NAME

pty – pseudo-terminal driver

DESCRIPTION

The **pty** driver provides support for a pair of devices collectively known as a *pseudo-terminal*. The two devices comprising a pseudo-terminal are known as a *controller* and a *slave*. The slave device distinguishes between the **B0** baud rate and other baud rates specified in the **c_cflag** word of the **termios** structure, and the **CLOCAL** flag in that word. It does not support any of the other **termio**(7I) device control functions specified by flags in the **c_cflag** word of the **termios** structure and by the **IGNBRK**, **IGNPAR**, **PARMRK**, or **INPCK** flags in the **c_iflag** word of the **termios** structure, as these functions apply only to asynchronous serial ports. All other **termio**(7I) functions must be performed by STREAMS modules pushed atop the driver; when a slave device is opened, the **ldterm**(7M) and **ttcompat**(7M) STREAMS modules are automatically pushed on top of the stream, providing the standard **termio**(7I) interface.

Instead of having a hardware interface and associated hardware that supports the terminal functions, the functions are implemented by another process manipulating the controller device of the pseudo-terminal.

The controller and the slave devices of the pseudo-terminal are tightly connected. Any data written on the controller device is given to the slave device as input, as though it had been received from a hardware interface. Any data written on the slave terminal can be read from the controller device (rather than being transmitted from a UART).

By default, 48 pseudo-terminal pairs are configured as follows:

/dev/pty[p-r][0-9a-f] controller devices
/dev/tty[p-r][0-9a-f] slave devices

IOCTLS

The standard set of **termio ioctl**s are supported by the slave device. None of the bits in the **c_cflag** word have any effect on the pseudo-terminal, except that if the baud rate is set to **B0**, it will appear to the process on the controller device as if the last process on the slave device had closed the line; thus, setting the baud rate to **B0** has the effect of ''hanging up'' the pseudo-terminal, just as it has the effect of ''hanging up'' a real terminal.

There is no notion of ''parity'' on a pseudo-terminal, so none of the flags in the **c_iflag** word that control the processing of parity errors have any effect. Similarly, there is no notion of a ''break'', so none of the flags that control the processing of breaks, and none of the **ioctl**s that generate breaks, have any effect.

Input flow control is automatically performed; a process that attempts to write to the controller device will be blocked if too much unconsumed data is buffered on the slave device. The input flow control provided by the **IXOFF** flag in the **c_iflag** word is not supported.

The delays specified in the **c_oflag** word are not supported.

As there are no modems involved in a pseudo-terminal, the **ioctl**s that return or alter the state of modem control lines are silently ignored.

A few special **ioctl**s are provided on the controller devices of pseudo-terminals to provide the functionality needed by applications programs to emulate real hardware interfaces:

TIOCSTOP
: The argument is ignored. Output to the pseudo-terminal is suspended, as if a **STOP** character had been typed.

TIOCSTART
: The argument is ignored. Output to the pseudo-terminal is restarted, as if a **START** character had been typed.

TIOCPKT
: The argument is a pointer to an **int**. If the value of the **int** is non-zero, *packet* mode is enabled; if the value of the **int** is zero, packet mode is disabled. When a pseudo-terminal is in packet mode, each subsequent **read**(2) from the controller device will return data written on the slave device preceded by a zero byte (symbolically defined as **TIOCPKT_DATA**), or a single byte reflecting control status information. In the latter case, the byte is an inclusive-or of zero or more of the bits:

TIOCPKT_FLUSHREAD
: whenever the read queue for the terminal is flushed.

TIOCPKT_FLUSHWRITE
: whenever the write queue for the terminal is flushed.

TIOCPKT_STOP
: whenever output to the terminal is stopped using ^S.

TIOCPKT_START
: whenever output to the terminal is restarted.

TIOCPKT_DOSTOP
: whenever XON/XOFF flow control is enabled after being disabled; it is considered ''enabled'' when the **IXON** flag in the **c_iflag** word is set, the **VSTOP** member of the **c_cc** array is ^S and the **VSTART** member of the **c_cc** array is ^Q.

TIOCPKT_NOSTOP
: whenever XON/XOFF flow control is disabled after being enabled.

TIOCREMOTE
: The argument is a pointer to an **int**. If the value of the **int** is non-zero, *remote* mode is enabled; if the value of the **int** is zero, remote mode is disabled. This mode can be enabled or disabled independently of packet mode. When a pseudo-terminal is in remote mode, input to the slave device of the pseudo-terminal is flow controlled and not input edited (regardless of the mode the slave side of the pseudo-terminal). Each write to the controller device produces a record boundary for the process reading the slave device. In normal usage, a write of data is like the data typed as a line on the terminal; a write of 0 bytes is like typing an **EOF** character. Note: this means that a process writing to a pseudo-terminal controller in *remote* mode must keep track of line boundaries, and write only one line at a time to the controller. If, for example, it were to buffer up several **NEWLINE** characters and write them to the controller with one

write( ), it would appear to a process reading from the slave as if a single line containing several **NEWLINE** characters had been typed (as if, for example, a user had typed the **LNEXT** character before typing all but the last of those **NEWLINE** characters). Remote mode can be used when doing remote line editing in a window manager, or whenever flow controlled input is required.

EXAMPLES

```
#include <fcntl.h>
#include <sys/termios.h>

int fdm fds;

fdm = open("/dev/ptyp0, O_RDWR);     /* open master */
fds = open("/dev/ttyp0, O_RDWR);     /* open slave */
```

FILES

/dev/pty[p-z][0-9a-f]	pseudo-terminal controller devices
/dev/tty[p-z][0-9a-f]	pseudo-terminal slave devices

SEE ALSO

rlogin(1), **rlogind**(1M), **ldterm**(7M), **termio**(7I), **ttcompat**(7M),

NOTES

It is apparently not possible to send an **EOT** by writing zero bytes in **TIOCREMOTE** mode.