

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。

```
[plain]
1. cd ffmpeg
2.
3. make clean
4.
5. export NDK=/home/leixiaohua1020/cdtworkspace/android-ndk-r9d
6. export PREBUILT=$NDK/toolchains/arm-linux-androideabi-4.8/prebuilt
7. export PLATFORM=$NDK/platforms/android-8/arch-arm
8. export PREFIX=./ff-pure-onelib
9. build_one(){
10.     ./configure --target-os=linux --prefix=$PREFIX \
11.     --enable-cross-compile \
12.     --enable-runtime-cpudetect \
13.     --disable-asm \
14.     --arch=arm \
15.     --cc=$PREBUILT/linux-x86_64/bin/arm-linux-androideabi-gcc \
16.     --cross-prefix=$PREBUILT/linux-x86_64/bin/arm-linux-androideabi- \
17.     --disable-stripping \
18.     --nm=$PREBUILT/linux-x86_64/bin/arm-linux-androideabi-nm \
19.     --sysroot=$PLATFORM \
20.     --enable-gpl --enable-static --disable-shared --enable-nonfree --enable-version3 --enable-small \
21.     --enable-zlib --disable-ffprobe --disable-ffplay --disable-ffmpeg --disable-ffserver --disable-debug \
22.     --extra-cflags="-fPIC -DANDROID -D__thumb__ -mthumb -Wfatal-errors -Wno-deprecated -mfloat-abi=softfp -marm -march=armv7-a"
23. }
24.
25. build_one
26.
27. make
28. make install
29.
30. $PREBUILT/linux-x86_64/bin/arm-linux-androideabi-ld -rpath-link=$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
31.
32. cd ..
```

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

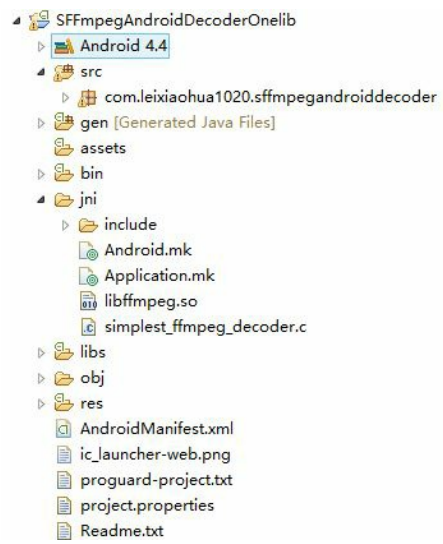
该脚本运行完后，会把上述的*.a文件打包为1个*.so文件：

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。

```

1.  /**
2.   * 最简单的基于FFmpeg的视频解码器-安卓 - 单库版
3.   * Simplest FFmpeg Android Decoder - One Library
4.   *
5.   * 雷霄骅 Lei Xiaohua
6.   * leixiaohua1020@126.com
7.   * 中国传媒大学/数字电视技术
8.   * Communication University of China / Digital TV Technology
9.   * http://blog.csdn.net/leixiaohua1020
10.  *
11.  * 本程序是安卓平台下最简单的基于FFmpeg的视频解码器。
12.  * 它可以将输入的视频数据解码成YUV像素数据。
13.  *
14.  * This software is the simplest decoder based on FFmpeg in Android.
15.  * It can decode video stream to raw YUV data.
16.  *
17.  */
18.  package com.leixiaohua1020.sffmpegandroiddecoder;
19.
20.
21.  import android.os.Bundle;
22.  import android.os.Environment;
23.  import android.app.Activity;
24.  import android.text.Editable;
25.  import android.util.Log;
26.  import android.view.Menu;
27.  import android.view.View;
28.  import android.view.View.OnClickListener;
29.  import android.widget.Button;
30.  import android.widget.EditText;
31.  import android.widget.TextView;
32.
33.  public class MainActivity extends Activity {
34.
35.
36.      @Override
37.      protected void onCreate(Bundle savedInstanceState) {
38.          super.onCreate(savedInstanceState);
39.          setContentView(R.layout.activity_main);
40.
41.          Button startButton = (Button) this.findViewById(R.id.button_start);
42.          final EditText urlEditText_input= (EditText) this.findViewById(R.id.input_url);
43.          final EditText urlEditText_output= (EditText) this.findViewById(R.id.output_url);
44.
45.          startButton.setOnClickListener(new OnClickListener() {
46.              public void onClick(View arg0){
47.
48.                  String folderurl=Environment.getExternalStorageDirectory().getPath();
49.
50.                  String urltext_input=urlEditText_input.getText().toString();
51.                  String inputurl=folderurl+"/"+urltext_input;
52.
53.                  String urltext_output=urlEditText_output.getText().toString();
54.                  String outputurl=folderurl+"/"+urltext_output;
55.
56.                  Log.i("inputurl",inputurl);
57.                  Log.i("outputurl",outputurl);
58.
59.                  decode(inputurl,outputurl);
60.
61.              }
62.          });
63.      }
64.
65.      @Override
66.      public boolean onCreateOptionsMenu(Menu menu) {
67.          // Inflate the menu; this adds items to the action bar if it is present.
68.          getMenuInflater().inflate(R.menu.main, menu);
69.          return true;
70.      }
71.
72.      //JNI
73.      public native int decode(String inputurl, String outputurl);
74.
75.      static{
76.          System.loadLibrary("ffmpeg");
77.          System.loadLibrary("sffdecoder");
78.      }
79.  }

```

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

```

1.  /**
2.   * 最简单的基于FFmpeg的视频解码器-安卓 - 单库版
3.   * Simplest FFmpeg Android Decoder - One Library
4.   *

```

```

5.  * 雷霄骅 Lei Xiaohua
6.  * leixiaohua1020@126.com
7.  * 中国传媒大学/数字电视技术
8.  * Communication University of China / Digital TV Technology
9.  * http://blog.csdn.net/leixiaohua1020
10. *
11. * 本程序是安卓平台下最简单的基于FFmpeg的视频解码器。
12. * 它可以将输入的视频数据解码成YUV像素数据。
13. *
14. * This software is the simplest decoder based on FFmpeg in Android.
15. * It can decode video stream to raw YUV data.
16. *
17. */
18.
19.
20. #include <stdio.h>
21. #include <time.h>
22.
23. #include "libavcodec/avcodec.h"
24. #include "libavformat/avformat.h"
25. #include "libswscale/swscale.h"
26. #include "libavutil/log.h"
27.
28. #ifdef ANDROID
29. #include <jni.h>
30. #include <android/log.h>
31. #define LOGE(format, ...) __android_log_print(ANDROID_LOG_ERROR, ">_<", format, ##__VA_ARGS__)
32. #define LOGI(format, ...) __android_log_print(ANDROID_LOG_INFO, "^_^", format, ##__VA_ARGS__)
33. #else
34. #define LOGE(format, ...) printf(">_< " format "\n", ##__VA_ARGS__)
35. #define LOGI(format, ...) printf("^_^ " format "\n", ##__VA_ARGS__)
36. #endif
37.
38.
39. //Output FFmpeg's av_log()
40. void custom_log(void *ptr, int level, const char* fmt, va_list vl){
41.     FILE *fp=fopen("/storage/emulated/0/av_log.txt","a+");
42.     if(fp){
43.         vfprintf(fp,fmt,vl);
44.         fflush(fp);
45.         fclose(fp);
46.     }
47. }
48.
49. JNIEXPORT jint JNICALL Java_com_leixiaohua1020_sffmpegandroiddecoder_MainActivity_decode
50. (JNIEnv *env, jobject obj, jstring input_jstr, jstring output_jstr)
51. {
52.     AVFormatContext *pFormatCtx;
53.     int i, videoindex;
54.     AVCodecContext *pCodecCtx;
55.     AVCodec *pCodec;
56.     AVFrame *pFrame,*pFrameYUV;
57.     uint8_t *out_buffer;
58.     AVPacket *packet;
59.     int y_size;
60.     int ret, got_picture;
61.     struct SwsContext *img_convert_ctx;
62.     FILE *fp_yuv;
63.     int frame_cnt;
64.     clock_t time_start, time_finish;
65.     double time_duration = 0.0;
66.
67.     char input_str[500]={0};
68.     char output_str[500]={0};
69.     char info[1000]={0};
70.     sprintf(input_str,"%s",(*env)->GetStringUTFChars(env,input_jstr, NULL));
71.     sprintf(output_str,"%s",(*env)->GetStringUTFChars(env,output_jstr, NULL));
72.
73.     //FFmpeg av_log() callback
74.     av_log_set_callback(custom_log);
75.
76.     av_register_all();
77.     avformat_network_init();
78.     pFormatCtx = avformat_alloc_context();
79.
80.     if(avformat_open_input(&pFormatCtx,input_str,NULL,NULL)!=0){
81.         LOGE("Couldn't open input stream.\n");
82.         return -1;
83.     }
84.     if(avformat_find_stream_info(pFormatCtx,NULL)<0){
85.         LOGE("Couldn't find stream information.\n");
86.         return -1;
87.     }
88.     videoindex=-1;
89.     for(i=0; i<pFormatCtx->nb_streams; i++){
90.         if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
91.             videoindex=i;
92.             break;
93.         }
94.     }
95.     if(videoindex==-1){
96.         LOGE("Couldn't find a video stream.\n");

```

```

96.     return -1;
97. }
98. pCodecCtx=pFormatCtx->streams[videoindex]->codec;
99. pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
100. if(pCodec==NULL){
101.     LOGE("Couldn't find Codec.\n");
102.     return -1;
103. }
104. if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
105.     LOGE("Couldn't open codec.\n");
106.     return -1;
107. }
108.
109. pFrame=av_frame_alloc();
110. pFrameYUV=av_frame_alloc();
111. out_buffer=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height,1));
112. av_image_fill_arrays(pFrameYUV->data, pFrameYUV->linesize,out_buffer,
113.     AV_PIX_FMT_YUV420P,pCodecCtx->width, pCodecCtx->height,1);
114.
115. packet=(AVPacket *)av_malloc(sizeof(AVPacket));
116.
117. img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
118.     pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
119.
120.
121. sprintf(info, "[Input    ]s\n", input_str);
122. sprintf(info, "%s[Output  ]s\n",info,output_str);
123. sprintf(info, "%s[Format   ]s\n",info, pFormatCtx->iformat->name);
124. sprintf(info, "%s[Codec    ]s\n",info, pCodecCtx->codec->name);
125. sprintf(info, "%s[Resolution]dx%d\n",info, pCodecCtx->width,pCodecCtx->height);
126.
127.
128. fp_yuv=fopen(output_str,"wb+");
129. if(fp_yuv==NULL){
130.     printf("Cannot open output file.\n");
131.     return -1;
132. }
133.
134. frame_cnt=0;
135. time_start = clock();
136.
137. while(av_read_frame(pFormatCtx, packet)>=0){
138.     if(packet->stream_index==videoindex){
139.         ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
140.         if(ret < 0){
141.             LOGE("Decode Error.\n");
142.             return -1;
143.         }
144.         if(got_picture){
145.             sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
146.                 pFrameYUV->data, pFrameYUV->linesize);
147.
148.             y_size=pCodecCtx->width*pCodecCtx->height;
149.             fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
150.             fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
151.             fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
152.             //Output info
153.             char pictype_str[10]={0};
154.             switch(pFrame->pict_type){
155.                 case AV_PICTURE_TYPE_I:sprintf(pictype_str,"I");break;
156.                 case AV_PICTURE_TYPE_P:sprintf(pictype_str,"P");break;
157.                 case AV_PICTURE_TYPE_B:sprintf(pictype_str,"B");break;
158.                 default:sprintf(pictype_str,"Other");break;
159.             }
160.             LOGI("Frame Index: %5d. Type:%s",frame_cnt,pictype_str);
161.             frame_cnt++;
162.         }
163.     }
164.     av_free_packet(packet);
165. }
166. //flush decoder
167. //FIX: Flush Frames remained in Codec
168. while (1) {
169.     ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
170.     if (ret < 0)
171.         break;
172.     if (!got_picture)
173.         break;
174.     sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
175.         pFrameYUV->data, pFrameYUV->linesize);
176.     int y_size=pCodecCtx->width*pCodecCtx->height;
177.     fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
178.     fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
179.     fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
180.     //Output info
181.     char pictype_str[10]={0};
182.     switch(pFrame->pict_type){
183.         case AV_PICTURE_TYPE_I:sprintf(pictype_str,"I");break;
184.         case AV_PICTURE_TYPE_P:sprintf(pictype_str,"P");break;
185.         case AV_PICTURE_TYPE_B:sprintf(pictype_str,"B");break;
186.         default:sprintf(pictype_str,"Other");break;

```

```

187.     }
188.     LOGI("Frame Index: %5d. Type:%s", frame_cnt, pictype_str);
189.     frame_cnt++;
190. }
191. time_finish = clock();
192. time_duration=(double)(time_finish - time_start);
193.
194. sprintf(info, "%s[Time      ]%fms\n", info, time_duration);
195. sprintf(info, "%s[Count      ]%d\n", info, frame_cnt);
196.
197.     sws_freeContext(img_convert_ctx);
198.
199.     fclose(fp_yuv);
200.
201.     av_frame_free(&pFrameYUV);
202.     av_frame_free(&pFrame);
203.     avcodec_close(pCodecCtx);
204.     avformat_close_input(&pFormatCtx);
205.
206.     return 0;
207. }

```

Android.mk文件位于jni/Android.mk，如下所示。

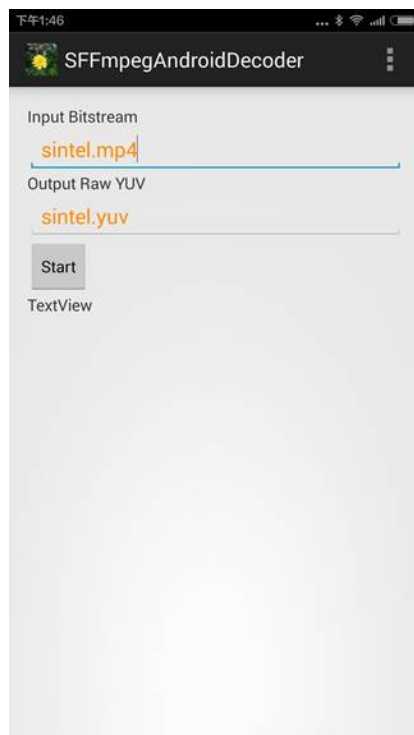
```

[plain]
1.  # Android.mk for FFmpeg
2.  #
3.  # Lei Xiaohua  雷霄骅
4.  # leixiaohua1020@126.com
5.  # http://blog.csdn.net/leixiaohua1020
6.  #
7.
8.  LOCAL_PATH := $(call my-dir)
9.
10. # FFmpeg library
11. include $(CLEAR_VARS)
12. LOCAL_MODULE := ffmpeg
13. LOCAL_SRC_FILES := libffmpeg.so
14. include $(PREBUILT_SHARED_LIBRARY)
15.
16.
17. # Program
18. include $(CLEAR_VARS)
19. LOCAL_MODULE := sffdecoder
20. LOCAL_SRC_FILES :=simplest_ffmpeg_decoder.c
21. LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
22. LOCAL_LDLIBS := -llog -lz
23. LOCAL_SHARED_LIBRARIES := ffmpeg
24. include $(BUILD_SHARED_LIBRARY)

```

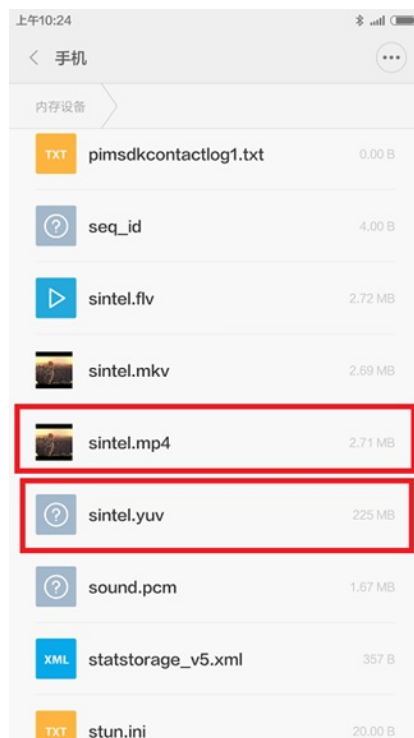
运行结果

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

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/47011021>

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

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com