NAME

getmsg, getpmsg – get next message off a stream

SYNOPSIS

#include <stropts.h>

int getmsg(int *fildes*, **struct strbuf** **ctlptr*, **struct strbuf** **dataptr*, **int** **flagsp***);**

int getpmsg(int *fildes*, **struct strbuf** **ctlptr*, **struct strbuf** **dataptr*, **int** **bandp*, **int** **flagsp***);**

DESCRIPTION

getmsg( ) retrieves the contents of a message (see **intro**(2)) located at the stream head read queue from a STREAMS file, and places the contents into user specified buffer(s). The message must contain either a data part, a control part, or both. The data and control parts of the message are placed into separate buffers, as described below. The semantics of each part is defined by the STREAMS module that generated the message.

The function **getpmsg( )** does the same thing as **getmsg( )**, but provides finer control over the priority of the messages received. Except where noted, all information pertaining to **getmsg( )** also pertains to **getpmsg( )**.

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;     /* maximum buffer length */
int len;        /* length of data */
char *buf;      /* ptr to buffer */
```

buf points to a buffer in which the data or control information is to be placed, and **maxlen** indicates the maximum number of bytes this buffer can hold. On return, **len** contains the number of bytes of data or control information actually received, or 0 if there is a zero-length control or data part, or -1 if no data or control information is present in the message. *flagsp* should point to an integer that indicates the type of message the user is able to receive. This is described later.

ctlptr is used to hold the control part from the message and *dataptr* is used to hold the data part from the message. If *ctlptr* (or *dataptr*) is **NULL** or the **maxlen** field is –1, the control (or data) part of the message is not processed and is left on the stream head read queue. If *ctlptr* (or *dataptr*) is not **NULL** and there is no corresponding control (or data) part of the messages on the stream head read queue, **len** is set to –1. If the **maxlen** field is set to 0 and there is a zero-length control (or data) part, that zero-length part is removed from the read queue and **len** is set to 0. If the **maxlen** field is set to 0 and there are more than zero bytes of control (or data) information, that information is left on the read queue and **len** is set to 0. If the **maxlen** field in *ctlptr* or *dataptr* is less than, respectively, the control or data part of the message, **maxlen** bytes are retrieved. In this case, the remainder of the message is left on the stream head read queue and a non-zero return value is provided, as described below under RETURN VALUES.

By default, **getmsg( )** processes the first available message on the stream head read queue. However, a user may choose to retrieve only high priority messages by setting the integer pointed by *flagsp* to **RS_HIPRI**. In this case, **getmsg( )** processes the next message only if it is a high priority message.

If the integer pointed by *flagsp* is 0, **getmsg( )** retrieves any message available on the stream head read queue. In this case, on return, the integer pointed to by *flagsp* will be set to **RS_HIPRI** if a high priority message was retrieved, or 0 otherwise.

For **getpmsg( )**, the flags are different. *flagsp* points to a bitmask with the following mutually-exclusive flags defined: **MSG_HIPRI, MSG_BAND**, and **MSG_ANY**. Like **getmsg( )**, **getpmsg( )** processes the first available message on the stream head read queue. A user may choose to retrieve only high-priority messages by setting the integer pointed to by *flagsp* to **MSG_HIPRI** and the integer pointed to by *bandp* to 0. In this case, **getpmsg( )** will only process the next message if it is a high-priority message. In a similar manner, a user may choose to retrieve a message from a particular priority band by setting the integer pointed to by *flagsp* to **MSG_BAND** and the integer pointed to by *bandp* to the priority band of interest. In this case, **getpmsg( )** will only process the next message if it is in a priority band equal to, or greater than, the integer pointed to by *bandp*, or if it is a high-priority message. If a user just wants to get the first message off the queue, the integer pointed to by *flagsp* should be set to **MSG_ANY** and the integer pointed to by *bandp* should be set to 0. On return, if the message retrieved was a high-priority message, the integer pointed to by *flagsp* will be set to **MSG_HIPRI** and the integer pointed to by *bandp* will be set to 0. Otherwise, the integer pointed to by *flagsp* will be set to **MSG_BAND** and the integer pointed to by *bandp* will be set to the priority band of the message.

If **O_NDELAY** and **O_NONBLOCK** are clear, **getmsg( )** blocks until a message of the type specified by *flagsp* is available on the stream head read queue. If **O_NDELAY** or **O_NONBLOCK** has been set and a message of the specified type is not present on the read queue, **getmsg( )** fails and sets **errno** to **EAGAIN**.

If a hangup occurs on the stream from which messages are to be retrieved, **getmsg( )** continues to operate normally, as described above, until the stream head read queue is empty. Thereafter, it returns 0 in the **len** fields of *ctlptr* and *dataptr*.

RETURN VALUES

Upon successful completion, a non-negative value is returned. A value of 0 indicates that a full message was read successfully. A return value of **MORECTL** indicates that more control information is waiting for retrieval. A return value of **MOREDATA** indicates that more data are waiting for retrieval. A return value of **MORECTL | MOREDATA** indicates that both types of information remain. Subsequent **getmsg** calls retrieve the remainder of the message. However, if a message of higher priority has come in on the stream head read queue, the next call to **getmsg** will retrieve that higher priority message before retrieving the remainder of the previously received partial message.

ERRORS

getmsg( ) or **getpmsg( )** will fail if one or more of the following are true:

EAGAIN The **O_NDELAY** or **O_NONBLOCK** flag is set, and no messages are available.

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

EBADMSG Queued message to be read is not valid for **getmsg**.

EFAULT *ctlptr*, *dataptr*, *bandp*, or *flagsp* points to an illegal address.

EINTR A signal was caught during the **getmsg** function.

EINVAL An illegal value was specified in *flagsp*, or the stream referenced by *fildes* is linked under a multiplexor.

ENOSTR A stream is not associated with *fildes*.

getmsg can also fail if a STREAMS error message had been received at the stream head before the call to **getmsg**. The error returned is the value contained in the STREAMS error message.

SEE ALSO

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

STREAMS Programming Guide