Software Build Systems

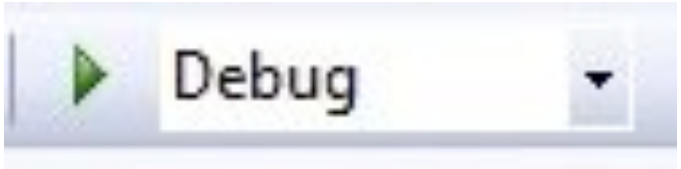
Marwan Abdellah
Blue Brain Project, EPFL

December 2012

Agenda

- The Magician
- Software Build Process
  - Basic Overview
  - Building In-Depth
- Build Automation
- Build Tools
  - GNU Tools
- Make Files
- CMake

The Magician



Software Build Process

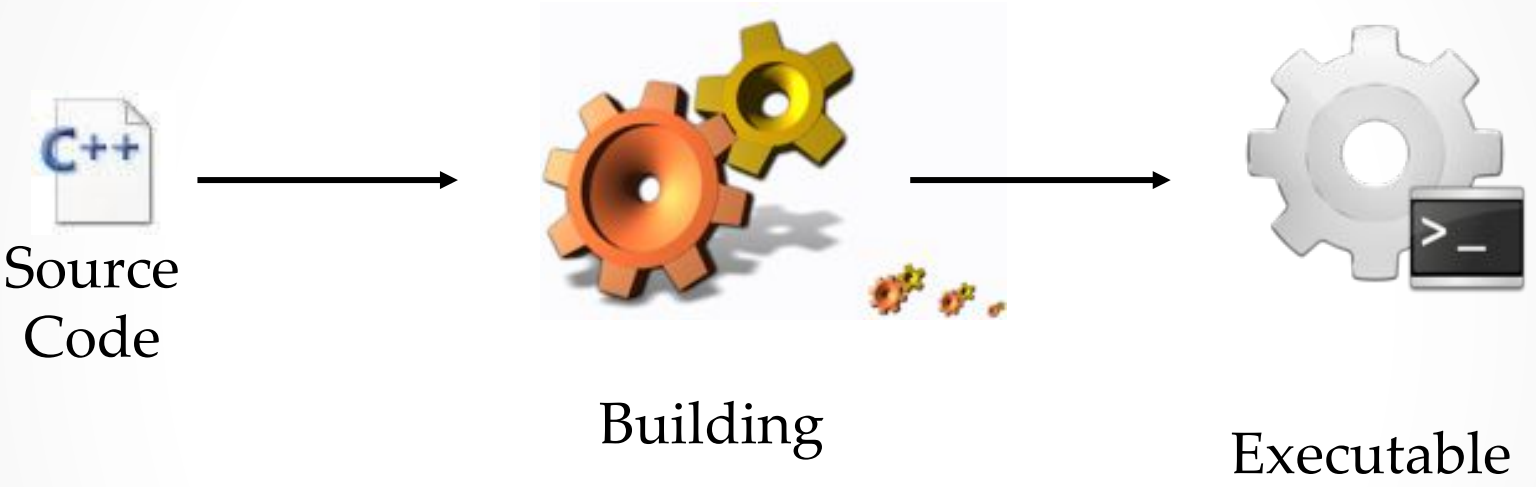
- The process of converting **source code files** into standalone **software artifact(s)** that can be run on a computer, or the result of doing so.

- Basic Building Process
  - **Compilation**: where source code files are converted into **executable code** or linked to **libraries**.
  - **Linking**: where the generated executable is linked against required libraries to find the desired symbols.

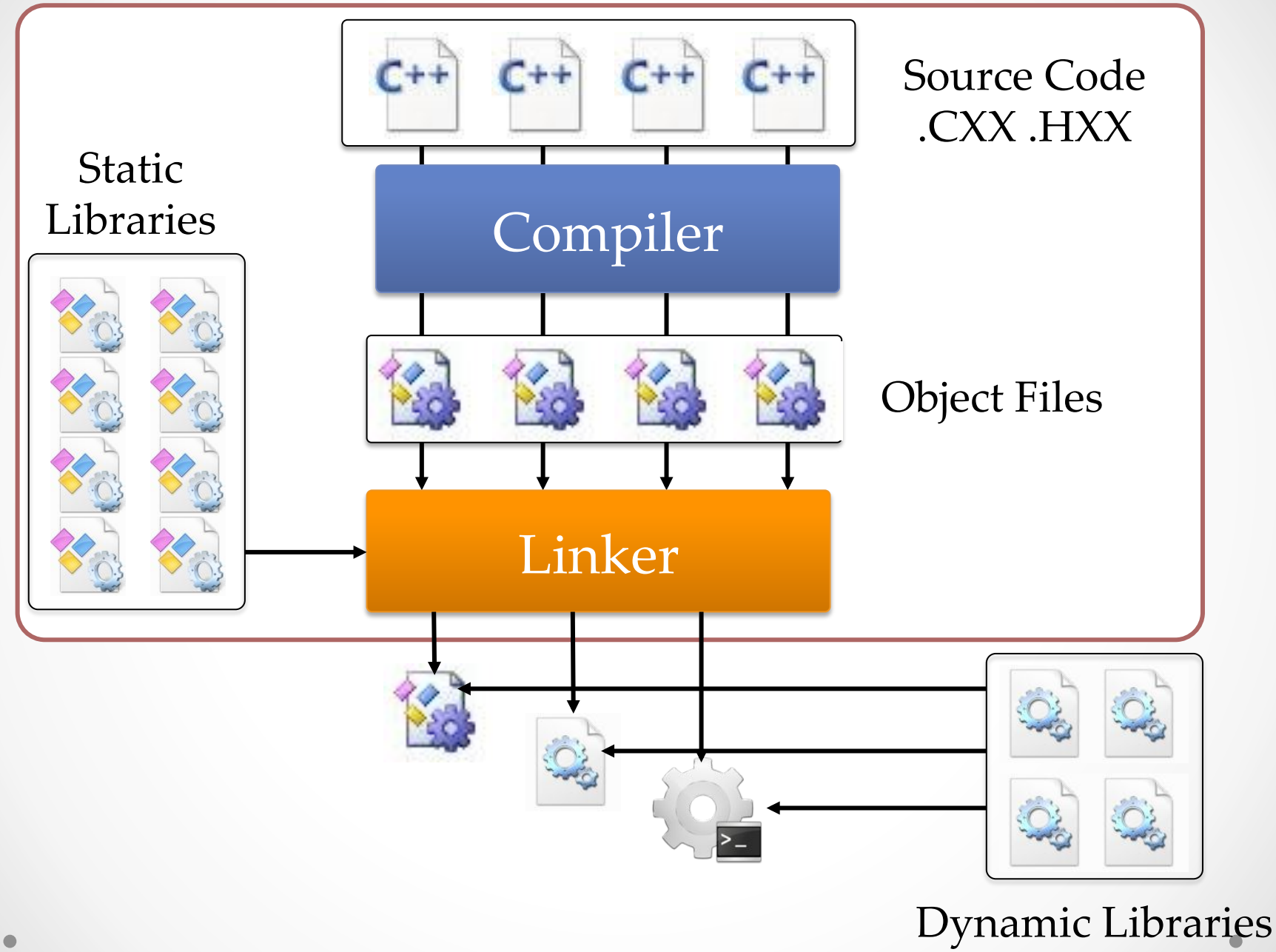
Software Build Systems

- For **simple programs**, the process consists of a single file being compiled.
- For **complex software**, the source code may consist of **many files** and may be combined in different ways to produce **many different versions and artifacts**.

Basic Build Process



A Bit-Detailed Overview


Source Code
.CXX .HXX
Static
Libraries
Compiler
Object Files
Linker
Dynamic Libraries

Build Utilities

- The process of building a program is usually managed by a **build tool.**
- Build Tool is a program that coordinates and controls other programs to compile and link the various files and do **other operations** in the correct order.
- It is **not only** concerned with compilation and linking.

Build Automation

- The act of scripting or automating build systems for:
  - Compiling source code into binary code.
  - Packaging binary code.
  - Running test units, regression tests, …
  - Deployment to production systems.
  - Creating documentation and/or release notes.

- Main Player in any **Software Cycle**

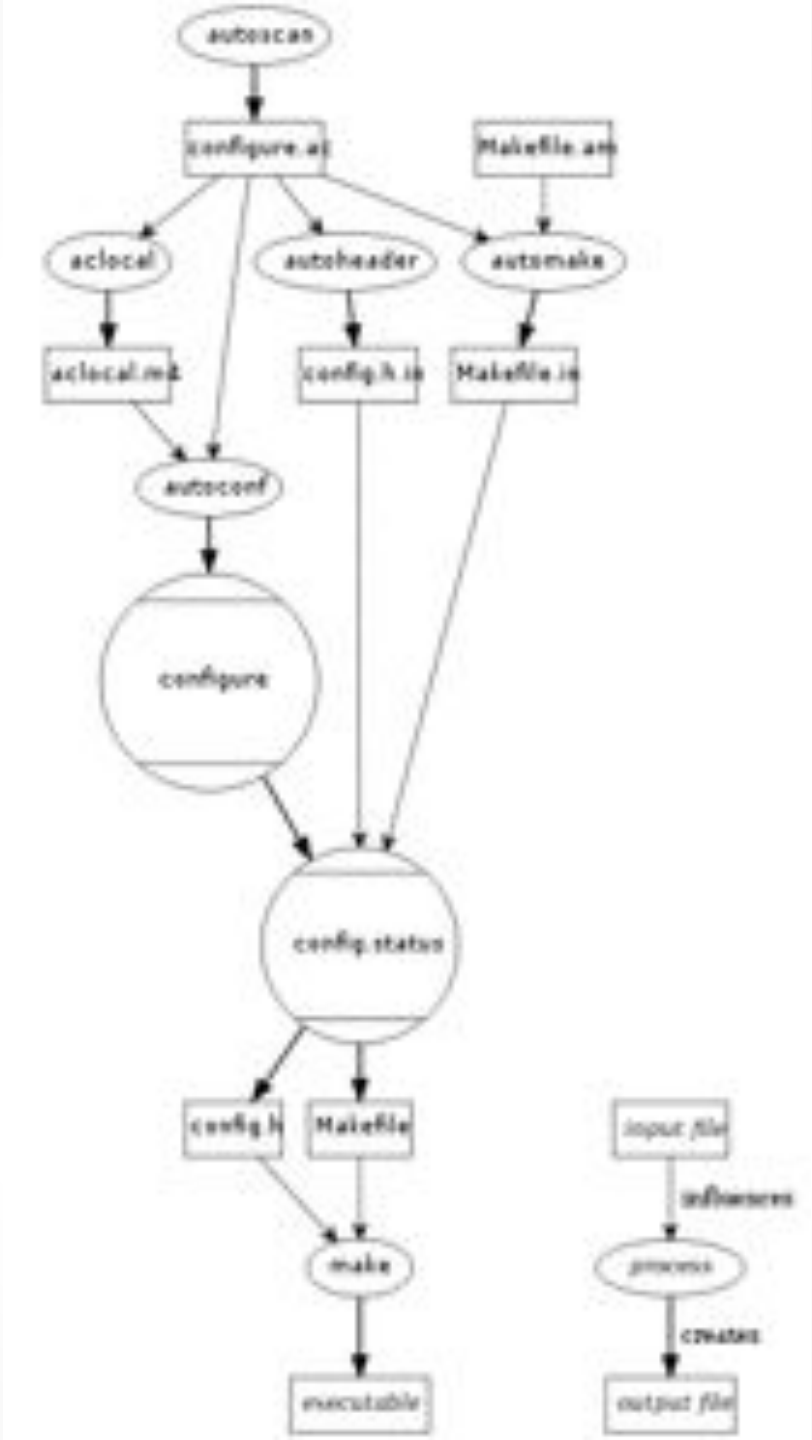
Popular Build Systems

- Make Based Systems
  - Generation of MakeFile.
- Non-Make Based Systems

GNU AutoTools

- AutoTools is part of the GNU toolchain and is widely used in many free-software and open-source packages.

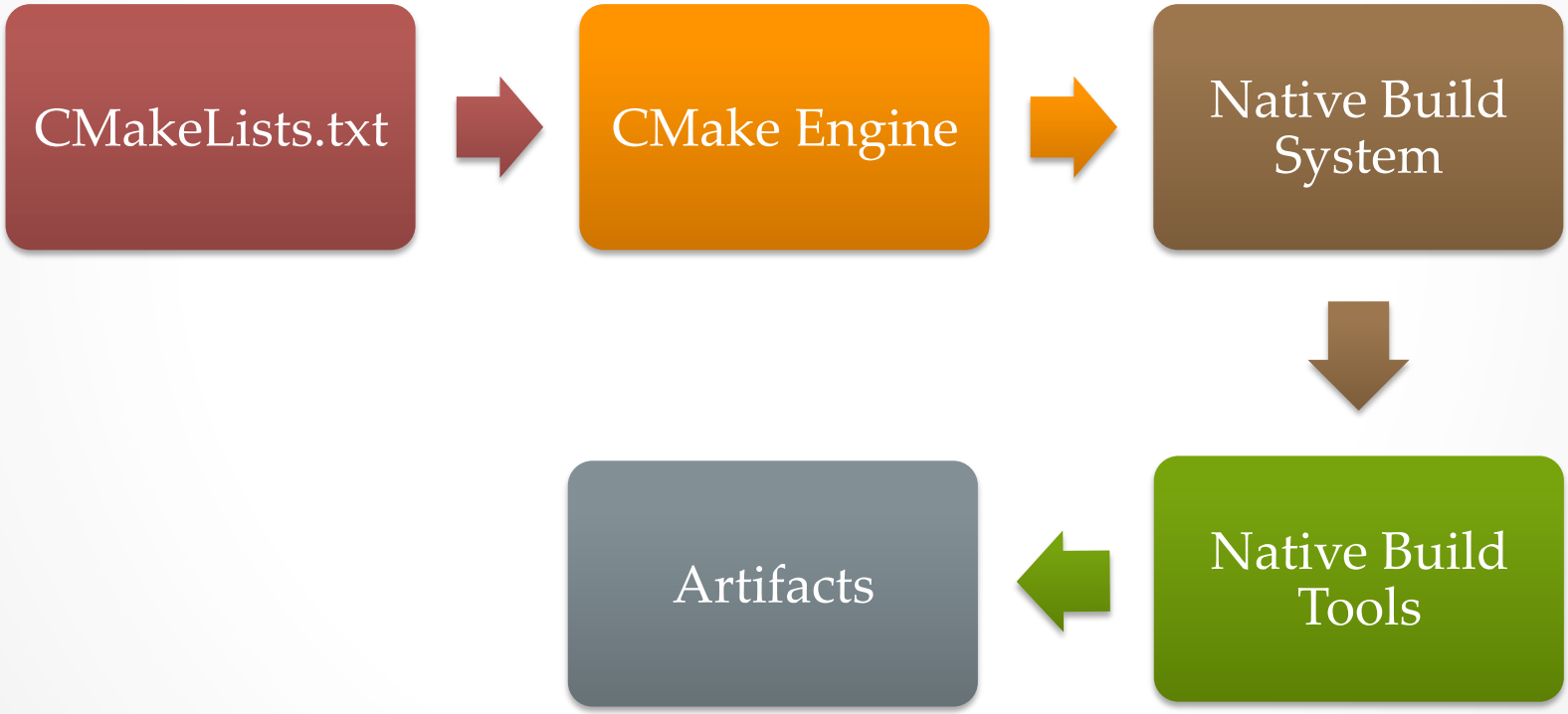
- In some environments called AutoCrap, or AutoHell


autoscan
configure.ac
Makefile.am
aclocal
autoheader
automake
aclocal.m4
config.h.in
Makefile.in
autoconf
configure
config.status
config.h
Makefile
make
executable
input file
process
output file

What is CMake !

- Tool for Generating Build Scripts for Native Build Environments
  - UNIX : MakeFile
  - Windows : MS VS Projects
  - MAC OSX : X-Code Build Scripts
- Features
  - It is an Open Source Tool.
  - Cross Platform
  - Managing Complex Software
  - Very Simple, Intuitive Syntax
  - Very Flexible, Extensible
  - Integrated Testing & Packaging (Ctest, CPack)
  - In-source and out-of-source building

CMake Build System



Basic Items

- CMakeLists.txt
  - Input **text files** that contain the **project parameters** and describe the **flow control** of the **build process** in simple CMake language.
- Modules
  - Special CMake file written for the purpose of finding a certain piece of software and to set it's libraries, include files and definitions into appropriate variables so that they can be used in the build process of another project.
  - (e.g. FindJava.cmake, FindZLIB.cmake, FindQt4.cmake)
- CMake Trees
  - Source Tree
  - Binary Tree

Basic Concepts

- Source Tree (Input)
  - CMakeLists.txt
  - Source Files (.CXX)
  - Header Files (.HXX)
- Binary Tree (Artifacts & Outputs)
  - Native Build System Files (Basically MakeFile)
  - Output Artifacts (Executables, Libraries, Packages, Docs, ..)
- Source and Binary trees may be
  - In the same directory **(in-source build)**
  - In different directories **(out-of-source build)**

Basic Concepts

- CMAKE_MODULE_PATH
  - Path to where the CMake modules are located
- CMAKE_INSTALL_PREFIX
  - Where to put files when calling 'make install'
- CMAKE_BUILD_TYPE
  - Type of build (Debug, Release, ...)
- BUILD_SHARED_LIBS
  - Switch between shared and static libraries

CMake Cache

- Created in the build tree (CMakeCache.txt)
- Contains Entries VAR:TYPE=VALUE
- Populated/Updated during configuration phase
- Speeds up build process
- GUI can be used to change values
- There should be no need to edit it manually!

Using CMake

- Create a Build directory ("out-of-source-build" concept)
  - >> mkdir build
  - >> cd build
- Configure the package for your system
  - >> cmake (or ccmake)
- Build the artifacts
  - >> make –j <Number of available cores>
- Install it:
  - make install

CMakeLists.txt

- INCLUDE_DIRECTORIES(“dir1” “dir2” ...)
- ADD_EXECUTABLE
- ADD_LIBRARY
- ADD_CUSTOM_TARGET
- ADD_DEPENDENCIES(target1 t2 t3) target1 depends on t2 and t3
- TARGET_LINK_LIBRARIES(target-name lib1 lib2 ...) Individual settings for each target

CMakeLists.txt

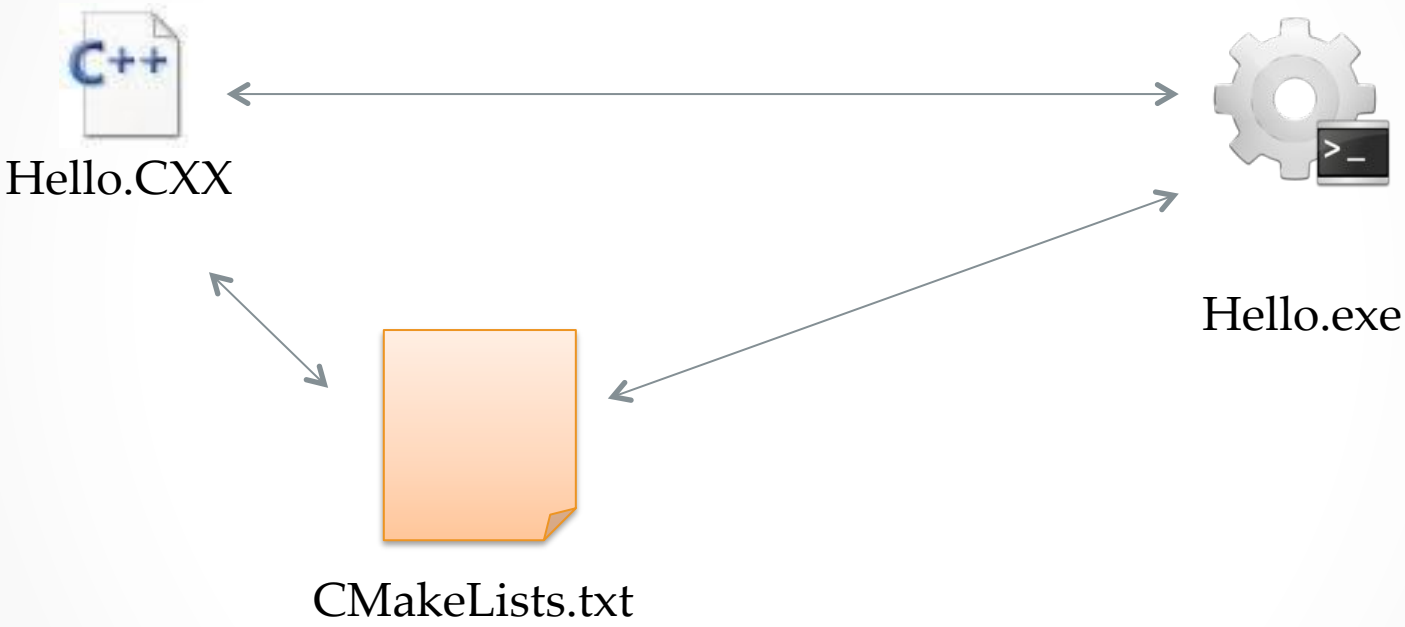
- LINK_LIBRARIES(lib1 lib2 ...) All targets link with the same set of libs
- SET_TARGET_PROPERTIES(...) lots of properties... OUTPUT_NAME, VERSION, ...
- MESSAGE(STATUS | FATAL_ERROR "message")
- INSTALL(FILES "f1" "f2" "f3" DESTINATION .)
- – DESTINATION relative to ${CMAKE_INSTALL_PREFIX}

CMakeLists.txt

- For more parameters and the CMake Language syntax,

  Check www.cmake.org -> Documentation

Let's Have an Example



Mastering CMake

CMake Tutorials

- Basic Kitware Tutorial
- Quick Introductory Tutorial
- A bit advanced CMake Tutorial
- More advanced Tutorial

Thanks For Paying Attention