Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'getmsg.3p' command

$ man getmsg.3p

GETMSG(3P) POSIX Programmer's Manual GETMSG(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

getmsg, getpmsg ? receive next message from a STREAMS file (STREAMS)

SYNOPSIS

```
#include <stropts.h>
int getmsg(int fildes, struct strbuf *restrict ctlptr,
    struct strbuf *restrict dataptr, int *restrict flagsp);
int getpmsg(int fildes, struct strbuf *restrict ctlptr,
    struct strbuf *restrict dataptr, int *restrict bandp,
    int *restrict flagsp);
```

DESCRIPTION

The getmsg() function shall retrieve the contents of a message located at the head of the STREAM head read queue associated with a STREAMS file and place the contents into one or more buffers. The message con? tains either a data part, a control part, or both. The data and control parts of the message shall be placed into separate buffers, as de? scribed below. The semantics of each part are defined by the originator of the message.

The getpmsg() function shall be equivalent to getmsg(), except that it provides finer control over the priority of the messages received. Ex? cept where noted, all requirements on getmsg() also pertain to getpmsg().

The fildes argument specifies a file descriptor referencing a STREAMS-based file.

The ctlptr and dataptr arguments each point to a strbuf structure, in which the buf member points to a buffer in which the data or control information is to be placed, and the maxlen member indicates the maxi? mum number of bytes this buffer can hold. On return, the len member shall contain the number of bytes of data or control information actu? ally received. The len member shall be set to 0 if there is a zero-length control or data part and len shall be set to -1 if no data or control information is present in the message.

When getmsg() is called, flagsp should point to an integer that indi? cates the type of message the process is able to receive. This is de? scribed further below.

The ctlptr argument is used to hold the control part of the message, and dataptr is used to hold the data part of the message. If ctlptr (or dataptr) is a null pointer or the maxlen member is -1, the control (or data) part of the message shall not be processed and shall be left on the STREAM head read queue, and if the ctlptr (or dataptr) is not a null pointer, len shall be set to -1. If the maxlen member is set to 0 and there is a zero-length control (or data) part, that zero-length part shall be removed from the read queue and len shall be set to 0. If the maxlen member is set to 0 and there are more than 0 bytes of con? trol (or data) information, that information shall be left on the read queue and len shall be set to 0. If the maxlen member in ctlptr (or dataptr) is less than the control (or data) part of the message, maxlen bytes shall be retrieved. In this case, the remainder of the message shall be left on the STREAM head read queue and a non-zero return value shall be provided.

By default, getmsg() shall process the first available message on the

STREAM head read queue. However, a process may choose to retrieve only high-priority messages by setting the integer pointed to by flagsp to RS_HIPRI. In this case, getmsg() shall only process the next message if it is a high-priority message. When the integer pointed to by flagsp is 0, any available message shall be retrieved. In this case, on re? turn, the integer pointed to by flagsp shall be set to RS_HIPRI if a high-priority message was retrieved, or 0 otherwise.

For getpmsg(), the flags are different. The flagsp argument points to a bitmask with the following mutually-exclusive flags defined: MSG_HIPRI, MSG_BAND, and MSG_ANY. Like getmsg(), getpmsg() shall process the first available message on the STREAM head read queue. A process 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() shall only process the next message if it is a high-priority message. In a similar manner, a process 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() shall 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-pri? ority message. If a process 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 shall be set to MSG_HIPRI and the integer pointed to by bandp shall be set to 0. Otherwise, the integer pointed to by flagsp shall be set to MSG_BAND and the integer pointed to by bandp shall be set to the priority band of the message.

If O_NONBLOCK is not set, getmsg() and getpmsg() shall block until a message of the type specified by flagsp is available at the front of the STREAM head read queue. If O_NONBLOCK is set and a message of the specified type is not present at the front of the read queue, getmsg() and getpmsg() shall fail and set errno to [EAGAIN].

If a hangup occurs on the STREAM from which messages are retrieved, getmsg() and getpmsg() shall continue to operate normally, as described above, until the STREAM head read queue is empty. Thereafter, they shall return 0 in the len members of ctlptr and dataptr.

RETURN VALUE

Upon successful completion, getmsg() and getpmsg() shall return a non-negative value. A value of 0 indicates that a full message was read successfully. A return value of MORECTL indicates that more control in? formation is waiting for retrieval. A return value of MOREDATA indi? cates that more data is waiting for retrieval. A return value of the bitwise-logical OR of MORECTL and MOREDATA indicates that both types of information remain. Subsequent getmsg() and getpmsg() calls shall re? trieve 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() or getpmsg() shall retrieve that higher-priority message be? fore retrieving the remainder of the previous message.

If the high priority control part of the message is consumed, the mes? sage shall be placed back on the queue as a normal message of band 0. Subsequent getmsg() and getpmsg() calls shall retrieve the remainder of the message. If, however, a priority message arrives or already exists on the STREAM head, the subsequent call to getmsg() or getpmsg() shall retrieve the higher-priority message before retrieving the remainder of the message that was put back.

Upon failure, getmsg() and getpmsg() shall return -1 and set errno to indicate the error.

ERRORS

The getmsg() and getpmsg() functions shall fail if:

EAGAIN The O_NONBLOCK flag is set and no messages are available.

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

EBADMSG

The queued message to be read is not valid for getmsg() or getpmsg() or a pending file descriptor is at the STREAM head.

EINTR A signal was caught during getmsg() or getpmsg().

EINVAL An illegal value was specified by flagsp, or the STREAM or mul?
tiplexer referenced by fildes is linked (directly or indirectly)
downstream from a multiplexer.

ENOSTR A STREAM is not associated with fildes.

In addition, getmsg() and getpmsg() shall fail if the STREAM head had processed an asynchronous error before the call. In this case, the value of errno does not reflect the result of getmsg() or getpmsg() but reflects the prior error.

The following sections are informative.

EXAMPLES

Getting Any Message

In the following example, the value of fd is assumed to refer to an open STREAMS file. The call to getmsg() retrieves any available message on the associated STREAM-head read queue, returning control and data information to the buffers pointed to by ctrlbuf and databuf, respec?
tively.

```
#include <stropts.h>
...
int fd;
char ctrlbuf[128];
char databuf[512];
struct strbuf ctrl;
struct strbuf data;
int flags = 0;
int ret;
ctrl.buf = ctrlbuf;
ctrl.maxlen = sizeof(ctrlbuf);
data.buf = databuf;
data.maxlen = sizeof(databuf);
ret = getmsg (fd, &ctrl, &data, &flags);
```

Getting the First Message off the Queue

In the following example, the call to getpmsg() retrieves the first

available message on the associated STREAM-head read queue.

```
#include <stropts.h>
...
int fd;
char ctrlbuf[128];
char databuf[512];
struct strbuf ctrl;
struct strbuf data;
int band = 0;
int flags = MSG_ANY;
int ret;
ctrl.buf = ctrlbuf;
ctrl.maxlen = sizeof(ctrlbuf);
data.buf = databuf;
data.maxlen = sizeof(databuf);
ret = getpmsg (fd, &ctrl, &data, &band, &flags);
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

The getmsg() and getpmsg() functions may be removed in a future ver?sion.

SEE ALSO

Section 2.6, STREAMS, poll(), putmsg(), read(), write()

The Base Definitions volume of POSIX.1?2017, <stropts.h>

The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .