NAME

tirdwr – Transport Interface read/write interface STREAMS module

SYNOPSIS

int ioctl(*fd*, **I_PUSH**, **"tirdwr");**

DESCRIPTION

tirdwr is a STREAMS module that provides an alternate interface to a transport provider which supports the Transport Interface (TI) functions of the Network Services library (see Section 3N). This alternate interface allows a user to communicate with the transport protocol provider using the **read**(2) and **write**(2) system calls. The **putmsg**(2) and **getmsg**(2) system calls may also be used. However, **putmsg** and **getmsg** can only transfer data messages between user and stream; control portions are disallowed.

The **tirdwr** module must only be pushed (see **I_PUSH** in **streamio**(7I)) onto a stream terminated by a transport protocol provider which supports the TI. After the **tirdwr** module has been pushed onto a stream, none of the Transport Interface functions can be used. Subsequent calls to TI functions cause an error on the stream. Once the error is detected, subsequent system calls on the stream return an error with **errno** set to **EPROTO**.

The following are the actions taken by the **tirdwr** module when pushed on the stream, popped (see **I_POP** in **streamio**(7I)) off the stream, or when data passes through it.

push When the module is pushed onto a stream, it checks any existing data destined for the user to ensure that only regular data messages are present. It ignores any messages on the stream that relate to process management, such as messages that generate signals to the user processes associated with the stream. If any other messages are present, the **I_PUSH** will return an error with **errno** set to **EPROTO**.

write The module takes the following actions on data that originated from a **write** system call:

All messages with the exception of messages that contain control portions (see the **putmsg** and **getmsg** system calls) are transparently passed onto the module's downstream neighbor.

Any zero length data messages are freed by the module and they will not be passed onto the module's downstream neighbor.

Any messages with control portions generate an error, and any further system calls associated with the stream fails with **errno** set to **EPROTO**.

read The module takes the following actions on data that originated from the transport protocol provider:

All messages with the exception of those that contain control portions (see the **putmsg** and **getmsg** system calls) are transparently passed onto the module's upstream neighbor.

The action taken on messages with control portions will be as follows:

Messages that represent expedited data generate an error. All further system calls associated with the stream fail with **errno** set to **EPROTO**.

Any data messages with control portions have the control portions removed from the message before to passing the message on to the upstream neighbor.

Messages that represent an orderly release indication from the transport provider generate a zero length data message, indicating the end of file, which will be sent to the reader of the stream. The orderly release message itself is freed by the module.

Messages that represent an abortive disconnect indication from the transport provider cause all further **write** and **putmsg** system calls to fail with **errno** set to **ENXIO**. All further **read** and **getmsg** system calls return zero length data (indicating end of file) once all previous data has been read.

With the exception of the above rules, all other messages with control portions generate an error and all further system calls associated with the stream will fail with **errno** set to **EPROTO**.

Any zero length data messages are freed by the module and they are not passed onto the module's upstream neighbor.

pop When the module is popped off the stream or the stream is closed, the module takes the following action:

If an orderly release indication has been previously received, then an orderly release request will be sent to the remote side of the transport connection.

SEE ALSO

intro(2), **getmsg**(2), **putmsg**(2), **read**(2), **write**(2), **intro**(3), **streamio**(7I), **timod**(7M)

STREAMS Programming Guide
Transport Interfaces Programming Guide