Alternately, we can remove all blank lines using `tr`, as discussed in the *Translating with tr* recipe in this chapter.

Displaying tabs as ^I

It is hard to distinguish tabs and repeated space characters. While writing programs in languages such as Python, tabs and spaces have different meanings for indentation purposes. Therefore, the use of tab instead of spaces causes problems in indentation. It may become difficult to track where the misplacement of the tab or space occurred by looking through a text editor. `cat` has a feature that can highlight tabs. This is very helpful in debugging indentation errors. Use the `-T` option with `cat` to highlight tab characters as `^I`. An example is as follows:

```
$ cat file.py
def function():
    var = 5
        next = 6
    third = 7


$ cat -T file.py
def function():
^Ivar = 5
        next = 6
^Ithird = 7^I
```

Line numbers

By using the `-n` flag for the `cat` command will output each line with a line number prefixed. It is to be noted that the `cat` command never changes a file; instead it produces an output on `stdout` with modifications to input according to the options provided. For example:

```
$ cat lines.txt
line
line
line


$ cat -n lines.txt
     1 line
     2 line
     3 line
```

`-n` will make the `cat` command output line numbers even for blank lines. If you want to skip numbering blank lines, use the `-b` option.

Recording and playing back of terminal sessions

When you need to show somebody how to do something in the terminal, or you need to prepare a tutorial on how to do something through the command line, you would normally type the commands manually and show them. Or, you could record a screencast and playback the video to them. There are other options for doing this. Using the commands `script` and `scriptreplay`, we can record the order and timing of the commands and save the data to text files. Using these files, others can replay and see the output of the commands on the terminal until the playback is complete.

Getting ready

The `script` and `scriptreplay` commands are available in most of the GNU/Linux distributions. Recording the terminal sessions to a file will be interesting. You can create tutorials of command-line hacks and tricks to achieve a task by recording the terminal sessions. You can also share the recorded files for others to playback and see how to perform a particular task using the command line.

How to do it...

We can start recording the terminal session using the following commands:

```
$ script -t 2> timing.log -a output.session
type commands;
…
..
exit
```

Note that this recipe will not work with shells that do not support redirecting only `stderr` to a file, such as the `csh` shell.