NAME

pdfork, **pdgetpid**, **pdkill** - System calls to manage process descriptors

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/procdesc.h>

pid_t
pdfork(*int *fdp*, *int flags*);

int
pdgetpid(*int fd*, *pid_t *pidp*);

int
pdkill(*int fd*, *int signum*);

DESCRIPTION

Process descriptors are special file descriptors that represent processes, and are created using **pdfork**(), a variant of fork(2), which, if successful, returns a process descriptor in the integer pointed to by *fdp*. Processes created via **pdfork**() will not cause SIGCHLD on termination. **pdfork**() can accept the flags:

PD_DAEMON Instead of the default terminate-on-close behaviour, allow the process to live until it is explicitly killed with kill(2).

This option is not permitted in capsicum(4) capability mode (see cap_enter(2)).

PD_CLOEXEC
Set close-on-exec on process descriptor.

pdgetpid() queries the process ID (PID) in the process descriptor *fd*.

pdkill() is functionally identical to kill(2), except that it accepts a process descriptor, *fd*, rather than a PID.

The following system calls also have effects specific to process descriptors:

fstat(2) queries status of a process descriptor; currently only the *st_mode*, *st_birthtime*, *st_atime*, *st_ctime* and *st_mtime* fields are defined. If the owner read, write, and execute bits are set then the process represented by the process descriptor is still alive.

poll(2) and select(2) allow waiting for process state transitions; currently only POLLHUP is defined, and will be raised when the process dies. Process state transitions can also be monitored using kqueue(2) filter EVFILT_PROCDESC; currently only NOTE_EXIT is implemented.

close(2) will close the process descriptor unless PD_DAEMON is set; if the process is still alive and this is the last reference to the process descriptor, the process will be terminated with the signal SIGKILL.

RETURN VALUES

pdfork() returns a PID, 0 or -1, as fork(2) does.

pdgetpid() and **pdkill**() return 0 on success and -1 on failure.

ERRORS

These functions may return the same error numbers as their PID-based equivalents (e.g. **pdfork**() may return the same error numbers as fork(2)), with the following additions:

[EINVAL]	The signal number given to **pdkill**() is invalid.
[ENOTCAPABLE]	The process descriptor being operated on has insufficient rights (e.g. CAP_PDKILL for **pdkill**()).

SEE ALSO

close(2), fork(2), fstat(2), kill(2), kqueue(2), poll(2), wait4(2), capsicum(4), procdesc(4)

HISTORY

The **pdfork**(), **pdgetpid**(), and **pdkill**() system calls first appeared in FreeBSD 9.0.

Support for process descriptors mode was developed as part of the TrustedBSD Project.

AUTHORS

These functions and the capability facility were created by Robert N. M. Watson <*rwatson@FreeBSD.org*> and Jonathan Anderson <*jonathan@FreeBSD.org*> at the University of Cambridge Computer Laboratory with support from a grant from Google, Inc.