NAME

pty − pseudo-terminal interfaces

DESCRIPTION

A pseudo-terminal is a pair of virtual character devices that provide a bidirectional communication channel. One end of the channel is called the *master*; the other end is called the *slave*. The slave end of the pseudo-terminal provides an interface that behaves exactly like a classical terminal. A process that expects to be connected to a terminal, can open the slave end of a pseudo-terminal and then be driven by a program that has opened the master end. Anything that is written on the master end is provided to the process on the slave end as though it was input typed on a terminal. For example, writing the interrupt character (usually control-C) to the master device would cause an interrupt signal (**SIGINT**) to be generated for the foreground process group that is connected to the slave. Conversely, anything that is written to the slave end of the pseudo-terminal can be read by the process that is connected to the master end. Pseudo-terminals are used by applications such as network login services (**ssh**(1), **rlogin**(1), **telnet**(1)), terminal emulators, **script**(1), **screen**(1), and **expect**(1).

Historically, two pseudo-terminal APIs have evolved: BSD and System V. SUSv1 standardized a pseudo-terminal API based on the System V API, and this API should be employed in all new programs that use pseudo-terminals.

Linux provides both BSD-style and (standardized) System V-style pseudo-terminals. System V-style terminals are commonly called Unix 98 pseudo-terminals on Linux systems. Since kernel 2.6.4, BSD-style pseudo-terminals are considered deprecated (they can be disabled when configuring the kernel); Unix 98 pseudo-terminals should be used in new applications.

Unix 98 pseudo-terminals

An unused Unix 98 pseudo-terminal master is opened by calling **posix_openpt**(3). (This function opens the master clone device, */dev/ptmx*; see **pts**(4).) After performing any program-specific initializations, changing the ownership and permissions of the slave device using **grantpt**(3), and unlocking the slave using **unlockpt**(3)), the corresponding slave device can be opened by passing the name returned by **ptsname**(3) in a call to **open**(2).

The Linux kernel imposes a limit on the number of available Unix 98 pseudo-terminals. In kernels up to and including 2.6.3, this limit is configured at kernel compilation time (**CONFIG_UNIX98_PTYS**), and the permitted number of pseudo-terminals can be up to 2048, with a default setting of 256. Since kernel 2.6.4, the limit is dynamically adjustable via */proc/sys/kernel/pty/max*, and a corresponding file, */proc/sys/kernel/pty/nr*, indicates how many pseudo-terminals are currently in use. For further details on these two files, see **proc**(5).

BSD pseudo-terminals

BSD-style pseudo-terminals are provided as pre-created pairs, with names of the form */dev/ptyXY* (master) and */dev/ttyXY* (slave), where X is a letter from the 16-character set [p-za-e], and Y is a letter from the 16-character set [0-9a-f]. (The precise range of letters in these two sets varies across Unix implementations.) For example, */dev/ptyp1* and */dev/ttyp1* constitute a BSD pseudo-terminal pair. A process finds an unused pseudo-terminal pair by trying to **open**(2) each pseudo-terminal master until an open succeeds. The corresponding pseudo-terminal slave (substitute "tty" for "pty" in the name of the master) can then be opened.

FILES

/dev/ptmx (Unix 98 master clone device)
*/dev/pts/** (Unix 98 slave devices)
/dev/pty[p-za-e][0-9a-f] (BSD master devices)
/dev/tty[p-za-e][0-9a-f] (BSD slave devices)

NOTES

A description of the **TIOCPKT** **ioctl**(2), which controls packet mode operation, can be found in **tty_ioctl**(4).

The BSD **ioctl**(2) operations **TIOCSTOP**, **TIOCSTART**, **TIOCUCNTL**, and **TIOCREMOTE** have not been implemented under Linux.

SEE ALSO

select(2), **setsid**(2), **forkpty**(3), **openpty**(3), **termios**(3), **pts**(4), **tty**(4), **tty_ioctl**(4)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.