NAME

condvar, **cv_init**, **cv_destroy**, **cv_wait**, **cv_wait_sig**, **cv_wait_unlock**, **cv_timedwait**, **cv_timedwait_sbt**, **cv_timedwait_sig**, **cv_timedwait_sig_sbt**, **cv_signal**, **cv_broadcast**, **cv_broadcastpri**, **cv_wmesg** - kernel condition variable

SYNOPSIS

#include <sys/param.h>
#include <sys/proc.h>
#include <sys/condvar.h>

void
cv_init(*struct cv *cvp, const char *desc*);

void
cv_destroy(*struct cv *cvp*);

void
cv_wait(*struct cv *cvp, lock*);

int
cv_wait_sig(*struct cv *cvp, lock*);

void
cv_wait_unlock(*struct cv *cvp, lock*);

int
cv_timedwait(*struct cv *cvp, lock, int timo*);

int
cv_timedwait_sbt(*struct cv *cvp, lock, sbintime_t sbt, sbintime_t pr, int flags*);

int
cv_timedwait_sig(*struct cv *cvp, lock, int timo*);

int
cv_timedwait_sig_sbt(*struct cv *cvp, lock, sbintime_t sbt, sbintime_t pr, int flags*);

void
cv_signal(*struct cv *cvp*);

void
cv_broadcast(*struct cv *cvp*);

void
cv_broadcastpri(*struct cv *cvp*, *int pri*);

*const char **
cv_wmesg(*struct cv *cvp*);

DESCRIPTION

Condition variables are used in conjunction with mutexes to wait for conditions to occur. Condition variables are created with **cv_init**(), where *cvp* is a pointer to space for a *struct cv*, and *desc* is a pointer to a null-terminated character string that describes the condition variable. Condition variables are destroyed with **cv_destroy**(). Threads wait on condition variables by calling **cv_wait**(), **cv_wait_sig**(), **cv_wait_unlock**(), **cv_timedwait**(), or **cv_timedwait_sig**(). Threads unblock waiters by calling **cv_signal**() to unblock one waiter, or **cv_broadcast**() or **cv_broadcastpri**() to unblock all waiters. In addition to waking waiters, **cv_broadcastpri**() ensures that all of the waiters have a priority of at least *pri* by raising the priority of any threads that do not. **cv_wmesg**() returns the description string of *cvp*, as set by the initial call to **cv_init**().

The *lock* argument is a pointer to either a mutex(9), rwlock(9), or sx(9) lock. A mutex(9) argument must be initialized with MTX_DEF and not MTX_SPIN. A thread must hold *lock* before calling **cv_wait**(), **cv_wait_sig**(), **cv_wait_unlock**(), **cv_timedwait**(), or **cv_timedwait_sig**(). When a thread waits on a condition, *lock* is atomically released before the thread is blocked, then reacquired before the function call returns. In addition, the thread will fully drop the *Giant* mutex (even if recursed) while the it is suspended and will reacquire the *Giant* mutex before the function returns. The **cv_wait_unlock**() function does not reacquire the lock before returning. Note that the *Giant* mutex may be specified as *lock*. However, *Giant* may not be used as *lock* for the **cv_wait_unlock**() function. All waiters must pass the same *lock* in conjunction with *cvp*.

When **cv_wait**(), **cv_wait_sig**(), **cv_wait_unlock**(), **cv_timedwait**(), and **cv_timedwait_sig**() unblock, their calling threads are made runnable. **cv_timedwait**() and **cv_timedwait_sig**() wait for at most *timo* / HZ seconds before being unblocked and returning EWOULDBLOCK; otherwise, they return 0. **cv_wait_sig**() and **cv_timedwait_sig**() return prematurely with a value of EINTR or ERESTART if a signal is caught, or 0 if signaled via **cv_signal**() or **cv_broadcast**().

cv_timedwait_sbt() and **cv_timedwait_sig_sbt**() functions take *sbt* argument instead of *timo*. It allows to specify relative or absolute unblock time with higher resolution in form of *sbintime_t*. The parameter *pr* allows to specify wanted absolute event precision. The parameter *flags* allows to pass additional **callout_reset_sbt**() flags.

RETURN VALUES

If successful, **cv_wait_sig**(), **cv_timedwait**(), and **cv_timedwait_sig**() return 0. Otherwise, a non-zero error code is returned.

cv_wmesg() returns the description string that was passed to **cv_init**().

ERRORS

cv_wait_sig() and **cv_timedwait_sig**() will fail if:

[EINTR] A signal was caught and the system call should be interrupted.

[ERESTART] A signal was caught and the system call should be restarted.

cv_timedwait() and **cv_timedwait_sig**() will fail if:

[EWOULDBLOCK] Timeout expired.

SEE ALSO

callout(9), locking(9), mtx_pool(9), mutex(9), rwlock(9), sema(9), sleep(9), sx(9)