



Cross-Platform Development with CMake

Julien Jomier

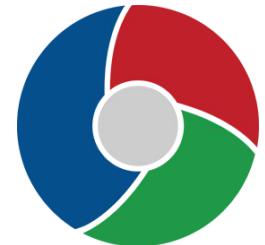
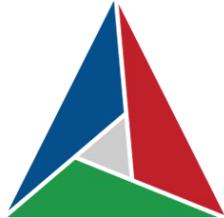


CODE RECKONS

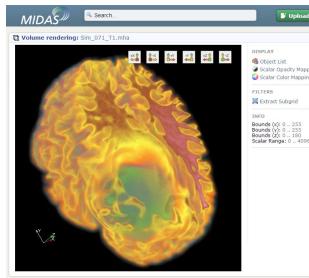
Science to the CORE

About me / Julien Jomier

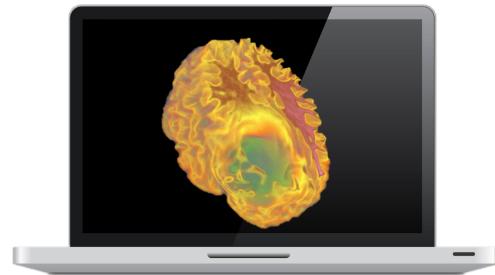
- Involved with CMake for 20+ years
- C++ developer
- CDash creator
- CMake educator
- Open Source enthusiast



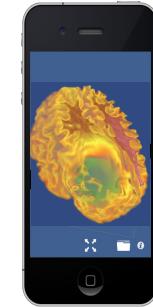
Kitware / Open Source Solutions



Web



Desktop



Mobile



Cloud /HPC



 **ParaView**

Universal Platforms

 **Resonant**

 **KWIVER**

 **3DSlicer**

 **Pulse
Physiology Engine**

 **imstk**

 **tomviz**



 **CMake**

 **Kitware**

Do I need a build system?

- In the beginning there was the command line:

```
% gcc hello.cxx -o hello
```

- Then shell script:

```
#!/bin/sh  
gcc hello.cxx -o hello  
% buildhello.sh
```

Do I need a build system?

- Then make files

hello: hello.cxx

 gcc hello.cxx

% make hello

- How to handle

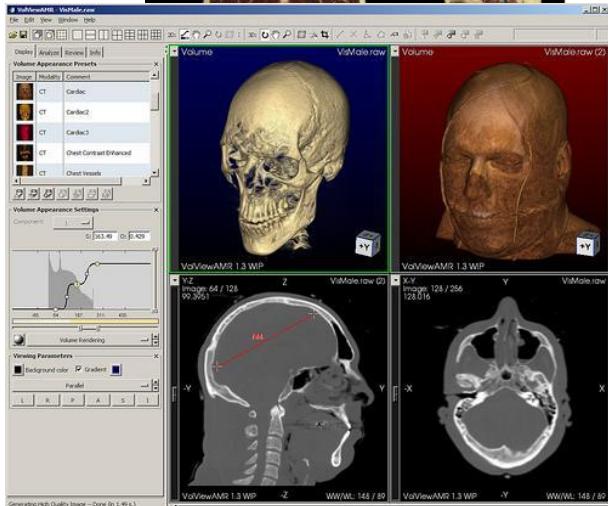
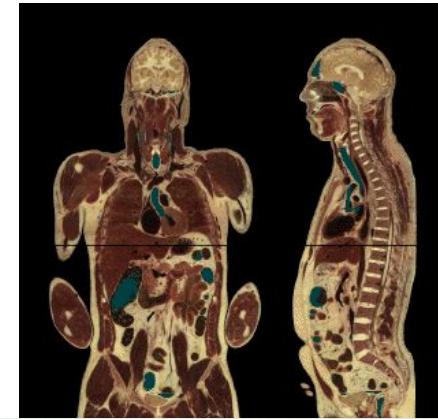
- Complex projects?
- Dependencies?
- Cross-Platform?

The screenshot shows a GitHub repository page for the VTK project. At the top, there's a header with 'Code', 'Pull requests 5', 'Actions', 'Projects 0', and 'Security'. Below that, it says 'Mirror of Visualization Toolkit repository' and provides a link to 'https://gitlab.kitware.com/vtk/vt'. It shows '72,795 commits', '7 branches', and '0 packages'. A pull request by 'martinken and krobot' titled 'Merge topic 'segy_2d'' is listed. The main content area shows a list of files with their corresponding clang-format reformat actions:

File	Action
vtkABI.h	clang-format: reformat using clang-format-8
vtkAOSDataArrayTemplate.h	Made more array APIs const
vtkAOSDataArrayTemplate.txx	clang-format: reformat using clang-format-8
vtkAbstractArray.cxx	Made more array APIs const
vtkAbstractArray.h	Made GetActualMemorySize const
vtkAndroidOutputWindow.cxx	clang-format: reformat using clang-format-8
vtkAndroidOutputWindow.h	clang-format: reformat using clang-format-8
vtkAnimationCue.cxx	clang-format: reformat using clang-format-8
vtkAnimationCue.h	clang-format: reformat using clang-format-8
vtkArray.cxx	clang-format: reformat using clang-format-8
vtkArray.h	clang-format: reformat using clang-format-8
vtkArrayCoordinates.cxx	clang-format: reformat using clang-format-8
vtkArrayCoordinates.h	clang-format: reformat using clang-format-8
vtkArrayDispatch.h	clang-format: reformat using clang-format-8
vtkArrayDispatch.txx	clang-format: reformat using clang-format-8
vtkArrayDispatchArrayList.in	cmake: clean up the array dispatch list creation
vtkArrayExtents.cxx	clang-format: reformat using clang-format-8
vtkArrayExtents.h	clang-format: reformat using clang-format-8
vtkArrayExtentsList.cxx	clang-format: reformat using clang-format-8
vtkArrayExtentsList.h	clang-format: reformat using clang-format-8

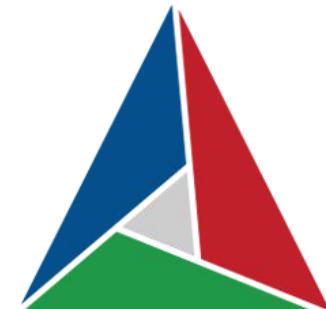
Where is CMake coming from?

- Insight Segmentation and Registration Toolkit (ITK)
<http://www.itk.org>
- Funded by National Library of Medicine (NLM)
As part of the Visible Human Project
 - Data: CT/MR/Slice 1994/1995
 - Code: (ITK) 1999
- CMake Release-1-0 branch created in 2001



What is CMake?

- **Cross-platform, open-source build system generator**
- Let you use the **native development tools** you love the most.
- It takes **plain text files** as input and **produces** project files or make files
- Family of Software Development Tools
 - Build = CMake
 - Test = CTest/CDash
 - Package = CPack



Changing the way we build C++

- Boost aims to give C++ a set of useful libraries like Java, Python, and C#
- CMake aims to give C++ compile portability like the compile once and run everywhere of Java, Python, and C#
 - Same build tool and files for all platforms
 - Easy to mix both large and small libraries



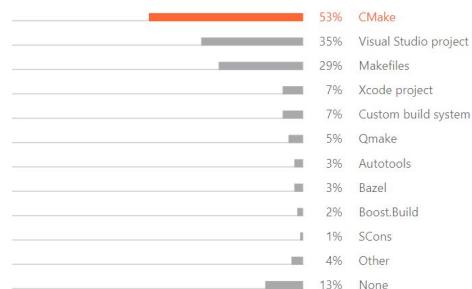
Build Tools Support

- **New IDE's and compilers**
 - Visual Studio releases supported weeks after beta comes out
 - Xcode releases supported weeks after beta comes out
 - ninja (command line build tool from Google) support contributed to CMake as ninja matured
 - Apple Silicon
- **New compiler support**
 - Clang
 - gcc versions

CMake Popularity

- Jetbrains IDE- CMake is the most popular build tool at 53%

Which project models or build systems do you regularly use, if any?



- Job openings requiring Cmake experience:
 - Indeed.com, **464 jobs**, at Tesla Motors, D2S Corp, Mindsource, Quanergy,
 - LinkedIn.com, **486 jobs**, at Samsung, Johnson Controls, Apple, Uber, Toyota, Microsoft ...

CMake is a community effort

Contributors 971



About 29,500 results (0.22 seconds)

www.youtube.com/watch

[More Modern CMake - Deniz Bahadir - Meeting C++ 2018 ...](https://www.youtube.com/watch?v=JyfjwzXWQHg)

 More Modern CMake (Reupload with slide recording provided by speaker, thanks Deniz!)Deniz ...
Feb 25, 2019 · Uploaded by Meeting Cpp
1:05:32

www.youtube.com/watch

[Oh No! More Modern CMake - Deniz Bahadir - Meeting C++ ...](https://www.youtube.com/watch?v=JyfjwzXWQHg)

 Oh No! More Modern CMake - Deniz Bahadir - Meeting C++ 2019His CMake Talk from last year: <https://www.youtube.com/watch?v=JyfjwzXWQHg>
Jan 2, 2020 · Uploaded by Meeting Cpp
1:00:46

www.reddit.com > [cpp](#) > [comments](#) > [azfel1](#) > [modern_cmake](#)

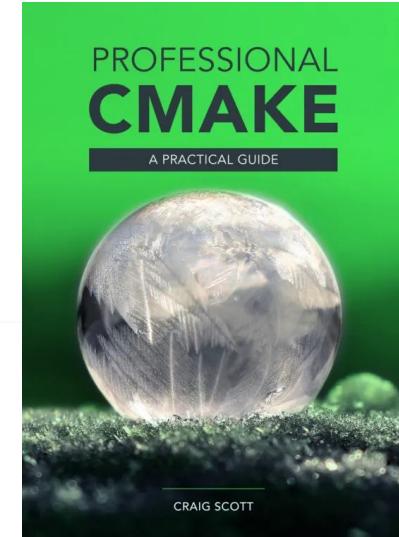
[Modern CMake Examples : cpp - Reddit](#)

 Modern CMake Examples ... IMHO the problem is CMake itself here, specifically ... you are already telling ...
Mar 10, 2019 · Uploaded by Meeting Cpp
49:52

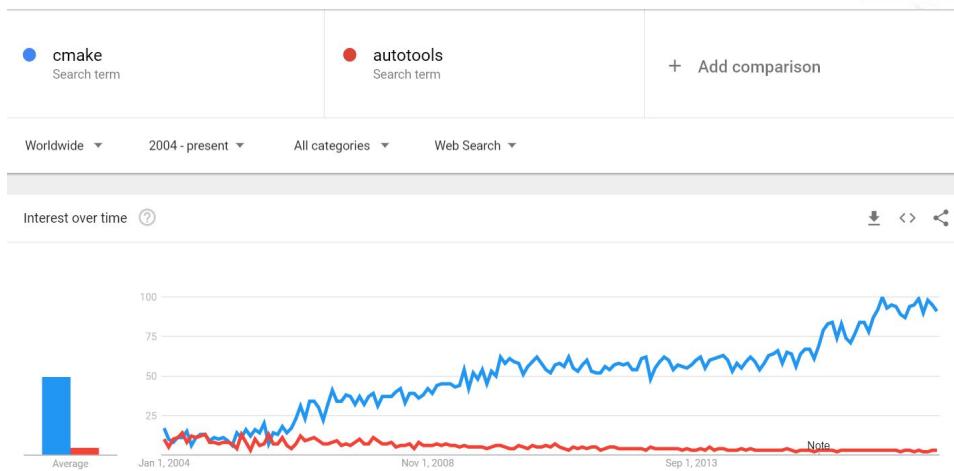
www.youtube.com/watch

[CppCon 2017: Mathieu Rupert "Using Modern CMake ..."](https://www.youtube.com/watch?v=JyfjwzXWQHg)

 ... Slides, PDFs, Source Code and other presenter materials are available at: <https://github.com/CppCon> ...
Oct 13, 2017 · Uploaded by CppCon
57:40

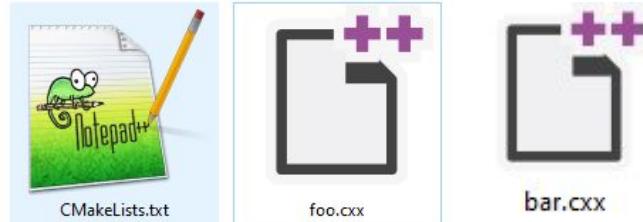


Growing Community



Writing CMake

Basic CMakeLists.txt



```
cmake_minimum_required(VERSION 3.15)
project(Example LANGUAGES CXX)

add_library(Bar STATIC bar.cxx)
add_executable(Foo foo.cxx)
target_link_libraries(Foo PRIVATE Bar)
```

Requiring CMake version

- First line of the top level CMakeLists.txt should always be `cmake_minimum_required`
- Allows projects to require a given version of CMake
- Allows CMake to be backwards compatible

project() command

- Necessary for the top-level CMake. Should be set after the **cmake_minimum_required command**
- VERSION: sets the PROJECT_VERSION_MAJOR/MINOR/TWEAK
- DESCRIPTION: sets the PROJECT_DESCRIPTION variable
- LANGUAGES:
 - C, CXX, FORTRAN, CSharp, CUDA, ASM
 - Default is C and CXX if not defined

Setting CMake Language Standards

- CMake offers a few different ways to specify which version of a language should be used.

```
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED TRUE)
```

```
[ 50%] Building CXX object main.cxx.o
/usr/bin/c++ -std=gnu++11 -o main.cxx.o -c main.cxx
```

- CMAKE_CXX_EXTENSIONS controls if compiler specific extensions are enabled

Flow Control - conditional

```
if(my_var)
    set(result ${my_var})
endif()
if(NOT my_var)
if(my_var AND my_var2)
if(my_var OR my_var2)
if(my_var MATCHES regexp)
if(TARGET target)
if(EXISTS file)
if(my_var LESS my_var2)
if(my_ver VERSION_EQUAL "2.0.2")
```

Flow Control : loops

```
foreach(F IN ITEMS a b c)
    message(${F})
endforeach()

set(items a b c)
foreach(F IN LISTS items)
    message(${F})
endforeach()
```

```
while(MY_VAR)
    message(${MY_VAR})
    set(MY_VAR FALSE)
endwhile()
```

Describe your source tree

- `add_subdirectory()`
- Variable values are inherited by CMakeLists.txt files in sub directories

Extending CMake

```
function(showcase_args myarg)
    message("myarg: ${myarg}")
    message("ARGV0: ${ARGV0}")
    message("contents of myarg: ${${myarg}}")
    message("extra arguments: ${ARGN}")
    message("# of arguments: ${ARGC}")
endfunction()

set(items a b c)
showcase_args(items)
```

CMake Commands

All commands

- `cmake --help-command-list`
- `cmake --help-command command_name`
- <https://cmake.org/cmake/help/latest/manual/cmake-commands.7.html>

CMake Features

- Automatic **dependency** generation
- Parallel builds
- Out of source builds
- System **introspection** (`try_compile`/`try_run`)
- Advanced RPATH handling
- Supports cross compilation

CMake Tips

- Treat CMake as code
- Targets are your friends
- Export your interface
- Upgrade your version of CMake when you can

Running CMake

Getting CMake

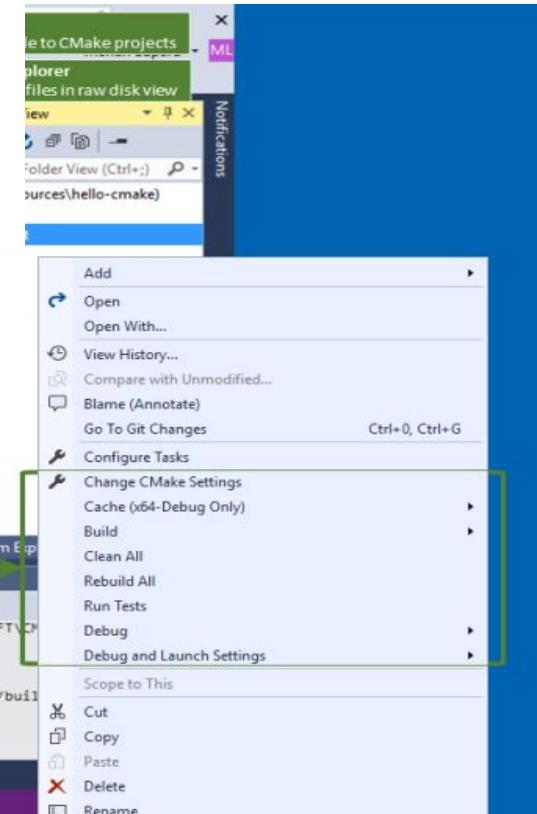
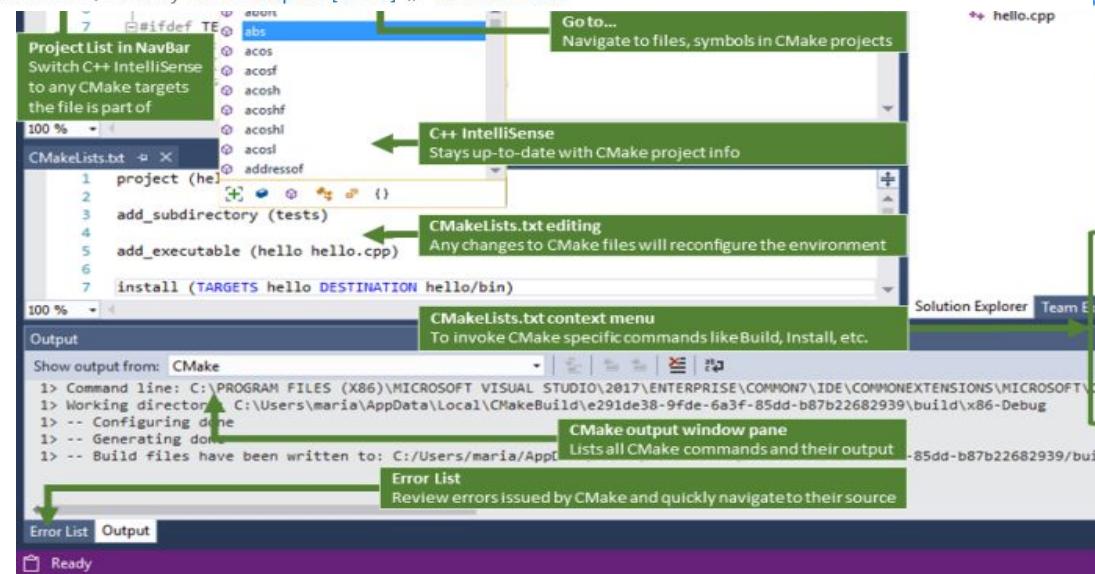
- www.cmake.org/download
 - From your Linux distribution
 - Visual Studio 2017+
-
- apt.kitware.com for Debian and Ubuntu
 - Snap universal linux package
 - pip install cmake
 - homebrew

Visual C++ Team Blog

C++ tutorials, C and C++ news, and information about the C++ IDE Visual Studio from the Microsoft C++ team.

CMake support in Visual Studio

October 5, 2016 by Marian Luparu [MSFT] // 56 Comments

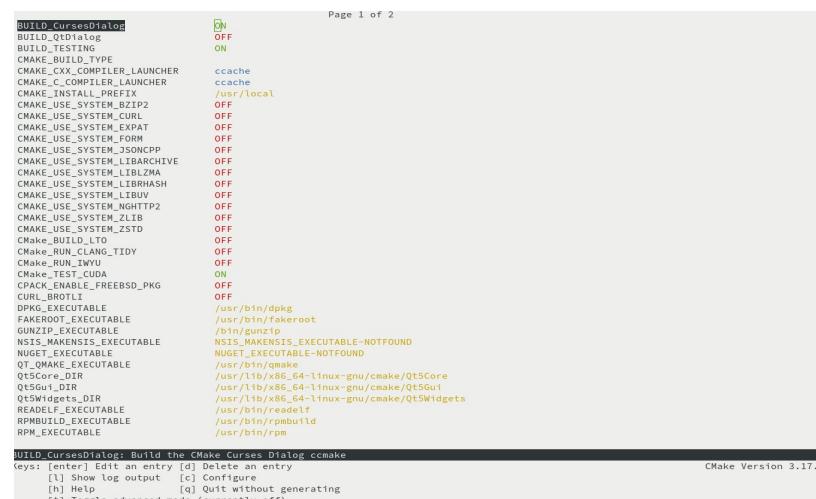
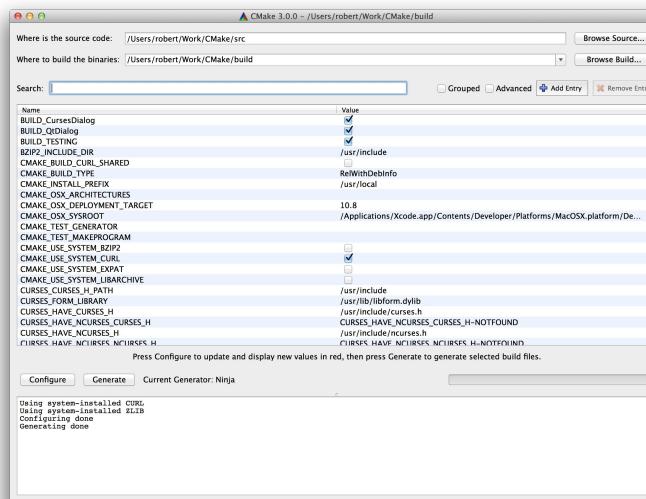


Running CMake

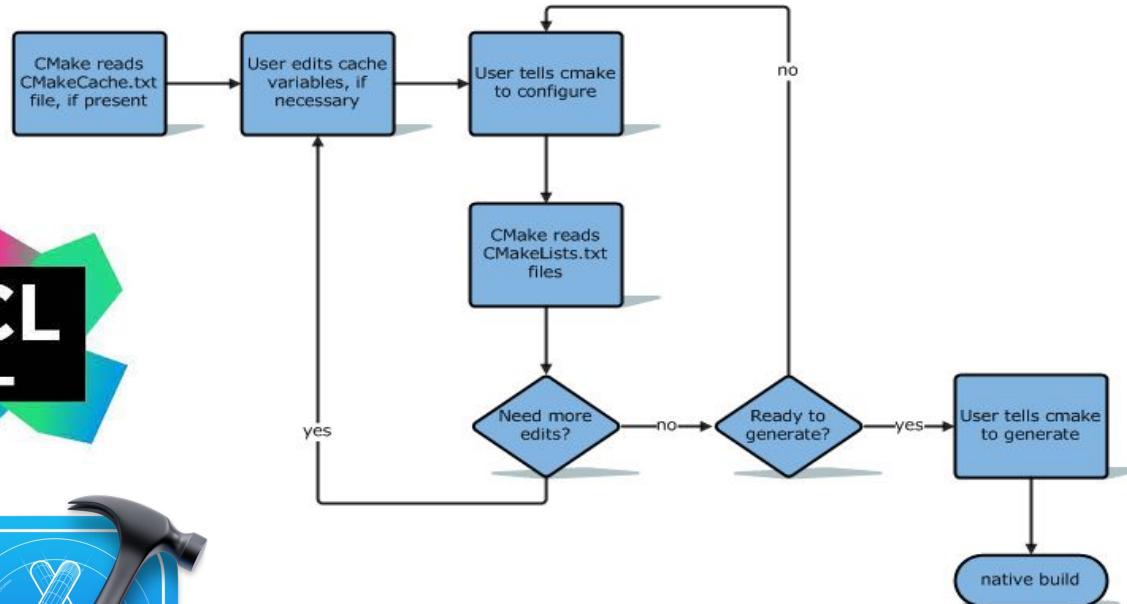
- Configure phase:
 - CMake parses its input files (`CMakeLists.txt`)
- Generate phase:
 - CMake writes out a build system for the specified generator (`Makefile`)

Running CMake

- cmake-gui (the Qt gui)
- ccmake (the terminal cli)
- cmake (non-interactive command line)



Running CMake



CMake Workflow

```
~/W/c/tutorial_build $ cmake ..../src/Tests/Tutorial/Complete/
```

[

Presets

- Specify the build directory, generator, cache variables, environment variables, and command line options for the cmake executable
- Done via CMakePresets.json
- Available since CMake 3.19

```
{
  "version": 1,
  "configurePresets": [
    {
      "name": "base",
      "hidden": true,
      "binaryDir": "${sourceDir}/build/${presetName}",
      "environment": {
        "CXX": "clang++-9"
      }
    },
    {
      "name": "clang9",
      "inherits": "base",
      "generator": "Ninja",
      "environment": {
        "CXX": "clang++-9"
      }
    },
    {
      "name": "gcc5",
      "inherits": "base",
      "generator": "Unix Makefiles",
      "environment": {
        "CXX": "g++-5"
      }
    }
  ]
}
```

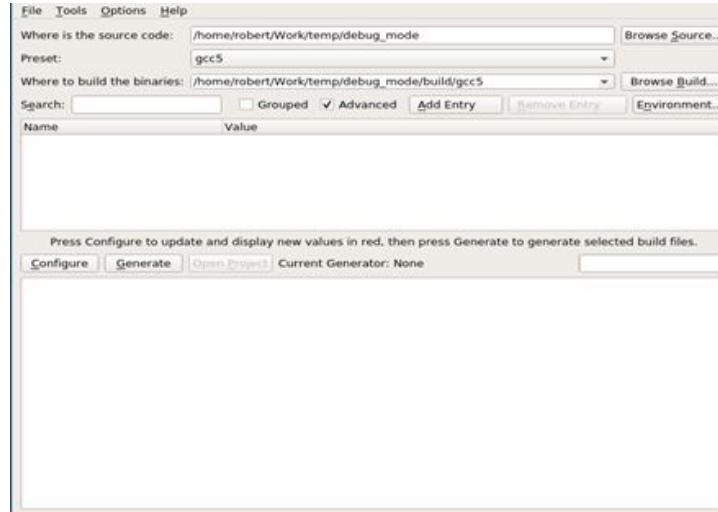
Presets

```
~/W/t/debug_mode $ ~/Work/cmake/build/bin/cmake -S . --list-presets
```

```
Available presets:
```

```
"clang9"
```

```
"gcc5"
```



Presets

```
~/W/t/debug_mode $ ~/Work/cmake/build/bin/cmake -S . --preset=gcc5
Preset environment variables:

CXX="g++-5"

-- The CXX compiler identification is GNU 5.5.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/g++-5 - skipped
```

```
~/W/t/debug_mode $ ~/Work/cmake/build/bin/cmake -S . --preset=clang9
Preset environment variables:

CXX="clang++-9"

-- The CXX compiler identification is Clang 9.0.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/clang++-9 - skipped
```

Dealing with Dependencies

CMake Find Modules

Our library “foo” needs PNG

```
find_package(PNG REQUIRED)
add_library(foo SHARED foo.cxx)
```

No different than if we built PNG as part of the project:

```
target_link_libraries(foo PRIVATE PNG::PNG)
```

FindPackage()

- Should work in all the following cases:
 - a. The package is **build from source with CMake**
 - b. The package is **installed**
 - c. The package is **not build with CMake**

CMake Find Modules

- CMake provides over **150 find modules**
 - `cmake --help-module-list`
 - <https://cmake.org/cmake/help/latest/manual/cmake-modules.7.html>

How do we maintain these modules?

Exporting Targets

- Install rules can generate imported targets

```
add_library(parasite STATIC eat_leaf.cxx)
install(TARGETS parasite root trunk leaf
        EXPORT tree-targets)
install(EXPORT tree-targets
        DESTINATION lib/cmake/tree)
```

- Installs library and target import rules
 - <prefix>/lib/libparasite.a
 - <prefix>/lib/cmake/tree/tree-targets.cmake

External Projects

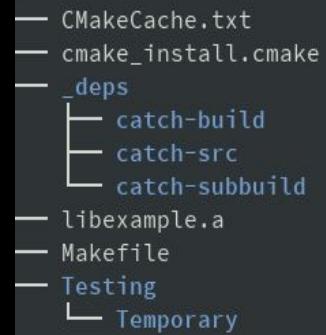
- Building/Importing targets **at build time**

```
ExternalProject_Add(foo
    GIT_REPOSITORY      git@github.com:FooCo/Foo.git
    GIT_TAG            origin/release/1.2.3
)
ExternalProject_Add(bar
    GIT_REPOSITORY      git@github.com:BarCo/Bar.git
    GIT_TAG            origin/release/2.3.4
    DEPENDS foo
)
```

FetchContent

- Building/Importing targets **at configure time**

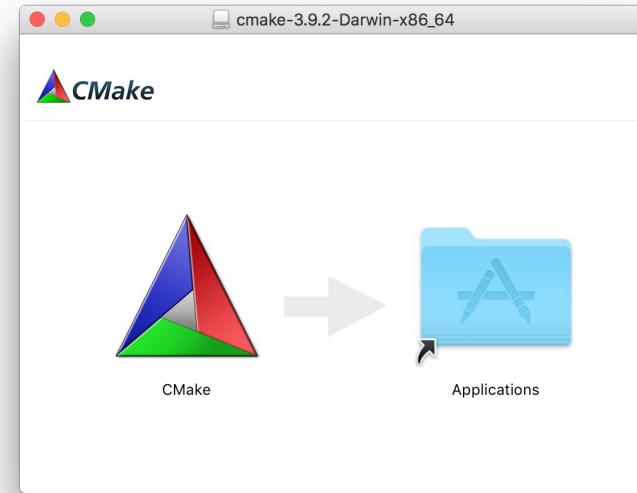
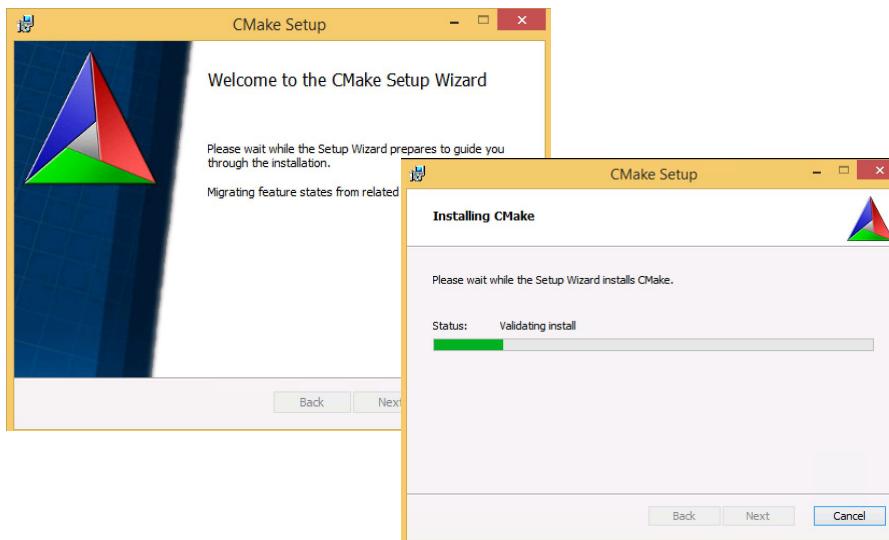
```
FetchContent_Declare(catch
    GIT_REPOSITORY https://github.com/catchorg/Catch2.git
    GIT_TAG        v2.2.1
)
FetchContent_GetProperties(catch)
if(NOT catch_POPULATED)
    FetchContent_Populate(catch)
    add_subdirectory(${catch_SOURCE_DIR} ${catch_BINARY_DIR})
endif()
```



Packaging

CPack

- CPack is bundled with CMake
- Creates professional platform specific installers



Using CPack

- On Windows install command line ZIP program, NSIS and WiX
- Setup your project to work with cpack
 - Get `make install` to work
 - `install(...)`
 - make sure your executables work with relative paths and can work from any directory
 - Set cpack option variables if needed
 - `include(CPack)`

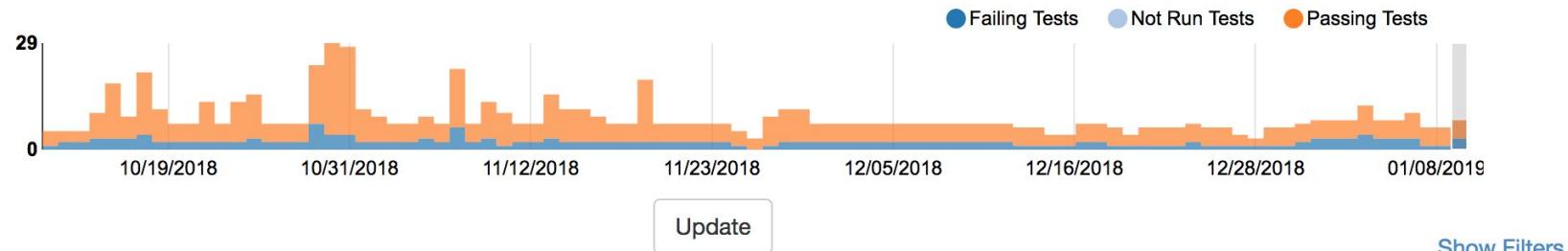
Continuous Integration

CTest/CDash

- CTest : Testing driver
- CDash: Web-based dashboard for collecting test results

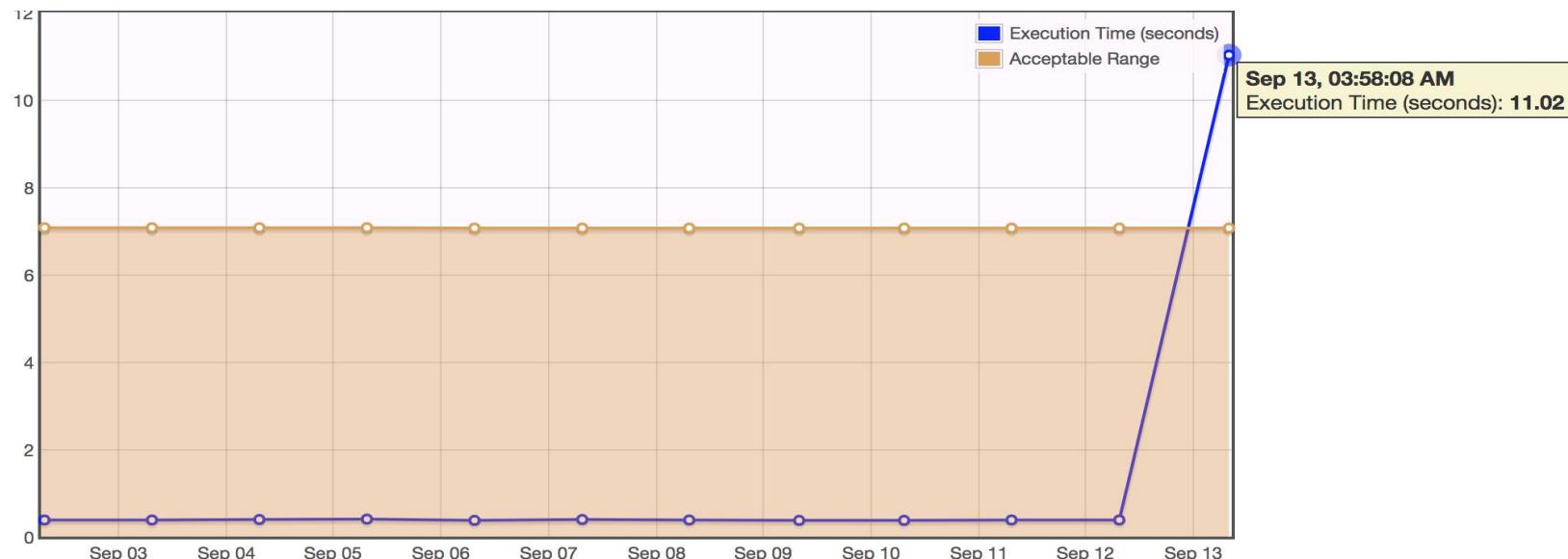
Scan Build		1 build						
Site	Build Name	Update		Configure		Build		Start Time ▾
		Revision	Error	Warn	Error	Warn		
elysim-linux.kitware	Linux-clang-scanbuild	8c3071	0	0	0	0	10 hours ago	
Nightly Expected							96 of 108 builds	
Site	Build Name	Update		Configure		Build		Start Time ▾
		Revision	Error	Warn	Error	Warn	Not Run	
trnsic.kitware	vs14-64-ninja	8c3071	0	0	0	0	0	2 ⁺² 479 ₋₂ 13 hours ago
trnsic.kitware	mingw64-make	8c3071	0	0	0	0	0	1 ⁺¹ 449 ₋₁ 6 hours ago
hythloth.kitware	Linux-ninja-gcov	8c3071	0	0	0	0	0	1 519 7 hours ago
trnsic.kitware	vs14-64-ide-v90	8c3071	0	0	0	0	0	1 ⁺¹ 387 ₋₁ 8 hours ago
vesper.kitware	watcom	8c3071	0	0	0	0	0	1 ⁺¹ 368 ₋₁ 8 hours ago
trnsic.kitware	vs9-64-ide	8c3071	0	0	0	0	0	1 ⁺¹ 381 ₋₁ 9 hours ago
vesper.kitware	bcc55	8c3071	0	0	0	0	0	1 ⁺¹ 370 ₋₁ 9 hours ago
dash3win7.kitware	vs14-64	8c3071	0	0	0	0	0	1 ⁺¹ 460 ₋₁ 9 hours ago
dashsun1.kitware.com	Solaris-10-8.11_Oracle-12.3	8c3071	0	0	0	0	0	415 1 hour ago

Track test health over time



Test Name ▾		Failure ▾	Timeout	Total Runs	Time
viewBuildError		100%	0%	2	21s 570ms
upgrade		100%	0%	2	1m 33s 790ms

Track test timing



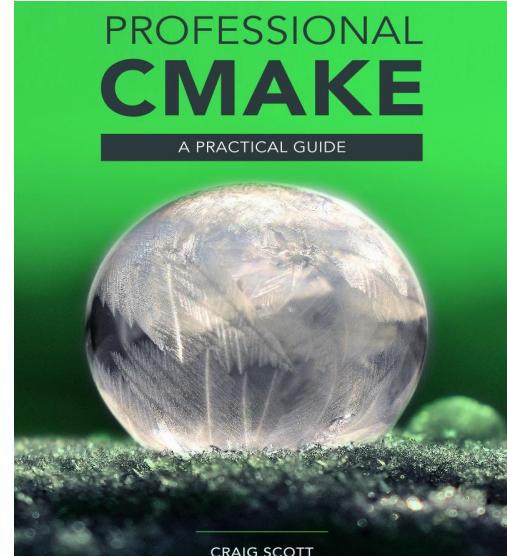
Test output

```
WaitForSingleObject returned unexpected status 0x102
In function testConsole, line 718: WaitForSingleObject#2 failed!
Failed with error: 0x2!
Error message: The system cannot find the file specified.
```

Learn more about CMake

Learning CMake

- ~~Copy some CMake code from 2010...~~
- Watch a talk on “Modern CMake”
- Professional CMake by Craig Scott
- Discourse Forum
 - *discourse.cmake.org*
- Documentation
 - www.cmake.org/cmake/help/latest/
- Tutorial
 - cmake.org/cmake/help/latest/guide/tutorial/index.html



CMake Technical Guides



cmake.org/cmake/help/latest/index.html

- [cmake-toolchains\(7\)](#)
- [cmake-variables\(7\)](#)
- [cpack-generators\(7\)](#)

Guides

- [CMake Tutorial](#)
- [User Interaction Guide](#)
- [Using Dependencies Guide](#)
- [Importing and Exporting Guide](#)
- [IDE Integration Guide](#)

CMake Testing Doc Code

Specify the C++ Standard

Next let's add some C++11 features to our project by replacing `atof` with `std::stod` in `tutorial.cxx`. At the same time, remove `#include <cstdlib>`.

```
.. literalinclude:: Step2/tutorial.cxx
:language: c++
:start-after: // convert input to double
:end-before: // calculate square root
```

We will need to explicitly state in the CMake code that it should use the correct flags. The easiest way to enable support for a specific C++ standard in CMake is by using the `:variable:`CMAKE_CXX_STANDARD`` variable. Set the `:variable:`CMAKE_CXX_STANDARD`` variable in the `CMakeLists.txt` file to 11 and `:variable:`CMAKE_CXX_STANDARD_REQUIRED`` to True. Make sure to add the ``CMAKE_CXX_STANDARD`` declarations above the call to `add_executable`.

```
.. literalinclude:: Step2/CMakeLists.txt
:language: cmake
:end-before: # configure a header file to pa
```

```
// A simple program that computes the square root of a number
#include <cmath>
#include <iostream>
#include <string>

#include "TutorialConfig.h"

int main(int argc, char* argv[])
{
    if (argc < 2) {
        // report version
        std::cout << argv[0] << " Version " << Tutorial_VERSION_MAJOR << "."
            << Tutorial_VERSION_MINOR << std::endl;
        std::cout << "Usage: " << argv[0] << " number" << std::endl;
        return 1;
    }

    // convert input to double
    const double inputValue = std::stod(argv[1]);

    // calculate square root
    const double outputValue = sqrt(inputValue);
    std::cout << "The square root of " << inputValue << " is " << outputValue
}
```

Specify the C++ Standard

Next let's add some C++11 features to our project by replacing `atof` with `std::stod` in `tutorial.cxx`. At the same time, remove `#include <cstdlib>`.

```
const double inputValue = std::stod(argv[1]);
```

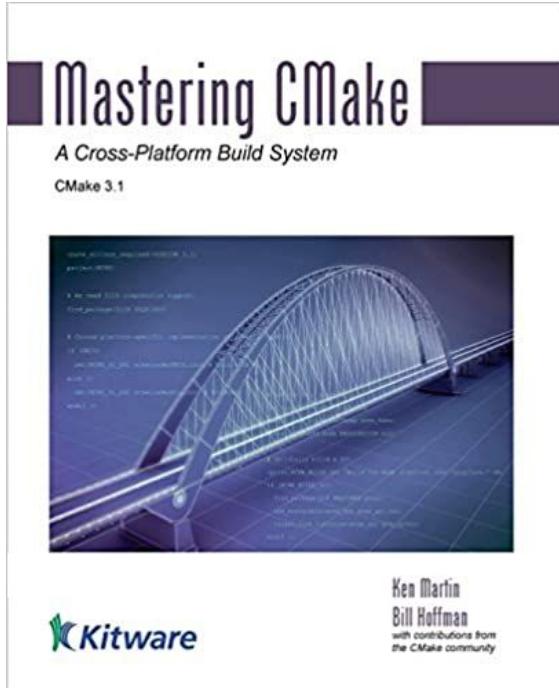
We will need to explicitly state in the CMake code that it should use the correct flags. The easiest way to enable support for a specific C++ standard in CMake is by using the `CMAKE_CXX_STANDARD` variable. For this tutorial, set the `CMAKE_CXX_STANDARD` variable in the `CMakeLists.txt` file to 11 and `CMAKE_CXX_STANDARD_REQUIRED` to True. Make sure to add the `CMAKE_CXX_STANDARD` declarations above the call to `add_executable`.

```
cmake_minimum_required(VERSION 3.10)

# set the project name and version
project(Tutorial VERSION 1.0)

# specify the C++ standard
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED True)
```

Mastering CMake is now open



gitlab.kitware.com/cmake/mastering-cmake

GitLab Projects Groups More Search or jump to... ▾ 11 ↗

M Mastering CMake

Project overview Details Activity Releases Repository Labels Merge requests Members

CMake > Mastering CMake

Mastering CMake Project ID: 7164 Leave project

5 Commits 1 Branch 0 Tags 3.3 MB Files 3.3 MB Storage

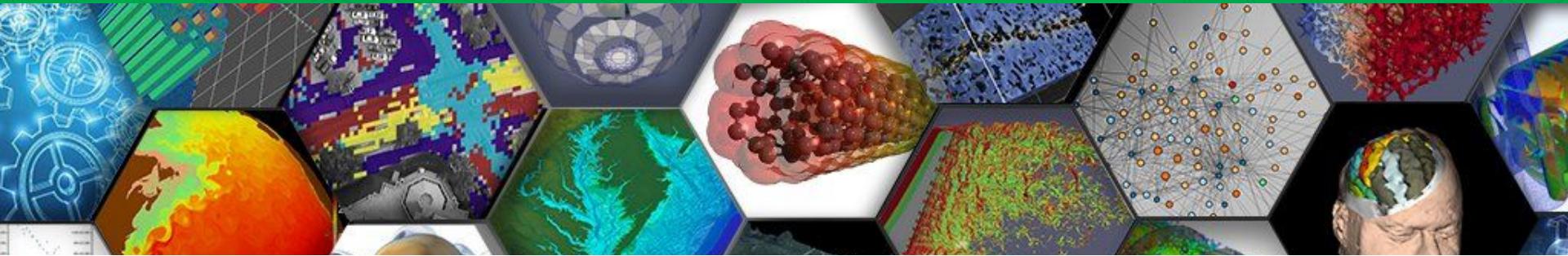
Mastering CMake Book

master mastering-cmake / + History Find file Web IDE Clone

Merge branch 'add-license' into 'master' Brad King authored 2 days ago 4f935526

Other

Name	Last commit	Last update
Book/Cover	Import Mastering CMake Source	2 days ago
Help	Import Mastering CMake Source	2 days ago
Utilities	Import Mastering CMake Source	2 days ago
.gitattributes	Import Mastering CMake Source	2 days ago
.gitignore	Import Mastering CMake Source	2 days ago
CMakeLists.txt	Import Mastering CMake Source	2 days ago
License.txt	Replace Copyright with CC BY 4.0 License	2 days ago



Thank You! Merci!

Julien Jomier
julien.jomier@kitware.com