NAME

IO::Pty – Pseudo TTY object class

VERSION

1.12

SYNOPSIS

```
use IO::Pty;

$pty = new IO::Pty;

$slave  = $pty->slave;

foreach $val (1..10) {
    print $pty "$val\n";
    $_ = <$slave>;
    print "$_";
}

close($slave);
```

DESCRIPTION

`IO::Pty` provides an interface to allow the creation of a pseudo tty.

`IO::Pty` inherits from `IO::Handle` and so provide all the methods defined by the `IO::Handle` package.

Please note that pty creation is very system-dependend. If you have problems, see IO::Tty for help.

CONSTRUCTOR

new

The `new` constructor takes no arguments and returns a new file object which is the master side of the pseudo tty.

METHODS

ttyname()

Returns the name of the slave pseudo tty. On UNIX machines this will be the pathname of the device. Use this name for informational purpose only, to get a slave filehandle, use **slave()**.

slave()

The `slave` method will return the slave filehandle of the given master pty, opening it anew if necessary. If IO::Stty is installed, you can then call `$slave->stty()` to modify the terminal settings.

close_slave()

The slave filehandle will be closed and destroyed. This is necessary in the parent after forking to get rid of the open filehandle, otherwise the parent will not notice if the child exits. Subsequent calls of `slave()` will return a newly opened slave filehandle.

make_slave_controlling_terminal()

This will set the slave filehandle as the controlling terminal of the current process, which will become a session leader, so this should only be called by a child process after a **fork()**, e.g. in the callback to `sync_exec()` (see Proc::SyncExec). See the `try` script (also `test.pl`) for an example how to correctly spawn a subprocess.

set_raw()

Will set the pty to raw. Note that this is a one-way operation, you need IO::Stty to set the terminal settings to anything else.

On some systems, the master pty is not a tty. This method checks for that and returns success anyway on such systems. Note that this method must be called on the slave, and probably should be called on the master, just to be sure, i.e.

```
$pty->slave->set_raw();
$pty->set_raw();
```

clone_winsize_from(*FH)
: Gets the terminal size from filehandle FH (which must be a terminal) and transfers it to the pty. Returns true on success and undef on failure. Note that this must be called upon the *slave*, i.e.

```
$pty->slave->clone_winsize_from(\*STDIN);
```

On some systems, the master pty also isatty. I actually have no idea if setting terminal sizes there is passed through to the slave, so if this method is called for a master that is not a tty, it silently returns OK.

See the `try` script for example code how to propagate SIGWINCH.

get_winsize()
: Returns the terminal size, in a 4–element list.

```
($row, $col, $xpixel, $ypixel) = $tty->get_winsize()
```

set_winsize($row, `$col`, `$xpixel`, `$ypixel`)
: Sets the terminal size. If not specified, `$xpixel` and `$ypixel` are set to 0. As with `clone_winsize_from`, this must be called upon the *slave*.

SEE ALSO

IO::Tty, IO::Tty::Constant, IO::Handle, Expect, Proc::SyncExec

MAILING LISTS

As this module is mainly used by Expect, support for it is available via the two Expect mailing lists, expectperl-announce and expectperl-discuss, at

```
http://lists.sourceforge.net/lists/listinfo/expectperl-announce
```

and

```
http://lists.sourceforge.net/lists/listinfo/expectperl-discuss
```

AUTHORS

Originally by Graham Barr <*gbarr@pobox.com*>, based on the Ptty module by Nick Ing-Simmons <*nik@tiuk.ti.com*>.

Now maintained and heavily rewritten by Roland Giersig <*RGiersig@cpan.org*>.

Contains copyrighted stuff from openssh v3.0p1, authored by Tatu Ylonen <ylo@cs.hut.fi>, Markus Friedl and Todd C. Miller <Todd.Miller@courtesan.com>.

COPYRIGHT

DISCLAIMER

Any Use of This Product, in Any Manner Whatsoever, Will Increase the Amount of Disorder in the Universe. Although No Liability Is Implied Herein, the Consumer Is Warned That This Process Will Ultimately Lead to the Heat Death of the Universe.