Getting ready

The `rename` command helps to change filenames using Perl regular expressions.
By combining the commands `find`, `rename`, and `mv`, we can perform a lot of things.

How to do it...

The easiest way of renaming image files in the current directory to our own filename, with a specific format, is by using the following script:

```
#!/bin/bash
#Filename: rename.sh
#Desc: Rename jpg and png files

count=1;
for img in `find . -iname '*.png' -o -iname '*.jpg' -type f -maxdepth
1`
do
  new=image-$count.${img##*.}

  echo "Renaming $img to $new"
  mv "$img" "$new"
  let count++

done
```

The output is as follows:

```
$ ./rename.sh
Renaming hack.jpg to image-1.jpg
Renaming new.jpg to image-2.jpg
Renaming next.png to image-3.png
```

The script renames all the `.jpg` and `.png` files in the current directory and its subdirectories to new filenames in the format `image-1.jpg`, `image-2.jpg`, `image-3.png`, `image-4.png`, and so on.

How it works...

In the previous script, we have used a `for` loop to iterate through the names of all files ending with a `.jpg` or `.png` extension. We use the `find` command to perform this search, where the `-o` option is used to specify multiple `-iname` options, which perform a case-insensitive match. By using `-maxdepth 1`, we make sure that `$img` will contain a filename only from the current directories, not its subdirectories.

We have initialized a variable `count=1` in order to keep track of the image number. The next step is to rename the file using the `mv` command. The new name of the file should be formulated for renaming. `${img##*.}` in the script parses the extension of the filename currently in the loop (see the *Slicing filenames based on extension* recipe for interpretation of `${img##*.}`).

`let count++` is used to increment the file number for each execution of the loop.

There are a variety of other ways to perform rename operations. Let us walk through a few of them:

- Renaming `*.JPG` to `*.jpg`:

  ```
  $ rename *.JPG *.jpg
  ```

- To replace space in the filenames with the `"_"` character:

  ```
  $ rename 's/ /_/g' *
  ```

  `# 's/ /_/g'` is the replacement part in the filename and `*` is the wildcard for the target files. It can be `*.txt` or any other wildcard pattern.

- To convert any filename of files from uppercase to lowercase and vice versa:

  ```
  $ rename 'y/A-Z/a-z/' *
  $ rename 'y/a-z/A-Z/' *
  ```

- To recursively move all the `.mp3` files to a given directory:

  ```
  $ find path -type f -name "*.mp3" -exec mv {} target_dir \;
  ```

- To recursively rename all the files by replacing space with the `"_"` character:

  ```
  $ find path -type f -exec rename 's/ /_/g' {} \;
  ```

Spell checking and dictionary manipulation

Most of the Linux distributions come with a dictionary file along with them. However, I find very few people to be aware of the dictionary file and hence, few make use of them. There is a command-line utility called `aspell` that functions as a spell checker. Let's go through a few scripts that make use of the dictionary file and the spell checker.

How to do it...

The `/usr/share/dict/` directory contains some of the dictionary files. Dictionary files are text files that contain a list of dictionary words. We can use this list to check whether a word is a dictionary word or not.

```
$ ls /usr/share/dict/
american-english  british-english
```