NAME

openpty, login_tty, forkpty − terminal utility functions

SYNOPSIS

#include <pty.h>

int openpty(int **amaster*, int **aslave*, char **name*,
const struct termios **termp*,
const struct winsize **winp*);

pid_t forkpty(int **amaster*, char **name*,
const struct termios **termp*,
const struct winsize **winp*);

#include <utmp.h>

int login_tty(int *fd*);

Link with −*lutil*.

DESCRIPTION

The **openpty**() function finds an available pseudoterminal and returns file descriptors for the master and slave in *amaster* and *aslave*. If *name* is not NULL, the filename of the slave is returned in *name*. If *termp* is not NULL, the terminal parameters of the slave will be set to the values in *termp*. If *winp* is not NULL, the window size of the slave will be set to the values in *winp*.

The **login_tty**() function prepares for a login on the terminal *fd* (which may be a real terminal device, or the slave of a pseudoterminal as returned by **openpty**()) by creating a new session, making *fd* the controlling terminal for the calling process, setting *fd* to be the standard input, output, and error streams of the current process, and closing *fd*.

The **forkpty**() function combines **openpty**(), **fork**(2), and **login_tty**() to create a new process operating in a pseudoterminal. The file descriptor of the master side of the pseudoterminal is returned in *amaster*. If *name* is not NULL, the buffer it points to is used to return the filename of the slave. The *termp* and *winp* arguments, if not NULL, will determine the terminal attributes and window size of the slave side of the pseudoterminal.

RETURN VALUE

If a call to **openpty**(), **login_tty**(), or **forkpty**() is not successful, −1 is returned and *errno* is set to indicate the error. Otherwise, **openpty**(), **login_tty**(), and the child process of **forkpty**() return 0, and the parent process of **forkpty**() returns the process ID of the child process.

ERRORS

openpty() fails if:

ENOENT
There are no available terminals.

login_tty() fails if **ioctl**(2) fails to set *fd* to the controlling terminal of the calling process.

forkpty() fails if either **openpty**() or **fork**(2) fails.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
forkpty(), **openpty**()	Thread safety	MT-Safe locale
login_tty()	Thread safety	MT-Unsafe race:ttyname

CONFORMING TO

These are BSD functions, present in glibc. They are not standardized in POSIX.

NOTES

The **const** modifiers were added to the structure pointer arguments of **openpty**() and **forkpty**() in glibc 2.8.

In versions of glibc before 2.0.92, **openpty**() returns file descriptors for a BSD pseudoterminal pair; since glibc 2.0.92, it first attempts to open a UNIX 98 pseudoterminal pair, and falls back to opening a BSD pseudoterminal pair if that fails.

BUGS

Nobody knows how much space should be reserved for *name*. So, calling **openpty**() or **forkpty**() with non-NULL *name* may not be secure.

SEE ALSO

fork(2), **ttyname**(3), **pty**(7)

COLOPHON

This page is part of release 5.02 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.