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

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

# internally.

#

# CMake build script for Google Mock.

#

# To run the tests for Google Mock 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'.

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

# A directory to find Google Test sources.

if (EXISTS "${CMAKE\_CURRENT\_SOURCE\_DIR}/gtest/CMakeLists.txt")

set(gtest\_dir gtest)

else()

set(gtest\_dir ../googletest)

endif()

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

include("${gtest\_dir}/cmake/hermetic\_build.cmake" OPTIONAL)

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

# Google Test also calls hermetic setup functions from add\_subdirectory,

# although its changes will not affect things at the current scope.

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

# ${gmock\_BINARY\_DIR}.

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

if (CMAKE\_VERSION VERSION\_LESS 3.0)

project(gmock CXX C)

else()

cmake\_policy(SET CMP0048 NEW)

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

endif()

cmake\_minimum\_required(VERSION 2.6.4)

if (COMMAND set\_up\_hermetic\_build)

set\_up\_hermetic\_build()

endif()

# Instructs CMake to process Google Test's CMakeLists.txt and add its

# targets to the current scope. We are placing Google Test's binary

# directory in a subdirectory of our own as VC compilation may break

# if they are the same (the default).

add\_subdirectory("${gtest\_dir}" "${gmock\_BINARY\_DIR}/${gtest\_dir}")

# These commands only run if this is the main project

if(CMAKE\_PROJECT\_NAME STREQUAL "gmock" 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(gmock\_build\_tests)

endif()

# Although Google Test's CMakeLists.txt calls this function, the

# changes there don't affect the current scope. Therefore we have to

# call it again here.

config\_compiler\_and\_linker() # from ${gtest\_dir}/cmake/internal\_utils.cmake

# Adds Google Mock's and Google Test's header directories to the search path.

set(gmock\_build\_include\_dirs

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

"${gmock\_SOURCE\_DIR}"

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

# This directory is needed to build directly from Google Test sources.

"${gtest\_SOURCE\_DIR}")

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

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

#

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

# with one of them.

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

# are used for other targets, to ensure that Google Mock can be compiled by

# a user aggressive about warnings.

if (MSVC)

cxx\_library(gmock

"${cxx\_strict}"

"${gtest\_dir}/src/gtest-all.cc"

src/gmock-all.cc)

cxx\_library(gmock\_main

"${cxx\_strict}"

"${gtest\_dir}/src/gtest-all.cc"

src/gmock-all.cc

src/gmock\_main.cc)

else()

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

target\_link\_libraries(gmock PUBLIC gtest)

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

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

endif()

# 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(gmock SYSTEM INTERFACE

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

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

target\_include\_directories(gmock\_main SYSTEM INTERFACE

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

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

endif()

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

#

# Install rules

install\_project(gmock gmock\_main)

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

#

# Google Mock's own tests.

#

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

# Google Mock itself.

#

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

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

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

if (gmock\_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()

if (MINGW OR CYGWIN)

if (CMAKE\_VERSION VERSION\_LESS "2.8.12")

add\_compile\_options("-Wa,-mbig-obj")

else()

add\_definitions("-Wa,-mbig-obj")

endif()

endif()

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

# C++ tests built with standard compiler flags.

cxx\_test(gmock-actions\_test gmock\_main)

cxx\_test(gmock-cardinalities\_test gmock\_main)

cxx\_test(gmock\_ex\_test gmock\_main)

cxx\_test(gmock-function-mocker\_test gmock\_main)

cxx\_test(gmock-generated-actions\_test gmock\_main)

cxx\_test(gmock-generated-matchers\_test gmock\_main)

cxx\_test(gmock-internal-utils\_test gmock\_main)

cxx\_test(gmock-matchers\_test gmock\_main)

cxx\_test(gmock-more-actions\_test gmock\_main)

cxx\_test(gmock-nice-strict\_test gmock\_main)

cxx\_test(gmock-port\_test gmock\_main)

cxx\_test(gmock-spec-builders\_test gmock\_main)

cxx\_test(gmock\_link\_test gmock\_main test/gmock\_link2\_test.cc)

cxx\_test(gmock\_test gmock\_main)

if (DEFINED GTEST\_HAS\_PTHREAD)

cxx\_test(gmock\_stress\_test gmock)

endif()

# gmock\_all\_test is commented to save time building and running tests.

# Uncomment if necessary.

# cxx\_test(gmock\_all\_test gmock\_main)

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

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

if (MSVC)

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

"${gtest\_dir}/src/gtest-all.cc" src/gmock-all.cc src/gmock\_main.cc)

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

"${gtest\_dir}/src/gtest-all.cc" src/gmock-all.cc src/gmock\_main.cc)

else()

cxx\_library(gmock\_main\_no\_exception "${cxx\_no\_exception}" src/gmock\_main.cc)

target\_link\_libraries(gmock\_main\_no\_exception PUBLIC gmock)

cxx\_library(gmock\_main\_no\_rtti "${cxx\_no\_rtti}" src/gmock\_main.cc)

target\_link\_libraries(gmock\_main\_no\_rtti PUBLIC gmock)

endif()

cxx\_test\_with\_flags(gmock-more-actions\_no\_exception\_test "${cxx\_no\_exception}"

gmock\_main\_no\_exception test/gmock-more-actions\_test.cc)

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

gmock\_main\_no\_rtti test/gmock-spec-builders\_test.cc)

cxx\_shared\_library(shared\_gmock\_main "${cxx\_default}"

"${gtest\_dir}/src/gtest-all.cc" src/gmock-all.cc src/gmock\_main.cc)

# Tests that a binary can be built with Google Mock as a shared library. On

# some system configurations, it may not possible to run the binary without

# knowing more details about the system configurations. We do not try to run

# this binary. To get a more robust shared library coverage, configure with

# -DBUILD\_SHARED\_LIBS=ON.

cxx\_executable\_with\_flags(shared\_gmock\_test\_ "${cxx\_default}"

shared\_gmock\_main test/gmock-spec-builders\_test.cc)

set\_target\_properties(shared\_gmock\_test\_

PROPERTIES

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

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

# Python tests.

cxx\_executable(gmock\_leak\_test\_ test gmock\_main)

py\_test(gmock\_leak\_test)

cxx\_executable(gmock\_output\_test\_ test gmock)

py\_test(gmock\_output\_test)

endif()