NAME

_lwp_cond_wait, _lwp_cond_timedwait – wait on a condition variable

SYNOPSIS

#include <sys/lwp.h>

int _lwp_cond_wait(lwp_cond_t **cvp*, **lwp_mutex_t** **mp*);

int _lwp_cond_timedwait(lwp_cond_t **cvp*, **lwp_mutex_t** **mp,* **timestruc_t** **abstime*);

DESCRIPTION

These functions are used to wait for the occurrence of a condition represented by an LWP condition variable. LWP condition variables must be initialized to zero before use.

_lwp_cond_wait( ) atomically releases the LWP mutex pointed to by *mp* and causes the calling LWP to block on the LWP condition variable pointed to by *cvp*. The blocked LWP may be awakened by **_lwp_cond_signal**(2), **_lwp_cond_broadcast**(2), or when interrupted by delivery of a signal. Any change in value of a condition associated with the condition variable cannot be inferred by the return of **_lwp_cond_wait( )** and any such condition must be re-evaluated.

_lwp_cond_timedwait( ) is similar to **_lwp_cond_wait( )**, except that the calling LWP will not block past the time of day specified by *abstime*. If the time of day becomes greater than *abstime* then **_lwp_cond_timedwait( )** returns with the error code **ETIME .**

_lwp_cond_wait( ), and**_lwp_cond_timedwait( )** always return with the mutex locked and owned by the calling lightweight process.

RETURN VALUES

Zero is returned when successful. A non-zero value indicates an error.

ERRORS

If any of the following conditions are detected, **_lwp_cond_wait( )**, and **_lwp_cond_timedwait( )** fail and return the corresponding value:

EINVAL *cvp* points to an invalid LWP condition variable or *mp* points to an invalid LWP mutex.

EFAULT *mp*, *cvp*, or *abstime* point to an illegal address.

If any of the following conditions occur, **_lwp_cond_wait( )**, and **_lwp_cond_timedwait( )** fail and return the corresponding value:

EINTR The call was interrupted by a signal or **fork**(2).

If any of the following conditions occur, **_lwp_cond_timedwait( )** fails and returns the corresponding value:

ETIME The time specified in *abstime* has passed.

EXAMPLES

_lwp_cond_wait() is normally used in a loop testing some condition, as follows:

```
lwp_mutex_t m;
lwp_cond_t cv;
int cond;

(void) _lwp_mutex_lock(&m);
while (cond == FALSE) {
        (void) _lwp_cond_wait(&cv, &m);
}
(void) _lwp_mutex_unlock(&m);
```

_lwp_cond_timedwait() is also normally used in a loop testing some condition. It uses an absolute timeout value as follows:

```
timestruc_t to;
lwp_mutex_t m;
lwp_cond_t cv;
int cond, err;

(void) _lwp_mutex_lock(&m);
to.tv_sec = time(NULL) + TIMEOUT;
to.tv_nsec = 0;
while (cond == FALSE) {
        err = _lwp_cond_timedwait(&cv, &m, &to);
        if (err == ETIME) {
                /* timeout, do something */
                break;
        }
}
(void) _lwp_mutex_unlock(&m);
```

This sets a bound on the total wait time even though the **_lwp_cond_timedwait( )** may return several times due to the condition being signalled or the wait being interrupted.

SEE ALSO

_lwp_cond_broadcast(2), **_lwp_cond_signal**(2), **_lwp_kill**(2), **_lwp_mutex_lock**(2), **fork**(2), **kill**(2)