POSIX signals

You can see more about POSIX signals on

- man7
- wikipedia

```
       Linux supports both POSIX reliable signals (hereinafter "standard
       signals") and POSIX real-time signals.

   Standard signals

       First the signals described in the original POSIX.1-1990 standard.

       Signal     Value     Action   Comment
       ──────────────────────────────────────────────────────────────────────
       SIGHUP        1       Term    Hangup detected on controlling terminal
                                     or death of controlling process
       SIGINT        2       Term    Interrupt from keyboard
       SIGQUIT       3       Core    Quit from keyboard
       SIGILL        4       Core    Illegal Instruction
       SIGABRT       6       Core    Abort signal from abort(3)
       SIGFPE        8       Core    Floating point exception
       SIGKILL       9       Term    Kill signal
       SIGSEGV      11       Core    Invalid memory reference
       SIGPIPE      13       Term    Broken pipe: write to pipe with no
                                     readers
       SIGALRM      14       Term    Timer signal from alarm(2)
       SIGTERM      15       Term    Termination signal
       SIGUSR1   30,10,16    Term    User-defined signal 1
       SIGUSR2   31,12,17    Term    User-defined signal 2
       SIGCHLD   20,17,18    Ign     Child stopped or terminated
       SIGCONT   19,18,25    Cont    Continue if stopped
       SIGSTOP   17,19,23    Stop    Stop process
       SIGTSTP   18,20,24    Stop    Stop typed at terminal
       SIGTTIN   21,21,26    Stop    Terminal input for background process
       SIGTTOU   22,22,27    Stop    Terminal output for background process
```

POSIX signals

SIGABRT
: The SIGABRT signal is sent to a process to tell it to **abort**, i.e. to terminate. The signal is usually initiated by the process itself when it calls `abort` function of the C Standard Library, but it can be sent to the process from outside like any other signal.

SIGALRM, SIGVTALRM and SIGPROF
: The SIGALRM, SIGVTALRM and SIGPROF signal is sent to a process when the time limit specified in a call to a preceding **alarm** setting function (such as `setitimer`) elapses. SIGALRM is sent when real or clock time elapses. SIGVTALRM is sent when CPU time used by the process elapses. SIGPROF is sent when CPU time used by the process and by the system on behalf of the process elapses.

SIGBUS
: The SIGBUS signal is sent to a process when it causes a **bus error**. The conditions that lead to the signal being sent are, for example, incorrect memory access alignment or non-existent physical address.

SIGCHLD
: The SIGCHLD signal is sent to a process when a **child process** terminates, is interrupted, or resumes after being interrupted. One common usage of the signal is to instruct the operating system to clean up the resources used by a child process after its termination without an explicit call to the `wait` system call.

SIGCONT
The SIGCONT signal instructs the operating system to **continue** (restart) a process previously paused by the SIGSTOP or SIGTSTP signal. One important use of this signal is in job control in the Unix shell.

SIGFPE
The SIGFPE signal is sent to a process when it executes an erroneous arithmetic operation, such as division by zero (the name "FPE", standing for **floating-point exception**, is a misnomer as the signal covers integer-arithmetic errors as well).[2]

SIGHUP
The SIGHUP signal is sent to a process when its controlling terminal is closed. It was originally designed to notify the process of a serial line drop (a **hangup**). In modern systems, this signal usually means that the controlling pseudo or virtual terminal has been closed.[3] Many daemons will reload their configuration files and reopen their logfiles instead of exiting when receiving this signal.[4] nohup is a command to make a command ignore the signal.

SIGILL
The SIGILL signal is sent to a process when it attempts to execute an **illegal**, malformed, unknown, or privileged instruction.

SIGINT
The SIGINT signal is sent to a process by its controlling terminal when a user wishes to **interrupt** the process. This is typically initiated by pressing Ctrl-C, but on some systems, the "delete" character or "break" key can be used.[5]

SIGKILL
The SIGKILL signal is sent to a process to cause it to terminate immediately (**kill**). In contrast to SIGTERM and SIGINT, this signal cannot be caught or ignored, and the receiving process cannot perform any clean-up upon receiving this signal.

SIGPIPE
The SIGPIPE signal is sent to a process when it attempts to write to a pipe without a process connected to the other end.

SIGPOLL
The SIGPOLL signal is sent when an event occurred on an explicitly watched file descriptor.[6] Using it effectively leads to making asynchronous I/O requests since the kernel will **poll** the descriptor in place of the caller. It provides an alternative to active polling.

SIGRTMIN to SIGRTMAX
The SIGRTMIN to SIGRTMAX signals are intended to be used for user-defined purposes. They are **real-time** signals.

SIGQUIT
The SIGQUIT signal is sent to a process by its controlling terminal when the user requests that the process **quit** and perform a core dump.

SIGSEGV
The SIGSEGV signal is sent to a process when it makes an invalid virtual memory reference, or segmentation fault, i.e. when it performs a ***seg*mentation *v*iolation**.[7]

SIGSTOP
The SIGSTOP signal instructs the operating system to **stop** a process for later resumption.

SIGSYS
The SIGSYS signal is sent to a process when it passes a bad argument to a system call. In practice, this kind of signal is rarely encountered since applications rely on libraries (e.g. libc) to make the call for them.

SIGTERM
The SIGTERM signal is sent to a process to request its **termination**. Unlike the SIGKILL signal, it can be caught and interpreted or ignored by the process. This allows the process to perform nice termination releasing resources and saving state if appropriate. SIGINT is nearly identical to SIGTERM.

SIGTSTP
The SIGTSTP signal is sent to a process by its controlling terminal to request it to **stop temporarily**. It is commonly initiated by the user pressing Ctrl-Z. Unlike SIGSTOP, the process can register a signal handler for or ignore the signal.

SIGTTIN and SIGTTOU
The SIGTTIN and SIGTTOU signals are sent to a process when it attempts to read **in** or write **out** respectively from the **tty** while in the background. Typically, these signals are received only by processes under job control; daemons do not have controlling terminals and, therefore, should never receive these signals.

SIGTRAP
The SIGTRAP signal is sent to a process when an exception (or **trap**) occurs: a condition that a debugger has requested to be informed of — for example, when a particular function is executed, or when a particular variable changes value.

SIGURG
The SIGURG signal is sent to a process when a socket has **urgent** or out-of-band data available to read.

SIGUSR1 and SIGUSR2
The SIGUSR1 and SIGUSR2 signals are sent to a process to indicate **user-defined conditions**.

SIGXCPU
The SIGXCPU signal is sent to a process when it has used up the **CPU** for a duration that **exceeds** a certain predetermined user-settable value.[8] The arrival of a SIGXCPU signal provides the receiving process a chance to quickly save any intermediate results and to exit gracefully, before it is terminated by the operating system using the SIGKILL signal.

SIGXFSZ
The SIGXFSZ signal is sent to a process when it grows a **file** larger than the maximum allowed **size**.