PLEX

A barbarian and a dragon

How to update your toolchain with style

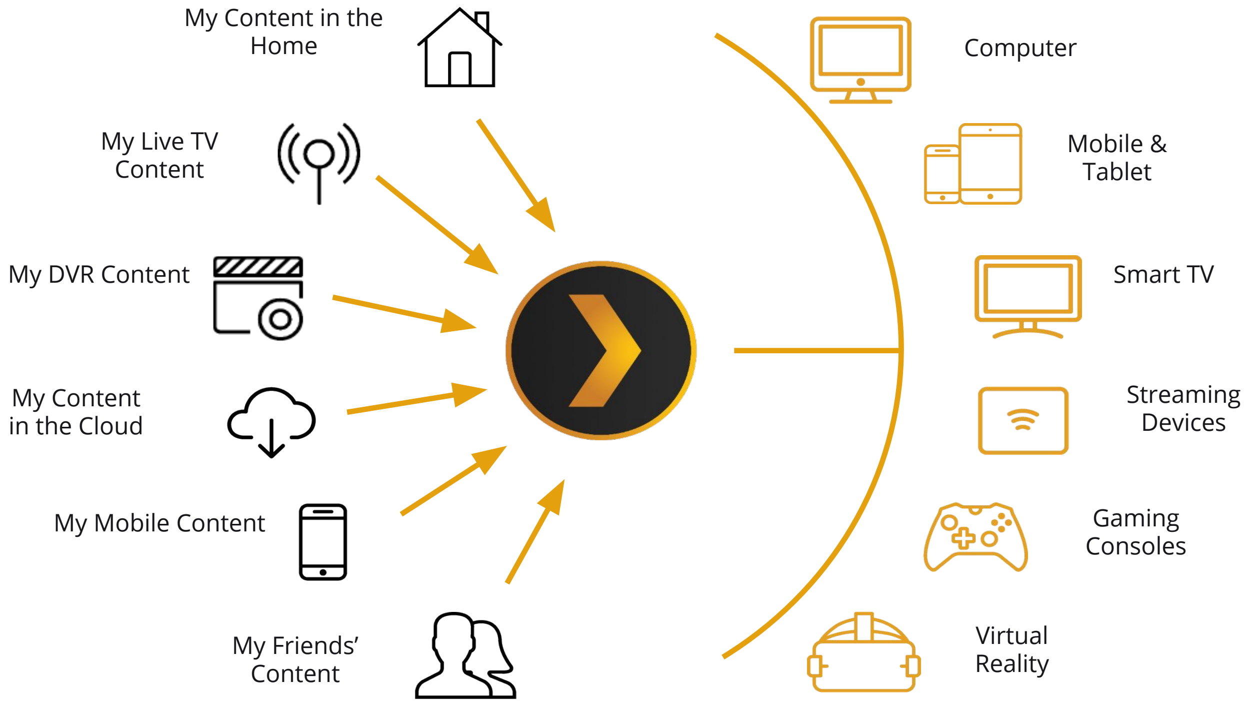
Tobias Hieta

Dragon killer, cheese mover and likes to open cans that contains worms

WTF is Plex?

PLEX

The leading media streaming software platform



PLEX

WTF is Conan?

WTF is Clang?

Plex Media Server

PLEX

60+ dependenc

Boost, zlib, freetype, ffmpeg, bzip, curl, expat, freeima

…

PLEX

JavaScript developer ->
HOLD MY BEER
KCCO
BLACK LAGER

1269 dependencies

27 targets

Linux (lot of different architectures), Android, iOS, macOS, Windows, FreeBSD

PLEX

Compilers …

Standard libraries

Many* build systems

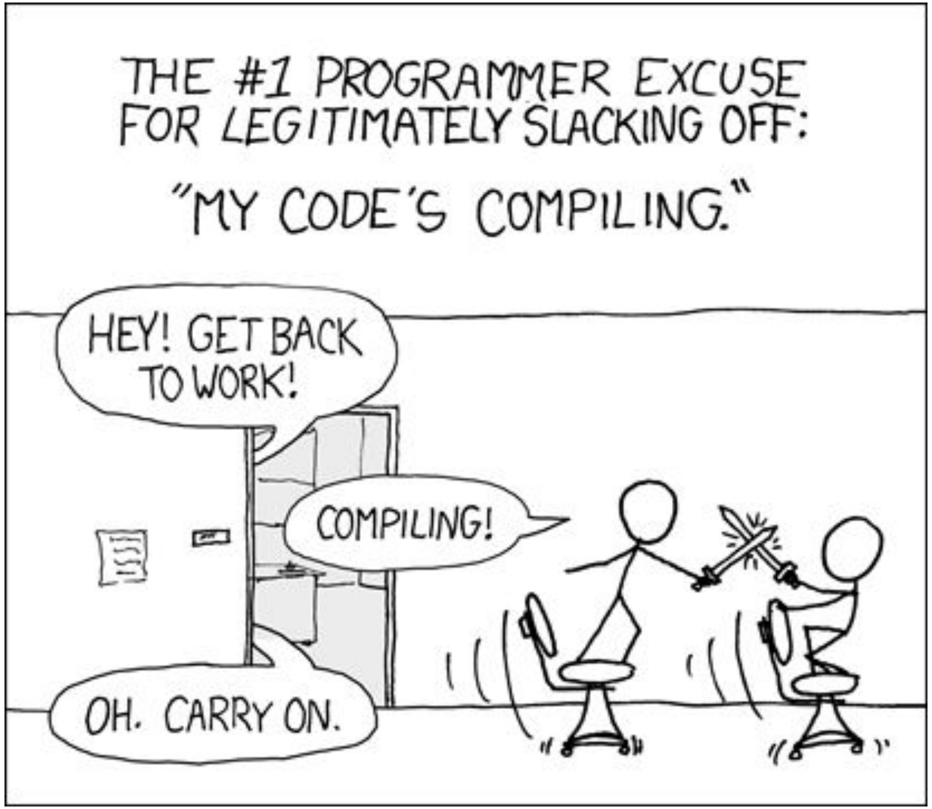
*= I think the scientific unit is ‘a fuckton’.

CMake, Auto~~tools~~hell, SCons, Make, Visual Studio, ndk-make, meson, build2, waf and whatever that boost thing is

They all suck

CMake probably sucks least.

Compile



PLEX

plex-dependency-builder

Rebuild the world

PLEX

scratchbox2

PLEX

Hidden changes

PLEX

Frustrated and angry

(We need)

A new hope

PLEX

Individual packages

Individual packages

Handle multiple build-systems

Individual packages
Handle multiple build-systems

Cross-compile

Individual packages
Handle multiple build-systems
Cross-compile

Manage deps and toolchain

Individual packages
Handle multiple build-systems
Cross-compile
Manage deps and toolchain

Flexible

PLEX

Individual packages
Handle multiple build-systems
Cross-compile
Manage deps and toolchain
Flexible

Reproducible builds

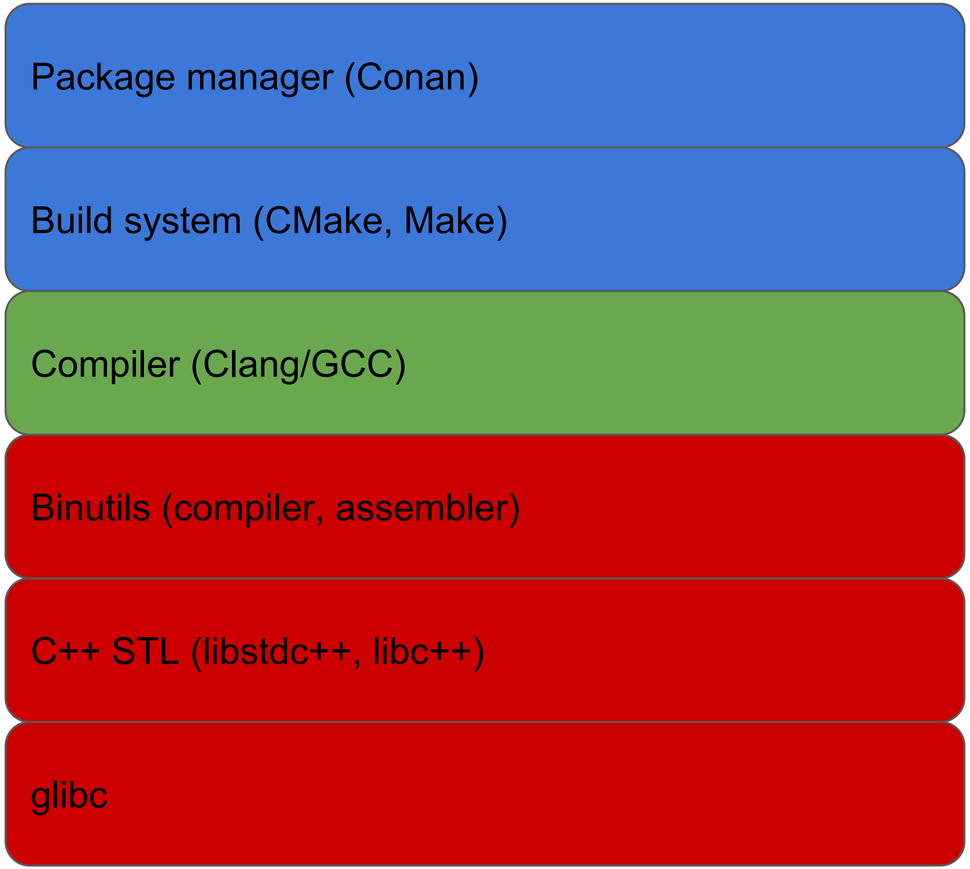
PLEX

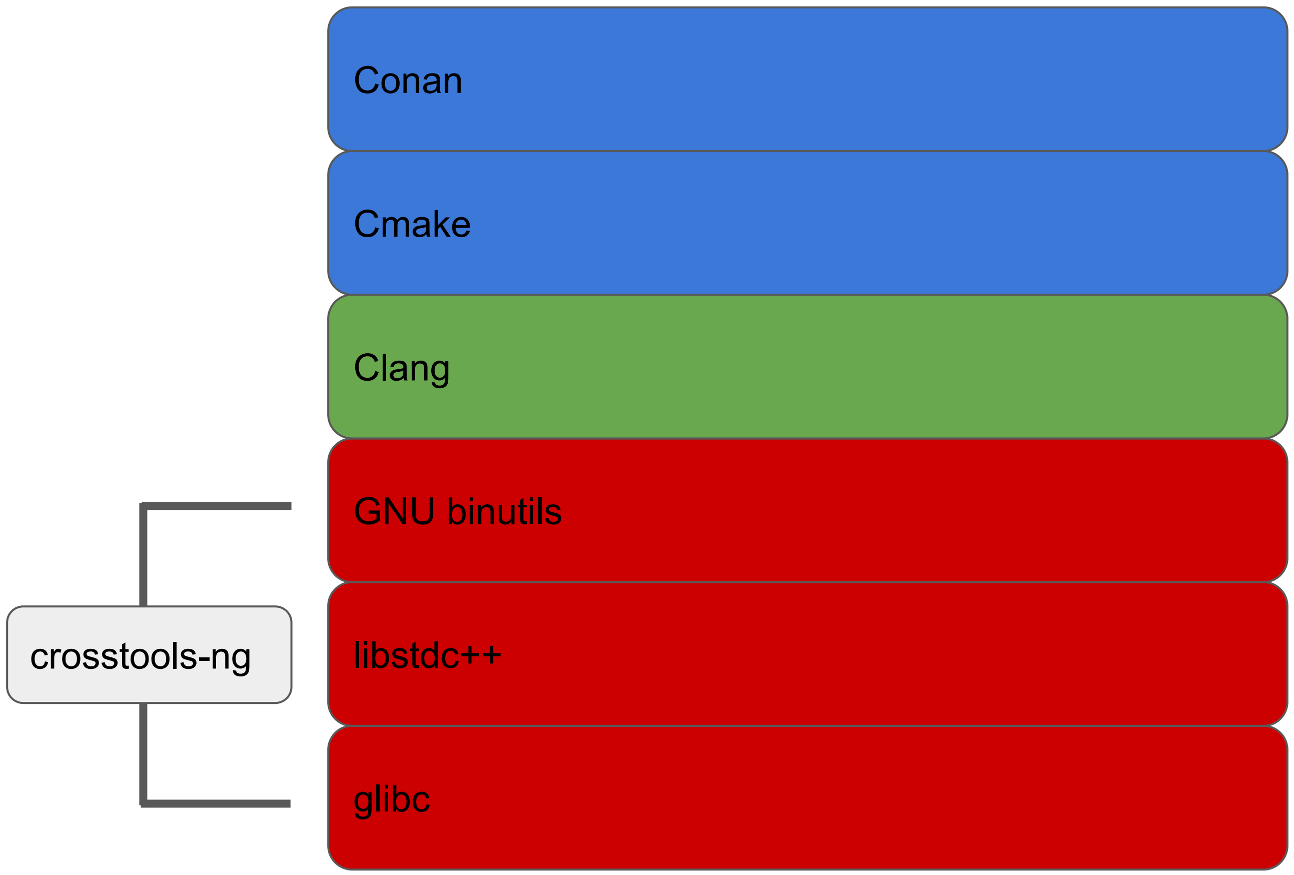
Unified toolchain

Compiler	C++11	C++14	C++17
GCC	4.8.1	6.1	7.0
Clang	3.3	3.4	5.0
Visual Studio	2015/2017	2017	2017 15.5

What makes a toolchain?

PLEX


Package manager (Conan)
Build system (CMake, Make)
Compiler (Clang/GCC)
Binutils (compiler, assembler)
C++ STL (libstdc++, libc++)
glibc


Conan
Cmake
Clang
GNU binutils
libstdc++
glibc
crosstools-ng

PLEX

GCC ⚔ Clang

Single binary
Multiple targets

A single compiler

to rule them all.

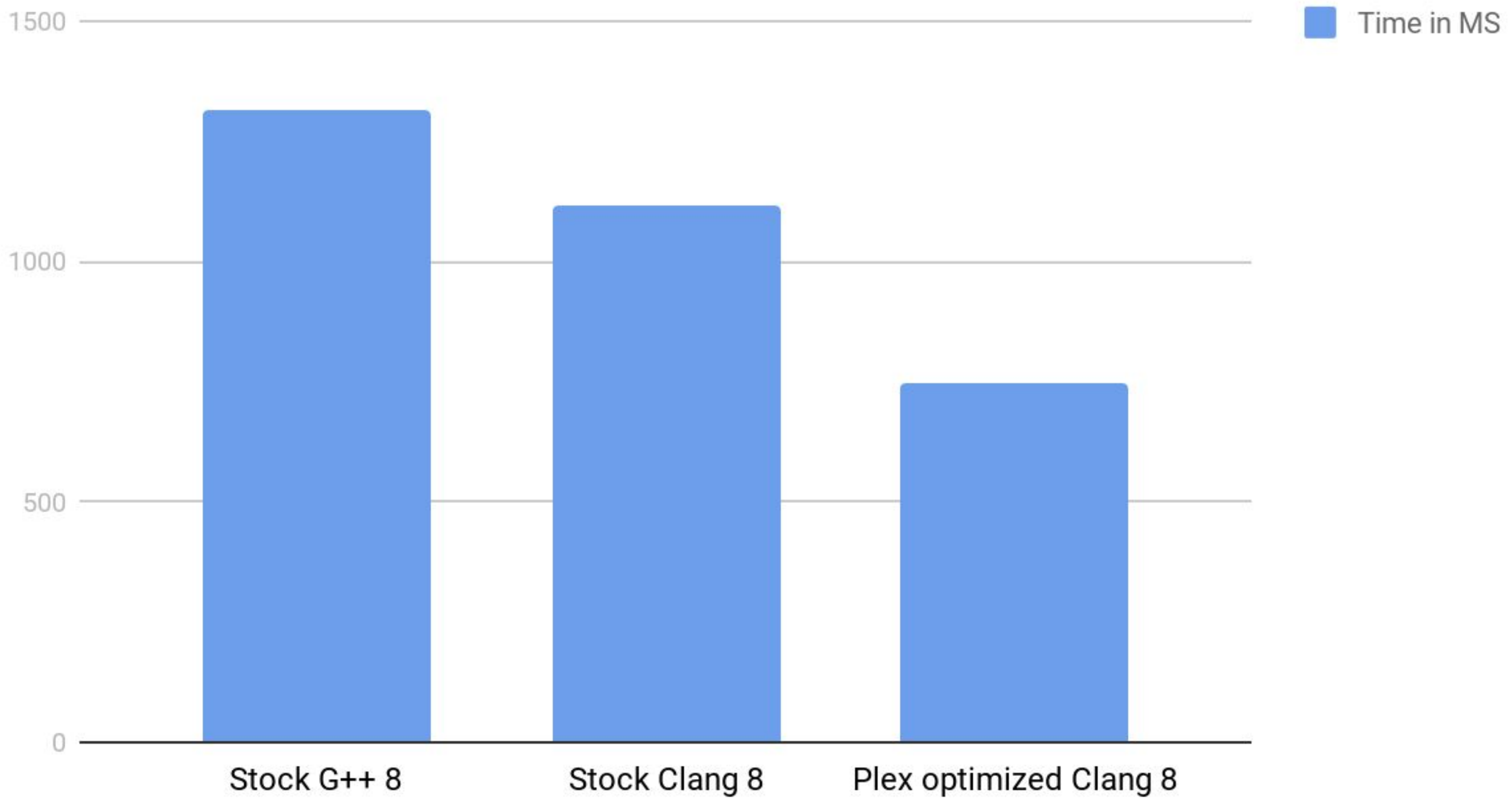
PLEX

Cool logo

PLEX

Optimize Clang

Compile time hello.cpp (mean of 15 runs)



PLEX

PGO and LTO

Bootstrap	Instrumented	Build Sources	Build final Clang	Profit
Download or build a stock version of Clang with standard flags. You can also use the one from your dist.	Build the Clang sources with Instrumentation, this inserts profiling collectors in all methods of Clang.	Build multiple sources with the Instrumented version of Clang to generate Profile data. This needs to be as diverse as possible. We build PMS 5 times in different configurations and different backends.	Using the stock Clang we can now build Clang again with the Profile information generated in the previous step. This will generate the final optimized version of Clang. Don’t forget to enable LTO	Enjoy faster builds!

16 targets
2 compilers

PLEX

Bumps on the road

Android builds crashed on exceptions

Crashes when soci ran into locked db

Random crashes in OpenSSL

PLEX

Nirvana

```
all_build_requirements = {
  "toolchain": [
    "cmakecache/1-15",
    "cmaketoolchain/1-14",
  ],
  "common": [
    "boost/1.59.0-45",
    "bzip2/1.0.6-13",
    "ca-bundle/2018-10-17-4",
    "cotire/1.8.0-391bf6b-12",
    "cppnetlib/0.10.1-45",
    "curl/7.56.1-50",
    "fmt/4.1.0-135ab5c-14",
    "freeimage/3.17.0-16",
    "libxml2/2.9.8-11",
    "minizip/1.2.8-16",
    "opencv/2.4.13-07711e4-21",
[bootstrap]
ref = 498732fb3c4b035f

[tools]
conan = 1.4.4
plexconantool = 5-48
plextoolchain = 1-55
noarch-tools = 1-8

[project]
variants = ["standard", "nano"]
```

PLEX

PLEX

Thanks!

@tobiashieta