenced by another thread.
EINVAL	The condition parameter is not valid.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_cond_init subroutine.

Using Condition Variables.

Threads Library Quick Reference.

pthread_attr_setdetachstate Subroutine

Purpose

Sets the value of the detachstate attribute of a thread attributes
object.

Library

Threads Library (libpthreads.a)

Syntax

```
#include <pthread.h>

int pthread_attr_setdetachstate (attr, detachstate)
pthread_attr_t *attr;
int detachstate;
```

Description

The pthread_attr_setdetachstate subroutine sets the value of the detachstate
attribute of the thread attributes object attr. This attribute specifies
the detached state of a thread created with this attributes object.

Note: The pthread.h header file must be the first included file of
each source file using the threads library.

Parameters

attr Specifies the thread attributes object.

detachstate Specifies the detached state to set. It must have one
of the following values:

PTHREAD_CREATE_DETACHED Specifies that the thread will be created
in detached state. This is the default value.

PTHREAD_CREATE_UNDETACHED Specifies that the thread will be created
in undetached state.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code
is returned.

Error Codes

The pthread_attr_setdetachstate subroutine is unsuccessful if the
following is true:

EINVAL The attr or detachstate parameters are not valid.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_attr_getdetachstate subroutine, pthread_attr_init subroutine,
pthread_create subroutine.

Creating Threads.

Threads Library Quick Reference.

pthread_attr_destroy Subroutine

Purpose

Deletes a thread attributes object.

Library

Threads Library (libpthreads.a)

Syntax

#include <pthread.h>

int pthread_attr_destroy (attr)
*pthread_attr_t *attr;*

Description

The pthread_attr_destroy subroutine destroys the thread attributes object attr, reclaiming its storage space. It has no effect on the threads previously created with that object.

Note: The pthread.h header file must be the first included file of each source file using the threads library.

Parameters

attr	Specifies the thread attributes object to delete.

Return Values

Upon successful completion, 0 is returned. Otherwise, an error code is returned.

Error Codes

The pthread_attr_destroy subroutine is unsuccessful if the following is true:

EINVAL	The attr parameter is not valid.

Implementation Specifics

This subroutine is part of the Base Operating System (BOS) Runtime.

Related Information

The pthread_attr_init subroutine, pthread_create subroutine.