NAME

kill − send signal to a process

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

```
#include <signal.h>

int kill(pid_t pid, int sig);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

kill():
_POSIX_C_SOURCE

DESCRIPTION

The **kill**() system call can be used to send any signal to any process group or process.

If *pid* is positive, then signal *sig* is sent to the process with the ID specified by *pid*.

If *pid* equals 0, then *sig* is sent to every process in the process group of the calling process.

If *pid* equals −1, then *sig* is sent to every process for which the calling process has permission to send signals, except for process 1 (*init*), but see below.

If *pid* is less than −1, then *sig* is sent to every process in the process group whose ID is *−pid*.

If *sig* is 0, then no signal is sent, but existence and permission checks are still performed; this can be used to check for the existence of a process ID or process group ID that the caller is permitted to signal.

For a process to have permission to send a signal, it must either be privileged (under Linux: have the **CAP_KILL** capability in the user namespace of the target process), or the real or effective user ID of the sending process must equal the real or saved set-user-ID of the target process. In the case of **SIGCONT**, it suffices when the sending and receiving processes belong to the same session. (Historically, the rules were different; see NOTES.)

RETURN VALUE

On success (at least one signal was sent), zero is returned. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EINVAL
An invalid signal was specified.

EPERM
The calling process does not have permission to send the signal to any of the target processes.

ESRCH
The target process or process group does not exist. Note that an existing process might be a zombie, a process that has terminated execution, but has not yet been **wait**(2)ed for.

STANDARDS

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

NOTES

The only signals that can be sent to process ID 1, the *init* process, are those for which *init* has explicitly installed signal handlers. This is done to assure the system is not brought down accidentally.

POSIX.1 requires that *kill(−1,sig)* send *sig* to all processes that the calling process may send signals to, except possibly for some implementation-defined system processes. Linux allows a process to signal itself, but on Linux the call *kill(−1,sig)* does not signal the calling process.

POSIX.1 requires that if a process sends a signal to itself, and the sending thread does not have the signal blocked, and no other thread has it unblocked or is waiting for it in **sigwait**(3), at least one unblocked signal must be delivered to the sending thread before the **kill**() returns.

Linux notes

Across different kernel versions, Linux has enforced different rules for the permissions required for an unprivileged process to send a signal to another process. In Linux 1.0 to 1.2.2, a signal could be sent if the effective user ID of the sender matched effective user ID of the target, or the real user ID of the sender matched the real user ID of the target. From Linux 1.2.3 until 1.3.77, a signal could be sent if the effective user ID of the sender matched either the real or effective user ID of the target. The current rules, which conform to POSIX.1, were adopted in Linux 1.3.78.

BUGS

In Linux 2.6 up to and including Linux 2.6.7, there was a bug that meant that when sending signals to a process group, **kill**() failed with the error **EPERM** if the caller did not have permission to send the signal to *any* (rather than *all*) of the members of the process group. Notwithstanding this error return, the signal was still delivered to all of the processes for which the caller had permission to signal.

SEE ALSO

kill(1), **_exit**(2), **pidfd_send_signal**(2), **signal**(2), **tkill**(2), **exit**(3), **killpg**(3), **sigqueue**(3), **capabilities**(7), **credentials**(7), **signal**(7)