## NAME

pthread_join – wait for termination of another thread

## SYNOPSIS

**#include <pthread.h>**

**int pthread_join(pthread_t** *th*, **void *****thread_return***);**

## DESCRIPTION

**pthread_join** suspends the execution of the calling thread until the thread identified by *th* terminates, either by calling **pthread_exit**(3) or by being cancelled.

If *thread_return* is not **NULL**, the return value of *th* is stored in the location pointed to by *thread_return*. The return value of *th* is either the argument it gave to **pthread_exit**(3), or **PTHREAD_CANCELED** if *th* was cancelled.

The joined thread **th** must be in the joinable state: it must not have been detached using **pthread_detach**(3) or the **PTHREAD_CREATE_DETACHED** attribute to **pthread_create**(3).

When a joinable thread terminates, its memory resources (thread descriptor and stack) are not deallocated until another thread performs **pthread_join** on it. Therefore, **pthread_join** must be called once for each joinable thread created to avoid memory leaks.

At most one thread can wait for the termination of a given thread. Calling **pthread_join** on a thread *th* on which another thread is already waiting for termination returns an error.

## CANCELLATION

**pthread_join** is a cancellation point. If a thread is canceled while suspended in **pthread_join**, the thread execution resumes immediately and the cancellation is executed without waiting for the *th* thread to terminate. If cancellation occurs during **pthread_join**, the *th* thread remains not joined.

## RETURN VALUE

On success, the return value of *th* is stored in the location pointed to by *thread_return*, and 0 is returned. On error, a non-zero error code is returned.

## ERRORS

**ESRCH**
: No thread could be found corresponding to that specified by *th*.

**EINVAL**
: The *th* thread has been detached.

**EINVAL**
: Another thread is already waiting on termination of *th*.

**EDEADLK**
: The *th* argument refers to the calling thread.

## AUTHOR

Xavier Leroy <Xavier.Leroy@inria.fr>

## SEE ALSO

**pthread_exit**(3), **pthread_detach**(3), **pthread_create**(3), **pthread_attr_setdetachstate**(3), **pthread_cleanup_push**(3), **pthread_key_create**(3).