|  |  |  |
| --- | --- | --- |
| **Signal** | **Number** | **Action** |
| SIGHUP | 1 | If a process is being run from terminal and that terminal suddenly goes away then the process receives this signal. “HUP” is short for “hang up” and refers to hanging up the telephone in the days of telephone modems. |
| SIGINT | 2 | End the process; Interruption generated when doing CTRLC on the keyboard. |
| SIGQUIT | 3 | The SIGQUIT signal is similar to SIGINT, except that it’s controlled by a different key—the QUIT character, usually *C-\*—and produces a core dump when it terminates the process, just like a program error signal. You can think of this as a program error condition “detected” by the user. |
| SIGILL | 4 | Illegal instruction. The program contained some machine code the CPU can't understand. |
| SIGTRAP | 5 | This signal is used mainly from within debuggers and program tracers. |
| SIGABRT | 6 | The program called the abort() function. This is an emergency stop. |
| SIGBUS | 7 | An attempt was made to access memory incorrectly. This can be caused by alignment errors in memory access etc. |
| SIGFPE | 8 | A floating point exception happened in the program. |
| SIGKILL | 9 | The process was explicitly killed by somebody wielding the kill program. |
| SIGUSR1 | 10 | Left for the programmers to do whatever they want. |
| SIGSEGV | 11 | An attempt was made to access memory not allocated to the process. This is often caused by reading off the end of arrays etc. |
| SIGUSR2 | 12 | Left for the programmers to do whatever they want. |
| SIGPIPE | 13 | 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. |
| SIGALRM | 14 | 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. |
| SIGTERM | 15 | The process was explicitly killed by somebody wielding the kill program. |
| SIGCHLD | 17 | The process had previously created one or more child processes with the fork() function. One or more of these processes has since died. |
| SIGCONT | 18 | (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 (“continues” it). |
| SIGSTOP | 19 | (To be read in conjunction with SIGCONT.) If a process is sent SIGSTOP it is paused by the operating system. All its state is preserved ready for it to be restarted (by SIGCONT) but it doesn't get any more CPU cycles until then. |
| SIGTSTP | 20 | Essentially the same as SIGSTOP. This is the signal sent when the user hits Control+Z on the terminal. (SIGTSTP is short for “terminal stop”) 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. |
| SIGTTIN | 21 | The operating system sends this signal to a backgrounded process when it tries to read input 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. |
| SIGTTOU | 22 | The operating system sends this signal to a backgrounded process when it tries to write output to its terminal. The typical response is as per SIGTTIN. |

https://www-uxsup.csx.cam.ac.uk/courses/moved.Building/signals.pdf