The next block of script is as follows:

```
 sed 's/.*<title>\(.*\)<\/title.*<author><name>\([^<]*\)<\/
name><email>
\([^<]*\).*/Author: \2 [\3] \nSubject: \1\n/'
```

This script matches the substring title by using `<title>\(.*\)<\/title`, the sender name by using `<author><name>\([^<]*\)<\/name>`, and e-mail by using `<email>\([^<]*\)`. Then back referencing is used as follows:

```
Author: \2 [\3] \nSubject: \1\n
```

This is to replace an entry for a mail with the matched items in an easy-to-read format. `\1` corresponds to the first substring match, `\2` for the second substring match, and so on.

The `SHOW_COUNT=5` variable is used to take the number of unread mail entries to be printed on the terminal.

`head` is used to display only the `SHOW_COUNT*3` lines from the first line. `SHOW_COUNT` is multiplied by three in order to show three lines of the output.

See also

- The *A primer on cURL* recipe in this chapter explains the `curl` command
- The *Basic sed primer* recipe in this chapter explains the `sed` command

Parsing data from a website

It is often useful to parse data from web pages by eliminating unnecessary details. `sed` and `awk` are the main tools that we will use for this task. You might have come across a list of actress rankings in a `grep` recipe in the *Chapter 4*, *Texting and driving*; it was generated by parsing the website page `http://www.johntorres.net/BoxOfficefemaleList.html`.

Let us see how we can parse the same data by using text-processing tools.

How to do it...

Let's go through the commands used to parse details of actresses from the website:

```
$ lynx -dump -nolist http://www.johntorres.net/BoxOfficefemaleList.html
| \
grep -o "Rank-.*" | \
sed -e 's/ *Rank-\([0-9]*\) *\(.*\)/\1\t\2/' | \
sort -nk 1 > actresslist.txt
```

The output will be as follows:

```
# Only 3 entries shown. All others omitted due to space limits
1   Keira Knightley
2   Natalie Portman
3   Monica Bellucci
```

How it works...

Lynx is a command-line web browser—it can dump a text version of a website as we would see in a web browser, rather than showing us the raw code. Hence, we can avoid the job of removing the HTML tags. We use the `-nolist` option for `lynx`, as we don't need the numbers that it adds automatically with each link. Parsing and formatting the lines that contain `Rank` is done by using `sed`, as follows:

```
sed -e 's/ *Rank-\([0-9]*\) *\(.*\)/\1\t\2/'
```

These lines are then sorted according to the ranks.

See also

- The *Basic sed primer* recipe in this chapter explains the `sed` command
- The *Downloading a web page as plain text* recipe in this chapter explains the `lynx` command

Image crawler and downloader

Image crawlers are very useful when we need to download all the images that appear in a web page. Instead of going through the HTML sources and picking all the images, we can use a script to parse the image files and download them automatically. Let's see how to do it.

How to do it...

Let's write a Bash script to crawl and download the images from a web page, as follows:

```
#!/bin/bash
#Desc: Images downloader
#Filename: img_downloader.sh

if [ $# -ne 3 ];
then
  echo "Usage: $0 URL -d DIRECTORY"
  exit -1
```