

原 最简单的基于FFmpeg的移动端例子：Android 推流器

2015年07月25日 14:10:50 阅读数：38274

=====

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](#)

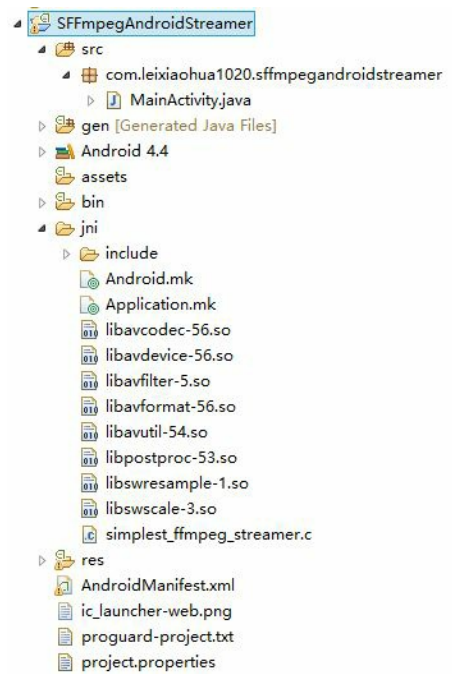
=====

本文记录一个安卓平台下基于FFmpeg的视频推流器。该推流器C语言的源代码来自于《[最简单的基于FFMPEG的推流器](#)》。相关的概念就不再重复记录了。



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\sffmpegandroidstreamer\MainActivity.java，如下所示。

[java]

```
1.  /**
2.   * 最简单的基于FFmpeg的推流器(RTMP)-安卓
3.   * Simplest FFmpeg Android Streamer (RTMP)
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的推流器。
12.  * 它可以将视频文件以流媒体的形式推送到服务器。
13.  *
14.  * This software is the simplest streamer based on FFmpeg in Android.
15.  * It can stream local media file to streaming media server (in RTMP).
16.  *
17.  */
18. package com.leixiaohua1020.sffmpegandroidstreamer;
19.
20. import android.os.Bundle;
21. import android.os.Environment;
22. import android.app.Activity;
23. import android.util.Log;
24. import android.view.Menu;
25. import android.view.View;
26. import android.view.View.OnClickListener;
27. import android.widget.Button;
28. import android.widget.EditText;
29.
30. public class MainActivity extends Activity {
31.
32.     @Override
33.     protected void onCreate(Bundle savedInstanceState) {
34.         super.onCreate(savedInstanceState);
35.         setContentView(R.layout.activity_main);
36.
37.         Button startButton = (Button) this.findViewById(R.id.button_start);
38.         final EditText urlEditText_input= (EditText) this.findViewById(R.id.input_url);
39.         final EditText urlEditText_output= (EditText) this.findViewById(R.id.output_url);
40.
41.         startButton.setOnClickListener(new OnClickListener() {
42.             public void onClick(View arg0){
43.
44.                 String folderurl=Environment.getExternalStorageDirectory().getPath();
45.
46.                 String urltext_input=urlEditText_input.getText().toString();
47.                 String inputurl=folderurl+"/"+urltext_input;
48.
49.                 String outputurl=urlEditText_output.getText().toString();
50.
51.                 Log.e("inputurl",inputurl);
52.                 Log.e("outputurl",outputurl);
53.                 String info="";
54.
55.                 stream(inputurl,outputurl);
56.
57.                 Log.e("Info",info);
58.             }
59.         });
60.     }
61.
62.
63.     @Override
64.     public boolean onCreateOptionsMenu(Menu menu) {
65.         // Inflate the menu; this adds items to the action bar if it is present.
66.         getMenuInflater().inflate(R.menu.main, menu);
67.         return true;
68.     }
69.
70.     //JNI
71.     public native int stream(String inputurl, String outputurl);
72.
73.     static{
74.         System.loadLibrary("avutil-54");
75.         System.loadLibrary("swresample-1");
76.         System.loadLibrary("avcodec-56");
77.         System.loadLibrary("avformat-56");
78.         System.loadLibrary("swscale-3");
79.         System.loadLibrary("postproc-53");
80.         System.loadLibrary("avfilter-5");
81.         System.loadLibrary("avdevice-56");
82.         System.loadLibrary("sffstreamer");
83.     }
84.
85.
86. }
```

[cpp]  

```
1.  /**
2.   * 最简单的基于FFmpeg的推流器-安卓
3.   * Simplest FFmpeg Android Streamer
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的流媒体推送器。
12.  * 它可以将视频数据以流媒体的形式发送出去。
13.  *
14.  * This software is the simplest streamer based on FFmpeg in Android.
15.  * It can stream local media file to streaming media server (in RTMP).
16.  */
17.
18.
19. #include <stdio.h>
20. #include <time.h>
21.
22. #include "libavcodec/avcodec.h"
23. #include "libavformat/avformat.h"
24. #include "libavutil/log.h"
25.
26. #ifdef ANDROID
27. #include <jni.h>
28. #include <android/log.h>
29. #define LOGE(format, ...) __android_log_print(ANDROID_LOG_ERROR, ">_<", format, ##_VA_ARGS_)
30. #define LOGI(format, ...) __android_log_print(ANDROID_LOG_INFO, "^_^", format, ##_VA_ARGS_)
31. #else
32. #define LOGE(format, ...) printf(">_< " format "\n", ##_VA_ARGS_)
33. #define LOGI(format, ...) printf("^_^ " format "\n", ##_VA_ARGS_)
34. #endif
35.
36.
37. //Output FFmpeg's av_log()
38. void custom_log(void *ptr, int level, const char* fmt, va_list vl){
39.
40.     //To TXT file
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.     //To Logcat
49.     //LOGE(fmt, vl);
50. }
51.
52. JNIEXPORT jint JNICALL Java_com_leixiaohua1020_sffmpegandroidstreamer_MainActivity_stream
53. (JNIEnv *env, jobject obj, jstring input_jstr, jstring output_jstr)
54. {
55.     AVOutputFormat *ofmt = NULL;
56.     AVFormatContext *ifmt_ctx = NULL, *ofmt_ctx = NULL;
57.     AVPacket pkt;
58.
59.     int ret, i;
60.     char input_str[500]={0};
61.     char output_str[500]={0};
62.     char info[1000]={0};
63.     sprintf(input_str,"%s",(*env)->GetStringUTFChars(env,input_jstr, NULL));
64.     sprintf(output_str,"%s",(*env)->GetStringUTFChars(env,output_jstr, NULL));
65.
66.     //input_str = "cuc_ieschool.flv";
67.     //output_str = "rtmp://localhost/publishlive/livestream";
68.     //output_str = "rtmp://233.233.233.233:6666";
69.
70.     //FFmpeg av_log() callback
71.     av_log_set_callback(custom_log);
72.
73.     av_register_all();
74.     //Network
75.     avformat_network_init();
76.
77.     //Input
78.     if ((ret = avformat_open_input(&ifmt_ctx, input_str, 0, 0)) < 0) {
79.         LOGE( "Could not open input file.");
80.         goto end;
81.     }
82.     if ((ret = avformat_find_stream_info(ifmt_ctx, 0)) < 0) {
83.         LOGE( "Failed to retrieve input stream information");
84.         goto end;
85.     }
86.
87.     int videoindex=-1;
88.     for(i=0; i<ifmt_ctx->nb_streams; i++)
89.         if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
90.             videoindex=i;
91.         }
```

```

90.         videoindex--,
91.         break;
92.     }
93.     //Output
94.     avformat_alloc_output_context2(&ofmt_ctx, NULL, "flv", output_str); //RTMP
95.     //avformat_alloc_output_context2(&ofmt_ctx, NULL, "mpegts", output_str); //UDP
96.
97.     if (!ofmt_ctx) {
98.         LOGE( "Could not create output context\n");
99.         ret = AERROR_UNKNOWN;
100.        goto end;
101.    }
102.    ofmt = ofmt_ctx->oformat;
103.    for (i = 0; i < ifmt_ctx->nb_streams; i++) {
104.        //Create output AVStream according to input AVStream
105.        AVStream *in_stream = ifmt_ctx->streams[i];
106.        AVStream *out_stream = avformat_new_stream(ofmt_ctx, in_stream->codec->codec);
107.        if (!out_stream) {
108.            LOGE( "Failed allocating output stream\n");
109.            ret = AERROR_UNKNOWN;
110.            goto end;
111.        }
112.        //Copy the settings of AVCodecContext
113.        ret = avcodec_copy_context(out_stream->codec, in_stream->codec);
114.        if (ret < 0) {
115.            LOGE( "Failed to copy context from input to output stream codec context\n");
116.            goto end;
117.        }
118.        out_stream->codec->codec_tag = 0;
119.        if (ofmt_ctx->oformat->flags & AVFMT_GLOBALHEADER)
120.            out_stream->codec->flags |= CODEC_FLAG_GLOBAL_HEADER;
121.    }
122.
123.    //Open output URL
124.    if (!(ofmt->flags & AVFMT_NOFILE)) {
125.        ret = avio_open(&ofmt_ctx->pb, output_str, AVIO_FLAG_WRITE);
126.        if (ret < 0) {
127.            LOGE( "Could not open output URL '%s'", output_str);
128.            goto end;
129.        }
130.    }
131.    //Write file header
132.    ret = avformat_write_header(ofmt_ctx, NULL);
133.    if (ret < 0) {
134.        LOGE( "Error occurred when opening output URL\n");
135.        goto end;
136.    }
137.
138.    int frame_index=0;
139.
140.    int64_t start_time=av_gettime();
141.    while (1) {
142.        AVStream *in_stream, *out_stream;
143.        //Get an AVPacket
144.        ret = av_read_frame(ifmt_ctx, &pkt);
145.        if (ret < 0)
146.            break;
147.        //FIX: No PTS (Example: Raw H.264)
148.        //Simple Write PTS
149.        if(pkt.pts==AV_NOPTS_VALUE){
150.            //Write PTS
151.            AVRational time_base1=ifmt_ctx->streams[videoindex]->time_base;
152.            //Duration between 2 frames (us)
153.            int64_t calc_duration=(double)AV_TIME_BASE/av_q2d(ifmt_ctx->streams[videoindex]->r_frame_rate);
154.            //Parameters
155.            pkt.pts=(double)(frame_index*calc_duration)/(double)(av_q2d(time_base1)*AV_TIME_BASE);
156.            pkt.dts=pkt.pts;
157.            pkt.duration=(double)calc_duration/(double)(av_q2d(time_base1)*AV_TIME_BASE);
158.        }
159.        //Important:Delay
160.        if(pkt.stream_index==videoindex){
161.            AVRational time_base=ifmt_ctx->streams[videoindex]->time_base;
162.            AVRational time_base_q={1,AV_TIME_BASE};
163.            int64_t pts_time = av_rescale_q(pkt.dts, time_base, time_base_q);
164.            int64_t now_time = av_gettime() - start_time;
165.            if (pts_time > now_time)
166.                av_usleep(pts_time - now_time);
167.        }
168.    }
169.
170.    in_stream = ifmt_ctx->streams[pkt.stream_index];
171.    out_stream = ofmt_ctx->streams[pkt.stream_index];
172.    /* copy packet */
173.    //Convert PTS/DTS
174.    pkt.pts = av_rescale_q_rnd(pkt.pts, in_stream->time_base, out_stream->time_base, AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX);
175.    pkt.dts = av_rescale_q_rnd(pkt.dts, in_stream->time_base, out_stream->time_base, AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX);
176.    pkt.duration = av_rescale_q(pkt.duration, in_stream->time_base, out_stream->time_base);
177.    pkt.pos = -1;
178.    //Print to Screen
179.    if(pkt.stream_index==videoindex){
180.        LOGE("Send %8d video frames to output URL\n",frame_index);
181.        frame_index++;

```

```

182.     }
183.     //ret = av_write_frame(ofmt_ctx, &pkt);
184.     ret = av_interleaved_write_frame(ofmt_ctx, &pkt);
185.
186.     if (ret < 0) {
187.         LOGE( "Error muxing packet\n");
188.         break;
189.     }
190.     av_free_packet(&pkt);
191.
192. }
193. //Write file trailer
194. av_write_trailer(ofmt_ctx);
195. end:
196. avformat_close_input(&ifmt_ctx);
197. /* close output */
198. if (ofmt_ctx && !(ofmt->flags & AVFMT_NOFILE))
199.     avio_close(ofmt_ctx->pb);
200. avformat_free_context(ofmt_ctx);
201. if (ret < 0 && ret != AVEERROR_EOF) {
202.     LOGE( "Error occurred.\n");
203.     return -1;
204. }
205. return 0;
206. }

```

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 := avcodec
13. LOCAL_SRC_FILES := libavcodec-56.so
14. include $(PREBUILT_SHARED_LIBRARY)
15.
16. include $(CLEAR_VARS)
17. LOCAL_MODULE := avdevice
18. LOCAL_SRC_FILES := libavdevice-56.so
19. include $(PREBUILT_SHARED_LIBRARY)
20.
21. include $(CLEAR_VARS)
22. LOCAL_MODULE := avfilter
23. LOCAL_SRC_FILES := libavfilter-5.so
24. include $(PREBUILT_SHARED_LIBRARY)
25.
26. include $(CLEAR_VARS)
27. LOCAL_MODULE := avformat
28. LOCAL_SRC_FILES := libavformat-56.so
29. include $(PREBUILT_SHARED_LIBRARY)
30.
31. include $(CLEAR_VARS)
32. LOCAL_MODULE := avutil
33. LOCAL_SRC_FILES := libavutil-54.so
34. include $(PREBUILT_SHARED_LIBRARY)
35.
36. include $(CLEAR_VARS)
37. LOCAL_MODULE := postproc
38. LOCAL_SRC_FILES := libpostproc-53.so
39. include $(PREBUILT_SHARED_LIBRARY)
40.
41. include $(CLEAR_VARS)
42. LOCAL_MODULE := swresample
43. LOCAL_SRC_FILES := libswresample-1.so
44. include $(PREBUILT_SHARED_LIBRARY)
45.
46. include $(CLEAR_VARS)
47. LOCAL_MODULE := swscale
48. LOCAL_SRC_FILES := libswscale-3.so
49. include $(PREBUILT_SHARED_LIBRARY)
50.
51. # Program
52. include $(CLEAR_VARS)
53. LOCAL_MODULE := sffstreamer
54. LOCAL_SRC_FILES := simplest_ffmpeg_streamer.c
55. LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
56. LOCAL_LDLIBS := -llog -lz
57. LOCAL_SHARED_LIBRARIES := avcodec avdevice avfilter avformat avutil postproc swresample swscale
58. include $(BUILD_SHARED_LIBRARY)

```

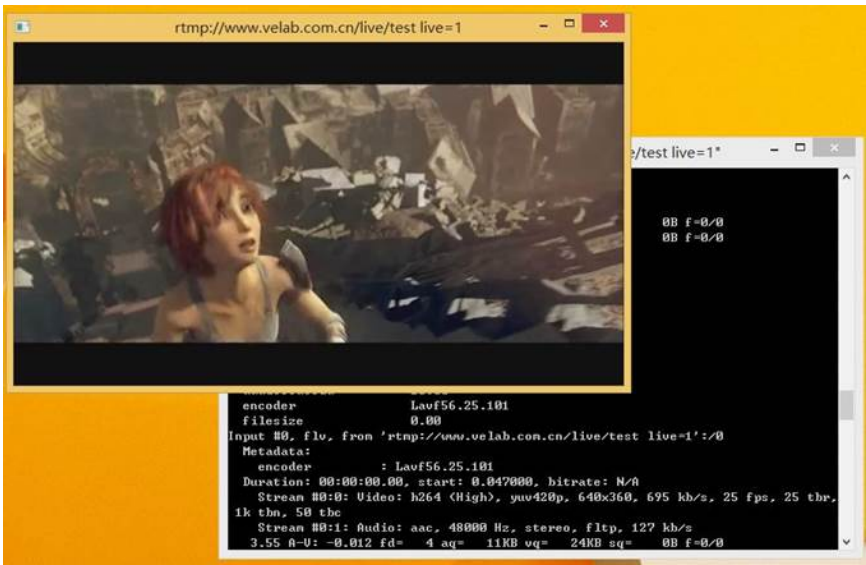
运行结果

App在手机上运行后的结果如下图所示。



注意需要把等待推送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。

推流后的视频可以在电脑端使用ffplay接收，如下图所示。



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/47056051>

文章标签：

FFmpeg

RTMP

Android

流媒体

JNI

个人分类：

Android多媒体

FFMPEG

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