########################################################################

# Note: CMake support is community-based. The maintainers do not use CMake

# internally.

#

# CMake build script for Google Test.

#

# To run the tests for Google Test itself on Linux, use 'make test' or

# ctest. You can select which tests to run using 'ctest -R regex'.

# For more options, run 'ctest --help'.

# When other libraries are using a shared version of runtime libraries,

# Google Test also has to use one.

option(

gtest\_force\_shared\_crt

"Use shared (DLL) run-time lib even when Google Test is built as static lib."

OFF)

option(gtest\_build\_tests "Build all of gtest's own tests." OFF)

option(gtest\_build\_samples "Build gtest's sample programs." OFF)

option(gtest\_disable\_pthreads "Disable uses of pthreads in gtest." OFF)

option(

gtest\_hide\_internal\_symbols

"Build gtest with internal symbols hidden in shared libraries."

OFF)

# Defines pre\_project\_set\_up\_hermetic\_build() and set\_up\_hermetic\_build().

include(cmake/hermetic\_build.cmake OPTIONAL)

if (COMMAND pre\_project\_set\_up\_hermetic\_build)

pre\_project\_set\_up\_hermetic\_build()

endif()

########################################################################

#

# Project-wide settings

# Name of the project.

#

# CMake files in this project can refer to the root source directory

# as ${gtest\_SOURCE\_DIR} and to the root binary directory as

# ${gtest\_BINARY\_DIR}.

# Language "C" is required for find\_package(Threads).

# Project version:

if (CMAKE\_VERSION VERSION\_LESS 3.0)

project(gtest CXX C)

set(PROJECT\_VERSION ${GOOGLETEST\_VERSION})

else()

cmake\_policy(SET CMP0048 NEW)

project(gtest VERSION ${GOOGLETEST\_VERSION} LANGUAGES CXX C)

endif()

cmake\_minimum\_required(VERSION 2.6.4)

if (POLICY CMP0063) # Visibility

cmake\_policy(SET CMP0063 NEW)

endif (POLICY CMP0063)

if (COMMAND set\_up\_hermetic\_build)

set\_up\_hermetic\_build()

endif()

# These commands only run if this is the main project

if(CMAKE\_PROJECT\_NAME STREQUAL "gtest" OR CMAKE\_PROJECT\_NAME STREQUAL "googletest-distribution")

# BUILD\_SHARED\_LIBS is a standard CMake variable, but we declare it here to

# make it prominent in the GUI.

option(BUILD\_SHARED\_LIBS "Build shared libraries (DLLs)." OFF)

else()

mark\_as\_advanced(

gtest\_force\_shared\_crt

gtest\_build\_tests

gtest\_build\_samples

gtest\_disable\_pthreads

gtest\_hide\_internal\_symbols)

endif()

if (gtest\_hide\_internal\_symbols)

set(CMAKE\_CXX\_VISIBILITY\_PRESET hidden)

set(CMAKE\_VISIBILITY\_INLINES\_HIDDEN 1)

endif()

# Define helper functions and macros used by Google Test.

include(cmake/internal\_utils.cmake)

config\_compiler\_and\_linker() # Defined in internal\_utils.cmake.

# Create the CMake package file descriptors.

if (INSTALL\_GTEST)

include(CMakePackageConfigHelpers)

set(cmake\_package\_name GTest)

set(targets\_export\_name ${cmake\_package\_name}Targets CACHE INTERNAL "")

set(generated\_dir "${CMAKE\_CURRENT\_BINARY\_DIR}/generated" CACHE INTERNAL "")

set(cmake\_files\_install\_dir "${CMAKE\_INSTALL\_LIBDIR}/cmake/${cmake\_package\_name}")

set(version\_file "${generated\_dir}/${cmake\_package\_name}ConfigVersion.cmake")

write\_basic\_package\_version\_file(${version\_file} VERSION ${GOOGLETEST\_VERSION} COMPATIBILITY AnyNewerVersion)

install(EXPORT ${targets\_export\_name}

NAMESPACE ${cmake\_package\_name}::

DESTINATION ${cmake\_files\_install\_dir})

set(config\_file "${generated\_dir}/${cmake\_package\_name}Config.cmake")

configure\_package\_config\_file("${gtest\_SOURCE\_DIR}/cmake/Config.cmake.in"

"${config\_file}" INSTALL\_DESTINATION ${cmake\_files\_install\_dir})

install(FILES ${version\_file} ${config\_file}

DESTINATION ${cmake\_files\_install\_dir})

endif()

# Where Google Test's .h files can be found.

set(gtest\_build\_include\_dirs

"${gtest\_SOURCE\_DIR}/include"

"${gtest\_SOURCE\_DIR}")

include\_directories(${gtest\_build\_include\_dirs})

########################################################################

#

# Defines the gtest & gtest\_main libraries. User tests should link

# with one of them.

# Google Test libraries. We build them using more strict warnings than what

# are used for other targets, to ensure that gtest can be compiled by a user

# aggressive about warnings.

cxx\_library(gtest "${cxx\_strict}" src/gtest-all.cc)

cxx\_library(gtest\_main "${cxx\_strict}" src/gtest\_main.cc)

# If the CMake version supports it, attach header directory information

# to the targets for when we are part of a parent build (ie being pulled

# in via add\_subdirectory() rather than being a standalone build).

if (DEFINED CMAKE\_VERSION AND NOT "${CMAKE\_VERSION}" VERSION\_LESS "2.8.11")

target\_include\_directories(gtest SYSTEM INTERFACE

"$<BUILD\_INTERFACE:${gtest\_build\_include\_dirs}>"

"$<INSTALL\_INTERFACE:$<INSTALL\_PREFIX>/${CMAKE\_INSTALL\_INCLUDEDIR}>")

target\_include\_directories(gtest\_main SYSTEM INTERFACE

"$<BUILD\_INTERFACE:${gtest\_build\_include\_dirs}>"

"$<INSTALL\_INTERFACE:$<INSTALL\_PREFIX>/${CMAKE\_INSTALL\_INCLUDEDIR}>")

endif()

target\_link\_libraries(gtest\_main PUBLIC gtest)

########################################################################

#

# Install rules

install\_project(gtest gtest\_main)

########################################################################

#

# Samples on how to link user tests with gtest or gtest\_main.

#

# They are not built by default. To build them, set the

# gtest\_build\_samples option to ON. You can do it by running ccmake

# or specifying the -Dgtest\_build\_samples=ON flag when running cmake.

if (gtest\_build\_samples)

cxx\_executable(sample1\_unittest samples gtest\_main samples/sample1.cc)

cxx\_executable(sample2\_unittest samples gtest\_main samples/sample2.cc)

cxx\_executable(sample3\_unittest samples gtest\_main)

cxx\_executable(sample4\_unittest samples gtest\_main samples/sample4.cc)

cxx\_executable(sample5\_unittest samples gtest\_main samples/sample1.cc)

cxx\_executable(sample6\_unittest samples gtest\_main)

cxx\_executable(sample7\_unittest samples gtest\_main)

cxx\_executable(sample8\_unittest samples gtest\_main)

cxx\_executable(sample9\_unittest samples gtest)

cxx\_executable(sample10\_unittest samples gtest)

endif()

########################################################################

#

# Google Test's own tests.

#

# You can skip this section if you aren't interested in testing

# Google Test itself.

#

# The tests are not built by default. To build them, set the

# gtest\_build\_tests option to ON. You can do it by running ccmake

# or specifying the -Dgtest\_build\_tests=ON flag when running cmake.

if (gtest\_build\_tests)

# This must be set in the root directory for the tests to be run by

# 'make test' or ctest.

enable\_testing()

if (WIN32)

file(GENERATE OUTPUT "${CMAKE\_CURRENT\_BINARY\_DIR}/$<CONFIG>/RunTest.ps1"

CONTENT

"$project\_bin = \"${CMAKE\_BINARY\_DIR}/bin/$<CONFIG>\"

$env:Path = \"$project\_bin;$env:Path\"

& $args")

elseif (MINGW OR CYGWIN)

file(GENERATE OUTPUT "${CMAKE\_CURRENT\_BINARY\_DIR}/RunTest.ps1"

CONTENT

"$project\_bin = (cygpath --windows ${CMAKE\_BINARY\_DIR}/bin)

$env:Path = \"$project\_bin;$env:Path\"

& $args")

endif()

############################################################

# C++ tests built with standard compiler flags.

cxx\_test(googletest-death-test-test gtest\_main)

cxx\_test(gtest\_environment\_test gtest)

cxx\_test(googletest-filepath-test gtest\_main)

cxx\_test(googletest-listener-test gtest\_main)

cxx\_test(gtest\_main\_unittest gtest\_main)

cxx\_test(googletest-message-test gtest\_main)

cxx\_test(gtest\_no\_test\_unittest gtest)

cxx\_test(googletest-options-test gtest\_main)

cxx\_test(googletest-param-test-test gtest

test/googletest-param-test2-test.cc)

cxx\_test(googletest-port-test gtest\_main)

cxx\_test(gtest\_pred\_impl\_unittest gtest\_main)

cxx\_test(gtest\_premature\_exit\_test gtest

test/gtest\_premature\_exit\_test.cc)

cxx\_test(googletest-printers-test gtest\_main)

cxx\_test(gtest\_prod\_test gtest\_main

test/production.cc)

cxx\_test(gtest\_repeat\_test gtest)

cxx\_test(gtest\_sole\_header\_test gtest\_main)

cxx\_test(gtest\_stress\_test gtest)

cxx\_test(googletest-test-part-test gtest\_main)

cxx\_test(gtest\_throw\_on\_failure\_ex\_test gtest)

cxx\_test(gtest-typed-test\_test gtest\_main

test/gtest-typed-test2\_test.cc)

cxx\_test(gtest\_unittest gtest\_main)

cxx\_test(gtest-unittest-api\_test gtest)

cxx\_test(gtest\_skip\_in\_environment\_setup\_test gtest\_main)

cxx\_test(gtest\_skip\_test gtest\_main)

############################################################

# C++ tests built with non-standard compiler flags.

# MSVC 7.1 does not support STL with exceptions disabled.

if (NOT MSVC OR MSVC\_VERSION GREATER 1310)

cxx\_library(gtest\_no\_exception "${cxx\_no\_exception}"

src/gtest-all.cc)

cxx\_library(gtest\_main\_no\_exception "${cxx\_no\_exception}"

src/gtest-all.cc src/gtest\_main.cc)

endif()

cxx\_library(gtest\_main\_no\_rtti "${cxx\_no\_rtti}"

src/gtest-all.cc src/gtest\_main.cc)

cxx\_test\_with\_flags(gtest-death-test\_ex\_nocatch\_test

"${cxx\_exception} -DGTEST\_ENABLE\_CATCH\_EXCEPTIONS\_=0"

gtest test/googletest-death-test\_ex\_test.cc)

cxx\_test\_with\_flags(gtest-death-test\_ex\_catch\_test

"${cxx\_exception} -DGTEST\_ENABLE\_CATCH\_EXCEPTIONS\_=1"

gtest test/googletest-death-test\_ex\_test.cc)

cxx\_test\_with\_flags(gtest\_no\_rtti\_unittest "${cxx\_no\_rtti}"

gtest\_main\_no\_rtti test/gtest\_unittest.cc)

cxx\_shared\_library(gtest\_dll "${cxx\_default}"

src/gtest-all.cc src/gtest\_main.cc)

cxx\_executable\_with\_flags(gtest\_dll\_test\_ "${cxx\_default}"

gtest\_dll test/gtest\_all\_test.cc)

set\_target\_properties(gtest\_dll\_test\_

PROPERTIES

COMPILE\_DEFINITIONS "GTEST\_LINKED\_AS\_SHARED\_LIBRARY=1")

############################################################

# Python tests.

cxx\_executable(googletest-break-on-failure-unittest\_ test gtest)

py\_test(googletest-break-on-failure-unittest)

py\_test(gtest\_skip\_check\_output\_test)

py\_test(gtest\_skip\_environment\_check\_output\_test)

# Visual Studio .NET 2003 does not support STL with exceptions disabled.

if (NOT MSVC OR MSVC\_VERSION GREATER 1310) # 1310 is Visual Studio .NET 2003

cxx\_executable\_with\_flags(

googletest-catch-exceptions-no-ex-test\_

"${cxx\_no\_exception}"

gtest\_main\_no\_exception

test/googletest-catch-exceptions-test\_.cc)

endif()

cxx\_executable\_with\_flags(

googletest-catch-exceptions-ex-test\_

"${cxx\_exception}"

gtest\_main

test/googletest-catch-exceptions-test\_.cc)

py\_test(googletest-catch-exceptions-test)

cxx\_executable(googletest-color-test\_ test gtest)

py\_test(googletest-color-test)

cxx\_executable(googletest-env-var-test\_ test gtest)

py\_test(googletest-env-var-test)

cxx\_executable(googletest-filter-unittest\_ test gtest)

py\_test(googletest-filter-unittest)

cxx\_executable(gtest\_help\_test\_ test gtest\_main)

py\_test(gtest\_help\_test)

cxx\_executable(googletest-list-tests-unittest\_ test gtest)

py\_test(googletest-list-tests-unittest)

cxx\_executable(googletest-output-test\_ test gtest)

py\_test(googletest-output-test --no\_stacktrace\_support)

cxx\_executable(googletest-shuffle-test\_ test gtest)

py\_test(googletest-shuffle-test)

# MSVC 7.1 does not support STL with exceptions disabled.

if (NOT MSVC OR MSVC\_VERSION GREATER 1310)

cxx\_executable(googletest-throw-on-failure-test\_ test gtest\_no\_exception)

set\_target\_properties(googletest-throw-on-failure-test\_

PROPERTIES

COMPILE\_FLAGS "${cxx\_no\_exception}")

py\_test(googletest-throw-on-failure-test)

endif()

cxx\_executable(googletest-uninitialized-test\_ test gtest)

py\_test(googletest-uninitialized-test)

cxx\_executable(gtest\_xml\_outfile1\_test\_ test gtest\_main)

cxx\_executable(gtest\_xml\_outfile2\_test\_ test gtest\_main)

py\_test(gtest\_xml\_outfiles\_test)

py\_test(googletest-json-outfiles-test)

cxx\_executable(gtest\_xml\_output\_unittest\_ test gtest)

py\_test(gtest\_xml\_output\_unittest --no\_stacktrace\_support)

py\_test(googletest-json-output-unittest --no\_stacktrace\_support)

endif()