Two configuration files are passed to the `script` command as arguments. One file is for storing timing information (`timing.log`) at which each of the commands is run, whereas the other file (`output.session`) is used for storing the command output. The `-t` flag is used to dump timing data to `stderr`. Here, you will see, `2>` is used to redirect `stderr` to `timing.log`.

By using the two files, `timing.log` (stores timing information) and `output.session` (stores command output information), we can replay the sequence of command execution as follows:

```
$ scriptreplay timing.log output.session
# Plays the sequence of commands and output
```

How it works...

Usually, we record desktop videos to prepare tutorials. However, videos require considerable amount of storage. On the other hand, a terminal script file is just a text file, usually only in the order of kilobytes.

You can share the `timing.log` and `output.session` files to anyone who wants to replay a terminal session in their terminal.

Finding files and file listing

`find` is one of the great utilities in the Unix/Linux command-line toolbox. It is a very useful command for shell scripts; however, many people do not use it to its fullest effectiveness. This recipe deals with most of the common ways to utilize `find` to locate files.

Getting ready

The `find` command uses the following strategy: `find` descends through a hierarchy of files, matches the files that meet specified criteria, and performs some actions. Let's go through different use cases of `find` and its basic usages.

How to do it...

To list all the files and folders from the current directory to the descending child directories, use the following syntax:

```
$ find base_path
```

`base_path` can be any location from which `find` should start descending (for example, `/home/slynux/`).

An example of this command is as follows:

```
$ find . -print
# Print lists of files and folders
```

`.` specifies current directory and `..` specifies the parent directory. This convention is followed throughout the Unix filesystem.

The `-print` argument specifies to print the names (path) of the matching files. When `-print` is used, `'\n'` will be the delimiting character for separating each file. Also, note that even if you omit `-print`, the `find` command will print the filenames by default.

The `-print0` argument specifies each matching filename printed with the delimiting character `'\0'`. This is useful when a filename contains a space character.

There's more...

In this recipe we have learned the usage of the most commonly-used `find` command with an example. The `find` command is a powerful command-line tool and it is armed with a variety of interesting options. Let us take a look at them.

Search based on filename or regular expression match

The `-name` argument specifies a matching string for the filename. We can pass wildcards as its argument text. The `*.txt` command matches all the filenames ending with `.txt` and prints them. The `-print` option prints the filenames or file paths in the terminal that matches the conditions (for example, `-name`) given as options to the `find` command.

```
$ find /home/slynux -name "*.txt" -print
```

The `find` command has an option `-iname` (ignore case), which is similar to `-name` but it matches filenames while ignoring the case.

For example:

```
$ ls
example.txt  EXAMPLE.txt  file.txt
$ find . -iname "example*" -print
./example.txt
./EXAMPLE.txt
```