CMake / CTest / CPack

Open Source Tools to build, test, and install software

Bill Hoffman
bill.hoffman@kitware.com

David Cole
david.cole@kitware.com

Julien Jomier
julien.jomier@kitware.com
Trilinos Effort

- Thanks to Tim Shead and Roscoe Bartlett!
- Short Effort
  - Major Work (Dec 08 – Feb 08)
  - CDash / CTest
  - Handle large scale project (50+ subprojects)
Kitware Quality Software Process

CTest/CMake

Software Repository

CDash Dashboards

Developers check in code

Developers review results
Overview

- Introduce myself and Kitware
- About CMake
  - Building with CMake
  - Testing with CTest/CDash
  - Packaging with CPack
**Introductions**

- **Bill Hoffman**
  - 1990 – 1999 GE Global Research Center Computer Vision Group
    - Large C++ research environments
  - 1999 – Present Vice President Kitware Inc

- **Kitware Inc**
  - Founded in 1998
  - Privately held
  - 55 employees and 2 locations (and hiring)
    - Clifton Park NY, USA
    - Chapel Hill, NC, USA
  - Supporting many open source projects
  - Medical Imaging, Super computing, Computer Vision
Kitware: Core Technologies

SOFTWARE PROCESS

- Medical Imaging
- Supercomputing Visualization
- Data Publication
- Computer Vision

Expertise in:
- Behavior/event recognition
- Detection and tracking
- Segmentation
- Change Detection

- Insight ToolKit (ITK)
- Visualization Toolkit (VTK)
- CMake
- ParaView
- Publications and consulting
ParaView

- Parallel Visualization application
- Turn-key wrapper around VTK
- Supports parallel data processing and rendering
- Supports tiled displays, Caves, etc.
- 3D widgets, LOD (level-of-detail) display
- Extended by XML modules
- Extensive animation support
$13 million over 6 years
Leading edge algorithms
Open Source Software
Why CMake

- A build system that just works
- A build system that is easy to use cross platform
- Typical Project without CMake (curl)

```
$ ls
CHANGES                RELEASE-NOTES curl-config.in missing
CMake                  acinclude.m4 curl-style.el mkinstdirs
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            m4              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.inc 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 – Very 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>00:41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00:20</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>
CMake (Everyone is using it)
KDE 2006 – Tipping Point!

- 800+ downloads per day from www.cmake.org
- Major Linux distributions and Cygwin provide CMake packages.
- KDE, Second Life, Boost (Experimentally), many others
What is CMake?

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

- Open-Source License

- History
  - Insight Segmentation & Registration Toolkit (~2000)
  - Changed the way we build
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
Who is involved?

- Users
  - KDE
  - Second Life
  - ITK
  - VTK
  - ParaView
  - Scribus
  - many more

- Supporters
  - Kitware
  - National Library of Medicine
  - Sandia National Labs
  - Los Alamos National Labs
  - NAMIC
  - ARL
CMake Documentation

- Mastering CMake Book
- Web Page: www.cmake.org
- http://www.cmake.org/Wiki/CMake
- mailing list: cmake@cmake.org
- Full reference documentation
  - Ships HTML, man, and command line help
- Tutorial included and tested in source tree (Tests/Tutorial/)
  - configured files
  - optional build components
  - install rules, test properties
  - system introspection
- CPack
- CTest with CDash
CMake Features

- One simple language for all platforms
  - Windows, Mac, Linux, UNIX variants
  - Embedded platforms via cross-compilation
- Generates native build systems
  - Makefiles (GNU, NMake, Borland, etc.)
  - KDevelop, Eclipse
  - Visual Studio 6,7,8,9 IDE
  - Xcode
- Out-of-source build trees leave source clean
- Interactive configuration via GUI
- Multiple configurations (Debug, Release, etc.)
CMake Features (cont.)

• Built-in rules for common targets
  • Executables
  • Shared Libraries / DLLs
  • Static Libraries (archives)
  • OS X Frameworks and App Bundles
• Custom rules for other targets
  • Generated Documentation
  • Generated sources and headers
• Configuration rules
  • System introspection
  • Persistent variables (options, cached results)
  • Configured header files
CMake Features (cont.)

- 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`
Input to CMake

- Simple scripting language in CMakeLists.txt file(s)
- Built-in commands for common rules
  * add_library(MyLib MyLib.cxx)
  * add_executable(MyExe MyMain.cxx)
- Example project using Boost:

```bash
cmake_minimum_required(VERSION 2.6)
project(MyProject)
find_package(Boost REQUIRED thread signals)
include_directories(${Boost_INCLUDE_DIRS})
add_executable(MyExe MyProjectMain.cxx)
target_link_libraries(MyExe ${Boost_LIBRARIES})
```
Installing CMake

- Easy to Get CMake
  - [http://www.cmake.org/cmake/resources/software.html](http://www.cmake.org/cmake/resources/software.html)
  - Many Linux distributions and Cygwin include CMake packages
    - `apt-get install cmake`
  - Installing CMake from www.cmake.org
    - Windows Binary installers
    - Linux Binaries – can be installed anywhere, (don’t need root)
    - Mac
    - Other UNIX on download page
    - Source can bootstrap on any Unix platform
CMake process

Configure Step

- Read CMakeCache.txt
  - Read CMakeLists.txt files
  - Write CMakeCache.txt

Generate Step

- Write Makefiles or projects
Cache editors: cmake-gui (qt), ccmake (curses)
Running CMake from the command line

- Useful for scripted builds or for projects with no options or with options correctly set by default on the first configure

```bash
#CC=gcc; CXX=g++
#CFLAGS, CXXFLAGS
cd MyProjectSourceDir
mkdir ../MyProjectSourceDir-build
mkdir ../MyProjectSourceDir-build
cd ../MyProjectSourceDir-build
make ../MyProjectSourceDir-build
(cmake -Dvar=value)
```
**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
CTest/CDash
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
### Nightly Changes as of 2008-02-21 02:00 EST

#### Style

<table>
<thead>
<tr>
<th>Site</th>
<th>Build Name</th>
<th>Update</th>
<th>Clg</th>
<th>Build</th>
<th>Test</th>
<th>Build Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insight Journal kthware</td>
<td>KWinStyle</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2008-02-21 02:28:33 EST</td>
</tr>
</tbody>
</table>

#### Nightly Expected

<table>
<thead>
<tr>
<th>Site</th>
<th>Build Name</th>
<th>Update</th>
<th>Clg</th>
<th>Build</th>
<th>Test</th>
<th>Build Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titanium IMTS ut</td>
<td>Linux64_Rocks_DCC_Ref</td>
<td>107</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2008-02-21 10:23:00 EST</td>
</tr>
<tr>
<td>kronor kthware</td>
<td>Darwin++</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>55.2</td>
</tr>
<tr>
<td>daph6 kthware</td>
<td>Linux64-g++3332</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18.3</td>
</tr>
<tr>
<td>RogueResearch3</td>
<td>Mac10.5-CMake-Xcode-dbg-prpc6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23.4</td>
</tr>
<tr>
<td>RogueResearch3</td>
<td>Mac10.5-CMake-Xcode-dbg-prc6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23.9</td>
</tr>
</tbody>
</table>
Trilinos (Multi-Package Dashboard)

<table>
<thead>
<tr>
<th>Project</th>
<th>Configure</th>
<th>Build</th>
<th>Test</th>
<th>Last submission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Error</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Warning</td>
<td>Warning</td>
<td>Warning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>Trilinos</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2009-04-30 12:54:32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SubProjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Epexa</td>
</tr>
<tr>
<td>Kokkos</td>
</tr>
<tr>
<td>Kokkos</td>
</tr>
<tr>
<td>Kokkos</td>
</tr>
<tr>
<td>Kokkos</td>
</tr>
<tr>
<td>Kokkos</td>
</tr>
<tr>
<td>Kokkos</td>
</tr>
</tbody>
</table>

Main Project

Sub Projects
Query Filters: customize views
CTest Command wrappers output

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

Found 1 Warnings

Errors are here.

<table>
<thead>
<tr>
<th>Source File</th>
<th>packages/kokkos/test/BaseSparseSolve/cxx_main.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Kokkos</td>
</tr>
<tr>
<td>Directory</td>
<td>/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/BUILD/packages/kokkos/test/BaseSparseSolve</td>
</tr>
<tr>
<td>Exit Condition</td>
<td>0</td>
</tr>
</tbody>
</table>

Warning while building C++ object file "CMakeFiles/Kokkos_BaseSparseSolve.dir/cxx_main.cpp.o" in target Kokkos_BaseSparseSolve.

Standard Output:

```
/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp: In member function '
/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:262: instanti
/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:266: warning: sugg
/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:287: instanti
/Users/bmpersc/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src/Kokkos_BaseSparseSolve.hpp:533: warning: sugg
```
### Valgrind / Purify

Dynamic analysis started on 2008-05-03 03:36:06

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Memory Leak</th>
<th>Uninitialized Memory Read</th>
<th>Potential Memory Leak</th>
<th>Uninitialized Memory Conditional</th>
<th>Mismatched Dealocate</th>
<th>Freeing Invalid Memory</th>
<th>Invalid Pointer Read</th>
<th>Invalid Pointer Write</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>QtChart-TestChartSeriesColors</td>
<td>Passed</td>
<td></td>
<td>1</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QtChart-TestChartWidget</td>
<td>Passed</td>
<td></td>
<td>1</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mice</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestHyperOctreeContourFilter</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestUncertaintyFloydFilter</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestMultiBlock</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TemporalStatistics</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestGenericCutter</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestActRequestlightingFlag</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestLabotPlacar</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestDcally</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestTestActor3DAlphaBlending</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestAreaSelections</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestTranslucentImageActorDepthFeasing</td>
<td>Passed</td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestGenericVariableAttributeGl::GLDepthFeasing</td>
<td>Passed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

```
$0023$ Memcheck, a memory error detector.
$0023$ Copyrights (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
$0023$ Using LinVEX rev 1792, a library for dynamic binary translation.
$0023$ Copyright (C) 2006-2007, and GNU GPL'd, by OpenWorks L.L.C.
$0023$ Using valgrind-3.2.0, a dynamic binary instrumentation framework.
$0023$ Copyrights (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
$0023$ For more details, rerun with: -v

$0023$

$0023$ ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 119 from 2)
$0023$ mallo/free: in use at exit: 33,204 bytes in 327 blocks.
$0023$ mallo/free: allocation: 37,724 allocs, 37,387 frees, 5,207,986 bytes allocated.
$0023$ For counts of detected errors, rerun with: -v
$0023$ searching for pointers to not freed blocks.
$0023$ checked 2,295,714 bytes.
$0023$ 64 bytes in 1 blocks are still reachable in loss record 15 of 34
$0023$ at @0x610DB7: realloc (in replace_malloc.c:39): on top 100:
$0023$ by 0x42F8183 (within /usr/lib/libX11.so.6.2.0)
$0023$ by 0x42F8088 (within /usr/lib/libX11.so.6.2.0)
$0023$ by 0x42F80F0: XDisplayStringDatabase (in /usr/lib/libX11.so.6.2.0)
$0023$ by 0x42F80B2 (within /usr/lib/libX11.so.6.2.0)
$0023$ by 0x42F81A4: XDisplayInitialize (in /usr/lib/libX11.so.6.2.0)
$0023$ by 0x42F81AC: XOpenDisplay (in /usr/lib/libX11.so.6.2.0)
$0023$ by 0x42F82D3: vkRendererWindow::Initialize() (vkRendererWindowInteractor.cpp:137)
$0023$ by 0x42F8DD: vkRendererWindow::Render() (vkRendererWindowInteractor.cpp:248)
$0023$ by 0x42F8F0: vkOpenGLRenderWindow::Render() (vkOpenGLRenderWindowInteractor.cpp:1046)
$0023$ by 0x42F8F0: TestMultiBlock(int, char**) (TestMultiBlock.cpp:142)
$0023$ by 0x42F8F0: main (GraphicsCxxTests.cpp:806)
```

```bash
```
CDash Email Notification

A submission to CDash for the project CMake has failing tests.
You have been identified as one of the authors who have checked
in changes that are part of this submission or you are listed in the default contact list.

Details on the submission can be found at http://www.cdash.org/CDash/buildSummary.php?buildid=322849

Project: CMake
Site: destiny.kitware
Build Name: HP-UX-aCC
Build Time: 2009-04-29T14:28:00 EDT
Type: Continuous
Tests failing: 85

*Tests failing* (first 5)
FindPackageTest (http://www.cdash.org/CDash/testDetails.php?test=21959898&build=322849)
FindModulesExecuteAll (http://www.cdash.org/CDash/testDetails.php?test=21959899&build=322849)

CDash on www.cdash.org
Kitware Hosted Cdash
http://www.cdash.org/CDashPublic/

CDash is a web-based software testing server. CDash aggregates, analyzes and displays the results of software testing processes submitted from clients located around the world. Developers depend on CDash to convey the state of a software system, and to continually improve its quality. To learn more about CDash visit the main CDash website.

Starting a project is easy and free, in just a few clicks you can start monitoring the quality of your software development.

Available Dashboards

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
<th>Submissions</th>
<th>First build</th>
<th>Last activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>automoc4</td>
<td>automoc4 is a tool which makes mac processing with Qt easier.</td>
<td>26</td>
<td>2009-04-14T12:33:22 EDT</td>
<td>2009-04-29T14:36:11 EDT</td>
</tr>
<tr>
<td>Avogadro</td>
<td>Avogadro is an advanced molecular editor designed for cross-platform use in computational chemistry, molecular modeling, bioinformatics, materials science, and related areas. It offers flexible rendering and a powerful plugin architecture.</td>
<td>57</td>
<td>2009-04-29T14:01:47 EST</td>
<td>2009-05-01T00:06:53 EDT</td>
</tr>
<tr>
<td>CMakePortes</td>
<td>A collection of popular open-source libraries which can be built using CMake.</td>
<td>266</td>
<td>2009-06-05T20:28:33 EDT</td>
<td>2009-05-01T13:36:47 EDT</td>
</tr>
</tbody>
</table>
CDash testing

- Experimental
- Nightly
- Purify / valgrind
- Coverage (gcov, bullseye)
- Configuration coverage
  - Make sure different OS’s, Libraries and options are covered
- Image difference testing
CMake Tutorial — Step 7
Adding dashboard support

- Demo
  - add dashboard support to tutorial with Cdash
  - Create a new CDash project
  - Save CTestConfig.cmake to project source root
  - Re-build project
  - Submit an Experimental build
Configuration Coverage

- CMake
  - Spaces in paths
  - Network paths
  - Different drive letters and directories
- VTK
  - Mesa
  - OpenGL
  - CMake release
  - Current CMake
CPack
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
CPack Components

- [http://www.cmake.org/Wiki/CMake:Component_Install_With_CPack](http://www.cmake.org/Wiki/CMake:Component_Install_With_CPack)
Using CPack

- On Windows install command line ZIP program, and NSIS
- setup your project to work with cpack
  - set cpack option variables if needed
  - include(CPack)
- Reuses existing project install rules
- Running cpack
  - make package (create all packages)
  - make package_source (create source package)
  - cpack -C CPackConfig.cmake -G NSIS
  - cpack -C CPackConfig.cmake -G ZIP
  - cpack -C CPackSourceConfig.cmake -G ZIP
CPack more information

• Mastering CMake
• http://www.cmake.org/Wiki/CMake:Packaging_With_CPack
Summary

• Build - CMake
• Test - CTest /CDash/KWStyle
• Deploy – CPack

• Links
  • www.kitware.com
  • www.cmake.org
  • www.cdash.org
  • https://svn.boost.org/trac/boost/wiki/CMake
  • bill.hoffman@kitware.com
Thank you!