SDK2 开发文档

重要说明

sdk2.aar 本身不包含 .so 文件，只包含 Java/Kotlin 代码。Native 库来自以下三个依赖模块：

- **fimage**：包含 libfimage.so（图像处理）
- **jzint**：包含 libzbcode.so（条码生成）
- **serial-port**：包含 libserial_port.so（串口通信）

这三个模块必须配置 16KB 对齐，才能在 Android 15+ 设备上正常运行。

使用步骤

1. 项目结构配置

在 `settings.gradle.kts` 中包含原生库模块：

```
// settings.gradle.kts
rootProject.name = "YourProject"

include(":app")
include(":fimage")          // 必须包含
include(":jzint")           // 必须包含
include(":serial-port")     // 必须包含
//
```

2. 全局配置

在 `gradle.properties` 中添加：

```
# 16KB 页面大小链接器标志（全局生效）
android.native.linkFlags=-Wl,-z,max-page-size=16384

# 启用 ART 配置文件
android.experimental.enableArtProfiles=true
```

3. 应用层配置

在 `app/build.gradle.kts` 中配置：

```
android {
    namespace = "com.yourcompany.yourapp"
    compileSdk = 34

    defaultConfig {
        minSdk = 21
        targetSdk = 34
    }
    //ndk架构
```

```
    flavorDimensions += "cpu_abi"
    productFlavors {
        create("allabi").apply {
            dimension = "cpu_abi"
            ndk.abiFilters.clear()
            ndk.abiFilters.add("arm64-v8a")
        }
    }

    // packaging 选项
    packaging {
        jniLibs {
            // 使用 true 确保 .so 文件未压缩打包
            useLegacyPackaging = true
        }
    }
}

dependencies {
    // 引用 sdk2.aar
    implementation files('libs/sdk2-release.aar')

    // 手动添加 sdk2 的依赖
    implementation 'androidx.annotation:annotation:1.7.1'
    implementation 'com.jcraft:jzlib:1.1.3'

    // 添加原生库模块（源代码模块）
    implementation project(':fimage')
    implementation project(':jzint')
    implementation project(':serial-port')
}
```

4. 原生库模块配置

每个原生库模块（fimage、jzint、serial-port）都需要配置 16KB 对齐。

fimage/build.gradle

```
plugins {
    id 'com.android.library'
}

android {
    namespace 'com.gainscha.fimage'
    compileSdk 34

    defaultConfig {
        minSdk 21
        targetSdk 34

        externalNativeBuild {
            cmake {
                // 启用灵活页面大小支持
                arguments "-DANDROID_SUPPORT_FLEXIBLE_PAGE_SIZES=ON"
            }
```

```
        }

        //构建 arm64-v8a 架构
        ndk {
            abiFilters 'arm64-v8a'
        }
    }

    externalNativeBuild {
        cmake {
            path "src/main/cpp/CMakeLists.txt"
            version "3.22.1"
        }
    }

    // 使用新的打包方式
    packaging {
        jniLibs {
            useLegacyPackaging = false
        }
    }
}
```

fimage/src/main/cpp/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.18.1)
project("fimage")

# 设置 16KB 页面对齐的链接器标志
set(CMAKE_SHARED_LINKER_FLAGS "${CMAKE_SHARED_LINKER_FLAGS} -Wl,-z,max-page-
size=16384")
set(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} -Wl,-z,max-page-
size=16384")

add_library(fimage SHARED fimage.cpp)
target_link_libraries(fimage ${log-lib} FreeImage)
```

jzint/build.gradle

```
apply plugin: 'com.android.library'

android {
    namespace 'com.gainscha.jzint'
    compileSdkVersion 34

    defaultConfig {
        minSdkVersion 21
        targetSdkVersion 34

        externalNativeBuild {
            cmake {
                arguments "-DANDROID_SUPPORT_FLEXIBLE_PAGE_SIZES=ON"
            }
        }
```

```
        ndk {
            abiFilters 'arm64-v8a'
        }
    }

    externalNativeBuild {
        cmake {
            path "src/main/cpp/CMakeLists.txt"
            version "3.22.1"
        }
    }

    packaging {
        jniLibs {
            useLegacyPackaging = false
        }
    }
}
```

jzint/src/main/cpp/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(zbcode)

# 设置 16KB 页面对齐的链接器标志
set(CMAKE_SHARED_LINKER_FLAGS "${CMAKE_SHARED_LINKER_FLAGS} -Wl,-z,max-page-
size=16384")
set(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} -Wl,-z,max-page-
size=16384")

add_library(zbcode SHARED main.cpp com_gainscha_jzint_JzInt.cpp)
target_link_libraries(zbcode zint-static ${log-lib})
```

serial-port/build.gradle

```
apply plugin: 'com.android.library'

android {
    namespace 'com.gainscha.serialport'
    compileSdkVersion 34

    defaultConfig {
        minSdkVersion 21
        targetSdkVersion 34

        externalNativeBuild {
            cmake {
                arguments "-DANDROID_SUPPORT_FLEXIBLE_PAGE_SIZES=ON"
            }
        }

        ndk {
            abiFilters 'arm64-v8a'
```

```
        }
    }

    externalNativeBuild {
        cmake {
            path 'CMakeLists.txt'
            version "3.22.1"
        }
    }

    packaging {
        jniLibs {
            useLegacyPackaging = false
        }
    }
}
```

serial-port/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.4.1)
project("serial_port")

# 设置 16KB 页面对齐的链接器标志
set(CMAKE_SHARED_LINKER_FLAGS "${CMAKE_SHARED_LINKER_FLAGS} -Wl,-z,max-page-
size=16384")
set(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} -Wl,-z,max-page-
size=16384")

add_library(serial_port SHARED src/main/cpp/SerialPort.c)
target_link_libraries(serial_port ${log-lib})

# 确保 16KB 页面对齐生效
set_target_properties(serial_port PROPERTIES
    LINK_FLAGS "-Wl,-z,max-page-size=16384"
)
```

构建和验证

构建步骤

```
# 1. 清理项目
./gradlew clean

# 2. 构建应用
./gradlew assembleAllabiRelease

# 3. APK 16KB 对齐（必须）
zipalign -f -p 16 app-allabi-release.apk app-allabi-release-aligned.apk

# 4. 验证对齐
zipalign -c -v 16 app-allabi-release-aligned.apk
```

验证 Native 库对齐

```
# 提取 APK
unzip app-allabi-release-aligned.apk -d temp

# 检查 .so 文件对齐（Align 值应为 0x4000）
readelf -l temp/lib/arm64-v8a/libfimage.so | grep LOAD
readelf -l temp/lib/arm64-v8a/libserial_port.so | grep LOAD
readelf -l temp/lib/arm64-v8a/libzbcode.so | grep LOAD
```

可使用android studio自带的apk分析器分析lib下方so库文件是否对齐
教程：支持 16 KB 的页面大小 | Compatibility | Android Developers

配置清单

使用 sdk2.aar 时，必须完成以下配置：

- [ ] settings.gradle.kts 包含 fimage、jzint、serial-port 模块
- [ ] gradle.properties 配置了全局链接器标志
- [ ] app/build.gradle.kts 添加了 sdk2.aar 依赖
- [ ] app/build.gradle.kts 手动添加了 sdk2 的传递依赖
- [ ] app/build.gradle.kts 添加了原生库模块依赖
- [ ] app/build.gradle.kts 限制为 arm64-v8a 架构
- [ ] app/build.gradle.kts 配置了 packaging.jniLibs.useLegacyPackaging = true
- [ ] fimage/build.gradle 配置了 16KB 对齐
- [ ] fimage/CMakeLists.txt 配置了链接器标志
- [ ] jzint/build.gradle 配置了 16KB 对齐
- [ ] jzint/CMakeLists.txt 配置了链接器标志
- [ ] serial-port/build.gradle 配置了 16KB 对齐
- [ ] serial-port/CMakeLists.txt 配置了链接器标志
- [ ] 构建后执行了 16KB 对齐
- [ ] 使用安卓编译器的分析器查看是否对齐

关键配置总结

配置层级	配置文件	关键配置项
全局	gradle.properties	`android.native.linkFlags=-Wl,-z,max-page-size=16384`
应用	app/build.gradle.kts	`ndk.abiFilters.add("arm64-v8a")`
应用	app/build.gradle.kts	`useLegacyPackaging = true`
应用	app/build.gradle.kts	手动添加 fimage/jzint/serial-port 依赖

配置层级	配置文件	关键配置项
原生库	build.gradle	`arguments "-DANDROID_SUPPORT_FLEXIBLE_PAGE_SIZES=ON"`
原生库	build.gradle	`abiFilters 'arm64-v8a'`
原生库	CMakeLists.txt	`set(CMAKE_SHARED_LINKER_FLAGS ... -Wl,-z,max-page-size=16384)`

Q1: 为什么使用 sdk2.aar 还需要原生库模块?

A: sdk2.aar 只包含 Java/Kotlin 代码，不包含 .so 文件。.so 文件来自 fimage、jzint、serial-port 三个模块，必须以源代码模块形式存在才能在编译时应用 16KB 对齐。

Q2: 可以把 fimage、jzint、serial-port 也做成 AAR 吗?

A: 不可以。预编译的 AAR 中的 .so 文件已经是 4KB 对齐，无法在应用构建时重新配置。必须使用源代码模块。

Q3: 为什么只支持 arm64-v8a?

A:

- arm64-v8a 是 64 位架构，更容易实现 16KB 对齐
- 现代 Android 设备绝大多数都支持 arm64-v8a
- 可以根据需要后续添加其他架构支持

Q4: useLegacyPackaging 为什么要设置为 true?

A: 设置为 true 可以确保 .so 文件以未压缩方式打包到 APK 中，这样才能保持 16KB 对齐。如果压缩，对齐会被破坏。

依赖关系图

```
app
 ├── sdk2.aar (只有 Java/Kotlin 代码)
 |     └── 需要手动添加传递依赖 ↓
 ├── fimage (源代码模块)
 |     └── libfimage.so (16KB 对齐)
 ├── jzint (源代码模块)
 |     └── libzbcode.so (16KB 对齐)
 └── serial-port (源代码模块)
       └── libserial_port.so (16KB 对齐)
```
