



[Home](#)
[Operating Systems](#)
[Linux](#)

[Linux Programmer's Guide Contents Page](#)

Linux Signals

1. [Introduction](#)

**System
Information**

2. [Inter-Process
Communication](#)
3. [Signals](#)

**Programming in
Various
Environments**

4. [Script Programming](#)
 1. [Script
 Variables](#)
 2. [Test
 Conditions](#)
 3. [Control and
 Iteration](#)
 4. [Commonly
 used
 Programs](#)
 5. [Shell
 Capabilities](#)
 6. [Example
 looping](#)

Signals

Linux Signals are:

Signal Name	Number	Description
SIGHUP	1	Hangup (POSIX)
SIGINT	2	Terminal interrupt (ANSI)
SIGQUIT	3	Terminal quit (POSIX)
SIGILL	4	Illegal instruction (ANSI)
SIGTRAP	5	Trace trap (POSIX)
SIGIOT	6	IOT Trap (4.2 BSD)
SIGBUS	7	BUS error (4.2 BSD)
SIGFPE	8	Floating point exception (ANSI)
SIGKILL	9	Kill(can't be caught or ignored) (POSIX)
SIGUSR1	10	User defined signal 1 (POSIX)
SIGSEGV	11	Invalid memory segment access (ANSI)
SIGUSR2	12	User defined signal 2 (POSIX)
SIGPIPE	13	Write on a pipe with no reader, Broken pipe (POSIX)
SIGALRM	14	Alarm clock (POSIX)

script	SIGTERM	15	Termination (ANSI)
7. Example	SIGSTKFLT	16	Stack fault
using			
Variables	SIGCHLD	17	Child process has stopped or exited, changed (POSIX)
8. Example	SIGCONT	18	Continue executing, if stopped (POSIX)
working			
with files	SIGSTOP	19	Stop executing(can't be caught or ignored) (POSIX)
9. Example	SIGTSTP	20	Terminal stop signal (POSIX)
install			
script	SIGTTIN	21	Background process trying to read, from TTY (POSIX)
5. C and C++			
Programming	SIGTTOU	22	Background process trying to write, to TTY (POSIX)
1. POSIX	SIGURG	23	Urgent condition on socket (4.2 BSD)
System			
Capabilities	SIGXCPU	24	CPU limit exceeded (4.2 BSD)
2. More	SIGXFSZ	25	File size limit exceeded (4.2 BSD)
POSIX	SIGVTALRM	26	Virtual alarm clock (4.2 BSD)
3. Threads	SIGPROF	27	Profiling alarm clock (4.2 BSD)
4. Mutexes	SIGWINCH	28	Window size change (4.3 BSD, Sun)
5. An example	SIGIO	29	I/O now possible (4.2 BSD)
viewmod			
program	SIGPWR	30	Power failure restart (System V)
6. An example			
serial			
program			
6. X Programming	As noted above, processes can ignore, block, or catch all signals except SIGSTOP and SIGKILL. If a process catches a signal, it means that it includes code that will take appropriate action when the signal is received. If the signal is not caught by the process, the kernel will take default action for the signal.		
7. Debugging			
8. Credits			
Linux			
Operating Systems			

[Home](#)

FIFOs

FIFOs are permanent objects and can be created using the `mkfifo(1)` or `mknod(1)` command. Inside the program, the FIFO can be created using the `mknod` command, then opened and read from or written to just like a normal file. The FIFO is normally in blocking mode when attempting to perform read operations.

[Linux Programmer's Guide Contents Page](#)