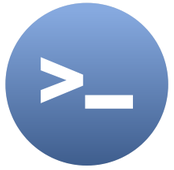
Downloading data using curl

DATA PROCESSING IN SHELL



Susan Sun
Data Person

What is curl?

`curl` :

- is short for **C**lient for **URL**s
- is a Unix command line tool
- transfers data to and from a server
- is used to download data from HTTP(s) sites and FTP servers

Checking curl installation

Check `curl` installation:

```
man curl
```

If `curl` has **not** been installed, you will see:

```
curl command not found.
```

For full instructions, see **https://curl.haxx.se/download.html**.

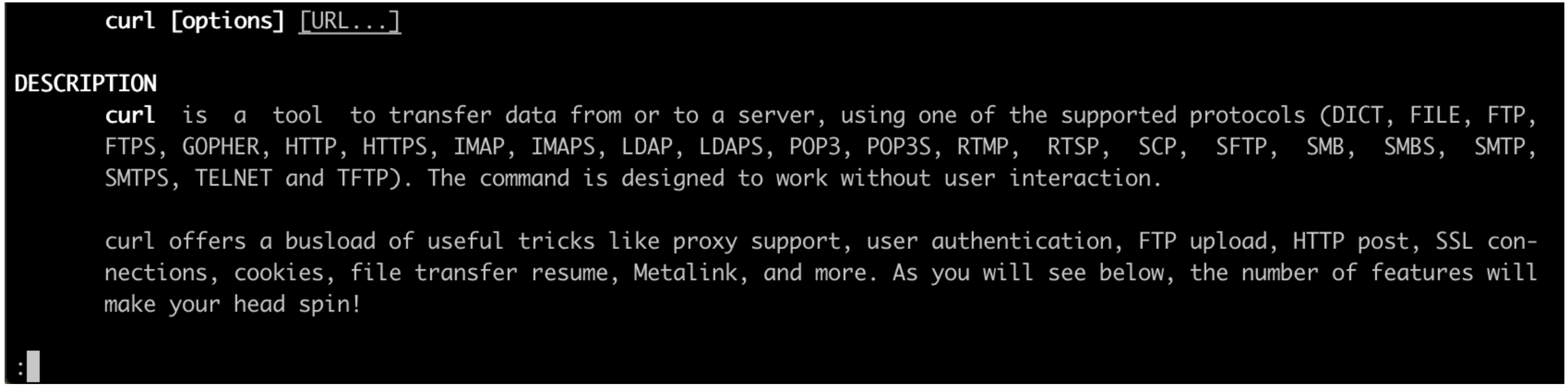
Browsing the curl Manual

If `curl` is installed, your console will look like this:

```
curl(1)                                  Curl Manual                                  curl(1)

NAME
       curl - transfer a URL

SYNOPSIS
       curl [options] [URL...]

DESCRIPTION
       curl  is  a  tool  to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP,
       FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP,  RTSP,  SCP,  SFTP,  SMB,  SMBS,  SMTP,
       SMTPS, TELNET and TFTP). The command is designed to work without user interaction.

       curl offers a busload of useful tricks like proxy support, user authentication, FTP upload, HTTP post, SSL con-
       nections, cookies, file transfer resume, Metalink, and more. As you will see below, the number of features will
:
```

Browsing the curl Manual

Press `Enter` to scroll.



Press `q` to exit.

Learning curl Syntax

Basic `curl` syntax:

```
curl [option flags] [URL]
```

URL is required.

`curl` also supports `HTTP` , `HTTPS` , `FTP` , and `SFTP` .

For a full list of the options available:

```
curl --help
```

Downloading a Single File

Example:

A single file is stored at:

```
https://websitename.com/datafilename.txt
```

Use the optional flag `-O` to save the file with its original name:

```
curl -O https://websitename.com/datafilename.txt
```

To rename the file, use the lower case `-o` + new file name:

```
curl -o renameddatafilename.txt https://websitename.com/datafilename.txt
```

Downloading Multiple Files using Wildcards

Oftentimes, a server will host multiple data files, with similar filenames:

```
https://websitename.com/datafilename001.txt
https://websitename.com/datafilename002.txt
...
https://websitename.com/datafilename100.txt
```

Using Wildcards (*)

Download every file hosted on `https://websitename.com/` that starts with `datafilename` and ends in `.txt` :

```
curl -O https://websitename.com/datafilename*.txt
```

Downloading Multiple Files using Globbing Parser

Continuing with the previous example:

```
https://websitename.com/datafilename001.txt
https://websitename.com/datafilename002.txt
...
https://websitename.com/datafilename100.txt
```

Using Globbing Parser

The following will download every file sequentially starting with `datafilename001.txt` and ending with `datafilename100.txt` .

```
curl -O https://websitename.com/datafilename[001-100].txt
```

Downloading Multiple Files using Globbing Parser

Continuing with the previous example:

```
https://websitename.com/datafilename001.txt
https://websitename.com/datafilename002.txt
...
https://websitename.com/datafilename100.txt
```

Using Globbing Parser

Increment through the files and download every Nth file (e.g. `datafilename010.txt` , `datafilename020.txt` , ... `datafilename100.txt`)

```
curl -O https://websitename.com/datafilename[001-100:10].txt
```

Preemptive Troubleshooting

`curl` has two particularly useful option flags in case of timeouts during download:

- `-L` Redirects the HTTP URL if a 300 error code occurs.
- `-C` Resumes a previous file transfer if it times out before completion.

Putting everything together:

```
curl -L -O -C https://websitename.com/datafilename[001-100].txt
```

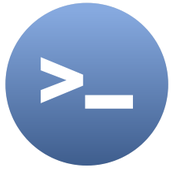
- All option flags come before the URL
- Order of the flags does not matter (e.g. `-L -C -O` is fine)

Happy curl-ing!

DATA PROCESSING IN SHELL

Downloading data using Wget

DATA PROCESSING IN SHELL



Susan Sun
Data Person

What is Wget?

`Wget` :

- derives its name from **World Wide Web** and **get**
- native to Linux but compatible for all operating systems
- used to download data from HTTP(s) and FTP
- better than **curl** at downloading multiple files recursively

Checking Wget Installation

Check if `Wget` is installed correctly:

```
which wget
```

If `Wget` **has** been installed, this will print the location of where `Wget` has been installed:

```
/usr/local/bin/wget
```

If `Wget` **has not** been installed, there will be no output.

Wget Installation by Operating System

Wget source code:

https://www.gnu.org/software/wget/

Linux:

run `sudo apt-get install wget`

MacOS:

use homebrew (https://brew.sh/) and run `brew install wget`

Windows:

download via **gnuwin32** (http://gnuwin32.sourceforge.net/packages/wget.htm)

Browsing the Wget Manual

Once installation is complete, use the `man` command to print the `Wget` manual:

```
WGET(1)                                                    GNU Wget                                                    WGET(1)

NAME
       Wget - The non-interactive network downloader.

SYNOPSIS
       wget [option]... [URL]...

DESCRIPTION
       GNU Wget is a free utility for non-interactive download of files from the Web.  It supports HTTP, HTTPS, and FTP protocols,
       as well as retrieval through HTTP proxies.

       Wget is non-interactive, meaning that it can work in the background, while the user is not logged on.  This allows you to
       start a retrieval and disconnect from the system, letting Wget finish the work.  By contrast, most of the Web browsers
       require constant user's presence, which can be a great hindrance when transferring a lot of data.
```

Learning Wget Syntax

Basic `Wget` syntax:

```
wget [option flags] [URL]
```

URL is required.

`Wget` also supports `HTTP`, `HTTPS`, `FTP`, and `SFTP`.

For a full list of the option flags available, see:

```
wget --help
```

Downloading a Single File

Option flags **unique** to `Wget` :

`-b` : Go to background immediately after startup

`-q` : Turn off the `Wget` output

`-c` : Resume broken download (i.e. continue getting a partially-downloaded file)

```
wget -bqc https://websitename.com/datafilename.txt
```

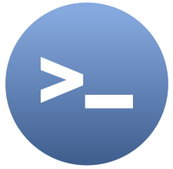
```
Continuing in background, pid 12345.
```

Have fun Wget-ing!

DATA PROCESSING IN SHELL

Advanced downloading using Wget

DATA PROCESSING IN SHELL



Susan Sun
Freelance Person

Multiple file downloading with Wget

Save a list of file locations in a text file.

```
cat url_list.txt
```

```
https://websitename.com/datafilename001.txt
https://websitename.com/datafilename002.txt
...
```

Download from the URL locations stored within the file `url_list.txt` using `-i` .

```
wget -i url_list.txt
```

Setting download constraints for large files

Set upper download bandwidth limit (in KB/s) with `--limit-rate` .

Syntax:

```
wget --limit-rate={rate} {file_location}
```

Example:

```
wget --limit-rate=200k -i url_list.txt
```

Setting download constraints for small files

Set a mandatory pause time (in seconds) between file downloads with `--wait` .

Syntax:

```
wget --wait={seconds} {file_location}
```

Example:

```
wget --wait=2.5 -i url_list.txt
```

curl versus Wget

`curl` advantages:

- Can be used for downloading *and* uploading files from 20+ protocols.
- Easier to install across all operating systems.

`Wget` advantages:

- Has many built-in functionalities for handling multiple file downloads.
- Can handle various file formats for download (e.g. file directory, HTML page).

Let's practice!

DATA PROCESSING IN SHELL