NAME

streamio – STREAMS ioctl commands

SYNOPSIS

#include <sys/types.h>
#include <stropts.h>
#include <sys/conf.h>

int ioctl (int *fildes*, **int** *command*, **. . .** /* *arg* */);

DESCRIPTION

STREAMS (see **intro**(2)) **ioctl** commands are a subset of the **ioctl**(2) commands, and perform a variety of control functions on streams.

fildes is an open file descriptor that refers to a stream. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a *command*-specific data structure. *command* and *arg* are interpreted by the stream head. Certain combinations of these arguments may be passed to a module or driver in the stream.

Since these STREAMS commands **ioctl**s, they are subject to the errors described in **ioctl**(2). In addition to those errors, the call will fail with **errno** set to **EINVAL**, without processing a control function, if the stream referenced by *fildes* is linked below a multiplexor, or if *command* is not a valid value for a stream.

Also, as described in **ioctl**(2), STREAMS modules and drivers can detect errors. In this case, the module or driver sends an error message to the stream head containing an error value. This causes subsequent calls to fail with **errno** set to this value.

IOCTLS

The following **ioctl** commands, with error values indicated, are applicable 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. If the stream is a pipe, the module will be inserted between the stream heads of both ends of the pipe. It then calls the open routine of the newly-pushed module. On failure, **errno** is set to one of the following values:

EINVAL	Invalid module name.
EFAULT	*arg* points outside the allocated address space.
ENXIO	Open routine 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*. To remove a module from a pipe requires that the module was pushed on the side it is being removed from. *arg* should be **0** in an **I_POP** request. On failure, **errno** is set to one of the following values:

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 null terminated character string pointed at by *arg*. The buffer pointed to by *arg* should be at least **FMNAMESZ**+1 bytes long. This requires the declaration **#include <sys/conf.h>**. On failure, **errno** is set to one of the following values:

EFAULT *arg* points outside the allocated address space.
EINVAL No module present in stream.

I_FLUSH This request flushes all input and/or output queues, depending on the value of *arg*. Legal *arg* values are:

FLUSHR Flush read queues.
FLUSHW Flush write queues.
FLUSHRW Flush read and write queues.

If a pipe or FIFO does not have any modules pushed, the read queue of the stream head on either end is flushed depending on the value of *arg*.

If **FLUSHR** is set and *fildes* is a pipe, the read queue for that end of the pipe is flushed and the write queue for the other end is flushed. If *fildes* is a FIFO, both queues are flushed.

If **FLUSHW** is set and *fildes* is a pipe and the other end of the pipe exists, the read queue for the other end of the pipe is flushed and the write queue for this end is flushed. If *fildes* is a FIFO, both queues of the FIFO are flushed.

If **FLUSHRW** is set, all read queues are flushed, that is, the read queue for the FIFO and the read queue on both ends of the pipe are flushed.

Correct flush handling of a pipe or FIFO with modules pushed is achieved via the **pipemod** module. This module should be the first module pushed onto a pipe so that it is at the midpoint of the pipe itself.

On failure, **errno** is set to one of the following values:

ENOSR Unable to allocate buffers for flush message due to insufficient STREAMS memory resources.

EINVAL Invalid *arg* value.

ENXIO Hangup received on *fildes*.

I_FLUSHBAND Flushes a particular band of messages. *arg* points to a **bandinfo** structure that has the following members:

```
unsigned  char      bi_pri;
int                 bi_flag;
```

The **bi_flag** field may be one of **FLUSHR**, **FLUSHW**, or **FLUSHRW** as described earlier.

I_SETSIG Informs the stream head that the user wishes the kernel to issue the **SIGPOLL** signal (see **signal**(3C)) 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 user should be signaled. It is the bitwise OR of any combination of the following constants:

S_INPUT Any message other than an **M_PCPROTO** has arrived on a stream head read queue. This event is maintained for compatibility with previous releases. This event is triggered even if the message is of zero length.

S_RDNORM An ordinary (non-priority) message has arrived on a stream head read queue. This event is triggered even if the message is of zero length.

S_RDBAND A priority band message (band **>** 0) has arrived on a stream head read queue. This event is triggered even if the message is of zero length.

S_HIPRI A high priority message is present on the stream head read queue. This event is triggered even if the message is of zero length.

S_OUTPUT The write queue just below the stream head is no longer full. This notifies the user that there is room on the queue for sending (or writing) data downstream.

S_WRNORM This event is the same as **S_OUTPUT**.

S_WRBAND A priority band greater than 0 of a queue downstream exists and is writable. This notifies the user that there is room on the queue for sending (or writing) priority data downstream.

S_MSG A STREAMS signal message that contains the **SIGPOLL** signal has reached the front of the stream head read queue.

S_ERROR An **M_ERROR** message has reached the stream head.

S_HANGUP An **M_HANGUP** message 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.

A user process may choose to be signaled only of high priority messages by setting the *arg* bitmask to the value **S_HIPRI**.

Processes that wish to receive **SIGPOLL** signals must explicitly register to receive them using **I_SETSIG**. If several processes register to receive this signal for the same event on the same stream, each process will be signaled when the event occurs.

If the value of *arg* is zero, the calling process will be unregistered and will not receive further **SIGPOLL** signals. On failure, **errno** is set to one of the following values:

EINVAL *arg* value is invalid or *arg* is zero and process is not registered to receive the **SIGPOLL** signal.

EAGAIN Allocation of a data structure to store the signal request failed.

I_GETSIG Returns the events for which the calling process is currently registered to be sent a **SIGPOLL** signal. The events are returned as a bitmask pointed to by *arg*, where the events are those specified in the description of **I_SETSIG** above. On failure, **errno** is set to one of the following values:

EINVAL Process not registered to receive the **SIGPOLL** signal.
EFAULT *arg* points outside the allocated address space.

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. It returns 0 if the named module is not present. On failure, **errno** is set to one of the following values:

EFAULT *arg* points outside the allocated address space.
EINVAL *arg* does not contain a valid module name.

I_PEEK Allows a user to retrieve the information in the first message on the stream head read queue without taking the message off the queue. **I_PEEK** is analogous to **getmsg**(2) except that it does not remove the message from the queue. *arg* points to a **strpeek** structure, which contains the following members:

```
struct  strbuf      ctlbuf;
struct  strbuf      databuf;
long                flags;
```

The **maxlen** field in the **ctlbuf** and **databuf strbuf** structures (see **getmsg**(2)) must be set to the number of bytes of control information and/or data information, respectively, to retrieve. **flags** may be set to **RS_HIPRI** or **0**. If **RS_HIPRI** is set, **I_PEEK** will look for a high priority message on the stream head read queue. Otherwise, **I_PEEK** will look for the first 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. It does not wait for a message to arrive. On return, **ctlbuf** specifies information in the control buffer, **databuf** specifies information in the data buffer, and **flags** contains the value **RS_HIPRI** or **0**. On failure, **errno** is set to the following value:

EFAULT *arg* points, or the buffer area specified in **ctlbuf** or **databuf** is, outside the allocated address space.

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

EINVAL Illegal value for **flags**.

I_SRDOPT Sets the read mode (see **read**(2)) using the value of the argument *arg*. Legal *arg* values are:

RNORM Byte-stream mode, the default.
RMSGD Message-discard mode.
RMSGN Message-nondiscard mode.

In addition, the stream head's treatment of control messages may be changed by setting the following flags in *arg*:

RPROTNORM Reject **read( )** with **EBADMSG** if a control message is at the front of the stream head read queue.

RPROTDAT Deliver the control portion of a message as data when a user issues **read( )**. This is the default behavior.

RPROTDIS Discard the control portion of a message, delivering any data portion, when a user issues a **read**( ).

On failure, **errno** is set to the following value:

EINVAL *arg* is not one of the above legal values.

I_GRDOPT Returns the current read mode setting in an **int** pointed to by the argument *arg*. Read modes are described in **read**( ). On failure, **errno** is set to the following value:

EFAULT *arg* points outside the allocated address space.

I_NREAD Counts the number of data bytes in data blocks in the first message on the stream head read queue, and places this value in the location pointed to by *arg*. The return value for the command is the number of messages on the stream head read queue. For example, if zero is returned in *arg*, but the **ioctl** return value is greater than zero, this indicates that a zero-length message is next on the queue. On failure, **errno** is set to the following value:

EFAULT *arg* points outside the allocated address space.

I_FDINSERT Creates a message from user specified buffer(s), adds information about another stream and sends the message downstream. The message contains a control part and an optional data part. The data and control parts to be sent are distinguished by placement in separate buffers, as described below.

arg points to a **strfdinsert** structure, which contains the following members:

```
struct strbuf    ctlbuf;
struct strbuf    databuf;
long             flags;
int              fildes;
int              offset;
```

The **len** field in the **ctlbuf strbuf** structure (see **putmsg**(2)) must be set to the size of a pointer plus the number of bytes of control information to be sent with the message. *fildes* in the **strfdinsert** structure specifies the file descriptor of the other stream. **offset**, which must be word-aligned, specifies the number of bytes beyond the beginning of the control buffer where **I_FDINSERT** will store a pointer. This pointer will be the address of the read queue structure of the driver for the stream corresponding to **fildes** in the **strfdinsert** structure. The **len** field in the **databuf strbuf** structure must be set to the number of bytes of data information to be sent with the message or zero if no data part is to be sent.

flags specifies the type of message to be created. An ordinary (non-priority) message is created if **flags** is set to 0, a high priority message is created if **flags** is set to **RS_HIPRI**. For normal messages, **I_FDINSERT** will block if the stream write queue is full due to internal flow control conditions. For high priority messages, **I_FDINSERT** does not block on this condition. For normal messages, **I_FDINSERT** does not block when the write queue is full and **O_NDELAY** or **O_NONBLOCK** is set. Instead, it fails and sets **errno** to **EAGAIN**.

I_FDINSERT also blocks, unless prevented by lack of internal resources, waiting for the availability of message blocks, regardless of priority or whether **O_NDELAY** or **O_NONBLOCK** has been specified. No partial message is sent. On failure, **errno** is set to one of the following values:

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

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

EFAULT *arg* points, or the buffer area specified in **ctlbuf** or **databuf** is, outside the allocated address space.

EINVAL One of the following: **fildes** in the **strfdinsert** structure is not a valid, open stream file descriptor; the size of a pointer plus **offset** is greater than the **len** field for the buffer specified through **ctlptr**; **offset** does not specify a properly-aligned location in the data buffer; an undefined value is stored in **flags**.

ENXIO Hangup received on **fildes** of the **ioctl** call or **fildes** in the **strfdinsert** structure.

ERANGE The **len** field 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** field for the buffer specified through **databuf** is larger than the maximum configured size of the data part of a message, or the **len** field for the buffer specified through **ctlbuf** is larger than the maximum configured size of the control part of a message.

I_FDINSERT can also fail if an error message was received by the stream head of the stream corresponding to **fildes** in the **strfdinsert** structure. In this case, **errno** will be set to the value in the 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 user **ioctl** requests to downstream modules and drivers. It allows information to be sent with the **ioctl**, and will return to the user any information sent upstream by the downstream recipient. **I_STR** blocks 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 fails with **errno** set to **ETIME**.

At most one **I_STR** can be active on a stream. Further **I_STR** calls will block until the active **I_STR** completes at the stream head. The default timeout interval for these requests is 15 seconds. The **O_NDELAY** and **O_NONBLOCK** (see **open**(2)) flags have no effect on this call.

To send requests downstream, *arg* must point to a **strioctl** structure which contains the following members:

int	**ic_cmd;**
int	**ic_timout;**
int	**ic_len;**
char	***ic_dp;**

ic_cmd is the internal **ioctl** command intended for a downstream module or driver and **ic_timout** is the number of seconds (-1 = infinite, 0 = use default, >0 = as specified) an **I_STR** request will 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** field 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 user (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 will convert the information pointed to by the **strioctl** structure to an internal **ioctl** command message and send it downstream. On failure, **errno** is set to one of the following values:

ENOSR Unable to allocate buffers for the **ioctl** message due to insufficient STREAMS memory resources.

EFAULT Either *arg* points outside the allocated address space, or the buffer area specified by **ic_dp** and **ic_len** (separately for data sent and data returned) is outside the allocated address space.

EINVAL **ic_len** is less than 0 or **ic_len** is larger than the maximum configured size of the data part of a message or **ic_timout** is less than -1.

ENXIO Hangup received on *fildes*.

ETIME A downstream **ioctl** timed out before acknowledgement 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 returned in the positive or negative acknowledgement message, in the event the ioctl command sent downstream fails. For these cases, **I_STR** will fail with **errno** set to the value in the message.

I_SWROPT Sets the write mode using the value of the argument *arg*. Legal 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 occurs, this bit must not be set in *arg*.

On failure, **errno** may be set to the following value:

EINVAL *arg* is not the above legal 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 Requests the stream associated with *fildes* to send a message, containing a file pointer, to the stream head at the other end of a stream pipe. The file pointer corresponds to *arg*, which must be an open file descriptor.

I_SENDFD converts *arg* into the corresponding system file pointer. It allocates a message block and inserts the file pointer in the block. The user id and group id associated with the sending process are also inserted. This message is placed directly on the read queue (see **intro**(2)) of the stream head at the other end of the stream pipe to which it is connected. On failure, **errno** is set to one of the following values:

EAGAIN The sending stream is unable to allocate a message block to contain the file pointer.

EAGAIN The read queue of the receiving stream head is full and cannot accept the message sent by **I_SENDFD**.

EBADF *arg* is not a valid, open file descriptor.

EINVAL *fildes* is not connected to a stream pipe.

ENXIO Hangup received on *fildes*.

I_RECVFD Retrieves the file descriptor associated with the message sent by an **I_SENDFD ioctl** over a stream pipe. *arg* is a pointer to a data buffer large enough to hold an **strrecvfd** data structure containing the following members:

```
int       fd;
uid_t     uid;
gid_t     gid;
```

fd is an integer file descriptor. **uid** and **gid** are the user id and group id, respectively, of the sending stream.

If **O_NDELAY** and **O_NONBLOCK** are clear (see **open**(2)), **I_RECVFD** will block until a message is present at the stream head. If **O_NDELAY** or **O_NONBLOCK** is set, **I_RECVFD** will fail with **errno** set to **EAGAIN** if no message is present at the stream head.

If the message at the stream head is a message sent by an **I_SENDFD**, a new user file descriptor is allocated for the file pointer contained in the message. The new file descriptor is placed in the **fd** field of the **strrecvfd** structure. The structure is copied into the user data buffer pointed to by *arg*. On failure, **errno** is set to one of the following values:

EAGAIN A message is not present at the stream head read queue, and the **O_NDELAY** or **O_NONBLOCK** flag is set.

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

EFAULT *arg* points outside the allocated address space.

EMFILE **NOFILES** file descriptors are currently open.

ENXIO Hangup received on *fildes*.

EOVERFLOW *uid* or *gid* is too large to be stored in the structure pointed to by *arg*.

I_LIST Allows the user to list all the module names on the stream, up to and including the topmost driver name. If *arg* is **NULL**, the return value is the number of modules, including the driver, that are on the stream pointed to by *fildes*. This allows the user to allocate enough space for the module names. If *arg* is non-**NULL**, it should point to an **str_list** structure that has the following members:

```
int       sl_nmods;
struct    str_mlist*sl_modlist;
```

The **str_mlist** structure has the following member:

```
char      l_name[FMNAMESZ+1];
```

sl_nmods indicates the number of entries the user has allocated in the array and on return, **sl_modlist** contains the list of module names. The return value indicates the number of entries that have been filled in. On failure, **errno** may be set to one of the following values:

EINVAL The **sl_nmods** member is less than 1.

EAGAIN Unable to allocate buffers

I_ATMARK Allows the user to see if the current message on the stream head read queue is ''marked'' by some module downstream. *arg* determines how the checking is done when there may be multiple marked messages on the stream head read queue. It may take the following values:

ANYMARK Check if the message is marked.

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

The return value is 1 if the mark condition is satisfied and 0 otherwise. On failure, **errno** is set to the following value:

EINVAL Invalid *arg* value.

I_CKBAND Check if the message of a given priority band exists on the stream head read queue. This returns 1 if a message of a given priority exists, 0 if not, or -1 on error. *arg* should be an integer containing the value of the priority band in question. On failure, **errno** is set to the following value:

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*. On failure, **errno** is set to the following value:

ENODATA No message on the stream head read queue.

I_CANPUT Check if a certain band is writable. *arg* is set to the priority band in question. The return value is 0 if the priority band *arg* is flow controlled, 1 if the band is writable, or -1 on error. On failure, **errno** is set to the following value:

EINVAL Invalid *arg* value.

I_SETCLTIME Allows the user to set the time the stream head will delay when a stream is closing and there are data on the write queues. Before closing each module and driver, the stream head will delay for the specified amount of time to allow the data to drain. Note, however, that the module or driver may itself delay in its close routine; this delay is independent of the stream head's delay and is not settable. If, after the delay, data are still present, data will be flushed. *arg* is the number of milliseconds to delay, rounded up to the nearest legal value on the system. The default is fifteen seconds. On failure, **errno** is set to the following value:

EINVAL Invalid *arg* value.

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

I_SERROPT Sets the error mode using the value of the argument *arg*.

Normally stream head errors are persistent i.e. once they are set due to an **M_ERROR** or **M_HANGUP** the error condition will remain until the stream is closed. This option can be used to set the stream head into non-persistent error mode i.e. once the error has been returned in response to a **read**(2), **getmsg**(2), **ioctl**(2), **write**(2), or **putmsg**(2) call the error condition will be cleared. The error mode can be controlled independently for read and write side errors. Legal *arg* values are either none or one of:

RERRNORM Persistent read errors, the default.

RERRNONPERSIST
Non-persistent read errors.

OR'ed with either none or one of:

WERRNORM Persistent write errors, the default.

WERRNONPERSIST
Non-persistent write errors.

When no value is specified e.g. for the read side error behavior then the behavior for that side will be left unchanged.

On failure, **errno** is set to the following value:

EINVAL *arg* is not one of the above legal values.

I_GERROPT Returns the current error mode setting in an **int** pointed to by the argument *arg*. Error modes are described above for **I_SERROPT.** On failure, **errno** is set to the following value:

EFAULT *arg* points outside the allocated address space.

The following four commands are used for connecting and disconnecting multiplexed STREAMS configurations.

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* gets connected below the multiplexing driver. **I_LINK** requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call returns a multiplexor ID number (an identifier used to disconnect the multiplexor, see **I_UNLINK**) on success, and -1 on failure. On failure, **errno** is set to one of the following values:

ENXIO Hangup received on *fildes*.

ETIME Time out before acknowledgement message was received at stream head.

EAGAIN Temporarily unable to allocate storage to perform the **I_LINK**.

ENOSR Unable to allocate storage to perform the **I_LINK** due to insufficient STREAMS memory resources.

EBADF *arg* is not a valid, open file descriptor.

EINVAL *fildes* stream does not support multiplexing.

EINVAL *arg* is not a stream, or is already linked under a multiplexor.

EINVAL The specified link operation would cause a ‘‘cycle’’ in the resulting configuration; that is, a driver would be linked into the multiplexing configuration in more than one place.

EINVAL *fildes* is the file descriptor of a pipe or FIFO.

An **I_LINK** can also fail while waiting for the multiplexing driver to acknowledge the link 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** will fail with **errno** set to the value in the message.

I_UNLINK Disconnects the two streams specified by *fildes* and *arg*. *fildes* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexor ID number that was returned by the **I_LINK**. If *arg* is -1, then all streams that were linked to *fildes* are disconnected. As in **I_LINK**, this command requires the multiplexing driver to acknowledge the unlink. On failure, **errno** is set to one of the following values:

ENXIO Hangup received on *fildes*.

ETIME Time out before acknowledgement message was received at stream head.

ENOSR Unable to allocate storage to perform the **I_UNLINK** due to insufficient STREAMS memory resources.

EINVAL *arg* is an invalid multiplexor ID number or *fildes* is not the stream on which the **I_LINK** that returned *arg* was performed.

EINVAL *fildes* is the file descriptor of a pipe or FIFO.

An **I_UNLINK** can also fail while waiting for the multiplexing driver to acknowledge the link 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_UNLINK** will fail with **errno** set to the value in the message.

I_PLINK 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* gets connected via a persistent link below the multiplexing driver. **I_PLINK** requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call creates a persistent link that continues to exist even if the file descriptor *fildes* associated with the upper stream to the multiplexing driver is closed. This call returns a multiplexor ID number (an identifier that may be used to disconnect the multiplexor, see **I_PUNLINK**) on success, and -1 on failure. On failure, **errno** is set to one of the following values:

ENXIO Hangup received on *fildes*.

ETIME Time out before acknowledgement message was received at the stream head.

EAGAIN Unable to allocate STREAMS storage to perform the **I_PLINK**.

EBADF *arg* is not a valid, open file descriptor.

EINVAL *fildes* does not support multiplexing.

EINVAL *arg* is not a stream or is already linked under a multiplexor.

EINVAL The specified link operation would cause a ''cycle'' in the resulting configuration; that is, if a driver would be linked into the multiplexing configuration in more than one place.

EINVAL *fildes* is the file descriptor of a pipe or FIFO.

An **I_PLINK** can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error on 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** will fail with **errno** set to the value in the message.

I_PUNLINK Disconnects the two streams specified by *fildes* and *arg* that are connected with a persistent link. *fildes* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexor ID number that was returned by **I_PLINK** when a stream was linked below the multiplexing driver. If *arg* is **MUXID_ALL** then all streams that are persistent links to *fildes* are disconnected. As in **I_PLINK**, this command requires the multiplexing driver to acknowledge the unlink. On failure, **errno** is set to one of the following values:

ENXIO Hangup received on *fildes*.

ETIME Time out before acknowledgement message was received at the stream head.

EAGAIN Unable to allocate buffers for the acknowledgement message.

EINVAL Invalid multiplexor ID number.

EINVAL *fildes* is the file descriptor of a pipe or FIFO.

An **I_PUNLINK** can also fail while waiting for the multiplexing driver to acknowledge the link 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_PUNLINK** will fail with **errno** set to the value in the message.

RETURN VALUES

Unless specified otherwise above, the return value from **ioctl** is **0** upon success and **–1** upon failure with **errno** set as indicated.

SEE ALSO

intro(2), **close**(2), **fcntl**(2), **getmsg**(2), **ioctl**(2), **open**(2), **poll**(2), **putmsg**(2), **read**(2), **write**(2), **signal**(3C), **signal**(5)

STREAMS Programming Guide