On Linux systems, numerous users often come across a program or process that locks up. The user will usually kill the software if the system does not do it first. Users may be familiar with some of the kill commands and signals, but does anyone understand all of them? There are four common kill commands and a total of 64 kill signals.

kill - The kill command will kill a process using the kill signal and PID given by the user. To use the SIGKILL signal with "kill", type one of the following for a process with a PID of 0710.

```
kill -9 0710
kill -SIGKILL 0710
```

The kill command accepts either the signal number or name (signals have both a number and name that can be referenced). The name must be in all caps.

killall - The killall command kills all process with a particular name. For instance, Nautilus may be running several times. To kill all of them type "killall -SIGQUIT nautilus". Also, if Firefox is running once and the user does not know the PID, use the killall command - "killall -9 firefox". The killall command also uses case-sensitive kill signals (they are all uppercase). Below demonstrates what will happen when the kill signal is typed in lowercase. Notice that the command uses the "s" as a parameter and then it does not know what to do with the rest of the information. It then tries to use "igkill" as a kill signal, but no such signal exists.

```
collier@Nacho-Laptop:~$ killall -sigkill nautilus
igkill: unknown signal; killall -l lists signals.
```

pkill - This command is a lot like killall except it allows partial names. So, "pkill -9 unity" will kill any process whose name begins with "unity".

xkill - This command allows users to kill a command by clicking the window. In a terminal, type "xkill" and then the cursor will change. Next, click a window to kill. The application should disappear and leave the memory. Then, the cursor will return to normal.

There are many kill signals that each serve a particular purpose. Typing "kill -l" will list the kill signals. Notice that all kill signals begin with "SIG"; this means SIGnal.

```
kill -l

1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1
36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4  39) SIGRTMIN+5
40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8  43) SIGRTMIN+9
```

```
44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13
52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9
56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6  59) SIGRTMAX-5
60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2  63) SIGRTMAX-1
64) SIGRTMAX
```

NOTE: Yes, certain numbers are missing because those signals are not supported on my system, or they were discontinued. If you run the same command, you may have different numbers missing/available.

TIP: Everyone has signals 1, 3, 9, and 15. Those are the most common signals. It is important to learn those very well when administering Linux systems.

Kill signals are not only used to close locked-up applications, but also stop software from performing unallowed tasks. This means some of these kill signals are part of security. Surprisingly, kill commands not only stop/kill processes but they also pause, continue, and restart processes.

SIGHUP - The SIGHUP signal disconnects a process from the parent process. This an also be used to restart processes. For example, "killall -SIGUP compiz" will restart Compiz. This is useful for daemons with memory leaks.

SIGINT - This signal is the same as pressing ctrl-c. On some systems, "delete" + "break" sends the same signal to the process. The process is interrupted and stopped. However, the process can ignore this signal.

SIGQUIT - This is like SIGINT with the ability to make the process produce a core dump.

SIGILL - When a process performs a faulty, forbidden, or unknown function, the system sends the SIGILL signal to the process. This is the ILLegal SIGnal.

SIGTRAP - This signal is used for debugging purposes. When a process has performed an action or a condition is met that a debugger is waiting for, this signal will be sent to the process.

SIGABRT - This kill signal is the abort signal. Typically, a process will initiate this kill signal on itself.

SIGBUS - When a process is sent the SIGBUS signal, it is because the process caused a bus error. Commonly, these bus errors are due to a process trying to use fake physical addresses or the process has its memory alignment set incorrectly.

SIGFPE - Processes that divide by zero are killed using SIGFPE. Imagine if humans got the death penalty for such math. NOTE: The author of this article was recently drug out to the street and shot for dividing by zero.

SIGKILL - The SIGKILL signal forces the process to stop executing immediately. The program cannot ignore this signal. This process does not get to clean-up either.

SIGUSR1 - This indicates a user-defined condition. This signal can be set by the user by programming the commands in sigusr1.c. This requires the programmer to know C/C++.

SIGSEGV - When an application has a segmentation violation, this signal is sent to the process.

SIGUSR2 - This indicates a user-defined condition.

SIGPIPE - When a process tries to write to a pipe that lacks an end connected to a reader, this signal is sent to the process. A reader is a process that reads data at the end of a pipe.

SIGALRM - SIGALRM is sent when the real time or clock time timer expires.

SIGTERM - This signal requests a process to stop running. This signal can be ignored. The process is given time to gracefully shutdown. When a program gracefully shuts down, that means it is given time to save its progress and release resources. In other words, it is not forced to stop. SIGINT is very similar to SIGTERM.

SIGCHLD - When a parent process loses its child process, the parent process is sent the SIGCHLD signal. This cleans up resources used by the child process. In computers, a child process is a process started by another process know as a parent.

SIGCONT - To make processes continue executing after being paused by the SIGTSTP or SIGSTOP signal, send the SIGCONT signal to the paused process. This is the CONTinue SIGnal. This signal is beneficial to Unix job control (executing background tasks).

SIGSTOP - This signal makes the operating system pause a process's execution. The process cannot ignore the signal.

SIGTSTP - This signal is like pressing ctrl-z. This makes a request to the terminal containing the process to ask the process to stop temporarily. The process can ignore the request.

SIGTTIN - When a process attempts to read from a tty (computer terminal), the process receives this signal.

SIGTTOU - When a process attempts to write from a tty (computer terminal), the process receives this signal.

SIGURG - When a process has urgent data to be read or the data is very large, the SIGURG signal is sent to the process.

SIGXCPU - When a process uses the CPU past the allotted time, the system sends the process this signal. SIGXCPU acts like a warning; the process has time to save the progress (if possible) and close before the system kills the process with SIGKILL.

SIGXFSZ - Filesystems have a limit to how large a file can be made. When a program tries to violate this limit, the system will send that process the SIGXFSZ signal.

SIGVTALRM - SIGVTALRM is sent when CPU time used by the process elapses.

SIGPROF - SIGPROF is sent when CPU time used by the process and by the system on behalf of the process elapses.

SIGWINCH - When a process is in a terminal that changes its size, the process receives this signal.

SIGIO - Alias to SIGPOLL or at least behaves much like SIGPOLL.

SIGPWR - Power failures will cause the system to send this signal to processes (if the system is still on).

SIGSYS - Processes that give a system call an invalid parameter will receive this signal.

SIGRTMIN* - This is a set of signals that varies between systems. They are labeled SIGRTMIN+1, SIGRTMIN+2, SIGRTMIN+3, ..., and so on (usually up to 15). These are user-defined signals; they must be programmed in the Linux kernel's source code. That would require the user to know C/C++.

SIGRTMAX* - This is a set of signals that varies between systems. They are labeled SIGRTMAX-1, SIGRTMAX-2, SIGRTMAX-3, ..., and so on (usually up to 14). These are user-defined signals; they must be programmed in the Linux kernel's source code. That would require the user to know C/C++.

SIGEMT - Processes receive this signal when an emulator trap occurs.

SIGINFO - Terminals may sometimes send status requests to processes. When this happens, processes will also receive this signal.

SIGLOST - Processes trying to access locked files will get this signal.

SIGPOLL - When a process causes an asynchronous I/O event, that process is sent the SIGPOLL signal.

Users can use these kill signals using one of the four kill commands. When sending a signal to a process owned by another user (like root), the user needs admin privileges and must use the sudo command. Be careful though, misuse of these signals can cause system damage. For instance, using SIGTERM on a GUI process like Compiz, X11, XFCE, Unity, Gnome-shell, etc. will make the system unviewable.