NAME

pty − pseudoterminal interfaces

DESCRIPTION

A pseudoterminal (sometimes abbreviated "pty") 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 pseudoterminal 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 pseudoterminal 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 pseudoterminal can be read by the process that is connected to the master end.

Data flow between master and slave is handled asynchronously, much like data flow with a physical terminal. Data written to the slave will be available at the master promptly, but may not be available immediately. Similarly, there may be a small processing delay between a write to the master, and the effect being visible at the slave.

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

Linux provides both BSD-style and (standardized) System V-style pseudoterminals. System V-style terminals are commonly called UNIX 98 pseudoterminals on Linux systems.

Since Linux 2.6.4, BSD-style pseudoterminals are considered deprecated: support can be disabled when building the kernel by disabling the **CONFIG_LEGACY_PTYS** option. (Starting with Linux 2.6.30, that option is disabled by default in the mainline kernel.) UNIX 98 pseudoterminals should be used in new applications.

UNIX 98 pseudoterminals

An unused UNIX 98 pseudoterminal 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 pseudoterminals. Up to and including Linux 2.6.3, this limit is configured at kernel compilation time (**CONFIG_UNIX98_PTYS**), and the permitted number of pseudoterminals can be up to 2048, with a default setting of 256. Since Linux 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 pseudoterminals are currently in use. For further details on these two files, see **proc**(5).

BSD pseudoterminals

BSD-style pseudoterminals are provided as precreated 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 pseudoterminal pair. A process finds an unused pseudoterminal pair by trying to **open**(2) each pseudoterminal master until an open succeeds. The corresponding pseudoterminal 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

Pseudoterminals are used by applications such as network login services (**ssh**(1), **rlogin**(1), **telnet**(1)), terminal emulators such as **xterm**(1), **script**(1), **screen**(1), **tmux**(1), **unbuffer**(1), and **expect**(1).

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

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

SEE ALSO

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