curl @ Google

Tony Aiuto
aiuto@google.com
April, 2018

The Scale

- ~100 direct dependencies (projects)
- Fans out to over 2 million dependent targets
- Every change impacts >400k tests

The Scale

- 50K programmers
- no explicit staffing
- 2 maintain curl in their 20% time

The scope

- The canonical build is Linux/x86_64

.... yawn

The scope

We also build for

- X86_32
- Arm7 / Arm64
- Mips
- Myriad2
- Ppc
- Shave
- Sparc
- ... and possible others

The scope

... as needed on

- Linux desktop & embedded
- OS X
- Windows
- Android
- IOS
- NaCL, Web Assembly
- Emscriptem
- Others

Google repository

- Single source tree
- Build from head
  - Branches are only allowed for freezing a release
- Forks are vigorously hunted down and killed
  - Each fork would require separate security audit
  - No staffing for updating forks

Local Mods

- Drop all except HTTPS, SMTP
- Transform source to improve build time
- BoringSSL rather than OpenSSL, but sometimes native SSL for desktop apps
- Pluggable DNS resolver
- A handful of other things

Standard procedure? No way

We build from a single source tree

- Scriptable rewriting of curl distro
- Crazy scripting to get the config.h right
- Ignore CMake & Visual Studio

Absolute paths

- At our scale -I directives in compiles costs real money
- Rewrite the code to change every #include

```
#!/usr/bin/python
r"""Normalize include paths for a source tree to a build top level
directory.

H=[Find all .h files in the source tree at TOP]
For all .c and .h files change:
  #include <[something/]X>
to
  #include "TOP/[something]X"

Example usage:
  cd third_party/curl
  normalize_includes.py \
      -I third_party/curl/src  -I third_party/curl/src/include \
      -I third_party/curl/src/lib \
      third_party/curl/src
"""
```

Configure is great

- Just not for Google
- Single source tree
  - Must build for all platforms
  - With minimal -I an -D option changes
- So we hand build the OS to config.h logic

Defer config.h to compile time

```
// Note: This is hand crafted. It is not part of the standard curl distribution.
// See third_party/curl/UPDATING.

#if defined(_WIN32)
#include "third_party/curl/src/abi/msvc/lib/curl_config.h"
#elif defined(__APPLE__)
#if defined(__i386__)
#include "third_party/curl/src/abi/darwin_piii/lib/curl_config.h"
#else
#include "third_party/curl/src/abi/darwin_k8/lib/curl_config.h"
#endif
#elif defined(__myriad2__)
#include "third_party/curl/src/abi/myriad2/lib/curl_config.h"
#elif defined(__i386__) || defined(__arm__)
#include "third_party/curl/src/abi/32bit/lib/curl_config.h"
#else
#include "third_party/curl/src/abi/64bit/lib/curl_config.h"
#endif
```

Force our own constraints

```
// GOOGLE_USE_BORINGSSL: Force use of boringssl, even if the config asked for
// a different SSL library.
#ifdef GOOGLE_USE_BORINGSSL

#ifndef HAVE_BORINGSSL
#define HAVE_BORINGSSL 1
#endif
#ifndef HAVE_LIBSSL
#define HAVE_LIBSSL 1
#endif
#ifdef USE_DARWINSSL
#undef USE_DARWINSSL
#endif
#ifndef USE_OPENSSL
#define USE_OPENSSL 1
#endif
```

Cross compile & configure

- Configure is not adequate for cross compile
- Configure only to get word sizes
  - CURL_SIZEOF_LONG computed for each platform
  - Features copied from master config.h to others

Thank you, Dan

- It just works.
  - For hundreds of applications
  - Without anyone trying hard
- That’s some good craftsmanship.

More

- Want more details?
  - https://goo.gl/H7jfkp

And thanks again to Dan, for providing an incredibly useful tool

Proxy Support

- Proxy works
- IFF you know what proxy to use
- Enterprise network admins are **very** inventive in the way they configure their networks

Proxy Configuration

- Via DHCP
  - You can usually ask the OS
  - Unless they give you a PAC file
- WPAD
  - Guidelines to find a PAC file
- PAC files
  - Execute this JavaScript please

Proxy Configuration

- Windows & OSX
  - Sequence of calls to try
  - They both bottom out with an OS provided library doing PAC evaluation
- Linux
  - Get DHCP setup
  - Use Chromium Embedded Framework for JS eval
- Legacy Unix
  - Not a goal

CASB: The new hotness

- Cloud Access Security Broker
  - Fully decrypting MITM proxies
  - Sometimes transparent, sometimes a proxy address
  - Admins *usually* force the proxy cert to users machines
  - Fun interactions if we tell curl to use a custom root certs list
  - CRL usually not provided

... and more misconfiguration

- Bypass lists
  - Admin says bypass proxy for these hosts
  - Or IP address
- Sometimes they can’t get it right
  - don’t validate SSL certs
- But the admin wants exactly as much security as their setup has afforded

ymmv

copy_defined_features.py:

```
"""Copy features of the form |#define X Y| from one file to another.

When we see #define X in the first file and /* #undef X */ in the second,
then replace the undef with the define. Admittedly, this is not truly
complete, in that we do not turn off features that are explicitly undef
in the first file. That is not a need at this time.
"""
```