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 'ioctl.3p' command

$ man ioctl.3p

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

ioctl ? control a STREAMS device (STREAMS)

SYNOPSIS

```
#include <stropts.h>
int ioctl(int fildes, int request, ... /* arg */);
```

DESCRIPTION

The ioctl() function shall perform a variety of control functions on STREAMS devices. For non-STREAMS devices, the functions performed by this call are unspecified. The request argument and an optional third argument (with varying type) shall be passed to and interpreted by the appropriate part of the STREAM associated with fildes.

The fildes argument is an open file descriptor that refers to a device.

The request argument selects the control function to be performed and shall depend on the STREAMS device being addressed.

The arg argument represents additional information that is needed by this specific STREAMS device to perform the requested function. The type of arg depends upon the particular control request, but it shall

be either an integer or a pointer to a device-specific data structure.

The ioctl() commands applicable to STREAMS, their arguments, and error conditions that apply to each individual command are described below.

The following ioctl() commands, with error values indicated, are appli? cable to all STREAMS files:

I_PUSH Pushes the module whose name is pointed to by arg onto the top of the current STREAM, just below the STREAM head. It then calls the open() function of the newly-pushed module.

The ioctl() function with the I_PUSH command shall fail if:

EINVAL Invalid module name.

ENXIO Open function of new module failed.

ENXIO Hangup received on fildes.

I_POP Removes the module just below the STREAM head of the STREAM pointed to by fildes. The arg argument should be 0 in an I_POP request.

The ioctl() function with the I_POP command shall fail if:

EINVAL No module present in the STREAM.

ENXIO Hangup received on fildes.

I_LOOK Retrieves the name of the module just below the STREAM head of the STREAM pointed to by fildes, and places it in a character string pointed to by arg. The buffer pointed to by arg should be at least FMNAMESZ+1 bytes long, where FM? NAMESZ is defined in <stropts.h>.

The ioctl() function with the I_LOOK command shall fail if:

EINVAL No module present in the STREAM.

I_FLUSH Flushes read and/or write queues, depending on the value of arg. Valid arg values are:

FLUSHR Flush all read queues.

FLUSHW Flush all write queues.

FLUSHRW Flush all read and all write queues.

The ioctl() function with the I_FLUSH command shall fail if:

EINVAL Invalid arg value.

EAGAIN or ENOSR

Unable to allocate buffers for flush message.

ENXIO Hangup received on fildes.

I_FLUSHBAND Flushes a particular band of messages. The arg argument points to a bandinfo structure. The bi_flag member may be one of FLUSHR, FLUSHW, or FLUSHRW as described above. The bi_pri member determines the priority band to be flushed.

I_SETSIG Requests that the STREAMS implementation send the SIGPOLL signal to the calling process when a particular event has occurred on the STREAM associated with fildes. I_SETSIG supports an asynchronous processing capability in STREAMS. The value of arg is a bitmask that specifies the events for which the process should be signaled. It is the bitwise-in? clusive OR of any combination of the following constants:

S_RDNORM A normal (priority band set to 0) message has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.

S_RDBAND A message with a non-zero priority band has ar? rived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.

S_INPUT A message, other than a high-priority message, has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.

S_HIPRI A high-priority message is present on a STREAM head read queue. A signal shall be generated even if the message is of zero length.

S_OUTPUT The write queue for normal data (priority band 0) just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) nor?

mal data downstream.

S_WRNORM Equivalent to S_OUTPUT.

S_WRBAND The write queue for a non-zero priority band just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) priority data downstream.

S_MSG A STREAMS signal message that contains the SIG? POLL signal has reached the front of the STREAM head read queue.

S_ERROR Notification of an error condition has reached the STREAM head.

S_HANGUP Notification of a hangup has reached the STREAM head.

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.

If arg is 0, the calling process shall be unregistered and shall not receive further SIGPOLL signals for the stream associated with fildes.

Processes that wish to receive SIGPOLL signals shall ensure that they explicitly register to receive them using I_SET? SIG. If several processes register to receive this signal for the same event on the same STREAM, each process shall be signaled when the event occurs.

The ioctl() function with the I_SETSIG command shall fail if:

EINVAL The value of arg is invalid.

EINVAL The value of arg is 0 and the calling process is not registered to receive the SIGPOLL signal.

EAGAIN There were insufficient resources to store the sig? nal request.

I_GETSIG Returns the events for which the calling process is cur?
rently registered to be sent a SIGPOLL signal. The events
are returned as a bitmask in an int pointed to by arg,
where the events are those specified in the description of
I_SETSIG above.
The ioctl() function with the I_GETSIG command shall fail
if:
EINVAL Process is not registered to receive the SIGPOLL
signal.

I_FIND Compares the names of all modules currently present in the
STREAM to the name pointed to by arg, and returns 1 if the
named module is present in the STREAM, or returns 0 if the
named module is not present.
The ioctl() function with the I_FIND command shall fail if:
EINVAL arg does not contain a valid module name.

I_PEEK Retrieves the information in the first message on the
STREAM head read queue without taking the message off the
queue. It is analogous to getmsg() except that this command
does not remove the message from the queue. The arg argu?
ment points to a strpeek structure.
The application shall ensure that the maxlen member in the
ctlbuf and databuf strbuf structures is set to the number
of bytes of control information and/or data information,
respectively, to retrieve. The flags member may be marked
RS_HIPRI or 0, as described by getmsg(). If the process
sets flags to RS_HIPRI, for example, I_PEEK shall only look
for a high-priority message on the STREAM head read queue.
I_PEEK returns 1 if a message was retrieved, and returns 0
if no message was found on the STREAM head read queue, or
if the RS_HIPRI flag was set in flags and a high-priority
message was not present on the STREAM head read queue. It
does not wait for a message to arrive. On return, ctlbuf
specifies information in the control buffer, databuf speci?

fies information in the data buffer, and flags contains the value RS_HIPRI or 0.

I_SRDOPT Sets the read mode using the value of the argument arg. Read modes are described in read(). Valid arg flags are:

RNORM Byte-stream mode, the default.

RMSGD Message-discard mode.

RMSGN Message-nondiscard mode.

The bitwise-inclusive OR of RMSGD and RMSGN shall return [EINVAL]. The bitwise-inclusive OR of RNORM and either RMSGD or RMSGN shall result in the other flag overriding RNORM which is the default.

In addition, treatment of control messages by the STREAM head may be changed by setting any of the following flags in arg:

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

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

RPROTDIS Discard the control part of a message, deliver? ing any data portion, when a process issues a read().

The ioctl() function with the I_SRDOPT command shall fail if:

EINVAL The arg argument is not valid.

I_GRDOPT Returns the current read mode setting, as described above, in an int pointed to by the argument arg. Read modes are described in read().

I_NREAD Counts the number of data bytes in the data part of the first message on the STREAM head read queue and places this value in the int pointed to by arg. The return value for the command shall be the number of messages on the STREAM head read queue. For example, if 0 is returned in arg, but

the ioctl() return value is greater than 0, this indicates that a zero-length message is next on the queue.

I_FDINSERT Creates a message from specified buffer(s), adds informa? tion about another STREAM, and sends the message down? stream. The message contains a control part and an optional data part. The data and control parts to be sent are dis? tinguished by placement in separate buffers, as described below. The arg argument points to a strfdinsert structure.

The application shall ensure that the len member in the ctlbuf strbuf structure is set to the size of a t_uscalar_t plus the number of bytes of control information to be sent with the message. The fildes member specifies the file de? scriptor of the other STREAM, and the offset member, which must be suitably aligned for use as a t_uscalar_t, speci? fies the offset from the start of the control buffer where I_FDINSERT shall store a t_uscalar_t whose interpretation is specific to the STREAM end. The application shall ensure that the len member in the databuf strbuf structure is set to the number of bytes of data information to be sent with the message, or to 0 if no data part is to be sent.

The flags member specifies the type of message to be cre? ated. A normal message is created if flags is set to 0, and a high-priority message is created if flags is set to RS_HIPRI. For non-priority messages, I_FDINSERT shall block if the STREAM write queue is full due to internal flow con? trol conditions. For priority messages, I_FDINSERT does not block on this condition. For non-priority messages, I_FDINSERT does not block when the write queue is full and O_NONBLOCK is set. Instead, it fails and sets errno to [EA? GAIN].

I_FDINSERT also blocks, unless prevented by lack of inter? nal resources, waiting for the availability of message blocks in the STREAM, regardless of priority or whether

O_NONBLOCK has been specified. No partial message is sent.

The ioctl() function with the I_FDINSERT command shall fail if:

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

EAGAIN or ENOSR

Buffers cannot be allocated for the message that is to be created.

EINVAL One of the following:

-- The fildes member of the strfdinsert struc? ture is not a valid, open STREAM file de? scriptor.

-- The size of a t_uscalar_t plus offset is greater than the len member for the buffer specified through ctlbuf.

-- The offset member does not specify a prop? erly-aligned location in the data buffer.

-- An undefined value is stored in flags.

ENXIO Hangup received on the STREAM identified by either the fildes argument or the fildes member of the strfdinsert structure.

ERANGE The len member for the buffer specified through databuf does not fall within the range specified by the maximum and minimum packet sizes of the topmost STREAM module; or the len member for the buffer specified through databuf is larger than the maximum configured size of the data part of a message; or the len member for the buffer specified through ctl? buf is larger than the maximum configured size of the control part of a message.

I_STR Constructs an internal STREAMS ioctl() message from the data pointed to by arg, and sends that message downstream.

This mechanism is provided to send ioctl() requests to downstream modules and drivers. It allows information to be sent with ioctl(), and returns to the process any informa? tion sent upstream by the downstream recipient. I_STR shall block until the system responds with either a positive or negative acknowledgement message, or until the request times out after some period of time. If the request times out, it shall fail with errno set to [ETIME].

At most, one I_STR can be active on a STREAM. Further I_STR calls shall block until the active I_STR completes at the STREAM head. The default timeout interval for these re? quests is 15 seconds. The O_NONBLOCK flag has no effect on this call.

To send requests downstream, the application shall ensure that arg points to a strioctl structure.

The ic_cmd member is the internal ioctl() command intended for a downstream module or driver and ic_timout is the num? ber of seconds (-1=infinite, 0=use implementation-defined timeout interval, >0=as specified) an I_STR request shall wait for acknowledgement before timing out. ic_len is the number of bytes in the data argument, and ic_dp is a pointer to the data argument. The ic_len member has two uses: on input, it contains the length of the data argument passed in, and on return from the command, it contains the number of bytes being returned to the process (the buffer pointed to by ic_dp should be large enough to contain the maximum amount of data that any module or the driver in the STREAM can return).

The STREAM head shall convert the information pointed to by the strioctl structure to an internal ioctl() command mes? sage and send it downstream.

The ioctl() function with the I_STR command shall fail if:

EAGAIN or ENOSR

Unable to allocate buffers for the ioctl() message.

EINVAL The ic_len member is less than 0 or larger than the maximum configured size of the data part of a mes? sage, or ic_timout is less than -1.

ENXIO Hangup received on fildes.

ETIME A downstream ioctl() timed out before acknowledge? ment was received.

An I_STR can also fail while waiting for an acknowledgement if a message indicating an error or a hangup is received at the STREAM head. In addition, an error code can be re? turned in the positive or negative acknowledgement message, in the event the ioctl() command sent downstream fails. For these cases, I_STR shall fail with errno set to the value in the message.

I_SWROPT Sets the write mode using the value of the argument arg.

Valid bit settings for arg are:

SNDZERO Send a zero-length message downstream when a write() of 0 bytes occurs. To not send a zero- length message when a write() of 0 bytes oc? curs, the application shall ensure that this bit is not set in arg (for example, arg would be set to 0).

The ioctl() function with the I_SWROPT command shall fail if:

EINVAL arg is not the above value.

I_GWROPT Returns the current write mode setting, as described above, in the int that is pointed to by the argument arg.

I_SENDFD Creates a new reference to the open file description asso? ciated with the file descriptor arg, and writes a message on the STREAMS-based pipe fildes containing this reference, together with the user ID and group ID of the calling process.

The ioctl() function with the I_SENDFD command shall fail

if:

EAGAIN The sending STREAM is unable to allocate a message block to contain the file pointer; or the read queue of the receiving STREAM head is full and cannot ac? cept the message sent by I_SENDFD.

EBADF The arg argument is not a valid, open file descrip? tor.

EINVAL The fildes argument is not connected to a STREAM pipe.

ENXIO Hangup received on fildes.

The ioctl() function with the I_SENDFD command may fail if:

EINVAL The arg argument is equal to the fildes argument.

I_RECVFD Retrieves the reference to an open file description from a message written to a STREAMS-based pipe using the I_SENDFD command, and allocates a new file descriptor in the calling process that refers to this open file description. The arg argument is a pointer to a strrecvfd data structure as de? fined in <stropts.h>.

The fd member is a file descriptor. The uid and gid members are the effective user ID and effective group ID, respec? tively, of the sending process.

If O_NONBLOCK is not set, I_RECVFD shall block until a mes? sage is present at the STREAM head. If O_NONBLOCK is set, I_RECVFD shall fail with errno set to [EAGAIN] if no mes? sage is present at the STREAM head.

If the message at the STREAM head is a message sent by an I_SENDFD, a new file descriptor shall be allocated for the open file descriptor referenced in the message. The new file descriptor is placed in the fd member of the strrecvfd structure pointed to by arg.

The ioctl() function with the I_RECVFD command shall fail if:

EAGAIN A message is not present at the STREAM head read

queue and the O_NONBLOCK flag is set.

EBADMSG

The message at the STREAM head read queue is not a message containing a passed file descriptor.

EMFILE All file descriptors available to the process are currently open.

ENXIO Hangup received on fildes.

I_LIST Allows the process to list all the module names on the STREAM, up to and including the topmost driver name. If arg is a null pointer, the return value shall be the number of modules, including the driver, that are on the STREAM pointed to by fildes. This lets the process allocate enough space for the module names. Otherwise, it should point to a str_list structure.

The sl_nmods member indicates the number of entries the process has allocated in the array. Upon return, the sl_modlist member of the str_list structure shall contain the list of module names, and the number of entries that have been filled into the sl_modlist array is found in the sl_nmods member (the number includes the number of modules including the driver). The return value from ioctl() shall be 0. The entries are filled in starting at the top of the STREAM and continuing downstream until either the end of the STREAM is reached, or the number of requested modules (sl_nmods) is satisfied.

The ioctl() function with the I_LIST command shall fail if:

EINVAL The sl_nmods member is less than 1.

EAGAIN or ENOSR

Unable to allocate buffers.

I_ATMARK Allows the process to see if the message at the head of the STREAM head read queue is marked by some module downstream. The arg argument determines how the checking is done when there may be multiple marked messages on the STREAM head

read queue. It may take on the following values:

ANYMARK Check if the message is marked.

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

The bitwise-inclusive OR of the flags ANYMARK and LASTMARK is permitted.

The return value shall be 1 if the mark condition is satis? fied; otherwise, the value shall be 0.

The ioctl() function with the I_ATMARK command shall fail if:

EINVAL Invalid arg value.

I_CKBAND Checks if the message of a given priority band exists on the STREAM head read queue. This shall return 1 if a mes? sage of the given priority exists, 0 if no such message ex? ists, or -1 on error. arg should be of type int.

The ioctl() function with the I_CKBAND command shall fail if:

EINVAL Invalid arg value.

I_GETBAND Returns the priority band of the first message on the STREAM head read queue in the integer referenced by arg.

The ioctl() function with the I_GETBAND command shall fail if:

ENODATA

No message on the STREAM head read queue.

I_CANPUT Checks if a certain band is writable. arg is set to the priority band in question. The return value shall be 0 if the band is flow-controlled, 1 if the band is writable, or -1 on error.

The ioctl() function with the I_CANPUT command shall fail if:

EINVAL Invalid arg value.

I_SETCLTIME This request allows the process to set the time the STREAM head shall delay when a STREAM is closing and there is data

on the write queues. Before closing each module or driver, if there is data on its write queue, the STREAM head shall delay for the specified amount of time to allow the data to drain. If, after the delay, data is still present, it shall be flushed. The arg argument is a pointer to an integer specifying the number of milliseconds to delay, rounded up to the nearest valid value. If I_SETCLTIME is not performed on a STREAM, an implementation-defined default timeout in? terval is used.

The ioctl() function with the I_SETCLTIME command shall fail if:

EINVAL Invalid arg value.

I_GETCLTIME Returns the close time delay in the integer pointed to by arg.

Multiplexed STREAMS Configurations

The following commands are used for connecting and disconnecting multi? plexed STREAMS configurations. These commands use an implementation-de? fined default timeout interval.

I_LINK Connects two STREAMs, where fildes is the file descriptor of the STREAM connected to the multiplexing driver, and arg is the file descriptor of the STREAM connected to another driver. The STREAM designated by arg is connected below the multiplexing driver. I_LINK requires the multiplexing driver to send an acknowledgement message to the STREAM head regarding the connection. This call shall return a multiplexer ID number (an identifier used to disconnect the multiplexer; see I_UNLINK) on success, and -1 on failure.

The ioctl() function with the I_LINK command shall fail if:

ENXIO Hangup received on fildes.

ETIME Timeout before acknowledgement message was received at STREAM head.

EAGAIN or ENOSR

Unable to allocate STREAMS storage to perform the

I_LINK.

EBADF The arg argument is not a valid, open file descrip?
tor.

EINVAL The fildes argument does not support multiplexing;
or arg is not a STREAM or is already connected down?
stream from a multiplexer; or the specified I_LINK
operation would connect the STREAM head in more than
one place in the multiplexed STREAM.

An I_LINK can also fail while waiting for the multiplexing driver to acknowledge the request, if a message indicating an error or a hangup is received at the STREAM head of fildes. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_LINK fails with errno set to the value in the mes?
sage.

I_UNLINK Disconnects the two STREAMs specified by fildes and arg.
fildes is the file descriptor of the STREAM connected to the multiplexing driver. The arg argument is the multi?
plexer ID number that was returned by the I_LINK ioctl() command when a STREAM was connected downstream from the multiplexing driver. If arg is MUXID_ALL, then all STREAMs that were connected to fildes shall be disconnected. As in I_LINK, this command requires acknowledgement.

The ioctl() function with the I_UNLINK command shall fail if:

ENXIO Hangup received on fildes.

ETIME Timeout before acknowledgement message was received
at STREAM head.

EAGAIN or ENOSR
Unable to allocate buffers for the acknowledgement
message.

EINVAL Invalid multiplexer ID number.

An I_UNLINK can also fail while waiting for the multiplex?

ing driver to acknowledge the request if a message indicat?
ing an error or a hangup is received at the STREAM head of
fildes. In addition, an error code can be returned in the
positive or negative acknowledgement message. For these
cases, I_UNLINK shall fail with errno set to the value in
the message.

I_PLINK Creates a persistent connection between two STREAMs, where
fildes is the file descriptor of the STREAM connected to
the multiplexing driver, and arg is the file descriptor of
the STREAM connected to another driver. This call shall
create a persistent connection which can exist even if the
file descriptor fildes associated with the upper STREAM to
the multiplexing driver is closed. The STREAM designated
by arg gets connected via a persistent connection below the
multiplexing driver. I_PLINK requires the multiplexing
driver to send an acknowledgement message to the STREAM
head. This call shall return a multiplexer ID number (an
identifier that may be used to disconnect the multiplexer;
see I_PUNLINK) on success, and -1 on failure.
The ioctl() function with the I_PLINK command shall fail
if:
ENXIO Hangup received on fildes.
ETIME Timeout before acknowledgement message was received
at STREAM head.
EAGAIN or ENOSR
Unable to allocate STREAMS storage to perform the
I_PLINK.
EBADF The arg argument is not a valid, open file descrip?
tor.
EINVAL The fildes argument does not support multiplexing;
or arg is not a STREAM or is already connected down?
stream from a multiplexer; or the specified I_PLINK
operation would connect the STREAM head in more than

one place in the multiplexed STREAM.

An I_PLINK can also fail while waiting for the multiplexing driver to acknowledge the request, if a message indicating an error or a hangup is received at the STREAM head of fildes. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_PLINK shall fail with errno set to the value in the message.

I_PUNLINK Disconnects the two STREAMs specified by fildes and arg from a persistent connection. The fildes argument is the file descriptor of the STREAM connected to the multiplexing driver. The arg argument is the multiplexer ID number that was returned by the I_PLINK ioctl() command when a STREAM was connected downstream from the multiplexing driver. If arg is MUXID_ALL, then all STREAMs which are persistent connections to fildes shall be disconnected. As in I_PLINK, this command requires the multiplexing driver to acknowl? edge the request.

The ioctl() function with the I_PUNLINK command shall fail if:

ENXIO Hangup received on fildes.

ETIME Timeout before acknowledgement message was received at STREAM head.

EAGAIN or ENOSR

Unable to allocate buffers for the acknowledgement message.

EINVAL Invalid multiplexer ID number.

An I_PUNLINK can also fail while waiting for the multiplex? ing driver to acknowledge the request if a message indicat? ing an error or a hangup is received at the STREAM head of fildes. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_PUNLINK shall fail with errno set to the value in

the message.

RETURN VALUE

Upon successful completion, ioctl() shall return a value other than -1 that depends upon the STREAMS device control function. Otherwise, it shall return -1 and set errno to indicate the error.

ERRORS

Under the following general conditions, ioctl() shall fail if:

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

EINTR A signal was caught during the ioctl() operation.

EINVAL The STREAM or multiplexer referenced by fildes is linked (di? rectly or indirectly) downstream from a multiplexer.

If an underlying device driver detects an error, then ioctl() shall fail if:

EINVAL The request or arg argument is not valid for this device.

EIO Some physical I/O error has occurred.

ENOTTY The file associated with the fildes argument is not a STREAMS device that accepts control functions.

ENXIO The request and arg arguments are valid for this device driver, but the service requested cannot be performed on this particular sub-device.

ENODEV The fildes argument refers to a valid STREAMS device, but the corresponding device driver does not support the ioctl() func? tion.

If a STREAM is connected downstream from a multiplexer, any ioctl() command except I_UNLINK and I_PUNLINK shall set errno to [EINVAL].

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The implementation-defined timeout interval for STREAMS has histori? cally been 15 seconds.

RATIONALE

None.

FUTURE DIRECTIONS

The ioctl() function may be removed in a future version.

SEE ALSO

Section 2.6, STREAMS, close(), fcntl(), getmsg(), open(), pipe(), poll(), putmsg(), read(), sigaction(), write()

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

COPYRIGHT