Modern CMake

Open source tools to build, test and package software: CMake, CTest, CPack, CDash

Kitware

Bill Hoffman

- CTO and a founder of Kitware Inc
- Originator of CMake build tool
- Barefoot/Sandals Ultra distance runner

Google Tech Talk 2009

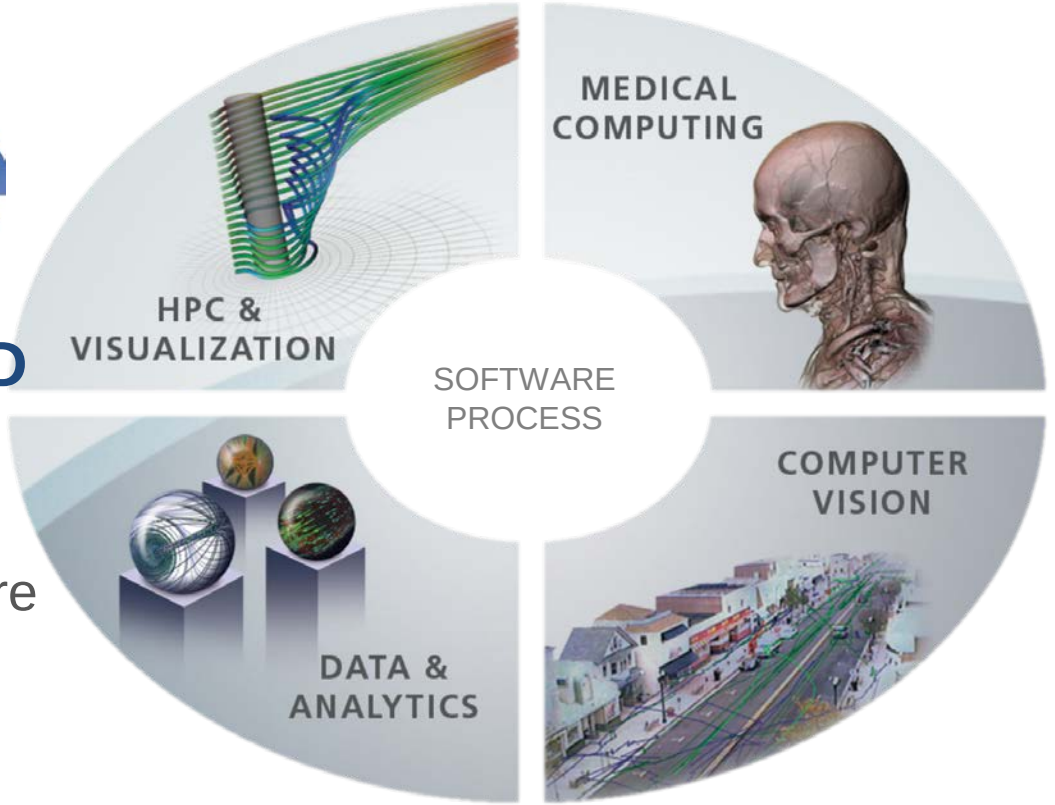
Leadville CO 2018

Kitware

Kitware

Collaborative software R&D

- Technical computing
- Algorithms & applications
- Software process & infrastructure
- Support & training
- Open source leadership



Supporting all sectors

- Industry, government & academia

Kitware

Kitware's customers & collaborators

Over 75 **academic** institutions…

Harvard
Massachusetts Institute of Technology
University of California, Berkeley
Stanford University
California Institute of Technology
Imperial College London
Johns Hopkins University
Cornell University
Columbia University
Robarts Research Institute
University of Pennsylvania
Rensselaer Polytechnic Institute
University of Utah
University of North Carolina

Over 50 **government** agencies and labs…

National Institutes of Health (NIH)
National Science Foundation (NSF)
National Library of Medicine (NLM)
Department of Defense (DOD)
Department of Energy (DOE)
Defense Advanced Research Projects Agency (DARPA)
Army Research Lab (ARL)
Air Force Research Lab (AFRL)
Sandia (SNL)
Los Alamos National Labs (LANL)
Argonne (ANL)
Oak Ridge (ORNL)
Lawrence Livermore (LLNL)

Over 100 **commercial** companies…

Automotive
Aircraft
Defense
Energy technology
Environmental sciences
Finance
Industrial inspection
Oil & gas
Pharmaceuticals
Publishing
3D Mapping
Medical devices
Security
Simulation

Kitware

Open source platforms

VTK & ParaView interactive visualization and analysis for scientific data

ITK & 3D Slicer medical image analysis and personalized medicine research

CMake cross-platform build system

- CDash, CTest, CPack, software process tools

Resonant informatics and infovis

KWIVER computer vision image and video analysis

- Other areas include: Simulation, ultrasound, physiology, information security, materials science, …

What is CMake?

- CMake is the **cross-platform**, **open-source build system** that lets you use the **native development tools** you love the most.
- It's a build system **generator**

Ninja

- It takes **plain text files** as input that describe your project and **produces** project files or make files for use with a wide variety of **native development tools.**

- Family of Software Development Tools
  - Build = CMake
  - Test = CTest/CDash
  - Package = CPack

Kitware

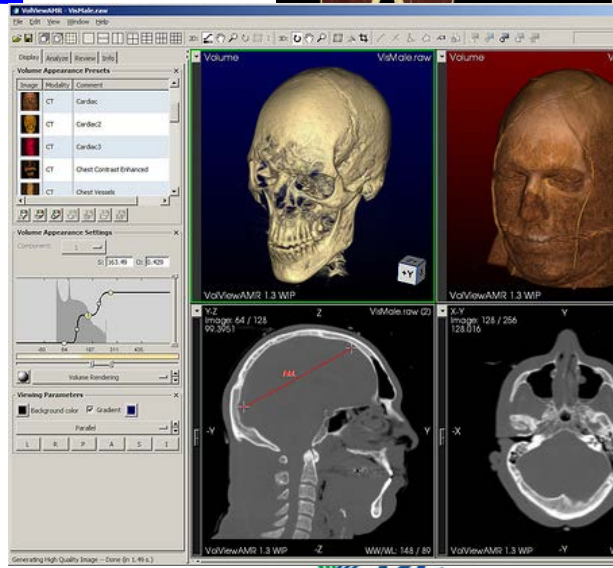
Modern CMake

- CMake is code, treat CMakeLists.txt like the rest of the code, comments
- CMake Targets are objects with public and private propeties
- Import third party libraries as imported targets
- Export your libraries so they can be used by other CMake projects

Kitware

CMake: History

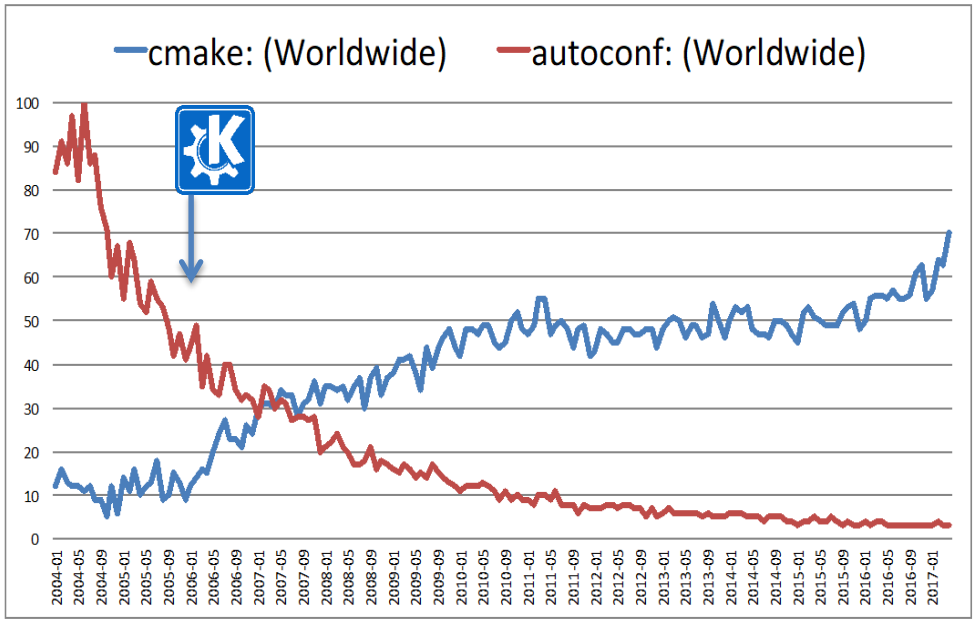
- Built for the Insight Segmentation and Registration Toolkit (ITK) http://www.itk.org
- Funded by National Library of Medicine (NLM): part of the Visible Human Project
  - Data CT/MR/Slice 1994/1995
  - Code (ITK) 1999
    - Cmake Release-1-0 branch created in 2001



Kitware

CMake has broad usage in the C++ world

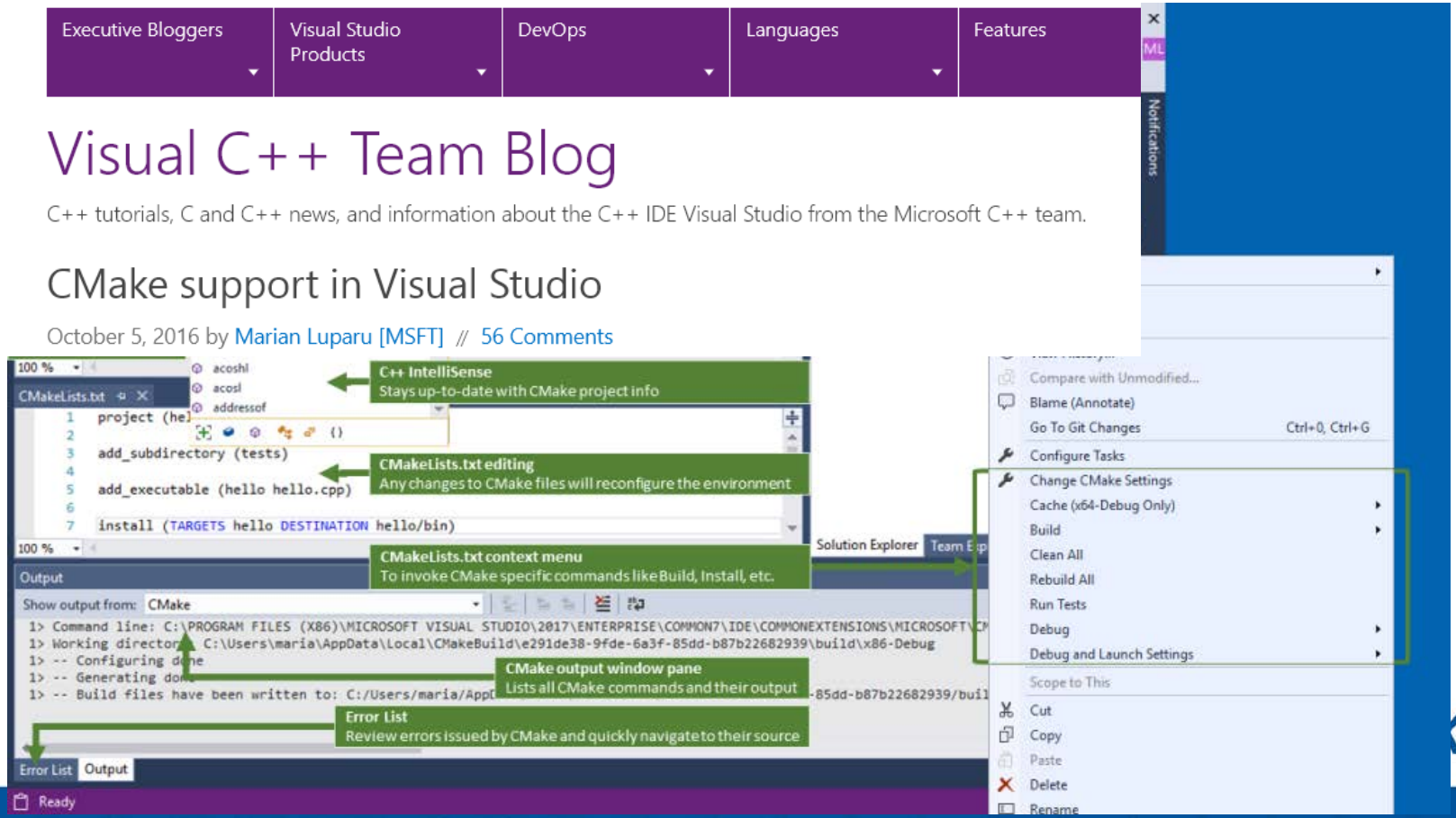
KDE 2006 - Tipping Point!



- 7000+ downloads per day from www.cmake.org

Indeed.com CMake jobs Full-time(263)

Adopted by Microsoft



Kitware

CMake: Features

- Automatic **dependency** generation (C, C++, CUDA, Fortran)
  - build a target in some directory, and everything this target depends on will be up to date
  - If a header file changes the correct files will be built.

Kitware

Fortran Module Order

Yes, it can get confusing. I am not aware of any references, others might be. The Intel Fortran Users guide discusses using modules and states the requirement rather succinctly as:

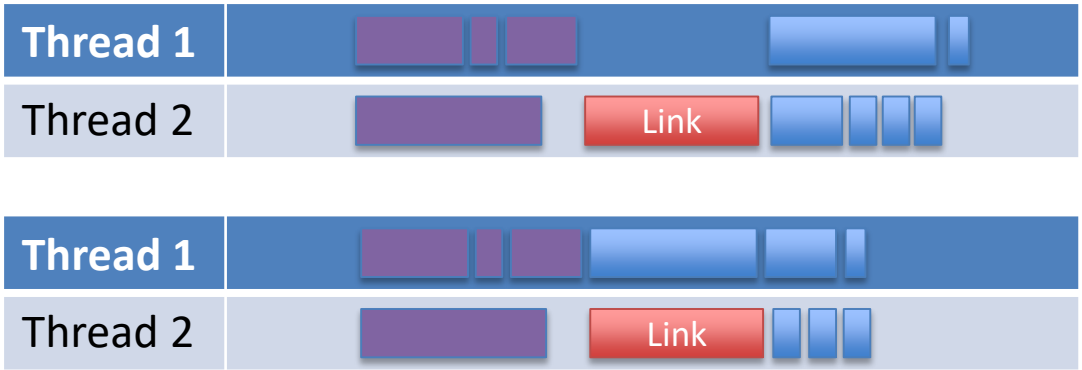
You need to make sure that the module files are created before they are referenced by another program or subprogram.

- Old way: make;make;make until it works
- CMake way: cmake; make or cmake; ninja
  - CMake will automatically order Fortran files based on use statements in the code for a library

Kitware

Ninja

- Improved parallelism for ninja builds in CMake 3.9



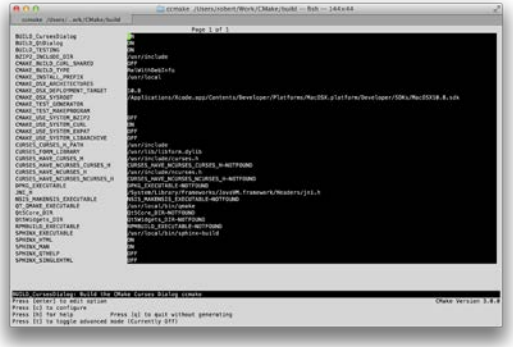
- Can control pools to limit concurrent links

Kitware

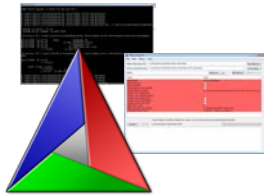
Random list of things CMake does well

- Excellent install commands
- Excellent packaging tools
- Ability to find/link system libraries
- Handles shared libraries and versioning across platforms (linux, mac, windows)
- Keeps up to date with current and obscure compilers
- Cross platform development support (Linux/Mac/Windows/android/HPC)
- Integration of static/dynamic analysis tools
- Integration of code coverage tools
- Excellent backwards compatibility with itself (policy system)
- Open and dynamic community accepting of changes small and large
- Supports many workflows and IDEs

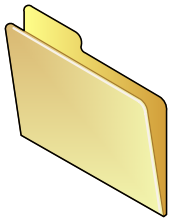
CMake Workflow



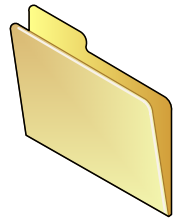
cmake –GNinja



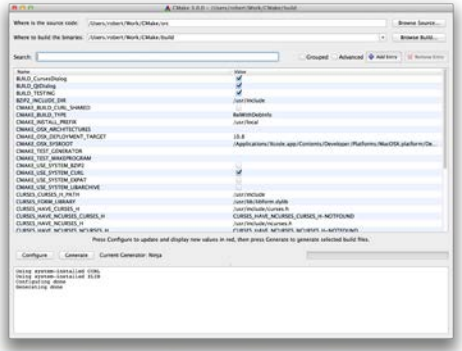
2. Run cmake-gui (or cmake or ccmake) to configure and generate native build system files



1. Edit files in the source tree



build tree



3. Open project files from the build tree and use the native build tools

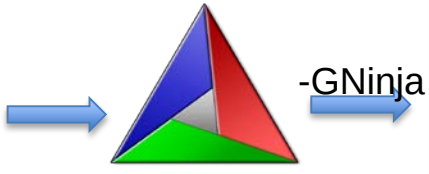
Kitware

Out of source builds

Project Source Tree
Library1 (CMakeLists.txt foo.cxx bar.cxx)
Library2 (CMakeLists.txt car.cxx car.h fun.F90)
Library3 (CMakeLists.txt gpu.cu ml.cxx)
App1 (CMakeLists.txt exe.cxx)
App2 (CMakeLists.txt exegui.cxx)

-GNinja

GCC Build Tree
CMakeCache.txt – stores info specific to this build
build.ninja



-GNinja

Clang Build Tree
CMakeCache.txt – stores info specific to this build
build.ninja

-Gunix Makefiles

GCC Build Tree
CMakeCache.txt – stores info specific to this build
Makefile

Kitware

Modern CMake

CMake Then and Now

CMake 2001

CMakeLists.txt

```
SUBDIRS = \
Code/Common \

ME = ITK
```

Code/Common/CMakeLists.txt

```
ME = ITKCommon

COMPILE_CLASSES =\
itkDataObject \
itkDirectory

WIN32_CLASSES =\
itkWin32OutputWindow
```

CMake 2008

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(ITK)
add_subdirectory(Code/Common)
```

Code/Common/CMakeLists.txt

```
set(ITKCommonSources
itkDataObject.cxx itkDirectory.cxx)
if(WIN32)
  set(ITKCommonSources
${ITKCommonSources}
itkWin32OutputWindow.cxx)
endif()
add_library(ITKCommon
${ITKCommonSources})
```

CMake 2018

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(ITK)
add_subdirectory(Code/Common)
```

Code/Common/CMakeLists.txt

```
add_library(ITKCommon)
target_sources(ITKCommon PRIVATE
itkDataObject.cxx itkDirectory.cxx ...)
if(WIN32)
  target_sources(ITKCommon PRIVATE
itkWin32OutputWindow.cxx)
endif()
```

Kitware

Targets are Objects

Kitware

Targets are Objects

- Developer can focus on a single target and not the whole system
  - What include directories will users need?
  - What –D flags will users need?
  - What compile flags will users need?
  - What version of C++ will users need?
  - What flags and options will the users not need?
    - controlled with public and private declarations

“Usage Requirements” aka Modern CMake

Modern style: target-centric

```
target_include_directories(mylib PUBLIC "mydir")
```

mylib and anything that links to gets `-Imydir`

Classic style: directory-centric

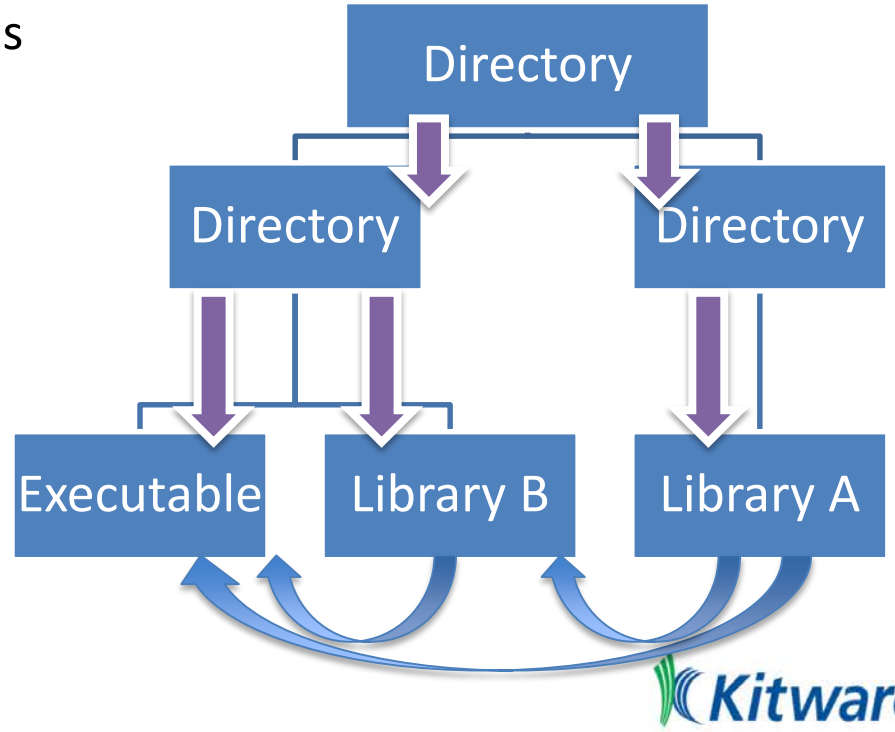
```
include_directories("mydir")
```

Targets in this directory and subdirs get `-Imydir`

Kitware

Before Usage Requirements

- Before Usage Requirements existed we used directory scoped commands such as:
  - `include_directories`
  - `compile_definitions`
  - `compile_options`
- Consumers have to know:
  - Does the dependency generate build tree files
  - Does the dependency use any new external package

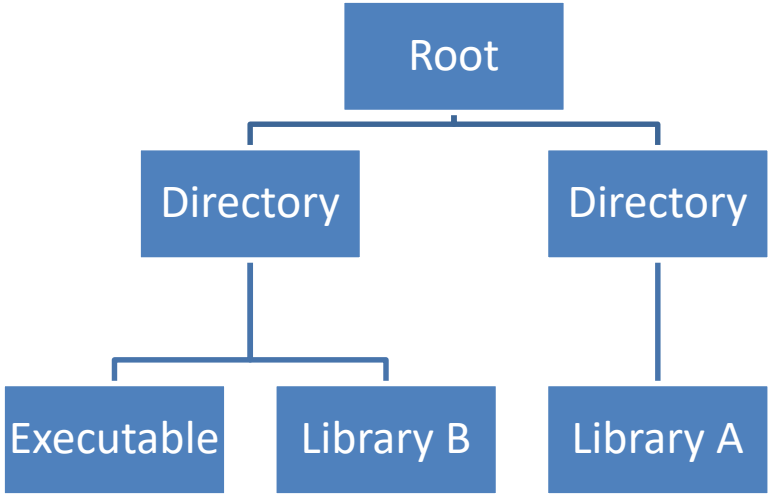


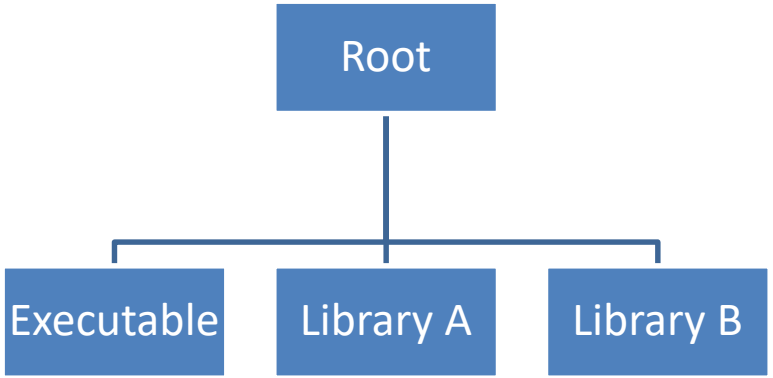
Modern CMake / Usage Requirements

- Modern CMake goal is to have each target fully describe how to properly use it.
- No difference between using internal and external generated targets

Kitware

Modern CMake layout independent





Kitware

Modern CMake Mostly about not using these commands

- add_compile_options()
- add_definitions()
- include_directories()
- link_directories()
- link_libraries()

And treating targets like objects

Kitware

Usage Requirements

- `target_link_libraries` is the foundation for usage requirements
- This foundation is formed by
  - PUBLIC
  - PRIVATE
  - INTERFACE

```
target_link_libraries(trunk PRIVATE root)
target_link_libraries(leaf PUBLIC trunk)
```

target_include_directories

- Propagates include directories

```
target_include_directories(leaf INTERFACE ${zlib_dir})
```

- Anything that links to leaf will automatically have the zlib_dir on the include line

Kitware

target_compile_options

- Propagates compiler options

```
target_compile_options(trunk PRIVATE -march=native)
```

- Only trunk will be built optimized for the current hardware. Anything that links to trunk will not get this flag

Kitware

target_compile_definitions

- Propagates pre-processor definitions

```
target_compile_definitions(root PUBLIC "ROOT_VERSION=42")
```

- Root will have ROOT_VERSION defined and anything that links to it will also

Kitware

INTERFACE Libraries

- An INTERFACE library target does not directly create build output, though it may have properties set on it and it may be installed, exported, and imported.

```
add_library(root INTERFACE)
target_compile_features(root INTERFACE cxx_std_11)
```

Kitware

IMPORTING / EXPORTING

Imported Targets

- Logical name for an outside library
- Reference like any other target

```
add_library(math STATIC IMPORTED)
set_property(TARGET math
             PROPERTY
             IMPORTED_LOCATION /usr/lib/libm.a
             )
target_link_libraries(trunk PUBLIC math)
```

Kitware

Imported Targets

- Per-configuration import rules
- Better than optimized/debug keywords

```
find_library(math_REL NAMES m)
find_library(math_DBG NAMES md)
add_library(math STATIC IMPORTED)
set_target_properties(math
                      PROPERTIES
                      IMPORTED_LOCATION "${math_REL}"
                      IMPORTED_LOCATION_DEBUG "${math_DBG}"
                      )

target_link_libraries(trunk PUBLIC math)
```

Kitware

Exporting Targets

- Install rules can generate imported targets

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

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

Kitware

Conditional Includes

- Able to specify include directories based on if we are building a library or using the installed version

```
target_include_directories(trunk PUBLIC
    $<BUILD_INTERFACE:
     ${CMAKE_CURRENT_SOURCE_DIR}/path/in/src/tree>
    $<INSTALL_INTERFACE:
     $<INSTALL_PREFIX>/include/package/>
  )
```

Kitware

Generating Export Package

- This is constructing components needed for the CMake-aware config package
- CMakePackageConfigHelpers can help with the generation of the <Name>Config.cmake file
- Exporting of find package calls has to replicated inside <Name>Config.cmake, but CMakeFindDependencyMacro helps simplfy this

Kitware

Generating Export Package

```
include(CMakePackageConfigHelpers)
# generate the config file that is includes the exports
configure_package_config_file(Config.cmake.in
  "${CMAKE_CURRENT_BINARY_DIR}/TreeConfig.cmake"
  INSTALL_DESTINATION "lib/cmake/example"
  )
```

```
include(CMakeFindDependencyMacro)
find_dependency(PNG REQUIRED)

include ( "${CMAKE_CURRENT_LIST_DIR}/TreeTargets.cmake" )
```

Kitware

Exporting Targets

```
# Create imported target root
add_library(root INTERFACE IMPORTED)

set_target_properties(root PROPERTIES
  INTERFACE_COMPILE_DEFINITIONS "ROOT_VERSION=42"
  INTERFACE_COMPILE_FEATURES "cxx_std_11"
  INTERFACE_COMPILE_OPTIONS "\$<\$<NOT:\$<CONFIG:DEBUG>>:>;\$
)

# Create imported target trunk
add_library(trunk SHARED IMPORTED)

set_target_properties(trunk PROPERTIES
  INTERFACE_INCLUDE_DIRECTORIES "${_IMPORT_PREFIX}/include/pa
)

# Create imported target leaf
add_library(leaf SHARED IMPORTED)

set_target_properties(leaf PROPERTIES
  INTERFACE_LINK_LIBRARIES "trunk"
)
```

Kitware

CMake 3.8: CUDA

```
add_library(support STATIC support_functions.cu)
set_target_properties(support PROPERTIES
  CUDA_SEPARABLE_COMPILATION ON
  POSITION_INDEPENDENT_CODE ON)
target_link_libraries(support PRIVATE compiler_info)

add_library(black_scholes
  black_scholes/Serial.cpp
  black_scholes/Parallel.cu
)
target_link_libraries(black_scholes PUBLIC compiler_info support)
```

```
[ 20%] Building CUDA object CMakeFiles/support.dir/support_functions.cu.o
/usr/local/cuda/bin/nvcc   -I/Users/robert/Work/cmake_tutorial/cuda_src/producer/compiler_inf
o  -arch=sm_30 -g -Xcompiler=-fPIC   -Xcompiler=-Wall -Xcompiler=-Wshadow,-Wunused-parameter
-std=c++11 -x cu -dc /Users/robert/Work/cmake_tutorial/cuda_src/producer/support_functions.cu
 -o CMakeFiles/support.dir/support_functions.cu.o
[ 40%] Linking CUDA static library libsupport.a
```

INSTALL RULES

Kitware

Install Rules

- Specify rules to run at install time
- Can install targets, files, or directories

```
add_library(leaf SHARED leaf.cxx)
install(TARGETS root trunk leaf parasite
  RUNTIME DESTINATION bin
  LIBRARY DESTINATION lib
  ARCHIVE DESTINATION lib
)
```

Install Rules

- To install files:

```
install(FILES
  trunk.h
  leaf.h
  DESTINATION include
)
```

Kitware

Using Config Modules

- `find_package` also supports config modules
- Config modules are generated by CMake export command
- Automatically generate import targets with all information, removing the need for consuming projects to write a find module

Kitware

CMake Scripts

- cmake –E command
  - Cross platform command line utility for:
  - Copy file, Remove file, Compare and conditionally copy, time, others

- cmake –P script.cmake
  - Cross platform scripting utility
  - Does not generate CMakeCache.txt
  - Ignores commands specific to generating build environment

Kitware

OBJECT Libraries

```
add_library(root  OBJECT root.cxx)
add_library(trunk OBJECT trunk.cxx)
add_library(leaf  SHARED leaf.cxx)
target_link_libraries(leaf root trunk)
```

```
[100%] Linking CXX shared library libleaf.so
/usr/bin/c++ -fPIC   -shared -Wl,-soname,libleaf.so
             -o libleaf.so leaf.cxx.o root.cxx.o trunk.cxx.o
```

Kitware

CTEST

Automatic Testing Benefits

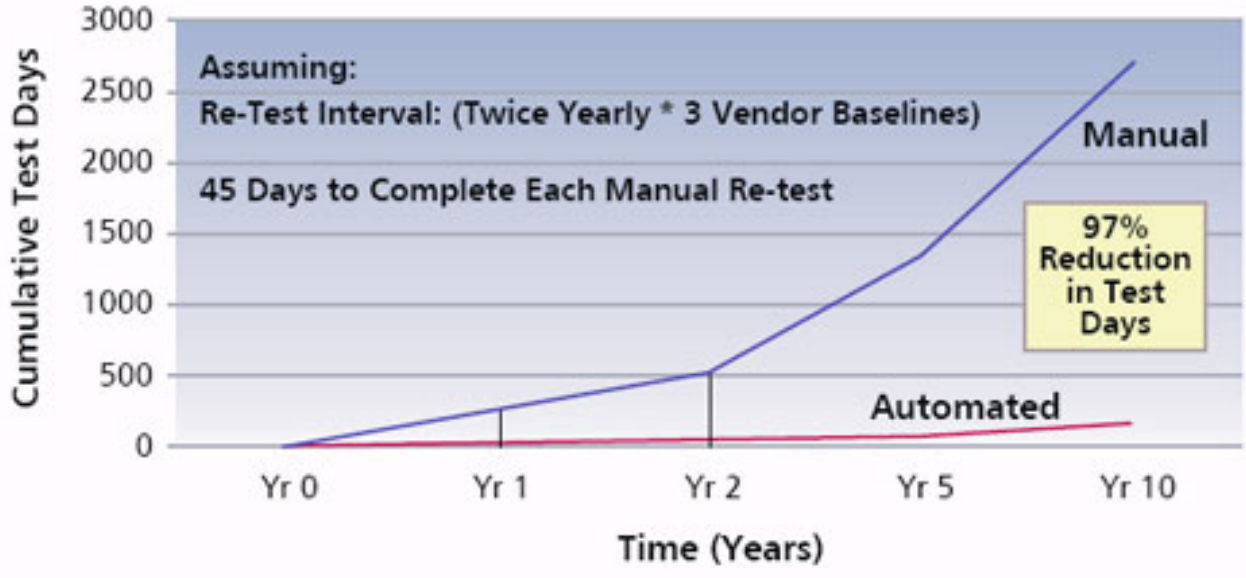
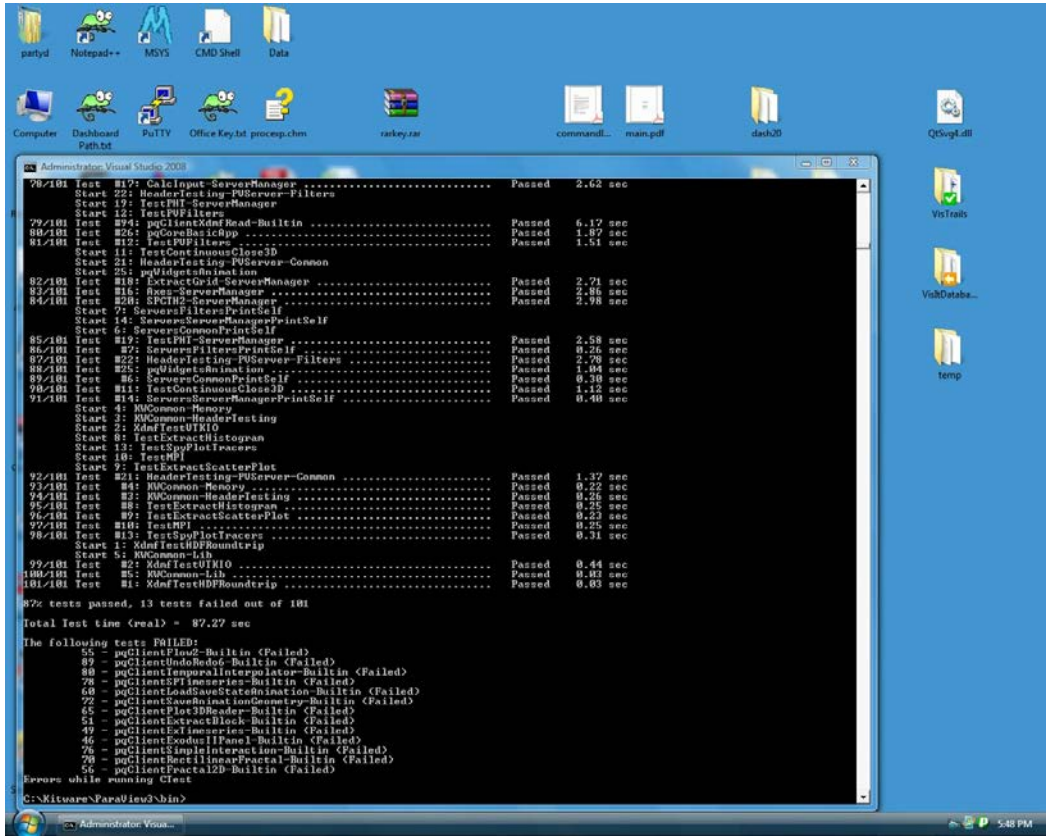


Figure 2. "Example Automated Software Testing Savings over Time"

"Automated Software Testing,"
1999, Dustin, et al, Addison Wesley

Kitware

Video of ParaView Nightly Testing



Testing with CMake

- Testing needs to be enabled by calling `include(CTest)` or `enable_testing()`

```
add_test(NAME testname
         COMMAND exename arg1 arg2 ...)
```

  - Executable should return 0 for a test that passes
- ctest – an executable that is distributed with cmake that can run tests in a project.
- CDash – Web based dashboard to show testing results.

Kitware

CTest

- Run ctest at the top of a binary directory to run all tests

```
$ ctest
Test project /tmp/example/bin
    Start 1: case1
1/1 Test #1: case1 .............................   Passed    0.00 sec
    Start 2: case2
2/2 Test #2: case2 .............................   Passed    0.00 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =   0.01 sec
```

Kitware

CTest

- -j option allows you to run tests in parallel
- -R option allows you to choose a test
- Running tests from Makefiles or projects
  - make test
  - Build RUN_TESTS project
- ctest --help for more information

Kitware

GoogleTest integration

```
include(GoogleTest)
add_executable(tests tests.cpp)
target_link_libraries(tests GTest::GTest)
```

- gtest_discover_tests: new in CMake 3.10.
  - CMake asks the test executable to list its tests.
    Finds new tests without rerunning CMake.

```
gtest_discover_tests(tests TEST_PREFIX new:)
```

Kitware

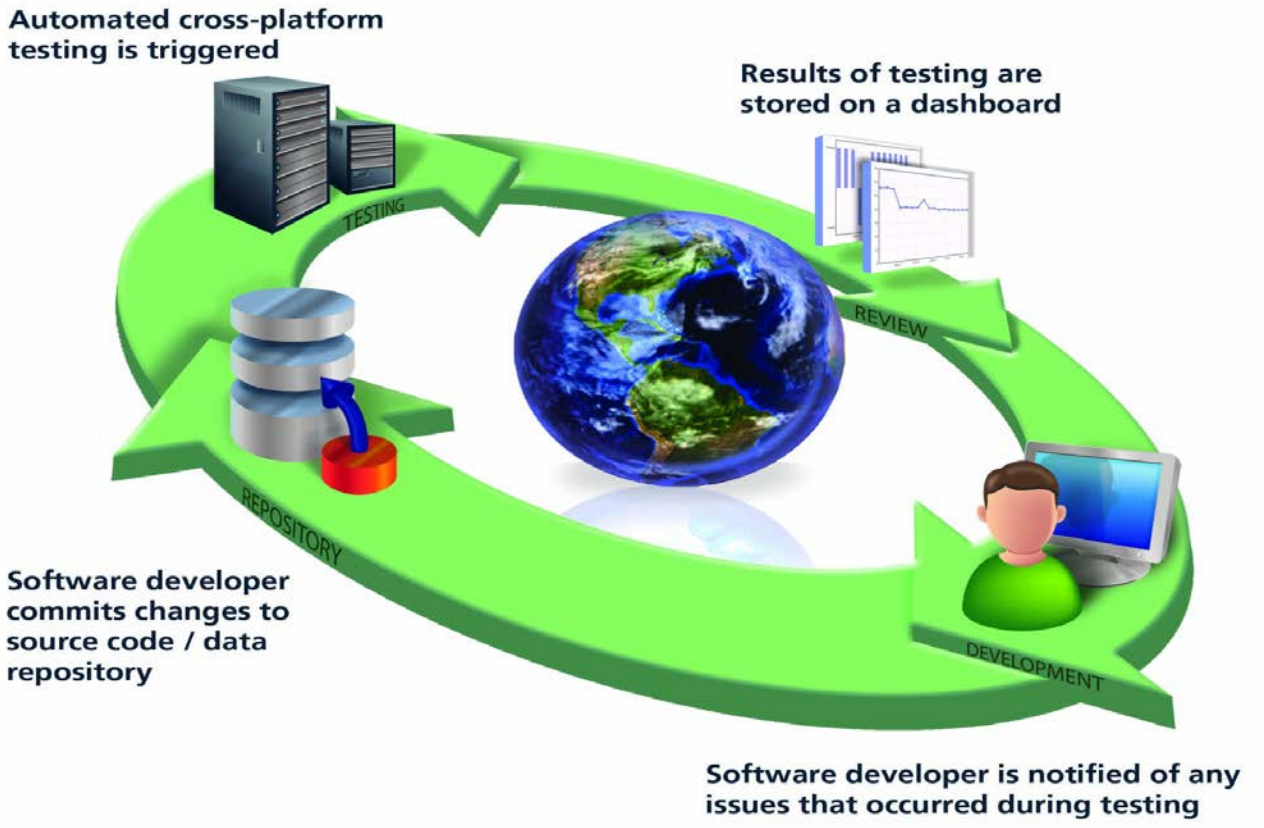
Static Analysis

- Supported tools include:
  - include-what-you-use
  - link-what-you-use
  - clang-tidy
  - cpplint
  - cppcheck
- Setup instructions available here:
  - https://blog.kitware.com/static-checks-with-cmake-cdash-iwyu-clang-tidy-lwyu-cpplint-and-cppcheck/

Kitware

CDash

Software Process Dashboards



Kitware

CDash Dashboard www.cdash.org

Style

Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
dashmacmini5.kitware	KWStyle	7	0	0	0	0				20 hours ago

Nightly Expected

Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
dash2win64-windows.kitware	Windows-VS9-ninja	7	0	0	0	0	0	0	309	7 hours ago
dash2win64.kitware	Windows-icl-11.1-64	7	0	0	0	0	0	0	314	10 hours ago
dash2win64.kitware	Windows-icl-11.1-32	7	0	0	0	0	0	0	314	12 hours ago
vs8.elemtech	Win64-VS80	73	0	0	0	0	0	0	325 +2	8 hours ago
amber12.kitware	Win64-vs10-WINSDK-7.1	7	0	0	0	0	0	0	314	19 hours ago
dash2win64.kitware	Win64-vs10-Tv90	7	0	0	0	0	0	0	320	15 hours ago
dash2win64.kitware	Win64-vs10	7	0	0	0	0	0	0	317	17 hours ago
vs8.elemtech	Win64-nmake80	73	0	0	0	0	0	0	318 +2	8 hours ago
dash2win64.kitware	Win64-nmake10	7	0	0	0	0	0	0	312	8 hours ago

CDash works with other CI tools

- Jenkins
- Buildbot
- Gitlab/CI
- ctest scripts and cronjobs
- CircleCI
- Travis

Kitware

Search for relevant results

Filters Help

Match all of the following rules:

Site	contains	microsoft	- +
Group	is	Nightly Expected	- +
Tests Failed	is greater than	0	- +

Apply Clear Create Hyperlink

Nightly Expected 6 builds

Site	Build Name	Update	Configure		Build		Test			
		Revision	Error	Warn	Error	Warn	Not Run	Fail	Pass	Start Time
gillesk.microsoft	VS2017 x86.rel	602d4c	0	0	0	0	0	4^{+4}_{-4}	471_{-4}	10 hours ago
gillesk.microsoft	VS2015 x64.rel	602d4c	0	0	0	0	0	4^{+3}	476_{-3}	10 hours ago
gillesk.microsoft	VS2012 x86.rel	602d4c	0	0	0	0	0	3^{+3}	412_{-3}	5 hours ago
gillesk.microsoft	VS2012 x64.rel	602d4c	0	0	0	0	0	3^{+3}	412_{-3}	5 hours ago
gillesk.microsoft	VS2017 x64.rel	602d4c	0	0	0	0	0	3^{+3}_{-4}	472_{-3}	10 hours ago
gillesk.microsoft	VS2015 x86.rel	602d4c	0	0	0	0	0	3^{+3}	477_{-3}	10 hours ago

Kitware

Compare results across systems

Testing summary for kwsys.testConsoleBuf performed between 2018-09-13T01:00:00 and 2018-09-14T01:00:00

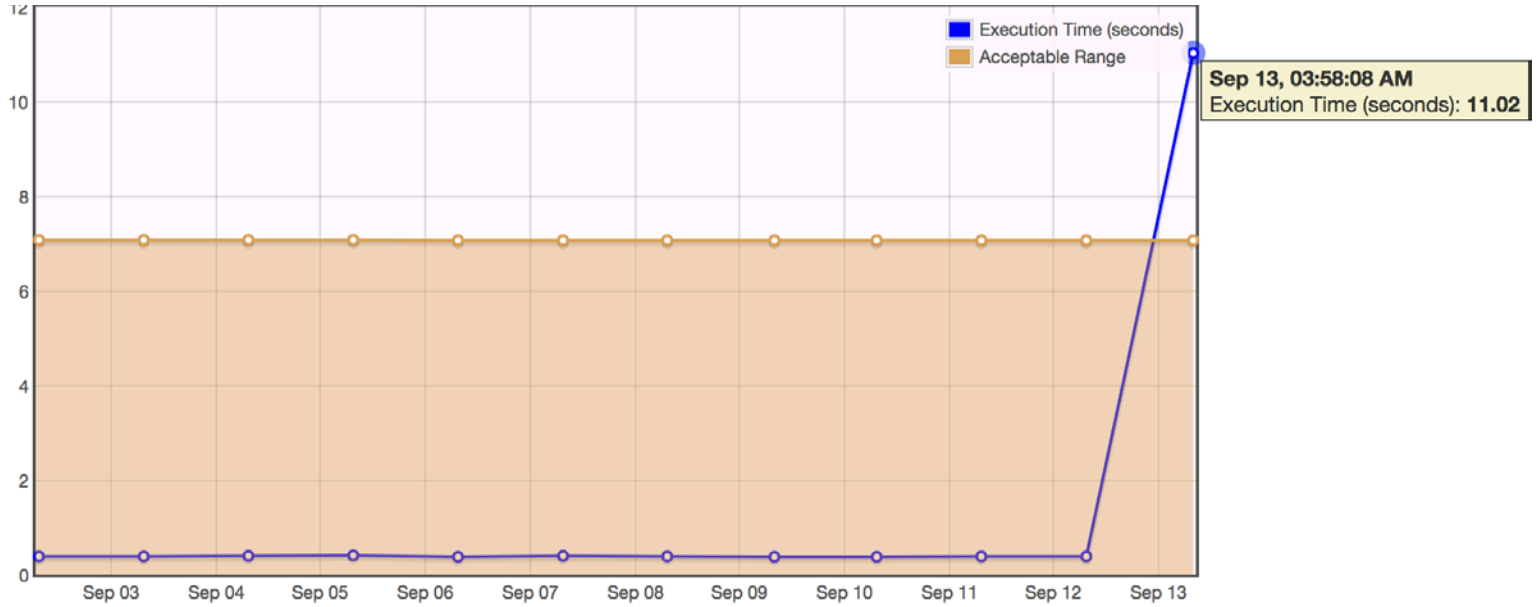
98% passed, 2 failed out of 104.

Show Test Failure Trend
Download Table as CSV File

Site ▲	Build Name	Build Stamp	Status ▲	Time (s)	Build Revision
gillesk.microsoft	VS2017 x86.rel	20180913-0100-Nightly	Failed	11.02	602d4c6e06673b9864ad2f8bb3d706d5bd440c1a
trinsic.kitware	vs14-64-ninja	20180913-0100-Nightly	Failed	13.66	602d4c6e06673b9864ad2f8bb3d706d5bd440c1a
aaargh.kitware.com	Linux-EL7-Intel-16.0.0	20180913-0100-Nightly	Passed	0.02	602d4c6e06673b9864ad2f8bb3d706d5bd440c1a
aaargh.kitware.com	Linux-EL7-Intel-16.0.1	20180913-0100-Nightly	Passed	0.02	602d4c6e06673b9864ad2f8bb3d706d5bd440c1a
aaargh.kitware.com	Linux-EL7-Intel-16.0.2	20180913-0100-Nightly	Passed	0.02	602d4c6e06673b9864ad2f8bb3d706d5bd440c1a

Kitware

Track test timing

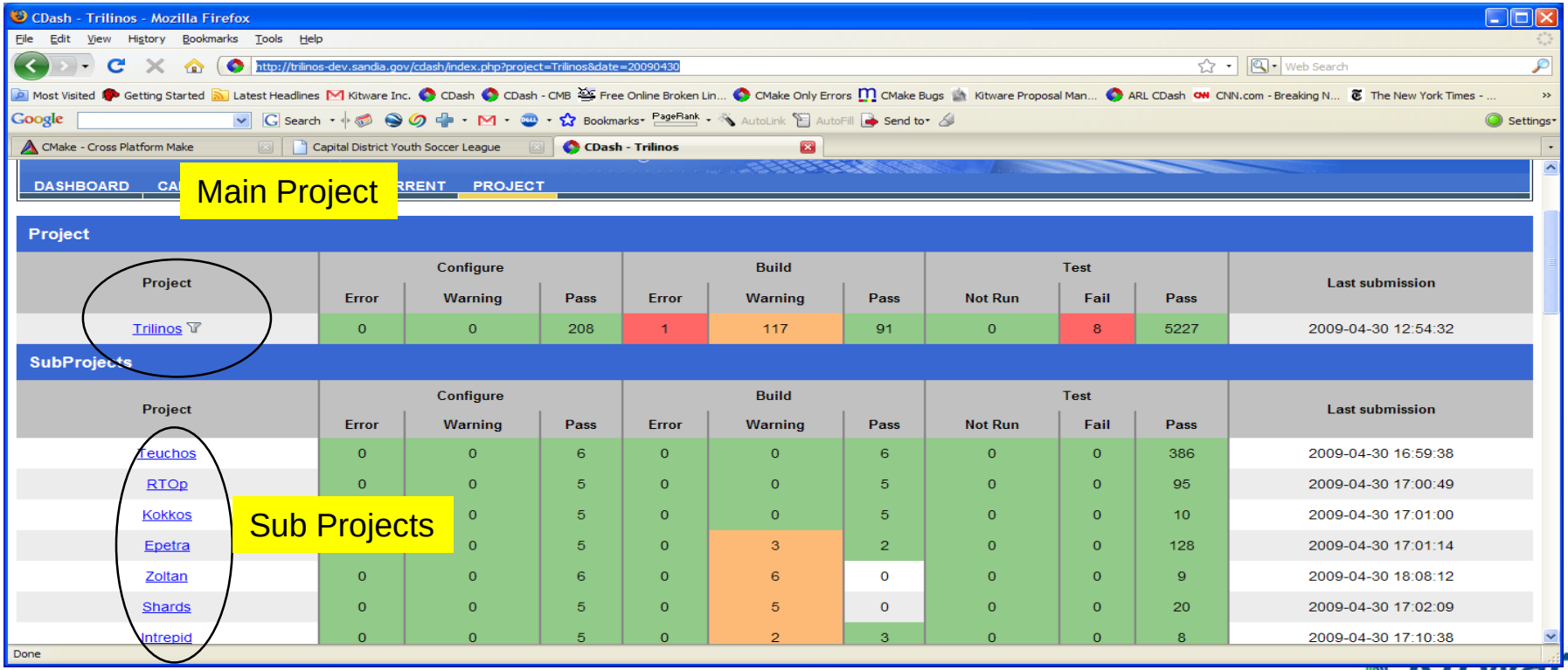


Test output

```
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.
```

Kitware

CDash Subproject Support



Project	Configure Error	Configure Warning	Configure Pass	Build Error	Build Warning	Build Pass	Test Not Run	Test Fail	Test Pass	Last submission
Trilinos	0	0	208	1	117	91	0	8	5227	2009-04-30 12:54:32

Project	Configure Error	Configure Warning	Configure Pass	Build Error	Build Warning	Build Pass	Test Not Run	Test Fail	Test Pass	Last submission
Teuchos	0	0	6	0	0	6	0	0	386	2009-04-30 16:59:38
RTOp	0	0	5	0	0	5	0	0	95	2009-04-30 17:00:49
Kokkos	0	0	5	0	0	5	0	0	10	2009-04-30 17:01:00
Epetra	0	0	5	0	3	2	0	0	128	2009-04-30 17:01:14
Zoltan	0	0	6	0	6	0	0	0	9	2009-04-30 18:08:12
Shards	0	0	5	0	5	0	0	0	20	2009-04-30 17:02:09
Intrepid	0	0	5	0	2	3	0	0	8	2009-04-30 17:10:38

CDash Queries

Show the HEAVY builds for the last two weeks:

Filters Help

Match all of the following rules:

Build Name	contains	HEAVY	- +
Build Time	is after	2 weeks weeks ago	- +

Apply Clear Create Hyperlink

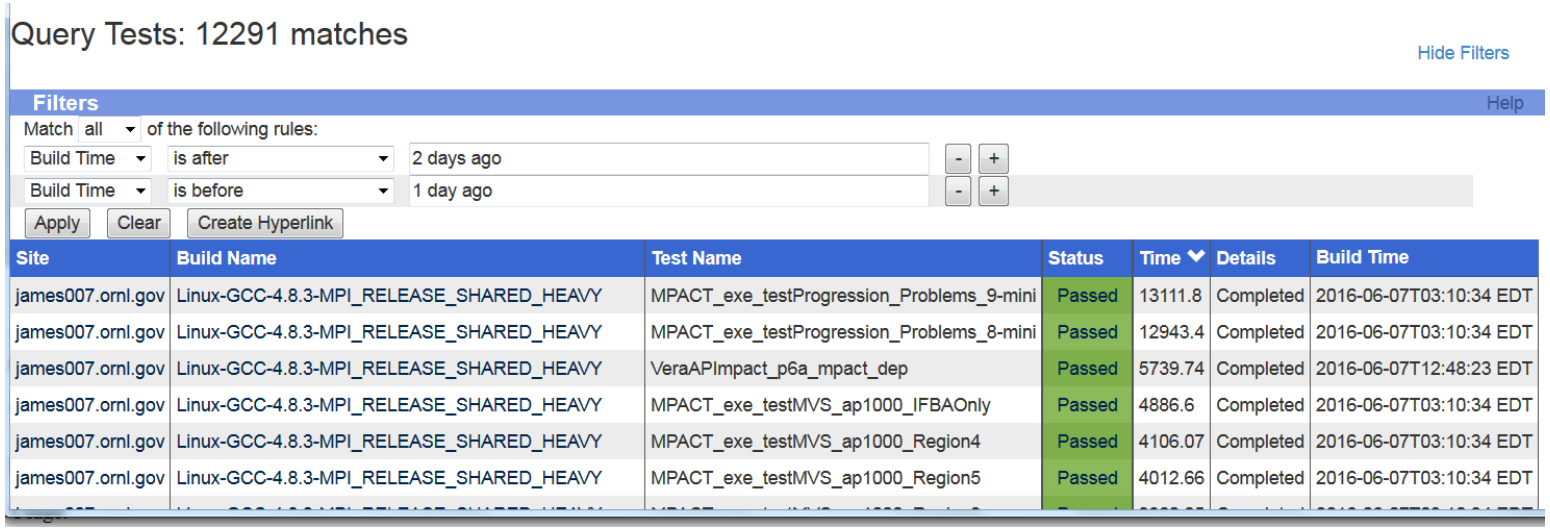
Nightly

Site	Build Name	Update	Configure		Build		Test				
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	Start Time	Labels
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	1	1796	21 hours ago	(19 labels)
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	0	1796	Jun 07, 2016 - 01:10 EDT	(19 labels)
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	1	1795	Jun 06, 2016 - 01:10 EDT	(19 labels)
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	0	1796	Jun 05, 2016 - 01:10 EDT	(19 labels)
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	0	1796	Jun 04, 2016 - 01:10 EDT	(19 labels)
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	1	1794	Jun 03, 2016 - 01:10 EDT	(19 labels)
james007.ornl.gov	Linux-GCC-4.8.3-	1	0	56	0	251	0	0	1795	Jun 02, 2016 - 01:10 EDT	(19

Kitware

CDash Queries

Show most expensive tests yesterday:



Query Tests: 12291 matches

Hide Filters

Filters — Help

Match all of the following rules:

Build Time | is after | 2 days ago

Build Time | is before | 1 day ago

Apply | Clear | Create Hyperlink

Site	Build Name	Test Name	Status	Time	Details	Build Time
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testProgression_Problems_9-mini	Passed	13111.8	Completed	2016-06-07T03:10:34 EDT
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testProgression_Problems_8-mini	Passed	12943.4	Completed	2016-06-07T03:10:34 EDT
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	VeraAPImpact_p6a_mpact_dep	Passed	5739.74	Completed	2016-06-07T12:48:23 EDT
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testMVS_ap1000_IFBAOnly	Passed	4886.6	Completed	2016-06-07T03:10:34 EDT
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testMVS_ap1000_Region4	Passed	4106.07	Completed	2016-06-07T03:10:34 EDT
james007.ornl.gov	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testMVS_ap1000_Region5	Passed	4012.66	Completed	2016-06-07T03:10:34 EDT

Kitware

CTest Command Wrappers Output

Build Time:2009-05-04T01:53:37 MDT

Found 1 Warnings

Errors are here.

Warning while building C++ object file "CMakeFiles/Kokkos_BaseSparseSolve.dir/cxx_main.cpp.o" in target Kokkos_BaseSparseSolve.	
Source File	packages/kokkos/test/BaseSparseSolve/cxx_main.cpp
Label	Kokkos
Command	"/Users/bmpersc/bin/gcc-4.3.3/bin/g++" "-mmacosx-version-min=10.5" "-ansi" "-pedantic" "-Wall" "-Wno-long-long" "-Wwrite-strings" "-g" "-O0" "-D_GLIBCXX_DEBUG" "-I/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/BUILD/packages/kokkos/src" "-I/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src" "-I/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/test/BaseSparseSolve/../BaseSparseMultiply" "-o" "CMakeFiles/Kokkos_BaseSparseSolve.dir/cxx_main.cpp.o" "-c" "/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/test/BaseSparseSolve/cxx_main.cpp"
Directory	/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/BUILD/packages/kokkos/test/BaseSparseSolve
Exit Condition	0
Standard Output	
Standard Error	/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp: In member functio /Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/test/BaseSparseSolve/cxx_main.cpp:262: instanti /Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:646: warning: sugg /Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:693: warning: sugg /Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp: In member functio /Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/test/BaseSparseSolve/cxx_main.cpp:287: instanti /Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:541: warning: sugg /Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:583: warning: sugg

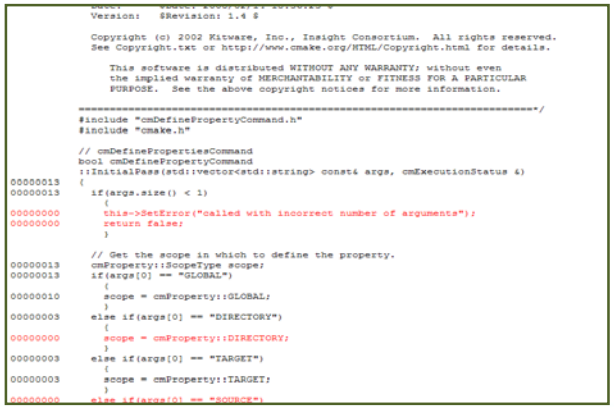
Kitware

CDash 1.5.0 © 2009 Kitware Inc.
[report problems]

Kitware

Coverage Display GCov/Bullseye

./Source/CTest/cmCTestUpdateHandler.cxx	68.21%	45	1
./Source/cmMakefileLibraryTargetGenerator.cxx	68.48%	60	2
./Source/cmTargetLinkLibrariesCommand.cxx	69.17%	17	1
./Source/cmGetPropertyCommand.cxx	69.31%	36	2
./Source/cmExportInstallFileGenerator.cxx	69.32%	16	2
./Source/kwsys/ProcessUNIX.c	69.33%	371	11
./Source/cmVariableWatch.cxx	69.44%	8	1
./Source/cmSystemTools.h	69.64%	1	5
./Source/cmComputeLinkDepends.cxx	69.89%	78	5
./Source/CTest/cmCTestStartCommand.cxx	70.00%	12	0
./Source/cmMakefileExecutableTargetGenerator.cxx	70.83%	16	1
./Source/cmLinkLibrariesCommand.cxx	70.83%	7	0
./Source/cmMakeDepend.cxx	71.01%	44	1
./Source/CTest/cmCTestBuildCommand.cxx	71.74%	26	0
./Source/cmsys/auto_ptr.hxx	71.88%	1	1
./Source/kwsys/testCommandLineArguments.cxx	71.88%	7	1
./Source/CTest/cmCTestSVN.cxx	72.07%	57	2
./Source/cmScriptGenerator.cxx	72.34%	20	1



```
            Version:   $Revision: 1.4 $

            Copyright (c) 2002 Kitware, Inc., Insight Consortium.  All rights reserved.
            See Copyright.txt or http://www.cmake.org/HTML/Copyright.html for details.

               This software is distributed WITHOUT ANY WARRANTY; without even
               the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
               PURPOSE.  See the above copyright notices for more information.

          ============================================================================*/
          #include "cmDefinePropertyCommand.h"
          #include "cmake.h"

          // cmDefinePropertiesCommand
          bool cmDefinePropertyCommand
          ::InitialPass(std::vector<std::string> const& args, cmExecutionStatus &)
00000013  {
00000013    if(args.size() < 1)
              {
00000000      this->SetError("called with incorrect number of arguments");
00000000      return false;
              }

            // Get the scope in which to define the property.
00000013    cmProperty::ScopeType scope;
00000013    if(args[0] == "GLOBAL")
              {
00000010      scope = cmProperty::GLOBAL;
              }
00000003    else if(args[0] == "DIRECTORY")
              {
00000000      scope = cmProperty::DIRECTORY;
              }
00000003    else if(args[0] == "TARGET")
              {
00000003      scope = cmProperty::TARGET;
              }
00000000    else if(args[0] == "SOURCE")
```

```
        Coverage produced by bullseye covbr tool:
          www.bullseye.com/help/ref_covbr.html
        * An arrow --> indicates incomplete coverage.
        * An X indicates a function that was invoked, a switch label that
          was exercised, a try-block that finished, or an exception handler
          that was invoked.
        * A T or F indicates a boolean decision that evaluated true or false,
          respectively.
        * A t or f indicates a boolean condition within a decision if the
          condition evaluated true or false, respectively.
        * A k indicates a constant decision or condition.
        * The slash / means this probe is excluded from summary results.
      ...
              20  #include "cmLocalGenerator.h"
              21  #include "cmGlobalGenerator.h"
              22
        X     23  bool cmCTestStartCommand
              24  ::InitialPass(std::vector<std::string> const& args, cmExecutionSta
              25  {
        -->F  26    if (args.size() < 1)
              27      {
              28      this->SetError("called with incorrect number of arguments");
              29      return false;
      ...
              37    cnt++;
              38
              39    this->CTest->SetSpecificTrack(0);
        -->F  40    if ( cnt < args.size() -1 )
              41      {
        -->   42      if ( args[cnt] == "TRACK" )
              43        {
              44        cnt ++;
              45        this->CTest->SetSpecificTrack(args[cnt].c_str());
      ...
              47        }
```

Kitware

Valgrind / Purify

Dynamic analysis started on 2009-05-03 03:36:06

Site Name: dash17.kitware
Build Name: Linux-g++4.0

Name	Status	Memory Leak	Uninitialized Memory Read	Potential Memory Leak	Uninitialized Memory Conditional	Mismatched Deallocate	Freeing Invalid Memory	Invalid Pointer Read	Invalid Pointer Write	Labels
QtChart-TestBarSeriesColors	Passed		1	25						
QtChart-TestChartWidget	Passed		1	26						
Mace	Passed			2						
TestHyperOctreeContourFilter	Passed			2	1					
TestUncertaintyTubeFilter	Passed			2						
TestMultiBlock	Passed			2						
TemporalStatistics	Passed			3						
TestGenericCutter	Passed			2						
TestActorLightingFlag	Passed			2						
TestLabelPlacer	Passed			2						
TestOpacity	Passed			2						
TestTextActor3DAlphaBlending	Passed			2						
TestAreaSelections	Passed			2						
TestTranslucentImageActorDepthPeeling	Passed		2	2						
TestGenericVertexAttributesGLSLDepthPeelingPass	Passed			2						

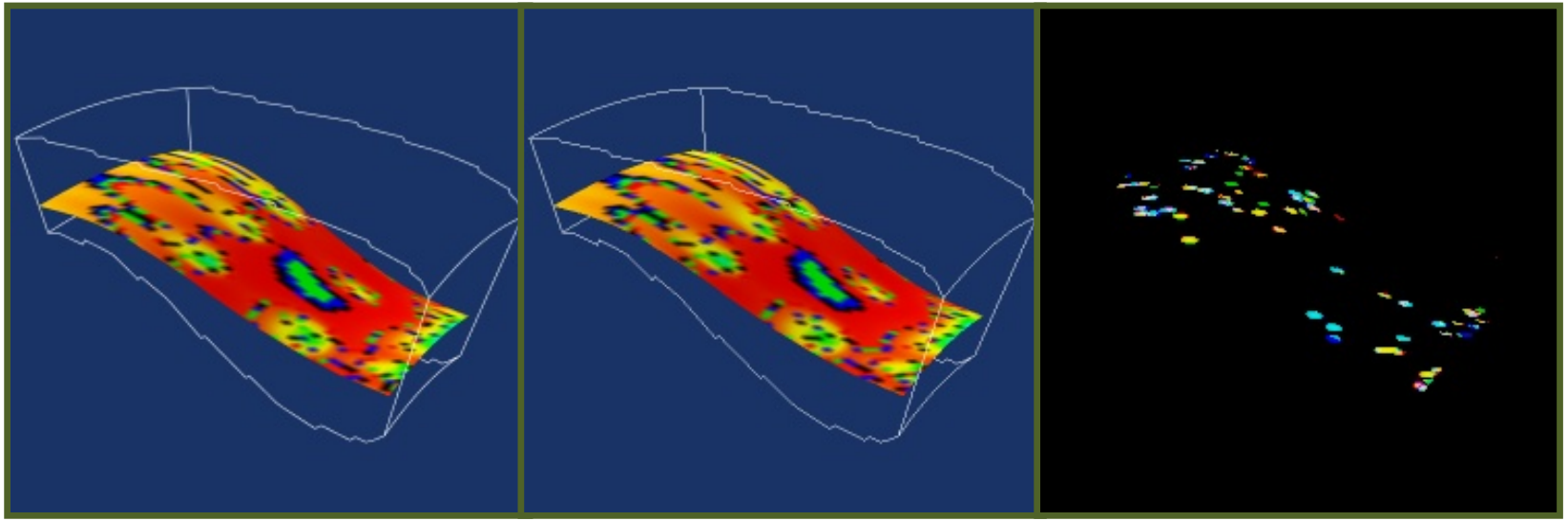
Find: asio Next Previous Highlight all Match case

Dynamic analysis started on 2009-05-04 03:37:17

Site Name: dash17.kitware
Build Name: Linux-g++4.0
TestMultiBlockPassed

```
==3002== Memcheck, a memory error detector.
==3002== Copyright (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
==3002== Using LibVEX rev 1732, a library for dynamic binary translation.
==3002== Copyright (C) 2004-2007, and GNU GPL'd, by OpenWorks LLP.
==3002== Using valgrind-3.2.3, a dynamic binary instrumentation framework.
==3002== Copyright (C) 2000-2007, and GNU GPL'd, by Julian Seward et al.
==3002== For more details, rerun with: -v
==3002==
==3002==
==3002== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 119 from 2)
==3002== malloc/free: in use at exit: 30,294 bytes in 327 blocks.
==3002== malloc/free: 37,724 allocs, 37,397 frees, 5,207,986 bytes allocated.
==3002== For counts of detected errors, rerun with: -v
==3002== searching for pointers to 327 not-freed blocks.
==3002== checked 2,298,764 bytes.
==3002==
 ==3002== 64 bytes in 1 blocks are still reachable in loss record 15 of 34
==3002==    at 0x401DC87: realloc (vg_replace_malloc.c:306)
==3002==    by 0x62F83E5: (within /usr/lib/libX11.so.6.2.0)
==3002==    by 0x62F908E: (within /usr/lib/libX11.so.6.2.0)
==3002==    by 0x62F95F0: XrmGetStringDatabase (in /usr/lib/libX11.so.6.2.0)
==3002==    by 0x659FB22: (within /usr/lib/libXt.so.6.0.0)
==3002==    by 0x65A0ED4: _XtDisplayInitialize (in /usr/lib/libXt.so.6.0.0)
==3002==    by 0x6596DC7: XtOpenDisplay (in /usr/lib/libXt.so.6.0.0)
==3002==    by 0x437DD13: vtkXRenderWindowInteractor::Initialize() (vtkXRenderWindowInteractor.cxx:317)
==3002==    by 0x42EFDD0: vtkRenderWindow::Render() (vtkRenderWindow.cxx:265)
==3002==    by 0x441E401: vtkXOpenGLRenderWindow::Render() (vtkXOpenGLRenderWindow.cxx:1846)
==3002==    by 0x8081AB6: TestMultiBlock(int, char**) (TestMultiBlock.cxx:142)
==3002==    by 0x805B2E8: main (GraphicsCxxTests.cxx:306)
==3002==
```

CDash Image Difference

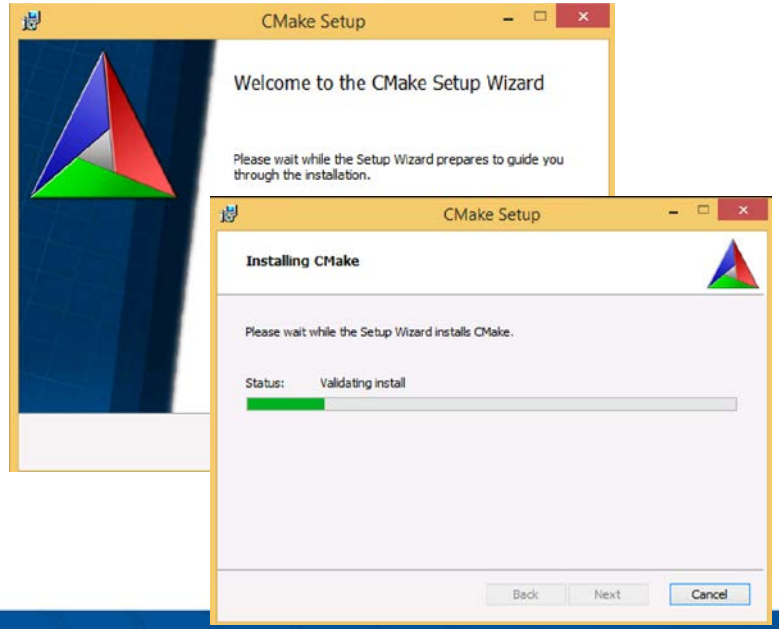


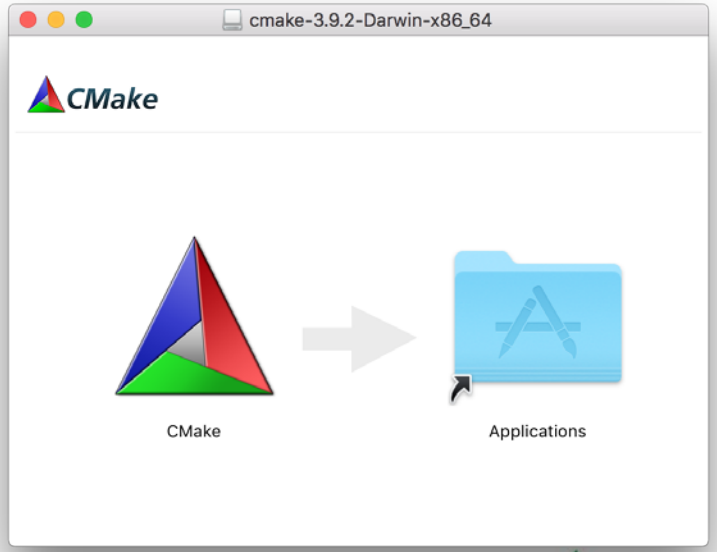
Kitware

CPack

What is CPack

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





Kitware

CPack Features

- Supports CMake-based and non-CMake-based projects
- Unix
  - TGZ and self-extracting TGZ (STGZ)
- Windows
  - WiX – MSI installers
  - NullSoft Scriptable Install System (NSIS / NSIS64)
- Mac OSX
  - DragNDrop
  - PackageMaker
- Deb
  - Debian packages
- RPM
  - RPM package manager

Kitware

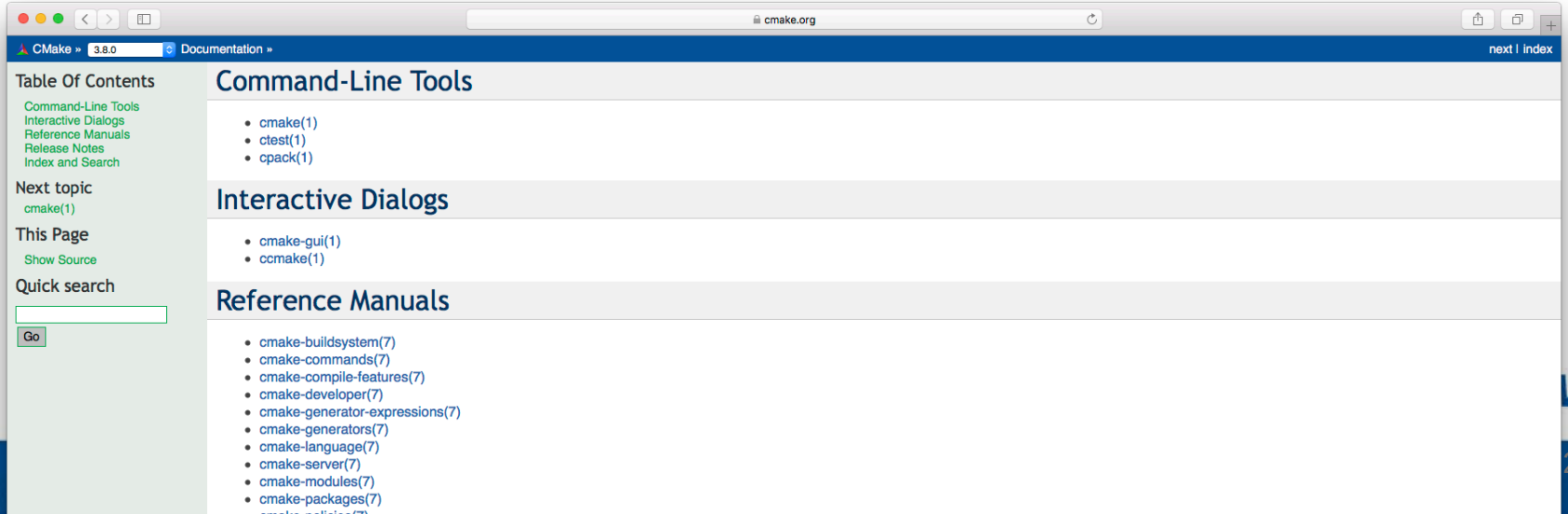
Using CPack

- 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)

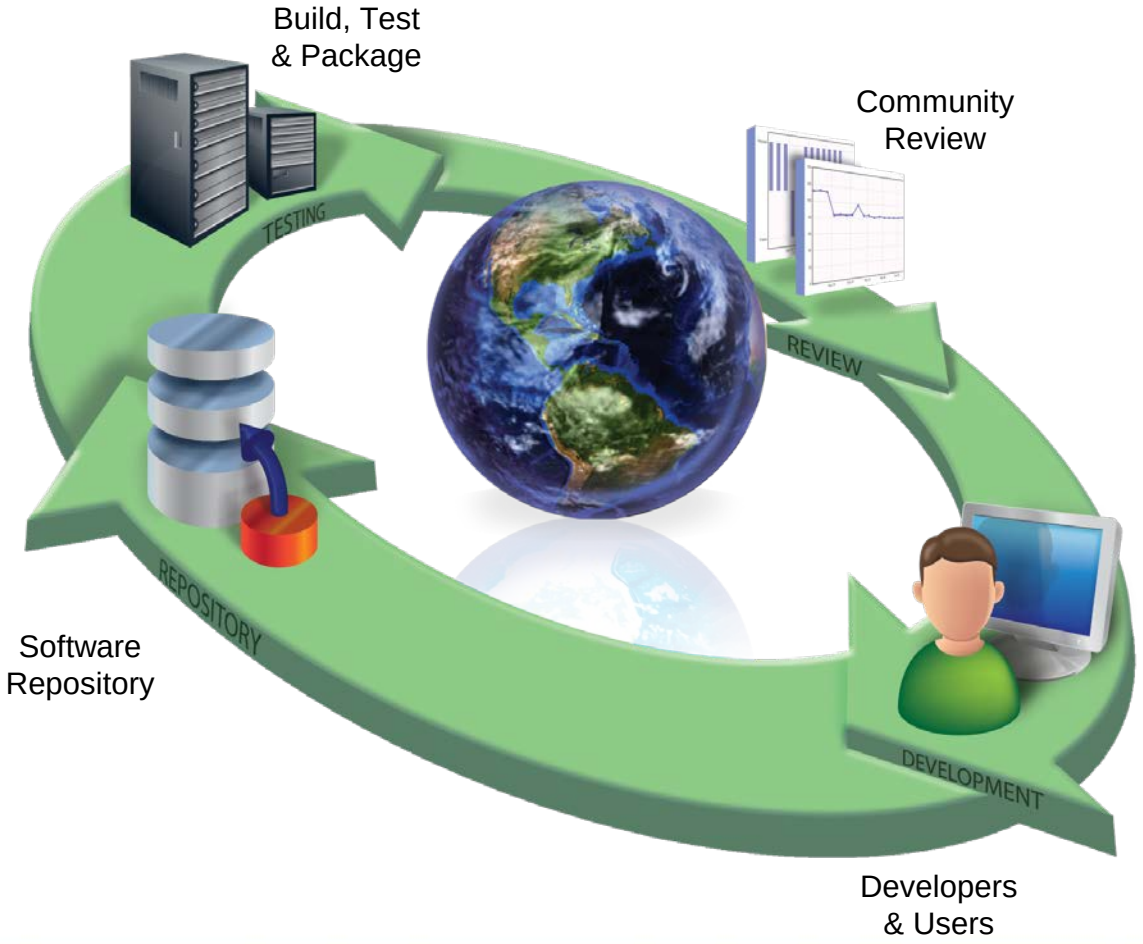
Kitware

Now that you are inspired

- Read “how to write a CMake buildsystem”
  - https://cmake.org/cmake/help/v3.8/manual/cmake-buildsystem.7.htmlExplore the CMake documentation
- Explore the CMake documentation
  - https://www.cmake.org/cmake/help/v3.8/



Thanks



Kitware