NAME

ldterm – standard STREAMS terminal line discipline module

SYNOPSIS

#include <sys/stream.h>
#include <sys/termios.h>

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

DESCRIPTION

ldterm is a STREAMS module that provides most of the **termio**(7I) terminal interface. This module does not perform the low-level device control functions specified by flags in the **c_cflag** word of the **termio/termios** structure or by the **IGNBRK, IGNPAR, PARMRK**, or **INPCK** flags in the **c_iflag** word of the **termio/termios** structure; those functions must be performed by the driver or by modules pushed below the **ldterm** module. **ldterm** performs all other **termio/termios** functions; some of them, however, require the cooperation of the driver or modules pushed below **ldterm** and may not be performed in some cases. These include the **IXOFF** flag in the **c_iflag** word and the delays specified in the **c_oflag** word.

ldterm also handles Extended Unix Code (EUC) and multi-byte characters.

The remainder of this section describes the processing of various STREAMS messages on the read- and write-side.

Read-side Behavior

Various types of STREAMS messages are processed as follows:

M_BREAK

When this message is received, depending on the state of the **BRKINT** flag, either an interrupt signal is generated or the message is treated as if it were an **M_DATA** message containing a single ASCII NUL character.

M_DATA

This message is normally processed using the standard **termio** input processing. If the **ICANON** flag is set, a single input record (‘‘line’’) is accumulated in an internal buffer and sent upstream when a line-terminating character is received. If the **ICANON** flag is not set, other input processing is performed and the processed data are passed upstream.

If output is to be stopped or started as a result of the arrival of characters (usually **CNTRL-Q** and **CNTRL-S**), **M_STOP** and **M_START** messages are sent downstream. If the **IXOFF** flag is set and input is to be stopped or started as a result of flow-control considerations, **M_STOPI** and **M_STARTI** messages are sent downstream.

M_DATA messages are sent downstream, as necessary, to perform echoing.

If a signal is to be generated, an **M_FLUSH** message with a flag byte of **FLUSHR** is placed on the read queue. If the signal is also to flush output, an **M_FLUSH** message with a flag byte of **FLUSHW** is sent downstream.

M_CTL

If the size of the data buffer associated with the message is the size of **struct iocblk, ldterm** will perform functional negotiation to determine where the

termio(7I) processing is to be done. If the command field of the **iocblk** structure (**ioc_cmd**) is set to **MC_NO_CANON**, the input canonical processing normally performed on **M_DATA** messages is disabled and those messages are passed upstream unmodified; this is for the use of modules or drivers that perform their own input processing, such as a pseudo-terminal in **TIOCREMOTE** mode connected to a program that performs this processing. If the command is **MC_DO_CANON**, all input processing is enabled. If the command is **MC_PART_CANON**, then an **M_DATA** message containing a **termios** structure is expected to be attached to the original **M_CTL** message. The **ldterm** module will examine the **iflag, oflag**, and **lflag** fields of the **termios** structure and from then on will process only those flags which have not been turned ON. If none of the above commands are found, the message is ignored; in any case, the message is passed upstream.

M_FLUSH
The read queue of the module is flushed of all its data messages and all data in the record being accumulated are also flushed. The message is passed upstream.

M_IOCACK
The data contained within the message, which is to be returned to the process, are augmented if necessary, and the message is passed upstream.

All other messages are passed upstream unchanged.

Write-side Behavior

Various types of STREAMS messages are processed as follows:

M_FLUSH
The write queue of the module is flushed of all its data messages and the message is passed downstream.

M_IOCTL
The function of this **ioctl** is performed and the message is passed downstream in most cases. The **TCFLSH** and **TCXONC ioctl**s can be performed entirely in the **ldterm** module, so the reply is sent upstream and the message is not passed downstream.

M_DATA
If the **OPOST** flag is set, or both the **XCASE** and **ICANON** flags are set, output processing is performed and the processed message is passed downstream along with any **M_DELAY** messages generated. Otherwise, the message is passed downstream without change.

All other messages are passed downstream unchanged.

IOCTLS

ldterm processes the following **TRANSPARENT ioctl**s. All others are passed downstream.

TCGETS/TCGETA
The message is passed downstream; if an acknowledgment is seen, the data provided by the driver and modules downstream are augmented and the acknowledgement is passed upstream.

TCSETS/TCSETSW/TCSETSF/TCSETA/TCSETAW/TCSETAF
The parameters that control the behavior of the **ldterm** module are changed. If a mode change requires options at the stream head to be changed, an **M_SETOPTS** message is sent upstream. If the **ICANON** flag is turned on or off, the read mode at the stream head is changed to message-nondiscard or byte-stream mode, respectively. If the **TOSTOP** flag is turned on or off, the tostop mode at the stream head is turned on or off, respectively. In any case, **ldterm** passes the ioctl on downstream for possible additional processing.

TCFLSH
If the argument is 0, an **M_FLUSH** message with a flag byte of **FLUSHR** is sent downstream and placed on the read queue. If the argument is 1, the write queue is flushed of all its data messages and an **M_FLUSH** message with a flag byte of **FLUSHW** is sent upstream and downstream. If the argument is 2, the write queue is flushed of all its data messages and an **M_FLUSH** message with a flag byte of **FLUSHRW** is sent downstream and placed on the read queue.

TCXONC
If the argument is 0 and output is not already stopped, an **M_STOP** message is sent downstream. If the argument is 1 and output is stopped, an **M_START** message is sent downstream. If the argument is 2 and input is not already stopped, an **M_STOPI** message is sent downstream. If the argument is 3 and input is stopped, an **M_STARTI** message is sent downstream.

TCSBRK
The message is passed downstream, so the driver has a chance to drain the data and then send and an **M_IOCACK** message upstream.

EUC_WSET
This call takes a pointer to an **eucioc** structure, and uses it to set the EUC line discipline's local definition for the code set widths to be used for subsequent operations. Within the stream, the line discipline may optionally notify other modules of this setting using **M_CTL** messages.

EUC_WGET
This call takes a pointer to an **eucioc** structure, and returns in it the EUC code set widths currently in use by the EUC line discipline.

SEE ALSO

termios(3), **console**(7D), **termio**(7I)

STREAMS Programming Guide