NAME

pthread_exit, thr_exit – thread termination

SYNOPSIS

POSIX

cc [*flag* ...] *file* ... **–lpthread** [*library* ...]

#include <pthread.h>
void pthread_exit(void **status***);**

Solaris

cc [*flag* ...] *file* ... **–lthread** [*library* ...]

#include <thread.h>
void thr_exit(void **status);*

MT-LEVEL

MT-Safe

DESCRIPTION

pthread_exit( ) and **thr_exit( )** terminates the calling threads, similar to how **exit**(3C) terminates calling processes. If the calling thread is not detached, then the thread's ID and the exit status specified by *status* are retained. The value *status* is then made available to any successful join with the terminating thread (see **pthread_join**(3T)); otherwise, *status* is disregarded allowing the thread's ID to be reclaimed immediately.

Upon thread termination, all thread-specific data bindings are released (see **pthread_key_create**(3T)), and its cancellation routines are called, but application visible process resources, including, but not limited to, mutexes and file descriptors are not released.

The cleanup handlers are called before the thread-specific data bindings are released (see **pthread_cancel**(3T)). Any cancellation cleanup handlers that have been pushed and not yet popped will be popped in reverse order of when they were pushed and then executed. If the thread still has any thread-specific data after all cancellation cleanup handlers have been executed, appropriate destructor functions will be called in an unspecified order. If any thread, including the **main( )**thread, calls **pthread_exit( )**, only that thread will exit.

If **main( )** returns or exits (either implicitly or explicitly), or any thread explicitly calls **exit( )**, the entire process will exit.

If any thread (except the **main( )** thread) implicitly or explicitly returns, the result is the same as if the thread called **pthread_exit( )** and it will return the value of *status* as the exit code.

The process will terminate with an exit status of **0** after the last thread has terminated (including the **main( )** thread). This action is the same as if the application had called **exit( )** with a zero argument at any time.

RETURN VALUES

pthread_exit( ) or **thr_exit( )** does not return to its caller.

SEE ALSO

exit(3C), **pthread_cancel**(3T), **pthread_create**(3T), **pthread_join**(3T), **pthread_key_create**(3T), **pthread_cancel**(3T).

NOTES

Although only POSIX implements cancellation, cancellation can be used with Solaris threads, due to their interoperability.

Do not call **pthread_exit( )** from a cancellation cleanup handler or destructor function that will be invoked as a result of either an implicit or explicit call to **pthread_exit( )**.

status should not reference any variables local to the calling thread.