# BITING THE CMAKE BULLET

LEARNING A META-BUILD SYSTEM

FOR FUN AND NOT PROFIT

ThePhD
@thephantomderp
https://github.com/ThePhD

Boston C++ Meetup
February 6th, 2018

# WHAT ARE WE DOING THIS FOR?

# RUNNING LIBRARY TESTS

- Header-only library still needed to build tests
  - Verify we are correct on all platforms
  - Use appveyor / travis-ci
- Tests are not very complicated
  - But TONS of target platforms

# TARGETS

- Compilers
  - GCC 7.x, 6.x, 5.x, 4.9 || LLVM 5.x, 4,x, 3.9.x, 3.8.x, 3.7.x, 3.6.x || VC++ v141, v140, v140_xp
- Platforms
  - Windows – Visual Studio (MSBuild) vs. Not-Visual Studio (MinGW, etc.)
  - Linux (compiler-based), Mac OSX – Xcode 9.x, 8.x, 7.x, 6.x
  - Debug + Release, x86 + x64
- Lua Version – 5.1, 5.2, 5.3, JIT-2.0, JIT-2.1

# FIRST “SOLUTION”

- Python “bootstrap.py”
  - Ad-hoc home rolled meta build system creating ninja.build file to run tests
  - Worked well enough to get off the ground without committing to a build system (back in 2014/2015)
- CMake recommended by contributor in early 2016
  - Rejected at the time due to stepping beyond just creating test harness

# PROBLEMS

- Unable to support all the platforms
  - Okay for GCC/LLVM and Linux
  - VC++ and MSBuild?
- Tacked-on spaghetti code for managing dependencies
  - Depended on fetched Lua using package manager
  - Expected everything to be laid out before hand

# DIVING INTO CMAKE

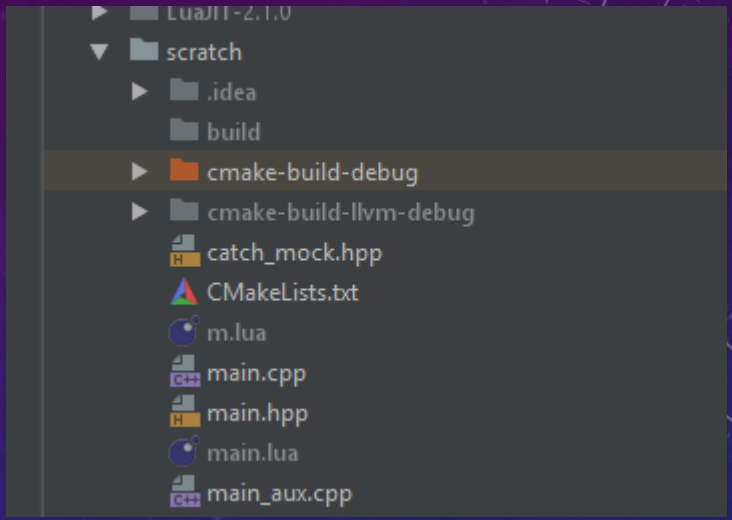
## CAN CMAKE SOLVE OUR PROBLEMS?

# THE FOUNDATION: PROJECT WITH TARGETS

- Project
  - Single top-level declaration, required CMake version, project version, languages utilized (!!)
  - One project, mutiple targets
  - Can set configuration for project version and how dependencies are managed (complex)

- Targets (Executables and Libraries)
  - Transparently link outputs to inputs with a single command (no copy)
  - Imported/Interface Libraries to handle prebuilt-objects/header-only libraries (!!)

# DIRECTORY-BASED




- Subdirectories included with add_subdirectory
  - Each directory represents a project
  - One CMakeLists.txt in directory
  - All variables directory-scoped, can pass up with set(NAME VALUE PARENT_SCOPE)
- Targets added with
  - add_executable(name …) – supply list of sources to compile executable
  - add_library(name TYPE …) – supply list of sources to compile library of TYPE (SHARED/STATIC)
  - add_custom_target(name …) – execute custom command / sequence of commands

# QUERY/MANIPULATE TARGETS

- target_sources(target sources…)
  - Append sources to target for compilation
  - Good for conditional inclusion of additional source files
- target_include_directories(target PRIVATE|PUBLIC|INTERFACE dirs…)
  - Add include directories with the propagation modifier
- target_link_libraries(target [PRIVATE|PUBLIC|INTERFACE library_target1]…)
  - Link libraries (and their outputs) into the target during build with the propagation modifier

# QUERY/MANIPULATE TARGETS II

- get_target_properties, set_target_properties
  - Bread and butter of setting languages, standards, and similar
  - Pull out a single property into a variable, or set multiple
  - Different properties based on target type are valid (INCLUDE_DIRECTORIES)

# CODE REUSE

- include(file)
  - Like C++ include – copy-paste into current scope
- add_subdirectory(directory [binary_output_directory])
  - Takes CMakeLists.txt from specified directory (local or absolute)
- Macros, Functions
  - Define and call in your own code
  - Note that Macros DO NOT introduce a new scope: functions do (useful later)

# HANDLING PREINSTALLED LIBRARIES

- find_package(NAME [[VERSION] [EXACT]] [REQUIRED])
  - Very big and present since earliest days of CMake
  - Using them is fairly simple
    - find_package(Threads) – finds the threading library (pthreads or similar)
    - find_package(Lua 5.3 EXACT REQUIRED) – finds Lua, fails build if 5.3 exactly cannot be found on system
  - Implementation a little more complex

# EXTERNALPROJECT

- Standard CMake Module – include(ExternalProject)
  - Allows git/mercurial/svn/cvs/raw-link clone/checkout/download (with HTTPS or MDS/SHA1 hash verification)
  - Performs steps in the order of download, configure, build, install, and test

- Used easily for download of Lua/LuaJIT
  - Lua: listed sources and compiled directly (written in ANSI C)
  - LuaJIT: too complex to just “pull, get sources, build”
    - Linux - Run “make”, use CMake copy operation to move outputs to expected location
    - Windows – Run “msvcbuild.bat”, use CMake copy operation to move outputs to expected location

# EXTERNALPROJECT

```
ExternalProject_Add(LUA_VANILLA
    BUILD_IN_SOURCE TRUE
    BUILD_ALWAYS TRUE
    TLS_VERIFY TRUE
    PREFIX ${LUA_BUILD_TOPLEVEL}
    SOURCE_DIR ${LUA_BUILD_TOPLEVEL}
    DOWNLOAD_DIR ${LUA_BUILD_TOPLEVEL}
    TMP_DIR "${LUA_BUILD_TOPLEVEL}-tmp"
    STAMP_DIR "${LUA_BUILD_TOPLEVEL}-stamp"
    INSTALL_DIR "${LUA_BUILD_INSTALL_DIR}"
    URL https://www.lua.org/ftp/lua-${LUA_VANILLA_VERSION}.tar.gz
    URL_MD5 ${LUA_VANILLA_MD5}
    URL_HASH SHA1=${LUA_VANILLA_SHA1}
    CONFIGURE_COMMAND ""
    BUILD_COMMAND ""
    INSTALL_COMMAND ""
    TEST_COMMAND ""
    BUILD_BYPRODUCTS "${LUA_VANILLA_LIB_SOURCES}" "${LUA_VANILLA_LUA_SOURCES}" "${LUA_VANILLA_LUAC_SOURCES}")
```

# EXTERNALPROJECT_ADDSTEP

- Allows for additional steps to be tacked onto an external project

```
ExternalProject_Add_Step(LUA_VANILLA
    prebuild
    # after download, before build
    DEPENDEES download
    DEPENDERS build
    BYPRODUCTS "${LUA_VANILLA_DESTINATION_LUA_HPP}"
    COMMENT "Moving \"${LUA_VANILLA_SOURCE_LUA_HPP}\" to \"${LUA_VANILLA_DESTINATION_LUA_HPP}\"..."
    COMMAND "${CMAKE_COMMAND}" -E copy "${LUA_VANILLA_SOURCE_LUA_HPP}" "${LUA_VANILLA_DESTINATION_LUA_HPP}")
f()
```

# AD-HOC HACKS

- Can set settings by appending to command line or prebuilt-variables
  - CMAKE_*
  - Inspect in-built variables such as if (MSVC)
- Older functions which affect entire project
  - add_definitions
  - add_compile_options

```
# # # General project flags
if (MSVC)
    add_definitions(/DUNICODE /D_UNICODE
        /D_SILENCE_CXX17_UNCAUGHT_EXCEPTION_DEPRECATION_WARNING
        /D_SILENCE_CXX17_CODECVT_HEADER_DEPRECATION_WARNING
        /D_CRT_SECURE_NO_WARNINGS /D_CRT_SECURE_NO_DEPRECATE)
    # Warning level, exceptions
    add_compile_options(/W4 /EHsc)
    if (NOT CMAKE_CXX_COMPILER_ID MATCHES "Clang")
        add_compile_options(/MP)
    endif()
else()
    if (PLATFORM MATCHES "x86")
        list(APPEND CMAKE_C_FLAGS "-m32")
        list(APPEND CMAKE_CXX_FLAGS "-m32")
        list(APPEND CMAKE_EXE_LINKER_FLAGS "-m32")
        list(APPEND CMAKE_SHARED_LINKER_FLAGS "-m32")
    endif()
    add_compile_options(-Wno-unknown-warning
        -Wno-unknown-warning-option
        -Wall -Wextra -Wpedantic
        -pedantic -pedantic-errors)
endif()
```

# AD HOC HACKS II

```
# # # General project output locations
if (PLATFORM MATCHES "x86" OR CMAKE_SIZEOF_VOID_P EQUAL 4)
    set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/x86/lib")
    set(CMAKE_LIBRARY_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/x86/lib")
    set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/x86/bin")
else()
    set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/x64/lib")
    set(CMAKE_LIBRARY_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/x64/lib")
    set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/x64/bin")
endif()
```

- Targets sometimes do not output to same location
  - Causes problems when running executable that relies on multiple DLLs
  - Simple fix: specify project output directory at top of project

# FUTURE LEARNING - GENERATOR EXPRESSIONS

- Apparently very powerful
  - Meant to make things simpler and more in-line
  - Only works in certain contexts, enforcing confusion

# IT (MOSTLY) WORKS!

ThePhD / sol2 build passing

Current | Branches | Build History | Pull Requests | Build #1245 | More options

develop woops
Commit 22127fa
Compare c9980bf...22127fa
Branch develop
ThePhD authored and committed

#1245 passed
Ran for 1 hr 58 min 4 sec
Total time 8 hrs 31 min 42 sec
a day ago

Restart build

Build Jobs

| | | | |
|---|---|---|---|
| 1245.1 | C++ | LUA_VERSION=luajit-2.0.5 GCC_VERSION=7 PLATFORM=x86 CI=true | 32 min 5 sec |
| 1245.2 | C++ | LUA_VERSION=luajit-2.1.0-beta3 GCC_VERSION=7 PLATFORM=x86 CI=true | 34 min 1 sec |
| 1245.3 | C++ | LUA_VERSION=5.3.4 GCC_VERSION=4.9 PLATFORM=x64 CI=true | 26 min 26 sec |
| 1245.4 | C++ | LUA_VERSION=5.3.4 GCC_VERSION=5 CI=true PLATFORM=x64 | 27 min 17 sec |
| 1245.5 | C++ | LUA_VERSION=5.3.4 GCC_VERSION=6 PLATFORM=x64 CI=true | 30 min 24 sec |
| 1245.6 | C++ | LUA_VERSION=5.3.4 GCC_VERSION=7 PLATFORM=x64 CI=true | 31 min 49 sec |
| 1245.7 | C++ | LUA_VERSION=5.3.4 LLVM_VERSION=3.6.2 PLATFORM=x64 CI=true | 26 min 4 sec |
| 1245.8 | C++ | LUA_VERSION=5.3.4 LLVM_VERSION=3.7.1 PLATFORM=x64 CI=true | 30 min 13 sec |
| 1245.9 | C++ | LUA_VERSION=5.3.4 LLVM_VERSION=3.8.1 PLATFORM=x64 CI=true | 33 min 47 sec |
| 1245.10 | C++ | LUA_VERSION=5.3.4 LLVM_VERSION=3.9.1 PLATFORM=x64 CI=true | 27 min 26 sec |
| 1245.11 | C++ | LUA_VERSION=5.3.4 LLVM_VERSION=4.0.1 PLATFORM=x64 CI=true | 29 min 39 sec |
| 1245.12 | C++ | LUA_VERSION=5.3.4 LLVM_VERSION=5.0.1 PLATFORM=x64 CI=true | 25 min 15 sec |
| 1245.13 | C++ | LUA_VERSION=5.2.4 GCC_VERSION=7 PLATFORM=x64 CI=true | 31 min 15 sec |
| 1245.14 | C++ | LUA_VERSION=5.1.5 GCC_VERSION=7 PLATFORM=x64 CI=true | 31 min 47 sec |
| 1245.15 | C++ | LUA_VERSION=luajit-2.0.4 GCC_VERSION=7 PLATFORM=x64 CI=true | 29 min 50 sec |
| 1245.16 | C++ | LUA_VERSION=luajit-2.0.5 GCC_VERSION=7 PLATFORM=x64 CI=true | 31 min 25 sec |
| 1245.17 | C++ | LUA_VERSION=luajit-2.1.0-beta3 GCC_VERSION=7 PLATFORM=x64 CI=true | 32 min 59 sec |

**LATEST BUILD** HISTORY

woops
2 days ago by ThePhD
develop 22127fa6
2.19.0-100
2 days ago in 3 hr 31 min

JOBS

| JOB NAME | TESTS | DURATION |
|---|---|---|
| Image: Visual Studio 2015; Environment: LUA_VERSION=5.3.4, MINGW_VERSION=6.3.0; Platform: x64 | | 10 min 59 sec |
| Image: Visual Studio 2015; Environment: LUA_VERSION=5.3.4, MINGW_VERSION=6.3.0; Platform: x86 | | 10 min |
| Image: Visual Studio 2015; Environment: LUA_VERSION=luajit-2.0.5, MINGW_VERSION=6.3.0; Platform: x64 | | 11 min 9 sec |
| Image: Visual Studio 2015; Environment: LUA_VERSION=luajit-2.0.5, MINGW_VERSION=6.3.0; Platform: x86 | | 12 min 21 sec |
| Image: Visual Studio 2015; Environment: LUA_VERSION=luajit-2.1.0-beta3, MINGW_VERSION=6.3.0; Platform: x64 | | 12 min 41 sec |
| Image: Visual Studio 2015; Environment: LUA_VERSION=luajit-2.1.0-beta3, MINGW_VERSION=6.3.0; Platform: x86 | | 10 min 8 sec |
| Image: Visual Studio 2015; Environment: LUA_VERSION=5.3.4; Platform: x64 | | 11 min 8 sec |
| Image: Visual Studio 2015; Environment: LUA_VERSION=5.3.4; Platform: x86 | | 11 min 32 sec |
| Image: Visual Studio 2017; Environment: LUA_VERSION=5.3.4; Platform: x64 | | 31 min 12 sec |
| Image: Visual Studio 2017; Environment: LUA_VERSION=5.3.4; Platform: x86 | | 27 min 35 sec |
| Image: Visual Studio 2017; Environment: LUA_VERSION=5.2.4; Platform: x64 | | 30 min 21 sec |
| Image: Visual Studio 2017; Environment: LUA_VERSION=5.1.5; Platform: x64 | | 31 min 23 sec |

# READ THE DOCS!

https://cmake.org/cmake/help/latest/

# FUTURE PROJECTS

- Sol2 interop and require_dll examples
  - https://github.com/ThePhD/sol2/tree/develop/examples/require_dll_example
  - https://github.com/ThePhD/sol2/tree/develop/examples/interop

- Lua Benchmarking Library
  - https://github.com/ThePhD/lua-bench

# THANK YOU!

QUESTIONS? AND, IF TIME PERMITS, AN EXAMPLE?