Lab Exercises 1

Dr. Sarvar Abdullaev
s.abdullaev@inha.uz

January 27, 2020

The purpose of this lab is to review HTTP by using cURL - a command-line tool for generating HTTP requests.

1 cURL Overview

cURL is a command line tool and a library which can be used to receive and send data between a client and a server or any two machines connected over the internet. It supports a wide range of protocols like `HTTP`, `FTP`, `IMAP`, `LDAP`, `POP3`, `SMTP` and many more.

Due to its versatile nature, cURL is used in many applications and for many use cases. For example, the command line tool can be used to download files, testing APIs and debugging network problems. In this article, we shall look at how you can use the cURL command line tool to perform various tasks.

2 Install cURL

If you use Linux, MacOS or Windows 10 then cURL comes preinstalled. For older versions of Windows, download latest version of cURL from here: `https://curl.haxx.se/windows/`.

Once you download the ZIP file and extract it, you will find a folder named `curl-<version number>-mingw`. Move this folder into a directory of your choice. In this lab, we will assume our folder is named `curl-7.63.0-win64-mingw`, and we have moved it under `C:\`.

Next, you should add cURL's bin directory to the Windows `PATH` environment variable, so that Windows can find it when you type curl in the command prompt. For this to work, you need to follow these steps:

1. Open the *"Advanced System Properties"* dialog by running `systempropertiesadvanced` from the Windows Run dialog (Windows key + R).
2. Click on the *"Environment Variables"* button.
3. Double-click on *"Path"* from the *"System variables"* section, and add the path `C:\curl-7.63.0-win64-mingw\bin`. For Windows 10, you can do this with the *"New"* button on the right. On older versions of Windows, you can type in `;C:\curl-7.63.0-win64-mingw\bin` (notice the semicolon at the beginning) at the end of the *"Value"* text box.

Once you complete the above steps, you can type `curl` to check if this is working. If everything went well, you should see the following output:

```
C:\Users\Administrator>curl
curl: try 'curl --help' or 'curl --manual' for more information
```

3 Getting Started

The basic syntax of using cURL is simply:

```
curl <url>
```

This fetches the content available at the given URL, and prints it onto the terminal. For example, if you run `curl example.com`, you should be able to see the HTML page printed, as shown below: Try below code to

```
~ $ curl example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif
;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 50px;
        background-color: #fff;
        border-radius: 1em;
    }
```

Figure 1

see what HTTP request and response you received. `-v` stand for *verbose mode*

```
curl -v example.com
```

You can save HTTP response literally to a file `dump.txt`. `-trace-ascii` turns on trace mode and stores everything to a given file:

```
curl --trace-ascii dump.txt --trace-time example.com
```

You can save body of the HTTP response to a file `resp.html`.

```
curl -o resp.html example.com
```

Note you can use this to download any file to your system.

```
curl -o iut_logo.png https://inha.uz/static/inha/img/logo2.png
```

Or you want to leave the original name of the file:

```
curl -O https://inha.uz/static/inha/img/logo2.png
```

4 Following Redirects

By default, when cURL receives a redirect after making a request, it doesn't automatically make a request to the new URL. As an example of this, consider the URL `http://www.facebook.com`. When you make a request using to this URL, the server sends a HTTP 3XX redirect to `https://www.facebook.com/`. However, the response body is otherwise empty. So, if you try this out, you will get an empty output.

Receive HTTP 302 redirect to different location response.

```
curl www.facebook.com
```

Follows the redirect.

```
curl -L www.facebook.com
```

When debugging issues with a website, you may want to view the HTTP response headers sent by the server. To enable this feature, you can use the `-i` option.

Let us continue with our previous example, and confirm that there is indeed a HTTP 3XX redirect when you make a HTTP request to `http://www.facebook.com/`, by running:

```
curl -L -i http://www.facebook.com/
```

If you use the `-o/-O` option in combination with `-i`, the response headers and body will be saved into a single file.

You can also turn on verbose (`-v`) or trace modes(`--trace-ascii`) to see literally how HTTP communication occurred between your computer and Facebook server.

5 Setting HTTP Request Headers

When testing APIs, you may need to set custom headers on the HTTP request. cURL has the `-H` option which you can use for this purpose. If you want to send the custom header `X-My-Custom-Header` with the value of `123` to `https://httpbin.org/get`, you should run:

```
curl -H 'X-My-Custom-Header: 123' https://httpbin.org/get
```

`httpbin.org` is a very useful website that allows you to view details of the HTTP request that you sent to it.

The data returned by the URL shows that this header was indeed set:

```
~ $ curl -H 'X-My-Custom-Header: 123' https://httpbin.org/get
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Connection": "close",
    "Host": "httpbin.org",
    "Referer": "",
    "User-Agent": "curl/7.54.0",
    "X-My-Custom-Header": "123"
  },
  "origin": "[redacted]",
  "url": "https://httpbin.org/get"
}
```

Figure 2

You can also override any default headers sent by cURL such as the "User-Agent" or "Host" headers. The HTTP client (in our case, cURL) sends the "User-Agent" header to tell the server about the type and version of the client used. Also, the client uses the "Host" header to tell the server about the site it should serve. This header is needed because a web server can host multiple websites at a single IP address.

Also, if you want to set multiple headers, you can simply repeat the `-H` option as required.

```
curl -H 'User-Agent: Mozilla/5.0' -H 'Host: www.afisha.uz' gazeta.uz
```

However, cURL does have certain shortcuts for frequently used flags. You can set the "User-Agent" header with the `-A` option:

```
curl -A Mozilla/5.0 http://httpbin.org/get
```

The "Referer" header is used to tell the server the location from which they were referred to by the previous site. It is typically sent by browsers when requesting Javascript or images linked to a page, or when following redirects. If you want to set a "Referer" header, you can use the `-e` flag:

```
curl -e http://www.google.com/ http://httpbin.org/get
```

Otherwise, if you are following a set of redirects, you can simply use `-e ';auto'` and cURL will take care of setting the redirects by itself.

6 Making HTTP POST requests

By default, cURL sends GET requests, but you can also use it to send POST requests with the `-d` or `--data` option. All the fields must be given as `key=value` pairs separated by the ampersand (`&`) character. As an example, you can make a POST request to `httpbin.org` with some parameters:

```
curl --data "firstname=boolean&lastname=world" https://httpbin.org/post
```

From the output, you can easily tell that we posted two parameters (this appears under the "form" key):

```
~ $ curl --data "firstname=boolean&lastname=world" https://httpbin.org/post
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "firstname": "boolean",
    "lastname": "world"
  },
  "headers": {
    "Accept": "*/*",
    "Connection": "close",
    "Content-Length": "32",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.54.0"
  },
  "json": null,
  "origin": "[redacted]",
  "url": "https://httpbin.org/post"
}
```

Figure 3

Any special characters such as `@, %, =` or spaces in the value should be URL-encoded manually. So, if you wanted to submit a parameter "email" with the value "`test@example.com`", you would use:

```
curl --data "email=test%40example.com" https://httpbin.org/post
```

Alternatively, you can just use `--data-urlencode` to handle this for you. If you wanted to submit two parameters, `email` and `name`, this is how you should use the option:

```
curl --data-urlencode "email=test@example.com"
     --data-urlencode "name=Boolean World" https://httpbin.org/post
```

If the `--data` parameter is too big to type on the terminal, you can save it to a file and then submit it using @, like so:

```
curl --data @params.txt example.com
```

So far, we have seen how you can make POST requests using cURL. If you want to upload files using a POST request, you can use the `-F` ("form") option. Here, we will submit the file `test.c`, under the parameter name file:

```
curl -F file=@test.c https://httpbin.org/post
```

```
~ $ curl -F file=@test.c https://httpbin.org/post
{
  "args": {},
  "data": "",
  "files": {
    "file": "int main() {\n\tprintf(\"Hello world!\\n\");\n\treturn 0;\n}"
  },
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Connection": "close",
    "Content-Length": "250",
    "Content-Type": "multipart/form-data; boundary=------------------------69a45
7f9a3be53e2",
    "Expect": "100-continue",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.54.0"
  },
  "json": null,
  "origin": "[redacted]",
  "url": "https://httpbin.org/post"
}
```

Figure 4

This shows the content of the file, showing that it was submitted successfully:

You can also submit JSON data using the `--data` option. However, most servers expect to receive a POST request with key-value pairs, similar to the ones we have discussed previously. So, you need to add an additional header called '*Content-Type: application/json*' so that the server understands it's dealing with JSON data and handles it appropriately. Also, you don't need to URL-encode data when submitting JSON.

So if you had the following JSON data and want to make a POST request to `https://httpbin.org/post`:

```
{
  "email": "test@example.com",
  "name": ["Boolean", "World"]
}
```

Then, you can submit the data with:

```
curl    --data '{"email":"test@example.com", "name": ["Boolean", "World"]}'
        -H 'Content-Type: application/json' https://httpbin.org/post
```

In this case, you can see the data appear under the json value in the httpbin.org output:

You can also save the JSON file, and submit it in the same way as we did previously:

```
curl --data @data.json https://httpbin.org/post
```

7 Changing the request method

Previously, we have seen how you can send POST requests with cURL. Sometimes, you may need to send a POST request with no data at all. In that case, you can simply change the request method to POST with the `-X` option, like so:

```
curl -X POST https://httpbin.org/post
```

You can also change the request method to anything else, such as PUT, DELETE or PATCH. One notable exception is the HEAD method, which cannot be set with the `-X` option. The HEAD method is used to check if a document is present on the server, but without downloading the document. To use the HEAD method, use the `-I` option:

```
curl -I https://www.booleanworld.com/
```

```
~ $ curl --data '{"email":"test@example.com", "name": ["Boolean", "World"]}' -H
'Content-Type: application/json' https://httpbin.org/post
{
  "args": {},
  "data": "{\"email\":\"test@example.com\", \"name\": [\"Boolean\", \"World\"]}"
,
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Connection": "close",
    "Content-Length": "58",
    "Content-Type": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.54.0"
  },
  "json": {
    "email": "test@example.com",
    "name": [
      "Boolean",
      "World"
    ]
  },
  "origin": "[redacted]",
  "url": "https://httpbin.org/post"
}
```

Figure 5

8 Replicating Browser Requests with cURL

If you want to replicate a request made through your browser through cURL, you can use the Chrome, Firefox and Safari developer tools to get a cURL command to do so.

The steps involved are the same for all platforms and browsers:

1. Open developer tools in Firefox/Chrome (typically F12 on Windows/Linux and Cmd+Shift+I on a Mac)
2. Go to the network tab
3. Select the request from the list, right click it and select "Copy as cURL"

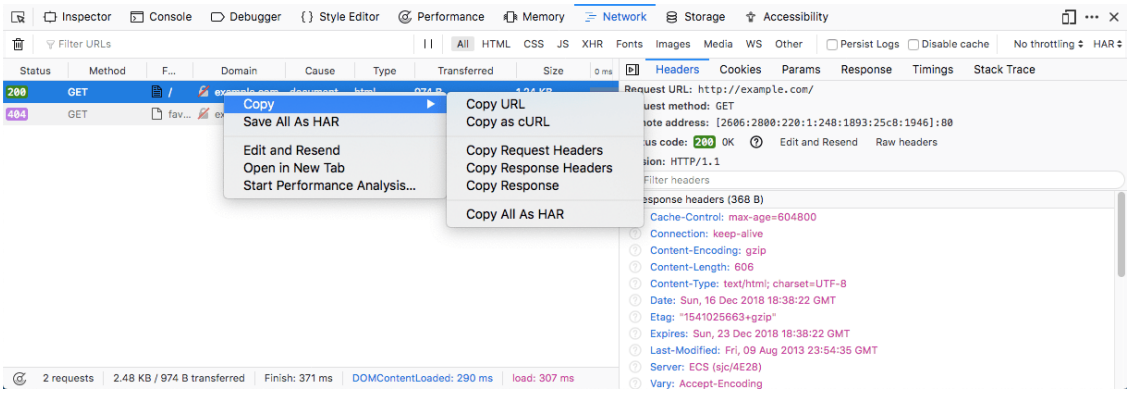

Figure 6

The copied command contains all the headers, request methods, cookies etc. needed to replicate the exact same request. You can paste the command in your terminal to run it.

9 (Optional) Other useful things

- Making authenticated requests:

```
curl -u username:password https://example.com/
```

- Ignoring security certificates:

```
curl -k https://example.com/
```

- Testing if web server supports particular application level protocol

```
curl -v --tlsv1.2 https://example.com/
curl -v --sslv3 https://example.com/
curl -v --http1.0 https://example.com/
curl -v --http1.1 https://example.com/
curl -v --http2 https://example.com/
```

10 (Optional) Using Postman

Postman is a GUI-based client for generating and sending HTTP requests. It is commonly used for testing APIs. Download Postman from `https://www.getpostman.com` and replicate above HTTP requests using Postman.