- It can be obtained by running uptime. For example:

```
$ uptime
 12:40:53 up  6:16,  2 users,  load average: 0.00, 0.00, 0.00
```

See also

- The *Scheduling with cron* recipe explains how to schedule tasks

Killing processes and send or respond to signals

Termination of processes is an important task we always come across, including the need to terminate all the instances of a program. The command line provides several options for terminating programs. An important concept regarding processes in Unix-like environments is that of **signals**. Signals are an inter-process communication mechanism used to interrupt a running process to perform some action. Termination of a program is also performed by using the same technique. This recipe is an introduction to signals and their usage.

Getting ready

Signals are an inter-process mechanism available in Linux. We can interrupt a process using a specific signal, which is associated with an integer value. When a process receives a signal, it responds by executing a signal handler. It is possible to send and receive signals and respond according to the signals in shell scripts as well. `KILL` is a signal used to terminate a process. The events such as *Ctrl + C*, and *Ctrl + Z* also send two types of signals. The `kill` command is used to send signals to processes and the `trap` command is used to handle the received signals.

How to do it...

1. In order to list all the signals available, use:

   ```
   $ kill -l
   ```

   It will print signal numbers and corresponding signal names.

2. Terminate a process as follows:

   ```
   $ kill PROCESS_ID_LIST
   ```

   The `kill` command issues a `TERM` signal by default. The process ID list is to be specified with space as a delimiter between process IDs.

3. In order to specify a signal to be sent to a process via the kill command, use:

   ```
   $ kill -s SIGNAL PID
   ```

   The `SIGNAL` argument is either a signal name or a signal number. Though there are many signals specified for different purposes, we frequently use only a few signals. They are as follows:

   - `SIGHUP 1`: Hangup detection on death of the controlling process or terminal
   - `SIGINT 2`: Signal which is emitted when *Ctrl* + *C* is pressed
   - `SIGKILL 9`: Signal used to force kill the process
   - `SIGTERM 15`: Signal used to terminate a process by default
   - `SIGTSTP 20`: Signal emitted when *Ctrl* + *Z* is pressed

4. We frequently use force kill for processes. In order to force kill a process, use:

   ```
   $ kill -s SIGKILL PROCESS_ID
   ```

   Or:

   ```
   $ kill -9 PROCESS_ID
   ```

There's more...

Let's walk through the additional commands used for terminating and signalling processes.

The kill family of commands

The `kill` command takes the process ID as the argument. There are also a few other commands in the `kill` family that accept the command name as the argument and send a signal to the process.

The `killall` command terminates the process by name as follows:

```
$ killall process_name
```

In order to send a signal to a process by name, use:

```
$ killall -s SIGNAL process_name
```

In order to force kill a process by name, use:

```
$ killall -9 process_name
```

For example:

```
$ killall -9 gedit
```