curl user survey 2020 analysis

“I’ve encountered only one curl bug ever”

summary and analysis by Daniel Stenberg

version 1.1 - June 22, 2020

Table of Contents

About curl

Curl is a mature and well established open source project that produces the curl tool and the libcurl library. We are a small project, with few maintainers, with little commercial backing and yet we're over 22 years old and we have gathered help from almost 2,200 contributors through the years. Our products run in several billion Internet connected devices, applications, tools and services. *curl is one of the world's most widely used software components. Possibly even **the** most widely used component!*

See https://curl.haxx.se for everything not answered in this summary.

Survey Background

We do this user survey annually in an attempt to catch trends, views and longer running changes in the project, its users and in how curl fits into the wider ecosystem.

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 subset of users that we happened to reach. We simply have to work with what we have.

This year, the survey was up 14 days from May 18 to and including May 31. This was the 7th annual survey as the first one ran in 2014.

The survey was announced on the curl-users and curl-library mailing lists (with one reminder), numerous times on Daniel's twitter feed (@bagder) and on Daniel's blog (https://daniel.haxx.se/blog). The survey was also announced on the curl web site with an "alert style" banner on most pages on the site that made it hard to miss for web visitors.

On the Google thing

We use a service run by Google to perform the survey, which leads to us loosing 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 not found a compelling and competitive alternative provider for the survey.

Some respondents have spent a share of their valuable time expanding on "corporate greed" and how we are stupid and lazy who use this service for the survey in some of the free-text input fields. To those friends, I would rather urge that you instead focus that energy on helping us to use an alternative service next year. My guess is that it won't happen.

Number of responses

We were lucky enough to get submissions from 930 respondents this year, up 27% from 732 last year. I am very happy with this turnout as this is certainly more than we use to get:

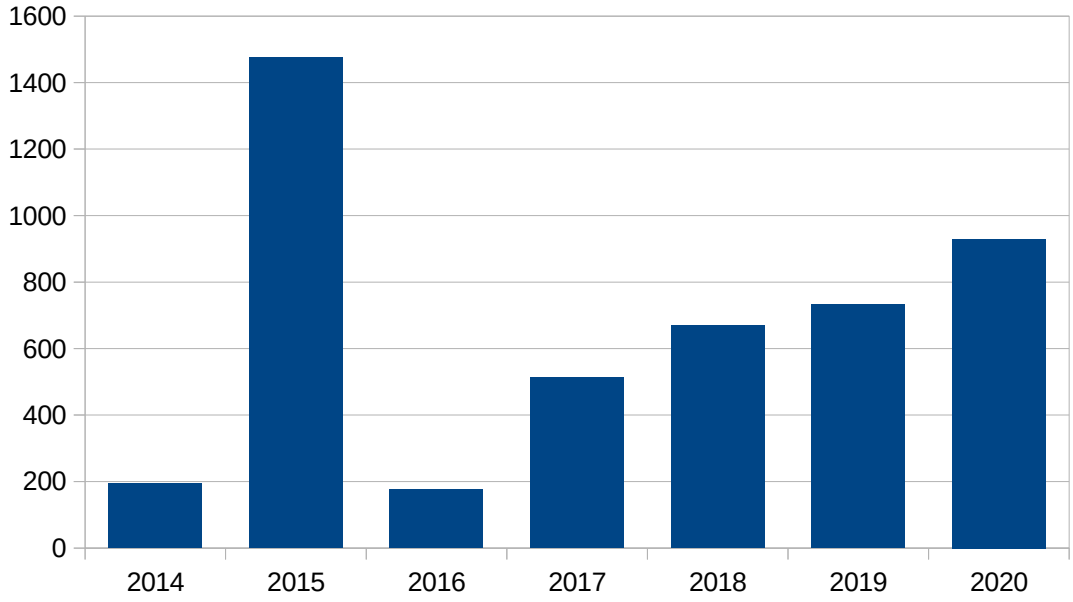


Figure 1: Number of responses

It is unclear what made us reach more users this year or what we can do to reach even more next year.

Returning respondents?

(n = 912)

To understand trends or shifting opinions over time, it is helpful if we can ask the same users over the years and see what they think we improve or worsen. Unfortunately, we seem to keep reaching out to people who never participated in the survey before. Only about one out of 8 respondents say they answered the 2019 survey.

The fact that mostly a new set of users answer the survey every year helps us make sure that the answers we get are not just the same set of people year over year, but instead as we see similar results year-to-year with ¾ of the respondents replaced we can possibly consider the results to carry more weight than otherwise.

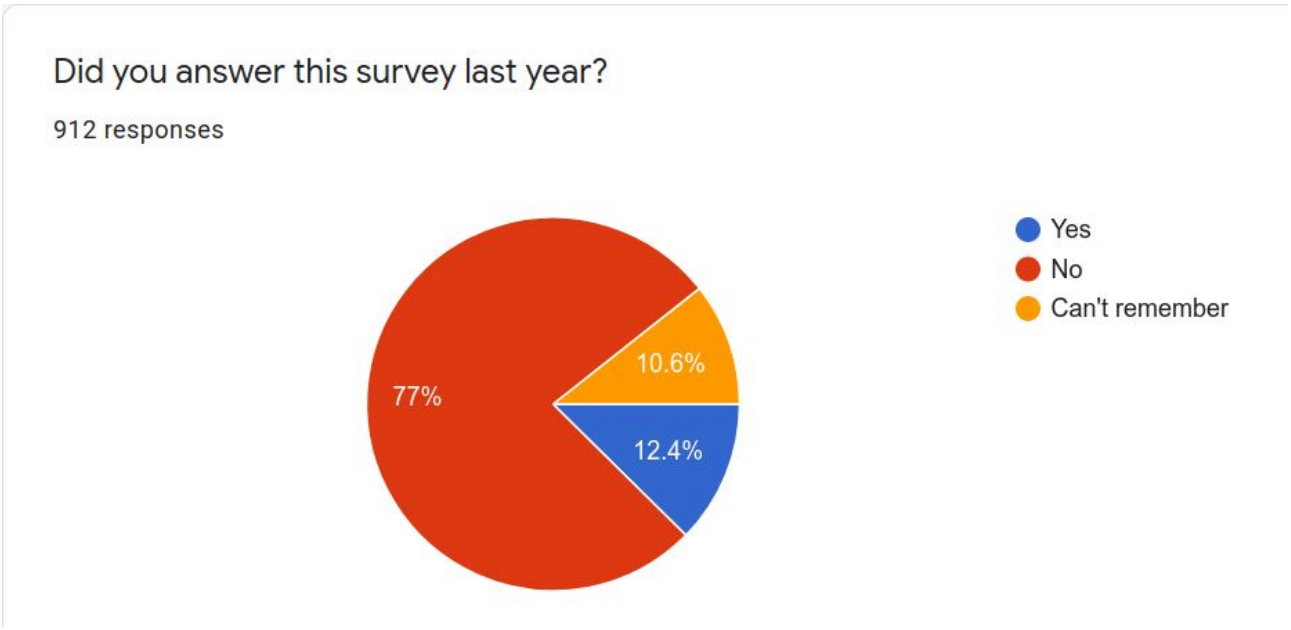


Figure 2: Answered last year's survey

The upside of this is of course that we are reaching out to even more users and that’s also good.

Where are the submissions coming from?

(n = 914)

The question actually asks “where do you live?” and shows that we have a massive European slant on this survey. More than 57% live in Europe:

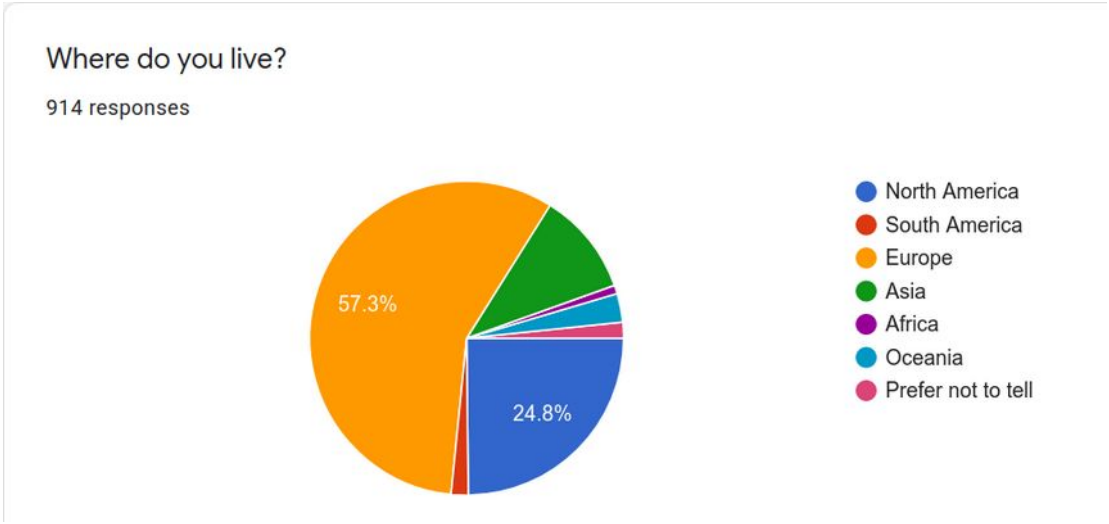


Figure 3: Where does the user live pie chart

This pattern follows previous years very closely. With Asia reaching a little over 10% as the biggest change:

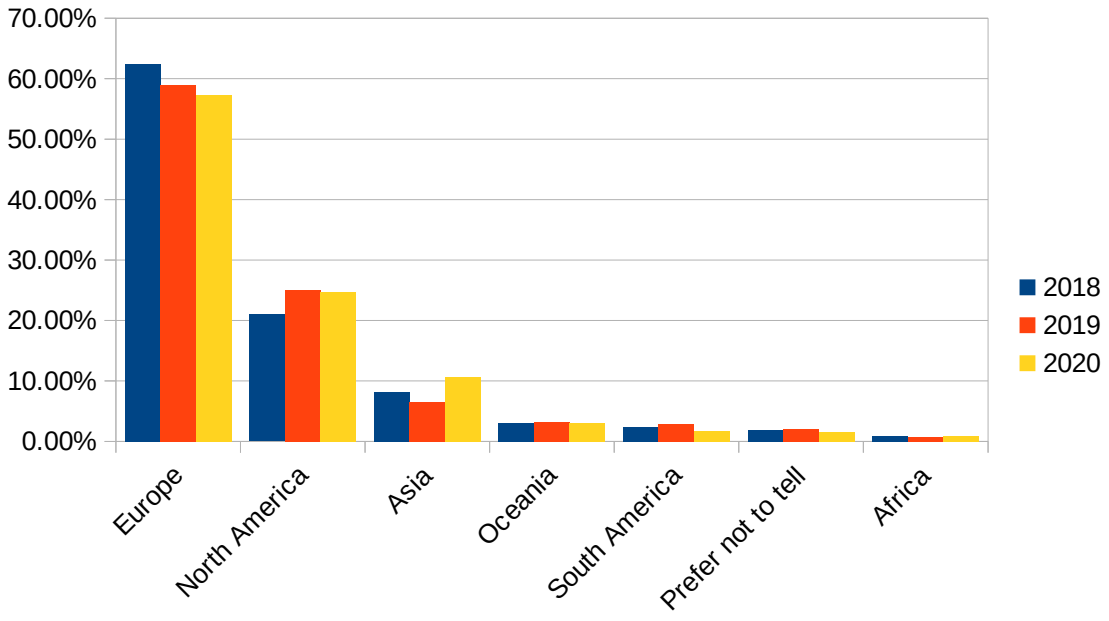


Figure 4: Where does the user live

I think it shows that open source participation in general is strong in Europe, and that my own European presence (time zone and otherwise) shows. Looking the top contributors and maintainers in curl, there’s a strong European focus there as well.

What kind of users?

(n = 907)

What are the kinds of users who fill in the survey? Even though we seem to have reach mostly new users compared to last year, it seems the distribution over various (self-declared) roles were still roughly the same. 31.5% are “backend developers” (35.8% last year) and 20.3% (15.3) are sysadmins:

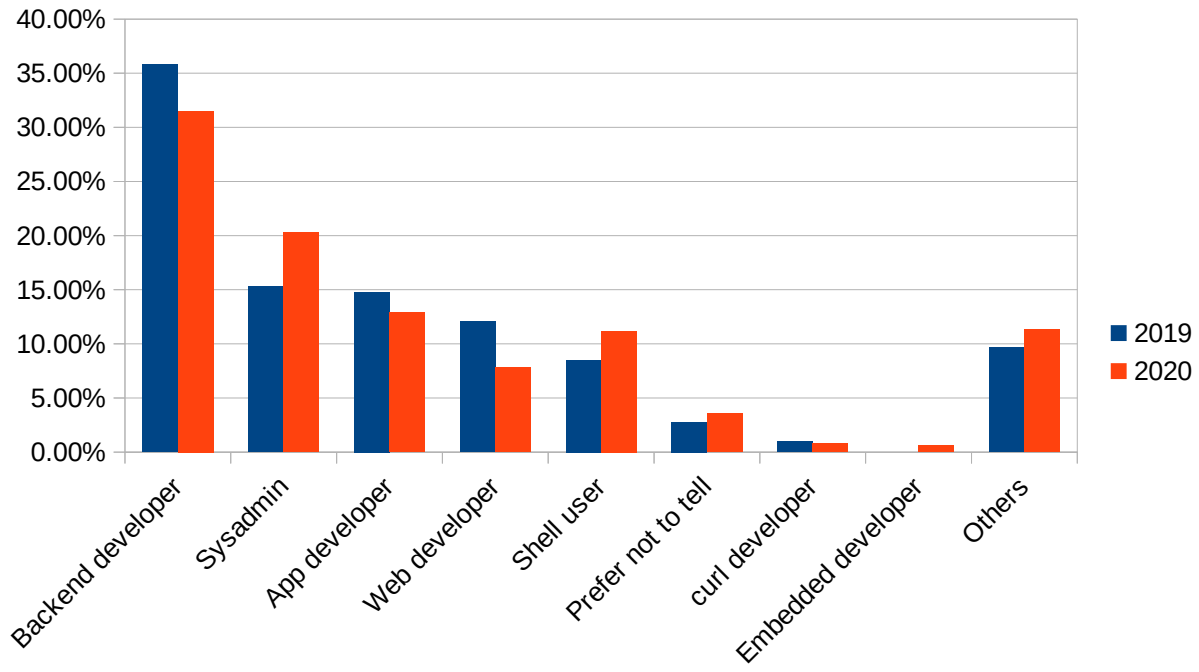


Figure 5: Kind of user

The amount of “others” increased from 9.7% to 11.3% and I think it is perfectly understandable that so many users think of themselves as not fitting one of the suggested labels. We are all unique.

(This question was introduced in 2019)

Protocols

(n = 920)

This is possibly the number one graph and set of numbers I often come back to over the year, when new people discover that curl actually supports a particular protocol or question why we still support one of them.

Two protocols are by far the most popular ones: HTTPS and HTTP.

Four additional protocols are used by 10% or more of all users: FTP, SFTP , FTPS and FILE (in popularity order). Among these top-6 protocols, we see very little variation between the years. They seem to be used at roughly the same proportion year by year.

Six protocols can boast 10% or more of the users, and 15 protocols have 5% users - or more. They are in order: SCP, SMTP, SMTPS, TELNET, IMAP, IMAPS, SMB, GOPHER and LDAP.

This year the least used protocol turns out to be RTSP at 2.3%. Last year's last position DICT (then at 1%) climbed to second to last this year with 2.5%.

The order among the top six protocols only changed in one way: SFTP overtook FTPS and is now the 4th most popular curl protocol.

Table 1: Protocol usage share 2020

#	Protocol	User share 2020	User share 2019
1	HTTPS	**97.3%**	96.7%
2	HTTP	**93.5%**	92.8%
3	FTP	**30.5%**	29.2%
4	SFTP	**16.5%**	13.5%
5	FTPS	**15.1%**	16.0%
6	FILE	**11.5%**	9.9%
7	SCP	**8.7%**	7.8%
8	SMTP	**6.8%**	5.5%
9	SMTPS	**6.0%**	4.5%
10	TELNET	**5.5%**	4.8%
11	IMAP	**5.4%**	4.7%
12	SMB	**5.3%**	3.4%
13	IMAPS	**5.3%**	4.8%
14	GOPHER	**5.2%**	2.8%
15	LDAP	**5.1%**	5.0%
16	TFTP	**4.8%**	4.0%
17	POP3	**4.2%**	3.2%
18	LDAPS	**4.1%**	4.5%
19	POP3S	**3.8%**	2.8%

20	RTMP	**3.2%**	1.9%
21	MQTT	**3.0%**	n/a
22	SMBS	**3.0%**	1.4%
23	RTMPS	**2.7%**	n/a
24	DICT	**2.5%**	1.0%
25	RTSP	**2.3%**	2.1%

A general observation on this year's protocol summary is that almost all protocols increased their user share! HTTPS reached its highest level in any survey. Only two protocols are used by a less share of curl users in 2020 than in 2019; FTPS and LDAPS. The top 10 of the greatest "climbers" this year in user-share compared to last year's share are:

Table 2: Most growing protocols 2020

#	**Protocol**	**Compared to 2019**
1	DICT	250%
2	GOPHER	186%
3	RTMP	168%
4	SMB	156%
5	POP3S	136%
6	SMTPS	133%
7	POP3	131%
8	SMTP	124%
9	SFTP	122%
10	TFTP	120%

MQTT support has been added to curl recently but has not existed for a full year yet and it is still considered experimental and is not enabled by default. RTMPS was previously mistakenly missing among the alternatives.

Platforms

(n = 916)

One of curls strongest selling points is its availability on many platforms, and users take advantage of this. This year we might see a small trend change in that the number of single-platform users increased and the 2-3 platforms team shrunk a little. It might of course also just be a little single year blimp in the data. Next year we'll know.

54.9% of the users in 2020 say they use curl on 2-3 platforms, 28.3% on a single platform and 10.3% on 4-5 platforms. The >5 platforms group is at 6.6%.

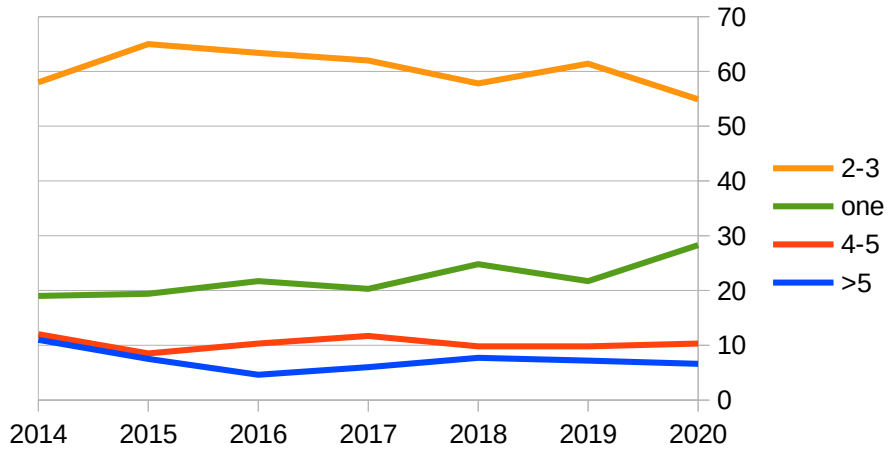


Figure 6: Number of platforms

The particular platforms that are used in 2020 also roughly matches the ones used previous years. Here's a graph showing the ones with 4% or higher user share:

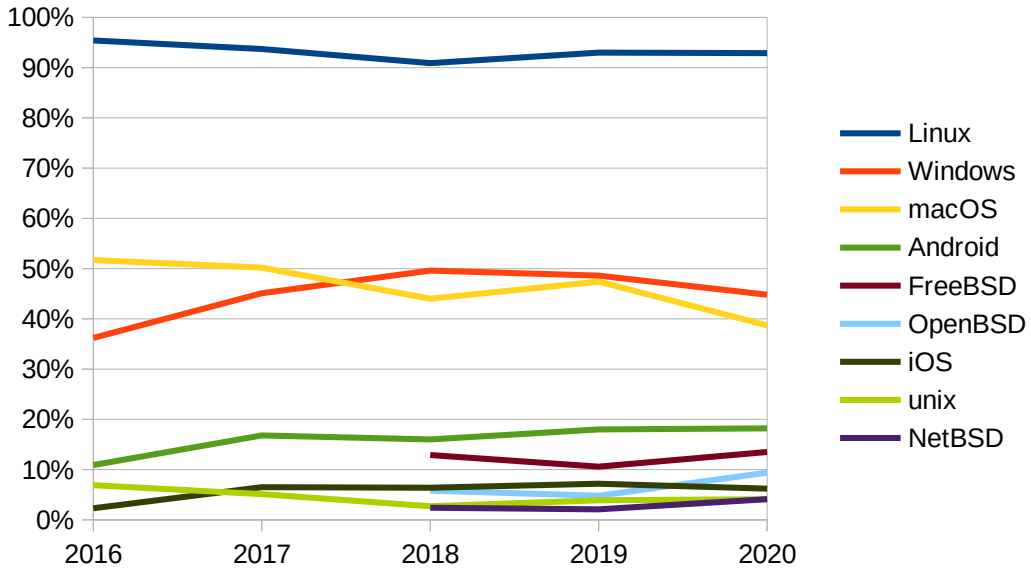


Figure 7: Which platforms are used

The exact platform distribution in 2020:

Table 3: Platform usage

#	**Platform**	**2020**
1	Linux	92.9%
2	Windows	44.8%
3	macOS	38.7%
4	Android	18.2%
5	FreeBSD	13.5%
6	OpenBSD	9.4%
7	iOS	6.2%
8	unix	4.1%
9	NetBSD	4.1%
10	Solaris	3.7%
11	AIX	1.8%
12	VMS	1.5%
13	HPUX	1.3%
14	RTOS	1.2%
15	IRIX	0.9%

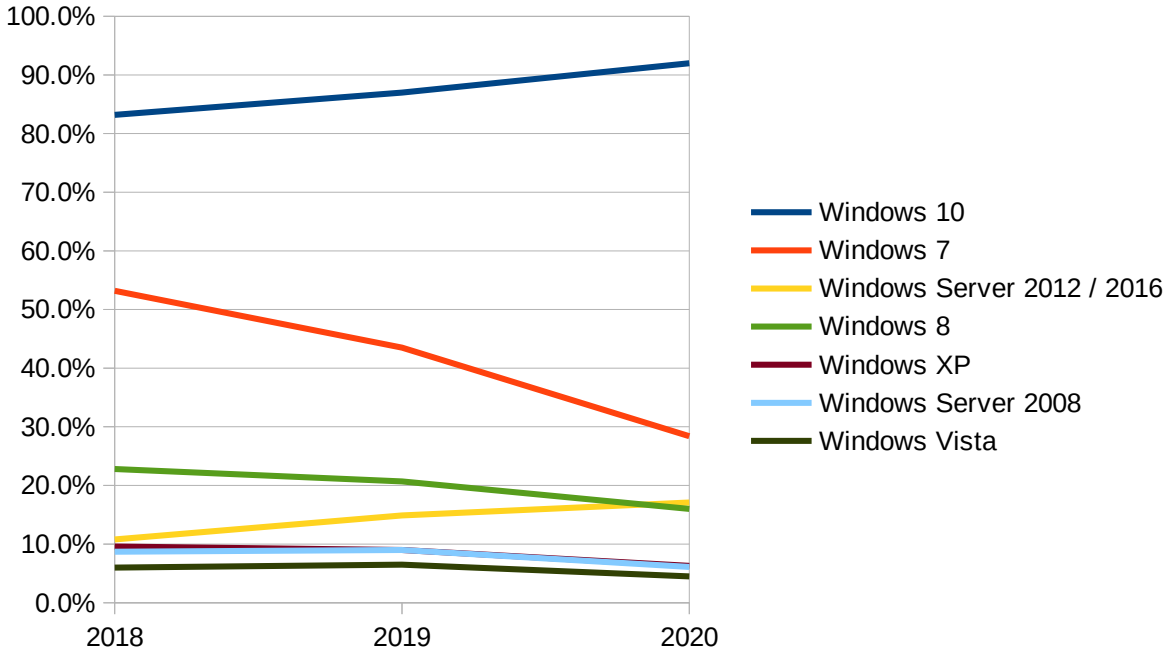
The “other” write-in option got a lot of answers, including Illumnos, Haiku, FreeDOS, Junos, WSL, OpenStep, OpenIndiana, IBM i, VenusOS, MorphOS. Genode, game consoles, Nintendo Switch, AmigaOS, SCO OpenServer.

What Windows versions

(n = 426)

There are two obvious trends here. Windows 10 (92.0%) is increasing its share as the primary Windows version people run curl on, and Windows 7 (28.4%) is rapidly decreasing. This graph below show the versions that got selected by 4% or more of the users. In the third place we got server 2012/2016 (17.1%) pass Windows 8 (16.0%) this year.

I believe some users this year was a little bit confused about how to answer when they've use curl in WSL on Windows 10, which apparently to a certain amount of users is not plain Windows and yet not plain Linux either...



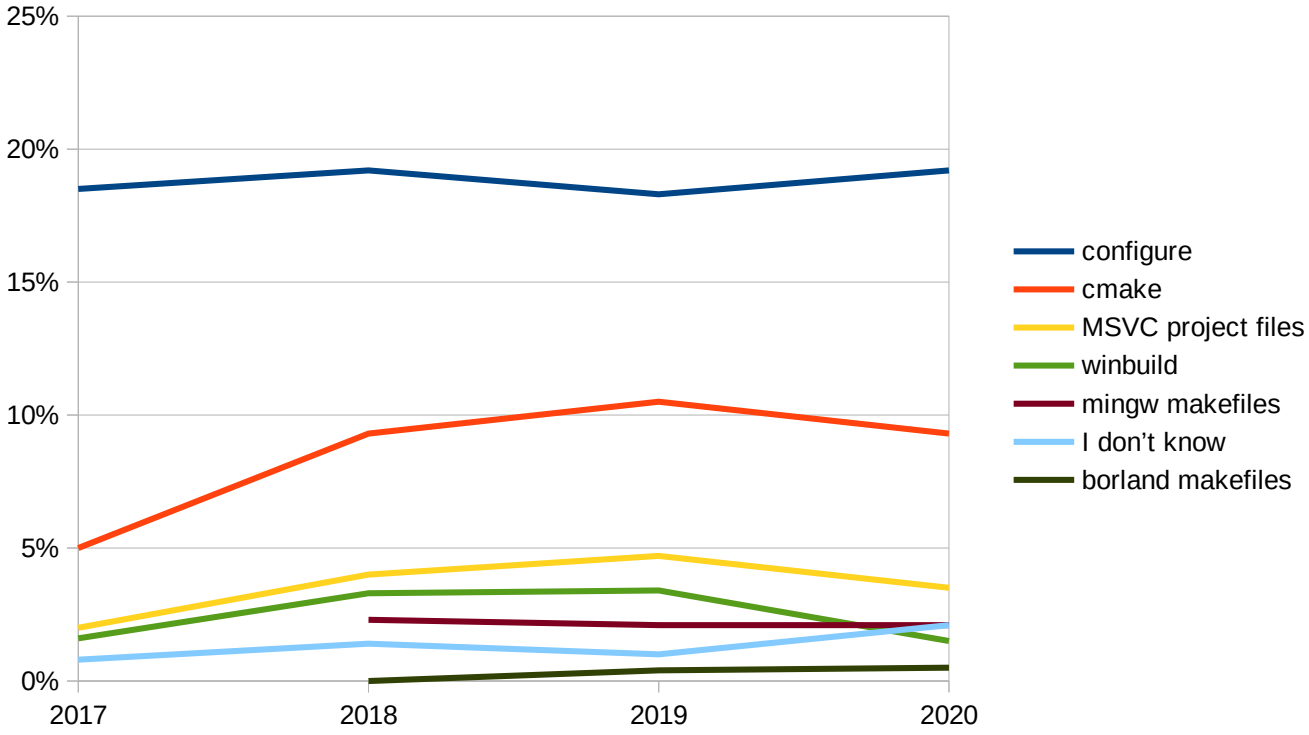
Among the still selected but perhaps believed-to-be-extinct versions in the bottom (less than 4% user share) we see:

#	**Version**	**User share 2020**
8	Windows Server 2003	2.6%
9	Windows 2000	2.1%
10	Windows 98	1.6%
11	Windows 95	1.4%
12	Windows CE/Embedded	1.2%

Building curl

(n = 912)

This question of course here for us to get a clue as to what relative importance and use the different build options we provide in curl have. Most users (75.7%) who answered this question said “No” – they don’t build curl at all.



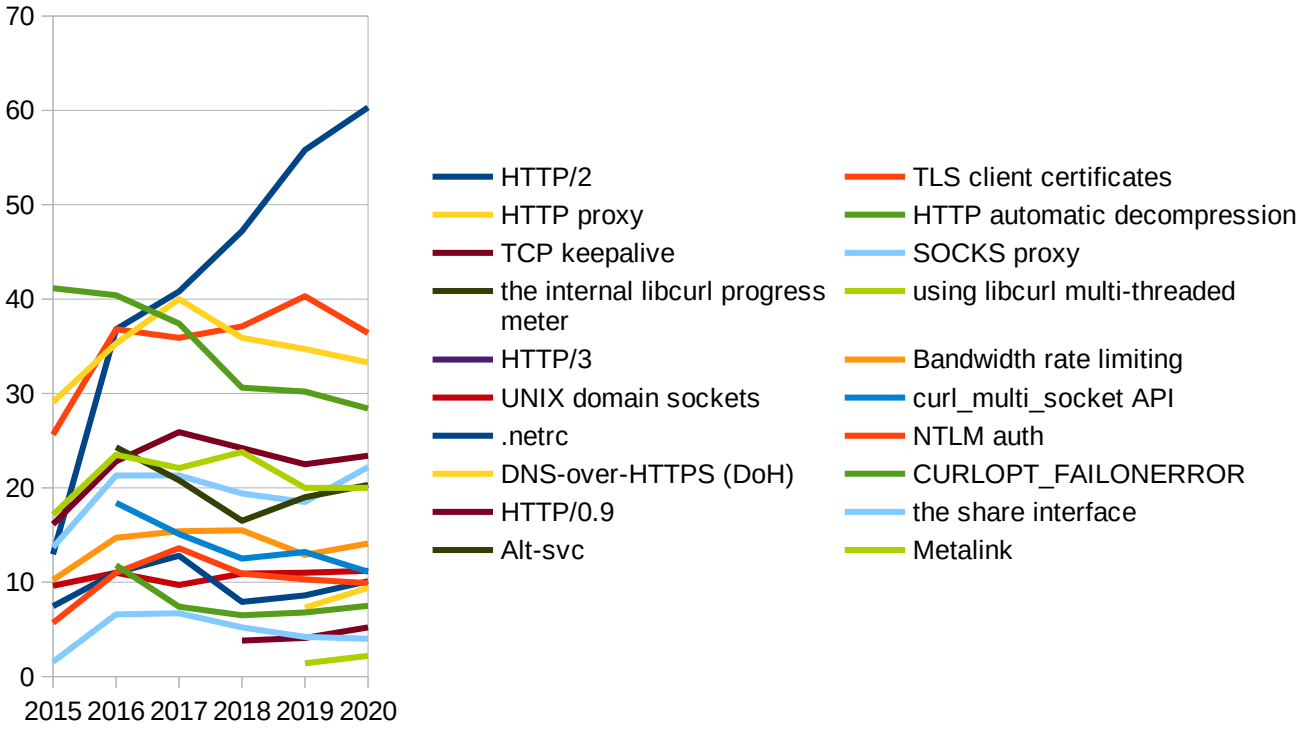
The configure/autotool setup (19.2%) remains how the majority of us who build curl does it, with cmake lingering at roughly half that share (9.3%). Then there’s another fairly big step down to third place where 3.5% of the users use our provided MSVC project files, 2.1% on the mingw makefiles and 1.5% of the users build using the “winbuild” makefiles.

Several persons filled in “gentoo” but I would qualify that as not building curl yourself as you’re just letting something else build it, and the same goes for “yocto”, “homebrew” and other distro(-like) builder concepts.

Features

(n = 775)

HTTP/2 celebrated five years since the RFC was published this spring and it has been racing the curl feature chart. Again another question to figure out something of a relative feature popularity among various things curl can do and how they vary over time.



A flurry of colored lines in here, but I think the main take-way is that most features remain at a fairly stable level, while HTTP/2 is growing, automatic decompression is shrinking and HTTP/3 immediately popped in at a rather high position.

Now, a large amount of sites support HTTP/2 these days and curl does HTTP/2 by default for HTTPS since a good while back so I believe the HTTP/2 number should probably be close to the HTTPS share (some people of course still build and use curl without HTTP/2 support built-in), but this question also reflects people's awareness (or lack thereof) of what curl features they use.

The exact numbers this year compared to last year, the ones that shrunk this year are marked in red:

#	Feature	2020	2019
1	HTTP/2	60.3%	55.8
2	TLS client certificates	36.4%	40.3
3	HTTP proxy	33.3%	34.7
4	HTTP automatic decompression	28.4%	30.2
5	TCP keepalive	23.4%	22.5
6	SOCKS proxy	22.2%	18.5
7	the internal libcurl progress meter	20.3%	19.0
8	using libcurl multi-threaded	20.0%	20.0

9	HTTP/3	14.2%	
10	Bandwidth rate limiting	14.1%	12.9
11	UNIX domain sockets	11.2%	11.0
12	curl_multi_socket API	11.1%	13.2
13	.netrc	10.1%	8.6
14	NTLM auth	9.9%	10.3
15	DNS-over-HTTPS (DoH)	9.4%	7.3
16	CURLOPT_FAILONERROR	7.5%	6.8
17	HTTP/0.9	5.2%	4.1
18	the share interface	4.0%	4.2
19	Alt-svc	3.1%	
20	Metalink	2.2%	1.4

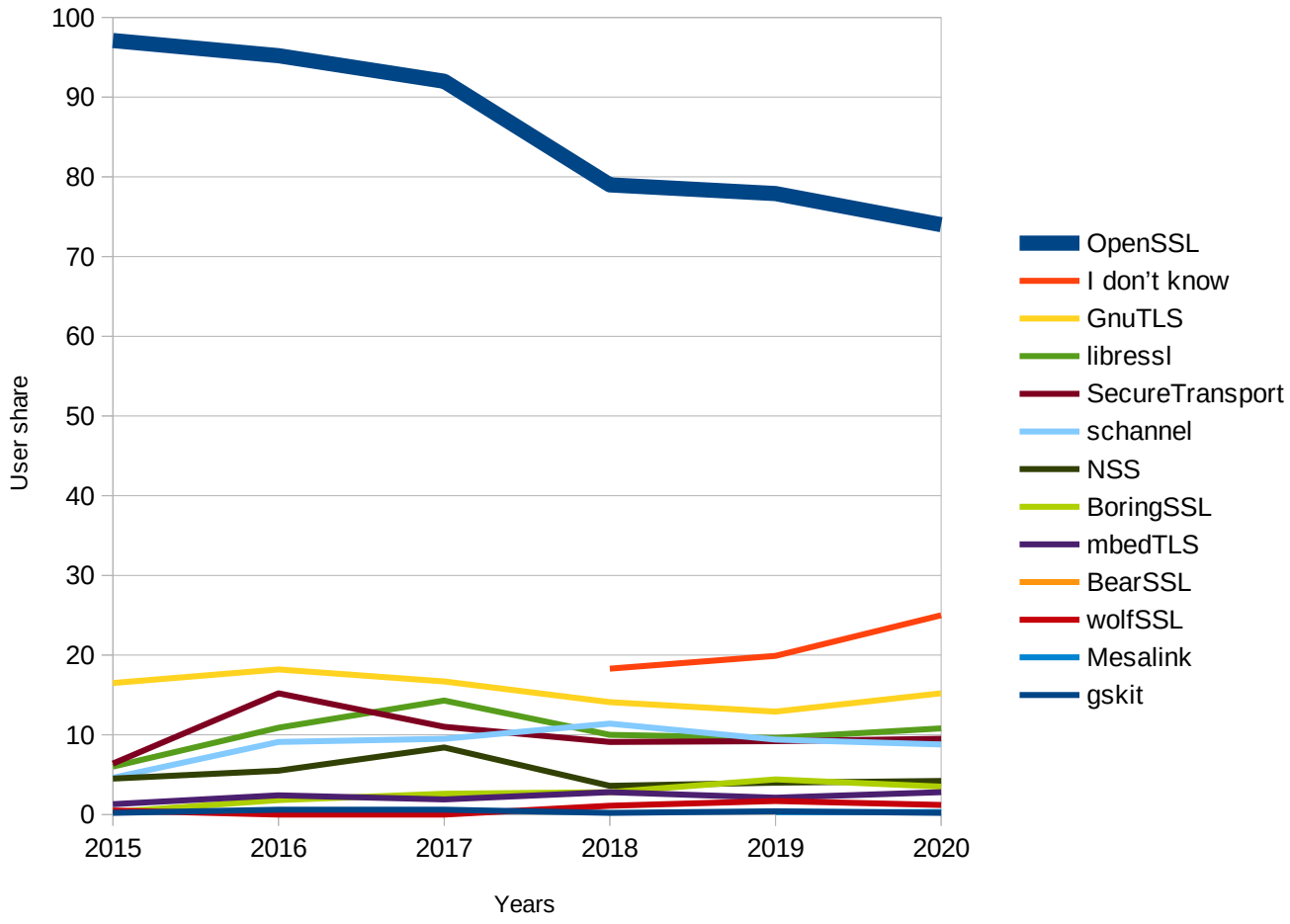
SSL backends

(n = 888)

While 25% of the respondents didn't know (up from 19.9% last year), that still leaves 75% who know. This is also a multiple choice question since many users use several different curl builds with potentially different backends.

Windows ships curl using schannel since two years anow, macOS ships curl using libressl now but used Secure Transport before when they switched over from OpenSSL. Redhat dropped NSS from curl for OpenSSL and GnuTLS options are provided on Debian and Ubuntu at least.

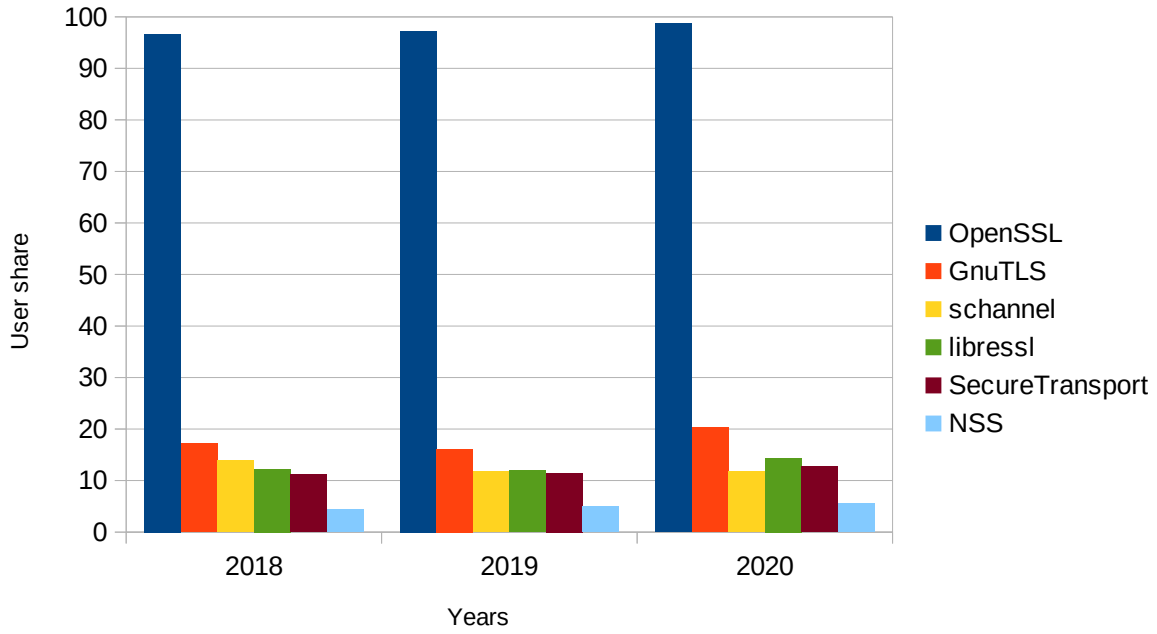
In the curl project we've dropped support for two TLS libraries and added more over the recent years. So how does all this play out in the graph? Not too visible. OpenSSL (74%) is shrinking a bit but otherwise things seem to remain among the respondents.



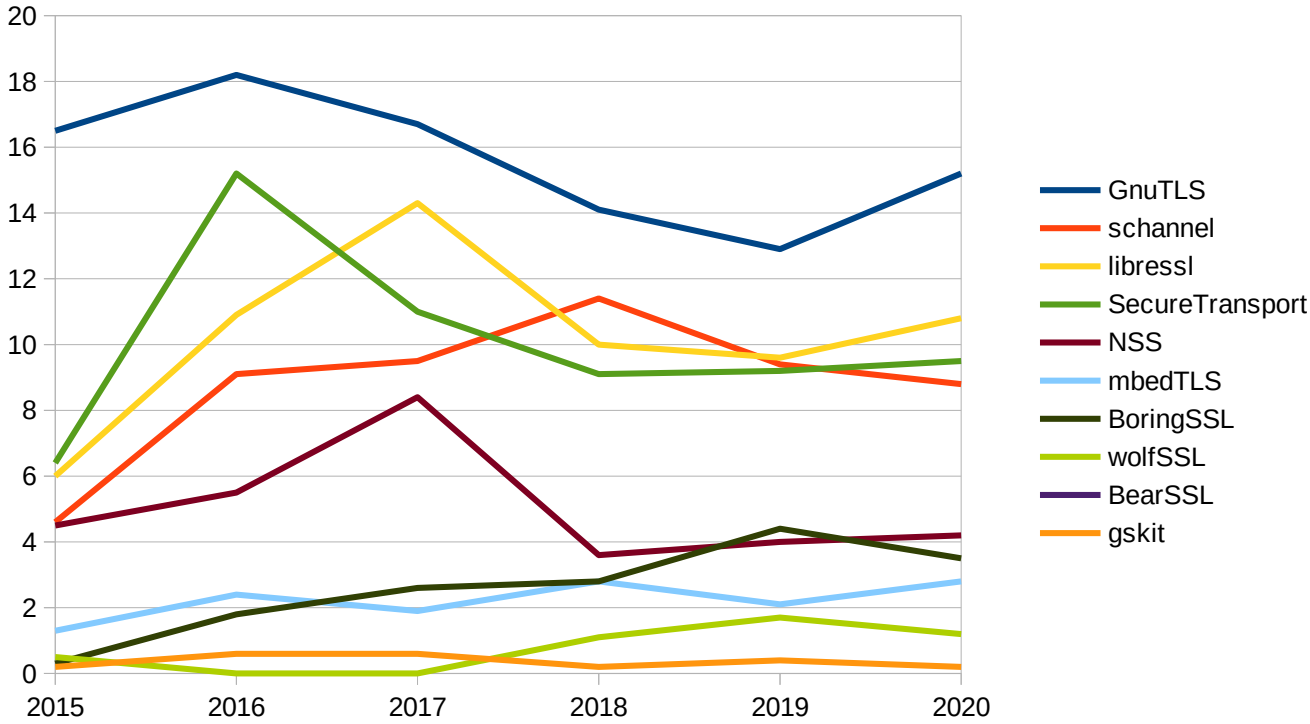
The drop for all backends in 2018 can probably be explained by us adding the "I don't know" answer that year. Previous years those users probably didn't respond to the question, giving all the backends a relatively higher share.

With 25% saying "I don't know" this year and 74% marking OpenSSL, it could mean that 98.7% of those that do know use curl with OpenSSL… This could then mean that 2020 OpenSSL has a larger user share than ever before as 2018 (96.7%) and 2019 (97.3%) don't get that high when counted as a share of those who do know…

Adjusted for the I don’t knows, showing the top six backends the three last years:



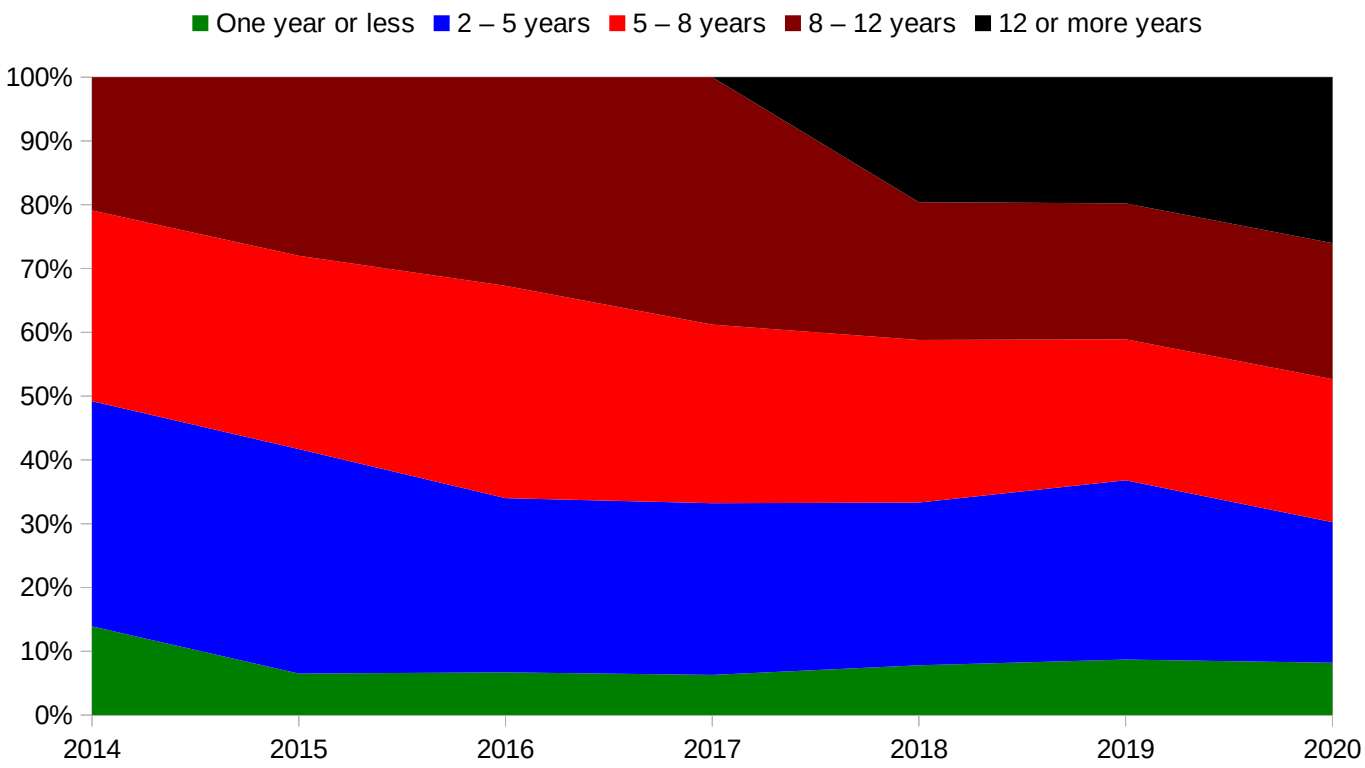
If we remove the dominant player from the SSL backend graph and the “I don’t knows” answers, we can zoom in a little bit on the others:



Years of curl use

n = 910

How well do we attract new users and how well to we retain old users? The data seems to suggest we're doing ok. Most notably the 12 years or more category grew a lot this year.



Group	User share 2020
One year or less	8.2%
2 – 5 years	22.0%
5 – 8 years	22.4%
8 – 12 years	21.3%
12 or more years	26.0%

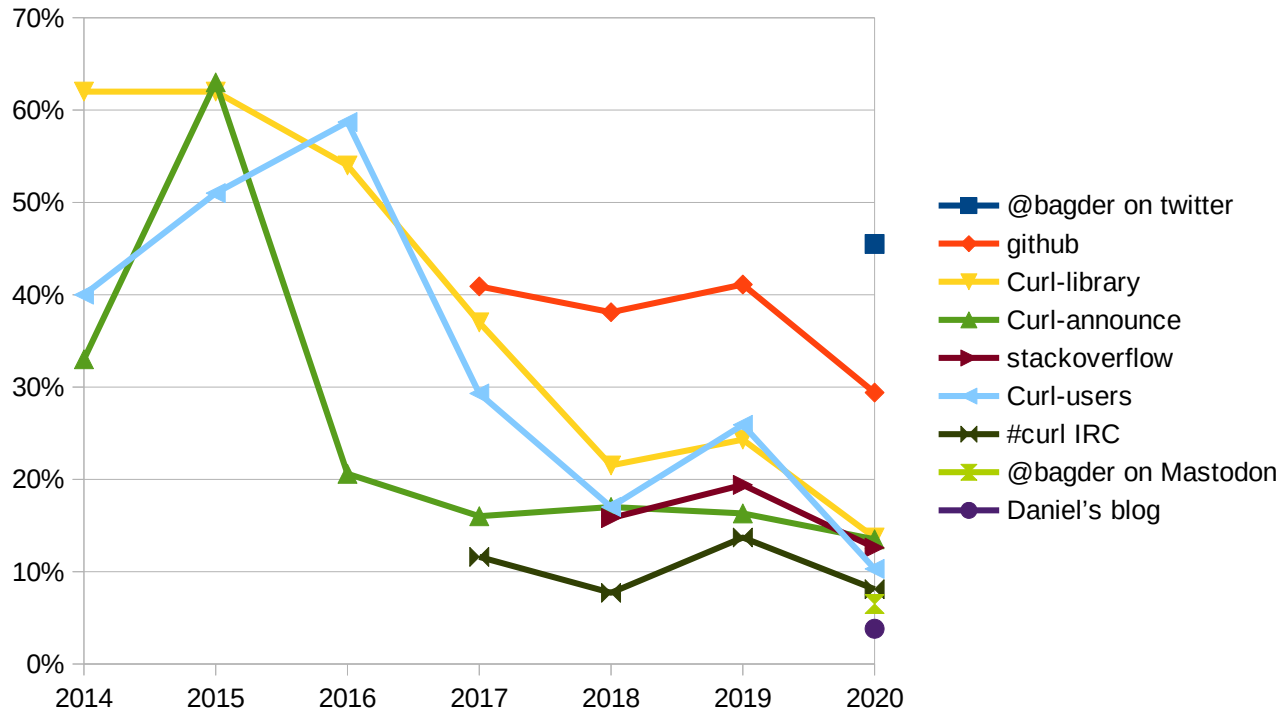
Participating channels

(n = 446)

This question has one of the most notable changes over time. I think we can safely say that the trend is here that respondents to the curl user survey are way less subscribed to any of the mailing lists today than what they were a few years back. This trend has clearly been going on for a while.

I complicated the picture a little bit by adding a new option ("@bagder on twitter"), and I counted the write-ins and summed them up under "@badger on mastodon" and "Daniel's blog". There was no other write-in worth mentioning.

I have some 22,000 followers on twitter and I mentioned this survey numerous times during the polling period. That probably explains why 45% of the answers said they follow me (that's still just 203 persons, not even 1% of my followers).



curl and libcurl are of course used everywhere and virtually by everyone. Maybe we should consider it good that the shares go down as that means we're succeeding in reaching out to users outside of our little bubble when asking for user input! The exact numbers were:

Channel	2020	2019
@bagder on twitter	45.5%	
github	29.4%	41.1%
Curl-library	13.7%	24.3%
Curl-announce	13.5%	16.3%
stackoverflow	12.6%	19.4%
Curl-users	10.3%	25.9%
#curl IRC	8.1%	13.7%

@bagder on Mastodon	6.5%	
Daniel’s blog	3.8%	

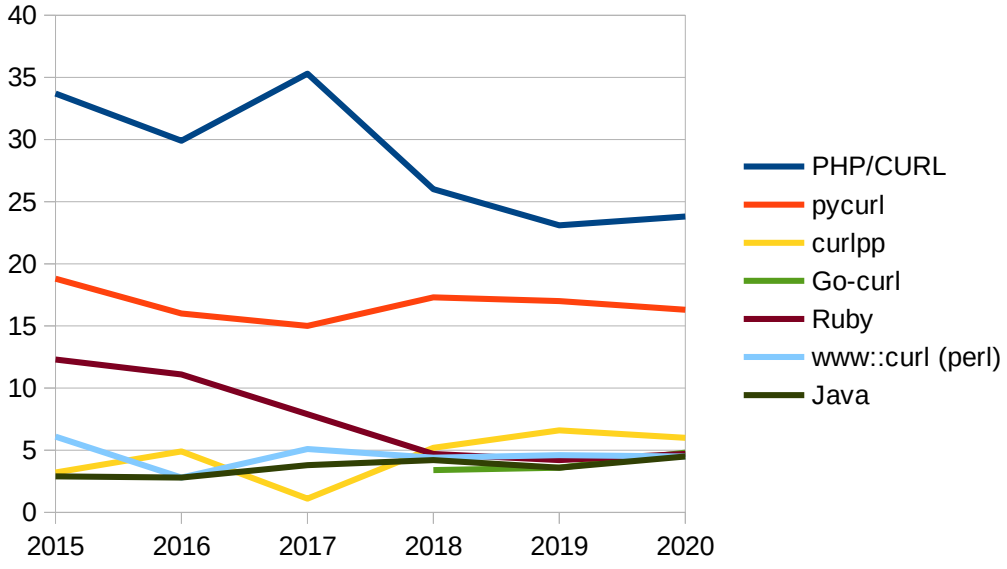
How do you “access” libcurl

N = 848

Among the users who answer to the survey, this question seems to be fairly stable. This year’s number are:

Binding	Share of users 2020
curl	79.7%
plain C	34.9%
PHP/CURL	23.8%
pycurl	16.3%
curlpp	6%
Go-curl	4.8%
Ruby	4.7%
www::curl (perl)	4.5%
Java	4.5%
Node-libcurl	3.9%
Rust-curl	3.9%
Lua	3.5%
.NET core	2.5%
Common Lisp	1.7%
R curl	1.5%
Tclcurl	1.3%

PHP/CURL remains the most used non-native binding in a class of its own. Ignoring the bindings under 4% share this year, the development over the years look like this:

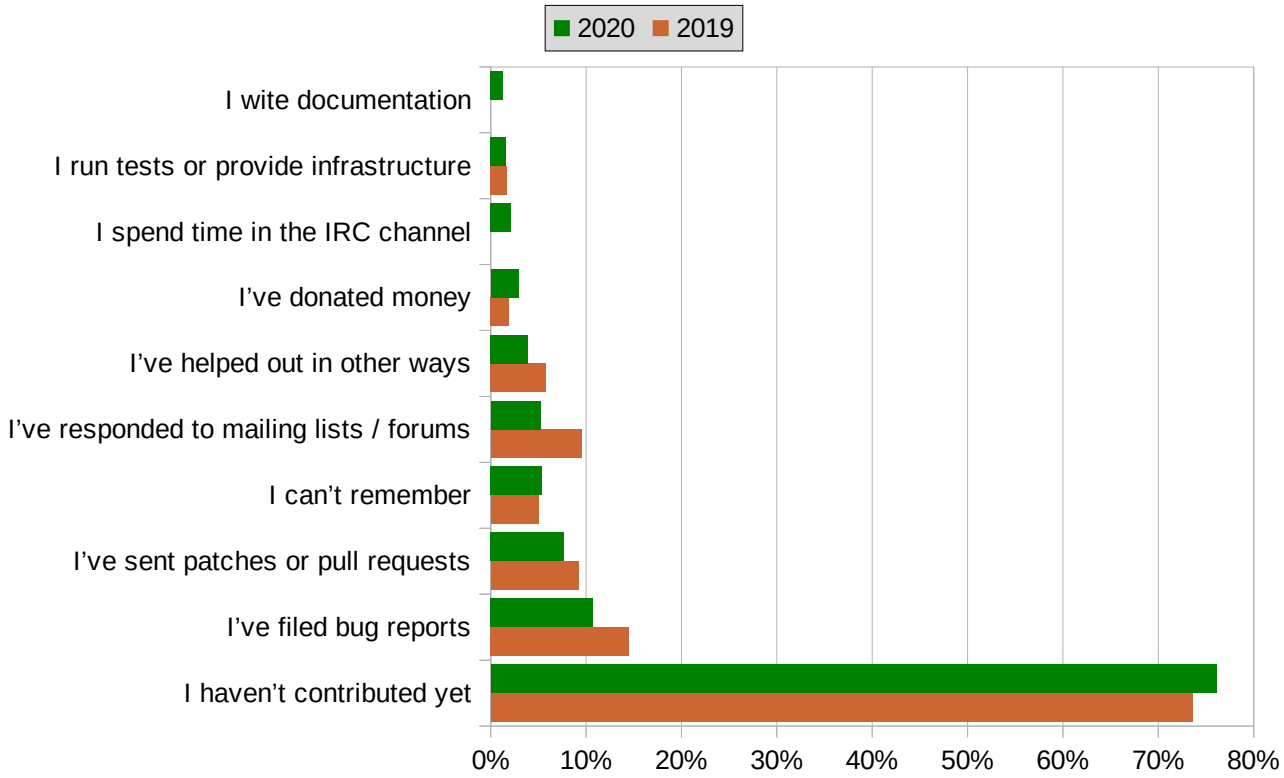


Lots of people also chose “other” and wrote in additional suggestions for bindings they use and we certainly got to see a wide variety there, including: restclient-cpp, libPasCurl, ocurl, hhvm, libgit2, cpr, Apache mod_md, Net::curl:: perl, pgsql-http, Powershell, PureBasic, Guile-curl, freebasic, Fortran 2008, qjs, Xojo wrapper and perl anyevent::yacurl!

Contributions

N = 765

curl is open source and exists only thanks to contributors helping out. How have the respondents helped the project so far?

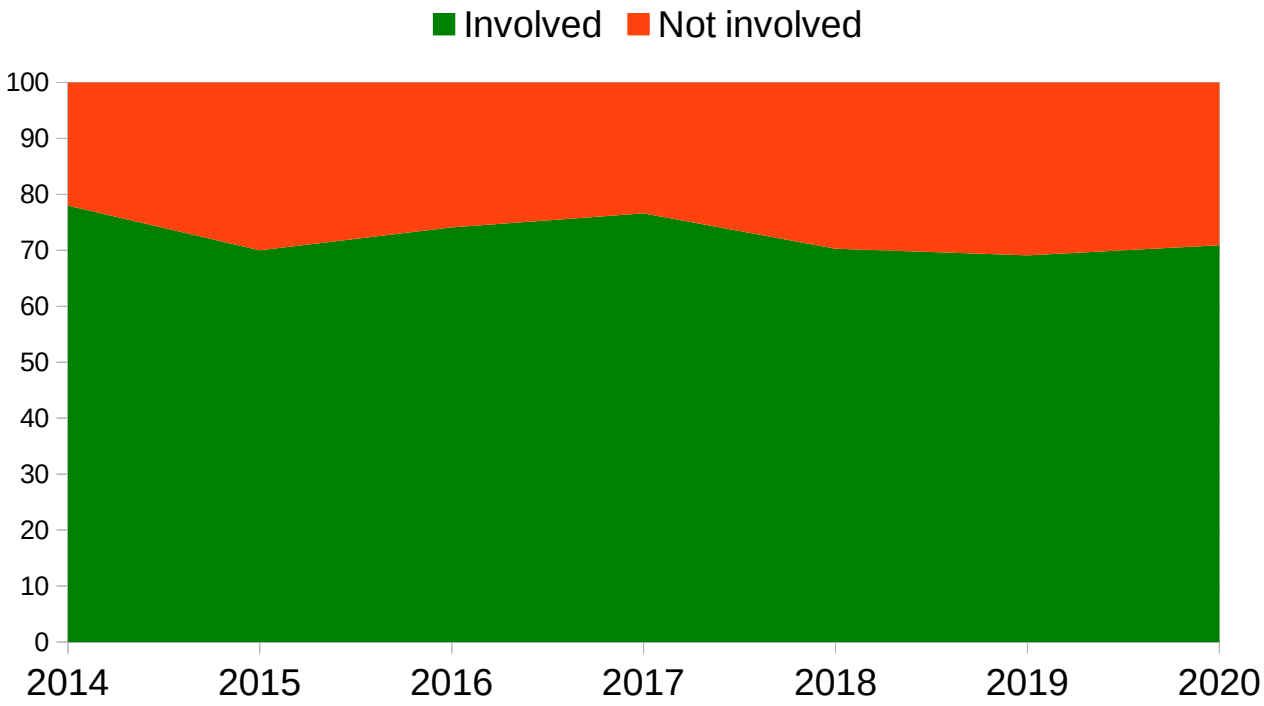


The number of "haven't contributed yet" grew from 73.6% last year to 76.1% this year while the share of bug reports shrunk from 14.5% to 10.7%.

Other projects

N = 908

People who respond in our survey are certainly open source savy and participate in other projects to a large extent. This year 70.9% of the users said they are “involved” in other open source projects, and thereby keeping up the general level from previous years.



Reasons not to contribute to the project

N = 850

76% haven’t contributed to curl but yet 70% are involved in other projects, meaning there are a lot of users answering here that *could* participate if the conditions are right. How come they haven’t contributed (more)?

The top reason (60.9%) among the offered answers to this question remains *Everything works to my satisfaction*. It has always been the by-far most voted excuse. Reason number two and three are similarly reasons that basically says the fault is not ours: (#2) *I don’t have time* (43.5%) and (#3) *I don’t have the energy* (19.0%). At position #4 we have a newcomer this year up from #5 last year *Too hard to get started and figure out where to tweak* (17.8%).

The full results look like this:

#	**Why not contribute?**	**2020**
1	Everything works to my satisfaction	60.9%
2	I don’t have time	43.5%
3	I don’t have the energy	19.0%
4	Too hard to get started and figure out where to tweak	17.8%
5	Things get fixed fast enough	15.3%
6	I don’t know the programming languages used	14.1%
7	My work/legal reasons prohibit me	4.7%
8	I don’t like or approve of GitHub	4.4%
9	I don’t like or use email	1.7%
10	I can’t deal with the tools	1.6%
11	the project doesn’t want my changes	1.3%
12	I find it hard to work with the curl developers	0.8%

With 17.8% (of the ones who don’t contribute more) thinking it’s hard to know where to start, it feels like an implied mission for us. But I really need help and feedback here as I’m at loss as to *how* to improve this.

More than 160 of the answers contained a free text write-in. They can be divided into a few categories:

“**be worse**” (with variations such as “have more bugs” and “break more often”)

“**tell us**” – various suggestions that we should communicate better when/where/if we need help, including marking bugs as “good first issues” and providing “a clear roadmap”

“**nothing**” – basically saying that we do enough but the user’s situation is the limiting factor

the rest – various random suggestions from added protocols, to removing protocols to changing language etc

How good is the project to handle…

Asked to grade the project in these seven different areas between 1 to 5, this is designed to track if the project trends in a different direction or perhaps improves or degrades over time, but here again we get an almost amazingly stable result over time.

The average scores on the different areas ended like this (compared to two previous years):

#	**Area**	**2020**	**2019**	**2018**
1	security	4.4	4.52	4.59
2	giving credits	4.27	4.49	4.4
3	information	4.21	4.26	4.29
4	bug reports	4.2	4.43	4.3
5	patch handling	4.2	4.36	4.38
6	newcomers	3.8	4.09	3.93
7	female and minorities	3.45	3.76	3.6

As you can see in this table, all seven areas scored slightly lower this year than last. I believe the difference to be so small that it’s not significant. We haven’t really done worse and we certainly have not really done better.

We have work to do and we can and should improve in all areas! Tell us how!

Which are the curl project’s best areas?

N = 787

As a project lead I can’t think of a much better result than year after year having these areas voted as the best:

1. the quality of the products

2. multi-platform availability

3. support of many protocols

4. documentation

#	**Areas**	**2020**	**2019**
1	the quality of the products, curl/libcurl	57.7%	56.0%
2	its availability and functionality on many platforms	55.5%	54.9%
3	the support of many protocols	42.8%	34.8%
4	documentation	37.9%	39.0%
5	the libcurl API	34.3%	32.6%
6	the features of the protocol implementations	25.7%	22.8%
7	standards compliance	25.4%	28.9%
8	support of multiple SSL backends	20.6%	19.6%
9	security	18.2%	20.4%
10	footprint of the library/executable	16.1%	15.4%
11	transfer speeds	10.3%	11.2%
12	project leadership	9.3%	13.0%
13	the user and developer community	7.6%	7.9%
14	bug fix rate	5.1%	8.2%
15	project web site and infrastructure	3.9%	2.7%
16	its build environment/setup	3.4%	2.9%
17	welcoming to new users and contributors	2.9%	3.5%
18	Test suite	2.8%	2.1%

Which are the curl project's worst areas?

N = 192

Just a quarter of the amount of people who answered about the *best areas* answered to this. Interestingly, the highest rated answers to this question remain the same over the years but with the little detail that since 2018 number three and four (the libcurl API and the web site) changed place with each other.

#	**Areas**	**2020**
1	documentation	29.7%
2	its build environment/setup	20.3%
3	the libcurl API	18.8%
4	project web site and infrastructure	13.0%
5	welcoming to new users and contributors	9.9%
6	test suite	9.9%
7	footprint of the library/executable	7.8%
8	security	6.8%
9	the support of many protocols	6.8%
10	transfer speeds	6.3%
11	the features of the protocol implementations	6.3%
12	support of multiple SSL backends	5.2%
13	the user and developer community	4.7%
14	bug fix rate	4.7%
15	its availability and functionality on many platforms	3.6%
16	the quality of the products, curl/libcurl	3.6%
17	standards compliance	2.1%
18	project leadership	1.0%

Since we started this annual survey we have enhanced the documentation *significantly* and I would say across the board. All documents are better and more thorough now, written in better language. Still it has remained the number one "worst area" in all surveys since 2014 – while at the same time remaining the #4 voted "best area" … I think this just highlights that doing documentation is very hard.

The build environment's second place I believe is similar. We have improved it over the years and removed some confusion points and reduced duplications. I suspect this is an area where people are opinionated and I suspect it is hard to really provide a build environment that a subset won't dislike.

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

N = 737

Reality check. What's the competition? This question keeps showing that there's no viable multi-platform option, and especially not one that deals with multiple protocols.

I think this is underscored by the fact that our respondents keep being set on using wget or code from wget (63.6%) if they can't use libcurl and on #3, write it yourself (10.3%). On #2 we see a trend-changer ("native lib in Python, Perl, Go, Rust etc" at 48%, up from 26.7% in 2019) - which shows that our biggest competitors are the ones written specifically for specific languages. Number #4 ("windows native" at 10.0%, up from 4.7% last year) seem to indicate that a notable share of users would be fine to pick a platform-specific solution. Also #6, "macos native" grew notably to 8.8% from 2.9% last year.

#	**Alternatives**	**2020**	**2019**
1	wget	63.6%	44.7%
2	native lang lib	48.0%	26.7%
3	homegrown	10.3%	5.7%
4	windows native	10.0%	4.7%
5	macos native	8.8%	2.9%
6	asio	7.2%	4.5%
7	Qt	6.0%	1.9%
8	libsoup	3.0%	1.3%
9	poco	2.7%	1.5%
10	Cpp-netlib	2.6%	0.8%
11	serf	0.9%	0.0%
12	neon	0.8%	0.0%

What alternative download utilities do you normally use?

N = 817

New question for this year, as I tried to focus the previous question on libcurl alternatives and this one on curl command line tool alternatives. To no surprise at all, wget is by far the most common curl alternative “download utility”. And of course it will vary from use case to use case which of the alternatives you’d use at any given time, but still…

#	Alternative	2020
1	wget	86.8%
2	nc	25.7%
3	FileZilla	22.6%
4	ftp	21.9%
5	aria2c	14.7%
6	lynx	13.0%
7	httpie	10.5%
8	fetch	9.9%
9	Powershell	9.8%
10	lftp	8.2%
11	wget2	8.1%
12	w3m	7.5%
13	Httrack	3.8%
14	axel	3.3%
15	Pavuk	0.5%

Write-ins for “other” that was entered by more than one user:

#	Tool	Votes
1	scp / sftp	14
2	winscp	4
3	rsync	4
4	elinks	2
5	ncftp	2
6	Chrome	2
7	Firefox	2

If you miss support for something, tell us what!

N = 510

This question is like a desert menu. What of all these tasty option do you think you could enjoy at any point in the future? I have a strong suspicion that's exactly how users answer to this question…

This year, I removed a few earlier options as they're already implemented or are being implemented just now (HTTP/3, MQTT, parallel transfers with the curl tool) and I added a few new ones that I've already been considering myself and that were suggested in last year's survey. Surprisingly high numbers on some of them I think. ("ESNI" has been renamed but I kept the name in the survey as I believe it is still more widely known under this older name.)

#	**Missing features**	**2020**
1	websockets	45.3%
2	rsync	24.3%
3	bittorrent	22.2%
4	DNS-over-TLS	20.8%
5	DNSSEC (DANE)	18.4%
6	gRPC	16.1%
7	GraphQL	14.9%
8	HSTS	14.5%
9	SMB v2/v3	13.3%
10	SRV records	12.9%
11	Thread-safe curl_global_init	12.5%
12	DNS: URLs	8.0%
13	ESNI	6.9%
14	CoAP	3.3%

We got *many* great suggestions of what you've been missing. Here's a slightly filtered and cleaned up list.

1. --json $x <=> -d $x -H "content-type: application/json"
2. A websocket to fifo interface or similar would be cool for cli interaction
3. Ability to get both the non-2XX response status code AND body AND a nonzero exit code. I believe there was a feature request opened for this and it was closed because it's possible with extremely advanced bash constructs.
4. AIA & OCSP
5. all features/options in .curlrc should also be environmental variables. example, yum/rpm (rhel package manager) uses curl in the backend. a weird chicken/egg problem exists when port 80 is blocked. Need to install the rpm for RHEL to be able to install packages through

satellite (which installs a certificate). But I don't trust the cert yet. And only workaround is to use .curlrc with "insecure" added. Would be much easier if could use ENV var instead!

6. An option that combines -o and -O; in other words, I want to specify an output directory but still have curl auto-detect the file name
7. Authenticate with AWS STS
8. Auto detect os proxy settings and use them. You say curl does not do java scripting, but you can use (within curlib) os implementation for that.
9. AWS S3
10. Bandwidth throttling for HTTP/2
11. better saml token claim and cookie method for new schems
12. build in GPG / usign (openwrt) / signify support!
13. clarification about DOS (FreeDOS) target support
14. classic mac os support is very hard to get working. the support library it is based on is near impossible to find as the original site for it is gone. managed to build it but then lost those files again. a rewrite of mac os support that doesn't depend on it would be fantastic.
15. Clearer manpage. It needs some structure
16. CONFIGURABLE COLOR OUTPUT
17. Control over cURL connection pool, ability to stop/restart resolving process (threaded POSIX)
18. cross-platform asynchronous api, not using poll but curl abstracts away epoll/kqueue/windows-specific. Select/Poll based is very slow and the other one which supports epoll or even libuv isn't a great api for new developers.
19. curl_easy_abort()
20. dynamic loading of libraries so a [lib]curl binary can survive libraries being missing. Particularly true of TLS. Configure/build is fine for individuals/sites, not if one wishes to have [lib]curl be part of a product which can land on an arbitrary system and to the right thing (if possible.)
21. Easily embedded for ios apps
22. Easy to understand examples for Curl command line. Lots of examples.
23. EBCDIC-compatible test suite (to be able to distinguish between text and binary data transfers on test servers)
24. ETag control while resume downloading
25. Expose asynchronous DNS client in API
26. Faster transfer speeds over SFTP (incl. tuning CURLOPT_UPLOAD_BUFFERSIZE from command line)
27. Gemini

28. Graphs on VRM
29. gRPC and SRV records would be cool.
30. HTTP Content-Disposition header parser/helper for applications
31. integrated json request building, like httpie
32. native c++
33. please change the command layout, it's difficult and not really straightforward. wget does this better in my opinion but i like curl better :)
34. Persistent socket mode like ssh. Curl cmd line listens on Unix socket for requests from future cmd line reqs and then sends them via persistent connection. Uses cookies and TLS session caching too.
35. Rfc7574 “Peer-to-Peer Streaming Peer Protocol (PPSPP)”
36. S3, including multi-part upload
37. SimpleCGI over Unix socket
38. The raw TCP protocol. Would be nice to have libcurl manage the connection, transfer, etc.
39. wire level debug trace
40. would really like a libcurl API to extract the certificate information as binary DER so we would not need openssl specific code to do it. The API that gives a broken down text representation is not sufficient for a browser.

These items were mentioned as missing that we already support – with my comments:

41. Compare TLS/SSL certificate with a locally stored one to really verify a host (*We support custom CA cert bundles that enable this.)*
42. automatic parsing of filename from headers similar to "wget https://example.com/file.tar.gz" (*Supported since years with -J*)
43. support for HTTPS over an HTTPS proxy. (*Support already, depending on TLS backend*)
44. client controlled easy transfers (e.g. using curl_easy_recv on a handle not marked as connect only) (*I presume this means an pull-driven API like https://github.com/curl/fcurl – which then obviously already exists)*
45. Getting Final-Redirected URL without temp file. Link: https://stackoverflow.com/a/3077316 (*Seems to be a misunderstanding, since it writes to /dev/null it is **not** saving a temp file.*)

What feature/bug would you like to see the project REMOVE?

N = 30

Entirely free text. I applied some manual filters and here are the least nonsensical of the suggestions. Most of them will of course be argued against with this simple phrase: *we don't deliberately break existing use cases*.

- -X
- 'v' and amend 'trace-ascii' to be a bit less similar; '-i' seems pointless with '-D'
- ancient protocols
- Anything that makes the developers life harder :-)
- Automatic use of HTTP POST method if request data is passed. Forgetting the -X flag should yield and error when passing data.
- CMake
- Complex integration with multiple SSL libraries, especially integrations that expose the internals of the SSL libraries (such that you have to know which SSL library curl is using to poke at certain features). I'd happily deal with whatever subset of libraries (and hard minimum version requirements on those libraries) curl supported, in exchange for abstracting portably over their features.
- Curl URL globbing prints the URL of each requested page to stdout, this is a problem if I wanted to read multiple URLs with glob and combine all their output into a pipe. These headers are mixed into my output. This can be sort of mitigated with --silent, but I didn't want to actually disable errors, I don't mind error output on stderr.
- everything but HTTP(S)
- FTP
- I don't think you can remove support for large companies who use your product to crush humans under their overzealous, greedprone feet. You just can't code around human behaviors, and I don't think you'd be able to fight a battle vs AI... so let's just skip this and say continue to make it available for all, even those who are despicable. Because that's fair to all. And fair to all is important.
- IIRC curl does some "goofy" stuff like "re-checking" with select() or poll() before a read. Maybe this was needed in some weird case, but it seems like there is just a little bit too much going on between curl_multi_socket_action() and read() :)
- I wouldn't mind if Apple support was removed.
- Old TLS ciphers should need to be explicitly enabled… though I suppose you already do this with compile-time functionality
- Plethora of build systems. CMake + autotools cover just about everything IMO.

- SMTP
- The progress indicator, it’s only disturbing
- Trimmed and super-portable version of plain-http-only libcurl will he great

Which of these API(s) would you use if they existed?

N = 569

This is like asking a child what kind of candy they would like. I think people over-select these. Mostly based on the fact that #4 is again at 26% and we've tried to offer the fcurl project which provides pretty much exactly "a read()/write() style API" but the interest has remained very low…

#	**API**	**2020**
1	JSON generation/parsing	52.0%
2	header parsing/extracting	43.9%
3	establishing a websocket connection	41.1%
4	a read()/write() style API for downloading and uploading	26.0%
5	handling of TLS certs/keys in memory	20.7%
6	Server-side support library for HTTP(S)	19.3%
7	HTTP Content-Disposition header parser/helper for applications	17.0%
8	PAC support	8.8%

The write-in suggestions for new APIs included…

- s2n client support (*how is that a new API?)*
- per-multi bandwidth limitation settings
- Pluggable async DNS resolver
- link header parsing
- WPAD
- Protocol stage callbacks that always get called
- ability to check certificates for revoke (by CRL, OCSP, OCSP stapling)

Do you wish to attend the next curl://up meeting/conference?

N = 796

This question is perhaps mostly present to make people aware that they can actually attend this annual event… The share of “No” went up even more this year (the 3rd year we ask the question) from 60.9% to 64.8%. The “Maybe” share was at 22.5% and among the three different Yes, the one for Europe got the largest share (7.2%), North America got 2.3% and “independent of location” 3.5%

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

N = 30

30 users filled this in, but some of the answers contained multiple questions. Here’s a filtered list of suggestions:

- Do you embed curl in a commercial product?
- Do you like the curl:// logo (Yes/No)?
- Have an «Not Applicable» «N/A» or similar choice on each question, to reflect that one doesn't have an opinion on the question or that it doesn't apply to me.
- How could a more ergonomic command line interface be introduced? (*Now that’s a question that would be hard to answer to in a small and easy way…)*
- How do you integrate the curl CLI? E.g., *nix with pipes, pagers, etc
- How do you learn curl: man, help, docs, net?
- how involved are you with curl/community?
- Is our web page intuitive?
- Is the documentation intuitive? Would you recommend curl to others?
- Is it hard to get into the project?
- Is it hard to use the tool? Do you miss any example? Are the tutorials intuitive? Do you miss any tutorial? In general, how would you rate curl?
- Please indicate the part you think should be needed performance improvement.
- What should be documented better?
- Which GitHub issue would you like to get resolved most?
- Which undefined behavior did you encounter we should fix?
- do you consider yourself part of a minority in the community?

Anything else you think we should know?

N = 148

Okay, yes, this is of course the most open-ended question we could make but it seems users also took the opportunity to tell us something here. Roughly one in six did.

I've included most of the comments here verbatim, sorted alphabetically. Included to show all members of the curl community the level of love and appreciation we get. It makes me all warm to read through this.

(lib)curl is already better than any expectations I have! Great work!

(lib)curl is awesome; thanks for working on it!

<3

♥ curl!

As a distro maintainer, I wanted to say that I really appreciate all of the work that you guys do!

Awesome work

awesome work, thank you!!

Bagder is the Bset!

bees are cool

Can't imagine life without CURL. Thank you.

curl är så jävla bra

Curl does everything i expect it to do since 2000 - Thanks a lot!

Curl has incredible quality that other projects should aspire to. The most painful aspects are when it has to interface to other projects (such as OpenSSL).

Curl is a solid piece of software and i'm glad it exists

cURL is a useful tool; I use it mostly for HTTP/S GET/POST requests and not much else. It saves me from implementing HTTP.

CURL is amazing! Thanks for your efforts!

Curl is awesome! Only drawback is I sometimes struggle to find correct options to do what I want, because there are so many! Or – which is worse – I assume something is default and it is not. But I do not know how to improve it.

Curl is awesome. I always hold the curl project up as one of the best examples of an open source project for code quality, features, and the community that surrounds it. Also, I haven't ticked HTTP/3 support as I haven't actually used it yet, but purely from a lack of time to experiment and not a lack of interest, I look forward to playing with it in the near future!

Curl is great, but borderline kitchen sink. Don't make it overly complex

curl is great!

curl is great! thanks for your effort!!

curl is indispensable and I sincerely appreciate all of the hard work over many years.

Curl is really great, works like a charm! Thank you

curl is, by far, one of the most amazing libraries I've ever had the privilege of using. Thank you!

curl rocks!

curl works so well I just think of it as a command line tool that doesn't require further thought, like `ls` or `ssh`. Thank you for years of consistency! It's interesting to see so much evidence of community, I'll try to pay more attention to it now that I know about it

Don't drop SCO Unix support

Feel good about this. Everybody is coming together to make this a remarkable project.

For end-users like me, your tool is awesome. It's not much but thank you for your work and the ease of use it provides as long as we know the options… which are well documented anyway. Again, thank you.

FreeDOS does not mirror newer Curl versions, but that can be immediately fixed by me. However, I've never rebuilt it and find it hard to verify dependencies (and licenses), and the current "developer" of Curl for DOS isn't in contact with any of us. I don't even have available the latest release. I'm not a pro developer and know nothing of networking, but I'm still sympathetic. However, for such a niche (and "obsolete") platform, it's hard to get even motivated people to work together.

Generally happy with libcurl.

Great job, just wish it would be easier to use and compile.

Great work

Great work, I use it for debugging http/https services a lot

Great!

Great. Keep going at it !!

Have a nice day :]

Have been using curl since ye olden days of 1999.

Honestly, I had no idea that curl/libcurl could do most of the things it can - reading your (infrequent) mastodon posts about it is REALLY helpful. Please continue to post there (this is for Badger, but you know. Everyone at Haxx, and in Horizon (:)

I **love** curl <3

I always confuse "-O" and "-o" when using curl, not to mention their usages are totally opposite in Wget.

I am not entirely sure how I could do my job without curl.

I don't *need* to download stuff (and I didn't know any of the proposed alternatives that I hadn't checked), however I use curl a lot as a trusted client to test my server code, especially on H2 where

telnet is not very useful anymore. This is why standards compliance and early availability are its most important features to me, but I know I don't even represent a tiny fraction of the users.

I haven't yet tried metalink files, but am keen to do so soon. They could be very useful for pulling (and verifying) large build dependencies from offsite locations.

I just appreciate a tool that works, has always worked, and looks like it always will (i.e. is following all the new protocols)

I like libcurl

I like stumbling upon Daniel's blog posts from time to time.

I love your project! Appreciate it, and used a lot on my projects! Very simple API

I often worry about the expanding scope of what curl and libcurl handle, but figure that you know what you are doing and that you are probably basing them on legitimate use-cases for new .

I really enjoy the blog posts with technical detail :)

I still have plenty of videos to watch from Your channel on YouTube which I subscribed to in the morning and where I can join the mailing list. I think I will find the answer of my latter question via a simple SE (DuckDuckGo in my case) query ,so discard it, please!

I think I am a bit of an outlier because I am a NetSurf web browser developer and help maintain the internal scheme fetcher system. Hence the large number of odd operating systems I build for. Hope the response is useful and thanks for all the hard work.

I think it'd be beneficial to add an option for splitting executables into domain-specific tools, like curl-mail for email protocols curl-www for http(s), etc. to allow for easier searching of necessary/desired flags rather than having to skip over flags for SMB, POP3, and so on while looking for flags to modify HTTP requests.

I use hundreds of command line utilities and this survey made me realize I was just taking curl for granted, thanks for the great work.

I'm a Pentester, whenever there is an issue connecting to a target we resort back to debugging the issue with cURL. I couldn't imagine working without it! Seriously!

I'm grateful that you've created such a high-quality library and made it available for any open-source project to use. Thank you, Daniel, and all the contributors!

I'm just a very grateful user.

I'm not very involved with curl.

I'm really just a dev, using networking libraries wrapping curl and using curl CLI for automated test. I don't have a really analytic view of curl strength and weaknesses, but I know this project is fantastic because I would use at least twice the time to develop webservices if I had to handle all this on my own. And yet my company does not contribute the least bit to OSS, even if I think I'm far from the only dev to take advantage of such OSS projects. So thank you, really thank you, for your work and your software.

It would be nice if the test suite runs faster (in parallel?)

It´s a great and valuable Project. Thanks for doing it!

Just keep it goin'. Thanks!

K.I.S.S.

keep being awesome!

Keep it up =)

Keep on rocking

Keep up the awesome work!

Keep up the good work ;-)

keep up the good work in maintaining the great curl tool up to date and useable for mankind.

Keep up the good work!

Keep up the good work!

Keep up the great work

Keep up the great work!

Keep up the great work.

libcurl eats up a lot of memory for each handle

Merchandise (e.g., t-shirt, hoodie, stickers) to support the project! Also keep up the amazing work!

nice work!

Not sure if I contributed much, but wanted to say thanks for all the work you and all the other contributors have put into this. Hope to see another great year

Please keep on doing what you're doing!

please, make more simple high level API over plain C for cpp.

Postman collections are common in various places. Ignoring personal preference, Postman masks may errors cURL exposes, so I'm glad to use cURL instead.

Really love using libcurl in projects, keep up the good work.

Thank you for being an awesome project steward.

Thank you for being so vocal about getting paid for your OSS work, especially on such foundational software. I'll personally contribute soon to try to make up for 15+ years of free usage. Well deserved.

Thank you for excellent program.

Thank you for making curl. Love you.

Thank you for making developing and life easier.

thank you for making this

Thank you for providing and still supporting this essential utility. I use the cli on a almost daily basis or as part of another product

Thank you for the cute and power of the curl!

Thank you for your efforts!

Thank you for your work and leadership in this valuable tool. I don't have many use cases for curl, but when I need to use it, it works. If I don't know how to use it, there's the documentation and the web to show me how. If I ever lead an open-source project, I will use curl as my model.

Thank you so much for your amazing work!

Thank you!

Thank you!

Thank you.

Thank your for communicating with the community !

Thanks a lot for all of your hard work, cURL is awesome! <3

Thanks Daniel for this amazing tool. I use it almost everyday!!

Thanks for creating maintaining curl!

Thanks for giving the world the gift of curl!

thanks for making curl! I learned from this survey that curl/libcurl have a lot more features than I thought, I had no idea it supported many ssl backends

Thanks for the fantastic tools

Thanks for this library

Thanks for your job!

Thanks, Daniel.

Thanks!

Thanks!

Thanks!!!

thanx !

That you guys rock!

The attendance could be a (yes/no/maybe - if close to my location). Which is your location?(uuee, europe, ...). I would like to attend remotely, if possible. "easy" and "multi", why not using generic terms to describe what the code does? The functionality implemented by "easy" is not easy, but simple. "easy" is a relative word. You can still make a "most used functionality" for newcomers or less professional/technical users. Please, the homepage requires a clear distinction between documentation, tutorials and examples. Documentation would be the API, where each structure and function should be described, examples would be pieces of code, and tutorials would be long documents (or videos) which describe what to do step by step. In general I love the project and as soon as I will get documented enough to make some contributions.

The utility is simply awesome. The best feature I have discovered so far are the cookie support through that cookie txt file. I find the documentation for libcurl alright, but I think it could maybe use more examples to fill it out. I am planning on using GTK to create a Postman/Insomnia alternative because I am tired of Electron :). I think I am gonna forego libsoup, and instead use libcurl again. I had used it on a senior design project in college, and used libsoup in a recent project, but if I had to guess libcurl has more features which would mean my application would have more features :). Thanks for your continued work.

There are many questions I can't answer due to only being a user and unfamiliar with the curl development process, but the survey itself improved my confidence in the curl development process. Please keep going.

this is (IMO) the most powerful transfer tool. Many many thanks

This is not directly about curl; but the Firefox network debugger has an option to copy to the clipboard a curl command that replays a past query. I find this extremely useful to tinker with requests to badly documented APIs

To me curl and libcurl is a tool so I don't get that deep but there are times when I need it to download or upload some file. I do a lot with RESTful api's with it on the client side.

us users probably expect too much from libcurl because we do not know an effective way to access the information on its capabilities. Mailing lists are the best. Email usage has been reduced by stupid alternatives like WhatsApp, Slack and Discord.

wget is typically more intuitive for downloads. httpie has far prettier output. please make some noise about all the sites that ask users to curl a script and pipe it to bash as a means of installing software. this is unhealthy. thanks for curl, it's highly useful.

Would like to have a way to download a list of URIs from a file while maintaining keep-alive and maximum download speed

Yes! A big ..... THANKS :--)

you are awesome please keep awesoming even harder

You are awesome!

you are doing awesome work. thank you

You are great! Thank you! If my financial situation would be a little better, I would love to support you that way

You guys are awesome, thanks for doing the survey

You guys are awesome.

You rock !

You're awesome, thank you so much! :)

You're awesome!!

You're doing great work!

Your hard work on libcurl/curl is appreciated!

Your project is really great and I love its sustainability for many years.

Summing up the 2020 user survey

Wow.

There’s a lot of information, detail and feedback in here. Enough to keep us busy for the next year and I know that I personally will come back to this analysis over the year to relearn what people think and said about curl this year.

I intend to run the survey again next year, presumably with polished and improved questions. User feedback is good.

This analysis paper alone has taken me many hours to gather and produce. I hope you like it. If you find errors or mistakes, let me know. If they’re important or big enough, I will publish updates.

Daniel Stenberg