NAME

openpty, **forkpty** - auxiliary functions to obtain a pseudo-terminal

LIBRARY

System Utilities Library (libutil, -lutil)

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ioctl.h>
#include <termios.h>
#include <libutil.h>

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

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

DESCRIPTION

The function **openpty**() attempts to obtain the next available pseudo-terminal from the system (see pty(4)). If it successfully finds one, it subsequently changes the ownership of the slave device to the real UID of the current process, the group membership to the group "tty" (if such a group exists in the system), the access permissions for reading and writing by the owner, and for writing by the group, and invalidates any current use of the line by calling revoke(2).

If the argument *name* is not NULL, **openpty**() copies the pathname of the slave pty to this area. The caller is responsible for allocating the required space in this array.

If the arguments *termp* or *winp* are not NULL, **openpty**() initializes the termios and window size settings from the structures these arguments point to, respectively.

Upon return, the open file descriptors for the master and slave side of the pty are returned in the locations pointed to by *amaster* and *aslave*, respectively.

The **forkpty**() function first calls **openpty**() to obtain the next available pseudo-terminal from the system. Upon success, it forks off a new process. In the child process, it closes the descriptor for the master side of the pty, and calls login_tty(3) for the slave pty. In the parent process, it closes the descriptor for the slave side of the pty. The arguments *amaster*, *name*, *termp*, and *winp* have the same meaning as described for **openpty**().

RETURN VALUES

The **openpty**() function returns 0 on success, or -1 on failure.

The **forkpty**() function returns -1 on failure, 0 in the slave process, and the process ID of the slave process in the parent process.

ERRORS

The **openpty**() function may fail and set the global variable errno for any of the errors specified for the grantpt(3), posix_openpt(2), ptsname(3), and unlockpt(3) functions and the revoke(2) system call.

In addition to this, **forkpty**() may set it to any value as described for fork(2).

SEE ALSO

chmod(2), chown(2), fork(2), getuid(2), open(2), revoke(2), login_tty(3), pty(4), termios(4), group(5)

HISTORY

The **openpty**() and **forkpty**() functions first appeared in 4.3BSD-Reno.

BUGS

openpty() writes the slave terminal’s name to *name*, but does not check that sufficient space is available. It is advisable to use ptsname(3) instead.