NAME

putnextctl – send a control message to a queue

SYNOPSIS

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

int putnextctl(queue_t *q, int type);
```

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI).

ARGUMENTS

q Queue to which the message is to be sent.

type Message type (must be control, not data type).

DESCRIPTION

putnextctl( ) tests the *type* argument to make sure a data type has not been specified, and then attempts to allocate a message block. **putnextctl**( ) fails if *type* is **M_DATA**, **M_PROTO**, or **M_PCPROTO**, or if a message block cannot be allocated. If successful, **putnextctl( )** calls the **put**(9E) routine of the queue pointed to by *q* with the newly allocated and initialized messages.

A call to **putnextctl(***q*,*type***)** is an atomic equivalent of **putctl(***q***->q_next**,**type***)*. The STREAMS framework provides whatever mutual exclusion is necessary to insure that dereferencing *q* through its **q_next** field and then invoking **putctl**(9F) proceeds without interference from other threads.

putnextctl( ) should always be used in preference to **putctl**(9F)**.**

RETURN VALUES

On success, **1** is returned. If *type* is a data type, or if a message block cannot be allocated, **0** is returned.

CONTEXT

putnextctl( ) can be called from user or interrupt context.

EXAMPLE

The **send_ctl** routine is used to pass control messages downstream. **M_BREAK** messages are handled with **putnextctl**( ) (line 8). **putnextctl1**(9F) (line 13) is used for **M_DELAY** messages, so that *parm* can be used to specify the length of the delay. In either case, if a message block cannot be allocated a variable recording the number of allocation failures is incremented (lines 9, 14). If an invalid message type is detected, **cmn_err**(9F) panics the system (line 18).

```
 1 void
 2 send_ctl(queue_t *wrq, u_char type, u_char parm)
 3 {
 4                 extern int num_alloc_fail;
 5
 6                 switch (type) {
 7                 case M_BREAK:
 8                         if (!putnextctl(wrq, M_BREAK))
 9                                 num_alloc_fail++;
10                         break;
11
12                 case M_DELAY:
```

```
13                      if (!putnextctl1(wrq, M_DELAY, parm))
14                              num_alloc_fail++;
15                      break;
16
17              default:
18                      cmn_err(CE_PANIC, "send_ctl: bad message type passed");
19                      break;
20              }
21  }
```

SEE ALSO

put(9E), **cmn_err**(9F), **datamsg**(9F), **putctl**(9F), **putnextctl1**(9F)

Writing Device Drivers
STREAMS Programming Guide