Modern CMake

Open Source Tools to Build Test and Deploy C++ Software

Bill Hoffman
bill.hoffman@kitware.com

Alexander Neundorf
neundorf@kde.org
Kitware Quality Software Process

- Software Repository
- CTest/CMake
- CDash Dashboards
- Developers check in code
- Developers review results
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
- CMake
  Release-1-0 branch created in late 2001
- Change the way “everyone” builds c++.
Why CMake? It’s easy, and works well

- A build system that just works
- A build system that is easy to use cross platform

Typical Project without CMake (curl)

$ ls
CHANGES LICENSE curl-config.in missing
CMake acinclude.m4 curl-style.el mkinstalldirs
CMakeLists.txt aclocal.m4 depcomp notes
build docs notes~
COPYING buildconf include packages
CVS buildconf.bat install-sh reconf
ChangeLog compile lib sample.emacs
Makefile config.guess libcurl.pc.in src
Makefile.am config.sub ltmain.sh tests
Makefile.in configure m4 vc6curl.dsw
README configure.ac maketgz

$ ls src/
CMakeLists.txt Makefile.riscos curlsrc.dsp hugehelp.h version.h
CVS Makefile.vc6 curlsrc.dsw macos writeenv.c
Makefile.Watcom Makefile.vc8 curlutil.c main.c writeenv.h
Makefile.am config-amigaos.h curlutil.h makefile.amiga writeout.c
Makefile.b32 config-mac.h getpass.c makefile.dj writeout.h
Makefile.in config-riscos.h getpass.h mkhelp.pl
Makefile.inc config-win32.h homedir.c setup.h
Makefile.m32 config.h.in homedir.h urlglob.c
Makefile.netware curl.rc hugehelp.c urlglob.h
Why CMake? It’s fast

http://blog.qgis.org/?q=node/16: “I was quite surprised with the speed of building Quantum GIS codebase in comparison to Autotools.”

<table>
<thead>
<tr>
<th>Task</th>
<th>CMake</th>
<th>Autotools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure</td>
<td>0:08</td>
<td>Automake: 0:41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configure: 0:20</td>
</tr>
<tr>
<td>Make</td>
<td>12:15</td>
<td>21:16</td>
</tr>
<tr>
<td>Install</td>
<td>0:20</td>
<td>0:36</td>
</tr>
<tr>
<td>Total</td>
<td>12:43</td>
<td>22:43</td>
</tr>
</tbody>
</table>

http://taskwarrior.org/projects/taskwarrior/news
Why CMake? Everyone is using it

KDE 2006 – Tipping Point!

- Google Search Trends and ohloh comparisons with auto*
- 1400+ downloads per day from www.cmake.org
- Major Linux distributions and Cygwin provide CMake packages
- KDE, Second Life, Boost (Experimentally), many others
Why CMake? Quickly adapt to new technologies

• New build 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

• New compiler support
  – clang
  – gcc versions
How CMake Changes 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
CMake is no longer SCREAM MAKE

- Commands may be uppercase or lowercase
  ```
  ADD_EXECUTABLE(Tutorial tutorial.cxx)
  ```
  is equivalent to
  ```
  add_executable(Tutorial tutorial.cxx)
  ```

- No need to repeat variables
  - `endforeach(MYVAR),
    endif(THIS AND THAT OR
    THEOTHER),
    endmacro(DoReallyCoolStuff),
    endfunction(DoBetterStuff)
  - `endforeach(), endif(), endmacro(),
    endfunction()`
CMake Features - continued

• Automatic analysis
  – Implicit dependencies (C, C++, Fortran)
  – Transitive link dependencies
  – Ordering of linker search path and RPATH

• Advanced Makefile generation
  – Modular, Fast, Parallel
  – Color and progress display
  – Help targets – make help
  – Preprocessor targets – make foo.i
  – Assembly targets – make foo.s
CMake Scripts

- cmake –E command
  - Cross platform command line utility
  - Ex. Copy file, Remove file, Compare and conditionally copy, time etc

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

• Module introduced in CMake 2.8
  – Allows the download, configure, build and install of software via custom commands

• Kitware Source Article: October 2009

- Google Protocol buffers
- CLAPACK
- VTK
- Qt
- Trilinos
- Curl
- Boost

Titan
Testing with CMake, CTest and CDash

• Testing command in CMake
  – `add_test ( testname exename arg1 arg2 arg3 …)`
  – Executable is expected to return 0 for passed
  – Can set other test passing conditions based on output matching.

• ctest – an executable that is distributed with cmake that can run tests in a project.
  – Used for continuous integration testing
  – Client for CDash
  – Can be use for both CMake based projects and other build systems
The image is of the CMake Dashboard from www.cdash.org. The dashboard shows various builds and tests, with columns for build date, test status, and error/warn/Min. The builds are categorized as nightly expected, continuous, or experimental. The screenshot includes a table with build names, site names, and build dates for different platforms and configurations. The background of the dashboard has a blue and grey gradient.
CPack

• CPack is bundled with CMake
• Creates professional platform specific installers
  – TGZ and Self extract TGZ (STGZ), NullSoft Scriptable Install System (NSIS), OSX PackageMaker, RPM, Deb
Simple Qt Example

cmake_minimum_required(VERSION 2.8)
project(helloQt)
# find required dependencies
find_package(Qt4 REQUIRED)
# create the executable
add_executable(helloQt WIN32 MACOSX_BUNDLE myqt.cxx )
target_link_libraries(helloQt ${QT_QTMAIN_LIBRARY} ${QT_LIBRARIES})
# installation and packaging
install(TARGETS helloQt DESTINATION bin)
include (InstallRequiredSystemLibraries)
set (CPACK_PACKAGE_VERSION_MAJOR "1")
set (CPACK_PACKAGE_VERSION_MINOR "0")
set(CPACK_PACKAGE_EXECUTABLES "helloQt" "Hello Qt")
include (CPack)
Simple Qt Example with Boost

cmake_minimum_required(VERSION 2.8)
project(helloQt)
# find required dependencies
find_package(Qt4 REQUIRED)
include(${QT_USE_FILE})
set(Boost_USE_STATIC_LIBS ON)
find_package(Boost REQUIRED signals)
include_directories(${Boost_INCLUDE_DIRS})
# create the executable
add_executable(helloQt WIN32 MACOSX_BUNDLE myqt.cxx)
target_link_libraries(helloQt ${QT_QTMAIN_LIBRARY} ${QT_LIBRARIES} ${Boost_LIBRARIES})
# installation and packaging
install(TARGETS helloQt DESTINATION bin)
include (InstallRequiredSystemLibraries)
set (CPACK_PACKAGE_VERSION_MAJOR "1")
set (CPACK_PACKAGE_VERSION_MINOR "0")
set (CPACK_PACKAGE_EXECUTABLES "helloQt" "Hello Qt")
include (CPack)
Finding and using software

• targets with includes and links
• import/export targets
• Alex will talk about