NAME

_signotifywait, _lwp_sigredirect – deliver process signals to specific LWPs

SYNOPSIS

#include <sys/lwp.h>

int _signotifywait(void);

int _lwp_sigredirect(lwpid_t *target_lwp*, **int** *signo***);**

DESCRIPTION

In a multi-threaded process, signals that are generated for a process are delivered to one of the threads that does not have that signal masked. If all of the application threads are masking that signal, its delivery waits until one of them unmasks it.

The disposition of the each thread's signal mask is unknown to the kernel when it generates signals for the process. The **_signotifywait( )** and **_lwp_sigredirect( )** functions provide a mechanism to direct instances of signals generated for the process to application-specified LWPs. Each process has a set of signals pending for the process, and for each LWP there is a set of signals pending for that LWP. If no signals are pending, these sets are empty.

There is also a process-wide signal set, termed the *notification* set, manipulated by these functions. A signal generated for the process where the signal number is not in the notification set is called an *unnotified* signal.

In a multi-threaded program there is an **aslwp,** a special LWP endowed with powers to handle signals that are generated for a process. The **_signotifywait( )** function is used to await signals generated for the process, and should be called only from the **aslwp.** In general, these functions are not to be called from the application-level.

If there is a pending unnotified signal when **_signotifywait( )** is called, that signal is selected and the call returns immediately. If there is not a signal pending, the call suspends the calling LWP until the generation of an unnotified signal; that signal then is selected and the function returns. In both cases, the selected signal number is set in the notification set and returned as the value of **_signotifywait( )**. The signal remains pending for the process, and any associated **siginfo**(5) information remains queued at the process.

The **_lwp_sigredirect( )** function requests that a signal pending for the process be delivered to the LWP specified by *target_lwp*. It is an error if *signo* is not currently in the notification set of the process. The signal specified by *signo* is removed from pending for the process and is made pending for the *target_lwp*. If there is an associated *siginfo* information structure queued at the process, that *siginfo* is queued to the *target_lwp*.

Whenever a signal is cleared from the set of signals pending for the process, the corresponding signal is cleared from the notification set. After a successful call to **_lwp_sigredirect( )**, the signal *signo* is cleared from the notification set and from the set of signals pending for the process. If another instance of *signo* is queued for the process, the signal number is again set in the process pending mask, and if another LWP is blocked in a call to **_signotifywait( )**, its wait for an unnotified signal will be satisfied. The effects described in this paragraph also apply when the signal *signo* is returned by a call to **sigtimedwait( )** and *signo* was not pending for the calling LWP.

RETURN VALUES

The function **_signotifywait( )** returns the signal number of the pending but hitherto unnotified signal. The function **_lwp_sigredirect( )** returns zero when successful; otherwise, a non-zero value indicates an error.

ERRORS

No error conditions are specified for **_signotifywait()**.

If the following conditions occurs, **_lwp_sigredirect( )** fails and return the corresponding value:

EINVAL The signal *signo* was not pending for the process, or *signo* was not in the notification set.

ESRCH The *target_lwp* cannot be found in the current process.

SEE ALSO

_lwp_create(2), **_lwp_kill**(2), **sigtimedwait**(3R), **siginfo**(5), **signal**(5)

NOTES

This mechanism for delivering signals to multi-threaded processes is subject to change in future versions of Solaris. Any process with explicit knowledge of this mechanism may not be compatible from release to release.