7. Print the last five lines as follows:

```
$ tail -n 5 file
```

8. In order to print all lines excluding the first M lines, use the following code:

```
$ tail -n +(M+1)
```

For example, to print all lines except the first five lines, M + 1 = 6, therefore the command will be as follows:

```
$ seq 100 | tail -n +6
```

This will print from 6 to 100.

One of the important usages of `tail` is to read a constantly growing file. Since new lines are constantly appended to the end of the file, `tail` can be used to display all new lines as they are written to the file. When we run `tail` without specifying any options, it will read the last 10 lines and exit. However, by that time, new lines would have been appended to the file by a process. To constantly monitor the growth of file, `tail` has a special option `-f` or `--follow`, which enables `tail` to follow the appended lines and keep being updated as data is added:

```
$ tail -f growing_file
```

You will probably want to use this on logfiles. The command to monitor the growth of the files would be:

```
# tail -f /var/log/messages
```

Or:

```
$ dmesg | tail -f
```

We frequently run `dmesg` to look at kernel ring buffer messages either to debug the USB devices or to look at `sdX` (`X` is the minor number for the `sd` device corresponding to a SCSI disk). The `-f` tail can also add a sleep interval `-s`, so that we can set the interval during which the file updates are monitored.

`tail` has the interesting property that allows it to terminate after a given process ID dies.

Suppose we are reading a growing file, and a process `Foo` is appending data to the file, the `-f` tail should be executed until the process `Foo` dies.

```
$ PID=$(pidof Foo)
$ tail -f file --pid $PID
```

When the process `Foo` terminates, `tail` also terminates.

Let us work on an example.

1. Create a new file `file.txt` and open the file in gedit (you can use any text editor).
2. Add new lines to the file and make frequent file saves in the gedit.
3. Now run the following commands:

```
$ PID=$(pidof gedit)
$ tail -f file.txt --pid $PID
```

When you make frequent changes to the file, it will be written to the terminal by the `tail` command. When you close the gedit, the `tail` command will get terminated.

Listing only directories – alternative methods

Listing only directories via scripting can be deceptively difficult. This recipe is worth knowing since it introduces multiple ways of listing only directories with various useful techniques.

Getting ready

There are multiple ways of listing directories only. When you ask people about these techniques, the first answer that they would probably give is `dir`. However, the `dir` command is just another command like `ls`, but with fewer options. Let us see how to list directories.

How to do it...

There are several ways in which directories in the current path can be displayed:

1. Using `ls` with `-l` to print directories:

```
$ ls -d */
```

2. Using `ls -F` with `grep`:

```
 $ ls -F | grep "/$"
```

3. Using `ls -l` with `grep`:

```
$ ls -l | grep "^d"
```

4. Using `find` to print directories:

```
$ find . -type d -maxdepth 1 -print
```