## NAME

getpid, getppid − get process identification

## SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>

pid_t getpid(void);
pid_t getppid(void);
```

## DESCRIPTION

**getpid**() returns the process ID (PID) of the calling process. (This is often used by routines that generate unique temporary filenames.)

**getppid**() returns the process ID of the parent of the calling process. This will be either the ID of the process that created this process using **fork**(), or, if that process has already terminated, the ID of the process to which this process has been reparented (either **init**(1) or a "subreaper" process defined via the **prctl**(2) **PR_SET_CHILD_SUBREAPER** operation).

## ERRORS

These functions are always successful.

## CONFORMING TO

POSIX.1-2001, POSIX.1-2008, 4.3BSD, SVr4.

## NOTES

If the caller's parent is in a different PID namespace (see **pid_namespaces**(7)), **getppid**() returns 0.

From a kernel perspective, the PID (which is shared by all of the threads in a multithreaded process) is sometimes also known as the thread group ID (TGID). This contrasts with the kernel thread ID (TID), which is unique for each thread. For further details, see **gettid**(2) and the discussion of the **CLONE_THREAD** flag in **clone**(2).

### C library/kernel differences

From glibc version 2.3.4 up to and including version 2.24, the glibc wrapper function for **getpid**() cached PIDs, with the goal of avoiding additional system calls when a process calls **getpid**() repeatedly. Normally this caching was invisible, but its correct operation relied on support in the wrapper functions for **fork**(2), **vfork**(2), and **clone**(2): if an application bypassed the glibc wrappers for these system calls by using **syscall**(2), then a call to **getpid**() in the child would return the wrong value (to be precise: it would return the PID of the parent process). In addition, there were cases where **getpid**() could return the wrong value even when invoking **clone**(2) via the glibc wrapper function. (For a discussion of one such case, see BUGS in **clone**(2).) Furthermore, the complexity of the caching code had been the source of a few bugs within glibc over the years.

Because of the aforementioned problems, since glibc version 2.25, the PID cache is removed: calls to **getpid**() always invoke the actual system call, rather than returning a cached value.

On Alpha, instead of a pair of **getpid**() and **getppid**() system calls, a single **getxpid**() system call is provided, which returns a pair of PID and parent PID. The glibc **getpid**() and **getppid**() wrapper functions transparently deal with this. See **syscall**(2) for details regarding register mapping.

## SEE ALSO

**clone**(2), **fork**(2), **gettid**(2), **kill**(2), **exec**(3), **mkstemp**(3), **tempnam**(3), **tmpfile**(3), **tmpnam**(3), **credentials**(7), **pid_namespaces**(7)

## COLOPHON

This page is part of release 5.05 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/.