NAME

qwait, qwait_sig – STREAMS wait routines

SYNOPSIS

#include <sys/stream.h>
#include <sys/ddi.h>

void qwait(queue_t **q***);**

int qwait_sig(queue_t **q***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

qp   Pointer to the queue that is being opened or closed.

DESCRIPTION

qwait( ) and **qwait_sig( )** are used to wait for a message to arrive to the **put**(9E) or **srv**(9E) procedures. **qwait( )** and **qwait_sig( )** can also be used to wait for **qbufcall**(9F) or **qtimeout**(9F) callback procedures to execute. These routines can be used in the **open**(9E) and **close**(9E) procedures in a STREAMS driver or module. **qwait( )** and **qwait_sig( )** atomically exit the inner and outer perimeters associated with the queue, and wait for a thread to leave the module's **put**(9E), **srv**(9E), or **qbufcall**(9F) / **qtimeout**(9F) callback procedures. Upon return they re-enter the inner and outer perimeters.

This can be viewed as there being an implicit wakeup when a thread leaves a **put**(9E) or **srv**(9E) procedure or after a **qtimeout**(9F) or **qbufcall**(9F) callback procedure has been run in the same perimeter.

qprocson(9F) must be called before calling **qwait( )** or **qwait_sig( )**.

qwait( ) is not interrupted by a signal, whereas **qwait_sig( )** is interrupted by a signal. **qwait_sig( )** normally returns non-zero, and returns zero when the waiting was interrupted by a signal.

qwait( ) and **qwait_sig( )** are similar to **cv_wait( )** and **cv_wait_sig( )** (see **condvar**(9F)), except that the mutex is replaced by the inner and outer perimeters and the signalling is implicit when a thread leaves the inner perimeter.

RETURN VALUES

0   For **qwait_sig( )**, indicates that the condition was not necessarily signaled and the function returned because a signal was pending.

CONTEXT

These functions can only be called from an **open**(9E) or **close**(9E) routine.

EXAMPLES

The open routine sends down a **T_INFO_REQ** message and waits for the **T_INFO_ACK**. The arrival of the **T_INFO_ACK** is recorded by resetting a flag in the unit structure (**WAIT_INFO_ACK**).

The example assumes that the module is **D_MTQPAIR** or **D_MTPERMOD**.

```
xxopen(qp, . . .)
        queue_t *qp;
{
        struct xxdata *xx;
```

```
        /* Allocate xxdata structure */
        qprocson(qp);
        /* Format T_INFO_ACK in mp */
        putnext(qp, mp);
        xx->xx_flags |= WAIT_INFO_ACK;
        while (xx->xx_flags & WAIT_INFO_ACK)
                qwait(qp);
        return (0);
}

xxrput(qp, mp)
        queue_t *qp;
        mblk_t *mp;
{
        struct xxdata *xx = (struct xxdata *)q->q_ptr;

        ...

        case T_INFO_ACK:
                if (xx->xx_flags & WAIT_INFO_ACK) {
                        /* Record information from info ack */
                        xx->xx_flags &= ~WAIT_INFO_ACK;
                        freemsg(mp);
                        return;
                }

        ...
}
```

SEE ALSO

close(9E), **open**(9E), **put**(9E), **srv**(9E) **condvar**(9F), **mt-streams**(9F), **qbufcall**(9F), **qprocson**(9F), **qtimeout**(9F)

STREAMS Programming Guide

Writing Device Drivers