cmake\_minimum\_required(VERSION 2.4.4)

set(CMAKE\_ALLOW\_LOOSE\_LOOP\_CONSTRUCTS ON)

project(zlib C)

set(VERSION "1.2.11")

option(ASM686 "Enable building i686 assembly implementation")

option(AMD64 "Enable building amd64 assembly implementation")

set(INSTALL\_BIN\_DIR "${CMAKE\_INSTALL\_PREFIX}/bin" CACHE PATH "Installation directory for executables")

set(INSTALL\_LIB\_DIR "${CMAKE\_INSTALL\_PREFIX}/lib" CACHE PATH "Installation directory for libraries")

set(INSTALL\_INC\_DIR "${CMAKE\_INSTALL\_PREFIX}/include" CACHE PATH "Installation directory for headers")

set(INSTALL\_MAN\_DIR "${CMAKE\_INSTALL\_PREFIX}/share/man" CACHE PATH "Installation directory for manual pages")

set(INSTALL\_PKGCONFIG\_DIR "${CMAKE\_INSTALL\_PREFIX}/share/pkgconfig" CACHE PATH "Installation directory for pkgconfig (.pc) files")

include(CheckTypeSize)

include(CheckFunctionExists)

include(CheckIncludeFile)

include(CheckCSourceCompiles)

enable\_testing()

check\_include\_file(sys/types.h HAVE\_SYS\_TYPES\_H)

check\_include\_file(stdint.h HAVE\_STDINT\_H)

check\_include\_file(stddef.h HAVE\_STDDEF\_H)

#

# Check to see if we have large file support

#

set(CMAKE\_REQUIRED\_DEFINITIONS -D\_LARGEFILE64\_SOURCE=1)

# We add these other definitions here because CheckTypeSize.cmake

# in CMake 2.4.x does not automatically do so and we want

# compatibility with CMake 2.4.x.

if(HAVE\_SYS\_TYPES\_H)

list(APPEND CMAKE\_REQUIRED\_DEFINITIONS -DHAVE\_SYS\_TYPES\_H)

endif()

if(HAVE\_STDINT\_H)

list(APPEND CMAKE\_REQUIRED\_DEFINITIONS -DHAVE\_STDINT\_H)

endif()

if(HAVE\_STDDEF\_H)

list(APPEND CMAKE\_REQUIRED\_DEFINITIONS -DHAVE\_STDDEF\_H)

endif()

check\_type\_size(off64\_t OFF64\_T)

if(HAVE\_OFF64\_T)

add\_definitions(-D\_LARGEFILE64\_SOURCE=1)

endif()

set(CMAKE\_REQUIRED\_DEFINITIONS) # clear variable

#

# Check for fseeko

#

check\_function\_exists(fseeko HAVE\_FSEEKO)

if(NOT HAVE\_FSEEKO)

add\_definitions(-DNO\_FSEEKO)

endif()

#

# Check for unistd.h

#

check\_include\_file(unistd.h Z\_HAVE\_UNISTD\_H)

if(MSVC)

set(CMAKE\_DEBUG\_POSTFIX "d")

add\_definitions(-D\_CRT\_SECURE\_NO\_DEPRECATE)

add\_definitions(-D\_CRT\_NONSTDC\_NO\_DEPRECATE)

include\_directories(${CMAKE\_CURRENT\_SOURCE\_DIR})

endif()

if(NOT CMAKE\_CURRENT\_SOURCE\_DIR STREQUAL CMAKE\_CURRENT\_BINARY\_DIR)

# If we're doing an out of source build and the user has a zconf.h

# in their source tree...

if(EXISTS ${CMAKE\_CURRENT\_SOURCE\_DIR}/zconf.h)

message(STATUS "Renaming")

message(STATUS " ${CMAKE\_CURRENT\_SOURCE\_DIR}/zconf.h")

message(STATUS "to 'zconf.h.included' because this file is included with zlib")

message(STATUS "but CMake generates it automatically in the build directory.")

file(RENAME ${CMAKE\_CURRENT\_SOURCE\_DIR}/zconf.h ${CMAKE\_CURRENT\_SOURCE\_DIR}/zconf.h.included)

endif()

endif()

set(ZLIB\_PC ${CMAKE\_CURRENT\_BINARY\_DIR}/zlib.pc)

configure\_file( ${CMAKE\_CURRENT\_SOURCE\_DIR}/zlib.pc.cmakein

${ZLIB\_PC} @ONLY)

configure\_file( ${CMAKE\_CURRENT\_SOURCE\_DIR}/zconf.h.cmakein

${CMAKE\_CURRENT\_BINARY\_DIR}/zconf.h @ONLY)

include\_directories(${CMAKE\_CURRENT\_BINARY\_DIR} ${CMAKE\_SOURCE\_DIR})

#============================================================================

# zlib

#============================================================================

set(ZLIB\_PUBLIC\_HDRS

${CMAKE\_CURRENT\_BINARY\_DIR}/zconf.h

zlib.h

)

set(ZLIB\_PRIVATE\_HDRS

crc32.h

deflate.h

gzguts.h

inffast.h

inffixed.h

inflate.h

inftrees.h

trees.h

zutil.h

)

set(ZLIB\_SRCS

adler32.c

compress.c

crc32.c

deflate.c

gzclose.c

gzlib.c

gzread.c

gzwrite.c

inflate.c

infback.c

inftrees.c

inffast.c

trees.c

uncompr.c

zutil.c

)

if(NOT MINGW)

set(ZLIB\_DLL\_SRCS

win32/zlib1.rc # If present will override custom build rule below.

)

endif()

if(CMAKE\_COMPILER\_IS\_GNUCC)

if(ASM686)

set(ZLIB\_ASMS contrib/asm686/match.S)

elseif (AMD64)

set(ZLIB\_ASMS contrib/amd64/amd64-match.S)

endif ()

if(ZLIB\_ASMS)

add\_definitions(-DASMV)

set\_source\_files\_properties(${ZLIB\_ASMS} PROPERTIES LANGUAGE C COMPILE\_FLAGS -DNO\_UNDERLINE)

endif()

endif()

if(MSVC)

if(ASM686)

ENABLE\_LANGUAGE(ASM\_MASM)

set(ZLIB\_ASMS

contrib/masmx86/inffas32.asm

contrib/masmx86/match686.asm

)

elseif (AMD64)

ENABLE\_LANGUAGE(ASM\_MASM)

set(ZLIB\_ASMS

contrib/masmx64/gvmat64.asm

contrib/masmx64/inffasx64.asm

)

endif()

if(ZLIB\_ASMS)

add\_definitions(-DASMV -DASMINF)

endif()

endif()

# parse the full version number from zlib.h and include in ZLIB\_FULL\_VERSION

file(READ ${CMAKE\_CURRENT\_SOURCE\_DIR}/zlib.h \_zlib\_h\_contents)

string(REGEX REPLACE ".\*#define[ \t]+ZLIB\_VERSION[ \t]+\"([-0-9A-Za-z.]+)\".\*"

"\\1" ZLIB\_FULL\_VERSION ${\_zlib\_h\_contents})

if(MINGW)

# This gets us DLL resource information when compiling on MinGW.

if(NOT CMAKE\_RC\_COMPILER)

set(CMAKE\_RC\_COMPILER windres.exe)

endif()

add\_custom\_command(OUTPUT ${CMAKE\_CURRENT\_BINARY\_DIR}/zlib1rc.obj

COMMAND ${CMAKE\_RC\_COMPILER}

-D GCC\_WINDRES

-I ${CMAKE\_CURRENT\_SOURCE\_DIR}

-I ${CMAKE\_CURRENT\_BINARY\_DIR}

-o ${CMAKE\_CURRENT\_BINARY\_DIR}/zlib1rc.obj

-i ${CMAKE\_CURRENT\_SOURCE\_DIR}/win32/zlib1.rc)

set(ZLIB\_DLL\_SRCS ${CMAKE\_CURRENT\_BINARY\_DIR}/zlib1rc.obj)

endif(MINGW)

add\_library(zlib SHARED ${ZLIB\_SRCS} ${ZLIB\_ASMS} ${ZLIB\_DLL\_SRCS} ${ZLIB\_PUBLIC\_HDRS} ${ZLIB\_PRIVATE\_HDRS})

add\_library(zlibstatic STATIC ${ZLIB\_SRCS} ${ZLIB\_ASMS} ${ZLIB\_PUBLIC\_HDRS} ${ZLIB\_PRIVATE\_HDRS})

set\_target\_properties(zlib PROPERTIES DEFINE\_SYMBOL ZLIB\_DLL)

set\_target\_properties(zlib PROPERTIES SOVERSION 1)

if(NOT CYGWIN)

# This property causes shared libraries on Linux to have the full version

# encoded into their final filename. We disable this on Cygwin because

# it causes cygz-${ZLIB\_FULL\_VERSION}.dll to be created when cygz.dll

# seems to be the default.

#

# This has no effect with MSVC, on that platform the version info for

# the DLL comes from the resource file win32/zlib1.rc

set\_target\_properties(zlib PROPERTIES VERSION ${ZLIB\_FULL\_VERSION})

endif()

if(UNIX)

# On unix-like platforms the library is almost always called libz

set\_target\_properties(zlib zlibstatic PROPERTIES OUTPUT\_NAME z)

if(NOT APPLE)

set\_target\_properties(zlib PROPERTIES LINK\_FLAGS "-Wl,--version-script,\"${CMAKE\_CURRENT\_SOURCE\_DIR}/zlib.map\"")

endif()

elseif(BUILD\_SHARED\_LIBS AND WIN32)

# Creates zlib1.dll when building shared library version

set\_target\_properties(zlib PROPERTIES SUFFIX "1.dll")

endif()

if(NOT SKIP\_INSTALL\_LIBRARIES AND NOT SKIP\_INSTALL\_ALL )

install(TARGETS zlib zlibstatic

RUNTIME DESTINATION "${INSTALL\_BIN\_DIR}"

ARCHIVE DESTINATION "${INSTALL\_LIB\_DIR}"

LIBRARY DESTINATION "${INSTALL\_LIB\_DIR}" )

endif()

if(NOT SKIP\_INSTALL\_HEADERS AND NOT SKIP\_INSTALL\_ALL )

install(FILES ${ZLIB\_PUBLIC\_HDRS} DESTINATION "${INSTALL\_INC\_DIR}")

endif()

if(NOT SKIP\_INSTALL\_FILES AND NOT SKIP\_INSTALL\_ALL )

install(FILES zlib.3 DESTINATION "${INSTALL\_MAN\_DIR}/man3")

endif()

if(NOT SKIP\_INSTALL\_FILES AND NOT SKIP\_INSTALL\_ALL )

install(FILES ${ZLIB\_PC} DESTINATION "${INSTALL\_PKGCONFIG\_DIR}")

endif()

#============================================================================

# Example binaries

#============================================================================

add\_executable(example test/example.c)

target\_link\_libraries(example zlib)

add\_test(example example)

add\_executable(minigzip test/minigzip.c)

target\_link\_libraries(minigzip zlib)

if(HAVE\_OFF64\_T)

add\_executable(example64 test/example.c)

target\_link\_libraries(example64 zlib)

set\_target\_properties(example64 PROPERTIES COMPILE\_FLAGS "-D\_FILE\_OFFSET\_BITS=64")

add\_test(example64 example64)

add\_executable(minigzip64 test/minigzip.c)

target\_link\_libraries(minigzip64 zlib)

set\_target\_properties(minigzip64 PROPERTIES COMPILE\_FLAGS "-D\_FILE\_OFFSET\_BITS=64")

endif()