NAME

putmsg, putpmsg – send a message on a stream

SYNOPSIS

#include <stropts.h>

int putmsg(int *fildes*, **const struct strbuf** **ctlptr*, **const struct strbuf** **dataptr*, **int** *flags***);**

int putpmsg(int *fildes*, **const struct strbuf** **ctlptr*, **const struct strbuf** **dataptr*, **int** *band*, **int** *flags***);**

DESCRIPTION

putmsg( ) creates a message from user-specified buffer(s) and sends the message to a STREAMS file. The message may contain either a data part, a control part, or both. The data and control parts to be sent are distinguished by placement in separate buffers, as described below. The semantics of each part is defined by the STREAMS module that receives the message.

The function **putpmsg( )** does the same thing as **putmsg( )**, but provides the user the ability to send messages in different priority bands. Except where noted, all information pertaining to **putmsg( )** also pertains to **putpmsg( )**.

fildes specifies a file descriptor referencing an open stream. *ctlptr* and *dataptr* each point to a **strbuf** structure, which contains the following members:

```
int maxlen;      /* not used here */
int len;         /* length of data */
void *buf;       /* ptr to buffer */
```

ctlptr points to the structure describing the control part, if any, to be included in the message. The **buf** field in the **strbuf** structure points to the buffer where the control information resides, and the **len** field indicates the number of bytes to be sent. The **maxlen** field is not used in **putmsg( )** (see **getmsg**(2)). In a similar manner, *dataptr* specifies the data, if any, to be included in the message. *flags* indicates what type of message should be sent and is described later.

To send the data part of a message, *dataptr* must not be **NULL** and the **len** field of *dataptr* must have a value of 0 or greater. To send the control part of a message, the corresponding values must be set for *ctlptr*. No data (control) part is sent if either *dataptr* (*ctlptr*) is **NULL** or the **len** field of *dataptr* (*ctlptr*) is negative.

For **putmsg( )**, if a control part is specified, and *flags* is set to **RS_HIPRI**, a high priority message is sent. If no control part is specified, and *flags* is set to **RS_HIPRI**, **putmsg( )** fails and sets **errno** to **EINVAL.** If *flags* is set to 0, a normal (non-priority) message is sent. If no control part and no data part are specified, and *flags* is set to 0, no message is sent, and 0 is returned.

The stream head guarantees that the control part of a message generated by **putmsg( )** is at least 64 bytes in length.

For **putpmsg( )**, the flags are different. *flags* is a bitmask with the following mutually-exclusive flags defined: **MSG_HIPRI** and **MSG_BAND**. If *flags* is set to 0, **putpmsg( )** fails and sets **errno** to **EINVAL.** If a control part is specified and *flags* is set to **MSG_HIPRI** and *band* is set to 0, a high-priority message is sent. If *flags* is set to **MSG_HIPRI** and either no control part is specified or *band* is set to a non-zero value, **putpmsg( )** fails and sets **errno** to **EINVAL.** If flags is set to **MSG_BAND**, then a message is sent in the priority band specified by *band*. If a control part and data part are not specified and *flags* is set to **MSG_BAND**, no message is sent and 0 is returned.

Normally, **putmsg( )** will block if the stream write queue is full due to internal flow control conditions. For high-priority messages, **putmsg( )** does not block on this condition. For other messages, **putmsg( )** does not block when the write queue is full and **O_NDELAY** or **O_NONBLOCK** is set. Instead, it fails and sets **errno** to **EAGAIN.**

putmsg( ) or **putpmsg( )** also blocks, unless prevented by lack of internal resources, waiting for the availability of message blocks in the stream, regardless of priority or whether **O_NDELAY** or **O_NONBLOCK** has been specified. No partial message is sent.

RETURN VALUES

Upon successful completion, a value of 0 is returned. Otherwise, a value of **-1** is returned and **errno** is set to indicate the error.

ERRORS

putmsg( ) fails if one or more of the following are true:

EAGAIN A non-priority message was specified, the **O_NDELAY** or **O_NONBLOCK** flag is set and the stream write queue is full due to internal flow control conditions.

EBADF *fildes* is not a valid file descriptor open for writing.

EFAULT *ctlptr* or *dataptr* points to an illegal address.

EINTR A signal was caught during the **putmsg( )** function.

EINVAL An undefined value was specified in *flags*, or *flags* is set to **RS_HIPRI** and no control part was supplied.

EINVAL The stream referenced by *fildes* is linked below a multiplexor.

EINVAL For **putpmsg( )**, if *flags* is set to **MSG_HIPRI** and *band* is nonzero.

ENOSR Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.

ENOSTR *fildes* is not associated with a stream.

ENXIO A hangup condition was generated downstream for the specified stream, or the other end of the pipe is closed.

ERANGE The size of the data part of the message does not fall within the range specified by the maximum and minimum packet sizes of the topmost stream module. This value is also returned if the control part of the message is larger than the maximum configured size of the control part of a message, or if the data part of a message is larger than the maximum configured size of the data part of a message.

putmsg( ) also fails if a STREAMS error message had been processed by the stream head before the call to **putmsg( )**. The error returned is the value contained in the STREAMS error message.

SEE ALSO

intro(2), **getmsg**(2), **poll**(2), **read**(2), **write**(2)

STREAMS Programming Guide