libcurl in the Real World

Dan Fandrich

May 2026
curl://up 2026

Using libcurl

- Designed to satisfy many applications
- Large API to make everyone happy
  - 100 functions
  - 351 CURLOPT_ options
  - 123 CURLE_ errors
  - Many more: 1270 symbols in total

Using libcurl

- No application uses it all
- But what parts *do* they use?
- Many are optional
- Some are mutually exclusive or deprecated
  - `curl_form*` vs `curl_mime*`
  - `curl_multi_socket` vs `curl_multi_socket_action`
  - `CURLOPT_PROGRESSFUNCTION vs CURLOPT_XFERINFOFUNCTION`
  - `CURLE_OPERATION_TIMEOUTED vs CURLE_OPERATION_TIMEDOUT`

Changing libcurl

- libcurl tries to be highly backward-compatible
- Bug fixes are actually behaviour changes
- Sometimes necessary to
  - Deprecate APIs
  - Change return codes
  - Change behaviour

What parts are being used?

- How can curl developers know what parts are being used?
- Annual survey gives clues
- Tracking people’s issues, feature requests, pull requests

What parts are being used?

- libcurl is Open Source
- Many libcurl-using applications are as well
- Why not just look?
- Just download the source and see

Finding Applications

- Just Google it
- Manual, doesn’t scale
- Need to manually download hundreds/thousands of apps
- Distributions have done the hard work for us
- Limited to C/C++ applications using libcurl directly

Choosing a Distro

- Restrict search to Linux only for maximum reach
- Distrowatch lists 1201 distros
- Big one to get as many apps as possible
- Fairly recent to get up-to-date versions
- Easy to find just libcurl-using applications
- Easy to download the source code

Choosing a Distro

- Arch: 409
- Debian Testing: 446
- Fedora 43: 397
- Fedora 43 w/RPM Fusion: 419
- nixOS: 253
- openSUSE Tumbleweed: 306

Choosing a Distro

- Settled on Debian Testing
- 441 rdepends packages
- 367 individual source tarballs
- 28.5 GiB source code
- 1,710,001 source files

Type of Analysis

- Run-time
  - Limited to what features we can figure out
  - Not scalable
- Static binary
  - nm on the executable
  - Skips use of options, error codes, macros, etc.
- Compile of source
  - Too difficult with many different applications
  - Would need tooling to extract the data we want

Type of Analysis

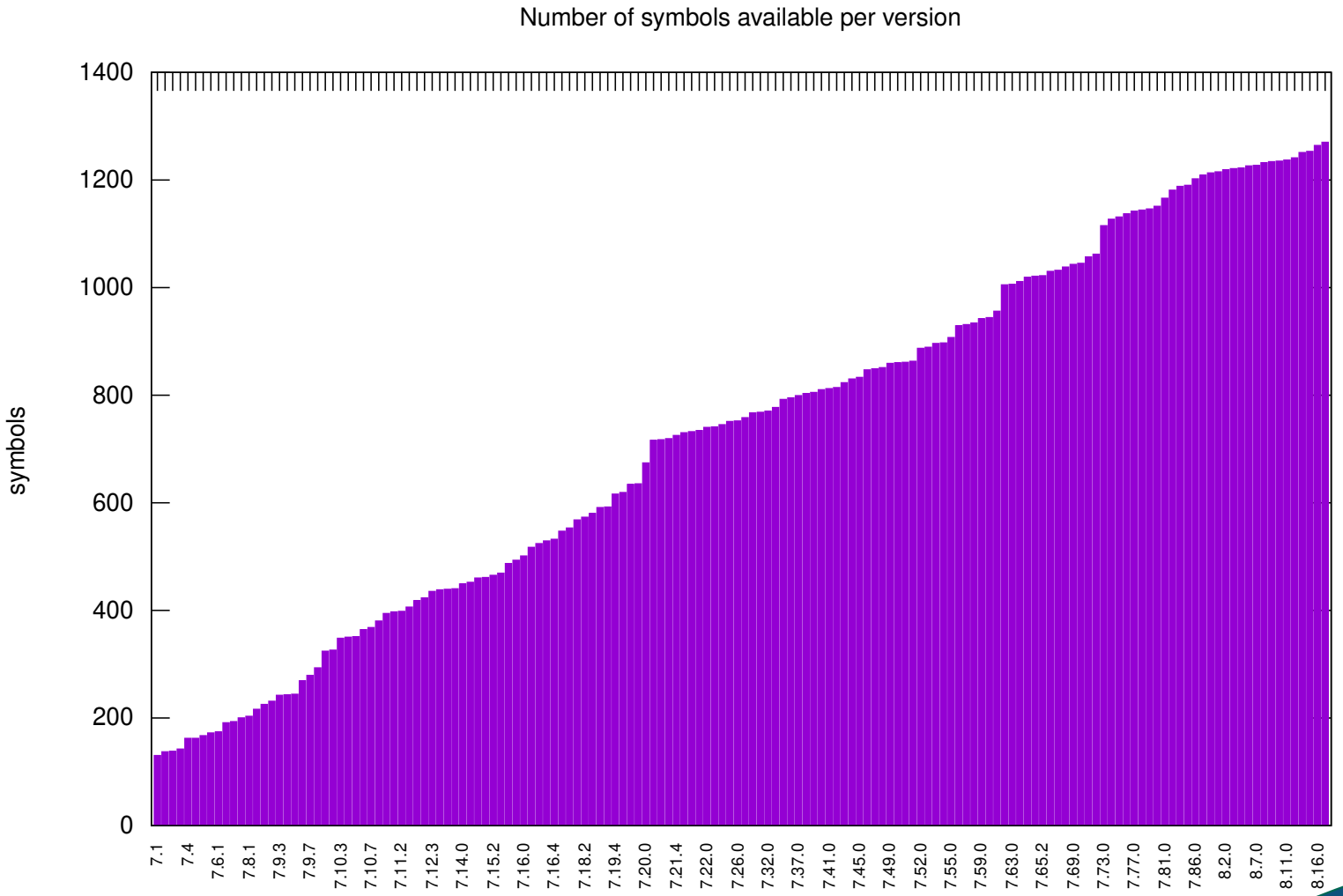
- Static source
  - Most flexible
  - Access to all features used by the program
  - Access to all build configurations
- Cons:
  - Misses obfuscated symbols, e.g. macros wrapping `CURLOPT_*`
  - Simplistic scanning means some references in e.g. docs count
  - Can include source files using non-C bindings

Processing

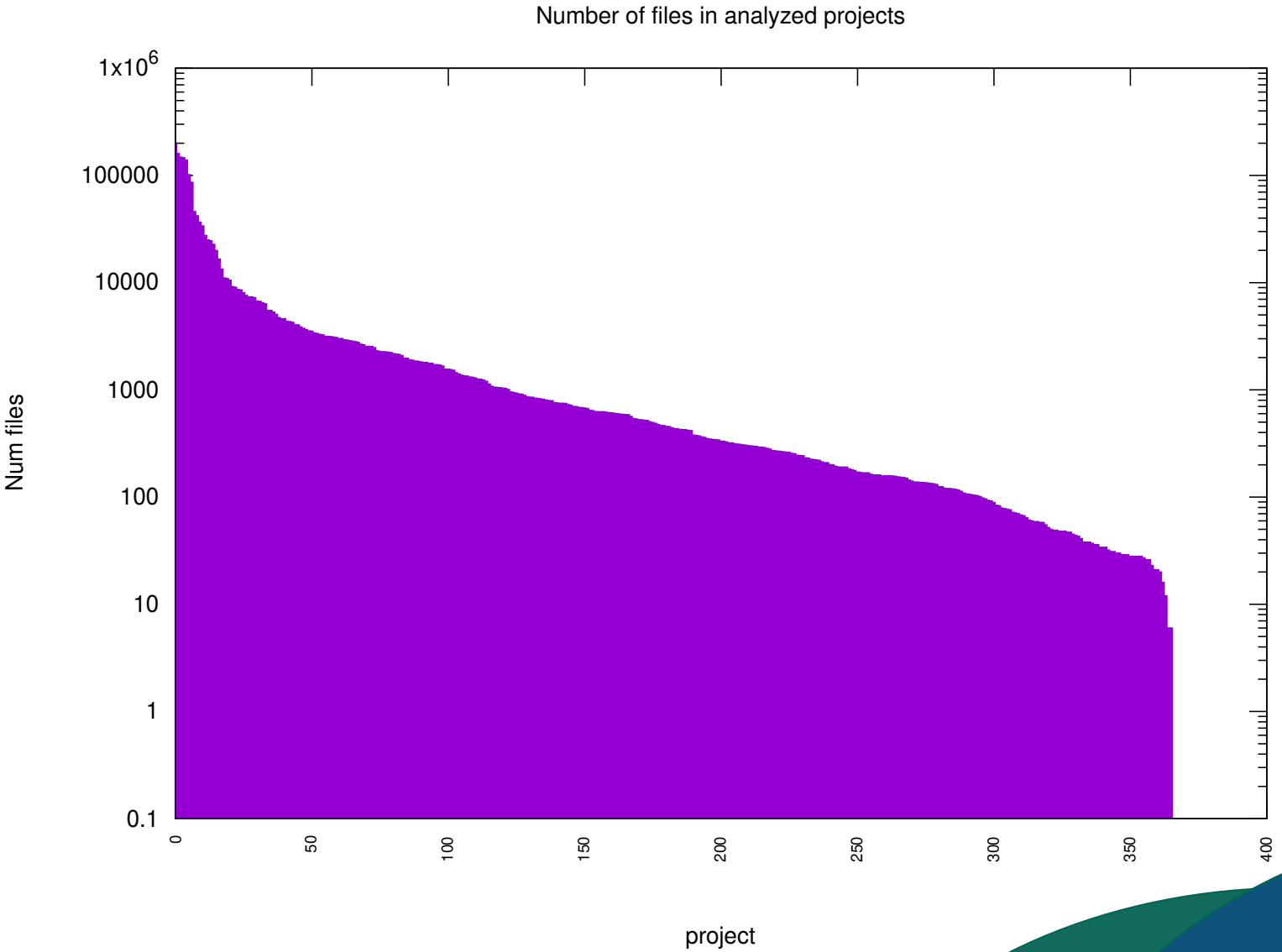
- Some vendored libcurl sources found embedded in packages
- Some copies of `curl.h` found
- Some copies of `symbols-in-versions` found
- Removed (mostly) to avoid diluting results

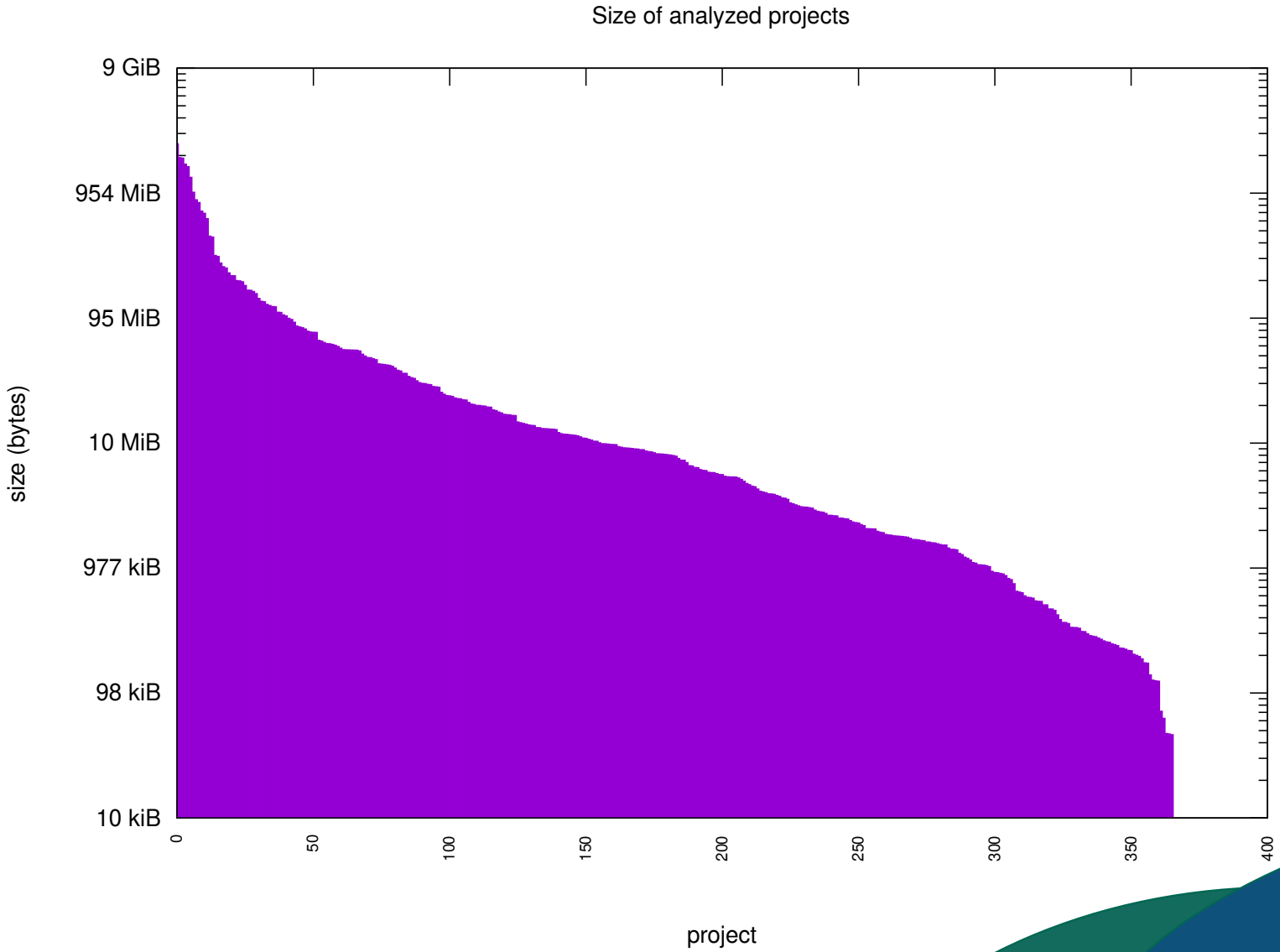
Processing

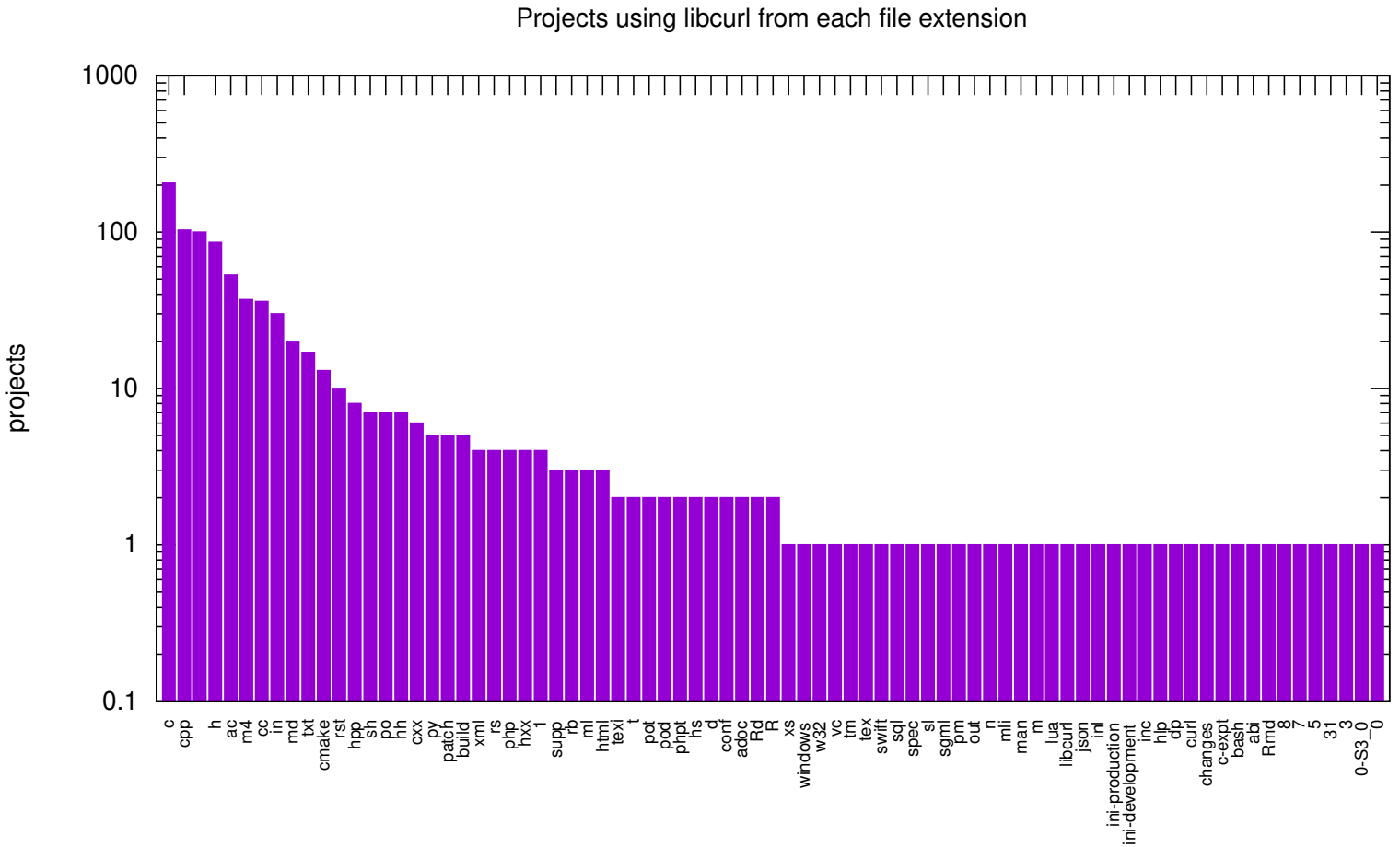
- Extract all functions & symbols from curl source code
- Files like `symbols-in-versions`, `test1135`, `VERSIONS.md` made it easy(ish)
- Supplanted with version info from man page `.md` files
- Massive grep of all symbols over all source code
- Results entered into sqlite3 DB

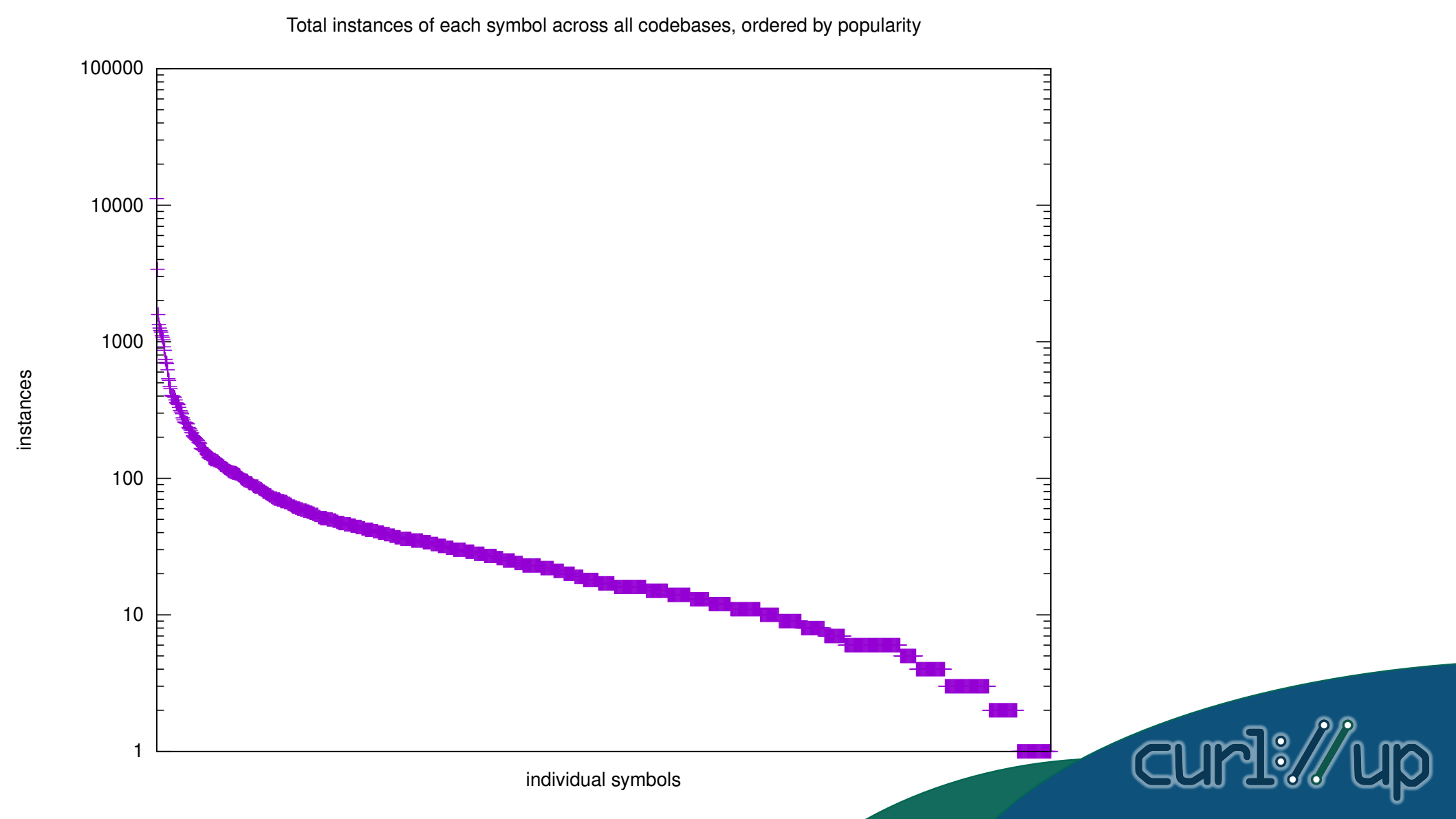

Number of symbols available per version
symbols
version
0
200
400
600
800
1000
1200
1400
7.1
7.4
7.6.1
7.8.1
7.9.3
7.9.7
7.10.3
7.10.7
7.11.2
7.12.3
7.14.0
7.15.2
7.16.0
7.16.4
7.18.2
7.19.4
7.20.0
7.21.4
7.22.0
7.26.0
7.32.0
7.37.0
7.41.0
7.45.0
7.49.0
7.52.0
7.55.0
7.59.0
7.63.0
7.65.2
7.69.0
7.73.0
7.77.0
7.81.0
7.86.0
8.2.0
8.7.0
8.11.0
8.16.0

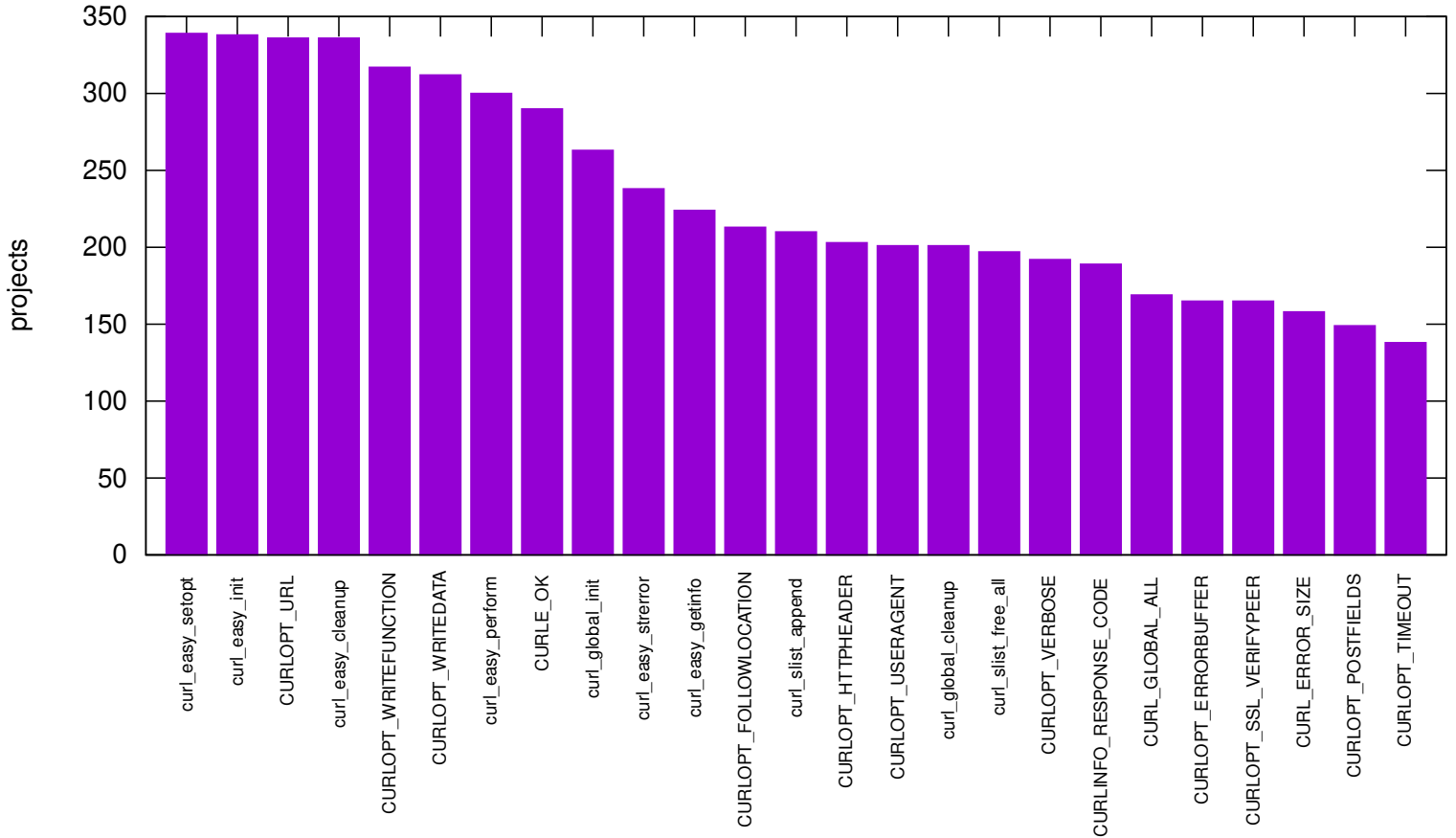
curl://up

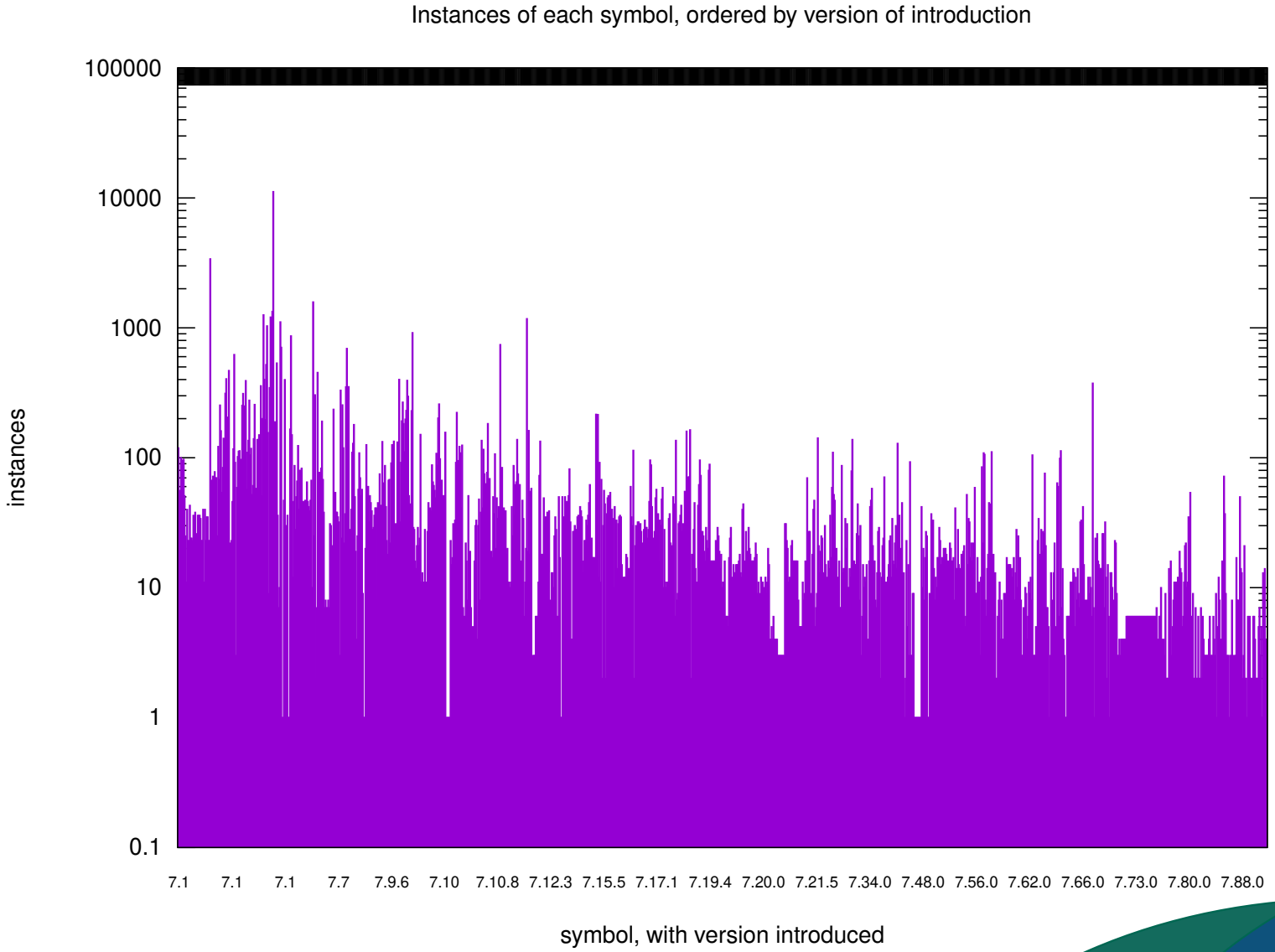

Number of files in analyzed projects
Num files
1x10^6
100000
10000
1000
100
10
1
0.1
0
50
100
150
200
250
300
350
400
project


Size of analyzed projects
9 GiB
954 MiB
95 MiB
10 MiB
977 kiB
98 kiB
10 kiB
size (bytes)
0
50
100
150
200
250
300
350
400
project

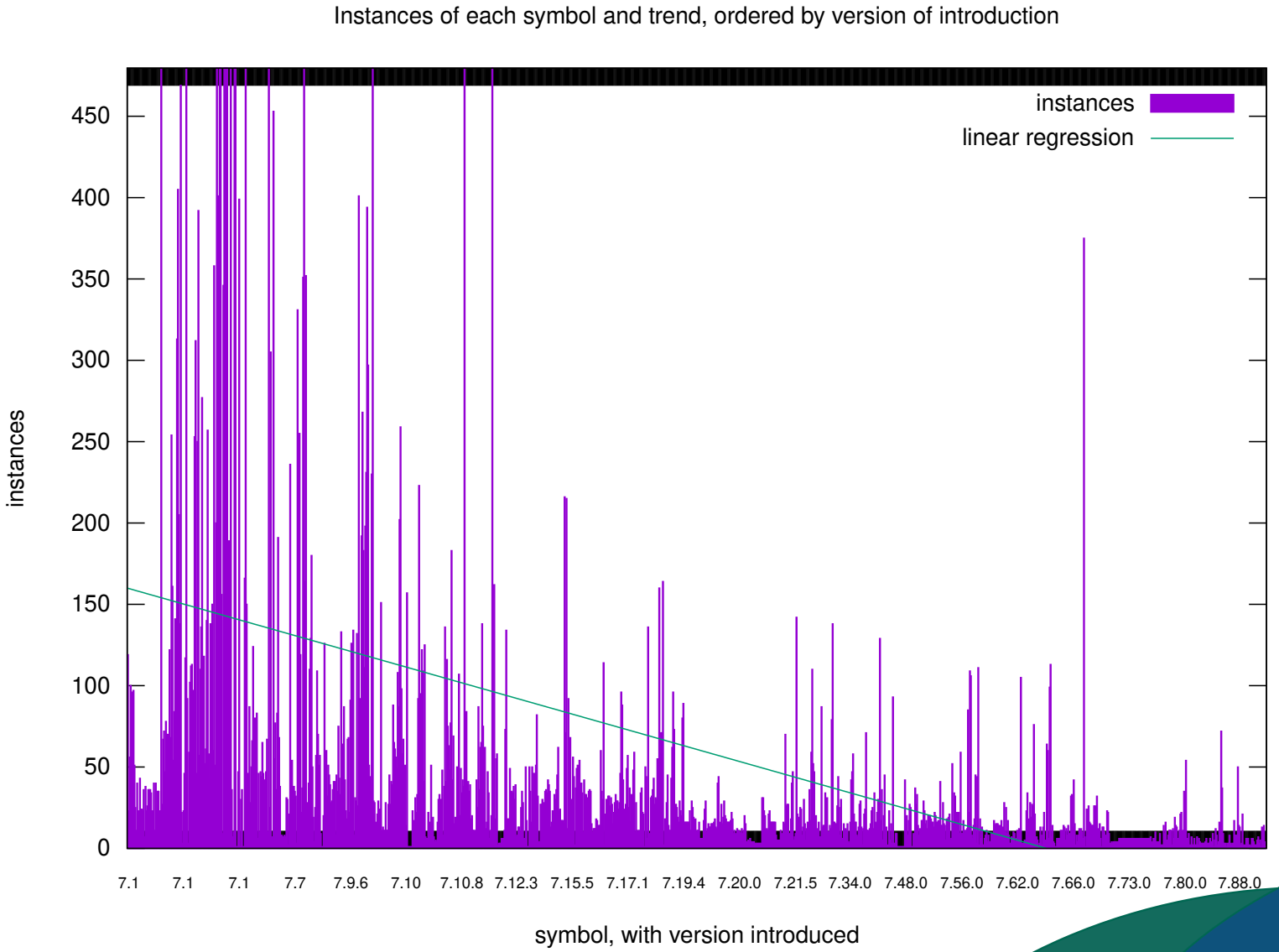

Projects using libcurl from each file extension
projects
1000
100
10
1
0.1
c
cpp
h
ac
m4
cc
in
md
txt
cmake
rst
hpp
sh
po
hh
cxx
py
patch
build
xml
rs
php
hxx
1
supp
rb
ml
html
texi
t
pot
pod
phpt
hs
d
conf
adoc
Rd
R
xs
windows
w32
vc
tm
tex
swift
sql
spec
sl
sgml
pm
out
n
mli
man
m
lua
libcurl
json
inl
ini-production
ini-development
inc
hlp
dp
curl
changes
c-expt
bash
abi
Rmd
8
7
5
31
3
0-S3_0
0
extensions


Total instances of each symbol across all codebases, ordered by popularity
100000
10000
1000
100
10
1
instances
individual symbols
curl://up

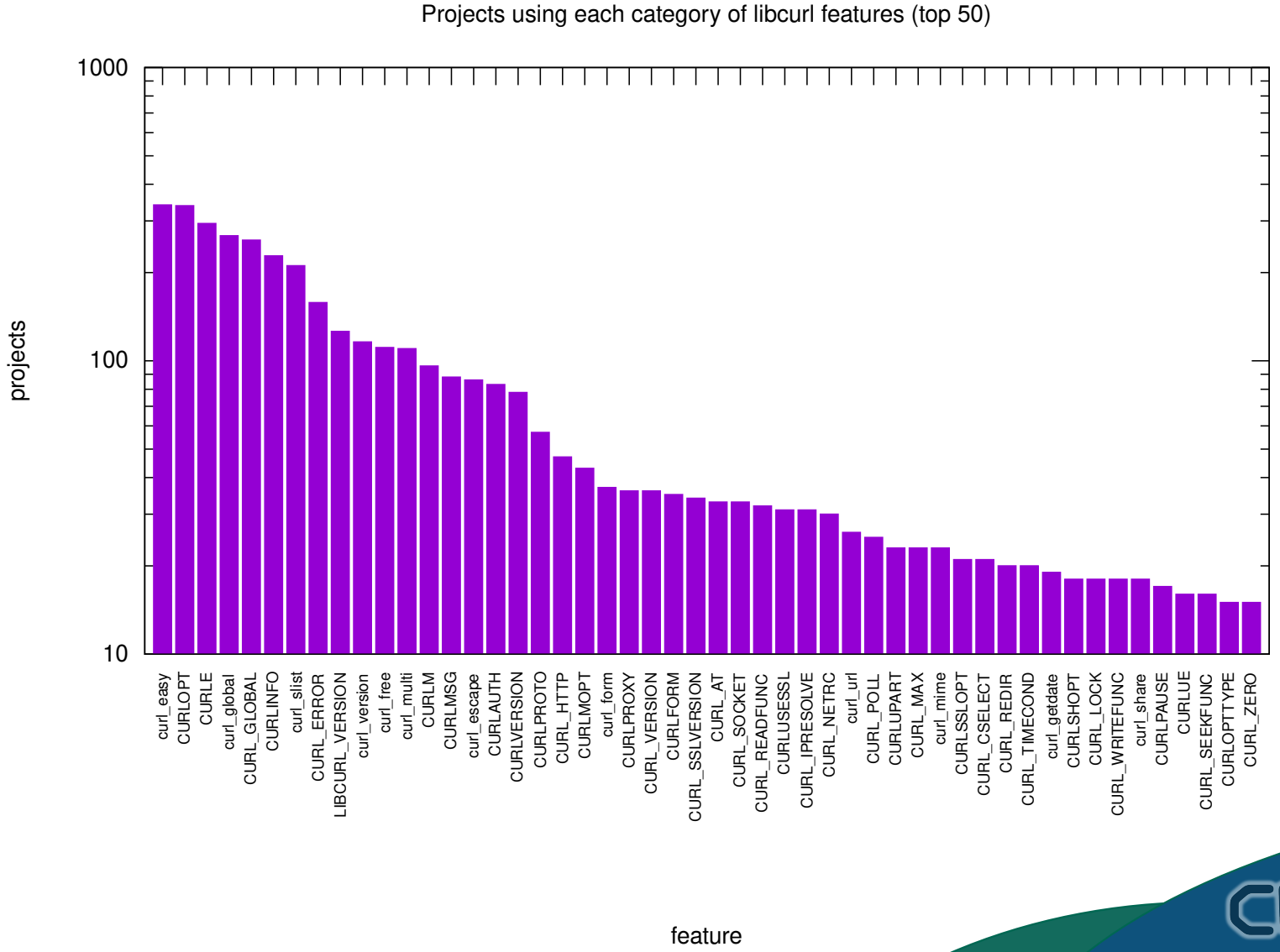

Symbols used by most projects
projects
individual symbols
0
50
100
150
200
250
300
350
curl_easy_setopt
curl_easy_init
CURLOPT_URL
curl_easy_cleanup
CURLOPT_WRITEFUNCTION
CURLOPT_WRITEDATA
curl_easy_perform
CURLE_OK
curl_global_init
curl_easy_strerror
curl_easy_getinfo
CURLOPT_FOLLOWLOCATION
curl_slist_append
CURLOPT_HTTPHEADER
CURLOPT_USERAGENT
curl_global_cleanup
curl_slist_free_all
CURLOPT_VERBOSE
CURLINFO_RESPONSE_CODE
CURL_GLOBAL_ALL
CURLOPT_ERRORBUFFER
CURLOPT_SSL_VERIFYPEER
CURL_ERROR_SIZE
CURLOPT_POSTFIELDS
CURLOPT_TIMEOUT

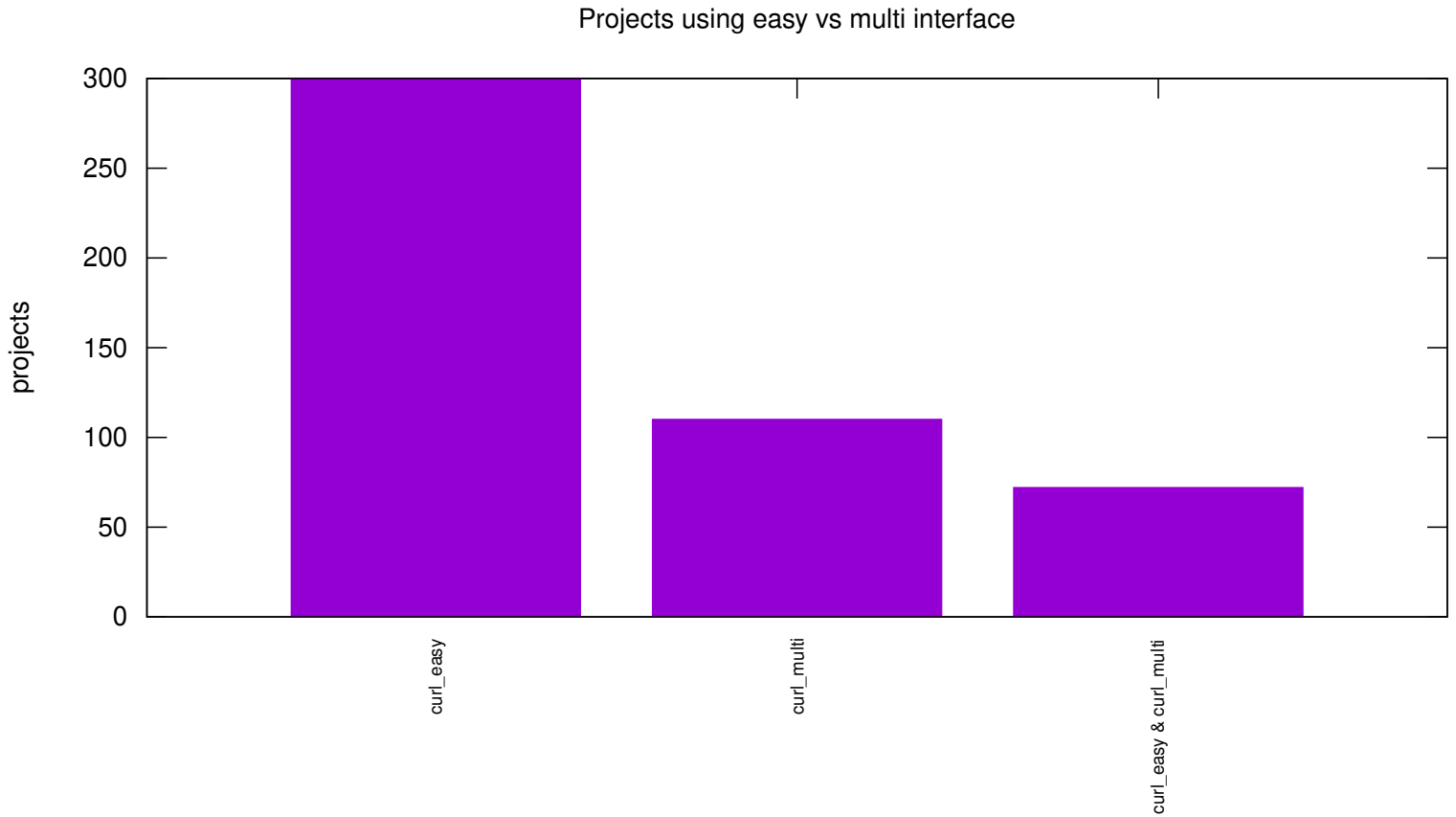

Instances of each symbol, ordered by version of introduction
100000
10000
1000
100
10
1
0.1
instances
7.1
7.1
7.1
7.7
7.9.6
7.10
7.10.8
7.12.3
7.15.5
7.17.1
7.19.4
7.20.0
7.21.5
7.34.0
7.48.0
7.56.0
7.62.0
7.66.0
7.73.0
7.80.0
7.88.0
symbol, with version introduced

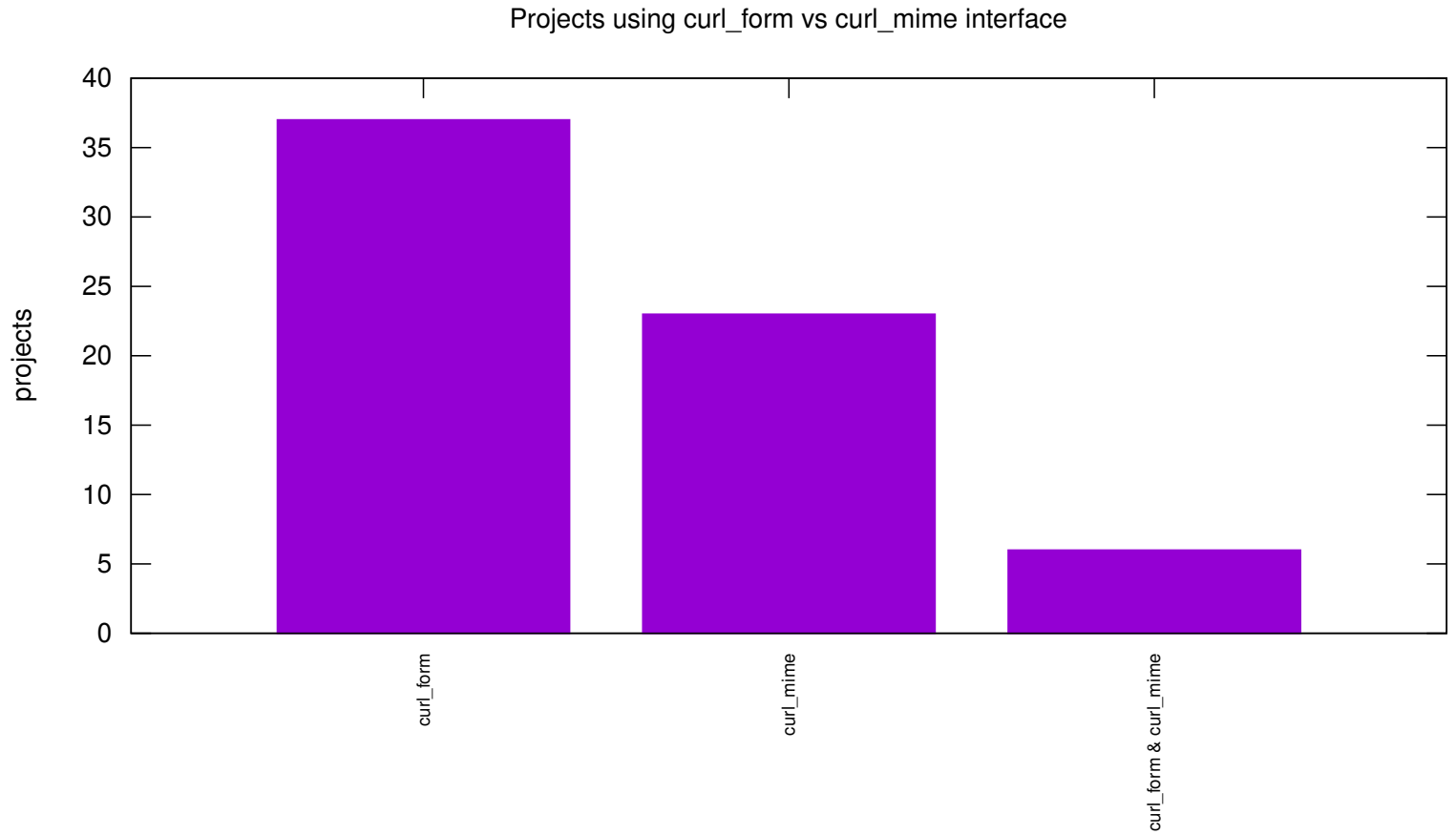
curl://up

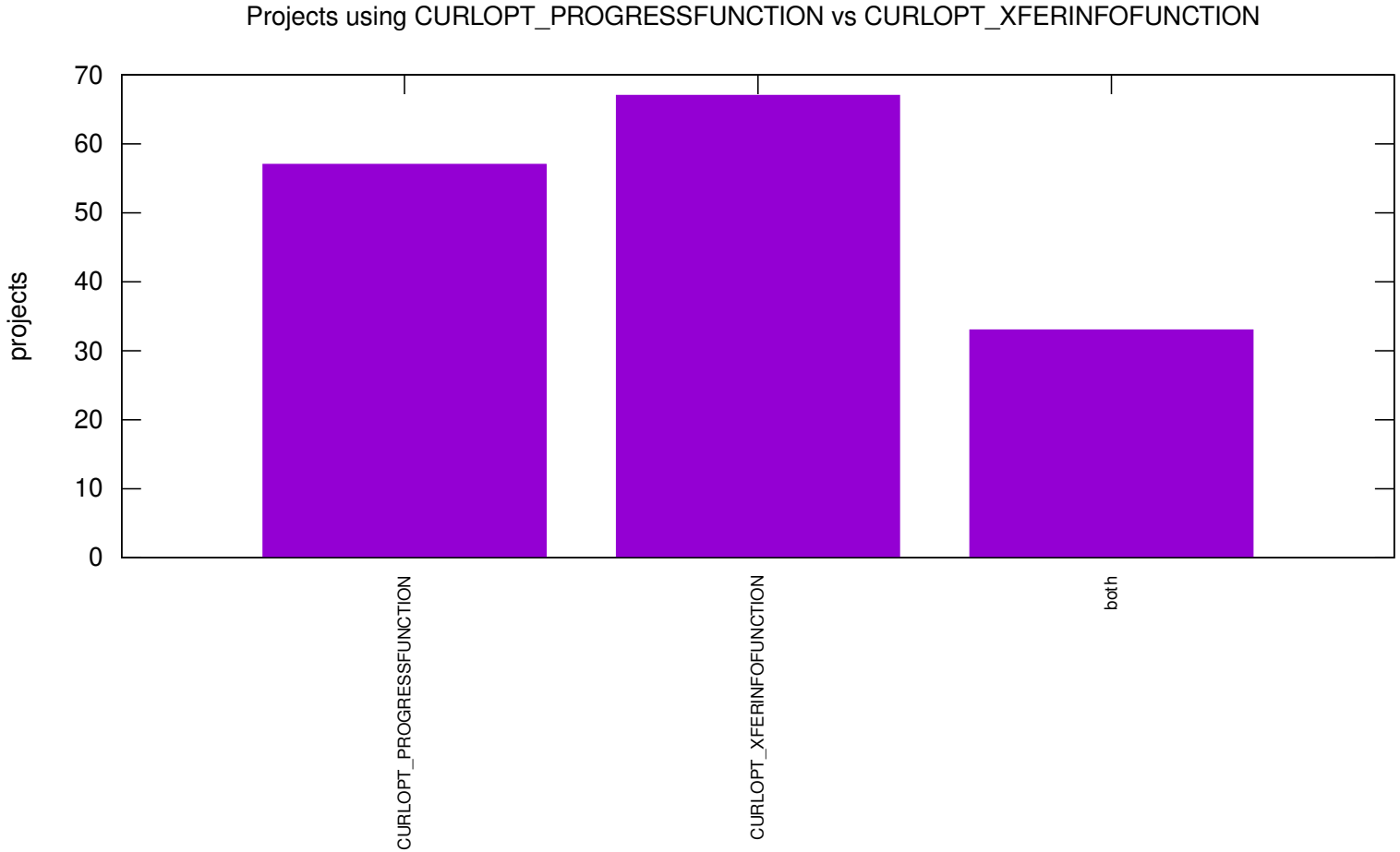

Instances of each symbol and trend, ordered by version of introduction
instances
linear regression
instances
0
50
100
150
200
250
300
350
400
450
7.1
7.1
7.1
7.7
7.9.6
7.10
7.10.8
7.12.3
7.15.5
7.17.1
7.19.4
7.20.0
7.21.5
7.34.0
7.48.0
7.56.0
7.62.0
7.66.0
7.73.0
7.80.0
7.88.0
symbol, with version introduced

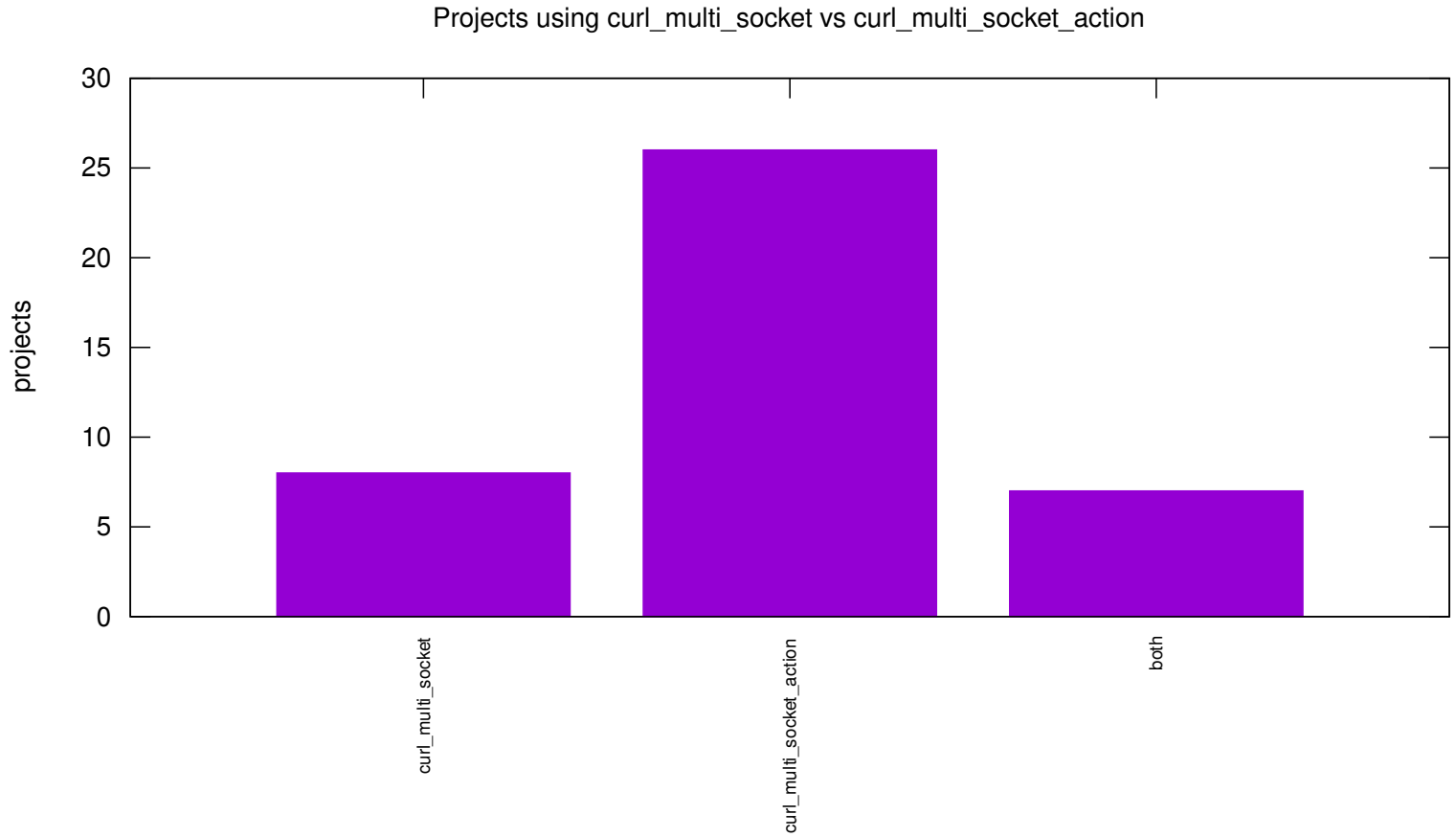
curl://up

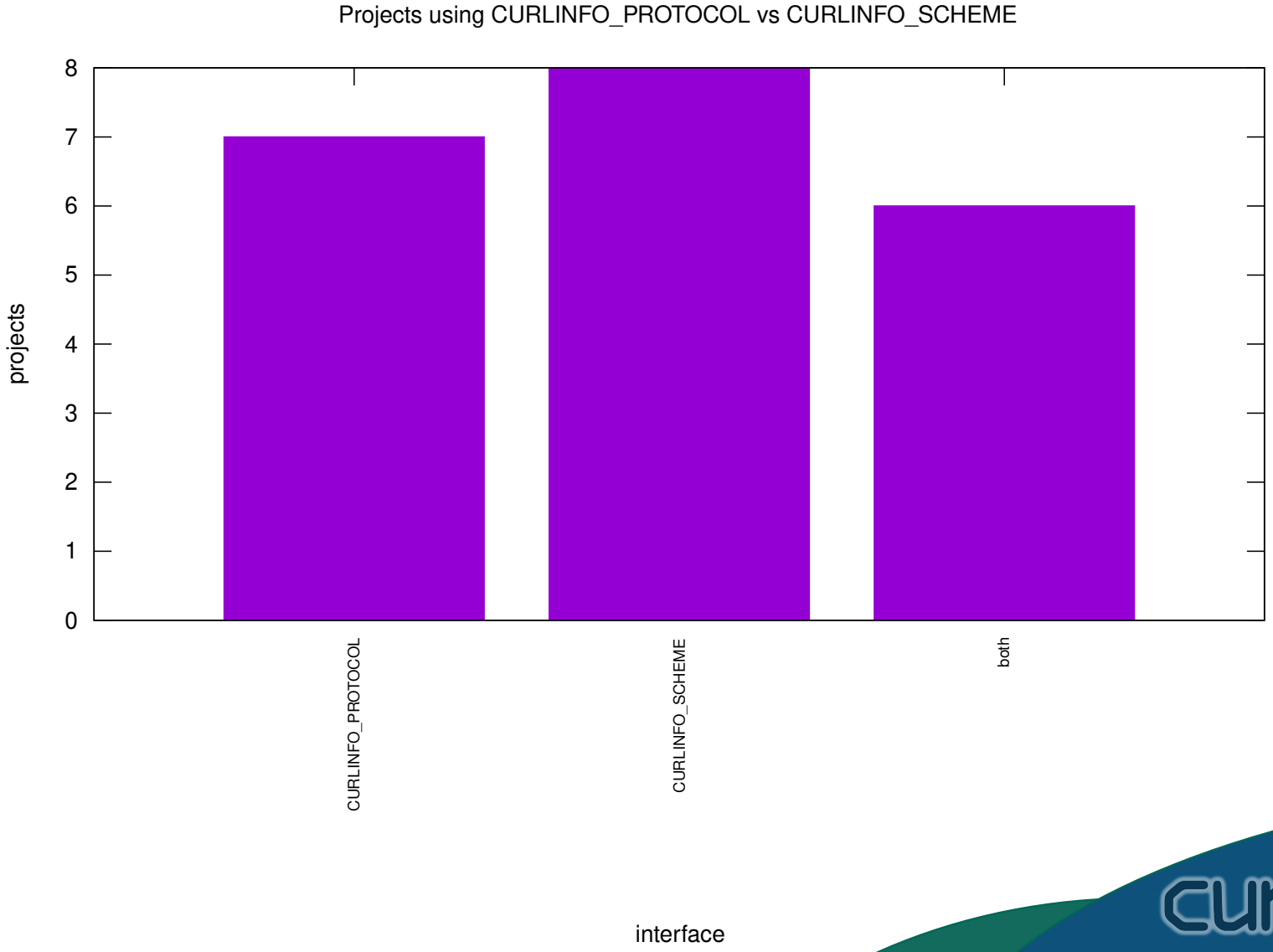

Projects using each category of libcurl features (top 50)
projects
1000
100
10
curl_easy
CURLOPT
CURLE
curl_global
CURL_GLOBAL
CURLINFO
curl_slist
CURL_ERROR
LIBCURL_VERSION
curl_version
curl_free
curl_multi
CURLM
CURLMSG
curl_escape
CURLAUTH
CURLVERSION
CURLPROTO
CURL_HTTP
CURLMOPT
curl_form
CURLPROXY
CURL_VERSION
CURLFORM
CURL_SSLVERSION
CURL_AT
CURL_SOCKET
CURL_READFUNC
CURLUSESSL
CURL_IPRESOLVE
CURL_NETRC
curl_url
CURL_POLL
CURLUPART
CURL_MAX
curl_mime
CURLSSLOPT
CURL_CSELECT
CURL_REDIR
CURL_TIMECOND
curl_getdate
CURLSHOPT
CURL_LOCK
CURL_WRITEFUNC
curl_share
CURLPAUSE
CURLUE
CURL_SEEKFUNC
CURLOPTTYPE
CURL_ZERO
feature

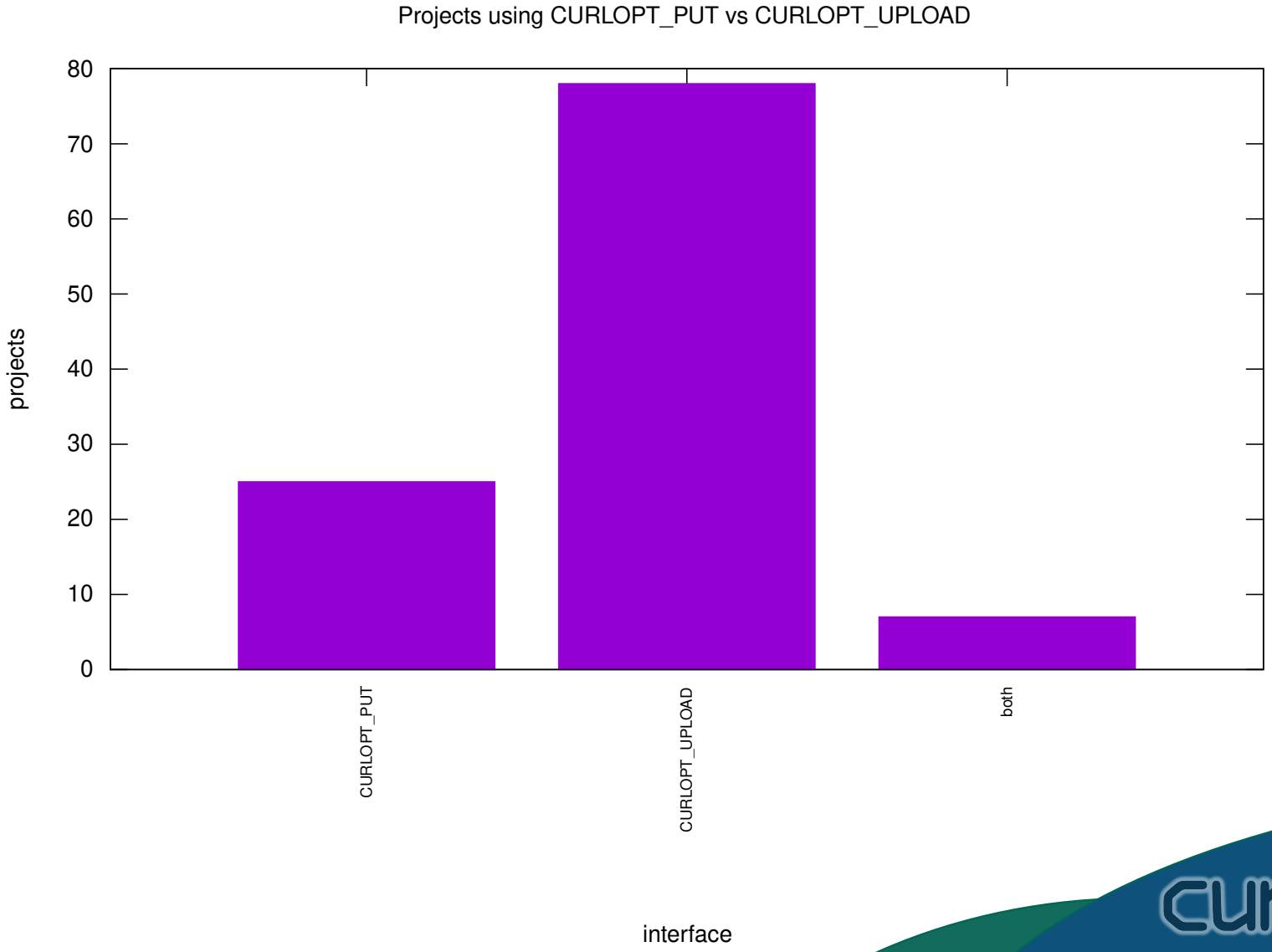

Projects using easy vs multi interface
projects
300
250
200
150
100
50
0
curl_easy
curl_multi
curl_easy & curl_multi
interface

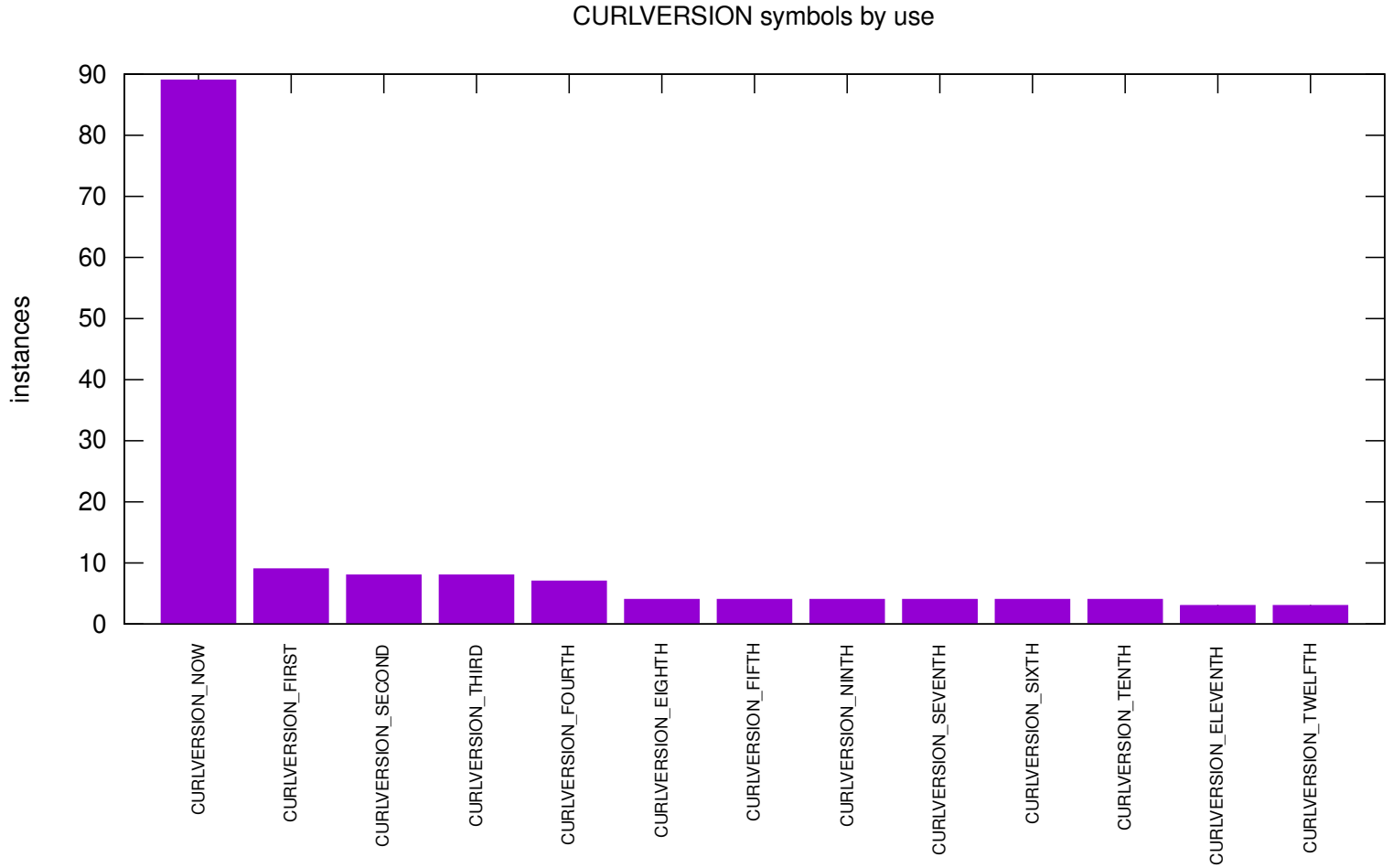

Projects using curl_form vs curl_mime interface
projects
40
35
30
25
20
15
10
5
0
curl_form
curl_mime
curl_form & curl_mime
interface


Projects using CURLOPT_PROGRESSFUNCTION vs CURLOPT_XFERINFOFUNCTION
projects
70
60
50
40
30
20
10
0
CURLOPT_PROGRESSFUNCTION
CURLOPT_XFERINFOFUNCTION
both
interface


Projects using curl_multi_socket vs curl_multi_socket_action
projects
30
25
20
15
10
5
0
curl_multi_socket
curl_multi_socket_action
both
interface


Projects using CURLINFO_PROTOCOL vs CURLINFO_SCHEME
projects
0
1
2
3
4
5
6
7
8
CURLINFO_PROTOCOL
CURLINFO_SCHEME
both
interface


Projects using CURLOPT_PUT vs CURLOPT_UPLOAD
projects
80
70
60
50
40
30
20
10
0
CURLOPT_PUT
CURLOPT_UPLOAD
both
interface


CURLVERSION symbols by use
instances
90
80
70
60
50
40
30
20
10
0
CURLVERSION_NOW
CURLVERSION_FIRST
CURLVERSION_SECOND
CURLVERSION_THIRD
CURLVERSION_FOURTH
CURLVERSION_EIGHTH
CURLVERSION_FIFTH
CURLVERSION_NINTH
CURLVERSION_SEVENTH
CURLVERSION_SIXTH
CURLVERSION_TENTH
CURLVERSION_ELEVENTH
CURLVERSION_TWELFTH
CURLVERSION symbols

Unused Symbols

CURLFOLLOW_ALL

CURLFOLLOW_FIRSTONLY

CURLFOLLOW_OBEYCODE

CURLINFO_HTTPAUTH_USED

CURLINFO_PROXYAUTH_USED

CURLMINFO_NONE

CURLMINFO_XFERS_ADDED

CURLMINFO_XFERS_CURRENT

CURLMINFO_XFERS_DONE

CURLMINFO_XFERS_PENDING

CURLMINFO_XFERS_RUNNING

CURLMNOTIFY_EASY_DONE

CURLMNOTIFY_INFO_READ

CURLMNWC_CLEAR_CONNS

CURLMNWC_CLEAR_DNS

CURLMOPT_NETWORK_CHANGED

CURLMOPT_NOTIFYDATA

CURLMOPT_NOTIFYFUNCTION

CURLOPT_SSL_SIGNATURE_ALGORITHMS

CURLOPT_UPLOAD_FLAGS

CURLULFLAG_ANSWERED

CURLULFLAG_DELETED

CURLULFLAG_DRAFT

CURLULFLAG_FLAGGED

CURLULFLAG_SEEN

CURLWS_NOAUTOPONG

CURL_HAS_DECLSPEC_ATTRIBUTE

curl_easy_ssls_export

curl_easy_ssls_import

curl_multi_get_offt

curl_multi_notify_disable

curl_multi_notify_enable

curl_ws_start_frame

Symbols Used Exactly Once*

CURLFORM_ARRAY_END	CURL_HTTPPOST_CALLBACK	CURL_PROGRESS_STATS	curl_multi_waitfds
CURLFORM_ARRAY_START	CURL_HTTPPOST_FILENAME	CURL_PUSH_ERROROUT	curl_mvaprintf
CURLOPT_NOTHING	CURL_HTTPPOST_LARGE	CURL_TRAILERFUNC_ABORT	curl_mvfprintf
CURLSSLBACKEND_AWSLC	CURL_HTTPPOST_PTRBUFFER	CURL_TRAILERFUNC_OK	curl_mvprintf
CURLSSLBACKEND_BORINGSSL	CURL_HTTPPOST_PTRCONTENTS	CURL_UPKEEP_INTERVAL_DEFAULT	curl_mvsprintf
CURLSSLBACKEND_LIBRESSL	CURL_HTTPPOST_PTRNAME	CURL_WIN32	curl_pushheader_byname
CURLSSLOPT_EARLYDATA	CURL_HTTPPOST_READFILE	LIBCURL_COPYRIGHT	curl_strequal
CURLU_DISALLOW_USER	CURL_LOCK_TYPE_CONNECT	LIBCURL_HAS	curl_strnequal
CURLWS_OFFSET	CURL_LOCK_TYPE_COOKIE	curl_easy_nextheader	
CURL_EASY_NONE	CURL_LOCK_TYPE_DNS	curl_global_trace	
CURL_EASY_TIMEOUT	CURL_LOCK_TYPE_ NONE	curl_mfprintf	
CURL_HET_DEFAULT	CURL_LOCK_TYPE_SSL_SESSION	curl_mprintf	
CURL_HTTPPOST_BUFFER	CURL_PROGRESS_BAR	curl_msprintf	

* But likely not *really* used

Future

- What other data would be useful to have?
- Can we use these data to inform future curl decisions?
- What are we in the mood for deprecating now?

Thank-you!

Questions?