# Distributed under the OSI-approved BSD 3-Clause License. See accompanying

# file Copyright.txt or https://cmake.org/licensing for details.

cmake\_minimum\_required(VERSION ${CMAKE\_VERSION})

project(FortranCInterface C Fortran)

include(${FortranCInterface\_BINARY\_DIR}/Input.cmake OPTIONAL)

# Check if the C compiler supports '$' in identifiers.

include(CheckSourceCompiles)

check\_source\_compiles(C

"extern int dollar$(void);

int main() { return 0; }"

C\_SUPPORTS\_DOLLAR)

# List manglings of global symbol names to try.

set(global\_symbols

my\_sub # VisualAge

my\_sub\_ # GNU, Intel, HP, SunPro, PGI

my\_sub\_\_ # GNU g77

MY\_SUB # Intel on Windows

mysub # VisualAge

mysub\_ # GNU, Intel, HP, SunPro, PGI

MYSUB # Intel on Windows

${FortranCInterface\_GLOBAL\_SYMBOLS}

)

list(REMOVE\_DUPLICATES global\_symbols)

# List manglings of module symbol names to try.

set(module\_symbols

\_\_my\_module\_MOD\_my\_sub # GNU 4.3

\_\_my\_module\_NMOD\_my\_sub # VisualAge

\_\_my\_module\_\_my\_sub # GNU 4.2

\_\_mymodule\_MOD\_mysub # GNU 4.3

\_\_mymodule\_NMOD\_mysub # VisualAge

\_\_mymodule\_\_mysub # GNU 4.2

my\_module$my\_sub # HP

my\_module\_mp\_my\_sub\_ # Intel

MY\_MODULE\_mp\_MY\_SUB # Intel on Windows

my\_module\_my\_sub\_ # PGI

my\_module\_MP\_my\_sub # NAG

mymodule$mysub # HP

mymodule\_mp\_mysub\_ # Intel

MYMODULE\_mp\_MYSUB # Intel on Windows

mymodule\_mysub\_ # PGI

mymodule\_MP\_mysub # NAG

\_QMmy\_modulePmy\_sub # LLVMFlang

\_QMmymodulePmysub # LLVMFlang

${FortranCInterface\_MODULE\_SYMBOLS}

)

list(REMOVE\_DUPLICATES module\_symbols)

# Note that some compiler manglings cannot be invoked from C:

# SunPro uses "my\_module.my\_sub\_"

# PathScale uses "MY\_SUB.in.MY\_MODULE"

# Add module symbols only with Fortran90.

if(CMAKE\_Fortran\_COMPILER\_SUPPORTS\_F90)

set(myfort\_modules mymodule.f90 my\_module.f90)

set(call\_mod call\_mod.f90)

set\_property(SOURCE main.F PROPERTY COMPILE\_DEFINITIONS CALL\_MOD)

else()

set(module\_symbols)

endif()

# Generate C symbol sources.

set(symbol\_sources)

if(NOT CMAKE\_Fortran\_COMPILER\_ID MATCHES "^(PathScale|Cray)$")

# Provide mymodule\_ and my\_module\_ init symbols because:

# - PGI Fortran uses module init symbols

# but not for:

# - PathScale Fortran uses module init symbols but module symbols

# use '.in.' so we cannot provide them anyway.

# - Cray Fortran >= 7.3.2 uses module init symbols but module symbols

# use 'mysub$mymodule\_' so we cannot provide them anyway.

list(APPEND symbol\_sources mymodule\_.c my\_module\_.c MY\_MODULE.c MYMODULE.c)

endif()

foreach(symbol IN LISTS global\_symbols module\_symbols)

# Skip symbols with '$' if C cannot handle them.

if(C\_SUPPORTS\_DOLLAR OR NOT "${symbol}" MATCHES "\\$")

if("${symbol}" MATCHES "SUB")

set(upper "-UPPER")

else()

set(upper)

endif()

string(REPLACE "$" "S" name "${symbol}")

set(source ${CMAKE\_CURRENT\_BINARY\_DIR}/symbols/${name}${upper}.c)

configure\_file(${CMAKE\_CURRENT\_SOURCE\_DIR}/symbol.c.in ${source} @ONLY)

list(APPEND symbol\_sources ${source})

endif()

endforeach()

# Provide symbols through Fortran.

add\_library(myfort STATIC mysub.f my\_sub.f ${myfort\_modules})

# Provide symbols through C but fall back to Fortran.

add\_library(symbols STATIC ${symbol\_sources})

target\_link\_libraries(symbols PUBLIC myfort)

# In case the Fortran compiler produces PIC by default make sure

# the C compiler produces PIC even if it is not its default.

set\_property(TARGET symbols PROPERTY POSITION\_INDEPENDENT\_CODE 1)

if(CMAKE\_Fortran\_COMPILER\_ID STREQUAL "LFortran")

add\_compile\_options(--implicit-interface)

endif()

# Require symbols through Fortran.

add\_executable(FortranCInterface main.F call\_sub.f ${call\_mod})

target\_link\_libraries(FortranCInterface PUBLIC symbols)

# If IPO is enabled here, GCC gfortran >= 12.0 will obfuscate

# the strings of the return values in the compiled executable,

# which we use to regex match against later.

# The static libraries must be build with IPO and non-IPO objects,

# as that will ensure the verify step will operate on IPO objects,

# if requested by the system compiler flags.

if(CMAKE\_Fortran\_COMPILER\_ID STREQUAL "GNU" AND

CMAKE\_Fortran\_COMPILER\_VERSION VERSION\_GREATER\_EQUAL 12)

target\_compile\_options(FortranCInterface PRIVATE "-fno-lto")

if(NOT APPLE)

target\_compile\_options(myfort PRIVATE "-flto=auto" "-ffat-lto-objects")

endif()

endif()

if(CMAKE\_C\_COMPILER\_ID STREQUAL "GNU" AND

CMAKE\_C\_COMPILER\_VERSION VERSION\_GREATER\_EQUAL 12)

if(NOT APPLE)

target\_compile\_options(symbols PRIVATE "-flto=auto" "-ffat-lto-objects")

endif()

endif()

file(GENERATE OUTPUT exe-$<CONFIG>.cmake CONTENT [[

set(FortranCInterface\_EXE "$<TARGET\_FILE:FortranCInterface>")

]])