NAME

kill – terminate or signal processes

SYNOPSIS

/usr/bin/kill –s *signal pid***. . .**
/usr/bin/kill –**l** [*exit_status*]
/usr/bin/kill [–*signal*] *pid*. . .

AVAILABILITY

SUNWcsu

DESCRIPTION

The **kill** utility sends a signal to the process or processes specified by each *pid* operand.

For each *pid* operand, the **kill** utility will perform actions equivalent to the **kill**(2) function called with the following arguments:

1. The value of the *pid* operand will be used as the *pid* argument.
2. The *sig* argument is the value specified by the –**s** option, or by **SIGTERM**, if none of these options is specified.

The signalled process must belong to the current user unless the user is the super-user.

See **NOTES** for descriptions of the shell built-in versions of **kill**.

OPTIONS

The following options are supported:

–**l** (The letter ell.) Write all values of *signal* supported by the implementation, if no operand is given. If an *exit_status* operand is given and it is a value of the **?** shell special parameter and **wait** corresponding to a process that was terminated by a signal, the *signal* corresponding to the signal that terminated the process will be written. If an *exit_status* operand is given and it is the unsigned decimal integer value of a signal number, the *signal* corresponding to that signal will be written. Otherwise, the results are unspecified.

–**s** *signal* Specify the signal to send, using one of the symbolic names defined in the **<signal.h>** description. Values of *signal* will be recognised in a case-independent fashion, without the **SIG** prefix. In addition, the symbolic name **0** will be recognised, representing the signal value zero. The corresponding signal will be sent instead of **SIGTERM**.

OPERANDS

The following operands are supported:

pid One of the following:

1. A decimal integer specifying a process or process group to be signalled. The process or processes selected by positive, negative and zero values of the *pid* operand will be as described for the kill function. If process number 0 is specified, all processes in the process group are signalled. If the first *pid* operand is negative, it should be preceded by –– to keep it from being interpreted as an option.
2. A job control job ID that identifies a background process group to be signalled. The job control job ID notation is applicable only for invocations

of **kill** in the current shell execution environment.

Note the job control job ID type of *pid* is available only on systems supporting the job control option.

exit_status A decimal integer specifying a signal number or the exit status of a process terminated by a signal.

USAGE

Process numbers can be found by using **ps**(1).

The job control job ID notation is not required to work as expected when **kill** is operating in its own utility execution environment. In either of the following examples:

nohup kill %1 &

system(kill %1");"

kill operates in a different environment and will not share the shell's understanding of job numbers.

OUTPUT

When the **–l** option is not specified, the standard output will not be used.

When the **–l** option is specified, the symbolic name of each signal will be written in the following format:

"%s%c", <*signal***>, <***separator***>**

where the **<***signal***>** is in upper-case, without the **SIG** prefix, and the **<***separator***>** will be either a newline character or a space character. For the last signal written, **<***separator***>** will be a newline character.

When both the **–l** option and *exit_status* operand are specified, the symbolic name of the corresponding signal will be written in the following format:

"%s0, <*signal***>**

EXAMPLES

Any of the commands:

kill –9 100 –165

kill –s kill 100 –165

kill –s KILL 100 –165

sends the SIGKILL signal to the process whose process ID is **100** and to all processes whose process group ID is **165**, assuming the sending process has permission to send that signal to the specified processes, and that they exist.

To avoid an ambiguity of an initial negative number argument specifying either a signal number or a process group, the former will always be the case. Therefore, to send the default signal to a process group (for example, 123), an application should use a command similar to one of the following:

kill –TERM –123

kill –– –123

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **kill**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 At least one matching process was found for each *pid* operand, and the specified signal was successfully processed for at least one matching process.

>0 An error occurred.

SEE ALSO

csh(1), **jobs**(1), **ksh**(1), **ps**(1), **sh**(1), **shell_builtins**(1), **wait**(1), **kill**(2), **signal**(3C), **environ**(5), **signal**(5)

NOTES

sh

The Bourne shell, **sh**, has a built-in version of **kill** to provide the functionality of the **kill** command for processes identified with a *jobid*. The **sh** syntax is:

kill [–*sig*] [*pid*] [%*job*] . . .
kill –l

csh

The C-shell, **csh**, also has a built-in **kill** command, whose syntax is:

kill [–*sig*] [*pid*] [%*job*] . . .
kill –l

The **csh kill** built-in sends the **TERM** (terminate) signal, by default, or the signal specified, to the specified process ID, the *job* indicated, or the current *job*. Signals are either given by number or by name. There is no default. Typing **kill** does not send a signal to the current job. If the signal being sent is **TERM** (terminate) or **HUP** (hangup), then the job or process is sent a **CONT** (continue) signal as well.

–l List the signal names that can be sent.

ksh

The **ksh kill**'s syntax is:

kill [–*sig*] [*pid*] [%*job*] . . .
kill –l

The **ksh kill** sends either the **TERM** (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number or by names (as given in **signal**(5) stripped of the prefix "SIG"). If the signal being sent is **TERM** (terminate) or **HUP** (hangup), then the job or process will be sent a **CONT** (continue) signal if it is stopped. The argument *job* can be the process id of a process that is not a member of one of the active jobs. In the second form, **kill –l**, the signal numbers and names are listed.