curling the modern way

[Broom not included]
Daniel Stenberg
@bagder
@mastodon.social
https://daniel.haxx.se
### 101 operating systems

<table>
<thead>
<tr>
<th>AIX</th>
<th>AmigaOS</th>
<th>Android</th>
<th>ArcaOS</th>
<th>Aros</th>
<th>Atari FreeMiNT</th>
<th>BeOS</th>
<th>Blackberry 10</th>
<th>Blackberry Tablet OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell OS</td>
<td>CheriBSD</td>
<td>Chrome OS</td>
<td>Cisco IOS</td>
<td>DG/UX</td>
<td>DragonFly BSD</td>
<td>DR DOS</td>
<td>eCos</td>
<td>FreeDOS</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>FreeRTOS</td>
<td>Fuchsia</td>
<td>Garmin OS</td>
<td>Genode</td>
<td>Haiku</td>
<td>HardenedBSD</td>
<td>HP-UX</td>
<td>Hurd</td>
</tr>
<tr>
<td>Illumos</td>
<td>Integrity</td>
<td>iOS</td>
<td>ipadOS</td>
<td>IRIX</td>
<td>Linux</td>
<td>Lua RTOS</td>
<td>Mac OS 9</td>
<td>macOS</td>
</tr>
<tr>
<td>Maemo</td>
<td>Mbed</td>
<td>Meego</td>
<td>Micrium</td>
<td>MINIX</td>
<td>Moblin</td>
<td>MorphOS</td>
<td>MPE/iX</td>
<td>MS-DOS</td>
</tr>
<tr>
<td>NCR MP-RAS</td>
<td>NetBSD</td>
<td>NetWare</td>
<td>NextStep</td>
<td>Nintendo Switch</td>
<td>NonStop OS</td>
<td>NuttX</td>
<td>OpenBSD</td>
<td>OpenStep</td>
</tr>
<tr>
<td>Orbis OS</td>
<td>OS/2</td>
<td>OS/400</td>
<td>OS21</td>
<td>Plan 9</td>
<td>PlayStation Portable</td>
<td>QNX</td>
<td>Qubes OS</td>
<td>ReactOS</td>
</tr>
<tr>
<td>Redox</td>
<td>RISC OS</td>
<td>ROS</td>
<td>RTEMS</td>
<td>Sailfish OS</td>
<td>SCO Unix</td>
<td>Serenity</td>
<td>SINIX-Z</td>
<td>SkyOS</td>
</tr>
<tr>
<td>Solaris</td>
<td>Sortix</td>
<td>SunOS</td>
<td>Syllable OS</td>
<td>Symbian</td>
<td>Tizen</td>
<td>TPF</td>
<td>Tru64</td>
<td>tvOS</td>
</tr>
<tr>
<td>ucLinux</td>
<td>Ultrix</td>
<td>UNICOS</td>
<td>UnixWare</td>
<td>VMS</td>
<td>vxWorks</td>
<td>watchOS</td>
<td>Wear OS</td>
<td>WebOS</td>
</tr>
<tr>
<td>Wii System Software</td>
<td>Wii U</td>
<td>Windows</td>
<td>Windows CE</td>
<td>Xbox System</td>
<td>Xenix</td>
<td>Zephyr</td>
<td>z/OS</td>
<td>z/TPF</td>
</tr>
<tr>
<td>z/VM</td>
<td>z/VSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Operating systems known to have run curl*
## 28 CPU architectures

<table>
<thead>
<tr>
<th></th>
<th>Alpha</th>
<th>ARC</th>
<th>ARM</th>
<th>AVR32</th>
<th>C-SKY</th>
<th>CompactRISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elbrus</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m88k</td>
<td>MicroBlaze</td>
<td></td>
<td>MIPS</td>
<td>Nios</td>
<td>LoongArch</td>
<td>m68k</td>
</tr>
<tr>
<td>PowerPC</td>
<td>RISC-V</td>
<td></td>
<td>s390</td>
<td>SH4</td>
<td>SPARC</td>
<td></td>
</tr>
<tr>
<td>Tilera</td>
<td>VAX</td>
<td></td>
<td>x86</td>
<td>Xtensa</td>
<td>z/arch</td>
<td></td>
</tr>
</tbody>
</table>

CPU architectures known to have run curl
curl runs in all your devices
2 planets

Planets known to have run curl
use a recent curl version
Parallel transfers

by default, URLs are transferred serially, one by one
-Z(--)--parallel
By default up to 50 simultaneous
Change with --parallel-max [num]
Works for downloads and uploads

$ curl -Z -O https://example.com/[1-1000].jpg
--no-clobber

does not overwrite destination, adds a number instead

```bash
$ touch file
$ curl -O --no-clobber https://example.com/file
$ ls -l file.1
-rw-r--r-- 1 user user 1256 Oct 18 13:27 file.1
```
--remove-on-error

do not save leftovers on error

```bash
$ curl -O --remove-on-error --max-time 2 https://example.com/file
$ curl: (28) Operation timed out after 2000 ...
$ ls -l file
ls: cannot access 'file': No such file or directory
```
transfer controls

stop slow transfers
--speed-limit <speed> --speed-time <seconds>

transfer rate limiting

curl --limit-rate 100K https://example.com
	no more than this number of transfer starts per time unit

curl --rate 2/s https://example.com/[1-20].jpg

curl --rate 3/h https://example.com/[1-20].html

curl --rate 14/m https://example.com/day/[1-365]/fun.html

Truly limit the maximum file size accepted:

curl --maxfilesize 238M https://example.com/the-biggie -O
config file

“command lines in a file”
one option (plus argument) per line
$HOME/.curlrc is used by default
-K [file] or --config [file]
can be read from stdin
can be generated (and huge)
10MB line length limit
Variables in a config file

\texttt{user = "$USER:\$SECRET"} - cannot could not be done introducing this syntax risks breaking countless existing files what if you would rather read that info from a separate file?
--variable

Sets a curl variable on command line or in a config file
--variable name=content
variable = name=content

There can be an unlimited amount of variables.
A variable can hold up to 10M of content
Variables are set in a left to right order as it parses the command line or config file.
Setting variables

Content for the variable can be read from a file:
--variable name@file

Content for the variable can be read from stdin:
--variable name@-

Variable content may be binary
A variable can be imported from the environment. Error if non-existing:
--variable %name

Import a name from the environment, but if not set use a default value:
--variable %name=default

Import a name from the environment, but if not set read the default value from a file:
--variable %name@filename
Expanding variables

Variables can be used in command line option arguments
Must be explicitly asked for
Introducing the `--expand-` prefix
Reference a variable as `{{name}}`
A non-existing variable will expand as blank/nothing
Trying to show a variable with a null byte causes error
Examples:

`--expand-data "{{content}}"`

`--expand-url "https://{{host}}/user/{{user}}"`
Expansion functions

When expanding a variable, *functions* can be applied

`{{name:function}}`

Functions alter how the variable is expanded

Multiple functions can be applied in a left-to-right order:

`{{name:func1:func2:func3}}`

Provided functions:
trim, json, url and b64

```bash
$ curl --variable %DATA --expand-data '{{DATA:trim:b64}}'
https://example.com/
```
parses and manipulates URLs like ‘tr’ but for URLs

**companion tool** to curl

$ trurl --url https://curl.se --set host=example.com

$ trurl --url https://curl.se --get '{host}'

$ trurl --url https://curl.se/we/are.html --redirect here.html.html

$ trurl --url https://curl.se/we/../are.html --set port=8080

$ trurl "https://curl.se?name=hello" --append query=search=string

$ trurl "https://fake.host/search?q=answers&user=me#frag" --json

$ trurl "https://example.com?a=home&here=now&thisthen" -g '{query:a}'

[https://curl.se/trurl/](https://curl.se/trurl/)
outputs text, information and HTTP headers after a transfer is completed

curl -w "formatted string" http://example.com/
curl -w @filename http://example.com/
curl -w @- http://example.com/

Information from over 50 “variables”

curl -w "Type: %{content_type}\nCode: %{response_code}\n" ...

Show all the information as JSON:

curl -w ‘%{json}’ ...
--write-out

Show HTTP response header contents

```bash
curl -w "Server: %header{server}\nDate: %header{date}\n" ...
```

Show all HTTP response headers as JSON

```bash
curl -w "%{header_json}\n" ...
```
**JSON + JSON + JSON**

```bash
curl --json '{"name": "daniel"}' https://example.com
curl --json @object.json https://example.com
```

Sets **Content-Type: application/json** and **Accept: application/json**

**Create JSON easily**

```
jo -p name=jo n=17 parser=false | curl --json @- https://example.com/
```

**Receive/parson JSON easily**

```bash
curl --json '{"tool": "curl"}' https://example.com/ | jq
```

```bash
curl + jo + jq
jo -p name=jo n=17 | curl --json @- https://example.com/ | jq
```
$ curl -d moo --trace - https://curl.se/
  ∼ Info: processing: https://curl.se/
  ∼ Info:  Connect
  ∼ Info:  ALPN: of
  → Send SSL data, 0000: 16 03 01 02
  → Info:  TLSv1.3
  → Send SSL data, 0010: 92 0a 14 18 87 2b ec 23 19 ab
  → Info: Server certificate:
  → Info: ALPN: server accepted h2
  → Info: SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
  → Info:   Trying [2a04:4e42:e00::347]:443...
  → Info: processing: https://curl.se/
  $ curl -d moo --trace - https://curl.se/
  trace
  200
  => Send SSL data, 122 bytes (0x7a)
  => Recv SSL data, 5 bytes (0x5)
  => Send SSL data, 5 bytes (0x5)
  => Recv SSL data, 40 bytes (0x28)
  => Recv SSL data, 27 bytes (0x1b)
  => Recv SSL data, 22 bytes (0x16)
  => Recv SSL data, 3 bytes (0x3)
  => Send SSL data, 1 bytes (0x1)
  …
  …
  …
  …
  …
trace even more

--trace-config tells curl what more to include in the trace

“all”, “HTTP/2”, “HTTP/3”, “TLS”, …

This was previously only possible in debug builds

$ curl --trace dump --trace-config all https://example.com/one
curl hides protocol differences from users

HTTP/3 is **only for HTTPS**, there is no clear text version

```
--http3
```

```
--http3 races HTTP/3 against HTTP/1+2 and picks the winner
```

With HTTP/3, curl can do multiplexed transfers with `-Z`
You can help!
Thank you!

Questions?

Daniel Stenberg
@bagder@mastodon.social
https://daniel.haxx.se/