原 最简单的基于FFmpeg的移动端例子：Android 视频解码器-单个库版

2015年07月25日 11:42:31　阅读数：19520

==========

最简单的基于FFmpeg的移动端例子系列文章列表：

最简单的基于FFmpeg的移动端例子：Android HelloWorld

最简单的基于FFmpeg的移动端例子：Android 视频解码器

最简单的基于FFmpeg的移动端例子：Android 视频解码器-单个库版

最简单的基于FFmpeg的移动端例子：Android 推流器

最简单的基于FFmpeg的移动端例子：Android 视频转码器

最简单的基于FFmpeg的移动端例子附件：Android 自带播放器

最简单的基于FFmpeg的移动端例子附件：SDL Android HelloWorld

最简单的基于FFmpeg的移动端例子：IOS HelloWorld

最简单的基于FFmpeg的移动端例子：IOS 视频解码器

最简单的基于FFmpeg的移动端例子：IOS 推流器

最简单的基于FFmpeg的移动端例子：IOS 视频转码器

最简单的基于FFmpeg的移动端例子附件：IOS自带播放器

最简单的基于FFmpeg的移动端例子：Windows Phone HelloWorld

==========

本文记录另一个安卓平台下基于FFmpeg的视频解码器。与前一篇文章记录的解码器不同，本文记录的解码器不再使用libavcodec.so、libavformat.so等类库，而只使用了一个类库——libffmpeg.so。该视频解码器C语言的源代码来自于《 最简单的基于FFMPEG+SDL的视频播放器 》。相关的概念就不再重复记录了。

FFmpeg类库的打包

记录一下FFmpeg类库打包的方法。Android平台下FFmpeg类库一共包含下面几个：

- *libavformat-56.so*
- *libavcodec-56.so*
- *libavfilter-5.so*
- *libavdevice-56.so*
- *libavutil-54.so*
- *libpostproc-53.so*
- *libswresample-1.so*
- *libswscale-3.so*

由于数目繁多，直接使用这些类库还是比较麻烦的。因此可以将它们合并为一个类库。具体打包的命令就是下面脚本中“make install”后面的那个命令。

```
cd ffmpeg

make clean

export NDK=/home/leixiaohua1020/cdtworkspace/android-ndk-r9d
export PREBUILT=$NDK/toolchains/arm-linux-androideabi-4.8/prebuilt
export PLATFORM=$NDK/platforms/android-8/arch-arm
export PREFIX=../ff-pure-onelib
build_one(){
  ./configure --target-os=linux --prefix=$PREFIX \
--enable-cross-compile \
--enable-runtime-cpudetect \
--disable-asm \
--arch=arm \
--cc=$PREBUILT/linux-x86_64/bin/arm-linux-androideabi-gcc \
--cross-prefix=$PREBUILT/linux-x86_64/bin/arm-linux-androideabi- \
--disable-stripping \
--nm=$PREBUILT/linux-x86_64/bin/arm-linux-androideabi-nm \
--sysroot=$PLATFORM \
--enable-gpl --enable-static --disable-shared --enable-nonfree --enable-version3 --enable-small \
--enable-zlib --disable-ffprobe --disable-ffplay --disable-ffmpeg --disable-ffserver --disable-debug \
--extra-cflags="-fPIC -DANDROID -D__thumb__ -mthumb -Wfatal-errors -Wno-deprecated -mfloat-abi=softfp -marm -march=armv7-a"
}

build_one

make
make install

$PREBUILT/linux-x86_64/bin/arm-linux-androideabi-ld -rpath-link=$PLATFORM/usr/lib -L$PLATFORM/usr/lib -L$PREFIX/lib -soname libffmpeg.so -shared -nostdlib -Bsymbolic --whole-archive --no-undefined -
o $PREFIX/libffmpeg.so libavcodec/libavcodec.a libavfilter/libavfilter.a libswresample/libswresample.a libavformat/libavformat.a libavutil/libavutil.a libswscale/libswscale.a libpostproc/libpostproc.a libavdevice/libavdevice.a -lc -lm -lz -ldl -llog --dynamic-linker=/system/bin/linker $PREBUILT/linux-x86_64/lib/gcc/arm-linux-androideabi/4.8/libgcc.a

cd ..
```

```
[plain]
```

需要注意：

（1）与前面记录的脚本不同，这个脚本不再需要修改Configure的内容（生成的是*.a而不是*.so，并没有涉及到版本号问题）。

（2）前文记录的脚本里面Configure的时候是“--enable-shared --disable-static”，这个脚本里面Configure的时候设置的是“--enable-static --disable-shared”。编译完成后生成的是：

libavcodec.a

libavfilter.a

libswresample.a

libavformat.a

libavutil.a

libswscale.a

libpostproc.a

libavdevice.a

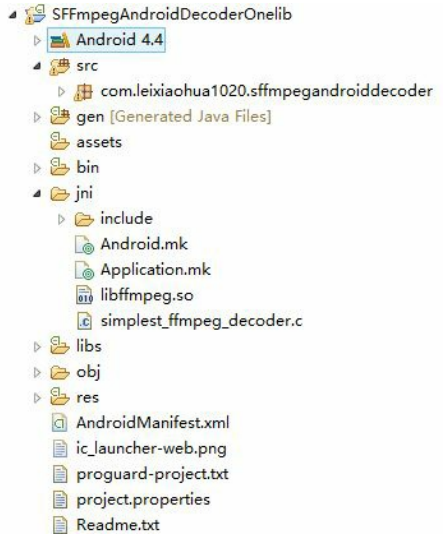
该脚本运行完后，会把上述的*.a文件打包为1个*.so文件：

libffmpeg.so

合并后的类库使用起来和合并前的类库使用方法没有区别。

源代码

项目的目录结构如图所示。Java源代码位于src目录，而C代码位于jni目录。



Android程序Java端代码位于src\com\leixiaohua1020\sffmpegandroiddecoder\MainActivity.java，如下所示。

```java
/**
 * 最简单的基于FFmpeg的视频解码器-安卓 - 单库版
 * Simplest FFmpeg Android Decoder - One Library
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序是安卓平台下最简单的基于FFmpeg的视频解码器。
 * 它可以将输入的视频数据解码成YUV像素数据。
 *
 * This software is the simplest decoder based on FFmpeg in Android.
 * It can decode video stream to raw YUV data.
 *
 */
package com.leixiaohua1020.sffmpegandroiddecoder;


import android.os.Bundle;
import android.os.Environment;
import android.app.Activity;
import android.text.Editable;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button startButton = (Button) this.findViewById(R.id.button_start);
        final EditText urlEdittext_input= (EditText) this.findViewById(R.id.input_url);
        final EditText urlEdittext_output= (EditText) this.findViewById(R.id.output_url);

        startButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0){

                String folderurl=Environment.getExternalStorageDirectory().getPath();

                String urltext_input=urlEdittext_input.getText().toString();
                String inputurl=folderurl+"/"+urltext_input;

                String urltext_output=urlEdittext_output.getText().toString();
                String outputurl=folderurl+"/"+urltext_output;

                Log.i("inputurl",inputurl);
                Log.i("outputurl",outputurl);

                decode(inputurl,outputurl);

            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    //JNI
    public native int decode(String inputurl, String outputurl);

    static{
        System.loadLibrary("ffmpeg");
        System.loadLibrary("sffdecoder");
    }
}
```

C语言端源代码位于jni/simplest_ffmpeg_decoder.c，如下所示。

```cpp
/**
 * 最简单的基于FFmpeg的视频解码器-安卓 - 单库版
 * Simplest FFmpeg Android Decoder - One Library
 *
```

```
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序是安卓平台下最简单的基于FFmpeg的视频解码器。
 * 它可以将输入的视频数据解码成YUV像素数据。
 *
 * This software is the simplest decoder based on FFmpeg in Android.
 * It can decode video stream to raw YUV data.
 *
 */


#include <stdio.h>
#include <time.h>

#include "libavcodec/avcodec.h"
#include "libavformat/avformat.h"
#include "libswscale/swscale.h"
#include "libavutil/log.h"

#ifdef ANDROID
#include <jni.h>
#include <android/log.h>
#define LOGE(format, ...)  __android_log_print(ANDROID_LOG_ERROR, "(>_<)", format, ##__VA_ARGS__)
#define LOGI(format, ...)  __android_log_print(ANDROID_LOG_INFO,  "(^_^)", format, ##__VA_ARGS__)
#else
#define LOGE(format, ...)  printf("(>_<) " format "\n", ##__VA_ARGS__)
#define LOGI(format, ...)  printf("(^_^) " format "\n", ##__VA_ARGS__)
#endif


//Output FFmpeg's av_log()
void custom_log(void *ptr, int level, const char* fmt, va_list vl){
    FILE *fp=fopen("/storage/emulated/0/av_log.txt","a+");
    if(fp){
        vfprintf(fp,fmt,vl);
        fflush(fp);
        fclose(fp);
    }
}

JNIEXPORT jint JNICALL Java_com_leixiaohua1020_sffmpegandroiddecoder_MainActivity_decode
  (JNIEnv *env, jobject obj, jstring input_jstr, jstring output_jstr)
{
    AVFormatContext *pFormatCtx;
    int             i, videoindex;
    AVCodecContext  *pCodecCtx;
    AVCodec         *pCodec;
    AVFrame *pFrame,*pFrameYUV;
    uint8_t *out_buffer;
    AVPacket *packet;
    int y_size;
    int ret, got_picture;
    struct SwsContext *img_convert_ctx;
    FILE *fp_yuv;
    int frame_cnt;
    clock_t time_start, time_finish;
    double  time_duration = 0.0;

    char input_str[500]={0};
    char output_str[500]={0};
    char info[1000]={0};
    sprintf(input_str,"%s",(*env)->GetStringUTFChars(env,input_jstr, NULL));
    sprintf(output_str,"%s",(*env)->GetStringUTFChars(env,output_jstr, NULL));

    //FFmpeg av_log() callback
  av_log_set_callback(custom_log);

    av_register_all();
    avformat_network_init();
    pFormatCtx = avformat_alloc_context();

    if(avformat_open_input(&pFormatCtx,input_str,NULL,NULL)!=0){
        LOGE("Couldn't open input stream.\n");
        return -1;
    }
    if(avformat_find_stream_info(pFormatCtx,NULL)<0){
        LOGE("Couldn't find stream information.\n");
        return -1;
    }
    videoindex=-1;
    for(i=0; i<pFormatCtx->nb_streams; i++)
        if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
            videoindex=i;
            break;
        }
    if(videoindex==-1){
        LOGE("Couldn't find a video stream.\n");
```

```
        return -1;
    }
    pCodecCtx=pFormatCtx->streams[videoindex]->codec;
    pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
    if(pCodec==NULL){
        LOGE("Couldn't find Codec.\n");
        return -1;
    }
    if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
        LOGE("Couldn't open codec.\n");
        return -1;
    }

    pFrame=av_frame_alloc();
    pFrameYUV=av_frame_alloc();
    out_buffer=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P,  pCodecCtx->width, pCodecCtx->height,1));
    av_image_fill_arrays(pFrameYUV->data, pFrameYUV->linesize,out_buffer,
        AV_PIX_FMT_YUV420P,pCodecCtx->width, pCodecCtx->height,1);

    packet=(AVPacket *)av_malloc(sizeof(AVPacket));

    img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
    pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);


  sprintf(info,   "[Input     ]%s\n", input_str);
  sprintf(info, "%s[Output    ]%s\n",info,output_str);
  sprintf(info, "%s[Format    ]%s\n",info, pFormatCtx->iformat->name);
  sprintf(info, "%s[Codec     ]%s\n",info, pCodecCtx->codec->name);
  sprintf(info, "%s[Resolution]%dx%d\n",info, pCodecCtx->width,pCodecCtx->height);


  fp_yuv=fopen(output_str,"wb+");
  if(fp_yuv==NULL){
        printf("Cannot open output file.\n");
        return -1;
    }

    frame_cnt=0;
    time_start = clock();

    while(av_read_frame(pFormatCtx, packet)>=0){
        if(packet->stream_index==videoindex){
            ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
            if(ret < 0){
                LOGE("Decode Error.\n");
                return -1;
            }
            if(got_picture){
                sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
                    pFrameYUV->data, pFrameYUV->linesize);

                y_size=pCodecCtx->width*pCodecCtx->height;
                fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
                fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
                fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
                //Output info
                char pictype_str[10]={0};
                switch(pFrame->pict_type){
                    case AV_PICTURE_TYPE_I:sprintf(pictype_str,"I");break;
                  case AV_PICTURE_TYPE_P:sprintf(pictype_str,"P");break;
                    case AV_PICTURE_TYPE_B:sprintf(pictype_str,"B");break;
                    default:sprintf(pictype_str,"Other");break;
                }
                LOGI("Frame Index: %5d. Type:%s",frame_cnt,pictype_str);
                frame_cnt++;
            }
        }
        av_free_packet(packet);
    }
    //flush decoder
    //FIX: Flush Frames remained in Codec
    while (1) {
        ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
        if (ret < 0)
            break;
        if (!got_picture)
            break;
        sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
            pFrameYUV->data, pFrameYUV->linesize);
        int y_size=pCodecCtx->width*pCodecCtx->height;
        fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
        fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
        fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
        //Output info
        char pictype_str[10]={0};
        switch(pFrame->pict_type){
            case AV_PICTURE_TYPE_I:sprintf(pictype_str,"I");break;
          case AV_PICTURE_TYPE_P:sprintf(pictype_str,"P");break;
            case AV_PICTURE_TYPE_B:sprintf(pictype_str,"B");break;
            default:sprintf(pictype_str,"Other");break;
```

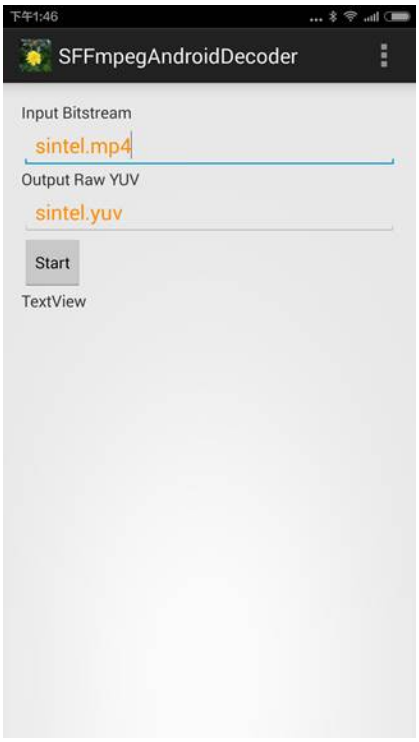
```
        }
        LOGI("Frame Index: %5d. Type:%s",frame_cnt,pictype_str);
        frame_cnt++;
    }
    time_finish = clock();
    time_duration=(double)(time_finish - time_start);

    sprintf(info, "%s[Time      ]%fms\n",info,time_duration);
    sprintf(info, "%s[Count     ]%d\n",info,frame_cnt);

    sws_freeContext(img_convert_ctx);

  fclose(fp_yuv);

    av_frame_free(&pFrameYUV);
    av_frame_free(&pFrame);
    avcodec_close(pCodecCtx);
    avformat_close_input(&pFormatCtx);

    return 0;
}
```

Android.mk文件位于jni/Android.mk，如下所示。

```
[plain]
# Android.mk for FFmpeg
#
# Lei Xiaohua 雷霄骅
# leixiaohua1020@126.com
# http://blog.csdn.net/leixiaohua1020
#

LOCAL_PATH := $(call my-dir)

# FFmpeg library
include $(CLEAR_VARS)
LOCAL_MODULE := ffmpeg
LOCAL_SRC_FILES := libffmpeg.so
include $(PREBUILT_SHARED_LIBRARY)


# Program
include $(CLEAR_VARS)
LOCAL_MODULE := sffdecoder
LOCAL_SRC_FILES :=simplest_ffmpeg_decoder.c
LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
LOCAL_LDLIBS := -llog -lz
LOCAL_SHARED_LIBRARIES := ffmpeg
include $(BUILD_SHARED_LIBRARY)
```

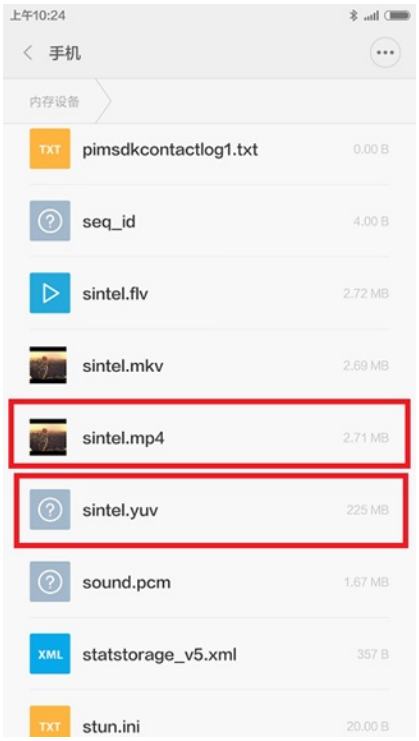
运行结果

App在手机上运行后的结果如下图所示。



注意需要把等待解码的视频文件拷贝至存储卡相应的目录中。例如对于上述截图的情况，需要将sintel.mp4拷贝至存储卡的根目录中。

单击“Start”按钮就可以将存储卡根目录中的视频文件解码为YUV文件（需要等待一段时间完成解码）。注意解码后的YUV文件体积巨大，可能会占用大量的存储卡空间。



下载

simplest ffmpeg mobile

项目主页

Github： https://github.com/leixiaohua1020/simplest_ffmpeg_mobile

开源中国： https://git.oschina.net/leixiaohua1020/simplest_ffmpeg_mobile

SourceForge： https://sourceforge.net/projects/simplestffmpegmobile/

CSDN工程下载地址： http://download.csdn.net/detail/leixiaohua1020/8924391

本解决方案包含了使用FFmpeg在移动端处理多媒体的各种例子：

[Android]

simplest_android_player: 基于安卓接口的视频播放器

simplest_ffmpeg_android_helloworld: 安卓平台下基于FFmpeg的HelloWorld程序

simplest_ffmpeg_android_decoder: 安卓平台下最简单的基于FFmpeg的视频解码器

simplest_ffmpeg_android_decoder_onelib: 安卓平台下最简单的基于FFmpeg的视频解码器-单库版

simplest_ffmpeg_android_streamer: 安卓平台下最简单的基于FFmpeg的推流器

simplest_ffmpeg_android_transcoder: 安卓平台下移植的FFmpeg命令行工具

simplest_sdl_android_helloworld: 移植SDL到安卓平台的最简单程序

[IOS]

simplest_ios_player: 基于IOS接口的视频播放器

simplest_ffmpeg_ios_helloworld: IOS平台下基于FFmpeg的HelloWorld程序

simplest_ffmpeg_ios_decoder: IOS平台下最简单的基于FFmpeg的视频解码器

simplest_ffmpeg_ios_streamer: IOS平台下最简单的基于FFmpeg的推流器

simplest_ffmpeg_ios_transcoder: IOS平台下移植的ffmpeg.c命令行工具

simplest_sdl_ios_helloworld: 移植SDL到IOS平台的最简单程序

文章标签： FFmpeg Android 视频解码 YUV JNI

个人分类： FFMPEG Android多媒体

所属专栏： FFmpeg