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 'stropts.h.0p' command

$ man stropts.h.0p

stropts.h(0P) POSIX Programmer's Manual stropts.h(0P)

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

stropts.h ? STREAMS interface (STREAMS)

SYNOPSIS

#include <stropts.h>

DESCRIPTION

The <stropts.h> header shall define the bandinfo structure, which shall include at least the following members:

int bi_flag Flushing type.

unsigned char bi_pri Priority band.

The <stropts.h> header shall define the strpeek structure, which shall include at least the following members:

struct strbuf ctlbuf The control portion of the message.

struct strbuf databuf The data portion of the message.

t_uscalar_t flags RS_HIPRI or 0.

The <stropts.h> header shall define the strbuf structure, which shall include at least the following members:

char *buf Pointer to buffer.

int len Length of data.

int maxlen Maximum buffer length.

The <stropts.h> header shall define the strfdinsert structure, which shall include at least the following members:

struct strbuf ctlbuf The control portion of the message.

struct strbuf databuf The data portion of the message.

int fildes File descriptor of the other STREAM.

t_uscalar_t flags RS_HIPRI or 0.

int offset Relative location of the stored value.

The <stropts.h> header shall define the strioctl structure, which shall include at least the following members:

int ic_cmd ioctl() command.

char *ic_dp Pointer to buffer.

int ic_len Length of data.

int ic_timout Timeout for response.

The <stropts.h> header shall define the strrecvfd structure, which shall include at least the following members:

int fd Received file descriptor.

gid_t gid GID of sender.

uid_t uid UID of sender.

The <stropts.h> header shall define the uid_t and gid_t types through typedef, as described in <sys/types.h>.

The <stropts.h> header shall define the t_scalar_t and t_uscalar_t types, respectively, as signed and unsigned opaque types of equal length of at least 32 bits.

The <stropts.h> header shall define the str_list structure, which shall include at least the following members:

struct str_mlist *sl_modlist STREAMS module names.

int sl_nmods Number of STREAMS module names.

The <stropts.h> header shall define the str_mlist structure, which shall include at least the following member:

char l_name[FMNAMESZ+1] A STREAMS module name.

The <stropts.h> header shall define at least the following symbolic

constants for use as the request argument to ioctl():

I_ATMARK Is the top message ``marked''?

I_CANPUT Is a band writable?

I_CKBAND See if any messages exist in a band.

I_FDINSERT Send implementation-defined information about another STREAM.

I_FIND Look for a STREAMS module.

I_FLUSH Flush a STREAM.

I_FLUSHBAND Flush one band of a STREAM.

I_GETBAND Get the band of the top message on a STREAM.

I_GETCLTIME Get close time delay.

I_GETSIG Retrieve current notification signals.

I_GRDOPT Get the read mode.

I_GWROPT Get the write mode.

I_LINK Connect two STREAMs.

I_LIST Get all the module names on a STREAM.

I_LOOK Get the top module name.

I_NREAD Size the top message.

I_PEEK Peek at the top message on a STREAM.

I_PLINK Persistently connect two STREAMs.

I_POP Pop a STREAMS module.

I_PUNLINK Dismantle a persistent STREAMS link.

I_PUSH Push a STREAMS module.

I_RECVFD Get a file descriptor sent via I_SENDFD.

I_SENDFD Pass a file descriptor through a STREAMS pipe.

I_SETCLTIME Set close time delay.

I_SETSIG Ask for notification signals.

I_SRDOPT Set the read mode.

I_STR Send a STREAMS ioctl().

I_SWROPT Set the write mode.

I_UNLINK Disconnect two STREAMs.

The <stropts.h> header shall define at least the following symbolic constant for use with I_LOOK:

FMNAMESZ The minimum size in bytes of the buffer referred to by the arg argument.

The <stropts.h> header shall define at least the following symbolic constants for use with I_FLUSH:

FLUSHR Flush read queues.

FLUSHRW Flush read and write queues.

FLUSHW Flush write queues.

The <stropts.h> header shall define at least the following symbolic constants for use with I_SETSIG:

S_BANDURG When used in conjunction with S_RDBAND, SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the STREAM head read queue.

S_ERROR Notification of an error condition reaches the STREAM head.

S_HANGUP Notification of a hangup reaches the STREAM head.

S_HIPRI A high-priority message is present on a STREAM head read queue.

S_INPUT A message, other than a high-priority message, has arrived at the head of a STREAM head read queue.

S_MSG A STREAMS signal message that contains the SIGPOLL signal reaches the front of the STREAM head read queue.

S_OUTPUT The write queue for normal data (priority band 0) just be? low the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) normal data downstream.

S_RDBAND A message with a non-zero priority band has arrived at the head of a STREAM head read queue.

S_RDNORM A normal (priority band set to 0) message has arrived at the head of a STREAM head read queue.

S_WRBAND The write queue for a non-zero priority band just below the STREAM head is no longer full.

S_WRNORM Equivalent to S_OUTPUT.

The <stropts.h> header shall define at least the following symbolic constant for use with I_PEEK:

RS_HIPRI Only look for high-priority messages.

The <stropts.h> header shall define at least the following symbolic constants for use with I_SRDOPT:

RMSGD Message-discard mode.

RMSGN Message-non-discard mode.

RNORM Byte-STREAM mode, the default.

RPROTDAT Deliver the control part of a message as data when a process issues a read().

RPROTDIS Discard the control part of a message, delivering any data part, when a process issues a read().

RPROTNORM Fail read() with [EBADMSG] if a message containing a control part is at the front of the STREAM head read queue.

The <stropts.h> header shall define at least the following symbolic constant for use with I_SWOPT:

SNDZERO Send a zero-length message downstream when a write() of 0 bytes occurs.

The <stropts.h> header shall define at least the following symbolic constants for use with I_ATMARK:

ANYMARK Check if the message is marked.

LASTMARK Check if the message is the last one marked on the queue.

The <stropts.h> header shall define at least the following symbolic constant for use with I_UNLINK:

MUXID_ALL Unlink all STREAMs linked to the STREAM associated with fildes.

The <stropts.h> header shall define the following symbolic constants for getmsg(), getpmsg(), putmsg(), and putpmsg():

MORECTL More control information is left in message.

MOREDATA More data is left in message.

MSG_ANY Receive any message.

MSG_BAND Receive message from specified band.

MSG_HIPRI Send/receive high-priority message.

The <stropts.h> header may make visible all of the symbols from <unistd.h>.

The <stropts.h> header may also define macros for message types using names that start with M_.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int    fattach(int, const char *);
int    fdetach(const char *);
int    getmsg(int, struct strbuf *restrict, struct strbuf *restrict,
           int *restrict);
int    getpmsg(int, struct strbuf *restrict, struct strbuf *restrict,
           int *restrict, int *restrict);
int    ioctl(int, int, ...);
int    isastream(int);
int    putmsg(int, const struct strbuf *, const struct strbuf *, int);
int    putpmsg(int, const struct strbuf *, const struct strbuf *, int,
           int);
```

The following sections are informative.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<sys_types.h>, <unistd.h>

The System Interfaces volume of POSIX.1?2017, close(), fattach(), fc?ntl(), fdetach(), getmsg(), ioctl(), isastream(), open(), pipe(), read(), poll(), putmsg(), signal(), write()

COPYRIGHT

event of any discrepancy between this version and the original IEEE and 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 .