

# Fun with ...



Greg Horie

# What is cURL?

- cURL = Client URL.
- Both an open source software library (libcurl) and command-line tool (curl).
  - First released in 1997.
- Compatible with both IPv4 and IPv6.
- Transfers data through various network protocols.
  - HTTP most commonly used.
- Many excellent features:
  - For HTTP, supports TLS and certificate validation.
  - Supports HTTP methods and header manipulation.
  - Multiple file transfers possible in a single command.
  - Proxy support.
  - Max/min transfer rate settings.
  - Etc.



# cURL Protocol Support

DICT	FILE	FTP
FTPS	GOPHER	HTTP
HTTPS	IMAP	IMAPS
LDAP	POP3	RTMP
RTSP	SCP	SFTP
SMB	SMBS	TELNET
TFTP		

# A Brief History of HTTP - v/0.9

## HTTP = Hypertext Transfer Protocol

- One of the most ubiquitous application protocols on the Internet.
  - Main protocol used in our browsers and for many APIs.

## HTTP/0.9 - 1991

- Very basic client-server, request-response protocol.
- Client request is a single ASCII character string.
- Server response is an ASCII character stream.
- Runs over a TCP/IP connection.
- Designed to transfer hypertext documents (HTML).
- The connection between server and client is closed after every request.



# A Brief History of HTTP - v/1.0

## HTTP/1.0 - 1996



- 1991 -1996
  - Web browsers emerged as the common interface to the Internet.
  - Internet growth boomed during this period.
- In May 1996, the HTTP Working Group (HTTP-WG) published RFC 1945.
  - Extended the list of HTTP methods and HTTP headers.
  - Retained ASCII encoding.
  - Response prefixed with a status line.
  - May transfer other documents in the response - Not limited to HTML.
  - Still adheres to 1 connect / disconnect per request / response.

# A Brief History of HTTP - v/1.1 and v/2.0

## HTTP/1.1 - 1997

- Still in wide use today.
- In Jan 1997, RFC 2068 released for HTTP/1.1.
  - More improvements in June 1999 with RFC 2616.
- Improvements made:
  - Request includes content-type, encoding, character set, and cookie metadata.
  - Performance improvements - Chunked responses and connection reuse.



## HTTP/2.0 - 2012

- Focuses on improving transport performance, low latency, and higher throughput.
- Binary stream rather than text.
- Fully multiplexed - multiple file requests in parallel over a single connection.
- No significant change to protocol semantics - i.e. headers, methods, etc.

# Lab Prep

## Ubuntu 18.04 / 20.04

```
$ sudo apt update  
$ sudo apt install curl
```

## CentOS 7

```
$ sudo yum check-update  
$ sudo yum install curl
```

## Other Operating Systems

<https://curl.haxx.se/download.html>

# Exercise - Basic cURL usage

## Try:

```
$ curl http://vicpimakers.ca
```

```
$ curl http://vicpimakers.ca --location
```

```
$ curl http://vicpimakers.ca --location --head
```

```
$ curl http://vicpimakers.ca --location --head --verbose
```

```
$ curl http://vicpimakers.ca --location --head --verbose \  
--trace debug.out
```



# Basic cURL usage - Discussion

- What is the difference between these 5 invocations of the curl command?
- Why do we have all these redirects?
- How are these different commands useful?

# Exercise - Save URL to File

## Try:

```
$ curl https://vicpimakers.ca -o vicpimakers_ca.html
```

```
$ curl -O https://vicpimakers.ca
```

```
$ curl -O https://vicpimakers.ca/robots.txt
```

# Save URL to File - Discussion

- What did each of these commands do?
- Was there anything wrong? How do we fix it?
- Can we use curl to crawl a site?

- Not without some effort. Instead try:

- ```
$ wget --spider --recursive vicpimakers.ca
```

# Exercise - Explore HTTP Methods

## Try:

```
$ curl --request GET https://vicpimakers.ca
```

```
$ curl --request POST https://vicpimakers.ca
```

```
$ curl --request HEAD https://vicpimakers.ca
```

```
$ curl --request POST \  
  --data 'url=https%3A%2F%2Fvicpimakers.ca' \  
  'https://cleanuri.com/api/v1/shorten'
```

# HTTP Methods - Discussion

- Any problems? What did you find?
- HTTP methods enable actions on a given "resource".
  - Popular for RESTful APIs.

| Method | Typical Action                                                                                                                       |
|--------|--------------------------------------------------------------------------------------------------------------------------------------|
| GET    | Retrieve resource data.                                                                                                              |
| POST   | Submit an entity (e.g. form or record) to the specified resource. Often leads to a change of state (e.g. insert record to database). |
| HEAD   | Asks for response similar to GET, but without the response body.                                                                     |
| PUT    | Replaces the target resource with the request payload.                                                                               |
| DELETE | Deletes the specified resource.                                                                                                      |

# HTTP Methods - Developer Tip

- cURL and RESTful JSON data can be a challenge.
- Consider HTTPie instead for this use case.

```
$ pip install httpie
```

```
$ http -v httpbin.org/post hello=World
```

**VS.**

```
$ curl -v -X POST httpbin.org/post -d '{"hello": "world"}'
```

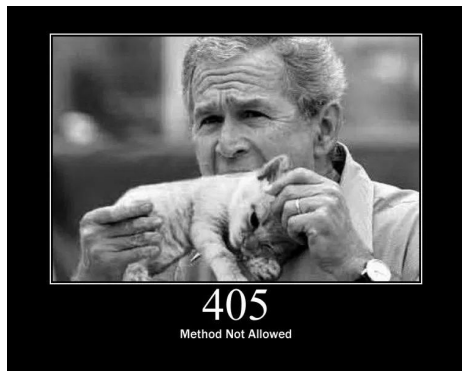
- Or switch to an appropriate language - e.g. python.

# Exercise - Explore Status Codes

## Try:

```
$ curl -Is https://www.twitter.com -L | grep HTTP/
```

```
$ curl -s -o /dev/null -w "%{http_code}" \  
-L https://www.twitter.com
```



<https://boingboing.net/2011/12/14/http-status-cats-by-girliemac.html>

# Exercise - Discussion

- Any questions?

| Status Codes | Meaning                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------|
| 1xx          | Informational Response.<br>Request received and understood. Request processing continues.                 |
| 2xx          | Success. Requested action was successful.<br>e.g. 200 - OK.                                               |
| 3xx          | Redirection. Client must take further action to complete request.<br>e.g. 301 - Moved Permanently.        |
| 4xx          | Client errors. Request contains syntax errors or cannot be fulfilled.<br>e.g. 404 - Not Found.            |
| 5xx          | Server errors. Server encountered an error - cannot fulfill request.<br>e.g. 500 - Internal Server Error. |



# Exercise - Explore Request Headers

## Try:

```
$ curl http://nghttp2.org -I -L -v
```

```
$ curl http://nghttp2.org -I -L -v --http2
```

# Headers - Discussion

- What did you find?
- You can further manipulate request headers with the curl -H (--header) option.
  - Not to be mistaken with the curl -I (--head) option.
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

# Exercise - FTP

## Try:

```
$ head -c 5MB /dev/urandom > randofile.bin
```

```
$ curl --user anonymous:anonymous \  
      --upload-file randofile.bin \  
      ftp://speedtest.tele2.net/upload/
```

```
$ curl --user anonymous:anonymous \  
      ftp://speedtest.tele2.net/
```

```
$ curl --user anonymous:anonymous \  
      -O ftp://speedtest.tele2.net/5MB.zip
```

# Exercise - Fun with Dict

## Try:

```
$ curl dict://dict.org/d:curl
```

**# Maybe something for your .bashrc**

```
$ alias dict='function _dict(){ curl "dict://dict.org/d:${1}"; }; _dict'  
$ dict curl
```

# Exercise - Fun Sites with cURL

## Try:

```
$ curl wttr.in
```

```
$ alias wttr='function _wttr(){ curl "wttr.in/${1}"; }; _wttr'
```

```
$ wttr
```

```
$ wttr toronto
```

```
$ wttr moon
```

```
$ curl grench.de/https://vicpimakers.ca
```

```
$ curl https://asciitv.fr
```

**# So much fun!**

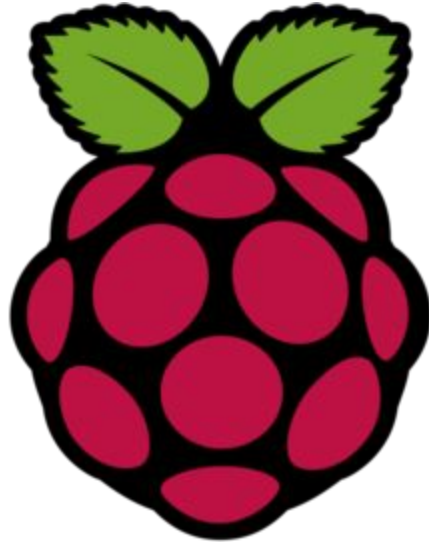
# Summary

- cURL is an excellent tool for CLI interactions with Internet resources.
- Commonly used with HTTP, but it supports many other protocols.
- If you're not using it, consider adding cURL to your toolbox.



# VicPiMakers Communications

- Please let us know if want to be in our VicPiMakers Slack group and/or mailing list.



# Backup Slides

