



Linux Ubuntu 26.04 Manual Pages on command 'pidfd_getfd.2'

\$ man pidfd_getfd.2

pidfd_getfd(2) System Calls Manual pidfd_getfd(2)

NAME

pidfd_getfd - obtain a duplicate of another process's file descriptor

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <sys/syscall.h>     /* Definition of SYS_* constants */
```

```
#include <unistd.h>
```

```
int syscall(SYS_pidfd_getfd, int pidfd, int targetfd,  
            unsigned int flags);
```

Note: glibc provides no wrapper for pidfd_getfd(), necessitating the use of syscall(2).

DESCRIPTION

The pidfd_getfd() system call allocates a new file descriptor in the calling process. This new file descriptor is a duplicate of an existing file descriptor, targetfd, in the process referred to by the PID file descriptor pidfd.

The duplicate file descriptor refers to the same open file description (see open(2)) as the original file descriptor in the process referred to by pidfd. The two file descriptors thus share file status flags and file offset. Furthermore, operations on the underlying file object (for example, assigning an address to a socket object using bind(2)) can equally be performed via the duplicate file descriptor.

The close-on-exec flag (FD_CLOEXEC; see fcntl(2)) is set on the file descriptor returned by pidfd_getfd().

The flags argument is reserved for future use. Currently, it must be specified as 0.

Permission to duplicate another process's file descriptor is governed by a ptrace access mode `PTRACE_MODE_ATTACH_REALCREDS` check (see `ptrace(2)`).

RETURN VALUE

On success, `pidfd_getfd()` returns a file descriptor (a nonnegative integer). On error, `-1` is returned and `errno` is set to indicate the error.

ERRORS

`EBADF` `pidfd` is not a valid PID file descriptor.

`EBADF` `targetfd` is not an open file descriptor in the process referred to by `pidfd`.

`EINVAL` `flags` is not 0.

`EMFILE` The per-process limit on the number of open file descriptors has been reached (see the description of `RLIMIT_NOFILE` in `getrlimit(2)`).

`ENFILE` The system-wide limit on the total number of open files has been reached.

`EPERM` The calling process did not have `PTRACE_MODE_ATTACH_REALCREDS` permissions (see `ptrace(2)`) over the process referred to by `pidfd`.

`ESRCH` The process referred to by `pidfd` does not exist (i.e., it has terminated and been waited on).

STANDARDS

Linux.

HISTORY

Linux 5.6.

NOTES

For a description of PID file descriptors, see `pidfd_open(2)`.

The effect of `pidfd_getfd()` is similar to the use of `SCM_RIGHTS` messages described in `unix(7)`, but differs in the following respects:

? In order to pass a file descriptor using an `SCM_RIGHTS` message, the two processes must first establish a UNIX domain socket connection.

? The use of `SCM_RIGHTS` requires cooperation on the part of the process whose file descriptor is being copied. By contrast, no such cooperation is necessary when using `pidfd_getfd()`.

? The ability to use `pidfd_getfd()` is restricted by a `PTRACE_MODE_ATTACH_REALCREDS` ptrace access mode check.

SEE ALSO

`clone3(2)`, `dup(2)`, `kcmp(2)`, `pidfd_open(2)`

