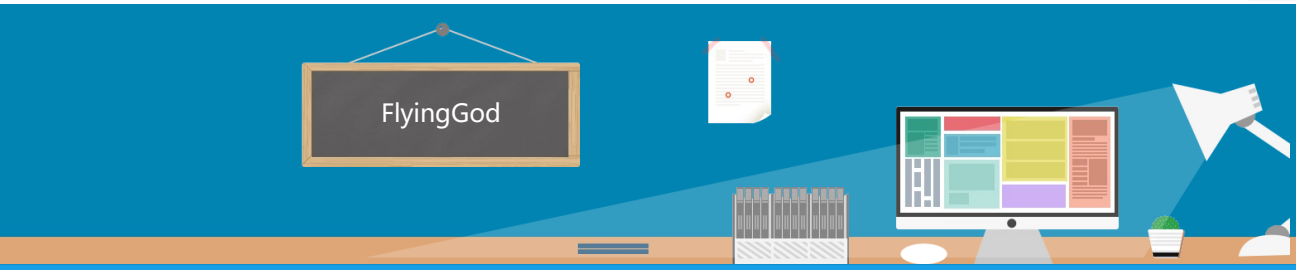

CMake: (三) 交叉编译

目录

CMAKE_TOOLCHAIN_FILE

```
#toolchain cmake file

SET(CMAKE_SYSTEM_NAME Linux)
SET(TOOLCHAIN_DIR "/home/xx/softwares/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux")
SET(3RDPART_LIBS_DIR "/home/xx/install")

#specify the cross compiler
SET(CMAKE_C_COMPILER ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-gcc CACHE FILEPATH "Archiver")
SET(CMAKE_CXX_COMPILER ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-g++ CACHE FILEPATH "Archiver")
#SET(CMAKE_GFORTRAN  ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-gfortran)
SET(CMAKE_AR ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-ar CACHE FILEPATH "Archiver")
SET(CMAKE_AS ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-as CACHE FILEPATH "Archiver")
SET(CMAKE_LD  ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-ld CACHE FILEPATH "Archiver")
SET(CMAKE_NM ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-nm CACHE FILEPATH "Archiver")
SET(CMAKE_STRIP  ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-strip CACHE FILEPATH "Archiver")

# where is the target environment
SET(CMAKE_FIND_ROOT_PATH  ${TOOLCHAIN_DIR} ${3RDPART_LIBS_DIR})

# search for programs in the build host directories
SET(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)

# for libraries and headers in the target directories
SET(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
SET(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
```

上面是一个在x86_64的Linux上编译arm32程序的toolchain.cmake,编译时候，可以使用如下指令

```
cd ./build
cmake -DCMAKE_TOOLCHAIN_FILE=~/toolchain.camke ../
```

参数说明

CMAKE_SYSTEM_NAME

这个参数是利用cmake进行交叉编译必须设置的，通常都是Linux或者Windows，声明要利用cmake创建运行在目标系统上的文件。如果要创建运行在没有操作系统的嵌入式环境，该参数要设置成Generic.如果CMAKE_SYSTEM_NAME进行了预先设置，CMAKE_CROSSCOMPLING经会被自动置位为True.所以可以被用来测试时候在交叉编译环境进行编译。

CMAKE_SYSTEM_VERSION

可选项，目标系统的版本，用的不多。

CMAKE_SYSTEM_PROCESS

可选项，目标系统的处理器，只有在目标程序要运行在不同硬件平台时候才需要进行设置针对不同的对象。它可以被用来做一些编译器编译选项的快速设定。

CMAKE_C_COMPLIER

C语言编译器，例如在shell中修改变量CC指向交叉编译的C编译器export CC=arm-linux-guneabihf-gcc, 或者在CMakeLists.txt内设置 SET(CMAKE_C_COMPILER ${TOOLCHAIN_DIR}/bin/arm-linux-gnueabihf-gcc)，但是这句话**只有写到PROJECT(project_name)之前才会起作用**，或者将一系列设置写在toolchain.cmake文件中，如上所述。

CMAKE_CXX_COMPLIER:CXX

CXX编译器，如果其是交叉编译工具，二者只需要声明一个。

主机环境

CMAKE_HOST_SYSTEM_NAME,CMAKE_HOST_SYSTEM_VERSION,CMAKE_HOST_SYSTEM_PROCESSOR,CMAKE_HOST_SYSTEM除了交叉环境编译情况下，主机和目标机器的变量值一样。

查找外部软件

一般工程都包含一些外部依赖的库或者工具，cmake提供给我们一些变量可以进行查找FIND_PROGRAM(), FIND_LIBRARY(), FIND_FILE(), FIND_PATH() and FIND_PACKAGE() 查找文件和返回结果。FIND_PACKAGE()其实并不进行查找，而是通过执行FindXXX.cmake模块，进而调用FIND_PROGRAM(), FIND_LIBRARY(), FIND_FILE(), FIND_PATH()去查找文件。例如当你编译软件时候，希望程序链接文件 /usr/lib/libjpeg.so，FIND_PACKAGE(JPEG)会直接在主机的相对位置查找此链接文件，所以我们必须设定CMake在指定的位置查找需要的文件。

下面的设置可以帮助我们完成相应的操作

CMAKE_FIND_ROOT_PATH

这是一个文件夹列表，如果你的目标环境安装在/opt/eldk/ppc_74xx，配置CMAKE_FIND_ROOT_PATH指向此处，然后

FIND_LIBRARY(BZ2_LIB bz2)将会搜索/opt/eldk/ppc_74xx/lib, /opt/eldk/ppc_74xx/usr/lib, /lib, /usr/lib 之后会将/opt/eldk/ppc_74xx/usr/lib/libbz2.so 作为结果返回。

CMAKE_FIND_ROOT_PATH默认设置为空。如果设置之后，cmake默认会首先搜索具有这个前缀的位置下文件，之后去其他位置查找。

- 每个FIND_XXX()可以通过参数NO_CMAKE_FIND_ROOT_PATH, ONLY_CMAKE_FIND_ROOT_PATH 和CMAKE_FIND_ROOT_PATH_BOTH设置查找范围
- 或者采用对所有的FIND_XXX()都起作用的的参数CMAKE_FIND_ROOT_PATH_MODE_PROGRAM, CMAKE_FIND_ROOT_PATH_MODE_LIBRARY 和 CMAKE_FIND_ROOT_PATH_MODE_INCLUDE进行设置。

如果你的工程不仅需要toolchain里面的工具，还有目标平台的附加库。我们还需要为这些依赖包建立一个安装文件夹，例如/home/ke/install,同时需要添加这个位置进去CMAKE_FIND_ROOT_PATH，之后FIND_XXX()才可以在这些位置进行查找。

> 如果之后你build packages在你的目标平台上面，也要安装在这个文件夹内。

CMKAE_FIND_ROOT_PATH_MODE_PROGRAM

这个变量为FIND_PROGRAM命令设置默认选项，可以是NEVER,ONLY,BOTH(默认).

1. NEVER：CMAKE_ROOT_PATH将会被FIND_PROGRAM()忽略,除非显示使能。
2. ONLY：FIND_PROGRAM只在CMAKE_FIND_ROOT_PATH为前缀的目录下查找需要的文件;
3. BOTH,先在CMAKE_FIND_ROOT_PATH查找之后再去其他位置。

在大多数情形下，FIND_PROGRAM()用来查找可以被EXECUTABLE_PROCESS() 或者 ADD_CUSTOM_COMMAND()执行的程序。所以在大多数情况下，主机的可执行程序是必须的，所以CMAKE_FIND_ROOT_PATH_MODE_PROGRAM通常被设置成为NEVER。

CMAKE_FIND_ROOT_PATH_MODE_LIBRARY

FIND_LIBRARY()命令的默认设置，设置链接需要的文件目录，通常都位于目标系统内，所以一般设置成为ONLY;

CMAKE_FIND_ROOT_PATH_MODE_INCLUDE

FIND_FILE()和FIND_PATH()的默认设置，用来查找include文件目录，文件通常位于目标机器中，所以一般设置为ONLY,我们也可以通过设置NO_CMAKE_FIND_ROOT_PATH, ONLY_CMAKE_FIND_ROOT_PATH 和CMAKE_FIND_ROOT_PATH_BOTH为FIND_FILE()和FIND_PATH()限定文件的查找位置。

使用其他可执行程序

In some cases during a build executables are created which are then used in ADD_CUSTOM_COMMAND() or ADD_CUSTOM_TARGET() during the same build process.When cross compiling this won't work without modifications because the executables cannot run on the build host. Starting with CMake 2.6 it is possible to "import" executable targets into a CMake project. When cross compiling this has to be used to import executables built in a native build into the cross-build. This can be done like this:

```
# when crosscompiling import the executable targets from a file
IF(CMAKE_CROSSCOMPILING)
  SET(IMPORT_EXECUTABLES "IMPORTFILE-NOTFOUND" CACHE FILEPATH "Point it to the export file from a native build")
  INCLUDE(${IMPORT_EXECUTABLES})
ENDIF(CMAKE_CROSSCOMPILING)

...

# only build the generator if not crosscompiling
IF(NOT CMAKE_CROSSCOMPILING)
   ADD_EXECUTABLE(mygenerator mygen.cpp)
   TARGET_LINK_LIBRARIES(mygenerator ${SOME_LIBS})
ENDIF(NOT CMAKE_CROSSCOMPILING)

# then use the target name as COMMAND, CMake >= 2.6 knows how to handle this
ADD_CUSTOM_COMMAND(OUTPUT ${CMAKE_CURRENT_BINARY_DIR}/generated.c
                  COMMAND mygenerator foo.dat -o ${CMAKE_CURRENT_BINARY_DIR}/generated.c
                  DEPENDS foo.dat )


...
# export the generator target to a file, so it can be imported (see above) by another build
# the IF() is not necessary, but makes the intention clearer
IF(NOT CMAKE_CROSSCOMPILING)
  EXPORT(TARGETS mygenerator FILE ${CMAKE_BINARY_DIR}/ImportExecutables.cmake )
ENDIF(NOT CMAKE_CROSSCOMPILING)
```

So during the native build the target "mygenerator" will be built and used in ADD_CUSTOM_COMMAND(). As command only the target name is used. CMake >= 2.6.0 recognizes this and creates the dependencies and will use the path to the created executable when executing the command. At the end the EXPORT() function (since CMake 2.6.0) is called, which "exports" the listed targets to the file ${CMAKE_BINARY_DIR}/ImportExecutables.cmake, which will look like this:

```
ADD_EXECUTABLE(mygenerator IMPORT)
SET_TARGET_PROPERTIES(mygenerator PROPERTIES
                      LOCATION /home/alex/build-native/bin/mygenerator )
```

This file is then included when cross compiling, it either has to be specified using -D or via the cmake GUI. Then later on the command for actually building mygenerator is excluded. In ADD_CUSTOM_COMMAND() mygenerator will be recognized as an imported target and it will be used when executing the command.

If the executable mygenerator also has to be built when cross compiling, then some more logic needs to be added, e.g. like this:

```
# when crosscompiling import the executable targets from a file
IF(CMAKE_CROSSCOMPILING)
  SET(IMPORT_EXECUTABLES "IMPORTFILE-NOTFOUND" CACHE FILEPATH "Point it to the export file from a native build")
  INCLUDE(${IMPORT_EXECUTABLES})
ENDIF(CMAKE_CROSSCOMPILING)

...

# always build the executable
ADD_EXECUTABLE(mygenerator mygen.cpp)
TARGET_LINK_LIBRARIES(mygenerator ${SOME_LIBS})

# but use different names for the command
IF(CMAKE_CROSSCOMPILING)
   SET(mygenerator_EXE native-mygenerator)
ELSE(CMAKE_CROSSCOMPILING)
   SET(mygenerator_EXE mygenerator)
ENDIF(CMAKE_CROSSCOMPILING)

# then use the target name as COMMAND, CMake >= 2.6 knows how to handle this
ADD_CUSTOM_COMMAND(OUTPUT ${CMAKE_CURRENT_BINARY_DIR}/generated.c
                  COMMAND ${mygenerator_EXE} foo.dat -o ${CMAKE_CURRENT_BINARY_DIR}/generated.c
                  DEPENDS foo.dat )


...
# export the generator target to a file, so it can be imported (see above) by another build
# the IF() is not necessary, but makes the intention clearer
# use the NAMESPACE option of EXPORT() to get a different target name for mygenerator when exporting
IF(NOT CMAKE_CROSSCOMPILING)
  EXPORT(TARGETS mygenerator FILE ${CMAKE_BINARY_DIR}/ImportExecutables.cmake NAMESPACE native- )
ENDIF(NOT CMAKE_CROSSCOMPILING)
```

分类: CMake

采男孩的小蘑菇
粉丝 - 130 关注 - 3
+加关注

« 上一篇： CMake: (二) 常见配置

» 下一篇： CMake: (四) CMake语法规则

posted @ 2020-09-22 17:51 采男孩的小蘑菇 阅读(4317) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

登录后才能查看或发表评论，立即 **登录** 或者 **逛逛** 博客园首页

编辑推荐：

· SQL 优化实战分享

· 分页列表缓存，你真的会吗

· 红黑树是怎么来的

· 一次 redis 主从切换导致的数据丢失与陷入只读状态故障

· [MAUI] 在 .NET MAUI 中复刻苹果 Cover Flow

即构专区：

· WebGL 及其在 WebRTC 中的应用

· 即视角 | 元宇宙社交：新瓶旧酒 or 老树新芽?

· 在线自习室场景爆发，在线教育平台用户时间争夺战打响

· 音视频开发进阶 | 第六讲：色彩和色彩空间-上篇

· 小程序官方直播插件怎么接入

ge

@张驰洲 你好，我在RVIZ中显示了点云，但是image显示不了，请问这是什么问题...

--WQ5277