CTest:Using CTEST and CDASH without CMAKE

From KitwarePublic

This is a short tutorial, which shows how to use CTest scripts to populate a CDash dashboard. It shows how to checkout, configure, compile and test a project with CTest when not using CMake.

As this tutorial is the outcome of a test setup, put together form many different sources, it will and can not be complete. It will only give an **good** overview. Detailed information can be found on the linked wiki pages or the CTest manual (`$ man ctest`).

Contents

Pre-requirements for the example setup

- Project is under subversion control
- No CTest files shall be stored in the project repository
- Use of the autotools build system

Setup files

A set of file is needed to form a complete system.

- steer.cmake
- CTestConfig.cmake
- CTestCustom.cmake
- CTestTestfile.cmake

steer.cmake

The steering script. It can be evoked from the command-line or via scheduled tasks for automatic running.

Execute script

```
$ ctest -S steer.cmake
```

- use **-vv** for extra verbosity during debugging
- see also here for the usage of command line arguments.

Example

The example below intends to be a realistic usecase showing non-trivial usage. All code in this paragraph should be in one file. It is "broken up" only for the purpose of explanation.

The first step is the retrieval of the build environment

Get the hostname of current machine

```
find_program(HOSTNAME_CMD NAMES hostname)
exec_program(${HOSTNAME_CMD} ARGS OUTPUT_VARIABLE HOSTNAME)
set(CTEST_SITE             "${HOSTNAME}")
```

Finds the paths to the program hostname, executes it, stores the result in ${HOSTNAME}. Then the *CTEST_SITE* variable is set, to be used in CDash.

Get the system information of current machine

```
find_program(UNAME NAMES uname)
macro(getuname name flag)
  exec_program("${UNAME}" ARGS "${flag}" OUTPUT_VARIABLE "${name}")
endmacro(getuname)
```

Defines the macro getuname, for easy usage of *uname*

```
getuname(osname -s)
getuname(osrel  -r)
getuname(cpu    -m)
set(CTEST_BUILD_NAME        "${osname}-${cpu}-prod")
```

Sets the build name, to be used in CDash.

Get necessary executables

The CTest command for svn (subversion) is found and set here:

```
find_program(CTEST_SVN_COMMAND NAMES svn)
```

This finds and sets the full path to the executable for make here:

```
find_program(MAKE NAMES make)
```

The build specific environment for the project

```
set(MODEL analysis)
```

Choose the dashboard group to be filled with this script. Not only the standard build groups (Experimental, Continous, Nighlty) can be chosen, but also build groups which can be defined in CDash.

```
set(CTEST_DASHBOARD_ROOT    "$ENV{HOME}/automatedBuild")
set(CTEST_SOURCE_DIRECTORY  "${CTEST_DASHBOARD_ROOT}/src")
set(CTEST_BINARY_DIRECTORY  "${CTEST_SOURCE_DIRECTORY}/build-${CTEST_BUILD_NAME}")
```

Set the source and binary directories of your project.

Commands for the execution of CTest

For an initial checkout, the *CTEST_CHECKOUT_COMMAND* has to be set:

```
set(CTEST_CHECKOUT_COMMAND     "${CTEST_SVN_COMMAND} co http://myRepositoryServer.com/myRepository/trunk
```

The initial checkout is only done if you specify the update command **`ctest_update(SOURCE "${CTEST_SOURCE_DIRECTORY}" RETURN_VALUE res)`** (see further down), which needs also to be configured :

```
set(CTEST_UPDATE_COMMAND              "${CTEST_SVN_COMMAND}")
```

If one wants to have a clean checkout every time, then the update column of the dashboard will always say 0. Because it was all checked out in the initial checkout and not in the update. However, if one uses the svn flag **`-r`** with *yesterdays* date for the initial checkout, the update columns will show the updates which have been done in the last day.

The commands which are executed in *ctest_configure* and *ctest build* have to be specified. Both are executed in the binary directory.

```
set(CTEST_CONFIGURE_COMMAND "${CTEST_SOURCE_DIRECTORY}/configure --enable-optimization=3 --disable-debug
set(CTEST_BUILD_COMMAND     "/usr/bin/make -j16")
```

Configure CTest

If the CTest configuration files CTestConfig.cmake, CTestCustom.cmake and CTestTestfile.cmake should or can not be stored in the repository itself, then they can be copied there automatically by CTest.

```
configure_file($ENV{HOME}/CTestConfiguartion/CTestConfig.cmake  ${CTEST_SOURCE_DIRECTORY}/CTestConfig.cma
configure_file($ENV{HOME}/CTestConfiguartion/CTestCustom.cmake ${CTEST_BINARY_DIRECTORY}/CTestCustom.cmak
configure_file($ENV{HOME}/CTestConfiguartion/CTestTestfile.cmake ${CTEST_BINARY_DIRECTORY}/CTestTestfile.
```

- *CTestConfig.cmake* should be put in the source directory
- *CTestCustom.cmake* and *CTestTestfile.cmake* should be put in the binary directory
- Full paths should be given

Then the custom configuration has to be read in.

```
ctest_read_custom_files("${CTEST_BINARY_DIRECTORY}")
```

General CTest settings

```
set(CTEST_TIMEOUT          "7200")
```

Sets the timeout value for this script in seconds. Here 120 min.

```
set( $ENV{LC_MESSAGES}      "en_EN" )
```

Set the output language to english, so that CTest can analyze it.

Run CTest

Start

```
ctest_start(${MODEL} TRACK ${MODEL} )
```

Starts CTest and assigns the build track to be filled. It creates automatically the binary directory **${CTEST_BINARY_DIRECTORY}**, if not present. Furthermore also the directory **${CTEST_BINARY_DIRECTORY}/Testing** is created, where all the result and log files are stored.

Update / Initial checkout

```
ctest_update(           SOURCE "${CTEST_SOURCE_DIRECTORY}" RETURN_VALUE res)
```

Performs the update from the repository. As the **CTEST_CHECKOUT_COMMAND** is set here, also the initial checkout done.

Configure build system

The *autoreconf* command of the autotools package is not supported by standard CTest, so the process has to be called manually.

```
execute_process(COMMAND "/usr/bin/autoreconf" "-f" "-i" WORKING_DIRECTORY ${CTEST_SOURCE_DIRECTORY} RESUL
        OUTPUT_VARIABLE autoreconfLog ERROR_VARIABLE autoreconfLog)
file(WRITE ${CTEST_BINARY_DIRECTORY}/Testing/autoreconf.log "${autoreconfLog}")
```

Runs the *autoreconf* command stores both stdout and stderr in a log file.

In order to execute the configure/build block only if *autoreconf* succeeded, they can be put into an if statement

```
if( NOT ${autoreconfResult} )
  ... the other build commands ...
endif( NOT ${autoreconfResult} )
```

Configure / Build block

```
ctest_configure(BUILD  "${CTEST_BINARY_DIRECTORY}" RETURN_VALUE res)
```

Configures the project. It is executed in the binary directory.

```
ctest_build(    BUILD  "${CTEST_BINARY_DIRECTORY}" RETURN_VALUE res)
```

Builds the project. It is executed in the binary directory.

```
execute_process(COMMAND "/usr/bin/make install -j 16" "-f" "-i" WORKING_DIRECTORY ${CTEST_BINARY_DIRECTOR
        RESULT_VARIABLE makeInstallResult OUTPUT_VARIABLE makeInstallLog ERROR_VARIABLE makeInstallLog)
        file(WRITE ${CTEST_BINARY_DIRECTORY}/Testing/makeinstall.log "${makeInstallLog}")
```

Like the *autoreconf* command, the install command is not part of the standard test. It has to be invoked manually. Both stdout and stderr are written into a log file.

Launch tests

```
ctest_test(     BUILD  "${CTEST_BINARY_DIRECTORY}" RETURN_VALUE res)
```

Invokes all tests, defined in CTestTestfile.cmake

Finialize the submission

After the all build steps, the results have to be submitted to CDash via

```
ctest_submit(                                           RETURN_VALUE res)
```

This uses the settings of CTestConfig.cmake.

Complete Example

The complete example **steer.cmake** could look like this :

```
# ----------------------------------------------------------
# -- Get environment
# ----------------------------------------------------------

## -- Set hostname
## --------------------------
find_program(HOSTNAME_CMD NAMES hostname)
exec_program(${HOSTNAME_CMD} ARGS OUTPUT_VARIABLE HOSTNAME)

set(CTEST_SITE                          "${HOSTNAME}")

## -- Set site / build name
## --------------------------

find_program(UNAME NAMES uname)
```

```
macro(getuname name flag)
  exec_program("${UNAME}" ARGS "${flag}" OUTPUT_VARIABLE "${name}")
endmacro(getuname)

getuname(osname -s)
getuname(osrel  -r)
getuname(cpu    -m)

set(CTEST_BUILD_NAME                    "${osname}-${cpu}-prod")

## -- SVN command
## ----------------
find_program(CTEST_SVN_COMMAND NAMES svn)

## -- make command
## -----------------
find_program(MAKE NAMES make)

# ---------------------------------------------------------
# -- build specific
# ---------------------------------------------------------

set(MODEL                               "analysis")

## -- DashBoard Root
set(CTEST_DASHBOARD_ROOT                "$ENV{HOME}/automatedBuild")

## -- SRC Dir
set(CTEST_SOURCE_DIRECTORY              "${CTEST_DASHBOARD_ROOT}/src")

## -- BIN Dir
set(CTEST_BINARY_DIRECTORY              "${CTEST_SOURCE_DIRECTORY}/build-${CTEST_BUILD_NAME}")

## -- Build options
set(OPTION_BUILD                        "-j16")

# ---------------------------------------------------------
# -- commands
# ---------------------------------------------------------

## -- Checkout command
if(NOT EXISTS "${CTEST_SOURCE_DIRECTORY}")
        set(CTEST_CHECKOUT_COMMAND     "${CTEST_SVN_COMMAND} co http://myRepositoryServer.com/myRepositor
endif(NOT EXISTS "${CTEST_SOURCE_DIRECTORY}")

## -- Update Command
set(CTEST_UPDATE_COMMAND                "${CTEST_SVN_COMMAND}")

## -- Configure Command
set(CTEST_CONFIGURE_COMMAND             "${CTEST_SOURCE_DIRECTORY}/configure configure --enable-optimizati

## -- Build Command
set(CTEST_BUILD_COMMAND                 "${MAKE} ${OPTION_BUILD}")

# ---------------------------------------------------------
# -- Configure CTest
# ---------------------------------------------------------

## -- CTest Config
configure_file($ENV{HOME}/CTestConfiguration/CTestConfig.cmake  ${CTEST_SOURCE_DIRECTORY}/CTestConfig.cma

## -- CTest Custom
configure_file($ENV{HOME}/CTestConfiguration/CTestCustom.cmake ${CTEST_BINARY_DIRECTORY}/CTestCustom.cmak

## -- CTest Testfile
```

```
configure_file($ENV{HOME}/CTestConfiguration/CTestTestfile.cmake ${CTEST_BINARY_DIRECTORY}/CTestTestfile.

## -- read CTestCustom.cmake file
ctest_read_custom_files("${CTEST_BINARY_DIRECTORY}")

# ----------------------------------------------------------
# -- Settings
# ----------------------------------------------------------

## -- Process timeout in seconds
set(CTEST_TIMEOUT           "7200")

## -- Set output to english
set( $ENV{LC_MESSAGES}      "en_EN" )

# ----------------------------------------------------------
# -- Run CTest
# ----------------------------------------------------------

## -- Start
message(" -- Start dashboard ${MODEL} - ${CTEST_BUILD_NAME} --")
ctest_start(${MODEL} TRACK ${MODEL})

## -- Update
message(" -- Update ${MODEL} - ${CTEST_BUILD_NAME} --")
ctest_update(           SOURCE "${CTEST_SOURCE_DIRECTORY}" RETURN_VALUE res)

## -- Run autoreconf
message(" -- Autoreconf ${MODEL} - ${CTEST_BUILD_NAME} --")
execute_process(COMMAND "/usr/bin/autoreconf" "-f" "-i" WORKING_DIRECTORY ${CTEST_SOURCE_DIRECTORY} RESUL
        OUTPUT_VARIABLE autoreconfLog ERROR_VARIABLE autoreconfLog)
file(WRITE ${CTEST_BINARY_DIRECTORY}/Testing/autoreconf.log "${autoreconfLog}")

if( NOT ${autoreconfResult} )

        ## -- Configure
        message(" -- Configure ${MODEL} - ${CTEST_BUILD_NAME} --")
        ctest_configure(BUILD  "${CTEST_BINARY_DIRECTORY}" RETURN_VALUE res)

        ## -- BUILD
        message(" -- Build ${MODEL} - ${CTEST_BUILD_NAME} --")
        ctest_build(    BUILD  "${CTEST_BINARY_DIRECTORY}" RETURN_VALUE res)

        ## -- INSTALL
        message(" -- Install ${MODEL} - ${CTEST_BUILD_NAME} --")
        execute_process(COMMAND "${MAKE} install ${OPTION_BUILD}" WORKING_DIRECTORY ${CTEST_BINARY_DIRECT
                RESULT_VARIABLE makeInstallResult OUTPUT_VARIABLE makeInstallLog ERROR_VARIABLE makeInsta
        file(WRITE ${CTEST_BINARY_DIRECTORY}/Testing/makeinstall.log "${makeInstallLog}")

        ## -- TEST
        message(" -- Test ${MODEL} - ${CTEST_BUILD_NAME} --")
        ctest_test(     BUILD  "${CTEST_BINARY_DIRECTORY}" RETURN_VALUE res)

endif( NOT ${autoreconfResult} )

## -- SUBMIT
message(" -- Submit ${MODEL} - ${CTEST_BUILD_NAME} --")
ctest_submit(                                              RETURN_VALUE res)

message(" -- Finished ${MODEL}  - ${CTEST_BUILD_NAME} --")
```

CTestConfig.cmake

This file sets the tranfer parameters that CTest will use when submitting to CDash.

```
ctest_submit(                                           RETURN_VALUE res)
```

Several different ways exist to submit results to the dashboard like *http*, *ftp*, *scp* and *XML-RPC*. However only the *http* case shall be discussed here. Details can be found an the |CTest Submission Issues page.

Example

This file can be also downloaded from CDash after creating a new project.

```
set(CTEST_PROJECT_NAME         "Test-Project")
set(CTEST_NIGHTLY_START_TIME   "01:00:00 CET")
set(CTEST_DROP_METHOD          "http")
set(CTEST_DROP_SITE            "myTestNode.myNetwork.com")
set(CTEST_DROP_LOCATION        "/cdash/submit.php?project=TestProject")
set(CTEST_DROP_SITE_CDASH      TRUE)
```

CTestCustom.cmake

This file is used to customize CTest. A detailed description of all options, can be found here. In order to be read in properly this file has to be placed in the binary directory **${CTEST_BINARY_DIRECTORY}**.

It can also be read from any other location by specifying

```
ctest_read_custom_files("<PATH TO CTestCustom.cmake FILE>")
```

Examples

This are only some examples of customization. The full list and detailed description can be found here.

- Change the maximum numbers of warnings

```
set(CTEST_CUSTOM_MAXIMUM_NUMBER_OF_WARNINGS "500" )
```

- Change the maximum numbers of errors

```
set(CTEST_CUSTOM_MAXIMUM_NUMBER_OF_ERRORS   "200" )
```

- Add *fake warnings* to a list of exceptions

```
set(CTEST_CUSTOM_WARNING_EXCEPTION
       ${CTEST_CUSTOM_WARNING_EXCEPTION}

       # -- doxygen warnings
       "are not documented:"
       "Skipping documentation"
       )
```

Every regular expression can be added as exception.

CTestTestfile.cmake

This file specifies the test which should be executed in

```
ctest_test(     BUILD  "${CTEST_BINARY_DIRECTORY}" RETURN_VALUE res)
```

The file should be placed in the binary directory **`${CTEST_BINARY_DIRECTORY}`**.

Examples

Each test consists of:

- The unique name of the test (eg.: **`testname1`**)
- The full path to the executable of the test (eg.: **`"$ENV{HOME}/bin/TEST_EXECUTABLE_1.sh"`**)
- A List of arguments to the executable (eg.: **`"ARGUMENT_1"`** **`"ARGUMENT_2"`** etc.)

```
ADD_TEST(testname1 "$ENV{HOME}/bin/TEST_EXECUTABLE_1.sh" "ARGUMENT_1")
ADD_TEST(testname2 "$ENV{HOME}/bin/TEST_EXECUTABLE_2.sh")
```

Retrieved from "https://www.paraview.org/Wiki/index.php?title=CTest:Using_CTEST_and_CDASH_without_CMAKE&oldid=48839"
