CURL

YLMRX

Created: 2019-07-03 Wed 11:43

ABOUT

Yoann "`fuzzy`" Lamouroux:

- Reverse-engineer and security expert @dataimpact
- (we're hiring 😀)
- Former sysadmin
- Trol ^ Wdocumented opinions:
  - `xoxopowo@twitter`
  - `legreffier@irc.freenode.net`

ABOUT LAST YEAR

ABOUT LAST YEAR

- 5' is short (except when prod is down)

ABOUT LAST YEAR

- 5' is short (except when prod is down)
- Now I have 20 (w00t)

ABOUT LAST YEAR

- 5' is short (except when prod is down)
- Now I have 20 (w00t)
- I hope I deal better with time

ABOUT LAST YEAR

- 5' is short (except when prod is down)
- Now I have 20 (w00t)
- I hope I deal better with time
- (so I made a slide about dealing with time)

ABOUT LAST YEAR

- 5' is short (except when prod is down)
- Now I have 20 (w00t)
- I hope I deal better with time
- (so I made a slide about dealing with time)
- No more curling jokes (sorry)

TRIVIA

- Project started in 1996
- Still maintained by Daniel Stenberg (@badger)
- libcurl for about every language out there
- The curl binary is in EVERY default install

ALL OF THEM

ALL OF THEM

- GNU/Linux, *BSD

ALL OF THEM

- GNU/Linux, *BSD
- MacOS

ALL OF THEM

- GNU/Linux, *BSD
- MacOS
- Windows 10 (recently)

SOME QUESTIONS

- curl is old
- curl is badly documented (?)
- DevTools (Firefox, Chrome) is good
- `httpie` is neater/prettier
- `python-requests`

SOME ANSWERS

- Old means:
  - Good
  - Stable/reliable
- DevTools are indeed good
- httpie is a curl wrapper
- python-requests is python (hang-on, brb)

DOCUMENTATION

- You usually need `curl` in critical situations
- No time to dig through 3k lines manual

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies
  - Cloudy jokes

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies
  - Cloudy jokes
  - (aka. mai', aka. Kloug{Front,Flare,…})

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies
  - Cloudy jokes
  - (aka. mai', aka. Kloug{Front,Flare,…})
  - … whatever cool kids use these days

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies
  - Cloudy jokes
  - (aka. mai', aka. Kloug{Front,Flare,…})
  - … whatever cool kids use these days
  - And shiny boxes (aka. docker)

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies
  - Cloudy jokes
  - (aka. mai', aka. Kloug{Front,Flare,…})
  - … whatever cool kids use these days
  - And shiny boxes (aka. docker)
  - Tighter firewall policy (aka. no internets)

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies
  - Cloudy jokes
  - (aka. mai', aka. Kloug{Front,Flare,…})
  - … whatever cool kids use these days
  - And shiny boxes (aka. docker)
  - Tighter firewall policy (aka. no internets)
  - Just because you can run Chrome in docker,

EVERYWHERE

- DevTools won't get you far beyond the browser
- Today's IT imply:
  - Reverse-proxies
  - Cloudy jokes
  - (aka. mai', aka. Kloug{Front,Flare,…})
  - … whatever cool kids use these days
  - And shiny boxes (aka. docker)
  - Tighter firewall policy (aka. no internets)
  - Just because you can run Chrome in docker,
  - … doesn't mean you should

BASICS

```
>> curl https://www.example.com/
```

Display body on stdout.

VERBOSE

```
curl -v https://httpbin.org  > /dev/null
* Rebuilt URL to: https://httpbin.org/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0*   Trying 34.230.
* TCP_NODELAY set
* Connected to httpbin.org (34.230.136.58) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
} [5 bytes data]
* (304) (OUT), TLS handshake, Client hello (1):
} [512 bytes data]
* (304) (IN), TLS handshake, Server hello (2):
{ [89 bytes data]
* TLSv1.2 (IN), TLS handshake, Certificate (11):
{ [4832 bytes data]
```

PREFIXES:

- `*` : is information
- `>` : protocol verbose **FROM** your computer (*)
- `<` : protocol verbose **TO** your computer (*)
- `}` : encrypted data **FROM** your computer
- `{` : encrypted data **TO** your computer
- `[xxx]` : size (in bytes) of data transferred.

(ssl verbose with brackets is shown only when `stdout` is redirected)

(*) : doesn't mean it's not encrypted

MORE VERBOSE

tcpdump might not be the answer (yet).

`--trace` and `--trace-ascii` for byte-per-byte analysis.

Use `-` or `filename` as an argument to write to `stdout` or to a file.

CUSTOM HEADERS

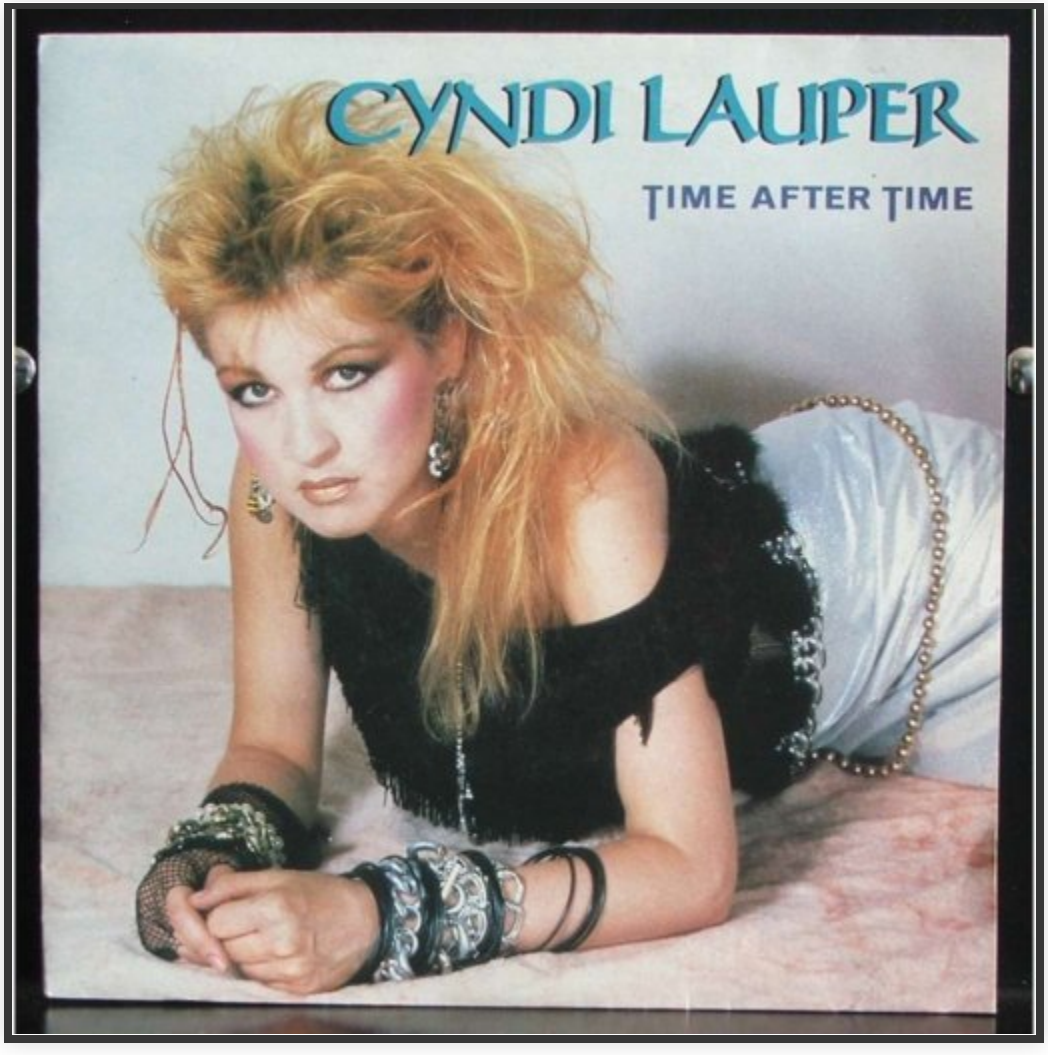
- `-H` (or `--header`) : to send custom headers
- Add 'Key: Value' for each headers
- `-A foo`: is a shortcut to `-H 'User-Agent: foo'`
- `-b foo=bar`: is a shortcut to `-H 'Cookie: foo=bar'`

(Cookies are just headers your browser is used to save)

COOKIES

- Not saved by default
- Use `-c` to **save** cookies to a file (`-` to display on stdout)
- Use `-b` to **read** from a file (**it won't by default**)

TIMER AFTER TIME



Have-you ever seen this ?:

```
time curl http://example.org
```

TRY :

```
curl --trace-time -v http://example.org
```

(Only works in verbose or trace mode)

TRY :

```
curl --trace-time -v http://example.org
```

(Only works in verbose or trace mode)

Unless you do want to check the cpu–time / user–time of an HTTP client request.

TRY :

```
curl --trace-time -v http://example.org
```

(Only works in verbose or trace mode)

Unless you do want to check the cpu–time / user–time of an HTTP client request.

(you don't)

ANOTHER APPROACH

You can `write` many variables on output, with the format string option including:

- Request information:
  - `http_code`
  - `http_version`
- Time and speed:
  - `time_total`
  - `speed_download`
- Many more…

FOR EXAMPLE:

```
curl -w "http/%{http_version} %{http_code} -- %{time_total}" -s -o/dev/null http://example.com
```

- Introducing `-oOUTFILE`, much prettier than "`>/dev/null`"
- Also introducing the `-s` (`--silent`) option to inhibit the ugly progress metric

FOR EXAMPLE:

```
curl -w "http/%{http_version} %{http_code} -- %{time_total}" -s -o/dev/null http://example.com
```

- Introducing `-oOUTFILE`, much prettier than "`>/dev/null`"
- Also introducing the `-s` (`--silent`) option to inhibit the ugly progress metric
  - We can also mention `--stderr` to control the error output

FOR EXAMPLE:

```
curl -w "http/%{http_version} %{http_code} -- %{time_total}" -s -o/dev/null http://example.com
```

- Introducing `-oOUTFILE`, much prettier than "`>/dev/null`"
- Also introducing the `-s` (`--silent`) option to inhibit the ugly progress metric
  - We can also mention `--stderr` to control the error output
  - Use with `-` to direct it to stdout

FOR EXAMPLE:

```
curl -w "http/%{http_version} %{http_code} -- %{time_total}" -s -o/dev/null http://example.com
```

- Introducing `-oOUTFILE`, much prettier than "`>/dev/null`"
- Also introducing the `-s` (`--silent`) option to inhibit the ugly progress metric
  - We can also mention `--stderr` to control the error output
  - Use with `-` to direct it to stdout
  - Or whatever filename

FOR EXAMPLE:

```
curl -w "http/%{http_version} %{http_code} -- %{time_total}" -s -o/dev/null http://example.com
```

- Introducing `-oOUTFILE`, much prettier than "`>/dev/null`"
- Also introducing the `-s` (`--silent`) option to inhibit the ugly progress metric
  - We can also mention `--stderr` to control the error output
  - Use with `-` to direct it to stdout
  - Or whatever filename
  - >15 years using shells, still can't handle std flows ?

FOR EXAMPLE:

```
curl -w "http/%{http_version} %{http_code} -- %{time_total}" -s -o/dev/null http://example.com
```

- Introducing `-oOUTFILE`, much prettier than "`>/dev/null`"
- Also introducing the `-s` (`--silent`) option to inhibit the ugly progress metric
  - We can also mention `--stderr` to control the error output
  - Use with `-` to direct it to stdout
  - Or whatever filename
  - >15 years using shells, still can't handle std flows ?
  - `curl` got your back.

DID YOU EVER ?

DID YOU EVER ?

...

DID YOU EVER ?

...

Need to edit `/etc/hosts` ?

DID YOU EVER ?

...

Need to edit `/etc/hosts` ?

```
curl -v --resolve www.example.com:443:1.2.3.4 https://www.example.com/
```

DID YOU EVER ?

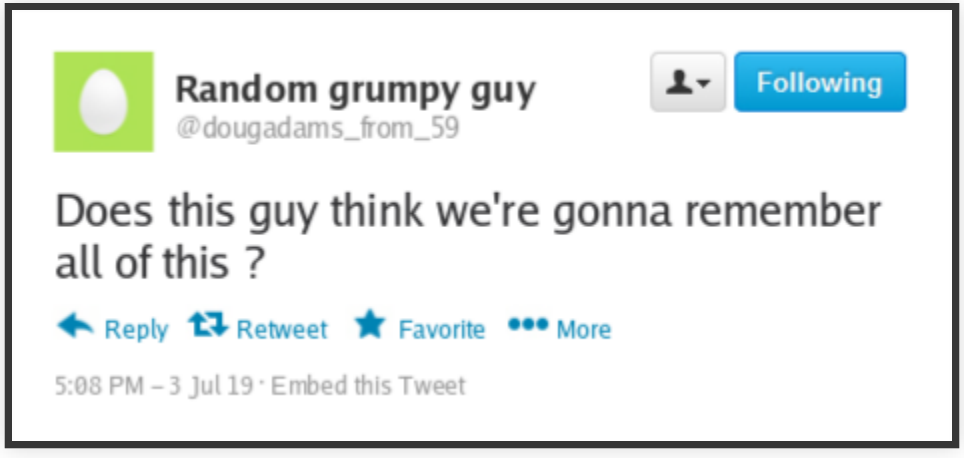
...

Need to edit `/etc/hosts` ?

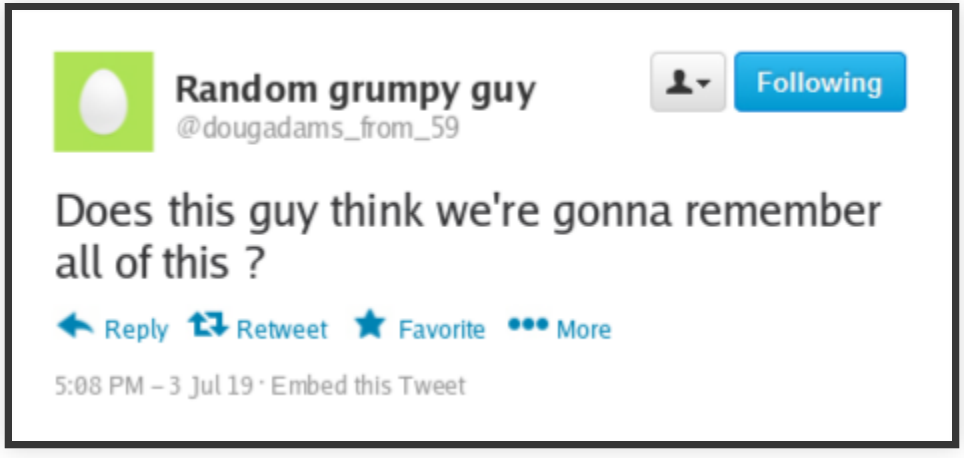
```
curl -v --resolve www.example.com:443:1.2.3.4 https://www.example.com/
```

No need to play around with "`Host`" header

MEMORY ALLOCATION PROBLEMS

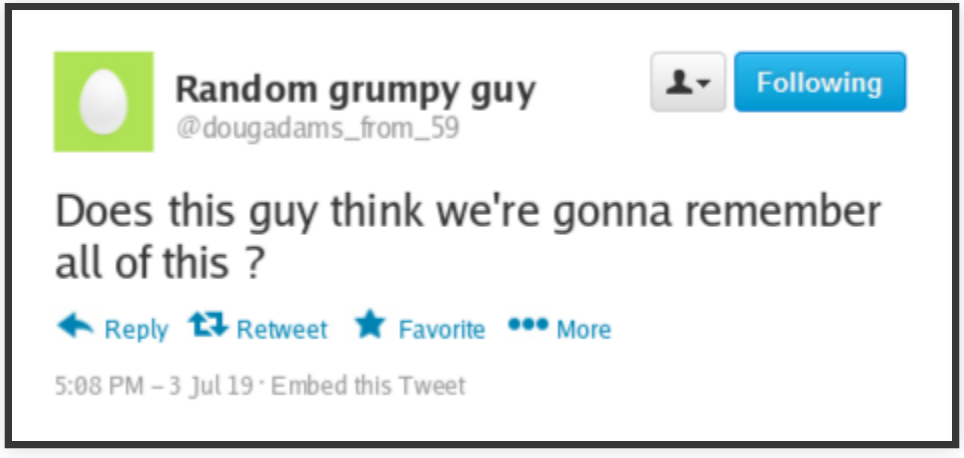


MEMORY ALLOCATION PROBLEMS



No. Don't.

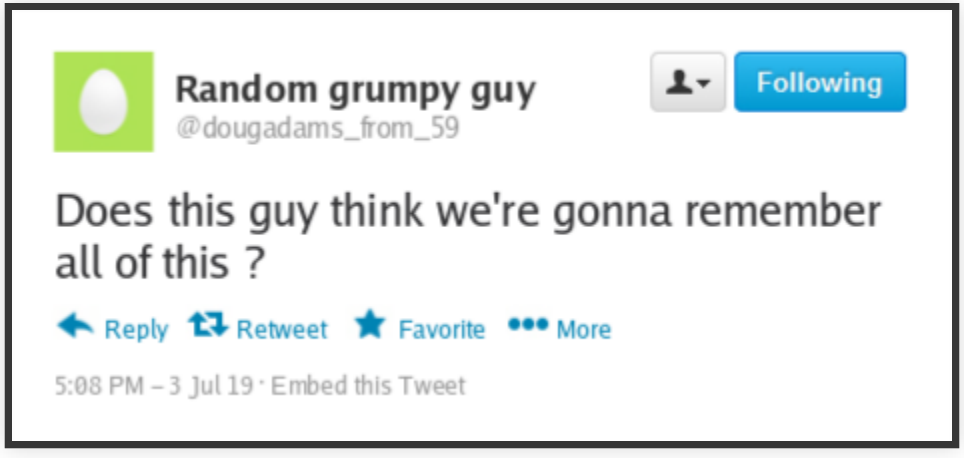
MEMORY ALLOCATION PROBLEMS



No. Don't.

All the options I mentioned can be added to `$HOME/.curlrc`

MEMORY ALLOCATION PROBLEMS



No. Don't.

All the options I mentioned can be added to `$HOME/.curlrc`

Or write several of these, and recall them with `-K filename`, or `--config`

CURL PLAYS NICE WITH OTHERS

Or you can avoid the options madness and ordering, by just right–clicking in Firefox (and Chrome) DevTools.

CURL PLAYS NICE WITH OTHERS

Or you can avoid the options madness and ordering, by just right–clicking in Firefox (and Chrome) DevTools.

And select "Copy as cURL"

CURL PLAYS NICE WITH OTHERS

Or you can avoid the options madness and ordering, by just right–clicking in Firefox (and Chrome) DevTools.

And select "Copy as cURL"

It works in BurpSuite too.

CURL PUTS THE C IN CURL.

```
> curl https://example.com \
            --header "Hello: World" -w '# %{http_code} -- %{time_total}'\
            --libcurl test.c  -so/dev/null
# 200 -- 0,339792%
> cat test.c
```

TEST.C

```
/********* Sample code generated by the curl command line tool **********
 * All curl_easy_setopt() options are documented at:
 * https://curl.haxx.se/libcurl/c/curl_easy_setopt.html
 ************************************************************************/
#include <curl/curl.h>

int main(int argc, char *argv[])
{
  CURLcode ret;
  CURL *hnd;
  struct curl_slist *slist1;

  slist1 = NULL;
  slist1 = curl_slist_append(slist1, "Hello: World");

  hnd = curl_easy_init();
  curl_easy_setopt(hnd, CURLOPT_BUFFERSIZE, 102400L);
  curl_easy_setopt(hnd, CURLOPT_URL, "https://example.com");
  curl_easy_setopt(hnd, CURLOPT_NOPROGRESS, 1L);
```

YOUR OWN STRESS-TEST

- Because after all, they're just glorified (yet customisable) loops with precise metrics
- Let's roll our own apache–bench

```
#include <curl/curl.h>
#include <omp.h>
#define MAX_THREAD 64
#define LASERS 1000
#define URL "http://www.example.com"

int main(int argc, char *argv[]) {
  int tid, i = 0;
  FILE *devnull;
  devnull = fopen("/dev/null", "w");
  #pragma omp parallel private(i) num_threads(MAX_THREAD)
  {
  #pragma omp for
    for(i = 0; i < LASERS; ++i) {
      tid = omp_get_thread_num();

      CURLcode ret;
      CURL *hnd;
      double total;
```

- Just removing some comments

```
#include <curl/curl.h>
#include <omp.h>
#define MAX_THREAD 64
#define LASERS 1000
#define URL "http://www.example.com"

int main(int argc, char *argv[]) {
  int tid, i = 0;
  FILE *devnull;
  devnull = fopen("/dev/null", "w");
  #pragma omp parallel private(i) num_threads(MAX_THREAD)
  {
  #pragma omp for
    for(i = 0; i < LASERS; ++i) {
      tid = omp_get_thread_num();

      CURLcode ret;
      CURL *hnd;
      double total;
```

- Just removing some comments
- And wrap some OpenMP magic around

```
#include <curl/curl.h>
#include <omp.h>
#define MAX_THREAD 64
#define LASERS 1000
#define URL "http://www.example.com"

int main(int argc, char *argv[]) {
  int tid, i = 0;
  FILE *devnull;
  devnull = fopen("/dev/null", "w");
  #pragma omp parallel private(i) num_threads(MAX_THREAD)
  {
  #pragma omp for
    for(i = 0; i < LASERS; ++i) {
      tid = omp_get_thread_num();

      CURLcode ret;
      CURL *hnd;
      double total;
```

- Just removing some comments
- And wrap some OpenMP magic around
- Compile with: `gcc mt_curl.c -fopenmp -lcurl`

```c
#include <curl/curl.h>
#include <omp.h>
#define MAX_THREAD 64
#define LASERS 1000
#define URL "http://www.example.com"

int main(int argc, char *argv[]) {
  int tid, i = 0;
  FILE *devnull;
  devnull = fopen("/dev/null", "w");
  #pragma omp parallel private(i) num_threads(MAX_THREAD)
  {
  #pragma omp for
    for(i = 0; i < LASERS; ++i) {
      tid = omp_get_thread_num();

      CURLcode ret;
      CURL *hnd;
      double total;
```

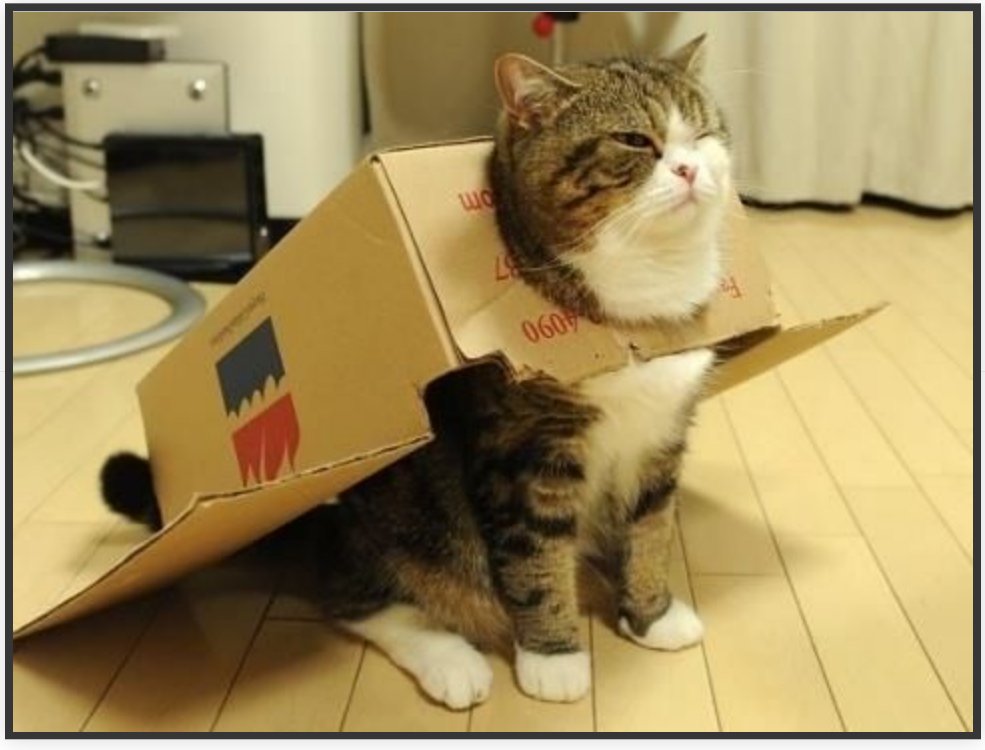
- Just removing some comments
- And wrap some OpenMP magic around
- Compile with: `gcc mt_curl.c -fopenmp -lcurl`
- Make sure the entire file is <42 LoC

```c
#include <curl/curl.h>
#include <omp.h>
#define MAX_THREAD 64
#define LASERS 1000
#define URL "http://www.example.com"

int main(int argc, char *argv[]) {
  int tid, i = 0;
  FILE *devnull;
  devnull = fopen("/dev/null", "w");
  #pragma omp parallel private(i) num_threads(MAX_THREAD)
  {
  #pragma omp for
    for(i = 0; i < LASERS; ++i) {
      tid = omp_get_thread_num();

      CURLcode ret;
      CURL *hnd;
      double total;
```

DEMO ?

THANK YOU

- Everyone @ PTS for all the event
- Dan Stanberg for all of the curling
- Have a safe trip back home ♥



QUESTIONS ?