android NDK

**ABI**

不同的android手机使用不同的CPU，因为支持不同的指令集。

**NEON**

NENO 是浮点的协处理器，用来计算算法。

-mfpu= vfp（浮点协处理器）

-mpu= neon

-march = armv7-a

**🡪使用NDK编译FFmpeg的库**

1. 准备ubuntu开发环境

进入linux,

sudo passwd 密码

sudo apt-get update // 更新软件源

apt-get openssh-server // 下载openssh,这个工具是ssh的服务端

wget <https://dl.google.com> // 使用wget下载

使用xshell连接到linux主机

二.下载ffmpeg源码

git clone <https://git.ffmpeg.org/ffmpeg.git> ffmpeg, 使用git下载ffmpeg源码

wget <http://ffmpeg.org/releases/ffmpeg-3.4.tar.bz2>

<http://www.ffmpeg.club/ffmpeg_download.html>

解压编译ffmpeg

tar –xvf ffmpeg-3.4.tar.bz2

apt-get install make

./config // 配置makefile

make –j16 // 编译的线程数量(8核16线程就开了16线程)

make install

config文件配置

--prefix 输出目录

--enable 开启模块，硬解码，neon

--disable 禁止模块，禁止ffmpeg工具

交叉编译参数 cross-prefix arch target cpu sysroot extra-cflags

🡪创建支持FFmpeg的项目

◆权限

需要读写SD卡权限：WRITE\_EXTERNAL\_STORAGE

为了避免Android6.0动态权限检测，可以将targetSdkVersion = 21

◆JNI库的路径

build.gradle中添加

sourceSets {

main {

jniLibs.srcDirs = [‘libs’]

}

}

◆配置CMake参数

externalNativeBuild {

cmake {

cppFlags “-std=c++11”

path = “CMakeLists.txt”

}

ndk {

abiFilters “armeabi-v7a”// 只加载armv7版本

}

}

◆CMake项目配置app/CMakeLists.txt

include\_directories(include) 引用的头文件路径

-DANDROID-ABI = armeabi-v7a 设置FFmpeg的库路径

set (FF\_DIR${CMAKE\_CURRENT\_SOUCE\_DIR}/libs/${ANDROID\_ABI})

创建FF\_DIR这样一个变量，括号内的大写字母也是变量，语法格式：${变量名}

cmake添加动态库 add\_library

添加源码类型的库

add\_library(native-lib SHARED src/main/cpp/native-lib.cpp)

添加已经编好的so库(如ffmpeg)

add\_library(avformat) SHARED IMPORTED // 注意：avformat 并不是so库的文件名，仅是一个符号

set\_target\_properties(avformat PROPERTIES IMPORTED\_LOCATON${FF\_DIR}/libavformat.so)

◆CMake链接动态so库（将所有用到的库全部链接到一个库当中，这样做的好处是在java中只需要引入一个native-lib库就行了）

target\_link\_libraries(#specifies the target library.

native-lib

android

avformat avcodec swscale avtuil

)

新建项目

注意点：勾选支持

一.添加头文件路径。在native-lib.cpp中引入FFmpeg的头文件。

1.1 将编译生成的include目录添加到项目的app目录下。

1.2 在native-lib.cpp中添加下列代码：

extern “C”{

#include <libavcodec/avcodec.h>

}

// 必须添加extern “C”字样，c++对函数名有重载

// 还需要指定头文件路径，否则项目找不到指定的头文件。

1.3 在CMakelist.txt中添加头文件路径(相对于本文件的路径)。

include\_directories(include) // 指定路径的名称是include

1.4 调用函数测试ffmpeg是否可以正常工作

hello += avcodec\_configuration();

1.5 加载ffmpeg的库文件，否则如果调用avcodec\_configuration()会报函数未定义。

将编译生成的so库拷贝到 项目的app/libs/armeabi-v7a目录下。

1.6 在CMakeList.txt中添加FFMpeg库所在路径的变量

set {FF ${CMAKE\_CURRENT\_SOURCE\_DIR}/libs/${ANDROID\_ABI}}

add\_library(avcodec SHARED IMPORTED )

set\_target\_properties(avcodec PROPERTIES IMPORTED\_locaton ${FF} /libavcodec.so) // 与Android.mk不同，这里还需要加lib前缀和.so后缀

1.7 在CMakeList.txt中对将要添加的库进行链接

target\_Link\_librarise

native\_lib

avcodec

1.8 在app/build.gralde中defaultConfig这一层的externNativeBuild中添加

ndk {

abiFilters “armeabi-v7a”

}

1.9 指定指定JN目录

main {

jniLibs.srcDirs = [‘libs’]

}

2.0 测试JN函数

public native boolean Open(String url, Object handle)

FILE \*fp = foen(url, “rb”);

if (!fp) {

}

FFmpeg SDK 软硬件解码基础

**解封装**

pts 是显示的时间 dts是解码的时间， 这个时间是用来做同步

1. av\_register\_all(), 注册所有的格式。包括解封装格式和加封装格式。
2. avformat\_network\_init(), FFmpeg本身也支持解封装RTSP的数据,如果要解封装网络数据格式。
3. avformat\_open\_input()。打开一个文件，并解析，可解析的内容包括：视频流、音频流、视频流参数、音频流参数、视频帧索引。
4. avformat\_find\_stream\_info()，查找格式和索引。有些早期格式它的索引并没有放到头当中，需要你到后面探测，就会用到此函数。
5. av\_find\_best\_stream(), 当视频被解封装出来后，需要分开处理音频和视频，需要找到对应的音频流和视频流
6. 解封装后涉及到的结构体：

AVFormatContext: 封装的上下文，

AVStream : 存放的是音频流或视频流的参数信息

AVPacket: 针对于具体的解封装完后的一个一个的数据包。

av\_read\_frame() 用于读取一个AVPacket,AVPacket里面包含了这个包的pts,dts,还有这个包的stream index(它是音频还是视频),是否是关键帧，AVPacket把h264数据的间隔符去掉了，

**avformat\_open\_input() 函数详解并示例打开mp4文件。int avformat\_open\_input()**

打开一个输入的封装器。在调用该函数之前，须确保av\_register\_all()和avformat\_network\_init()已调用。

参数说明：

AVFormatContext \*\*ps, 格式化的上下文。要注意，如果传入的是一个AVFormatContext\*的指针，则该空间须自己手动清理，若该指针传空，则FFmpeg会内部自己创建。

const char \*url, 传入的地址。支持http,RTSP,普通的本地文件。地址最终会存入到AVFormatContext结构体当中。

AVInputFormat \*fmt, 一般不用，指定输入的封装格式，一般传NULL，由FFmpeg自行探测。

AVDictionary \*\*options, 字典数组。用于参数传递，不传则写NULL。参见：libavformat/options\_table.h,其中包含了它支持的参数设置。

代码参考：

#include <libavformat/avformat.h>

stringFromJNI() {

// 初始化解封装

avformat\_register\_all()

// 初始化网络

avformat\_network\_init()

// 打开文件

AVFormatContext \*ic = NULL;

char path[] = “sdcard/1080.mp4”;

int ret = avformat\_open\_input(&ic, path, 0, 0);

if (ret == 0) {

LOGI(“avformat input open success”);

} else {

LOGE(“avformat\_open\_input failed. %s”av\_err2str(ret;)

}

}

提示函数未声明：

在CMakeLists.txt中添加avcodec, 和 avutil。