curl user survey analysis 2024

"As strange as it may sound it is hard to contribute to such a polished piece of software."

summary and analysis by Daniel Stenberg

Jun 17, 2024

About curl

Curl is an established and mature open source project that produces the curl tool and the libcurl library. While it is a small project, with just a few maintainers, its products run in several billions of Internet connected devices, applications, tools, games and services. curl is no doubt one of the world's most widely used software components. curl was first released in March 1998, building on its predecessors originating back to November 1996.

I believe a key to remaining relevant and to maintain a relevant place in people's toolboxes, curl as a project must keep up. Keep up with what users want, with how internet transfers are done, with Internet standards and with protocol development.

Survey Background

We run a curl user survey annually in an attempt to catch trends, views and long term changes in the project, its users, its surrounding and in how curl fits into the wider ecosystem. This year, the survey was up 14 days **from May 14 to and including May 28**. This was the 11th annual survey.

The survey was announced on the curl-users and curl-library mailing lists (with one reminders), numerous times on Daniel's Mastodon (@bagder@mastodon.social) on LinkedIn and on Daniel's blog (https://daniel.haxx.se/blog). The survey was also announced on the curl web site at the top of most pages on the site that made it hard to miss for visitors.

Survey Bias

We only reach and get responses from a small subset of users who voluntarily decide to fill in the questionnaire while the vast majority of users and curl developers never get to hear about it and never get an opportunity to respond. Self-selected respondents to a survey makes the results hard to interpret and judge. This should make us ask ourselves: is this what our users think, or is it just the opinions of the tiny subset of users that we happened to reach this year. We simply have to work with what we have.

Many statements listed in this document are verbatim quotes from respondents. We in the curl project do not necessarily agree with those or think the same way. Some might even be downright offensive. Beware.

Survey Stability

For several years we have witnessed how the responses to the surveys are strikingly similar year-to-year even while the majority of the people who respond say they did not answer the survey last year. It might imply that the responses are indicative for a wider population.

Hosted by Google

We use a service run by Google to perform the survey, which leads to us losing the share of users who refuse to use services hosted by them. We feel compelled to go with simplicity, no cost and convenience of the service rather than trying to please everyone. We have simply not put in the effort to switch to an alternative provider for the survey.

Ten quick things to take away

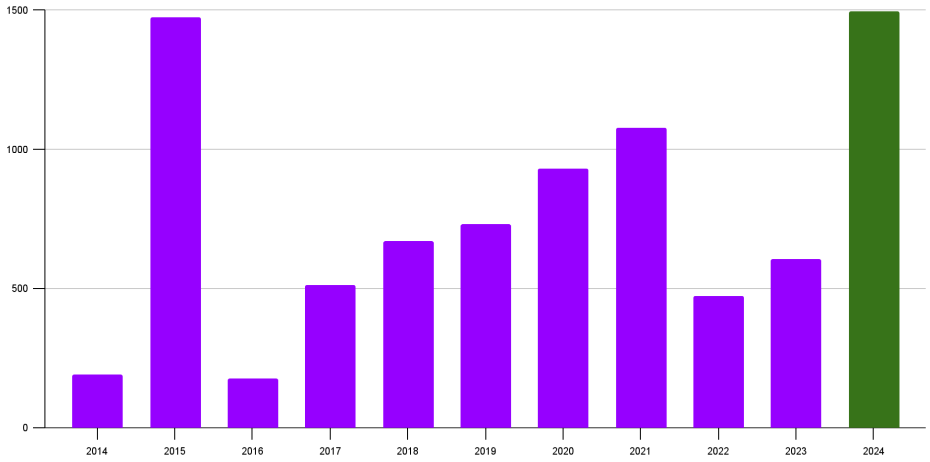
If you are in too much of a hurry to read it all, here are ten facts this year's survey revealed:

1. 96.4% of the users run it on Linux (*)
2. 98.6% use it for HTTPS (*)
3. 98% of Windows users run curl on x86 64 bit architecture (*)
4. 99% of the users who know, use curl with OpenSSL (*)
5. curl users run it on Android more than on FreeBSD
6. Windows 10 is the most used Windows version for curl use
7. More than 100 different command line options is a favorite for at least one
8. 83.1% rates our "security handling" 5 out of 5
9. 22.8% of users would like to see it offer recursive HTML download
10. 14.1% of users have used curl for 18 or more years

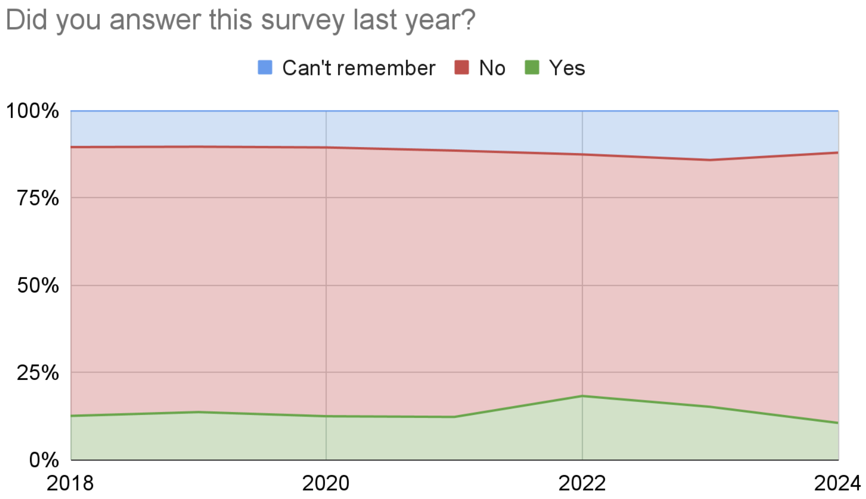
(*) = not exclusively - as these questions allowed respondents to select multiple answers the total ends up larger than 100%.

1. Survey responses

This was a record-breaking year in terms of participation in the survey and we *almost* reached 1500 this time (1496 to be exact) - 21 more responses than the previous record 1475 from 2015. Up 247% from last year's 606 responses. It is not easy to understand nor affect these fluctuations, but we are of course happy that we managed to reach out. A wider feedback is better.

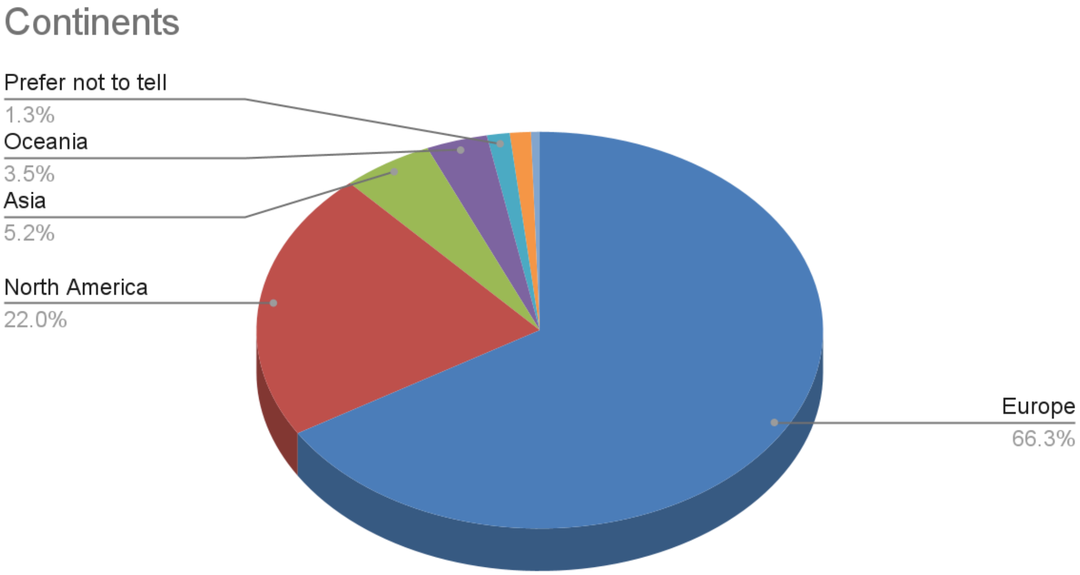


10.5% (the median rate from the last 7 years is at 12.0%) say they answered the survey last year. It is no surprise that this rate shrunk a bit when the number of responses took off this much. It still means that (at least) 26% of last year's respondents came back and answered this year. The "I can't remember" share was even larger at 12.1%.

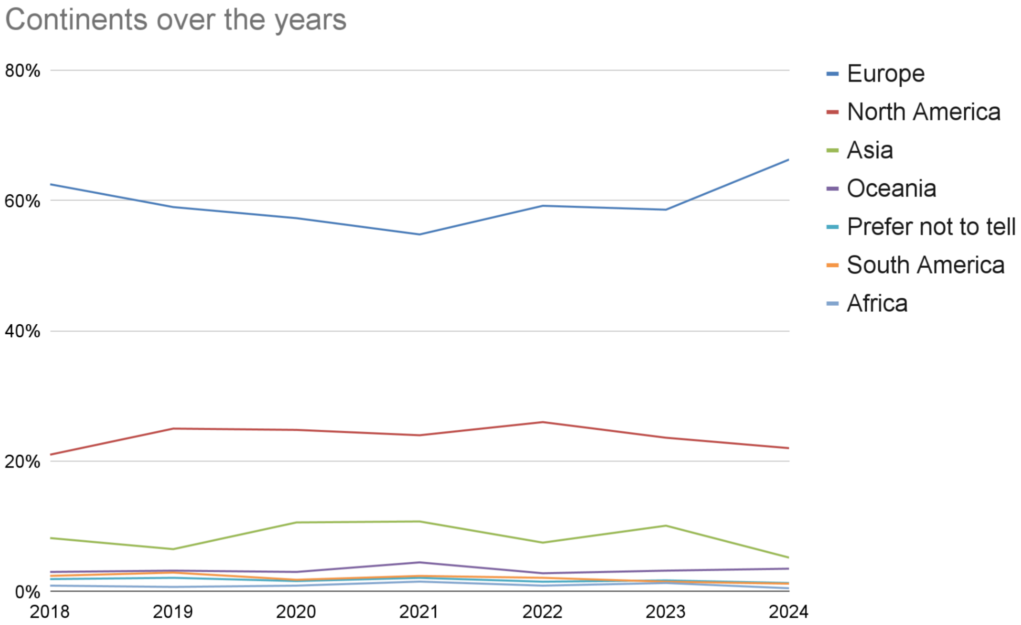


2. Continents

The users who answered the survey remain European to a large degree (66.3%). North America is 2nd as usual, at 22.0% this year. Europe is growing and North America is shrinking.

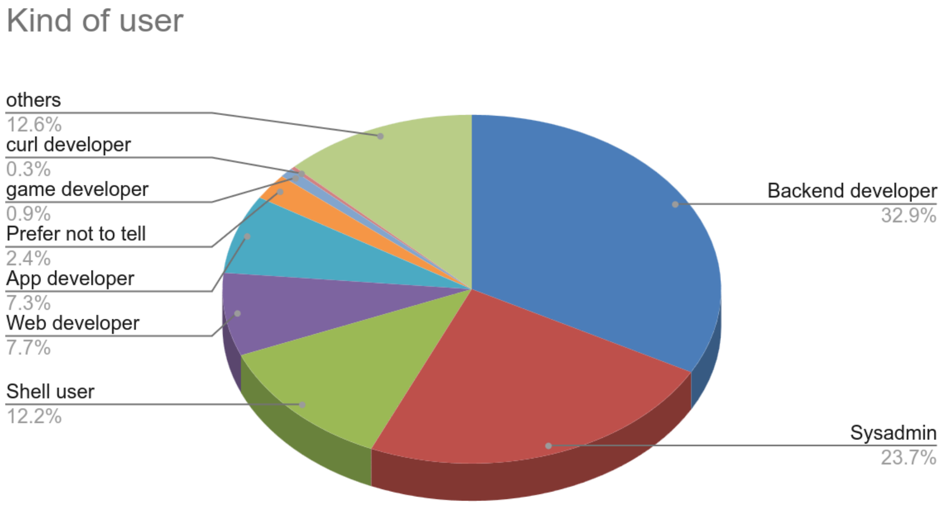


I have no good explanations why Europe remains so dominant among the respondents in the project - at a sustained rate year after year. It could be noticed that among the top contributors to curl, Europeans are in a majority as well.



3. Kind of users

The label most commonly applicable for the users who filled in the survey remains “backend developer”. Labels are tricky, and that is presumably why so many select “other”.



This “kind of user” distribution has remained almost identical in the surveys over the years.

4. Protocols

Asking users what protocol they have used with curl is a blunt tool. The people who respond to the survey are self-selected and we cannot tell if they answer factually. Maybe they confuse two protocols. Maybe they forget how long ago they actually used one of the protocols.

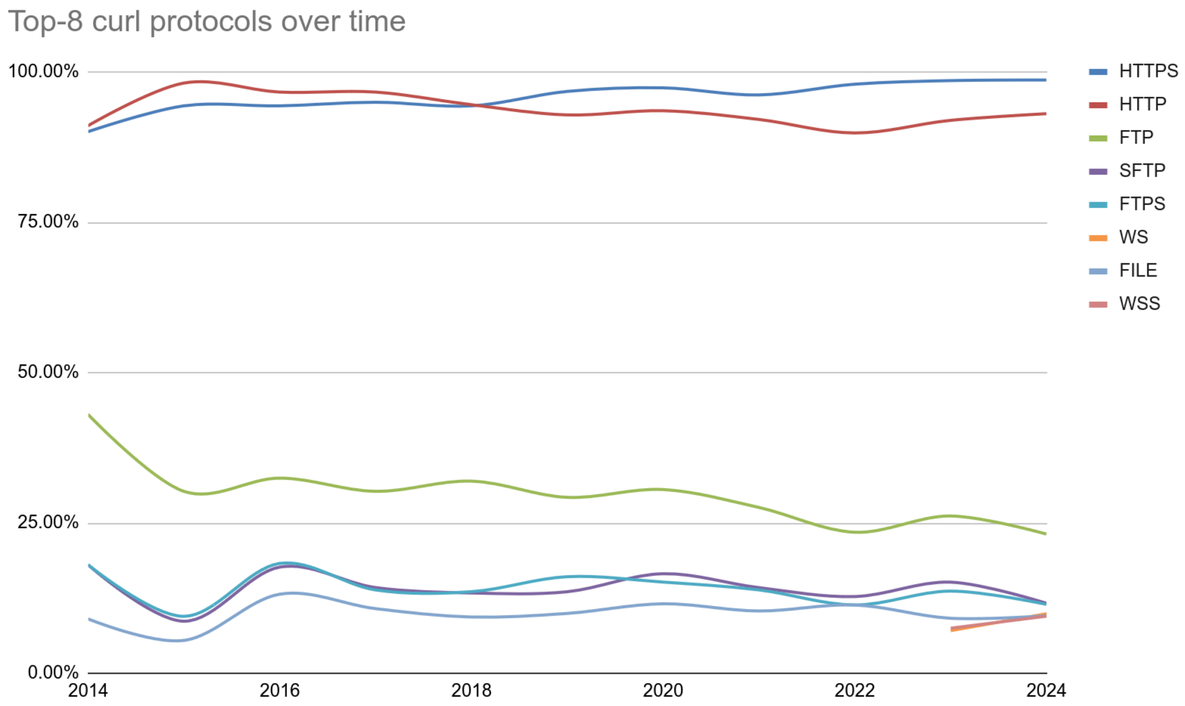
We speak of protocols as in different URL schemes curl supports. In spring of 2024, curl supports 28 different protocols where two of them (WS and WSS) are still labeled experimental, meaning that they need to be enabled explicitly at build time and therefore are not as accessible to all users as the others are. curl supported the same set of protocols last year.

1	**HTTPS**	98.60%
2	**HTTP**	93.00%

3	**FTP**	23.10%
4	**SFTP**	11.60%
5	**FTPS**	11.40%
6	**WS**	9.90%
7	**FILE**	9.50%
8	**WSS**	9.50%
9	**SCP**	6.50%
10	**SMTP**	5.50%
11	**IMAP**	4.60%
12	**LDAP**	4.60%
13	**IMAPS**	4.50%
14	**MQTT**	4.10%
15	**TFTP**	4.00%
16	**TELNET**	3.90%
17	**LDAPS**	3.80%
18	**SMTPS**	3.70%
19	**SMB**	3.50%
20	**GOPHER**	2.90%
21	**RTMP**	2.40%
22	**POP3**	2.10%
23	**RTSP**	1.90%
24	**POP3S**	1.70%
25	**RTMPS**	1.40%
26	**DICT**	1.40%
27	**SMBS**	1.30%
28	**GOPHERS**	1.20%

In general the positions among the protocols in the table are solid. The top protocols remain the same. Long term FTP has been shrinking and it keeps doing it this year. The two newcomers in the 2023 survey (WSS and WS) remain in the top-8, well above many of the protocols that have been supported for decades - in spite of them still being experimental only.

Seven protocols got a larger usage share this year compared to 2023: HTTPS, HTTP, WS, FILE, WSS and the totally unexpected TFTP and RTMP.



The relative differences for used protocols compared to 2023 show some rather big changes:

Protocol	2023	2024	relative change
WS	7.10%	9.90%	+39%
RTMP	1.80%	2.40%	+33%
WSS	7.50%	9.50%	+27%
TFTP	3.80%	4.00%	+5%
FILE	9.10%	9.50%	+4%
HTTP	91.90%	93.00%	+1%
HTTPS	98.50%	98.60%	+0%
RTMPS	1.50%	1.40%	-7%
DICT	1.50%	1.40%	-7%
SMB	3.80%	3.50%	-8%
MQTT	4.50%	4.10%	-9%
GOPHER	3.20%	2.90%	-9%
LDAP	5.10%	4.60%	-10%
FTP	26.10%	23.10%	-11%
RTSP	2.20%	1.90%	-14%
FTPS	13.60%	11.40%	-16%
TELNET	4.80%	3.90%	-19%
GOPHERS	1.50%	1.20%	-20%
SFTP	15.10%	11.60%	-23%
IMAP	6.00%	4.60%	-23%

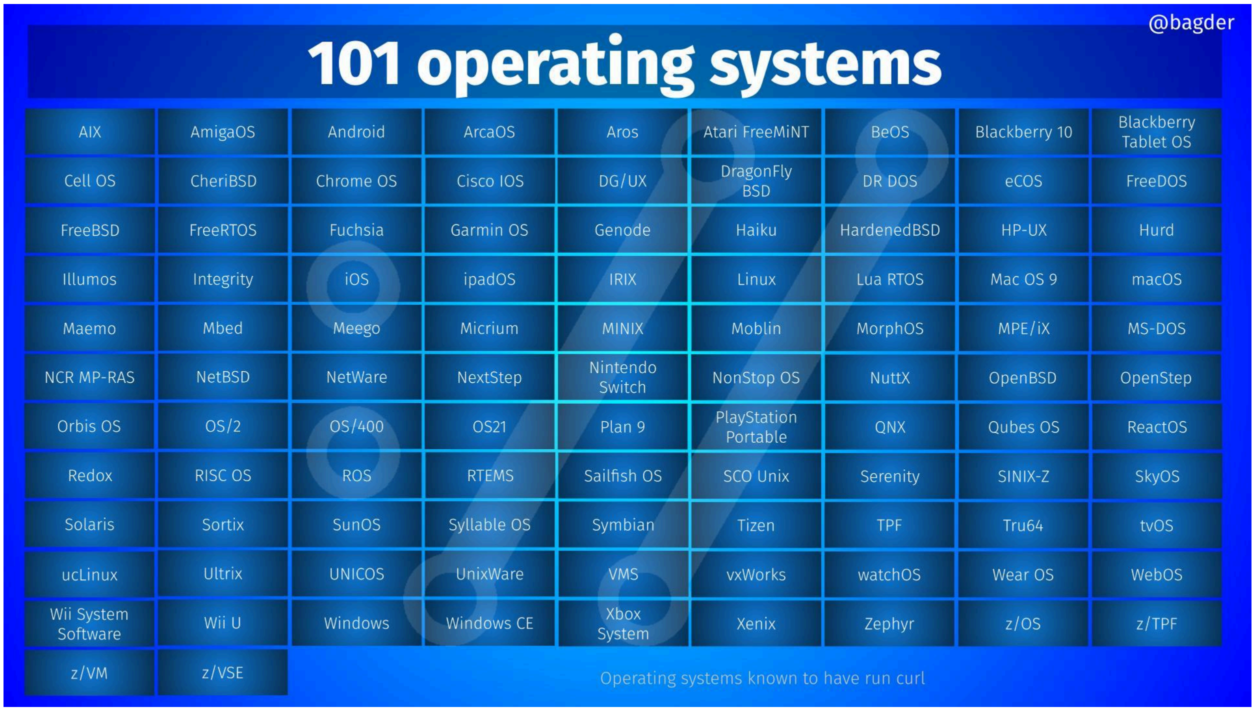
LDAPS	5.00%	3.80%	-24%
SMTP	7.30%	5.50%	-25%
SCP	9.60%	6.50%	-32%
POP3	3.20%	2.10%	-34%
IMAPS	7.60%	4.50%	-41%
SMTPS	6.80%	3.70%	-46%
SMBS	2.50%	1.30%	-48%
POP3S	3.30%	1.70%	-48%

On average, each respondent selected **3.32** protocols.

3.0% of the users (down from 4.95% last year) use 10 or more protocols.
19.0% use 5 or more (down from 20.0% last year).
57.7% (up from 55.3% last year) of the entire population only used one or two protocols, and virtually all of them selected HTTPS and HTTP.

5. Platforms

We have records of curl running on **101** different operating systems and **28** different CPU architectures over the years. Several of those were probably custom modified with changes we never got upstreamed, and many of the users on niche systems most likely did not respond to this survey.

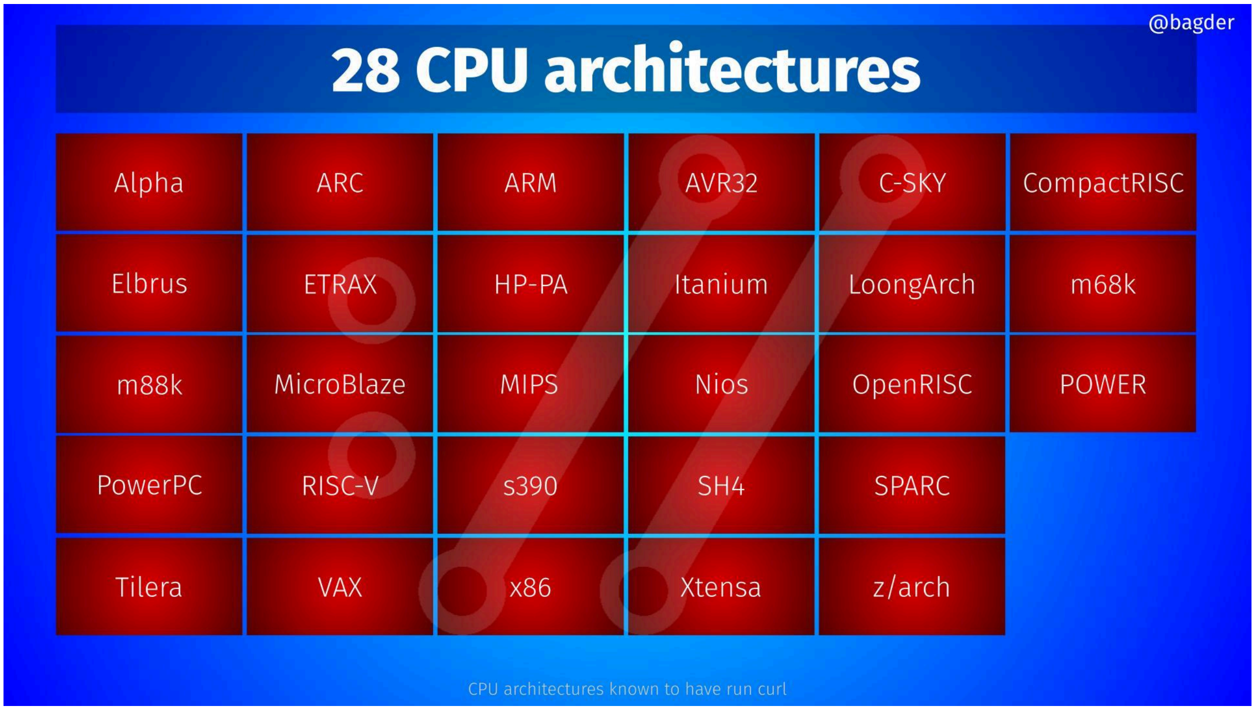


@bagder

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

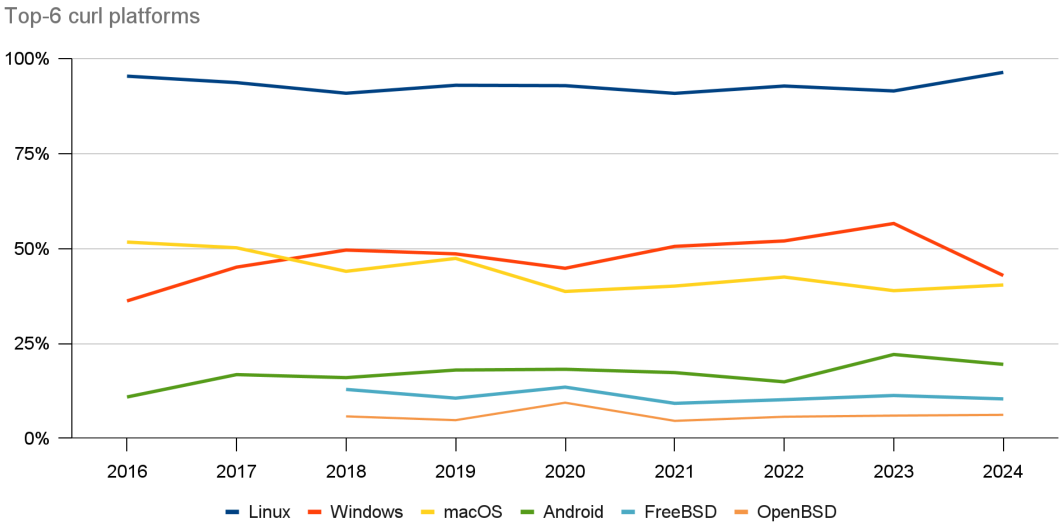


@bagder

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

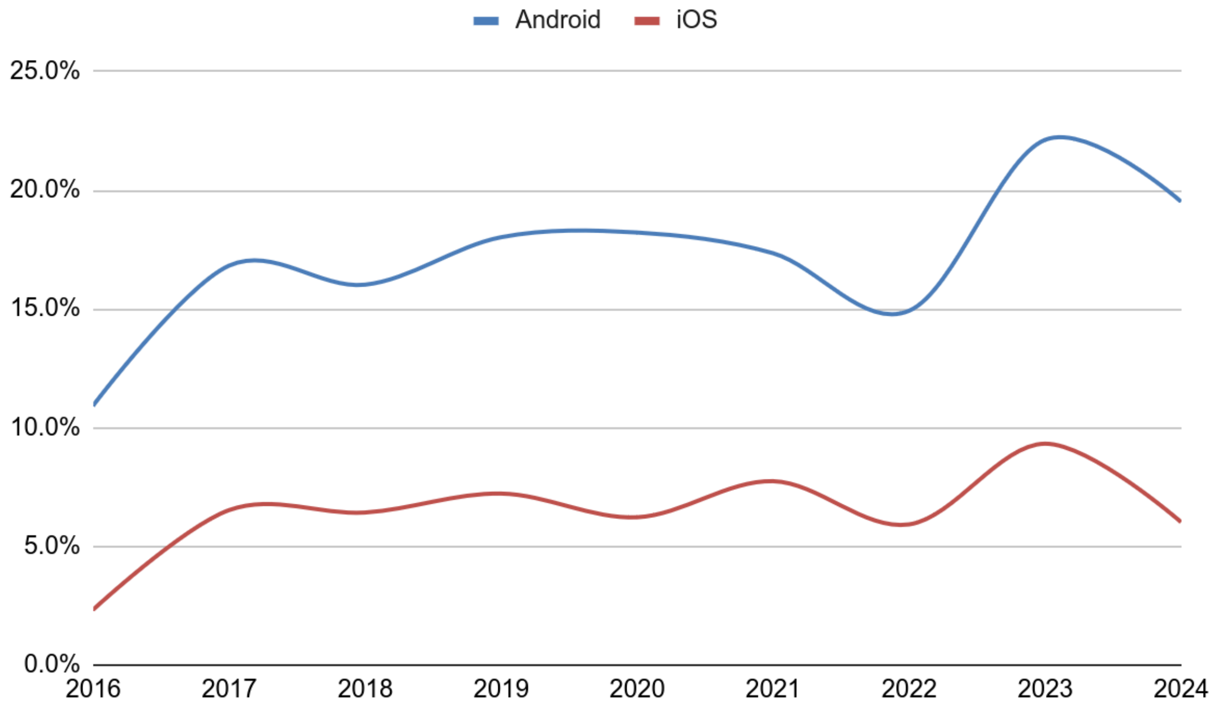


The top-6 platforms seem to remain the same over time. Linux grew a little bit again to a new record level: 96.4% of all users run curl on Linux.

The Windows usage share was growing the last few years but took a dive this year. Its 42.9% is lower than all years previously recorded after 2016. Just an anomaly or the sign of a new trend?

macOS is up a little this year (40.9%)), but still below the median of all years (42.5%).

Android and iOS usage keep following each other closely in the most interesting way and both go down a little this year. Here is the use of only these two platforms plotted over time. Android use of curl is currently more than three times the share of iOS.



On average, users selected **2.37** platforms each in the answer.

The complete platform distribution for 2024 with a comparison to last year.

Platform	**2024**	change from 2023
Linux	96.40%	4.90%
Windows	42.90%	-13.70%
macOS	40.40%	1.50%
Android	19.50%	-2.60%
FreeBSD	10.40%	-0.90%
OpenBSD	6.20%	0.20%
iOS	6.00%	-3.30%
NetBSD	2.40%	-0.80%
Game console	2.40%	-0.60%
Solaris	2.40%	0.60%
OpenIndiana	1.50%	0.20%
Another unix	1.30%	-0.40%
RTOS	0.90%	-0.40%
AIX	0.90%	-0.30%

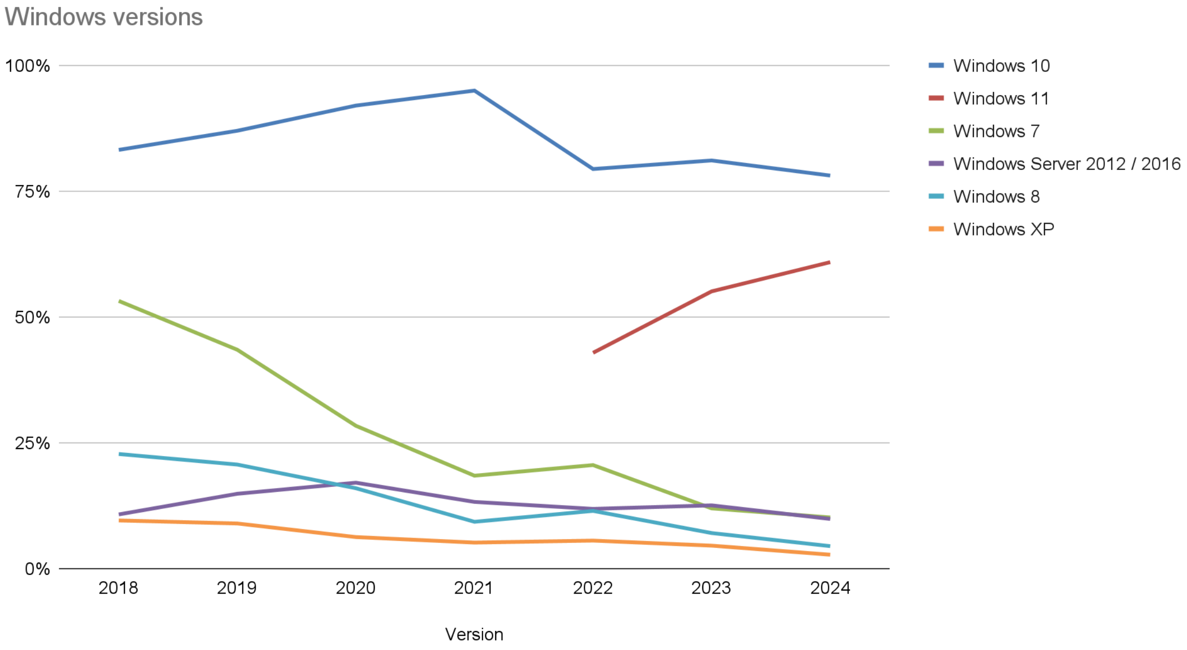
MS-DOS	0.70%	-0.80%
VMS	0.70%	0.00%
IBM I	0.60%	0.10%
IRIX	0.50%	0.20%
HPUX	0.30%	-0.50%
AmigaOS	0.30%	-0.40%

6. Windows versions

(The percentages shown for this question are the shares among the users who say they use curl on Windows.)

Windows 11 entered the survey two years ago and it continued its climb upwards this year. Up 5.8 points to 60.9%. The leader is still Windows 10 at 78.1% (81.1% last year), at its lowest share so far. Given the trajectories of these plots, my estimate says that Windows 10 probably remains the king even into next year's survey

All the other Windows versions keep shrinking, which I think makes total sense.

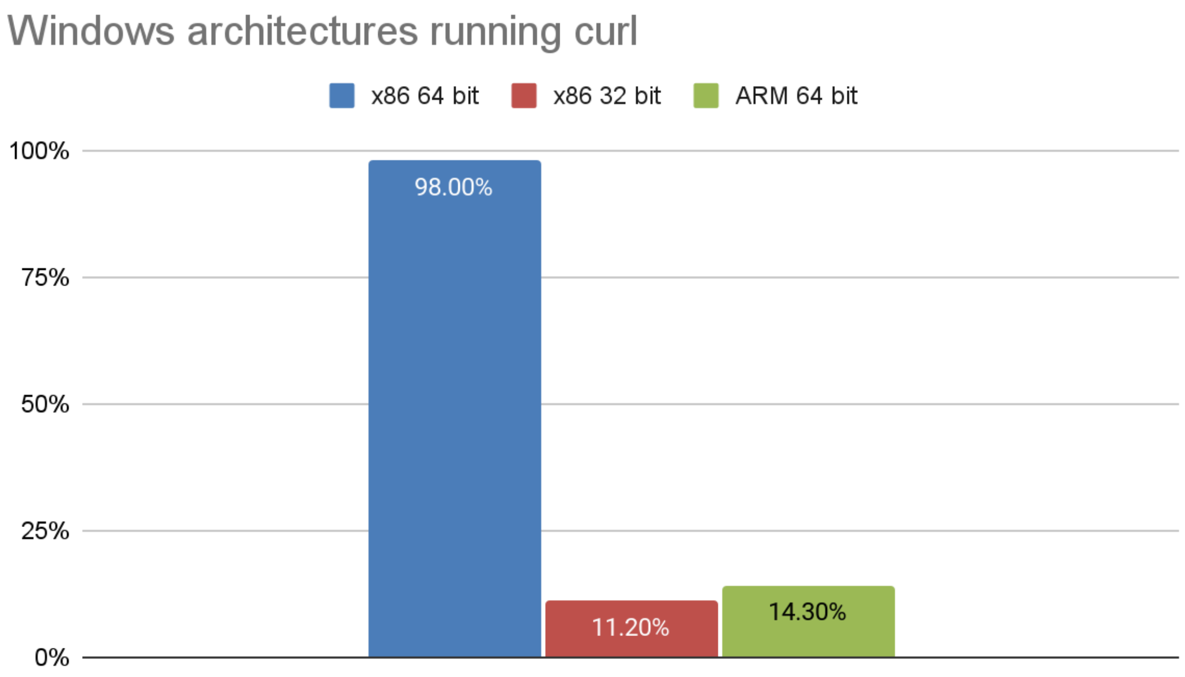


The full Windows version distribution 2024. The average response selected 1.73 versions.

Version	2024	Change
Windows 10	78.1%	-3.00%
Windows 11	60.9%	5.80%
Windows 7	10.2%	-1.80%
Windows Server 2012 / 2016	9.9%	-2.70%
Windows 8	4.5%	-2.60%
Windows XP	2.8%	-1.80%

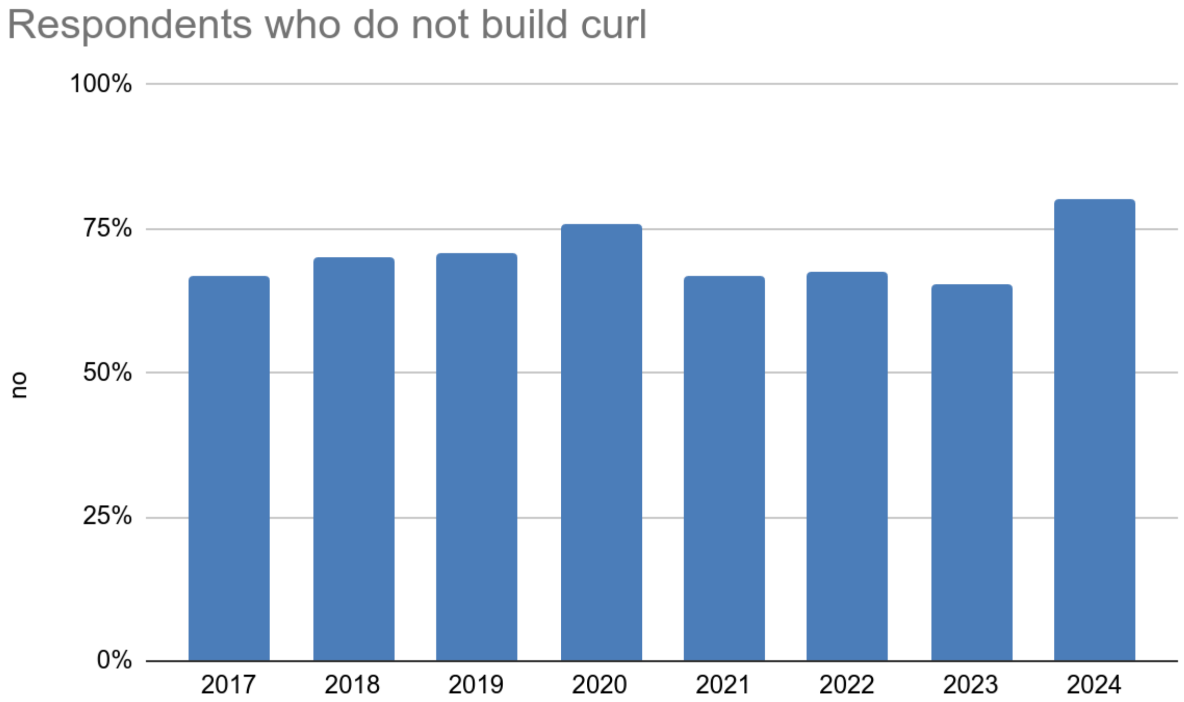
Windows Server 2008	2.2%	-0.40%
Windows Vista	1.3%	-1.00%
Windows Server 2022	1.1%	0.20%
Windows Server 2019	0.5%	-2.36%
Windows Server 2003	0.4%	-0.70%
Windows CE/Embedded	0.4%	-0.20%
Windows 95	0.4%	0.10%
Windows 2000	0.3%	-1.10%
Windows 98	0.1%	-0.50%

This year we introduced a new question: on which CPU architecture do you run curl on Windows? A whopping 98% run it on x86 64 bit CPUs. Illustrated in a little column chart, the difference between the platforms is distinct.

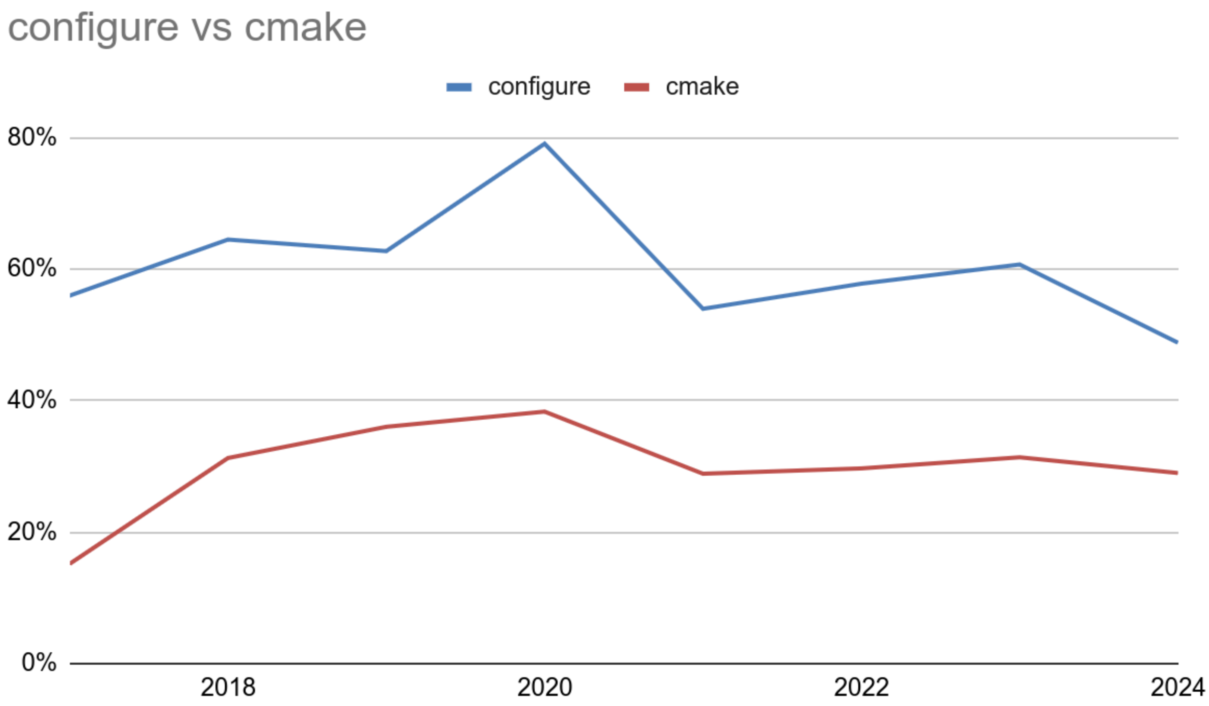


7. Building curl

80.3% of the users answered that they do not build curl themselves. More than ever before.



Out of those who do build curl themselves, the configure/autotools build method remains the most popular one. Comparing autotools use to the second most common provided method, Cmake, this is how the answers have been distributed over time. As a share of the users who build curl.



8. Features

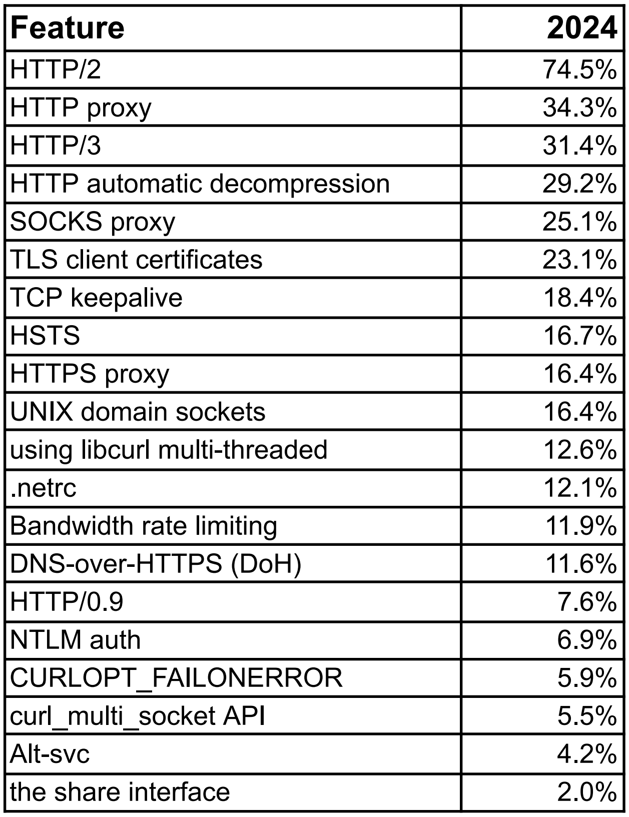
What do people do with curl?

HTTP/2 took a jump this year to 74.5% of the respondents after it almost flatlined last year at 65.4%. Out of the features asked about in the survey, this is clearly the most used one.

HTTP/3 continued its growth for the fourth consecutive year. This also being the first year curl offers the feature without the experimental label. 31.4% of the users used HTTP/3 (up from 26.9% last year).

Quite surprisingly, SOCKS proxy use is still increasing. Its growth surprised me already last year (22.2%), but this year it climbed even more, up to 25.1% of the respondents. It is still below the HTTP proxy rate, at 34.3% this year (up from 33.1%). Interestingly, HTTPS proxy use did instead shrink a little, to 16.4% (down from 20.7%). All in all of course showing that curl users certainly do a large amount of traffic over proxies.

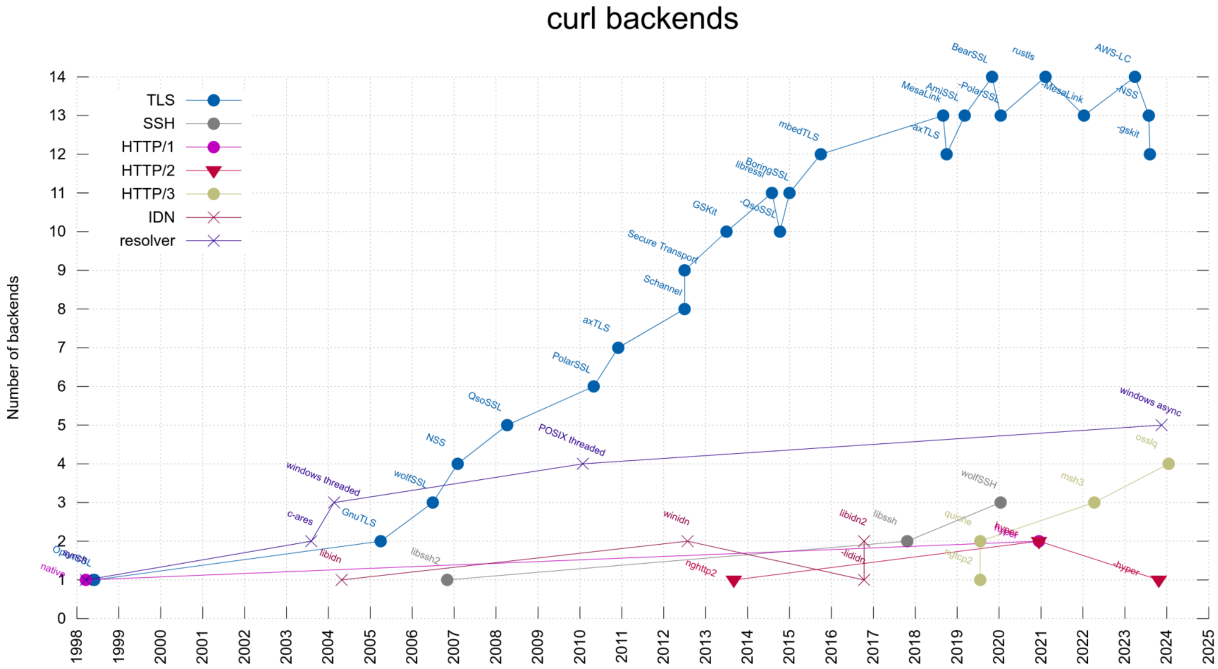
On average, users selected 3.66 features each.



Feature	2024
HTTP/2	74.5%
HTTP proxy	34.3%
HTTP/3	31.4%
HTTP automatic decompression	29.2%
SOCKS proxy	25.1%
TLS client certificates	23.1%
TCP keepalive	18.4%
HSTS	16.7%
HTTPS proxy	16.4%
UNIX domain sockets	16.4%
using libcurl multi-threaded	12.6%
.netrc	12.1%
Bandwidth rate limiting	11.9%
DNS-over-HTTPS (DoH)	11.6%
HTTP/0.9	7.6%
NTLM auth	6.9%
CURLOPT_FAILONERROR	5.9%
curl_multi_socket API	5.5%
Alt-svc	4.2%
the share interface	2.0%

9. TLS backends

At the time of this year's survey, curl supported 12 different TLS backends in its latest releases, down two from last year. But of course, respondents might have used older curl releases that still had support for now removed TLS libraries.

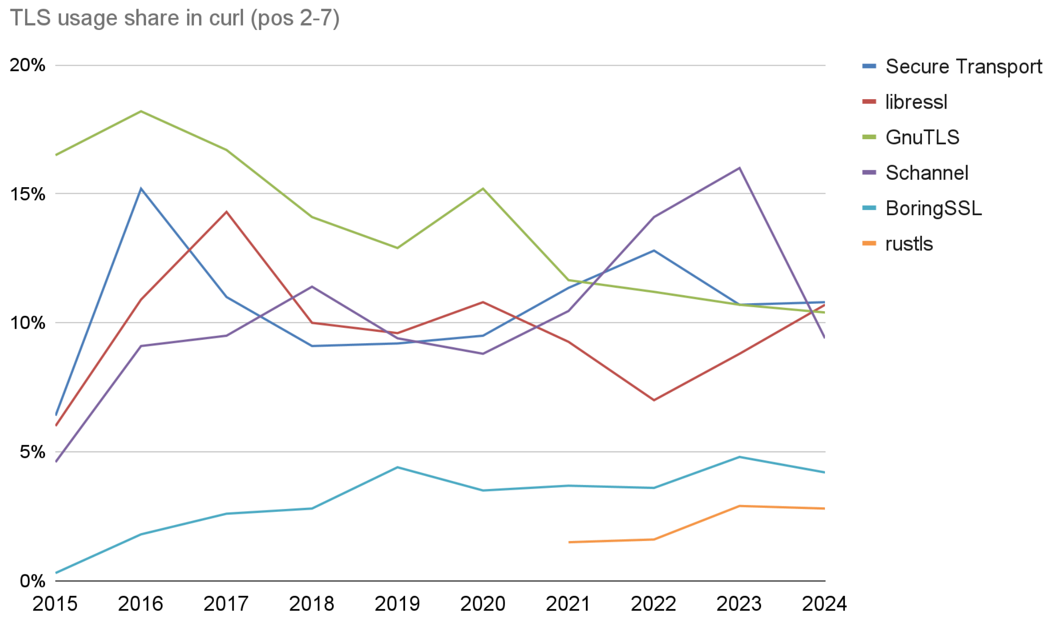


This year a record-breaking 32.7% said they did not know their curl's TLS backend. It might be a good thing that not all users know this, as it is not supposed to be an important factor for users.

OpenSSL remains the undisputed king of TLS backends for curl users at 66.9% (down from 71.6%). Of course we can assume that a share of the *don't-knows* actually run OpenSSL as well. We also cannot tell what share of users who answered "OpenSSL" even if they in reality are using one of the OpenSSL forks. This means that out of the people who did not select "I don't know", more than 99% of them use curl with OpenSSL. This implies that almost all users that run curl with other TLS backends *also* run curl with OpenSSL.

In the popularity contest position 2-7, there are notable changes: Schannel, which climbed up to 16% last year and the second position, plummets again down to 9.4% and the 5th place, while libressl makes a come-back and climbs up to 10.7% (up from 8.8%).

Secure Transport, libressl and GnuTLS are all now used roughly equally: 10.8%, 10.7% and 10.4%.



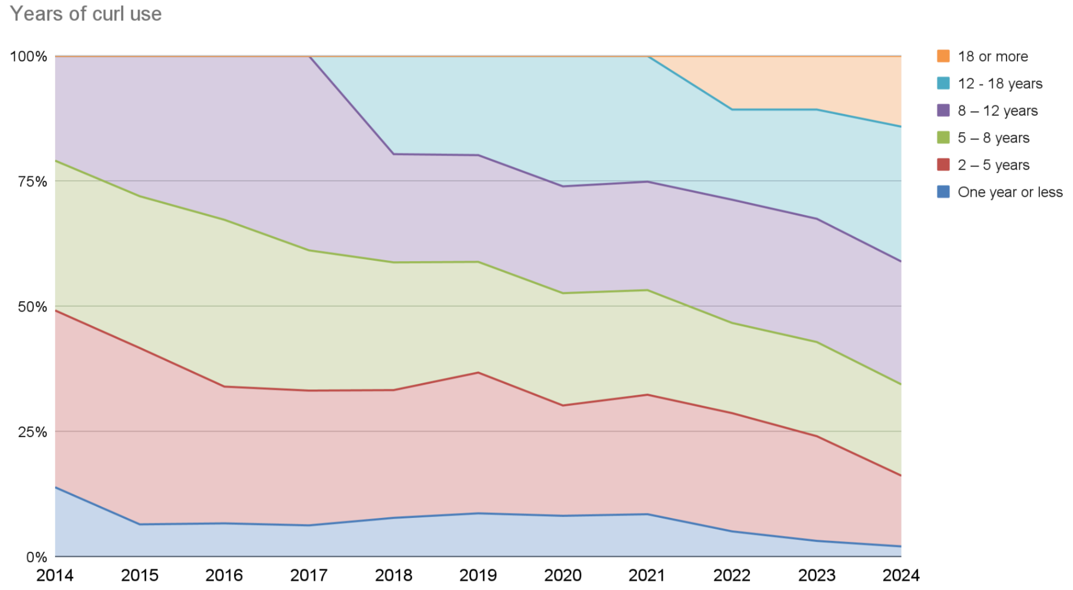
The complete TLs backend distribution in 2024 looks like this:

TLS	2024	vs last year
OpenSSL	66.90%	-4.7%
I don’t know	32.70%	6.5%
Secure Transport	10.80%	0.1%
libressl	10.70%	1.9%
GnuTLS	10.40%	-0.3%
Schannel	9.40%	-6.6%
BoringSSL	4.20%	-0.6%
rustls	2.80%	-0.1%
wolfSSL	2.70%	-0.2%
mbedTLS	2.60%	-1.7%
NSS	2.10%	-0.5%
BearSSL	1.30%	-0.4%
AmiSSL	0.50%	.0%
gskit	0.30%	-0.2%
AWS-LC	0.30%	-0.2%

10. curl use

This is the year curl turned 26 years old. (Even though sometimes we count the birth from the first date in 1996 when the precursor httpget was released.) The general sense is that we have quite satisfied and loyal users. curl mostly delivers on its promises and is a solid and trusted tool. Many of our users have used curl during their entire computer lives.

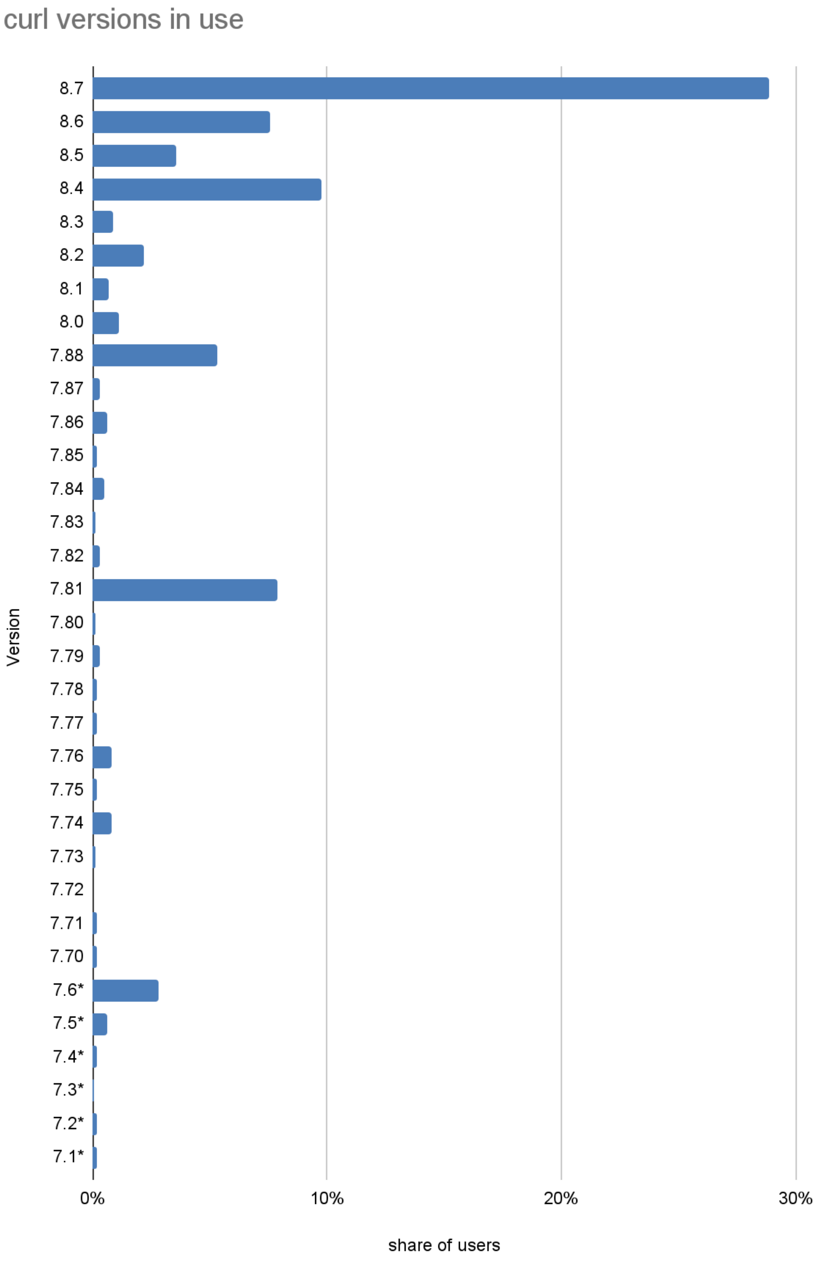
Two years ago we introduced an answer alternative for *18 years or more* as previously *12 years or more* was the top alternative. It makes sense that the older alternatives get more answers over time, but the jump from 10.7% to 14.1% in the 18+ years category is of course more than just another year having passed.



This year we added a question about which curl versions people are using, just to get a rough idea of the version spread among the respondents. Out of the 1289 persons who answered, 297 (23.0%) said they did not know their version.

When the survey launched, curl 8.7.1 was the most recent release, and we shipped 8.8.0 while the survey was still running. It was no surprise that the version spread was huge. A whopping 28.9% said they were using the latest version. Only 0.2% answered “7.1*” - the last curl version that matches that pattern would be 7.19.1, released in November 2009.

Out of the people who know what version they use, the average age of their curl releases was 272 days, which almost equals 8.2.1. The average is however skewed hard because there are a few extreme outliers. The median release was 8.5.0, meaning that **half of the curl user population are using versions within three versions old**. This is much more updated than I had anticipated.



Favorite command line options

This was a new question for the year and it was a challenge to handle. 908 respondents answered. Lots of answers included a mini motivation and explanation. Many answers mentioned more than one option. There was a mix of short and long versions. There were numerous typos. All this therefore required a certain amount of work to clean up the data before it could be processed and analyzed.

In the end 102 different options were selected as a favorite by at least one person. I think it works to prove the old saying that even though every user might only use N% of the options, all users have their own N%. An interesting detail is that 31 answers mentioned -vv* (using two, three or more vs) when in fact curl does not treat those any differently than the singular -v - which ended up the most selected favorite.

The top ten most selected favorite options:

#	Option	Votes
1	-v, --verbose	169
2	-L, --location	85
3	-k, --insecure	83
4	-I, --head	71
5	-H, --header	64
6	-O, --output	63
7	-s, --silent	58
8	-i, --include	49
9	-X, --request	40
10	--resolve	36

The follow-up question was even harder to manage: “Which curl command line option do you think needs improvement and how?” to which we got 217 answers. This is a treasure trove with ideas, nits and things to polish and work on going forward.

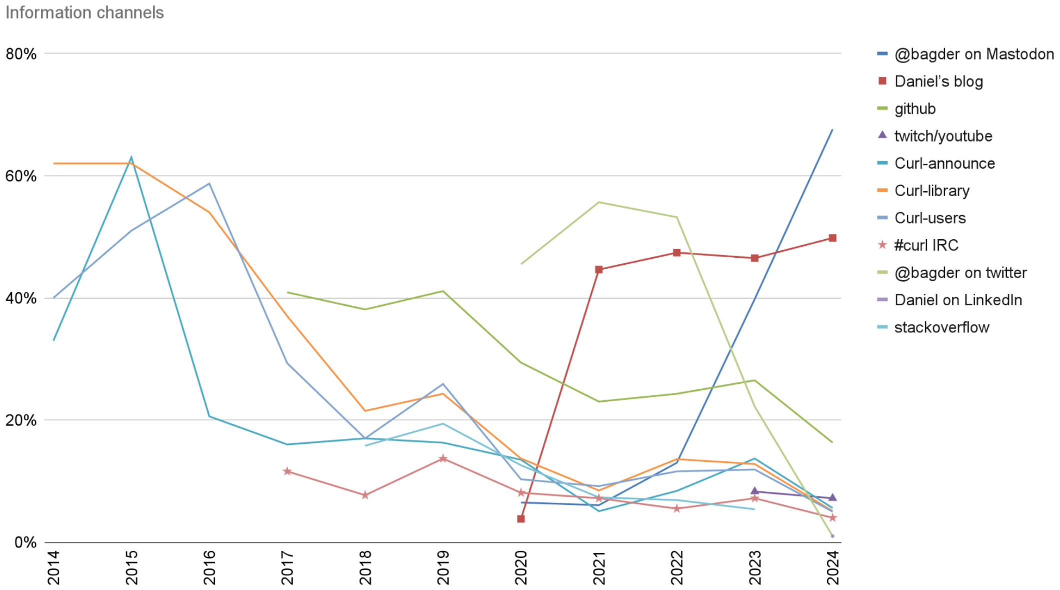
I think the most requested command line option *that we already provide* is --json.

A curated and edited version of the list can be found as Appendix A. I hope we can use details from there to improve a few command line options.

11. Participating channels

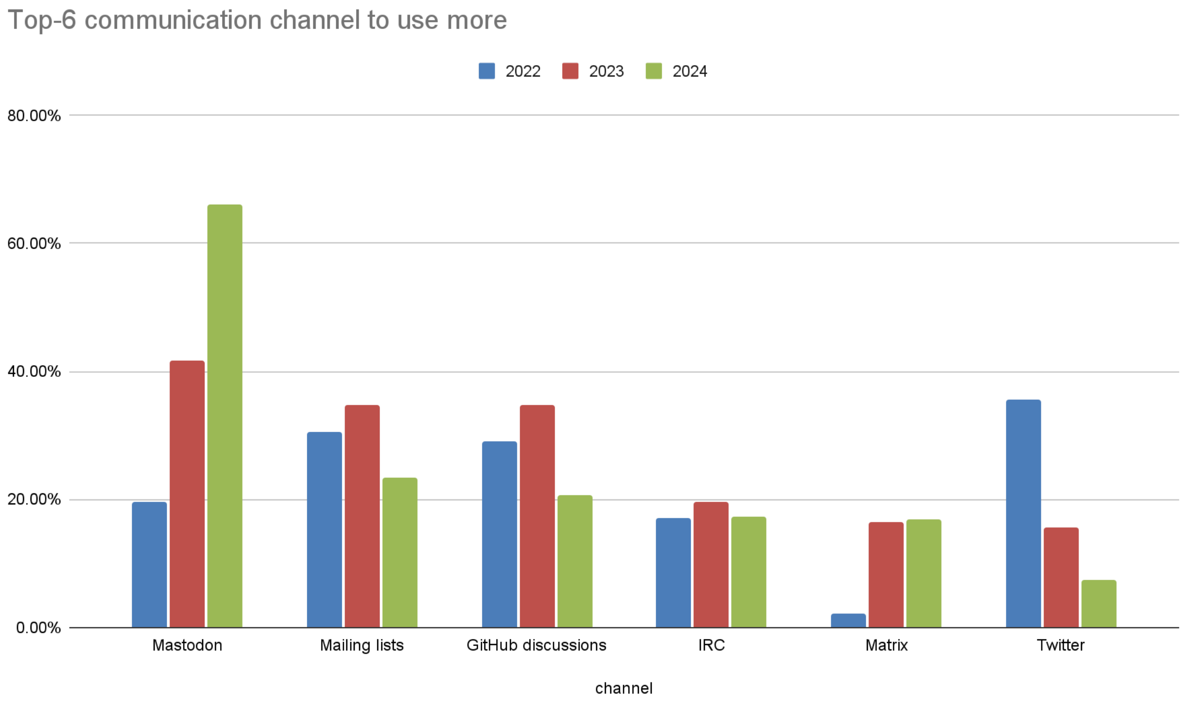
Where do users learn and talk about curl? This of course depends partly on where information is shared and discussions are held in the first place, but also on where users prefer to hang out. This question might to an even higher degree than others control itself, as the channels through which we tell people about the survey might of course be channels users pay attention to.

Since I have personally almost completely stayed away from Twitter/X since a while back, that answer alternative was not present in this year's survey. Some people still entered that in the free text form. I also removed stackoverflow after my decision to stay away from there. Instead my Mastodon account climbed even more this year to the number one channel to keep up with curl stuff, with my blog (daniel.haxx.se/blog) now being second. The general trend with the mailing list becoming less important over the years continues. I need to add my LinkedIn account as an option next year.



Channel	2024
@bagder on Mastodon	67.6%
Daniel's blog	49.8%
github	16.3%
twitch/youtube	7.2%
Curl-announce	5.6%
Curl-library	5.1%
Curl-users	5.0%
#curl IRC	4.0%
@bagder on twitter	1.0%
Daniel on LinkedIn	1.0%

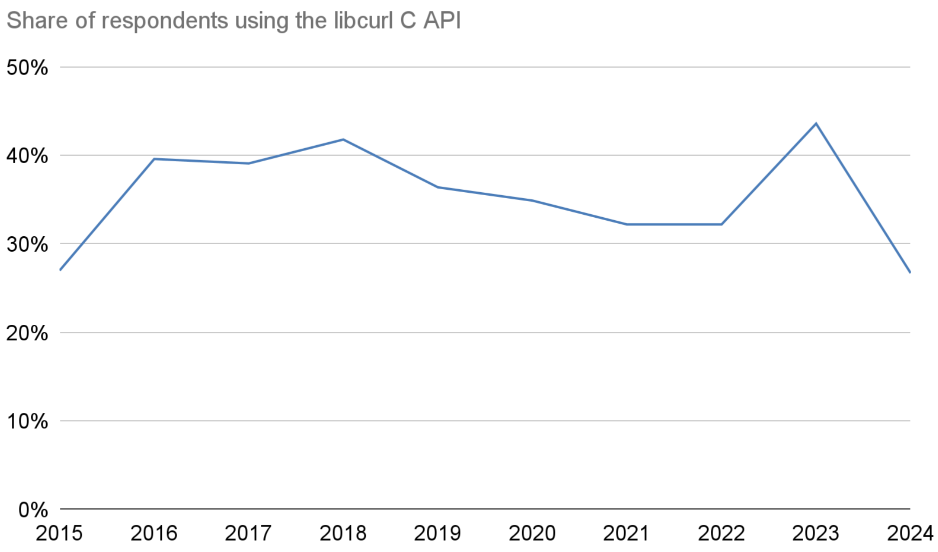
In 2022 we added a question to the survey about what communication channels users think we should use or use more. To talk about curl matters and subjects presumably. This year Mastodon takes off even stronger in this question and Twitter falls down further. Again: selection bias, but since this rhymes with my own mindset I will take this as a confirmation that going Mastodon is the right thing for me/us.



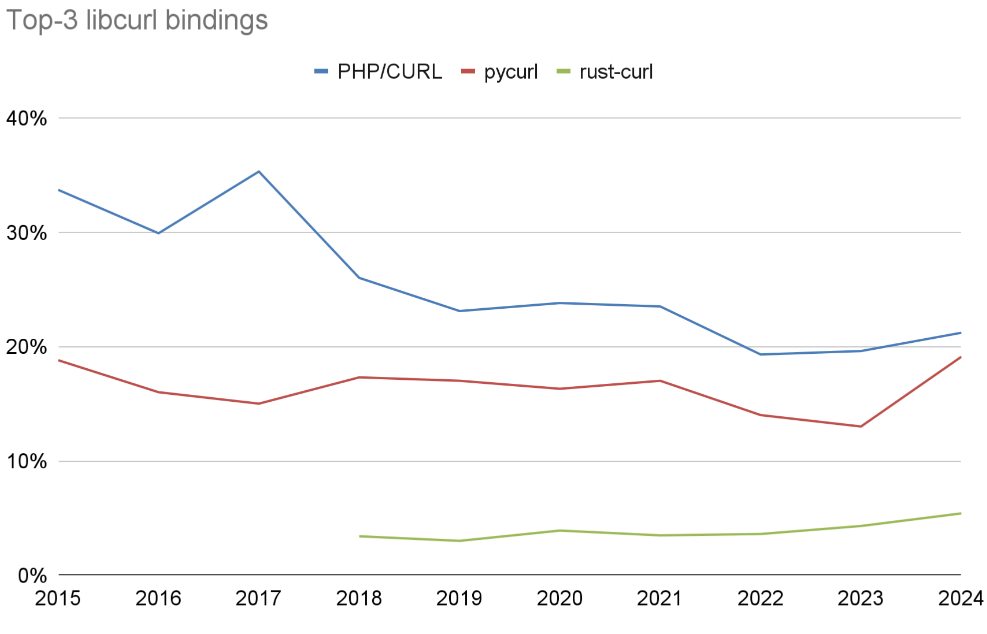
12. Accessing libcurl

libcurl is the network transfer engine of the command line tool and is commonly accessed by users via bindings. The bindings are what makes libcurl truly accessible to almost all developers everywhere. Which ones do people use?

This year 84% answered the command line tool curl, up almost six percentage points from last year. Clearly we reached more command line users with the survey this time.



The top-3 most used bindings below the C API have been the same ones every year since 2015 - *until this year*: the C++ bindings have been replaced by rust on the 3rd place while both PHP and Python saw a slight bounce back from the last few years..



Here’s the complete binding distribution for 2024:

Binding	2024
curl	84.0%
plain C	26.7%
PHP/CURL	21.2%
pycurl	19.1%
rust-curl	5.4%
Go-curl	4.5%
curlpp	3.9%
.NET core	3.8%
Node-libcurl	3.5%
Lua	3.5%
Ruby	3.3%
Java	3.3%
www::curl (perl)	3.1%
Common Lisp	1.1%
R curl	1.0%
cpr	0.9%
Tclcurl	0.5%
FreeBasic	0.5%
ocurl	0.2%

13. Contribution

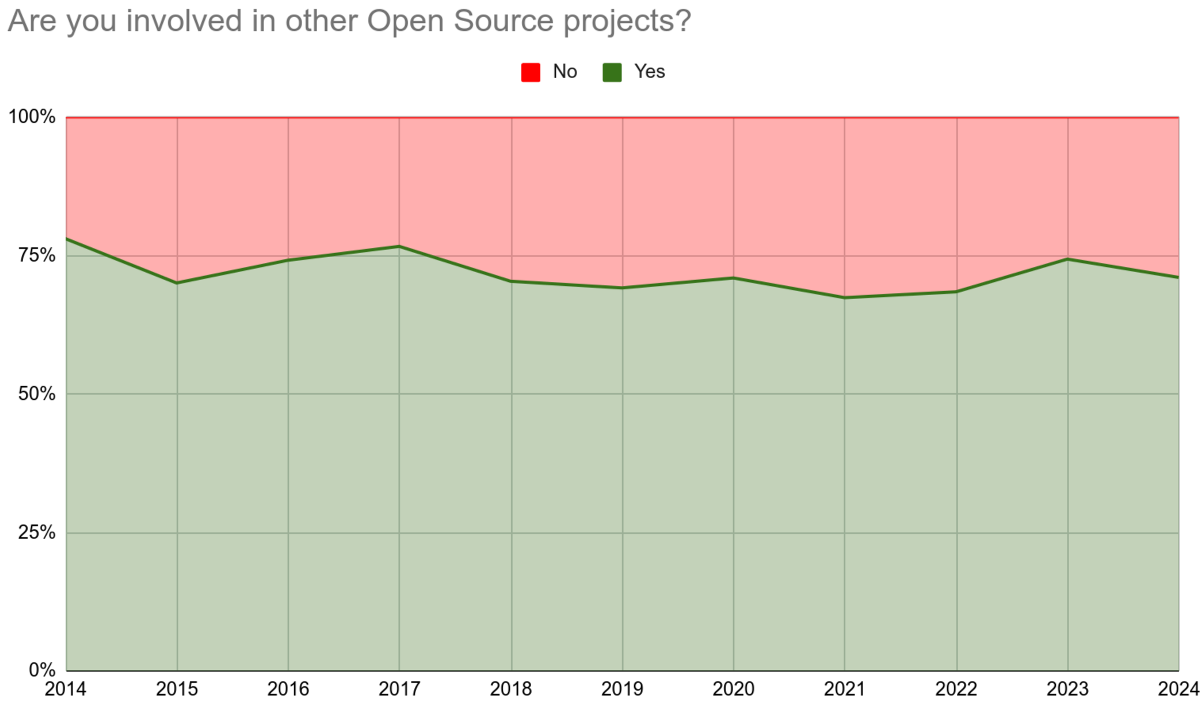
Users contribute to curl to a large degree. Maybe this question tells us more about the particular user population that answered the survey than what it says about curl users in general.

77.9% of the respondents have not contributed yet so there is a big growth potential here!

Help	2024
I haven’t contributed yet	77.9%
filed bug reports	7.7%
sent pull requests	6.7%
curl stickers on prominent places	5.2%
can’t remember	5.2%
helped out in other ways	3.8%
responded on mailing list / forum	3.1%
donated money	2.6%
run tests or provide infrastructure	1.7%

spend time in the IRC channel	1.6%
write documentation	1.4%

curl users continue to be involved in other Open Source projects to a very high degree. We are but one cog in a huge Open Source machinery ecosystem.71% said yes in 2024 (down from 74.3% in 2023) to the question *Are you involved in other open source projects?*



When asked for the reason why respondents have not contributed or not contributed more, the reasons they state are:

Why not contribute?	**2024**
Everything works to my satisfaction	66.6
I don’t have time	41.7
I don’t have the energy	26.1
Things get fixed fast enough	19.3
I don’t know the programming language	17.9

Too hard to get started	13.4
my work/legal reasons prohibit me	4.0
I don't like or approve of github	3.4
I don't like or use email	3.0
I can't deal with the tools	1.4
the project doesn't want my changes	1.1
I find it hard to work with the curl developers	0.8

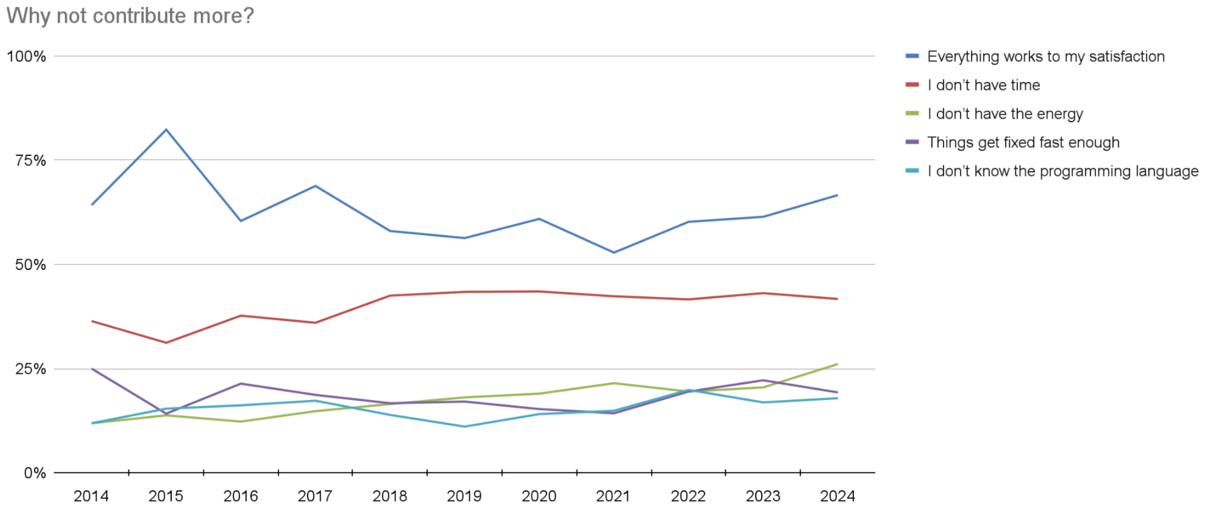
The results and distribution among the answers to this question continue to look roughly the same year over year. Presumably, this is what an old and mature project should get. Most things work, people are busy and the issues that do appear are typically fixed rather swiftly. curl is written in C, a language that is not the most hip one and probably is not even taught to younglings these days which might explain the 16.9% who do not know (or like) the language.

The top answer got 5.2 percentage points more than last year. The "I don't have the energy" got 5.6 more points than last year.

Personally I of course like that the rate of the more human and project oriented problems remain rated extremely low.

The "too hard to get started" is of course always an item to work on to reduce. The median rate over the years is 14.5% on that answer, so at least it has not trended upwards recently.

Looking at the top-5 reasons over the years, they remain remarkably stable over time.



This year, 255 respondents filled in something in the free-text form asking "*What could the curl project do/change to get (more) contributions from you?*".

After filtering out a lot of jokes mostly about adding bugs and making days longer than 24 hours and deduping, I provide the list of suggestions as Appendix B. Food for thought no doubt.

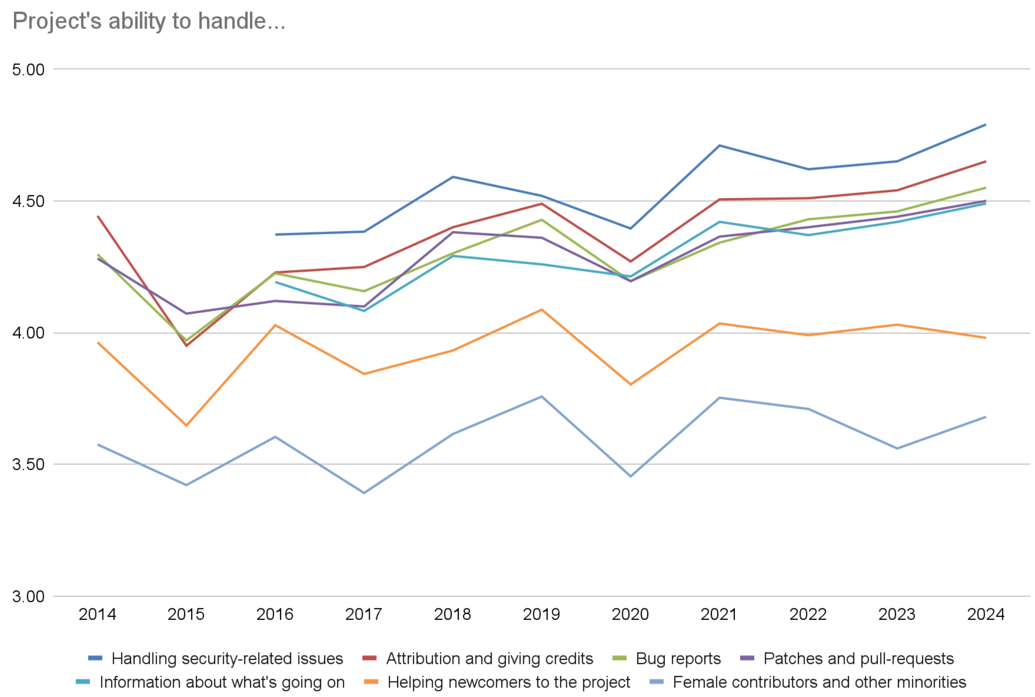
14. How good is the project and its members

The respondents were asked to rate how good we are at handling things in these seven different areas on a scale from 1 to 5. From really bad to really good.

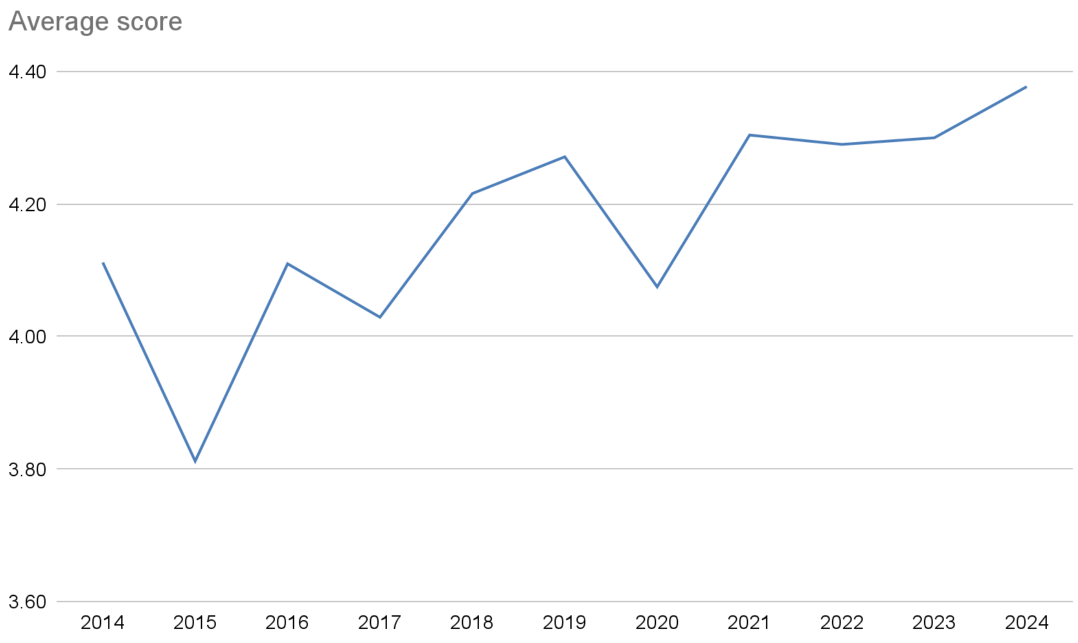
1. Handling security-related issues
2. Attribution and giving credits
3. Bug reports
4. Patches and pull-requests
5. Information about what's going on
6. Helping newcomers to the project
7. Female contributors and other minorities

The order of the above areas is also the order of how good the users rank us. The order has remained remarkably similar over the years. The bottom two areas are what we continuously take away from this and could get better at.

The score increased this year, if ever so slightly, in all areas except one.



Counting the average score on all 7 areas and plotting that in a graph shows that we reached a new topscore. It might imply we are not doing too bad as a project. This year's average score is 4.38, up from 4.30 last year.



categories	2024	diff to 2023
Handling security-related issues	4.79	0.14
Attribution and giving credits	4.65	0.11
Bug reports	4.55	0.09
Patches and pull-requests	4.50	0.06
Information about what's going on	4.49	0.07
Helping newcomers to the project	3.98	-0.05
Female contributors and other minorities	3.68	0.12
Average	4.38	0.08

In the security question, no less than 83.1% of the users gave us a 5 out of 5 rating.

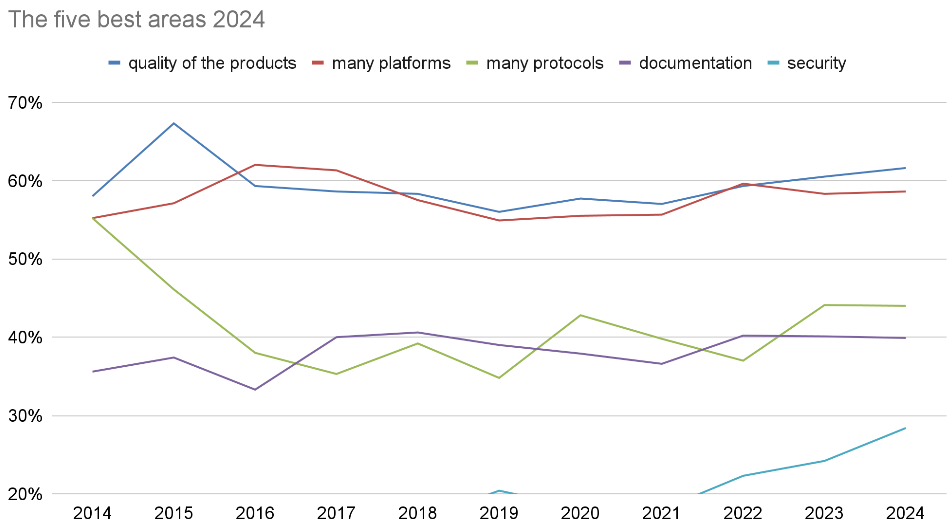
15. Best/worst areas

The four best areas of curl remain the same ones: the **quality of the products**, **its availability and functionality on many platforms**, **support of many protocols** and **documentation.** This year, the 5th best area was replaced as the libcurl API fell down several notches and instead **security** climbed up. This is the first real change in the top-list since we started asking this question.

It is of course tempting to explain this simply on the fact that there were fewer libcurl users filling in the survey this year.

Areas	2024	change
quality of the products	61.6%	1.10%
many platforms	58.6%	0.30%
many protocols	44.0%	-0.10%
documentation	39.9%	-0.20%
security	28.4%	4.20%
standards compliance	25.5%	-2.10%
the features of the protocol implementations	24.7%	0.70%
the libcurl API	22.2%	-11.00%
project leadership	20.3%	3.60%
support of multiple SSL backends	16.9%	-3.30%
footprint of the library/executable	15.6%	0.60%
bug fix rate	9.8%	-3.80%
transfer speeds	9.7%	-1.20%
the user and developer community	8.6%	-1.60%
project web site and infrastructure	4.5%	-0.10%
welcoming to new users and contributors	3.1%	-3.00%
test suite	2.8%	-2.20%
its build environment/setup	2.3%	-1.20%

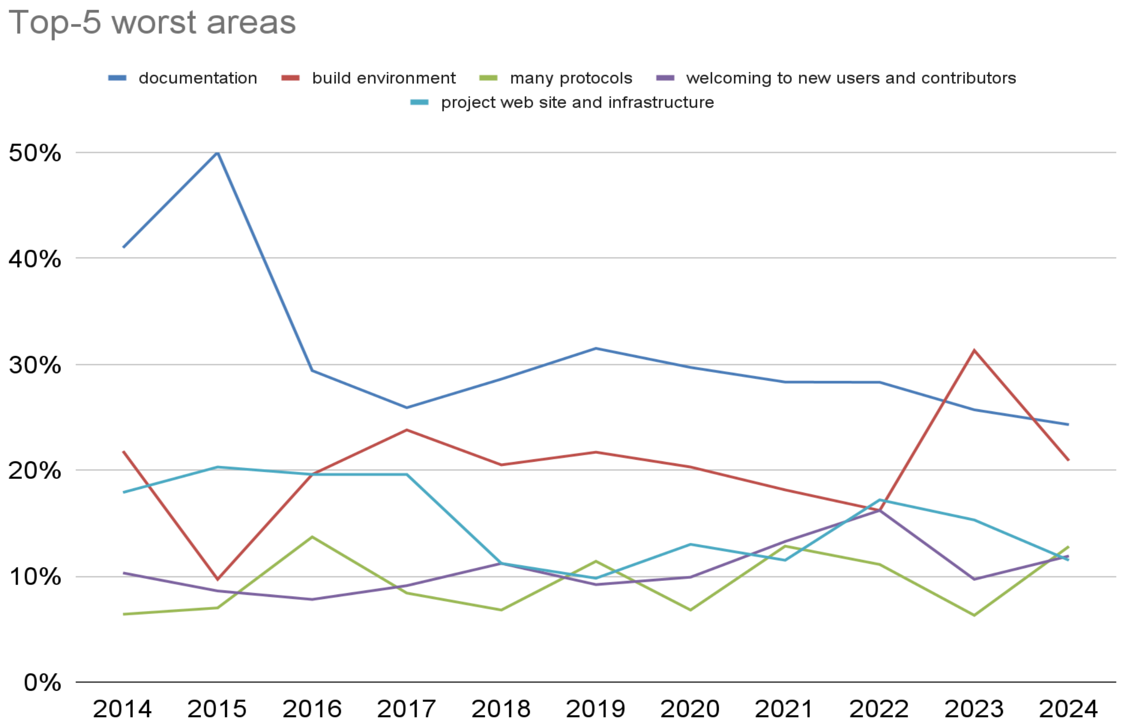
The top-5 best areas has shifted like this over the years since 2014:



The question for best and worst areas feature the exact same set of areas to select from and while this year 1194 persons answered the best areas question, only 235 answered the worst area. 235 is less than 20% of the total number of respondents. I think it is reasonable to presume that those that did not answer this question do not feel they can identify any particular bad areas.

It would be easy to say that the least voted areas for “best” would be the “worst” areas, but that is not how this game works. Documentation remains the most-voted worst area and remains the 4th voted best area. I suppose that reflects the challenges with project documentation perfectly. At least the rate has been gradually declining over the years.

Areas	**2024**	**change**
documentation	24.3%	-1.40%
build environment	20.9%	-10.40%
many protocols	12.8%	6.50%
welcoming to new users and contributors	11.9%	2.20%
project web site and infrastructure	11.5%	-3.80%
the libcurl API	10.6%	-4.70%
footprint of the library/executable	8.5%	2.20%
transfer speeds	8.5%	2.90%
support of multiple SSL backends	8.1%	3.90%
test suite	7.7%	-2.70%
the features of the protocol implementations	4.7%	-2.20%
security	3.8%	-1.80%
bug fix rate	3.0%	-1.20%
its availability and functionality on many platforms	3.0%	-0.50%
the user and developer community	2.6%	-0.90%
the quality of the products, curl/libcurl	2.1%	-0.70%
project leadership	1.7%	0.30%
standards compliance	0.9%	-1.20%



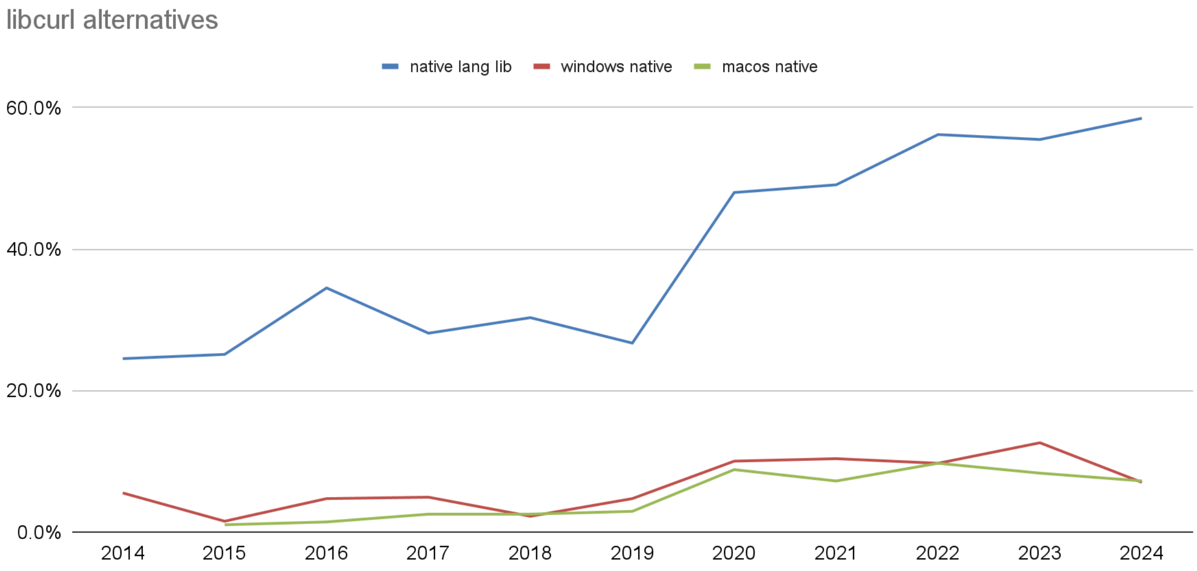
16. If you couldn't use libcurl, what would be your preferred transfer library alternatives?

Keeping an eye out for the competition and where to find inspiration for what libcurl is missing and how to further improve. The answers to this question keep indicating that there is no significant competing portable “system-level” internet transfer library. The real competition to libcurl are clearly the “native” libraries in other “language environments”. I think maybe the survey next year should offer some rust alternative(s) as direct choice(s).

The top answer remains the totally unrealistic “use wget or code ripped out from it”, but the native language library option has been climbing steadily over the years and the Windows and macOS native options remain fairly stable year over year.

Alternatives	**2024**
wget	69.2%
native lang lib	58.5%
homegrown	8.5%
windows native	7.0%
macos native	7.2%

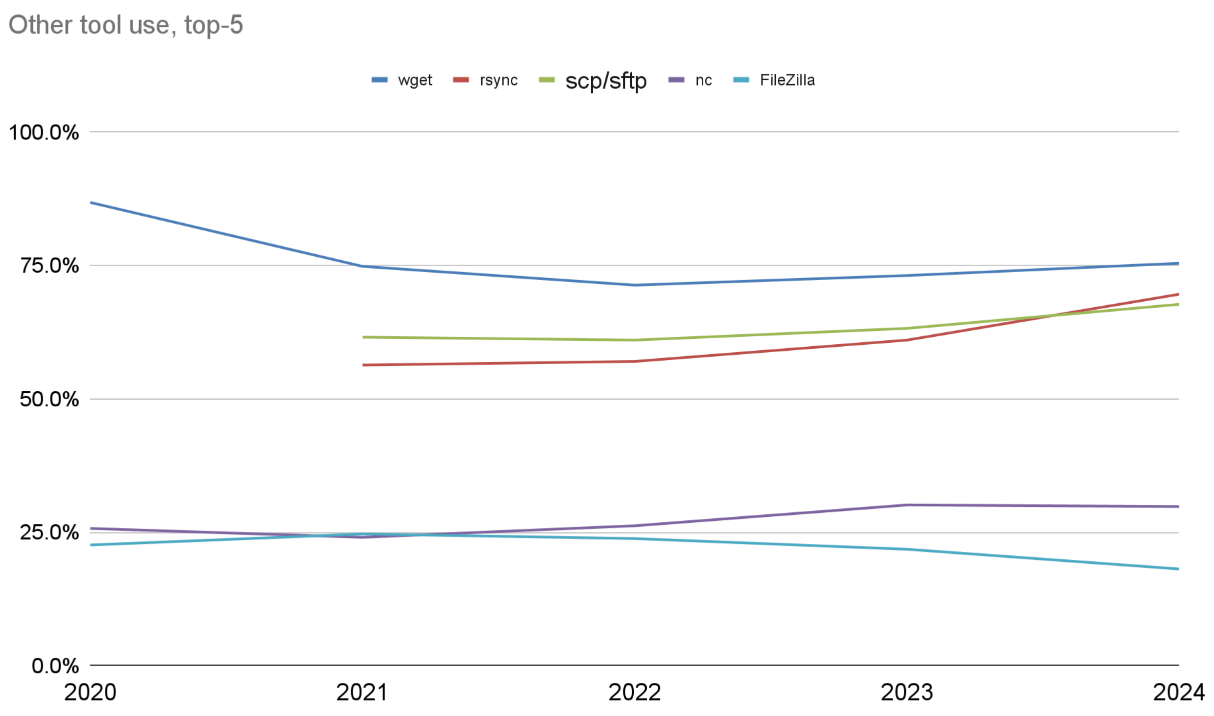
asio	3.9%
Qt	4.5%
libsoup	2.7%
poco	2.0%
Cpp-netlib	0.9%
neon	0.3%
serf	0.2%



17. Which other download utilities do you normally use?

curl users clearly use a large set of related download and file transfer utilities in addition to curl. The most popular ones by far are wget, rsync and scp/sftp year after year and the top-5 selected remain at fairly stable rates.

This is a question with a fairly large amount of write-ins as well.



Alternative	2024	change
wget	75.4%	2.30%
rsync	69.6%	8.60%
scp/sftp	67.7%	4.50%
nc	29.8%	-0.30%
FileZilla	18.1%	-3.70%
ftp	16.5%	1.40%
winscp	13.6%	-1.10%
aria2c	11.9%	-2.10%
fetch	10.2%	-0.10%
Powershell	9.6%	-4.40%
httpie	9.6%	1.40%
lynx	8.9%	-0.70%
wget2	8.8%	1.90%
lftp	8.0%	1.50%
w3m	5.7%	2.10%
Httrack	2.3%	-0.20%
axel	1.2%	-0.10%

18. Which of these features would you like to see curl support?

We brainstorm different things that we might at some point consider adding to curl, combined with protocol developments on the Internet as a whole, and then we ask users how they feel about these things.

Since we actually implement a lot of things over time and new ideas bubble up, comparing this list between years is not only hard but also probably pointless. It is however a great way to get a feel for what people think about various different potential features and protocols. We must also remember that saying that they would like to see these features added is a completely different thing than actually using them down the line - or that adding the specific feature to curl is even a good idea to begin with.

Missing features	**2024**
More JSON	32.4%
WebSocket in cmdline tool	28.1%
Recursive HTML DL	22.8%
gRPC	18.2%
DNS-over-TLS	18.1%
Parallel connection DL	17.7%
GraphQL	15.5%
TLS fingerprinting	14.2%
HTTPS RR	14.0%
SSL session persistence	13.2%
SRV records	12.7%
DNSSEC (DANE)	12.3%
Gemini	11.0%
data: URL support	10.5%
SMB v2/v3	9.5%
WHATWG URL syntax	9.2%
runtime name resolver selection	9.2%
DNS-over-QUIC	8.4%
GUI version of curl	8.4%
Auto-detect proxy	8.3%
OCSP	8.1%
HTTP/3 proxying	8.1%
option for HTTP header order	7.7%
0-RTT/early-data	7.1%
DNS: URLs	6.4%
AIA (dl certs)	4.8%
dynloadable protocols	4.2%

RFC 9421 (message sigs)	3.6%
WebTransport	3.5%
ManageSieve protocol	3.4%
COAP	1.9%

Users also provided a lot of free form proposals. I have edited and filtered them somewhat and show the curated list in Appendix C.

18. Have you used the trurl tool?

This new tool for URL manipulation is over a year old now. Have curl users started to adopt it? How many have used it? Only 6.6% of the respondents have. A great potential for growth.

19. Which of these APIs would you use if they existed in libcurl?

The distinction here between the previous question is that this is clearly a libcurl question. About APIs provided to applications using libcurl.

Maybe this list can serve as an indicator of what libcurl users want from libcurl. Although it is a little vague for some of them exactly what the options mean, and I think some of them are not likely to ever actually get implemented…

API	**2024**	**change**
JSON generation/parsing	59.8%	3.30%
a read()/write() style API for downloading and uploading	27.2%	-1.90%
A zero (or fewer) copy API	19.5%	-5.70%
Server-side support library for HTTP(S)	19.1%	-4.40%
Better aid for doing multi-threaded transfer applications	18.9%	-2.30%
HTTP Content-Disposition header parser/helper for applications	16.3%	0.00%
Pluggable async DNS resolver	14.0%	1.30%
Per-multi bandwidth limitation settings	6.2%	0.00%

20. Which question would you like to see in this survey next year?

I am not sure any of these will actually make it into next year's survey:

- on moving to free software platforms like sourcehut for infra hosting
- Something like: ""What level of curl user do you consider yourself?", with I presume at least 4 levels to choose from. (To accompany this being discerned through the rest of the questionnaire's questions. This is not a redundant question, I insist.).
- Questions on accessibility (e.g. ""Is accessibility good enough"", ""which areas of the project for which accessibility issues can be improved how?""), and include accessibility in ""Which are best/worst areas"" questions.
- About features and plans.
- Ask for favorite software using curl
- Did you know about the feature xyz?
- Distro wars: Which flavor of penguin or pufferfish are you?
- Do you enjoy the youtube channel?
- Ethnicity + gender - would be interesting to see if everyone completing the feedback is a white male and also how different people answer the questions (assuming there are enough responses for a breakdown to be anonymous)
- evaluation of trurl
- I think the survey is well made and should be kept as is
- Importance of specific unstable/beta features
- is wget better
- Level of technical expertise
- Maybe "in what areas do you think curl needs contributions". I don't know *that* curl needs contributors (but the questions seem to imply that), so that question could give you an idea which "call to arms" works in practice.
- Maybe something about the level of competence/familiarity with curl? I'm a very casual user, and many questions went over my head.
- Modify a section - "How good is the project and its members to handle...": the first few set of questions with rating 1 - 5 add text in middle (between "really bad" and "really good") saying "no opinion", "don't know", or something else of the likes.
- More questions about trurl! How is it being used, when have people found themselves reaching for a different tool and because of which feature gaps, etc.
- most frequently used curl command line option
- Questions about usability of curl in shell scripting

- Questions regarding the URL API in libcurl
- Should there be separate LTS stable branch releases of libcurl, along with an experimental bleeding-edge version?
- Something about using curl as part of reverse-engineering specifically
- Something for newbies to get more involved and start contributing to the project.
- Use-cases of Curl, what do users actually use it for? Like just basic REST api tests or just about anything with libcurl, etc.
- What attracted you to CURL? / How did you get started using CURL?
- What's your favorite microarchitecture?
- whether if we like potato or not
- Which question did you find hardest to answer, or needs most clarification?
- Which use cases do you have for integrating libcurl (perhaps separately, the curl cli)
- Why do you use cURL?
- Would anybody else use WebDAV support?
- Would I like to attend a curl conference in Australia? YES
- Would you consider micro-sponsorship of the project?
- Would you like curl://up to be hosted in South America?
- Would you like to join a ruffle to win curl stickers?
- Would you prefer to have a separate curl-tiny variant, with a minimal set of features (e.g. just HTTP+TLS) or other measures to reduce attack surface?

21. Anything else you think we should know?

This is an open ended question that allows respondents to pass on whatever message they felt the people of the project should get or know about. We got 267 write-ins, where most of them were different variations of saying thanks. This is deeply heart-warming and encouraging. The list below is somewhat filtered and curated

- ❤️
- 99% of the time, I use curl to quickly invoke REST API requests. I find it to be nice and fast for that. I'm aware it's more than just an HTTP request sender, but this is enough for me, at least for now.
- A great tool and a great community!
- Another big advantage of curl is its ubiquity - it is beneficial to use curl in shell scripts because it is almost guaranteed to be installed on any machine it would be run on.
- Being female is not being part of a minority. Half of mankind is female.
- Best open-source project!
- Codeberg and SourceHut

- communicate the features a little bit more actively, maybe provide a cool interactive tutorial (+ certification)
- CURL command line, the best thing my web developer friend ever introduced me to when this old school coder had to update a legacy system to communicate with a modern Government Gateway API. With CURL in my tool-belt, I went on to replace 3rd Party FTPS software, then a seamless transition to using SFTP to meet newer bank requirements. Their tech team assumed we would use Filezilla, they had not heard of CURL!
- curl has been a great tool for me, not only for being able to download files quickly and easily, but also for testing some implementations of my own protocols, or general web dev. I really have nothing bad to say about it. Keep up the great work, and thank you!
- curl has been my go to application to diagnose application and connectivity issues and has so far never failed me. Keep up the good work!
- Curl is a super useful tool for me. I enjoy when you write side-stories on mastodon/blog. Like funny requests of strangers or curl licenses on devices 😉😉
- curl is amazing! The communication around releases and in general via mastodon is very insightful as well! Thanks for all your work!
- Curl is an awesome software. Thanks for making it
- CURL is an extremely powerful and versatile app. It is also esoteric and intimidating to potential new users. The user community (as evidenced by the CURL user forum), seems to consist mostly of network gurus - there is very little if any new user activity.
- Curl is awesome, thank you for the hard work and your passion. Much love.
- curl is basically the ffmpeg of networking
- curl is definitely one of those tools that Just Works. it's really good. thanks!
- curl is a great tool, it is amazing that it is still free (I don't complain :-)), you are doing an excellent job!
- curl is great, I was aware it could do a lot more than I use it for but even just answering these survey questions make it clear that I've barely scratched curl's surface, and I've been using it for a decade at least!
- Curl is great. I didn’t know it did a tenth of the stuff that this survey makes clear that it does, and that’s useful to know and I’ll try it more. But for my needs, curl does everything I want and does it well.
- curl is much appreciated, and underappreciated. :) Thanks for curl.
- curl is my rock. Seeing it in the wild is like getting a taste of food from my childhood. Thanks for everything, y’all <3
- Curl is one of my favorite pieces of software. Keep up the great work!
- curl is open source software the way it should be
- curl is the single most important transfer library I have known for more than a decade now. Keep up the great work!
- The curl mailing list seemed more kind this past year.
- Curl Rocks!

- Curl should pay more attention to popular modern needs, such as TLS fingerprint simulation
- Curl simply works
- curl's CLI is one of those core tools that I use every day. Like git, or bash. It is just so reliable. I've literally never hit a bug in curl, and I can't say that about many tools. Thanks for the awesome software!
- Curl's great. Thanks for the amazing work
- Dan’s mastodon is fun :D
- Excellent work!
- Female contributors and other minorities?????? My dog or what? Stop this woke shit!
- For me the main curl point is the tool simplicity, availability (cygwin) and the possibility to replicate and test browser requests
- For me, NTLM = SPNEGO (not using NTLM, just krb5+HTTP on Linux)
- Add an alias for the misspelled "referer" command with the correct spelling [and correct the help manual text.]
- Fuck the haters ; keep up the good work dude.
- fundamental tool for standards based networking; amazing team and great results
- Great job, everyone!
- great project :) Love that it doesn't break every update - very stable
- Had not heard about trurl, but it looks cool -- I can foresee myself using it in the future. This whole survey has made me appreciate how great curl is, so I just made a small donation. Thank you so much!
- I hope my answers are useful. had no idea this was such a big venture, assumed curl was a long ago baked loaf, but what do i know. just impressed at how useful it is
- I am impressed by how fast the project fixed the major bug and kept communication and information up! You have done a great job and it is a great tool!
- I am so glad this project exists.
- I appreciate Your work!
- I didn't realize how much there is to curl, I figured it was just to pull files from the web, and it seems to do so much more, maybe some promotion on the capabilities could be good
- I don't know.Maybe let you know how much I love curl? I want to tell you that I love how powerful it is and it has become a must-have for me since I learned about it.
- I guess you know this already, but curl is awesome!!!
- I had no idea curl supported so many disparate protocols. I'll be using it more for sure.
- I intend to use libcurl in an upcoming feature in a project I'm working on, but I haven't gotten to it yet.
- "I like curl. trurl looks useful but I haven't had an excuse to use it at work yet,

and I love the curl stickers. Daniel and other maintainers have built a very solid tool that's used worldwide, you can be very proud of that.
- Hope to see Daniel again in FOSDEM 2025!"
- I liked your podcast! I hope you resume it.
- I love curl! thank you so much for it
- I love you
- I mainly use curl to download files and web pages
- I only used libcurl for a custom crawler many years ago (8y?) in C. I only use cli curl for downloading stuff that have weird filenames with wget so I don't have to rename it afterwards (using shell file redirection).
- "I patched the MakefileBuild.vc file to use the same dll name and runtime on debug build as on the release build, added ""/MANIFEST"" to the LFLAGS and embedded the generated manifest file using the manifest tool (mt).
- To build libcurl and the curl tool for Windows x86 target I'm using the C Compiler of Visual Studio 2008 (this compiler is still on ANSI C level) and the compiler of Visual Studio 2017 for the x86_64 target."
- I pretty much like curl the way it is, picking parts to realistically improve was not easy.
- I really like curl. I use it primarily as a debugging/development tool for HTTP APIs and I like that I can copy-paste .curl command-line into the API documentation to have working examples.
- I saw this poll linked from Daniel's Mastodon and clicked on it since I've used curl to automate a couple basic tasks. This poll went way over my head as a very basic user haha
- I think curl and other projects stepping off of GitHub could be a small part of making the world a better place… I still can't find it in myself to say that curl "should" do so. It's someone else's time and energy that would go into such an undertaking.
- I think Daniel is an inspirational figure, and I have a deep respect for his commitment to the project.
- I think the extra features above the transport layers are nice but I would rather have robust transport layers than that cream on top if it comes at the expense of the core protocols. e.g. in one of my use cases json+gql would be nice for quick testing, but in actual products I would probably not use the curl stack for that part of the application.
- I think you're a model standard project in the open source community
- I use and am satisfied with the URL API in libcurl
- I use curl above all for https troubleshooting
- i use webdav uploads a lot
- I was not aware of trurl, but it looks interesting.
- I wish curl would go ahead and complete the download even if the request for the size of the file returns an error. My server provides files for download generated on-the-fly and doesn't know the size of the file before the download completes. This means that for my Windows users I can't use curl, but have to

use the Windows ftp client, which is terrible.

- I would appreciate it much if you'd not use Google for the survey, maybe you could consider e.g. https://framaforms.org/ instead.
- I would like to thank you for developing and providing this free software! I am sure, I only use 1% of Curl’s features, but even then it is super useful.
- I'm extremely grateful for curl and how it helps me for my day to day work!
- I'm happy to fill this survey even if I'm merely a user!! Thanks for the time you all put into making curl!!
- I'm too new to this to know about 95% of the contents of the form, but I love libcurl and I hope I can get to learn it better. Thanks for this amazing project <3
- it would be great if there was a bit more on how to get best performance for https/http2 in the documentation -- or maybe an example app showing all that
- it's great! I always like seeing content from the curl community: steady, reliable, and timeless!
- Just my big *thanks* for the continuous amazing work =)
- Keep going, curl is awesome! Love
- keep the multi-platform run-on-everything vibe, please.
- Keep up the good work and make sure we don't have to suppress a lot of warnings in valgrind. So handle free, library unloading, pthread exits, etc even at exit.
- Keep up the good work! Also, Hexagon stickers
- Kudos on a great piece of software. I also enjoy the occasional blog posts.
- lib/curl is excellent software, I appreciate what you do
- Love all your work!
- Love Curl! The memes about it single-handedly holding up the web are very correct! Thank you!!!
- Love the project. Curl is intergalactic
- Love the tool, very useful for sysadmin work to understand what a web server is doing.
- Many thanks! Without you, many things on the internet would not work
- Met Daniel at FOSDEM and he was very nice!
- It might be better if you could reorder some of the question response options. For instance putting the not-applicable type option at the start of the list of options.
- One more time: you are all doing wonderfully. Thank you for all your hard work. I'm really sorry to hear, especially, about the burden that LLM tools are placing upon you all for triage of security vulnerability disclosures.
- One of the best surveys I've ever answered.
- Please keep supporting old platforms
- Please keep up the amazing work. If every project ran as well as Curl does…
- Really like curl. Thanks. I hope the project continues.
- Respect!
- Sometimes I might use it but without knowing I use it. Thanks for developing and maintaining it!

- Thank you all for your extremely hard work! Been programming for over four years now and I couldn't imagine doing so without curl in my pocket. :)
- Thank you for making software so good, I appear to only have 5's to give to you.
- Thank you for your great leadership of the curl project.
- Thank you for your Service! You are maintaining one of THE Open source tools.
- Thank you so much for a tool I use regularly in all sorts of ad hoc situations!
- Thank you so much for everything you do for the community and the world!
- Thank you so so much for the work that you do. In a world of increased enshittification and financialization of seemingly everything, curl is a rarity: a truly free and open source project run by brilliant people. From the bottom of my heart, thank you. (I'll donate, too!)
- Thank you to all the contributors for all of their hard work. curl is a gem.
- Thanks to this survey, I'm going to try trurl and have a look around the github, and other communication instances (IRC, mailing lists). Thank you all for the hard work!
- Thank you for your great work. We already spend a huge % of our income on PHP and PHP LTS with Freexian, we do not have more money to spend actually to support more projects, but you are next if we increase our income.
- The best answer I can give to any question here: curl is a tool I don't think about. It's just there, working when I need it. It blends into the background, in a very positive way
- The biggest thing I know about curl aside from how it downloads things is how the maintainer answers funny emails. Love it!
- The chrome dev tools "express this url fetch as a curl command" is really useful (even though you don't technically have anything to do with it - much of my curl use starts as browser use, though.
- The form building options like -F and --data-raw are very helpful for my precise CLI manipulation of web APIs.
- the insistence on curl "staying in its lane" (e.g. NOT doing things like payload parsing) and the simplicity that follows is, IMO, one of the best things about it. That's not to say I wouldn't love the functionality, but that, despite its huge growth, I've never felt like curl was bloated.
- The manpage can sometimes be difficult to navigate when you are hunting for an option. But I guess often just an internet search will turn up the result.
- The regression streak continues. It's unconscionable there wasn't, and still isn't a follow-up release once curl 8.7.* released with broken Content-Encoding decompression support. Users should not have to manually hunt for issues/patches mentioned on mailing lists, or already closed -- and hence largely hidden from view, masking it from showing it's very much an active issue in the current release -- bug reports to get a functioning library.
- The thing I dislike most about curl is that it's very hard to quickly find things in the man page.
- There are many open source maintainers, but I think Daniel is a nice role model. I really wish there were more like this guy. Thanks for that, and for curl too.

- There are reasons to stay on and to leave GitHub. I can't tell you which you should do, but if leaving and hosting elsewhere is viable and wanted, curl is in one of the best positions to show that it's possible.
- There are several curl features and protocols on this survey that I wasn't aware of and may now use.
- There's really no API debugging tool that I like more than curl piped to jq. fantastic program!
- This survey was way too long
- This survey is quite extensive for someone who only occasionally uses curl!
- trurl! I heard about it when Daniel mentioned it on the fediverse. But then I couldn't find any reference to it on the website when I was looking for it later, not remembering the name. So I used sed.
- Using "memory safe languages" is a growing concern. Are there any plans to port CURL/libCURL to Rust or even Zig?
- When Official Docker image?
- WHY Google forms? /o\
- Woah, this user survey pivoted into a dev survey quickly. Keep up the good work. :-)
- Would like more/improved -verbose information/diagnostics on failure at the interactive command line, and more pedagogical help. Man pages *with examples of use* would be excellent.
- You are appreciated! All of you
- You are awesome Daniel!
- You are awesome. besides the tool i love the way you act in the FOSS world with talks, blog-posts, ..
- You do awesome work, invaluable <3 thanks so much
- You guys are great! Thanks for this tool, all you have done for the open source, and your contribution to the contemporary (or post contemporary) internet.
- You make an amazing job! Congrats and thanks! :)
- You rock for security handling! I consider cURL an example of security done right (from development to community and disclosure handling) 💙
- You're a great ambassador for the project.
- You're an awesome person, Daniel. I strive to be more like you.
- You're great and this tool is invaluable to me over the years. thanks.
- Your tool next to wget was one of the first things I got introduced to on Linux systems. It helped me in so many ways. I just want to say thank you

22. Final words

Crunching the numbers, reading the comments, digesting the meaning, filtering the feedback and generating all these graphs in this report takes fricking forever, but I am happy to do it. The user survey is our only opportunity during the year to get direct feedback from real-world users and I want to make the most out of it.

Enjoy this year’s analysis. I hope we do this dance again next year.

/ Daniel

Appendix A

Which curl command line option do you think needs improvement and how?

--capath/--cacert: would be helpful to be able to specify untrusted intermediates or expose the underlying TLS library options

--ciphers and --tls13-ciphers, should apply to DNS-over-HTTPS as well

--connect-timeout when there's multiple hosts involved, would like to see one per connection and not overall time

--connect-to, can never remember the name, always think of --via

--cookie / --cookie-jar - I'm not sure how to fix these, but it is very subtle how they are different and sometimes you need just one or the other and sometimes both.

--data and --data-*: They do everything perfectly, but they are (at least for me) extremely counterintuitive. I need to look them up in the docs every time I use them and sometimes need several trial-and-error runs until I get the data sent exactly in the way I need to.

--dump-header should allow me to just see headers and output in one stream same as I would hand entering a request over telnet or "openssl s_client"

--haproxy-protocol when used with a HTTP forward proxy for HTTPS connection. The documentation isn't clear if the PROXY protocol is applied to the connection to the forward proxy or for the HTTPS tunnel (supporting both cases independently would make sense)

--help - it's large enough there would be value in just showing categories without a specific category selected.

--help never tells me what I need for making POST requests, I always have to go to --help all

--include is a confusing way to say "show me response headers". Maybe a rename (or alias) to --response-headers or similar?

--interface (which calls bind(2)) needs a shorthand

--json but for GET requests

--libcurl: support for other output languages than C

--netrc - I occasionally want `--netrc-pipe cmd` when I'm not using a shell and so I can't just `--netrc-file =(cmd)` , but also I get this is hard because what if cmd takes arguments

--netrc-optional - I would absolutely love if it could recognize a '~/.netrc.gpg' or '~/.authinfo.gpg' file and attempt to decrypt the contents by probing the $GPG_AGENT_INFO env var or the fallback S.gpg-agent socket pathname if necessary. This is perhaps an obscure use case, but it would give me so much peace of mind to delegate secure management of things like site-specific credentials and encryption tools instead of leaving them on disk unencrypted.

--output There needs to be a simple means of downloading a list of URLs, each with an associated destination path/name. The typical method for accomplishing this is to have a command line option point to a file containing a CRLF delimited list of {URL, destination_spec} pairs.

This can be accomplished with the current config file but that is a bit of a kludge:
• It requires modification of a typically default static file
• It requires separate lines for each URL and destination spec plus the command option

Such a file should also support comments to allow for annotations.
This should also be well documented early and with examples as it is a prime use case for noobies.

--remote-name-all # need a way to Download a list of url ending with /

--remote-name-all could really use a short flag

--remote-time should be default

--resolve could accept another hostname rather than IP address so the override works like a CNAME

--resolve is so useful but repetitive if I'm fetching just one URL. Maybe a simplified --resolve-all IP that just flattens all resolutions to a single address regardless of hostname and port number?

--resolve make the port optional?

--silent … I think the effect of this option should be default rather than have to request it

--time-cond should try to find local file for the mtime comparison when used with -O, --remote-name-all

--tlsv1.* have a non-intuitive behavior. Many people expect those options to use that specific version of TLS

--upload-file with non-blocking read to upload from stdin (or a socket). This is not event based right now and would be very useful for websockets.

-: behaves strangely/counterintuitively, at least in the old versions I'm forced to use. I believe it may be fixed in newer versions though."

-b "" to enable cookies is a bit of an odd way to do it. Mainly the need of passing an empty string after the flag.

-b/-c could be a single switch, possibly even a default user cookie jar.

-c /dev/null, there should be an option to turn on an in-memory cookie jar without writing it anywhere

-d, because I always forget the syntax :)

-d, it should not change a header, we have an API with GET and body.

-H is fiddly when there are lots of headers

-h, --help. curl -h "--some-option" => man page segment for "--some-option"

-H; in response to a 301/302/303, it keeps the additional headers in the GET request, which is usually unwanted

--next; i would be nice if there were something like a new --global- option prefix to set settings that should be ""inherited"" by the next URLs

-Hcontent-type:application/json could have a shortcut

-i: I would be happy to see headers printed to stderr so I could use -i together with piping output to jq. This way I will have both headers and pretty-printed json

-k (--insecure) because when I need it, I always have to look it up 😉

-k should be deprecated in favor of --insecure and complain loudly about being insecure

-L should be the default. Or if that can't happen, then a redirect response when not using -L should be signalled by some clear error message that suggests the user retry with -L, instead of silently saving a 302 error document which will confuse a user who expected the output file to contain some other kind of data. Particularly confusing with curl|bash runes, but confuses people all the time.

-L, I almost never mean --no-location

-LO - If there is a redirect on a URL (using -L), the -O option does not use the redirected URL for the filename. https://slack.com/ssb/download-win64-msi is a good example of this. curl -LO https://slack.com/ssb/download-win64-msi gives me a file "download-win64-msi" instead of the redirected file from the location header "slack-standalone-4.38.125.0.msi

-M; use a pager ;-)

-o and -s, when switching between curl and wget, I often find myself trying to use -O and -q instead, I know it's impossible to just reassign the name of command line options, but maybe one could introduce an environment variable like CURL_WGET_OPTIONS, that would be great

-O I have to look up frequently whether it's -o or -O and what the syntax to specify the output file :)

-O sometimes I'm downloading something from interval GitHub/gitlab enterprise and curl includes the temporary auth token from the query string in the file name

-O, I really want wget like functionality where I don't have to specify the name

-Z needs multi-threading download

-v it usually outputs more than I want, but it's the only way I know to see the content type and status of the response

-w - an easy way to suppress the normal output and _just_ have the -w output would be great -w is a bit of a hassle to use for just basic timings, even with a template file; would be nice to have an option to just show basic timings

-X POST. Make it more clear that it's not what people think it is

--resolve should just need a single IP to force all dns to resolve to that instead of specifying host names or ports

--resolve, but without needing to specify the ports where they don’t change (e.g. 80 and 443 when testing redirects to https). Even better if you could give a hostname to resolve, instead of having to look up the IP address separately.

-e, --referer` should have an additional alias that is spelled correctly. be the change you want to see in the world, and all that ;)

-v can show more time related info

--no-clobber that doesn't download an additional file but just skips the download entirely

a) simplifying the most popular curl variants; and b) using curl programmatically, e. g. a bit like UNIX pipes for web-related stuff (although I can also use ruby, so I am not that dependent on curl itself actually)

--range; would be nice to have support for decimal/binary suffixes (K/M/G, KiB, GB etc.) as well as the ability to use multiples of a given "sector" size (like dd's bs/count param). Rationale is, I use range downloads to fetch package metadata in my Linux package manager for a specific package structure (GPT partition table image), without downloading the entire package.

--cert-status should be able to return how many days valid is left Certificate revocation check options

curl --help < I don't really like the category thing curl PUT, easier adoption for put and GET API calls.

Downloading a file (like wget) could be improved - with automatic naming of the file

downloading files - wget is much cleaner

facilitating addition of mail attachments to simple leading text mails

For Windows CMD or PowerShell or Windows Terminal, it does not support multiple-line commands to post data, I have to run it in git-bash or WSL.

General information of TLS certificates would be cool

I always have trouble with -o/-O when (e.g.) downloading a patch from a GH commit...

I find http POST confusing

the -d flag could attempt to infer and set the content-type header that would be quite helpful. E.g. curl -d '{}' https://httpbin.org/post should set the content-type to application/json

a simplified version of the --resolve flag. Every once in a while I just want to play around with reverse proxy routing rules, and in that case I would prefer to simply write the ip address everything should result to instead of the whole host:port:address mapping. E.g. I would like to write `curl --resolve 123.123.123.123 https://mydomain.org` instead of `curl --resolve mydomain:80:123.123.123.123 https://mydomain.org`"

I sometimes wish it would have nicer defaults for just downloading a file and checking if all went well and throwing an appropriate error if not in a bash script. Or I don't know the best incantation for that. But I'm always struggling with it

I think -L should be on by default but I also kinda get why it isn't.

I use -H "Content-Type: foo/bar" very often, I always want a --content-type convenience option

I use -X because the exact type of request being sent is important for some HTTP APIs, and I prefer being explicit over implicit. But curl sometimes complains (in verbose mode) that it isn't necessary. I want to be able to be explicit without causing problems.

-v is overly verbose when I just want to see headers, but when checking the docs I found --dump-header instead. At first I thought --dump-header could not send headers to stdout, but "-D -" worked. This does not appear to be documented (in the man page)?

I wish -i wrote to stderr so that I could pipe directly to jq

I wish it had an HTTPie-like interface too. I forget the switches.

I wish the default behaviour when GETting a binary was to drop it on disk. That's the only reason 'wget foo.tgz" is still ingrained in my muscle memory .

I wish there was a fast-to-type single option to 1) disable the progress bar 2) output the http request headers to stdout 3) same with the response headers and 4) send the response body to /dev/null

I would like to see an option to dump exactly how a form is going to be transmitted after being built with various flags. Ideally including a way to just dump it to a file for examination.

I wouldn't mind --fail being the default.

I'd love to see -k be deprecated in favor of forcing the use of --insecure, simply because a lot of companies think -k means "make our SSL certificates work"

maybe -v could be formatted in a more appealing way

IMHO no globbing should have been the default behavior. It keeps biting me in the butt.

It would be cool if there was a short way to express "get the contents of this URL and write it to a file, following redirects, not printing a progress bar, and NOT writing the file if the server returns an error"

it would really handy to have inbuilt support for --json_pp and something to solve better newline parsing

JSON format response headers for piping to jq

when downloading an entire directory, I hope it can automatically create the directory structure based on the link path, like wget has always had. https://example.com/path/ → path/path1/file1 path/path2/file2 It would help me a lot if it could automatically create path/path1 path/path2 etc. Option --remote-header-name can be used normally with option --continue-at at the same time. Sometimes I need to obtain the remote header name and continue downloading."

maybe a --http2-only alias a la --http3-only?

Maybe have a way to download without specifying something in -o (the only reason i used wget still)

More precise control of output (http headers, body, stdout/stdin)

ability to iterate over list of files from directory listing from sftp (using pattern) to download files

netrc - allow site-specific sections in .netrc

maybe introduce multiple levels or topics for --verbose

Options that engage features that are not present from the selected backend should fail consistently.

Output timings. The format options are ok but could be easier to use and a standard template would be nice

Perhaps a default config file, sorry if it already exists

Possibly --write-out, I've used it several times instead of a full script to output some metadata but figured that some things are not available in --write-out so had to go back to scripting.

Proxy its very lacking and especially proxing dns request

quality of life command line options for common headers like "Content-Type: application/json"

Sending data with the request is a little cumbersome (the `@` syntax is nonstandard) and passing DNS resolution overrides is something I always have to look up.

Showing response headers with the response content should be more prominent in the docs

Specifying POST data and sending files (@ before path) are not obvious, need to carefully read docs

Support for newer SMB versions would be nice

I always need to look up how to use -d, --data-binary and the @ sign for posting entire files

The -o vs -O is bad but would break scripts

The entire file sending stuff

The formatted output used to surface TTFB, wait_time, etc. Having a simple —dump-all-stats-in-json would be nice for a lot of use cases. Filtering json is easier than passing arcane flags in 2024.

The option '--proto '=https' --tlsv1.3' Make a shorter/more memorable option to force TLS 1.3

the short option for Output being opposite in curl vs wget is occasionally disorienting

The various form data options; HHTPie is a lot easier in this regard.

there should be a simple flag to only display the response headers and not the body (this is only possible at the moment with redirection to /dev/null)

Timings, Js handling

Uploading a file as the body using the awkward --data "@filename.xml" syntax. Last time I tried this it was very picky about file paths and I ended up giving up trying to use a relative path and gave an absolute path. But maybe that's already fixed up in a newer version of curl and I was only seeing it because of the ancient work debian hosts.

Using --form to upload a file with @ is cumbersome and requires pre-escaping the path just in case. It would be great if that were not necessary.

Verbose/silent options, including printing/hiding errors.

ways to tone down (report/override/disable) --insecure

We can not directly use websocket urls, we have to enable or configure websocket, i guess this can be provided pre-configured.

We need session reuse resumption feature while using https

When it automatically "fixes" your path, like resolving /../, it really should print a warning saying how to make it stop trying to be smart

When making http requests I miss a friendlier way to enter key value pairs for the URL query. Totally not curls job but still, that's what I miss

would like a short option that tells http to be "clever"; enable all of -sSL --compressed etc

you always say not to use `-X POST` but I can never remember the thing you're "supposed" to do instead

Appendix B

What could the curl project do/change to get (more) contributions from you?

Provide more insights how to get involved into Foss without giving up on real life (I really loved the post on working open source, but it is unfortunately also a bit disappointing)"

--remote-header-name --remote-name

"good first issue" and mentoring type stuff. You probably already have this, I haven't checked

(please don't) break stuff or remove features that I use, but aren't being maintained enough

1. Enable CURL_LOCK DATA-CONNECT to support multi-threaded concurrency
2. TLS browser fingerprint simulation

A guide / blog post with steps on how to get started with contributing and a list of bugs to pick from for Open Source beginners would be most motivating

A guide to getting started for people inexperienced with contributing

A mentorship program would be quite compelling.

A modern handling of mailing lists would be beneficial for new people. I have found that more and more people are not as familiar or not familiar with mailing lists anymore. That isn't a slight on people, but I believe that shows that the technology has likely shifted to easier routes of handling it.

a public kanban listing what's to do and explicitly saying "we need more people, feel free to take a ticket"

A quickstart guide to development would be nice, more "advertisment" on the curl project/requests for more contributions

accept patches that make things more correct but don't necessarily fix a bug

Add meson support for easier embedding

andreas kling-style hacking videos

Architectural documentation of how Curl works, adopting a more expressive language than C to enforce invariants and constraints.

As I’m not looking actively at curl, communicating on current issues might « nerd-snipe » me, that’s unlikely though, but still possible.

As it depends on my time availability, I don't think much can be done. There are some projects that require most of my time, so I can not commit to more projects in general, without compromising other things.

As long as it works, I don't need to contribute. It's such good software.

As strange as it may sound it is hard to contribute to such a polished piece of software.

Ask for concrete things to help (like "please fill out this survey" / "please try to solve X")

Ask for money, I will send some.

Bazel integration

Be open to newbies and more mentoring

Better 'how to get started as a developer' documentation and 'good first issues'

Bit more accommodating of newbies

Build system that does not suck and does not require learning whole new stack of garbage.

Communicate that the curl project is search for new contributors/maintainers

Convince my company to collab

Create list of easy intro tasks or bugs to solve

Curl GUI

Fix Australia's broken copyright system. Lol. (I cannot release copyright. You cannot release to Public Domain. I have to license it in perpetuity, which is a legal headache to get right, and most open source licenses are not nearly enough to make it work here.)

Fix the TLSv1.3 0 byte transfer issue. It is a showstopper.

Get some sort of credit for curl contribution as to show in cv

Gmail WorkPlace

have a list of low-hanging fruits and easy issues to get started with the project and get to know the codebase better

Have a steady z/OS build and test environment

Have a todo / task list

Have newbie issues (for coding), transparency Report on monetary usage (for donations)

Highlight possible non-code related opportunities to contribute host curl on a free and open source code forge

I guess if you just like, died, and the project went to shit. You're doing great, keep up the good work

I may find myself contributing in the future out of inertia and necessity.

I only really contribute to open source projects when I encounter a deficiency or bug in my regular use, and so far I haven't really encountered any in my infrequentish use of curl.

I think it would be disingenuous of my to suggest changes because realistically I probably still wouldn't contribute. I have been eyeing the open source community for years but still haven't contributed in any way. I think it is because I am afraid I won't be able to follow through with contributing in a way I would like.

I was not even aware curl needed contributions, will keep it in my radar now!

I would not even know what language it uses under the hood. I am not a developer.

I would probably contribute more if curl had more bugs to fix. But it doesn't. So, no contributions for you.

I'd love an official curl shirt

I'd love to contribute if it didn't involve multiple forms and meetings with IP lawyers at my workplace.

I'm a documentation specialist and a hobbyist coder, but I don't know where my skills are needed. If it was clear where I could do something to help the project with my skill set, I would contribute.

I'm just between jobs (and have been in a really draining, crappy job for the last two years before that). Prior to ~2 years ago, I was not able to program on the level required to make meaningful contributions.

If the quality of the tool went down I might be forced to investigate bugs and send bug reports/patches. The quality of curl is just too high! :D

In theory I could contribute in my free time, but unfortunately not during my work time due to legal issues ("we cannot contribute to open source due to export control, dual use, liability, ..."). In practice I am very busy with all sorts of things.

internship

Keep support for old C compilers (i.e. for SCO Openserver).

List features that are waiting a contributor. Write a small spec for the feature, so the complexity for contributors decreases.

list of issues for people new to the project

Make internal architecture documentation more readily available

make it available to to mobile apps platforms and as simple as possible to start and contribute

Make it easier to get started / provide onboarding documentation.

Make the issues that need addressing and the need for help more visible. Its such a mature tool and I never hear complaints about it. When thinking of the open source projects that could use help, curl is never the one that comes to mind.

Maybe making it easier to understand what could be done. I get everyone has their itch to scratch and that is a great place to begin, but there must be stuff getting dusty that someone new could work on. Maybe listed by degree of difficulty.

Maybe more docs on how cURL works internally...

Maybe simplify build system

Migrate away from github.

Migrate to Codeberg and SourceHut. =)

more emphasis on community, good-first-issue

More GitHub issues maybe? That's where I search the most

More reach / visibility. I see 100% of curl news from Daniel's LinkedIn and haxx posts on HN. I don't actively look at the curl repo.

More Rust ;)

more streams while fixing bugs or working on the project

Newcomers workshops

Offer starting points for contributions

Publicize a list of things where help is wanted

regularly post specific "help wanted" requests

Remove use of autotools

Rewrite in python? (please don't do that!)

Rust rewrite

Show issue that are easy for beginners to do to start their journey of being curl contributors.

Some guide "How get started" would be useful.

Some reminders (how? I don't know) to contribute (time or money)

Start making mistakes

stay on github

Stop fixing your bugs as fast ;) curl's just functioning for my needs.

Stop working I guess

Stop working so well, I guess?

Suggest test setups With low threshold

Support more languages (Dart, Closure etc)

swag

Switch to codeberg, but more realistically it's more on my side than on side of curl project itself.

The curl project is already a model of good management: it is as approachable as any open source project could possibly be, I think.

the Debian and Ubuntu packaging are not optional here; (RedHat variations also important). these packages, their certs, libs and defaults are the pain point .. not the curl devs fault!

The honesty of the project and this survey is making me consider to contribute, actually.

The laws of time and space? I would contribute more if I had more available time

The only way to get contributions from me is to have more bugs and fix them less :)

There's nothing I need from it that warrants change (although I've been a long time user, so that may skew it a bit)

To need contributions from me, curl could fail to do a thing that I need from it. So far, this hasn’t happened.

Try to understand the underlying needs of populaces and how changes to the code might impact them.

Use Python

use something other than C

Using gerrit(hub.io) for reviews would make me more likely to contribute.

Appendix C

If you miss support for something, tell us what!

"--verify-content-hash sha256:aaaabbbbcccc111122224333

Better ability to extract pieces out of the response headers into the shell environment? -> My 'lazy' idea would be to have a -v but the debug output is JSON and you let jq do all the dirty work. This is to compete with Postman, to build test suites and so on."

"bare-metal" support/ease of portability to new platforms

"Write the data to the file named in the URL (or in redirects if I'm feeling daring), and timestamp the file to the last-modified-date". This is the main reason I'm still using wget.

"A curl repl/shell. I wrote a rudimentary one in the past: https://github.com/port8000/curlsh/blob/master/curlsh

The idea is to have a persistent set of headers and connection parameters and then be able to send several requests with them. E.g., testing a REST API, what I want is a tool that helps with setting common parameters and lets me issue many similar requests in a short time."

A shell script friendly (awk-able) progress meter for the curl command

A short version of the --data-urlencode command line switch

"A shortcut for -Hcontent-type:application/json

An easier way to take a POST {...} request from some API documentation and run it in curl"

Ability to pretty print JSON without needing to pipe to a tool like `jq` (that may not be installed on a system)

Allow --etag-save to create leading missing directory components. The ability to encode option arguments in a manner suitable for -K configurations.

better MQTT support

Better multipart form handling in libcurl api. Large multipart file upload (not put) might eat a "lot" of ram. Ofc it is better to use PUT and direct stream write, but still some old platforms need these multipart forms. + torrent support + unrar seek + zip support would be nice without downloading the content.

Better support of reading a list of urls from a file. $(cat FILE) is annoying

BitTorrent Protocol

cleaner, easier to use, intuitive CLI commands

Clearer documentation of what features are expected to be provided by curlitself vs. what should be accomplished by using shell-scripting on its output.

CLI protocol debugging/dumping with more control than -vvv

Color output. Easier to read verbose output.

Configuring which HTTP response codes are considered success/error for -f --fail

Curl cli to return status code simply

Detailed TLS configuration, especially configure curves for ECDH and ECDSA separately

Directory listing over SMB

Download file checksum handling.

Easier session handling from CLI: a single option to maintain state (cookies, HSTS, TLS session, proxy settings...) given a (temporary) directory"

Oblivious HTTPS

ENS (Ethereum Name Service)

Error and timeout handling seem to be stuck in 2000's shell-script backwards compatibility hell. Not much of a problem for me personally, but I often see other people's work that's completely oblivious of error handling.

generated cookie for https

HTTPS connectivity debugging checklist: does DNS resolve, TCP ack, TLS certificate okay, OPTION work, GET work

I have a homegrown tool that helps me masquerade as different other tools by using a template of request headers. Lynx headers look different than wget or Firefox or Chrome, etc. I'd definiteLY like to see similar in curl, but don't really need it

I want Japanese documents.

I wish the default behavior when GETting a binary was to drop it on disk. That's the only reason 'wget foo.tgz" is still ingrained in my muscle memory .

I would love sort of an implementation of subresource integrity in curl such that curl could verify the digest of a response before outputting it.

interactive connection sessions like openssl s_client (the syntax of that tool is somewhat of a nuisance)

It would be great to have some connection performance degradation mechanism, which could detect cases when connection throughput drops over time and reconnect using a fresh connection.

I miss a way to specify default curl options, which all newly created curl handles will use by default when they are created (i.e. in some callback passed to curl global init function)."

JSON for GET requests

ldap add, modify, delete

Machine-readable, detailed information what went wrong instead of just exit codes in curl (e.g. a file of JSON log events), OpenTelemetry Traces

Mainly try to keep the feature creep in check or it will turn into bitrot.

MASQUE

Mastodon's http signature draft

maybe jmap?

Metalink (I know, libmetalink got abandoned, that's why curl abandoned metalink...)

More browserlike defaults, like following redirects or upgrading security in response to upgrade-insecure headers.

More cleverness in .curlrc - most importantly site-specific options. Think .ssh/config

More intuitive sending of json payloads to rest APIs.

More quality of life features regarding easy and non verbose API testing

MQTT over TLS

Multi-threaded/Parallel Downloading

NSS TLS backend, for Firefox/Tor Browser emulation in curl-impersonate, due to hostile websites that dislike machine reading

Overwrite DNS by setting where a CNAME points to.

Probably a little more verbosity regarding ssl connection setup.

Reduce a set of http headers to the minimal set required to get the same response. Use case: reduce command obtained from Firefox's "copy as cURL"

rsync support (I have been requesting this for 10 years now, no worries ;-)

S3:// and gcs://   examples, workarounds, whack-a-mole etc

sftp speed is sloooow.

Some easier way to handle getting an OAuth2 token would be helpful.

Some simple options to display "performances" stats like all timers, sizes, transfer speed (what you get with browser network tools)

stabilizing CI jobs and making them finish faster, --xattr to save SHA hash and/or verify download against a hash.

Support for load-testing webservers and proxies

Thanks to Return: To Machine_Sys;Qualification ../.. User_UsineMachine;; To define the timing in it.

The ability to dump the built form before/instead of submitting.

TLS backend selection at runtime, per handle

Tutorial! A tutorial for simple minded people. An official one actually (I think most of my knowledge about curl I obtained from other websites, which is weird.)

url encoding on the CLI

wasm

we are stuck on an old version because we need HTTP1.1 pipelining

would love a revived classic Mac OS port; libs it depends on seem to no longer exist on the internet. relying on MacRelix as an operating environment instead of GUSI might be preferable.

Would love to see a curl-driven fully-FOSS competitor to Postman and Insomnia. Even if it had way fewer features than its competitors, I just want a tool that doesn't feel like it's constantly trying to extract a monthly subscription fee from me.