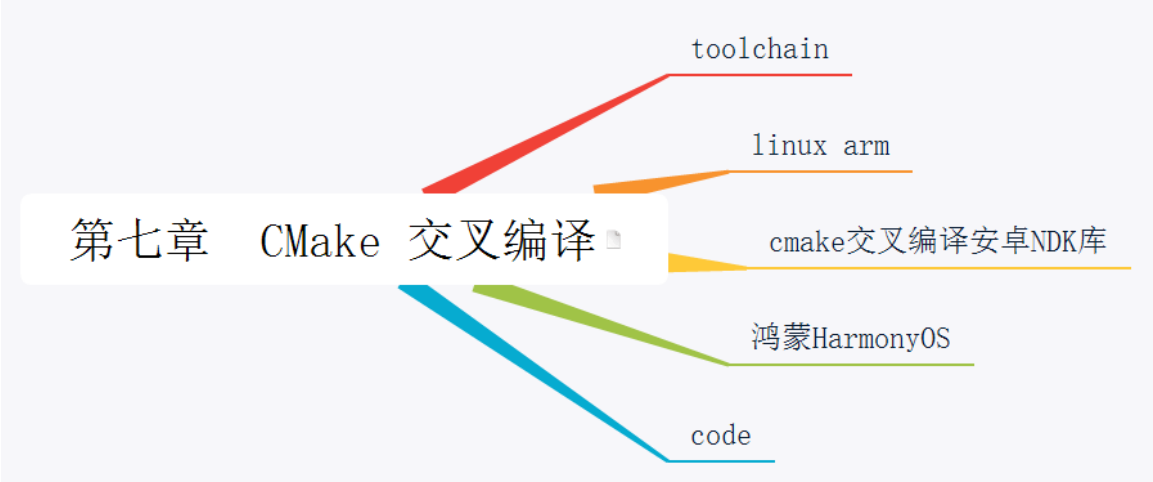
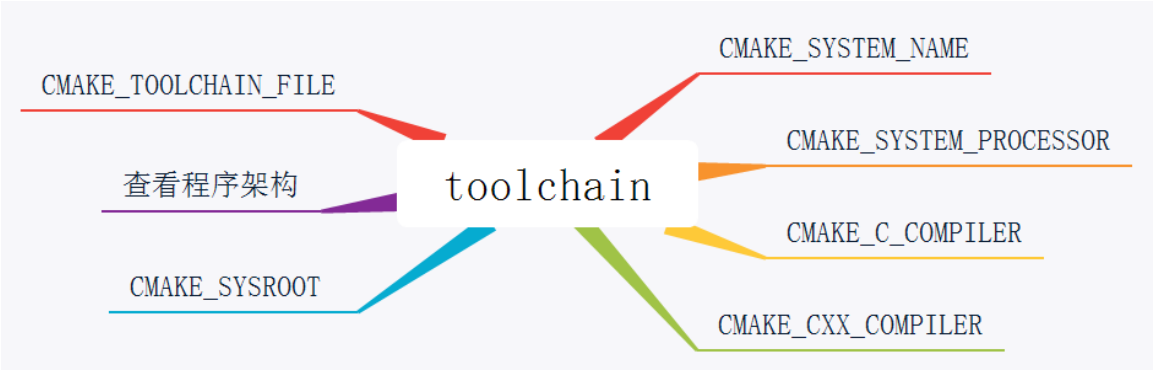
第七章 CMake 交叉编译

DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles" cmake -DANDROID_ABI=arm64-v8a -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles".....193.5.6. cmake_minimum_required (VERSION 3.10) project (mylib) file(WRITE mylib.h [=[const char * Mylib();]=]) file(WRITE mylib.cpp [=[#include <iostream> using namespace std; const char * Mylib() { cout<<"call Mylib"<<endl; return "mylib"; }]=]) #[[cmake -DCMAKE_TOOLCHAIN_FILE=D:/harmony_sdk/native/2.2.0.3/build/cmake/ohos.toolchain.cmake -S . -B b2 -G Ninja cmake -DCMAKE_TOOLCHAIN_FILE=C:\Users\xiaca\AppData\Local\Android\Sdk\ndk\24.0.8215888\build\cmake\android.toolchain.cmake -S . -B build -G "NMake Makefiles" cmake -DCMAKE_TOOLCHAIN_FILE=C:\Users\xiaca\AppData\Local\Android\Sdk\ndk\21.4.7075529\build\cmake\android.toolchain.cmake -S . -B build -G "NMake Makefiles" cmake -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_ABI=armeabi-v7a -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles" cmake -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_ABI=x86_64 -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles" cmake -DANDROID_ABI=x86_64 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles" cmake -DANDROID_ABI=x86 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles" cmake -DANDROID_ABI=armeabi-v7a -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles" cmake -DANDROID_ABI=arm64-v8a -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -

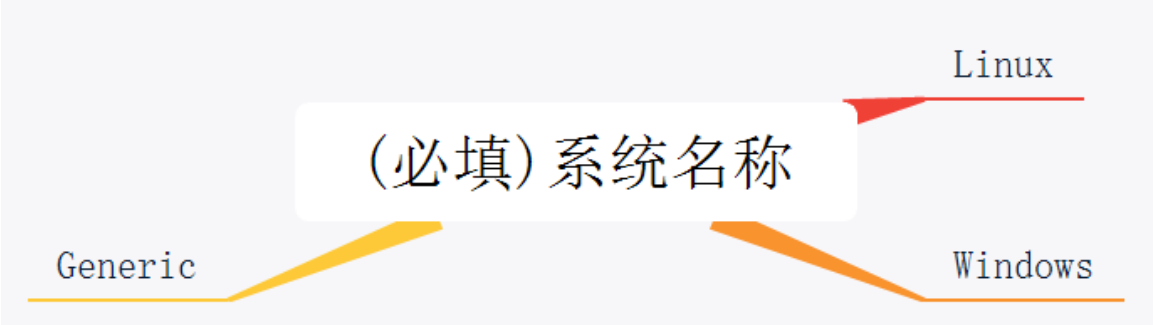


1. toolchain



1.1. CMAKE_SYSTEM_NAME

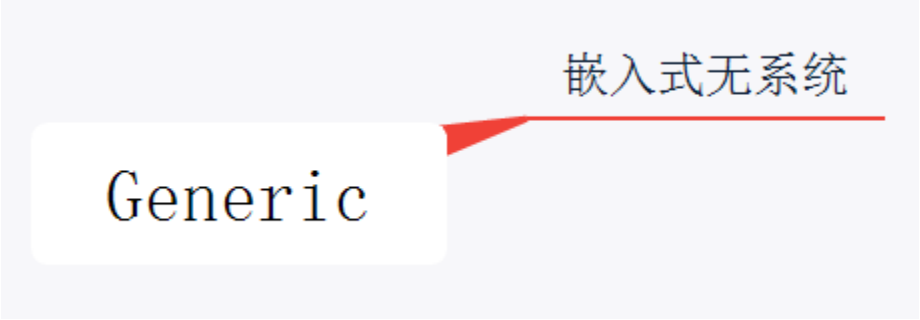
1.1.1. (必填)系统名称



Linux

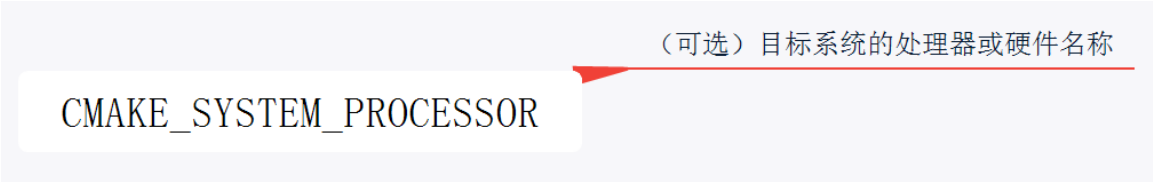
Windows

Generic



嵌入式无系统

1.2. CMAKE_SYSTEM_PROCESSOR

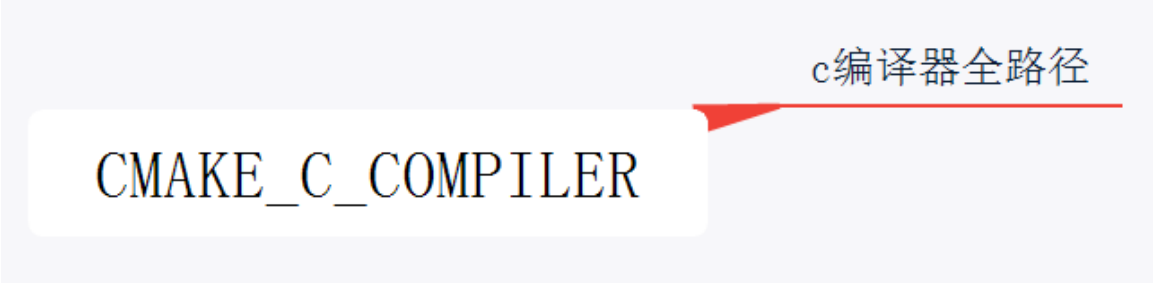


1.2.1. （可选）目标系统的处理器或硬件名称

用于加载${CMAKE_SYSTEM_NAME}-COMPILER_ID-${CMAKE_SYSTEM_PROCESSOR}.cmake

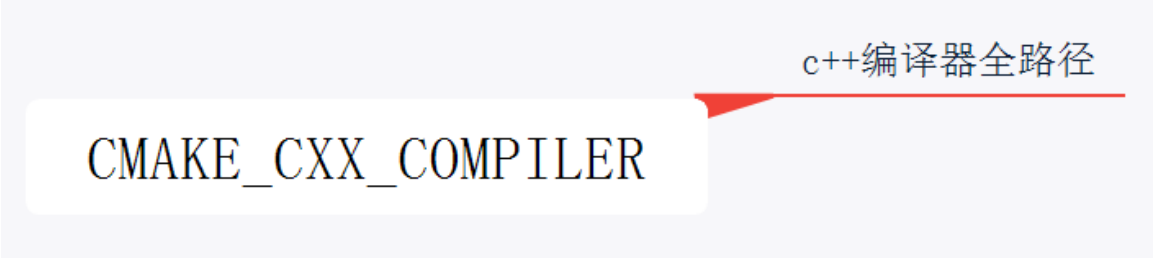
修改目标的编译器标志

1.3. CMAKE_C_COMPILER

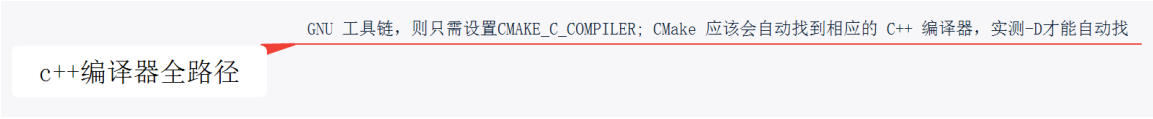


1.3.1. c编译器全路径

1.4. CMAKE_CXX_COMPILER



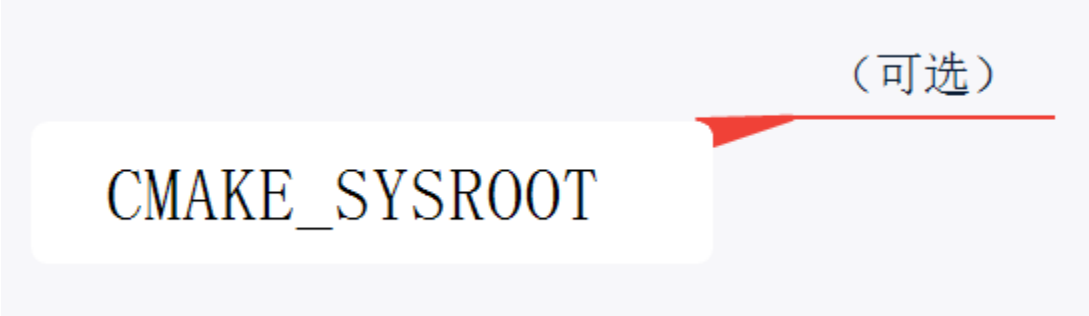
1.4.1. c++编译器全路径



GNU 工具链，则只需设置CMAKE_C_COMPILER; CMake

应该会自动找到相应的 C++ 编译器，实测-D才能自动找

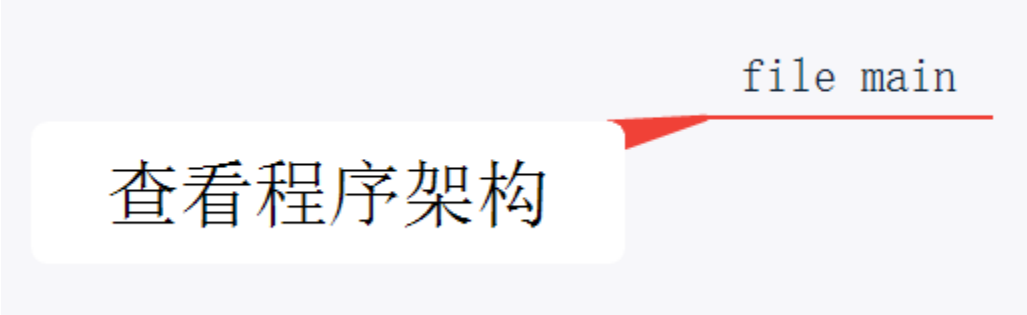
1.5. CMAKE_SYSROOT



1.5.1. （可选）

系统库头文件的路径

1.6. 查看程序架构



1.6.1. file main

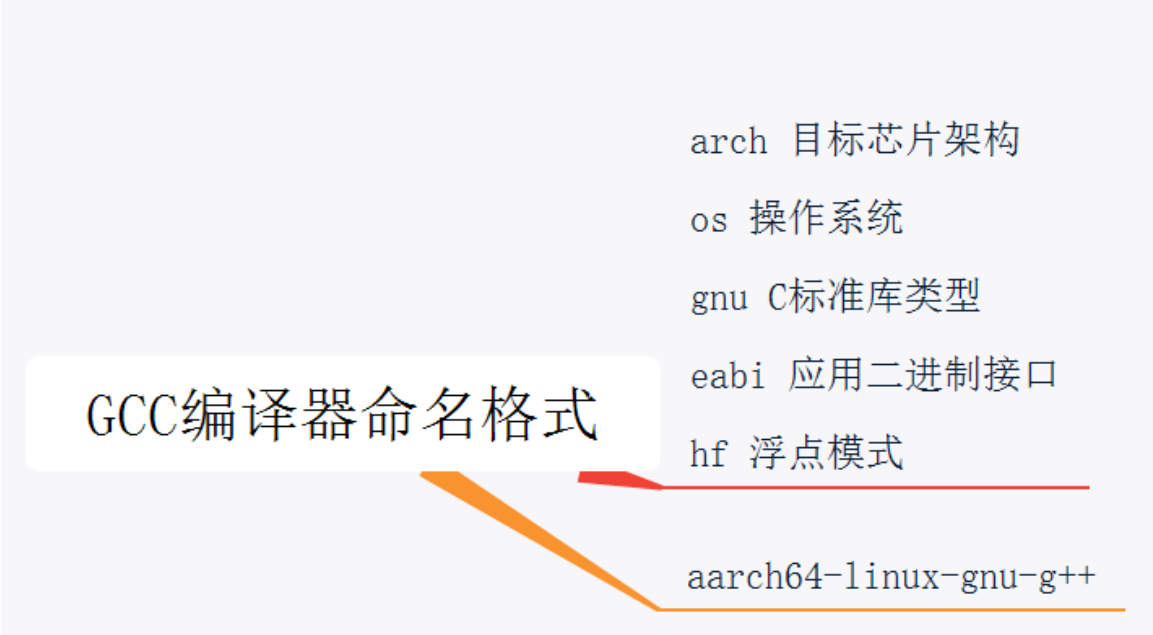
1.7. CMAKE_TOOLCHAIN_FILE

1.7.1. 指定文件路径

1.7.2. ohos.toolchain.cmake

2. linux arm

2.1. GCC编译器命名格式



2.1.1.

arch 目标芯片架构

os 操作系统

gnu C标准库类型

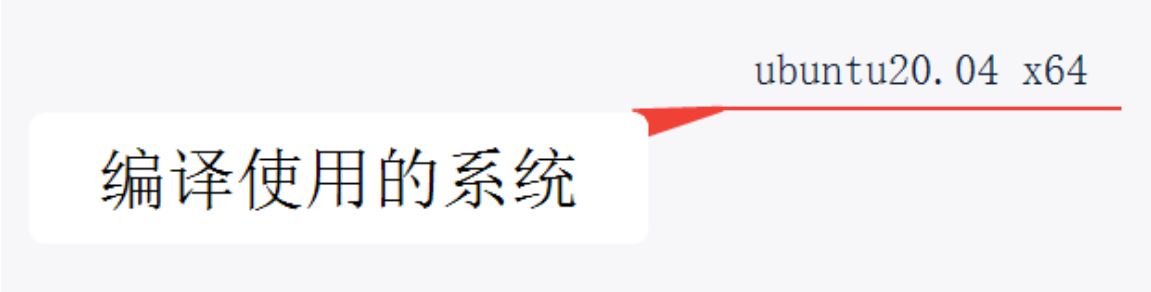
eabi 应用二进制接口

hf 浮点模式

2.1.2. aarch64-linux-gnu-g++

2.2. 测试环境

2.2.1. 编译使用的系统

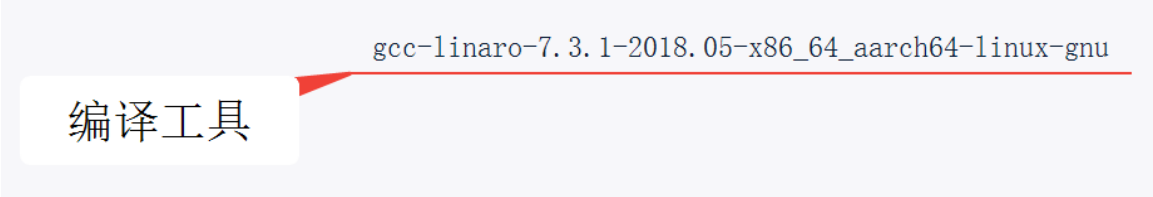


ubuntu20.04 x64

2.2.2. 目标系统

ubuntu arm版本

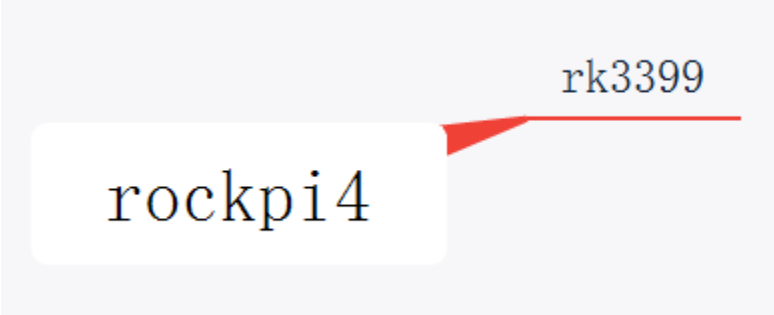
2.2.3. 编译工具



gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu

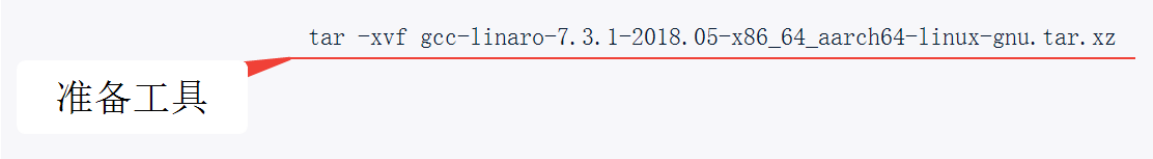
2.2.4. 开发板

rockpi4



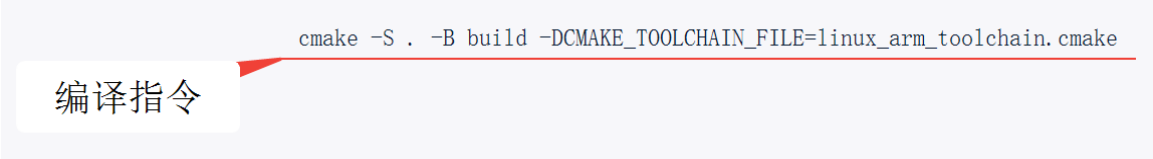
rk3399

2.3. 准备工具



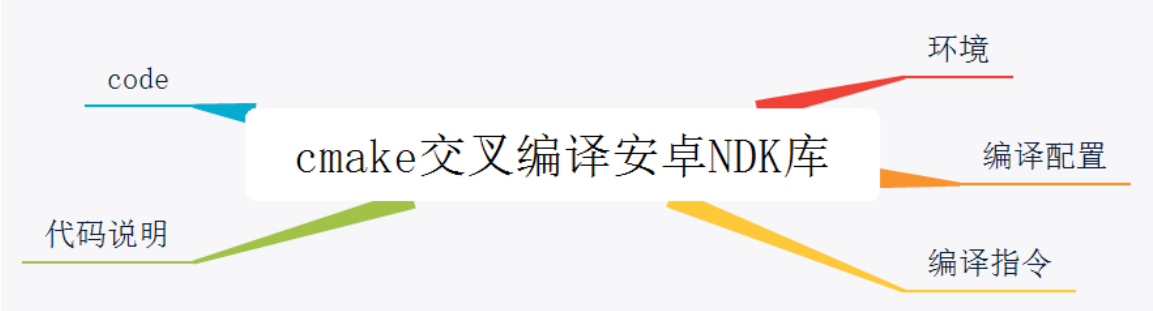
2.3.1. tar -xvf gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz

2.4. 编译指令



2.4.1. cmake -S . -B build -DCMAKE_TOOLCHAIN_FILE=linux_arm_toolchain.cmake

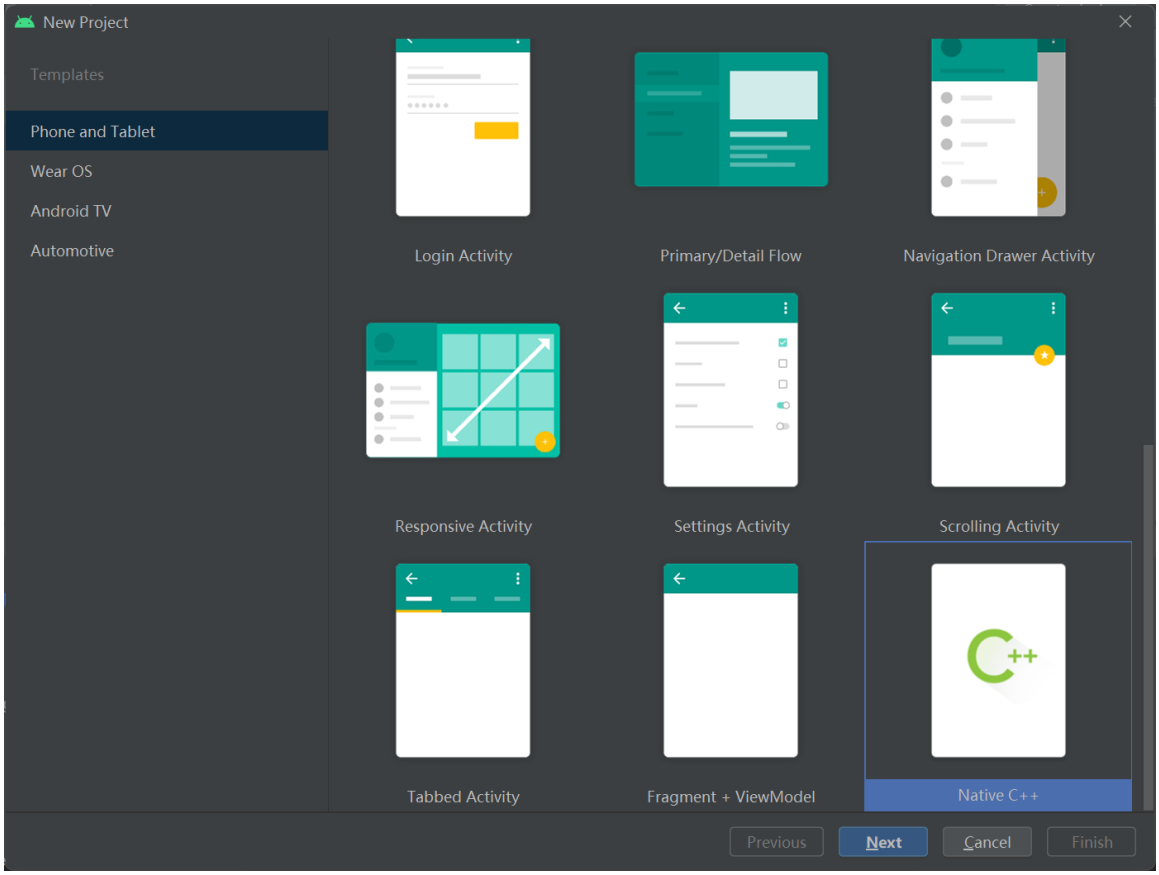
3. cmake交叉编译安卓NDK库



3.1. 环境

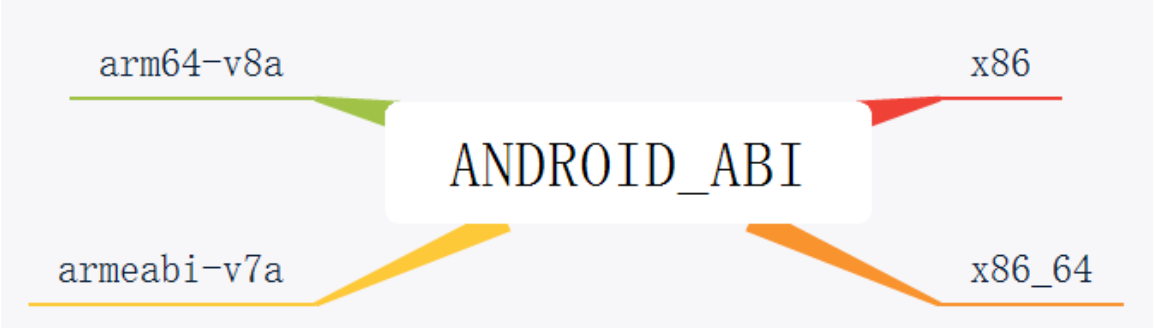
3.1.1. Android Studio Bumblebee

3.1.2. 创建native c++项目



3.2. 编译配置

3.2.1. ANDROID_ABI



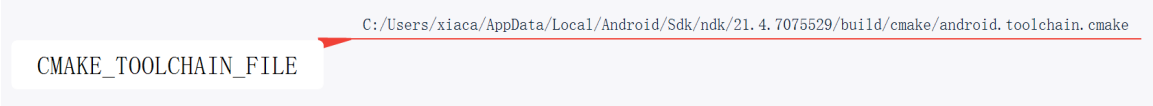
x86

x86_64

armeabi-v7a

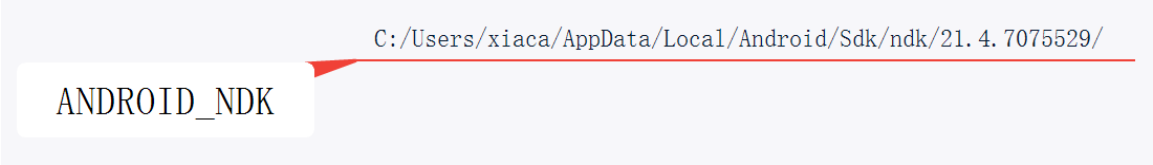
arm64-v8a

3.2.2. CMAKE_TOOLCHAIN_FILE



C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake

3.2.3. ANDROID_NDK

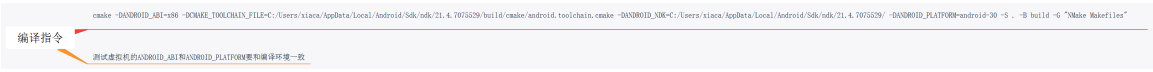


C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/

3.2.4. ANDROID_PLATFORM

android-30

3.3. 编译指令



3.3.1. cmake -DANDROID_ABI=x86 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"

3.3.2. 测试虚拟机的ANDROID_ABI和ANDROID_PLATFORM要和编译环境一致

3.4. 代码说明

3.4.1. 编译静态库

CMakeLists.txt

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.18)
project(mylib)
file(WRITE mylib.h [=[
const char *MyLib();
]=])

file(WRITE mylib.cpp [=[
#include "mylib.h"
const char *MyLib()
{
    return "mylib return";
};
]=])
# 给安卓使用的静态库
add_library(mylib STATIC mylib.cpp)
target_compile_options(mylib PRIVATE -fPIC)
```


```
cmake_minimum_required(VERSION 3.18)
project(mylib)
file(WRITE mylib.h [=[
const char *MyLib();
]=])

file(WRITE mylib.cpp [=[
#include "mylib.h"
const char *MyLib()
{
   return "mylib return";
};
]=])
# 给安卓使用的静态库
add_library(mylib STATIC mylib.cpp)
target_compile_options(mylib PRIVATE -fPIC)
```


编译四种不同的ABI

cmake -S . -B build -G "NMake Makefiles" -DANDROID_ABI=x86 -DANDROID_PLATFORM=android-30 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529

cmake -S . -B build -G "NMake Makefiles" -DANDROID_ABI=x86_64 -DANDROID_PLATFORM=android-30 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529

cmake -S . -B build -G "NMake Makefiles" -DANDROID_ABI=armeabi-v7a -DANDROID_PLATFORM=android-30 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529

cmake -S . -B build -G "NMake Makefiles" -DANDROID_ABI=arm64-v8a -DANDROID_PLATFORM=android-30 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529

3.4.2. 导入静态库

导入静态库

```
target_link_directories(a602cmake_android_ndk PUBLIC
        ${CMAKE_SOURCE_DIR}/mylib/build/)
target_link_libraries( # Specifies the target library.
        a602cmake_android_ndk

        # Links the target library to the log library
        # included in the NDK.
        ${log-lib}
        mylib
        )
```

target_link_directories(a602cmake_android_ndk PUBLIC

${CMAKE_SOURCE_DIR}/mylib/build/)

target_link_libraries(# Specifies the target library.

a602cmake_android_ndk

 # Links the target library to the log library

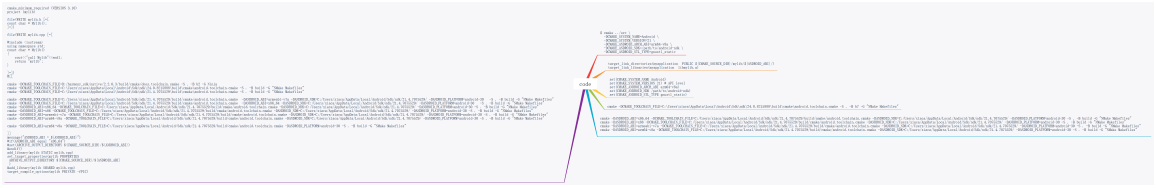
 # included in the NDK.

${log-lib}

mylib

)

3.5. code



3.5.1. $ cmake ../src \

**-DCMAKE_SYSTEM_NAME=Android **

**-DCMAKE_SYSTEM_VERSION=21 **

**-DCMAKE_ANDROID_ARCH_ABI=arm64-v8a **

**-DCMAKE_ANDROID_NDK=/path/to/android-ndk **

-DCMAKE_ANDROID_STL_TYPE=gnustl_static

3.5.2. target_link_directories(myapplication PUBLIC
${CMAKE_SOURCE_DIR}/mylib/${ANDROID_ABI}/)
target_link_libraries(myapplication libmylib.a)

3.5.3. set(CMAKE_SYSTEM_NAME Android)
set(CMAKE_SYSTEM_VERSION 21) # API level
set(CMAKE_ANDROID_ARCH_ABI arm64-v8a)
set(CMAKE_ANDROID_NDK /path/to/android-ndk)
set(CMAKE_ANDROID_STL_TYPE gnustl_static)

3.5.4. cmake -
DCMAKE_TOOLCHAIN_FILE=C:\Users\xiaca\AppData\Local\Android\Sdk\ndk\24.
0.8215888\build\cmake\android.toolchain.cmake -S . -B b7 -G "NMake Makefiles"

3.5.5. cmake -DANDROID_ABI=x86_64 -
DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.
4.7075529/build/cmake/android.toolchain.cmake -
DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529
/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DANDROID_ABI=x86 -
DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.
4.7075529/build/cmake/android.toolchain.cmake -
DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529
/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DANDROID_ABI=armeabi-v7a -
DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.
4.7075529/build/cmake/android.toolchain.cmake -
DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529
/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DANDROID_ABI=arm64-v8a -
DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.
4.7075529/build/cmake/android.toolchain.cmake -

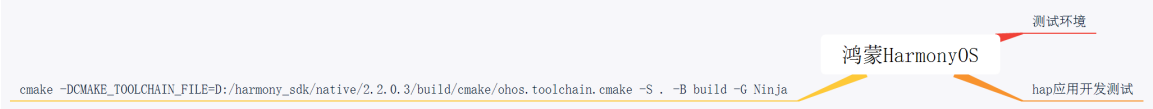
```
DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529
/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"

3.5.6. cmake_minimum_required (VERSION 3.10)
project (mylib)

file(WRITE mylib.h [=[
const char * Mylib();
]=])

file(WRITE mylib.cpp [=[

#include <iostream>
using namespace std;
const char * Mylib()
{
   cout<<"call Mylib"<<endl;
   return "mylib";
}

]=])
#[[

cmake -
DCMAKE_TOOLCHAIN_FILE=D:/harmony_sdk/native/2.2.0.3/build/cmake/ohos.to
olchain.cmake -S . -B b2 -G Ninja
cmake -
DCMAKE_TOOLCHAIN_FILE=C:\Users\xiaca\AppData\Local\Android\Sdk\ndk\24.
0.8215888\build\cmake\android.toolchain.cmake -S . -B build -G "NMake
Makefiles"
cmake -
```

DCMAKE_TOOLCHAIN_FILE=C:\Users\xiaca\AppData\Local\Android\Sdk\ndk\21.4.7075529\build\cmake\android.toolchain.cmake -S . -B build -G "NMake Makefiles"

cmake -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_ABI=armeabi-v7a -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_ABI=x86_64 -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DANDROID_ABI=x86_64 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DANDROID_ABI=x86 -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DANDROID_ABI=armeabi-v7a -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"
cmake -DANDROID_ABI=arm64-v8a -

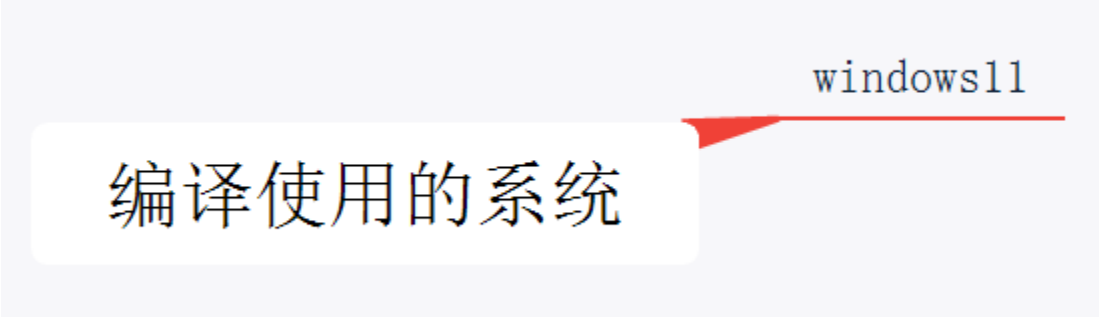
```
DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_NDK=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/ -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"

cmake -DANDROID_ABI=arm64-v8a -DCMAKE_TOOLCHAIN_FILE=C:/Users/xiaca/AppData/Local/Android/Sdk/ndk/21.4.7075529/build/cmake/android.toolchain.cmake -DANDROID_PLATFORM=android-30 -S . -B build -G "NMake Makefiles"

]]
message("ANDROID_ABI = ${ANDROID_ABI}")
#if(ANDROID_ABI equal "x86_64")
#set(ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_SOURCE_DIR}/${ANDROID_ABI})
#endif()
add_library(mylib STATIC mylib.cpp)
set_target_properties(mylib PROPERTIES
 ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_SOURCE_DIR}/${ANDROID_ABI}
 )
#add_library(mylib SHARED mylib.cpp)
target_compile_options(mylib PRIVATE -fPIC)
```

4. 鸿蒙HarmonyOS



4.1. 测试环境

4.1.1. 编译使用的系统



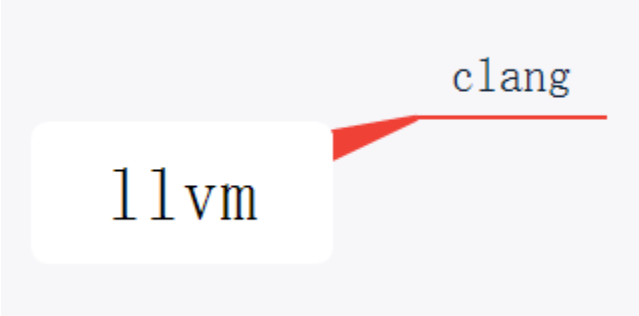
windows11

4.1.2. 目标系统

HarmonyOS 2.0

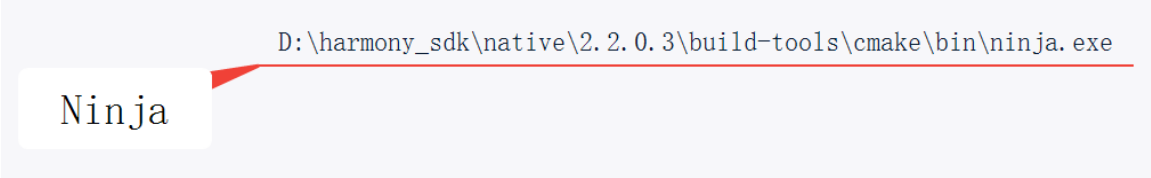
4.1.3. 编译工具

llvm



clang

Ninja



D:\harmony_sdk\native\2.2.0.3\build-tools\cmake\bin\ninja.exe

4.1.4. 手机

华为P40

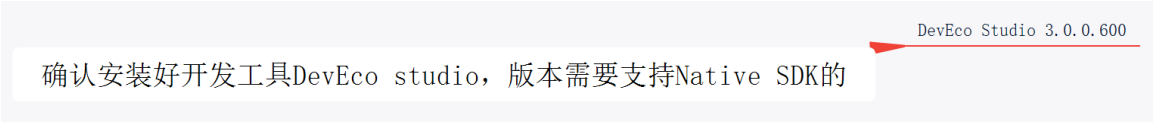
arm64-v8a

4.1.5. 开发工具

DevEco Studio 3.0.0.600

4.2. hap应用开发测试

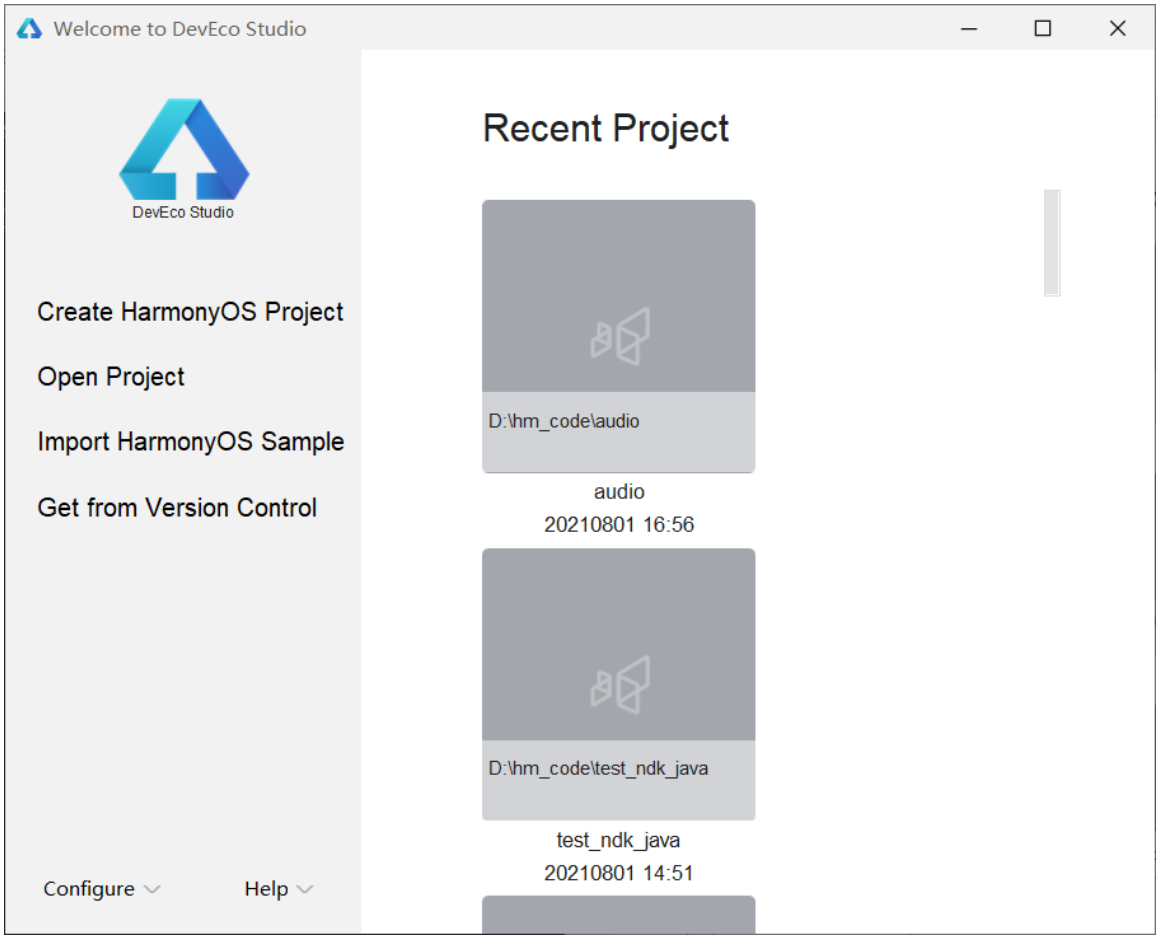
4.2.1. 确认安装好开发工具DevEco studio，版本需要支持Native SDK的

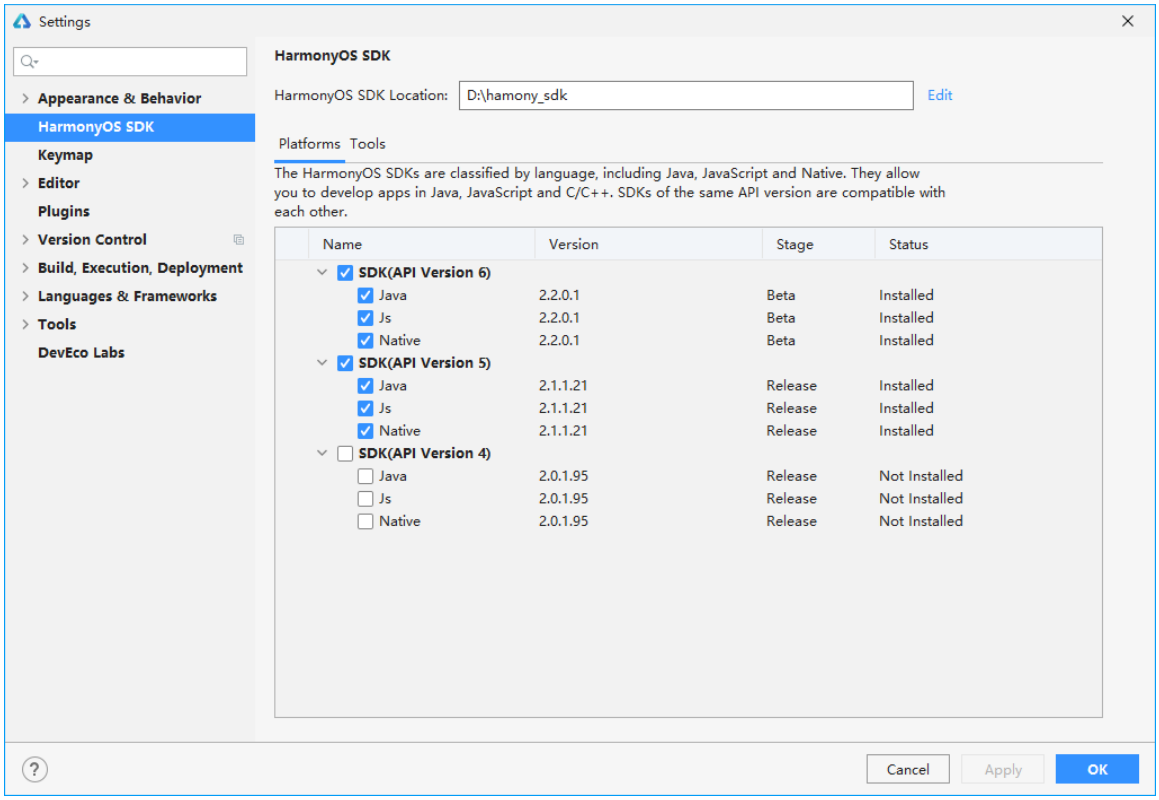


DevEco Studio 3.0.0.600

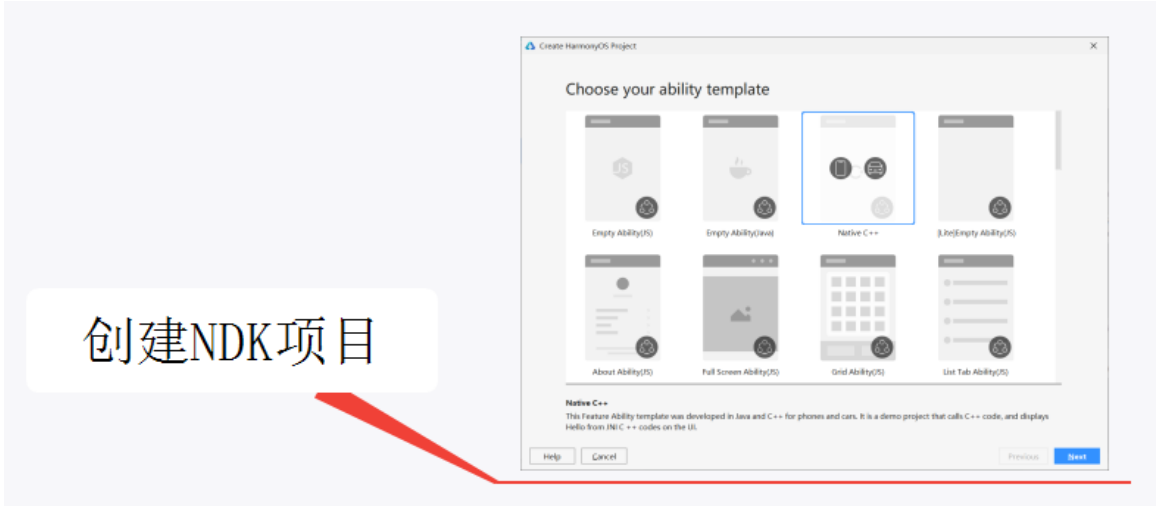
4.2.2. 设置安装Native SDK（NDK）

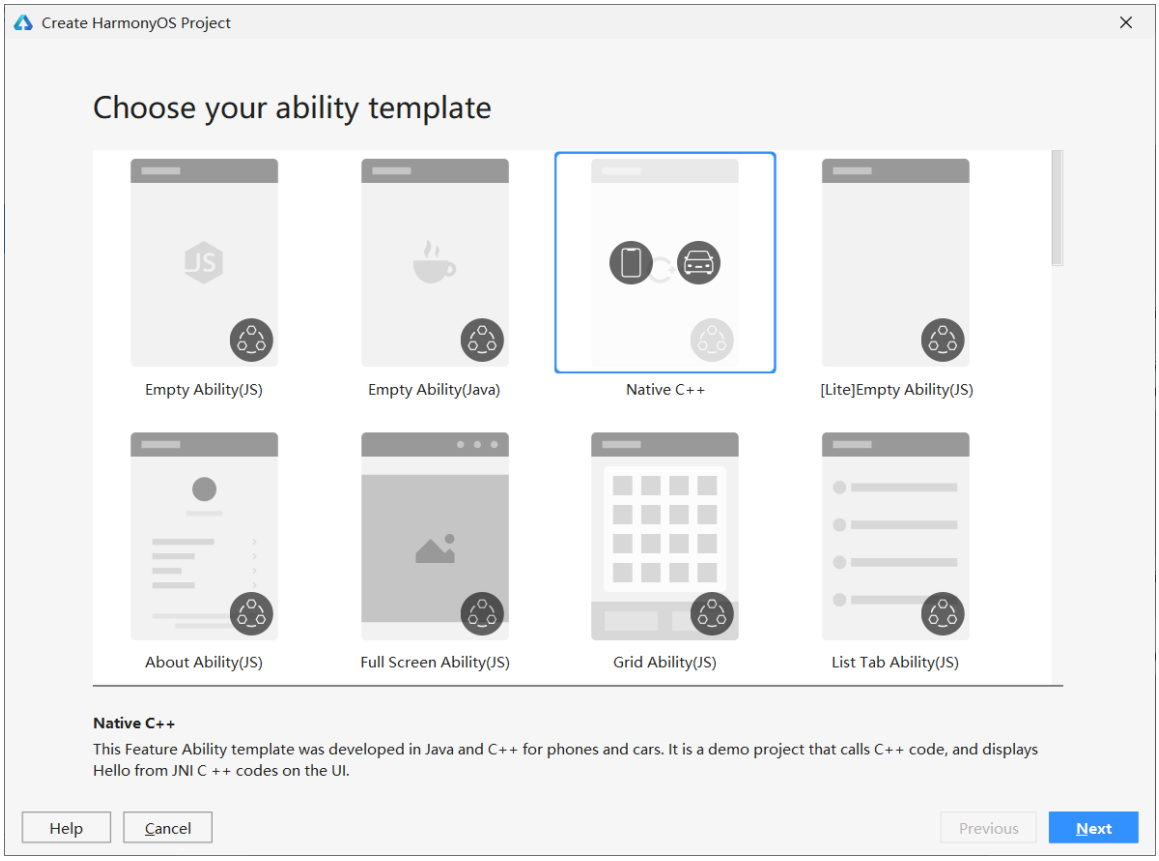

设置安装Native SDK（NDK）
Recent Project
Create HarmonyOS Project
Open Project
Import HarmonyOS Sample
Get from Version Control
Configure
Help


Welcome to DevEco Studio
DevEco Studio
Create HarmonyOS Project
Open Project
Import HarmonyOS Sample
Get from Version Control
Configure
Help
Recent Project
D:\hm_code\audio
audio
20210801 16:56
D:\hm_code\test_ndk_java
test_ndk_java
20210801 14:51


Settings
Appearance & Behavior
HarmonyOS SDK
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
DevEco Labs
HarmonyOS SDK
HarmonyOS SDK Location:
D:\hamony_sdk
Edit
Platforms
Tools
The HarmonyOS SDKs are classified by language, including Java, JavaScript and Native. They allow you to develop apps in Java, JavaScript and C/C++. SDKs of the same API version are compatible with each other.
Name
Version
Stage
Status
SDK(API Version 6)
Java 2.2.0.1 Beta Installed
Js 2.2.0.1 Beta Installed
Native 2.2.0.1 Beta Installed
SDK(API Version 5)
Java 2.1.1.21 Release Installed
Js 2.1.1.21 Release Installed
Native 2.1.1.21 Release Installed
SDK(API Version 4)
Java 2.0.1.95 Release Not Installed
Js 2.0.1.95 Release Not Installed
Native 2.0.1.95 Release Not Installed
Cancel
Apply
OK

4.2.3. 创建NDK项目


创建NDK项目
Choose your ability template
Native C++
Help
Cancel
Next



4.2.4. #导入mylib 静态库

add_library(mylib STATIC IMPORTED)

#指定导入库的路径

set_target_properties(mylib PROPERTIES IMPORTED_LOCATION

${CMAKE_CURRENT_SOURCE_DIR}/mylib /liblua.a)

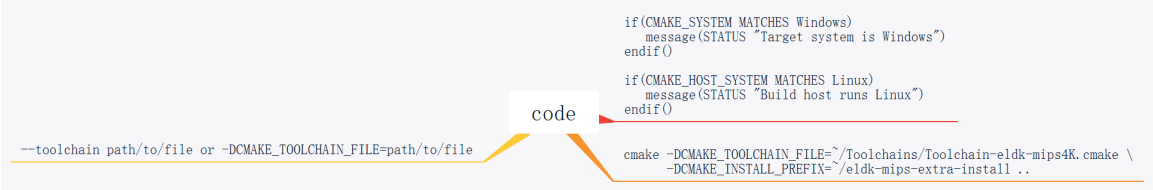
add_library(test_ndk SHARED test_ndk cpp)

target_link_libraries(test_ndk libhilog_ndk.z.so mylib)

4.3. cmake -

DCMAKE_TOOLCHAIN_FILE=D:/harmony_sdk/native/2.2.0.3/build/cmake/ohos.toolchain.cmake -S . -B build -G Ninja

5. code



5.1. if(CMAKE_SYSTEM MATCHES Windows)

message(STATUS "Target system is Windows")

endif()

if(CMAKE_HOST_SYSTEM MATCHES Linux)

message(STATUS "Build host runs Linux")

endif()

**5.2. cmake -DCMAKE_TOOLCHAIN_FILE=~/Toolchains/Toolchain-eldk-mips4K.cmake **

-DCMAKE_INSTALL_PREFIX=~/eldk-mips-extra-install ..

5.3. --toolchain path/to/file or -DCMAKE_TOOLCHAIN_FILE=path/to/file