NAME

ptm – STREAMS pseudo-tty master driver

DESCRIPTION

The pseudo-tty subsystem simulates a terminal connection, where the master side represents the terminal and the slave represents the user process's special device end point. In order to use the pseudo-tty subsystem, a node for the master side driver **/dev/ptmx** and **N** number of nodes for the slave driver must be installed. See **pts**(7D). The master device is set up as a cloned device where its major device number is the major for the clone device and its minor device number is the major for the **ptm** driver. There are no nodes in the file system for master devices. The master pseudo driver is opened using the **open**(2) system call with **/dev/ptmx** as the device parameter. The clone open finds the next available minor device for the **ptm** major device.

A master device is available only if it and its corresponding slave device are not already open. When the master device is opened, the corresponding slave device is automatically locked out. Only one open is allowed on a master device. Multiple opens are allowed on the slave device. After both the master and slave have been opened, the user has two file descriptors which are the end points of a full duplex connection composed of two streams which are automatically connected at the master and slave drivers. The user may then push modules onto either side of the stream pair.

The master and slave drivers pass all messages to their adjacent queues. Only the **M_FLUSH** needs some processing. Because the read queue of one side is connected to the write queue of the other, the **FLUSHR** flag is changed to the **FLUSHW** flag and vice versa. When the master device is closed an **M_HANGUP** message is sent to the slave device which will render the device unusable. The process on the slave side gets the errno **ENXIO** when attempting to write on that stream but it will be able to read any data remaining on the stream head read queue. When all the data has been read, **read( )** returns 0 indicating that the stream can no longer be used. On the last close of the slave device, a 0-length message is sent to the master device. When the application on the master side issues a **read( )** or **getmsg( )** and 0 is returned, the user of the master device decides whether to issue a **close( )** that dismantles the pseudo-terminal subsystem. If the master device is not closed, the pseudo-tty subsystem will be available to another user to open the slave device.

If **O_NONBLOCK** or **O_NDELAY** is set, read on the master side returns –1 with errno set to **EAGAIN** if no data is available, and write returns –1 with errno set to **EAGAIN** if there is internal flow control.

IOCTLS

The master driver supports the **ISPTM** and **UNLKPT** ioctls that are used by the functions **grantpt**(3C), **unlockpt**(3C) and **ptsname**(3C). The ioctl **ISPTM** determines whether the file descriptor is that of an open master device. On success, it returns the major/minor number of the master device which can be used to determine the name of the corresponding slave device. The ioctl **UNLKPT** unlocks the master and slave devices. It returns 0 on success. On failure, the errno is set to **EINVAL** indicating that the master device is not open.

FILES

/dev/ptmx	master clone device
/dev/pts/M	slave devices (M = 0 -> N-1)

SEE ALSO

grantpt(3C), **ptsname**(3C), **unlockpt**(3C), **pckt**(7M), **pts**(7D)

STREAMS Programming Guide