

**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)`