Introduction

The Web is becoming the face of technology and the central access point for data processing. Though shell scripting cannot do everything that languages like PHP can do on the Web, there are still many tasks to which shell scripts are ideally suited. In this chapter, we will explore some recipes that can be used to parse website content, download and obtain data, send data to forms, and automate website-usage tasks and similar activities. We can automate many activities that we perform interactively through a browser with a few lines of scripting. Access to the functionalities provided by the HTTP protocol with command-line utilities enables us to write scripts that are suitable for solving most of the web-automation utilities. Have fun while going through the recipes of this chapter.

Downloading from a web page

Downloading a file or a web page from a given URL is simple. A few command-line download utilities are available to perform this task.

Getting ready

`wget` is a file download command-line utility. It is very flexible and can be configured with many options.

How to do it...

A web page or a remote file can be downloaded by using `wget`, as follows:

```
$ wget URL
```

For example:

```
$ wget http://slynux.org
--2010-08-01 07:51:20--  http://slynux.org/
Resolving slynux.org... 174.37.207.60
Connecting to slynux.org|174.37.207.60|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15280 (15K) [text/html]
Saving to: "index.html"

100%[======================================>] 15,280      75.3K/s   in
0.2s

2010-08-01 07:51:21 (75.3 KB/s) - "index.html" saved [15280/15280]
```

It is also possible to specify multiple download URLs, as follows:

```
$ wget URL1 URL2 URL3 ..
```

How it works...

Usually, files are downloaded with the same filename as in the URL, and the download log information or progress is written to `stdout`.

You can specify the output filename with the `-O` option. If the file with the specified filename already exists, it will be truncated first and the downloaded file will be written to the specified file.

You can also specify a different `logfile` path rather than printing logs to `stdout`, by using the `-o` option:

```
$ wget ftp://example_domain.com/somefile.img -O dloaded_file.img -o log
```

By using the preceding command, nothing will be printed on screen. The log or progress will be written to the log, and the output file will be `dloaded_file.img`.

There is a chance that downloads might break due to unstable Internet connections. In that case, we can use the number of tries as an argument so that once interrupted, the utility will retry the download that many times before giving up.

To specify the number of tries, use the `-t` flag as follows:

```
$ wget -t 5 URL
```

Or to ask `wget` to keep trying infinitely, use `-t` as follows:

```
$ wget -t 0 URL
```

There's more...

The `wget` utility has several additional options that can be used under different problem domains. Let us go through a few of them.

Restricting the download speed

When we have a limited Internet bandwidth and many applications are sharing the Internet connection, and if a large file is given for download, it will suck all the bandwidth and may cause other processes to starve for the bandwidth. The `wget` command comes with a built-in option to specify the maximum bandwidth limit the download job can possess. Hence, all the applications can simultaneously access the Internet fairly.