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 'putmsg.3p' command

$ man putmsg.3p

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

putmsg, putpmsg ? send a message on a STREAM (STREAMS)

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

The putmsg() function shall create a message from a process buffer(s) and send the message to a STREAMS file. The message may contain either a data part, a control part, or both. The data and control parts are distinguished by placement in separate buffers, as described below. The semantics of each part are defined by the STREAMS module that receives the message.

The putpmsg() function is equivalent to putmsg(), except that the process can send messages in different priority bands. Except where

noted, all requirements on putmsg() also pertain to putpmsg().

The fildes argument specifies a file descriptor referencing an open STREAM. The ctlptr and dataptr arguments each point to a strbuf struc? ture.

The ctlptr argument points to the structure describing the control part, if any, to be included in the message. The buf member in the str? buf structure points to the buffer where the control information re? sides, and the len member indicates the number of bytes to be sent. The maxlen member is not used by putmsg(). In a similar manner, the argu? ment dataptr specifies the data, if any, to be included in the message.

The flags argument indicates what type of message should be sent and is described further below.

To send the data part of a message, the application shall ensure that dataptr is not a null pointer and the len member of dataptr is 0 or greater. To send the control part of a message, the application shall ensure that the corresponding values are set for ctlptr. No data (con? trol) part shall be sent if either dataptr(ctlptr) is a null pointer or the len member of dataptr(ctlptr) is set to -1.

For putmsg(), if a control part is specified and flags is set to RS_HIPRI, a high priority message shall be sent. If no control part is specified, and flags is set to RS_HIPRI, putmsg() shall fail and set errno to [EINVAL]. If flags is set to 0, a normal message (priority band equal to 0) shall be sent. If a control part and data part are not specified and flags is set to 0, no message shall be sent and 0 shall be returned.

For putpmsg(), the flags are different. The flags argument is a bitmask with the following mutually-exclusive flags defined: MSG_HIPRI and MSG_BAND. If flags is set to 0, putpmsg() shall fail and set 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 shall be 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() shall fail and set errno to [EIN? VAL]. If flags is set to MSG_BAND, then a message shall be 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 shall be sent and 0 shall be returned.

The putmsg() function shall block if the STREAM write queue is full due to internal flow control conditions, with the following exceptions:

* For high-priority messages, putmsg() shall not block on this condi? tion and continues processing the message.
* For other messages, putmsg() shall not block but shall fail when the write queue is full and O_NONBLOCK is set.

The putmsg() function shall also block, unless prevented by lack of in? ternal resources, while waiting for the availability of message blocks in the STREAM, regardless of priority or whether O_NONBLOCK has been specified. No partial message shall be sent.

RETURN VALUE

Upon successful completion, putmsg() and putpmsg() shall return 0; oth? erwise, they shall return -1 and set errno to indicate the error.

ERRORS

The putmsg() and putpmsg() functions shall fail if:

EAGAIN A non-priority message was specified, the O_NONBLOCK flag is set, and the STREAM write queue is full due to internal flow control conditions; or buffers could not be allocated for the message that was to be created.

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

EINTR A signal was caught during putmsg().

EINVAL An undefined value is specified in flags, or flags is set to RS_HIPRI or MSG_HIPRI and no control part is supplied, or the STREAM or multiplexer referenced by fildes is linked (directly or indirectly) downstream from a multiplexer, or flags is set to MSG_HIPRI and band is non-zero (for putpmsg() only).

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

ENOSTR A STREAM is not associated with fildes.

ENXIO A hangup condition was generated downstream for the specified

STREAM.

EPIPE or EIO

The fildes argument refers to a STREAMS-based pipe and the other end of the pipe is closed. A SIGPIPE signal is generated for the calling thread.

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 config? ured 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.

In addition, putmsg() and putpmsg() 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 putmsg() or putpmsg(), but reflects the prior error.

The following sections are informative.

EXAMPLES

Sending a High-Priority Message

The value of fd is assumed to refer to an open STREAMS file. This call to putmsg() does the following:

1. Creates a high-priority message with a control part and a data part, using the buffers pointed to by ctrlbuf and databuf, respec? tively.
2. Sends the message to the STREAMS file identified by fd.

```
#include <stropts.h>
#include <string.h>
...
int fd;
char *ctrlbuf = "This is the control part";
char *databuf = "This is the data part";
struct strbuf ctrl;
struct strbuf data;
```

```
int ret;
ctrl.buf = ctrlbuf;
ctrl.len = strlen(ctrlbuf);
data.buf = databuf;
data.len = strlen(databuf);
ret = putmsg(fd, &ctrl, &data, MSG_HIPRI);
```

Using putpmsg()

This example has the same effect as the previous example. In this exam? ple, however, the putpmsg() function creates and sends the message to the STREAMS file.

```
#include <stropts.h>
#include <string.h>
...
int fd;
char *ctrlbuf = "This is the control part";
char *databuf = "This is the data part";
struct strbuf ctrl;
struct strbuf data;
int ret;
ctrl.buf = ctrlbuf;
ctrl.len = strlen(ctrlbuf);
data.buf = databuf;
data.len = strlen(databuf);
ret = putpmsg(fd, &ctrl, &data, 0, MSG_HIPRI);
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

The putmsg() and putpmsg() functions may be removed in a future ver? sion.

SEE ALSO

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

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

COPYRIGHT