curling the modern way

[Broom not included]

February 3, 2023

Daniel Stenberg

Daniel Stenberg

@bagder
@mastodon.social

wolfSSL

https://daniel.haxx.se


@bagder
curl://

@bagder

101 operating systems

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

Operating systems known to have run curl

@bagder

28 CPU architectures

Alpha	ARC	ARM	AVR32	C-SKY	CompactRISC
Elbrus	ETRAX	HP-PA	Itanium	LoongArch	m68k
m88k	MicroBlaze	MIPS	Nios	OpenRISC	POWER
PowerPC	RISC-V	s390	SH4	SPARC	
Tilera	VAX	x86	Xtensa	z/arch	

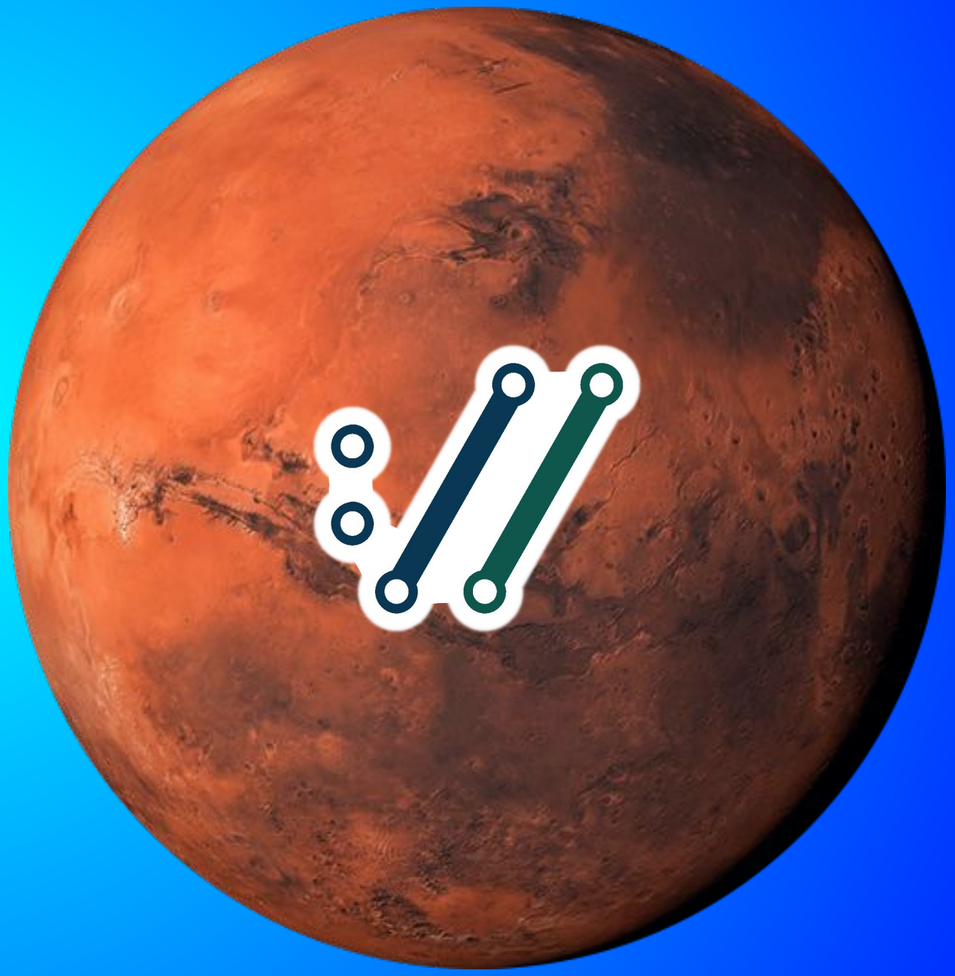
CPU architectures known to have run curl


@bagder
curl runs in all your devices
Windows 10
iOS
FORTNITE
chromeOS
NETFLIX
PLAYERUNKNOWN'S
SPIDER-MAN
ROKU
Roku Ultra

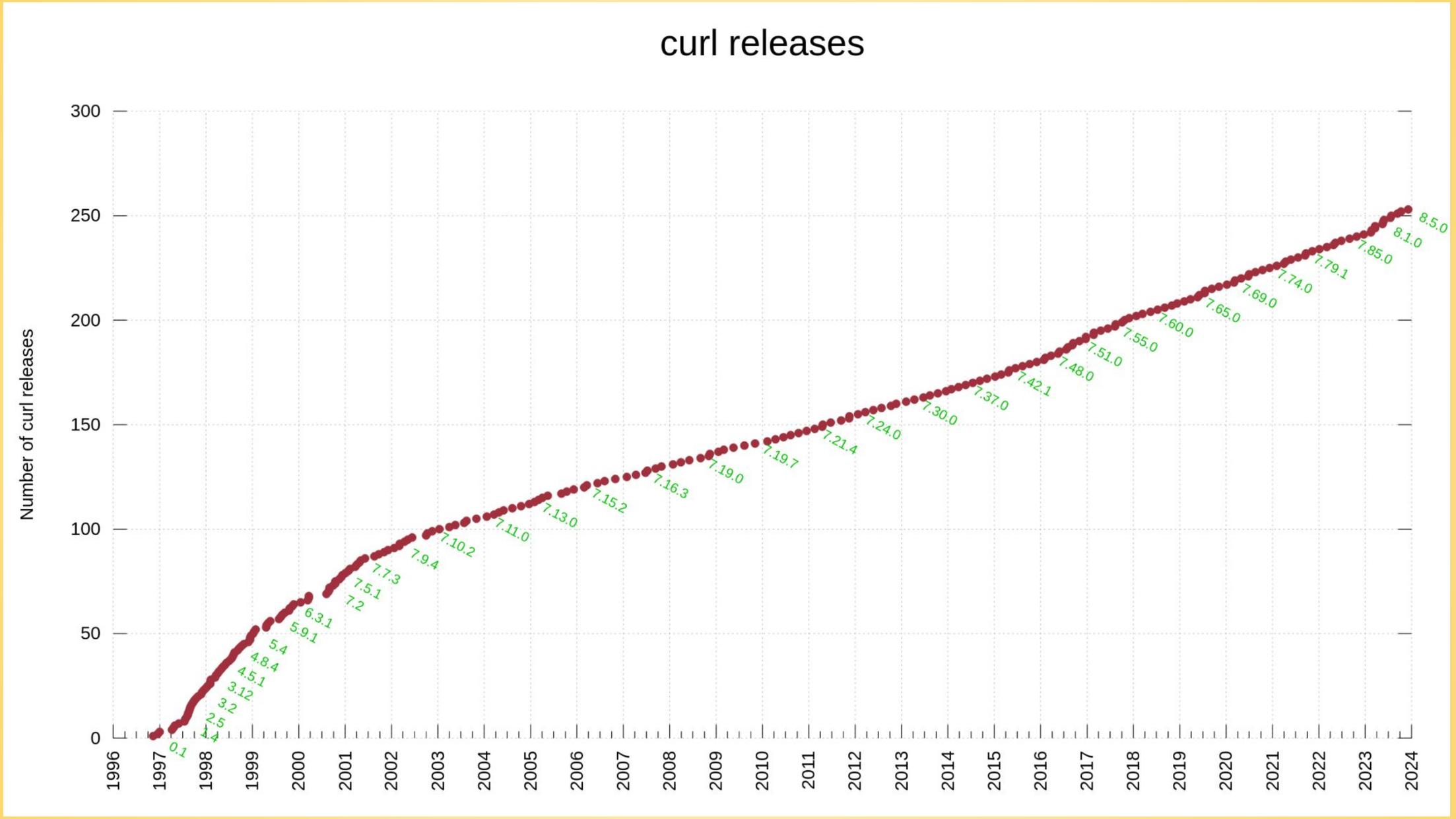

@bagder

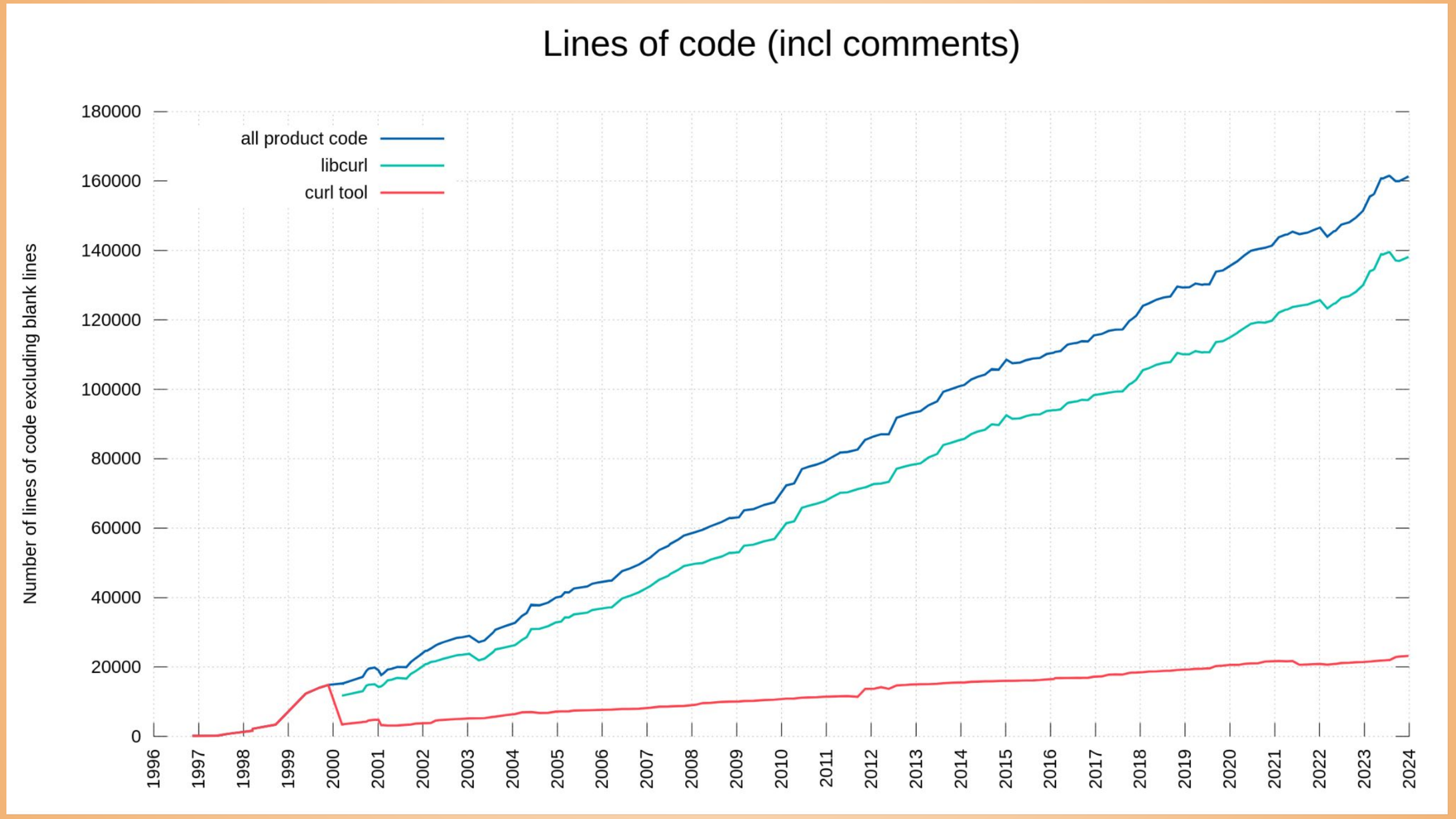
@bagder

2 planets

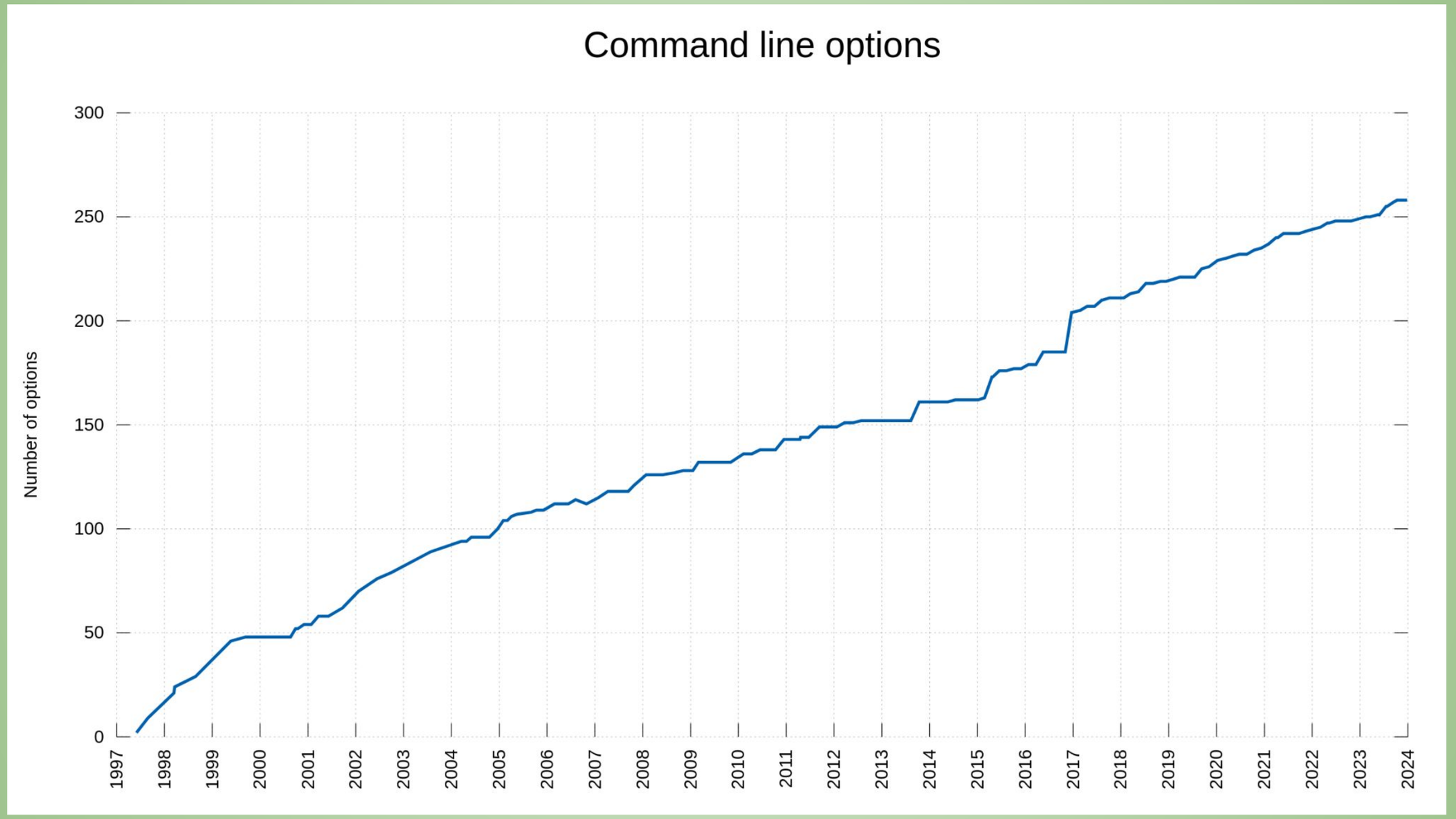


Planets known to have run curl


curl releases
Number of curl releases
300
250
200
150
100
50
0
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
0.1
1.4
2.5
3.2
3.12
4.5.1
4.8.4
5.4
5.9.1
6.3.1
7.2
7.5.1
7.7.3
7.9.4
7.10.2
7.11.0
7.13.0
7.15.2
7.16.3
7.19.0
7.19.7
7.21.4
7.24.0
7.30.0
7.37.0
7.42.1
7.48.0
7.51.0
7.55.0
7.60.0
7.65.0
7.69.0
7.74.0
7.79.1
7.85.0
8.1.0
8.5.0


Lines of code (incl comments)
all product code
libcurl
curl tool
Number of lines of code excluding blank lines
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

@bagder


Command line options
Number of options
0
50
100
150
200
250
300
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024


use a recent curl version

@bagder

Parallel transfers

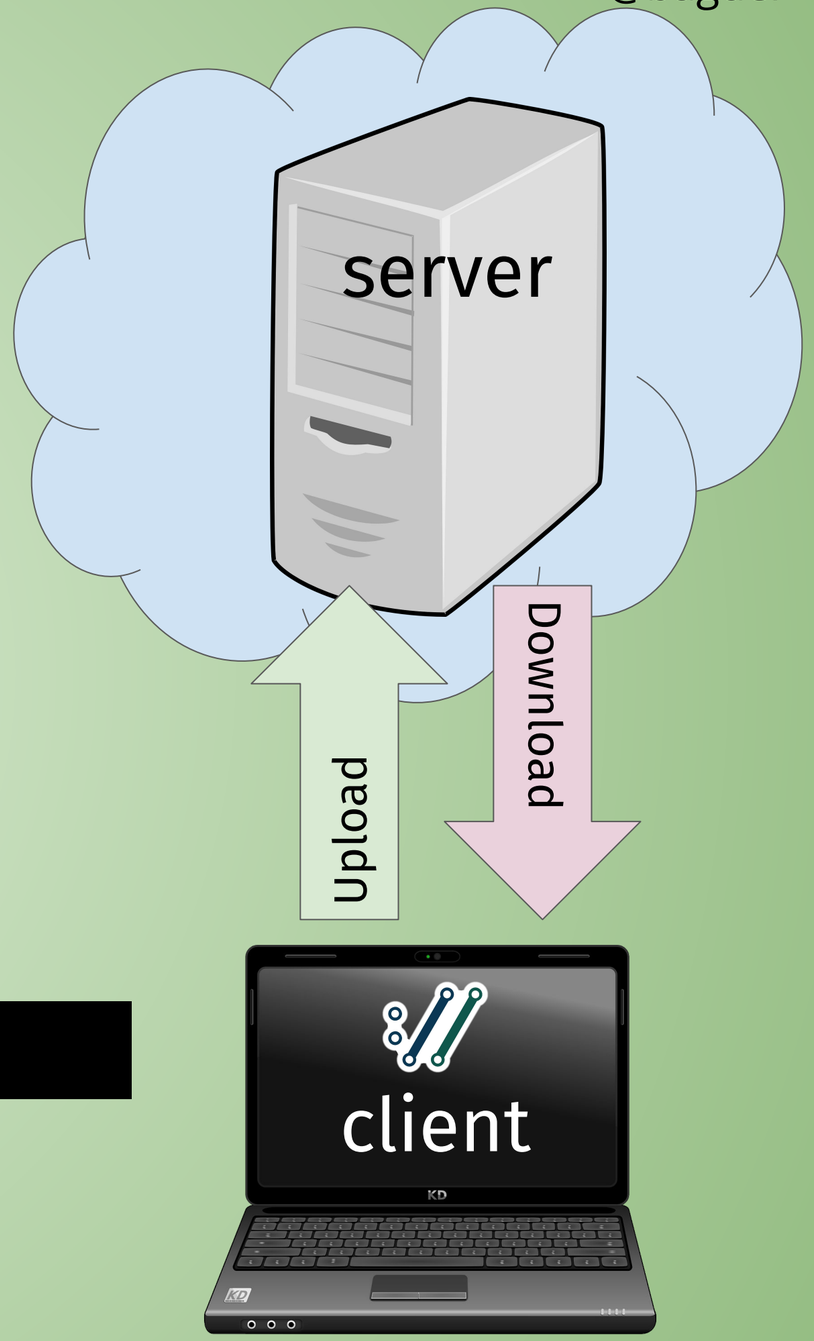
by default, URLs are transferred serially, one by one

`-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

```
$ 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

```
$ 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 --max-filesize 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

`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

Environment variables

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}}”
```

Variable content may be binary

Expansion functions

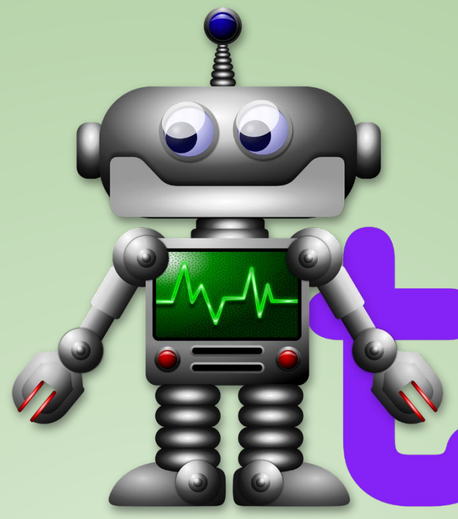
When expanding a variable, *functions* can be applied
`{{name:function}}`

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

```
$ curl --variable %DATA --expand-data '{{DATA:trim:b64}}'
https://example.com/
```



trurl

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


@bagder
JSON

--write-out

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

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

Show **all** HTTP response headers as JSON

```
curl -w "%{header_json}\n" …
```

JSON + JSON + JSON

```
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

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

curl + jo + jq

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

trace

```
curl -d moo --trace - https://curl.se/
```

```
$ curl -d moo --trace - https://curl.se/
== Info: processing: https://curl.se/
== Info:   Trying [2a04:4e42:e00::347]:443
== Info: Connecte
== Info: ALPN: of
=> Send SSL data,
0000: 16 03 01 02
== Info: TLSv1.3
=> Send SSL data,
0000: 01 00 01 fc
0010: 92 0a 4e bf
0020: 92 96 9a a1
…
== Info:  CAfile:
== Info:  CApath:
<= Recv SSL data,
0000: 16 03 03 00
== Info: TLSv1.3
<= Recv SSL data,
0000: 02 00 00 76
0010: 30 a0 f5 e0
…
== Info: SSL conn
== Info: ALPN: se
== Info: Server c
== Info:  subject
== Info:  start d
== Info:  expire
== Info:  subject
== Info:  issuer:
== Info:  SSL cer
…
```

```
== Info: using HTTP/2
== Info: h2 [:method: PO
== Info: h2 [:scheme: ht
== Info: h2 [:authority:
== Info: h2 [:path: /]
== Info: h2 [user-agent:
== Info: h2 [accept: */*
== Info: h2 [content-len
== Info: h2 [content-typ
== Info: Using Stream ID
=> Send SSL data, 5 byte
0000: 17 03 03 00 5d
=> Send SSL data, 1 byte
0000: 17
=> Send header, 137 byte
0000: 50 4f 53 54 20 2f
0010: 6f 73 74 3a 20 63
0020: 65 72 2d 41 67 65
0030: 2e 32 2e 31 0d 0a
0040: 2a 0d 0a 43 6f 6e
0050: 68 3a 20 33 0d 0a
0060: 70 65 3a 20 61 70
0070: 78 2d 77 77 77 2d
0080: 63 6f 64 65 64 0d
=> Send data, 3 bytes (0
0000: 6d 6f 6f
```

```
…
<= Recv header, 13 bytes (0x0d)
0000: 48 54 54 50 2f 32 20 32 30 30 20 0d 0a         HTTP/2 200 ..
<= Recv header, 22 bytes (0x16)
0000: 73 65 72 76 65 72 3a 20 6e 67 69 6e 78 2f 31 2e server: nginx/1.
0010: 32 31 2e 31 0d 0a                              21.1..
<= Recv header, 25 bytes (0x19)
0000: 63 6f 6e 74
0010: 78 74 2f 68
<= Recv header, 29
0000: 78 2d 66 72
0010: 20 53 41 4d
<= Recv header, 46
0000: 6c 61 73 74
0010: 68 75 2c 20
0020: 30 32 3a 30
<= Recv header, 28
0000: 65 74 61 67
0010: 38 30 36 33
<= Recv header, 22
0000: 61 63 63 65
0010: 79 74 65 73
<= Recv header, 27
0000: 63 61 63 68
0010: 61 78 2d 61
<= Recv header, 40
0000: 65 78 70 69
0010: 20 41 75 67
0020: 31 34 20 47
…
```

```
…
<= Recv data, 867 bytes (0x363)
0000: 3c 21 44 4f 43 54 59 50 45 20 48 54 4d 4c 20 50 <!DOCTYPE HTML P
0010: 55 42 4c 49 43 20 22 2d 2f 2f 57 33 43 2f 2f 44 UBLIC "-//W3C//D
0020: 54 44 20 48 54 4d 4c 20 34 2e 30 31 20 54 72 61 TD HTML 4.01 Tra
0030: 6e 73 69 74 69 6f 6e 61 6c 2f 2f 45 4e 22 20 22 nsitional//EN" "
0040: 68 74 74 70 3a 2f 2f 77 77 77 2e 77 33 2e 6f 72 http://www.w3.or
0050: 67 2f 54 52 2f 68 74 6d 6c 34 2f 6c 6f 6f 73 65 g/TR/html4/loose
0060: 2e 64 74 64 22 3e 0a 3c 68 74 6d 6c 20 6c 61 6e .dtd">.<html lan
0070: 67 3d 22 65 6e 22 3e 0a 3c 68 65 61 64 3e 0a 3c g="en">.<head>.<
0080: 74 69 74 6c 65 3e 63 75 72 6c 3c 2f 74 69 74 6c title>curl</titl
0090: 65 3e 0a 3c 6d 65 74 61 20 6e 61 6d 65 3d 22 76 e>.<meta name="v
00a0: 69 65 77 70 6f 72 74 22 20 63 6f 6e 74 65 6e 74 iewport" content
00b0: 3d 22 77 69 64 74 68 3d 64 65 76 69 63 65 2d 77 ="width=device-w
00c0: 69 64 74 68 2c 20 69 6e 69 74 69 61 6c 2d 73 63 idth, initial-sc
00d0: 61 6c 65 3d 31 2e 30 22 3e 0a 3c 6d 65 74 61 20 ale=1.0">.<meta
00e0: 63 6f 6e 74 65 6e 74 3d 22 74 65 78 74 2f 68 74 content="text/ht
00f0: 6d 6c 3b 20 63 68 61 72 73 65 74 3d 55 54 46 2d ml; charset=UTF-
0100: 38 22 20 68 74 74 70 2d 65 71 75 69 76 3d 22 43 8" http-equiv="C
0110: 6f 6e 74 65 6e 74 2d 54 79 70 65 22 3e 0a 3c 6c ontent-Type">.<l
0120: 69 6e 6b 20 72 65 6c 3d 22 73 74 79 6c 65 73 68 ink rel="stylesh
```

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
```

HTTP/3

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`

@bagder

https://everything.curl.dev/


@bagder
You can help!
curl

Thank you!

Questions?

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