**A list of signals and what they mean**

* “man 7 signal” gives the official manual page on signals.
* This is a fairly exhaustive list of signals. Only some of them will arise in the context of the make program.

|  |  |  |
| --- | --- | --- |
| ***No.*** | ***Short name*** | ***What it means*** |
|  |  |  |
| 1 | SIGHUP | If a process is being run from terminal and that terminal suddenly goes |
|  |  | away then the process receives this signal. “HUP” is short for “**h**ang **up**” |
|  |  | and refers to hanging up the telephone in the days of telephone modems. |
|  |  |  |
| 2 | SIGINT | The process was “**int**errupted”. This happens when you press Control+C on |
|  |  | the controlling terminal. |
|  |  |  |
| 3 | SIGQUIT |  |
|  |  |  |
| 4 | SIGILL | **Ill**egal instruction. The program contained some machine code the CPU |
|  |  | can't understand. |
|  |  |  |
| 5 | SIGTRAP | This signal is used mainly from within debuggers and program tracers. |
|  |  |  |
| 6 | SIGABRT | The program called the abort() function. This is an emergency stop. |
|  |  |  |
| 7 | SIGBUS | An attempt was made to access memory incorrectly. This can be caused by |
|  |  | alignment errors in memory access etc. |
|  |  |  |
| 8 | SIGFPE | A **f**loating **p**oint **e**xception happened in the program. |
|  |  |  |
| 9 | SIGKILL | The process was explicitly killed by somebody wielding the kill |
|  |  | program. |
|  |  |  |
| 10 | SIGUSR1 | Left for the programmers to do whatever they want. |
|  |  |  |
| 11 | SIGSEGV | An attempt was made to access memory not allocated to the process. This |
|  |  | is often caused by reading off the end of arrays etc. |
|  |  |  |
| 12 | SIGUSR2 | Left for the programmers to do whatever they want. |
|  |  |  |
| 13 | SIGPIPE | If a process is producing output that is being fed into another process that |
|  |  | consume it via a **pipe** (“producer | consumer”) and the consumer |
|  |  | dies then the producer is sent this signal. |
|  |  |  |
| 14 | SIGALRM | A process can request a “wake up call” from the operating system at some |
|  |  | time in the future by calling the alarm() function. When that time comes |
|  |  | round the wake up call consists of this signal. |
|  |  |  |
| 15 | SIGTERM | The process was explicitly killed by somebody wielding the kill |
|  |  | program. |
|  |  |  |
| *16* | *unused* |  |
|  |  |  |
| 17 | SIGCHLD | The process had previously created one or more **child** processes with the |
|  |  | fork() function. One or more of these processes has since died. |
|  |  |  |
| 18 | SIGCONT | (To be read in conjunction with SIGSTOP.) |
|  |  | If a process has been paused by sending it SIGSTOP then sending |
|  |  | SIGCONT to the process wakes it up again (“**cont**inues” it). |
|  |  |  |
| 19 | SIGSTOP | (To be read in conjunction with SIGCONT.) |
|  |  | If a process is sent SIGSTOP it is paused by the operating system. All its |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| ***No.*** | ***Short name*** | ***What it means*** |
|  |  |  |
|  |  | state is preserved ready for it to be restarted (by SIGCONT) but it doesn't |
|  |  | get any more CPU cycles until then. |
|  |  |  |
| 20 | SIGTSTP | Essentially the same as SIGSTOP. This is the signal sent when the user hits |
|  |  | Control+Z on the terminal. (SIGTSTP is short for “**t**erminal **st**o**p**”) The |
|  |  | only difference between SIGTSTP and SIGSTOP is that pausing is |
|  |  | only the *default* action for SIGTSTP but is the *required* action for |
|  |  | SIGSTOP. The process can opt to handle SIGTSTP differently but gets no |
|  |  | choice regarding SIGSTOP. |
|  |  |  |
| 21 | SIGTTIN | The operating system sends this signal to a backgrounded process when it |
|  |  | tries to read **in**put from its terminal. The typical response is to pause (as per |
|  |  | SIGSTOP and SIFTSTP) and wait for the SIGCONT that arrives when the |
|  |  | process is brought back to the foreground. |
|  |  |  |
| 22 | SIGTTOU | The operating system sends this signal to a backgrounded process when it |
|  |  | tries to write **out**put to its terminal. The typical response is as per |
|  |  | SIGTTIN. |
|  |  |  |
| 23 | SIGURG | The operating system sends this signal to a process using a network |
|  |  | connection when “**urg**ent” out of band data is sent to it. |
|  |  |  |
| 24 | SIGXCPU | The operating system sends this signal to a process that has exceeded its |
|  |  | CPU limit. You can cancel any CPU limit with the shell command |
|  |  | “ulimit -t unlimited” prior to running make though it is more |
|  |  | likely that something has gone wrong if you reach the CPU limit in make. |
|  |  |  |
| 25 | SIGXFSZ | The operating system sends this signal to a process that has tried to create a |
|  |  | file above the file size limit. You can cancel any file size limit with the |
|  |  | shell command “ulimit -f unlimited” prior to running make though it is |
|  |  | more likely that something has gone wrong if you reach the file size limit |
|  |  | in make. |
|  |  |  |
| 26 | SIGVTALRM | This is very similar to SIGALRM, but while SIGALRM is sent after a |
|  |  | certain amount of real time has passed, SIGVTALRM is sent after a certain |
|  |  | amount of time has been spent running the process. |
|  |  |  |
| 27 | SIGPROF | This is also very similar to SIGALRM and SIGVTALRM, but while |
|  |  | SIGALRM is sent after a certain amount of real time has passed, SIGPROF |
|  |  | is sent after a certain amount of time has been spent running the process |
|  |  | and running system code on behalf of the process. |
|  |  |  |
| 28 | SIGWINCH | (Mostly unused these days.) A process used to be sent this signal when one |
|  |  | of its windows was resized. |
|  |  |  |
| 29 | SIGIO | (Also known as SIGPOLL.) A process can arrange to have this signal sent |
|  |  | to it when there is some input ready for it to process or an output channel |
|  |  | has become ready for writing. |
|  |  |  |
| 30 | SIGPWR | A signal sent to processes by a power management service to indicate that |
|  |  | power has switched to a short term emergency power supply. The process |
|  |  | (especially long-running daemons) may care to shut down cleanlt before |
|  |  | the emergency power fails. |
|  |  |  |
| 31 | SIGSYS | Unused. |
|  |  |  |

My Findings About Signals

SIGINT and SIGQUIT are intended specifically for requests from the terminal: particular input characters can be assigned to generate these signals (depending on the terminal control settings). The default action for SIGINT is the same sort of process termination as the default action for SIGTERM and the unchangeable action for SIGKILL; the default action for SIGQUIT is also process termination, but additional implementation-defined actions may occur, such as the generation of a core dump. Either can be caught or ignored by the process if required.

The SIGUSR1 signals are set aside for you to use any way you want. They're useful for simple interprocess communication, if you write a signal handler for them in the program that receives the signal. **The default action is to terminate the process.**