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：38368

==========

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

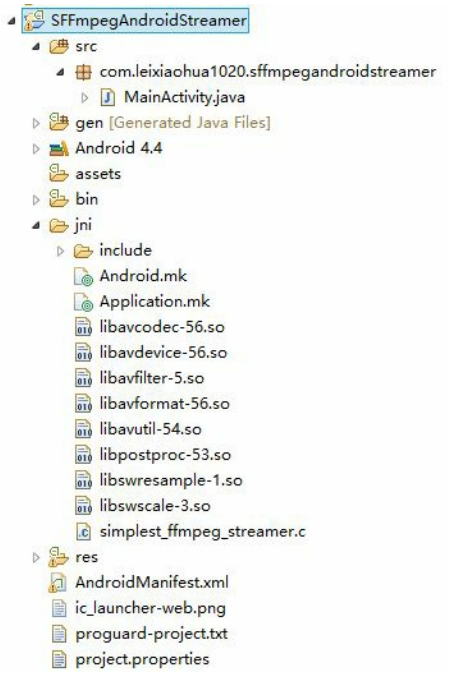
最简单的基于FFmpeg的移动端例子：Windows Phone HelloWorld

==========

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
/**
 * 最简单的基于FFmpeg的推流器(RTMP)-安卓
 * Simplest FFmpeg Android Streamer (RTMP)
 *
 * 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的推流器。
 * 它可以将视频文件以流媒体的形式推送到服务器。
 *
 * This software is the simplest streamer based on FFmpeg in Android.
 * It can stream local media file to streaming media server (in RTMP).
 *
 */
package com.leixiaohua1020.sffmpegandroidstreamer;

import android.os.Bundle;
import android.os.Environment;
import android.app.Activity;
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;

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 outputurl=urlEdittext_output.getText().toString();

                Log.e("inputurl",inputurl);
                Log.e("outputurl",outputurl);
                String info="";

                stream(inputurl,outputurl);

                Log.e("Info",info);
            }
        });
    }


    @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 stream(String inputurl, String outputurl);

    static{
        System.loadLibrary("avutil-54");
        System.loadLibrary("swresample-1");
        System.loadLibrary("avcodec-56");
        System.loadLibrary("avformat-56");
        System.loadLibrary("swscale-3");
        System.loadLibrary("postproc-53");
        System.loadLibrary("avfilter-5");
        System.loadLibrary("avdevice-56");
        System.loadLibrary("sffstreamer");
    }


}
```

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

```cpp
/**
 * 最简单的基于FFmpeg的推流器-安卓
 * Simplest FFmpeg Android Streamer
 *
 * 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的流媒体推送器。
 * 它可以将视频数据以流媒体的形式发送出去。
 *
 * This software is the simplest streamer based on FFmpeg in Android.
 * It can stream local media file to streaming media server (in RTMP).
 */


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

#include "libavcodec/avcodec.h"
#include "libavformat/avformat.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){

    //To TXT file
    FILE *fp=fopen("/storage/emulated/0/av_log.txt","a+");
    if(fp){
        vfprintf(fp,fmt,vl);
        fflush(fp);
        fclose(fp);
    }

    //To Logcat
    //LOGE(fmt, vl);
}

JNIEXPORT jint JNICALL Java_com_leixiaohua1020_sffmpegandroidstreamer_MainActivity_stream
  (JNIEnv *env, jobject obj, jstring input_jstr, jstring output_jstr)
{
  AVOutputFormat *ofmt = NULL;
    AVFormatContext *ifmt_ctx = NULL, *ofmt_ctx = NULL;
    AVPacket pkt;

    int ret, i;
    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));

    //input_str  = "cuc_ieschool.flv";
    //output_str = "rtmp://localhost/publishlive/livestream";
    //output_str = "rtp://233.233.233.233:6666";

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

    av_register_all();
    //Network
    avformat_network_init();

    //Input
    if ((ret = avformat_open_input(&ifmt_ctx, input_str, 0, 0)) < 0) {
        LOGE( "Could not open input file.");
        goto end;
    }
    if ((ret = avformat_find_stream_info(ifmt_ctx, 0)) < 0) {
        LOGE( "Failed to retrieve input stream information");
        goto end;
    }

    int videoindex=-1;
    for(i=0; i<ifmt_ctx->nb_streams; i++)
        if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
            videoindex=i;
```

```
            videoindex=i;
            break;
        }
    //Output
    avformat_alloc_output_context2(&ofmt_ctx, NULL, "flv",output_str); //RTMP
    //avformat_alloc_output_context2(&ofmt_ctx, NULL, "mpegts", output_str);//UDP

    if (!ofmt_ctx) {
        LOGE( "Could not create output context\n");
        ret = AVERROR_UNKNOWN;
        goto end;
    }
    ofmt = ofmt_ctx->oformat;
    for (i = 0; i < ifmt_ctx->nb_streams; i++) {
        //Create output AVStream according to input AVStream
        AVStream *in_stream = ifmt_ctx->streams[i];
        AVStream *out_stream = avformat_new_stream(ofmt_ctx, in_stream->codec->codec);
        if (!out_stream) {
            LOGE( "Failed allocating output stream\n");
            ret = AVERROR_UNKNOWN;
            goto end;
        }
        //Copy the settings of AVCodecContext
        ret = avcodec_copy_context(out_stream->codec, in_stream->codec);
        if (ret < 0) {
            LOGE( "Failed to copy context from input to output stream codec context\n");
            goto end;
        }
        out_stream->codec->codec_tag = 0;
        if (ofmt_ctx->oformat->flags & AVFMT_GLOBALHEADER)
            out_stream->codec->flags |= CODEC_FLAG_GLOBAL_HEADER;
    }

    //Open output URL
    if (!(ofmt->flags & AVFMT_NOFILE)) {
        ret = avio_open(&ofmt_ctx->pb, output_str, AVIO_FLAG_WRITE);
        if (ret < 0) {
            LOGE( "Could not open output URL '%s'", output_str);
            goto end;
        }
    }
    //Write file header
    ret = avformat_write_header(ofmt_ctx, NULL);
    if (ret < 0) {
        LOGE( "Error occurred when opening output URL\n");
        goto end;
    }

    int frame_index=0;

    int64_t start_time=av_gettime();
    while (1) {
        AVStream *in_stream, *out_stream;
        //Get an AVPacket
        ret = av_read_frame(ifmt_ctx, &pkt);
        if (ret < 0)
            break;
        //FIX：No PTS (Example: Raw H.264)
        //Simple Write PTS
        if(pkt.pts==AV_NOPTS_VALUE){
            //Write PTS
            AVRational time_base1=ifmt_ctx->streams[videoindex]->time_base;
            //Duration between 2 frames (us)
            int64_t calc_duration=(double)AV_TIME_BASE/av_q2d(ifmt_ctx->streams[videoindex]->r_frame_rate);
            //Parameters
            pkt.pts=(double)(frame_index*calc_duration)/(double)(av_q2d(time_base1)*AV_TIME_BASE);
            pkt.dts=pkt.pts;
            pkt.duration=(double)calc_duration/(double)(av_q2d(time_base1)*AV_TIME_BASE);
        }
        //Important:Delay
        if(pkt.stream_index==videoindex){
            AVRational time_base=ifmt_ctx->streams[videoindex]->time_base;
            AVRational time_base_q={1,AV_TIME_BASE};
            int64_t pts_time = av_rescale_q(pkt.dts, time_base, time_base_q);
            int64_t now_time = av_gettime() - start_time;
            if (pts_time > now_time)
                av_usleep(pts_time - now_time);

        }

        in_stream  = ifmt_ctx->streams[pkt.stream_index];
        out_stream = ofmt_ctx->streams[pkt.stream_index];
        /* copy packet */
        //Convert PTS/DTS
        pkt.pts = av_rescale_q_rnd(pkt.pts, in_stream->time_base, out_stream->time_base, AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX);
        pkt.dts = av_rescale_q_rnd(pkt.dts, in_stream->time_base, out_stream->time_base, AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX);
        pkt.duration = av_rescale_q(pkt.duration, in_stream->time_base, out_stream->time_base);
        pkt.pos = -1;
        //Print to Screen
        if(pkt.stream_index==videoindex){
            LOGE("Send %8d video frames to output URL\n",frame_index);
            frame_index++;
```

```
        }
        //ret = av_write_frame(ofmt_ctx, &pkt);
        ret = av_interleaved_write_frame(ofmt_ctx, &pkt);

        if (ret < 0) {
            LOGE( "Error muxing packet\n");
            break;
        }
        av_free_packet(&pkt);

    }
    //Write file trailer
    av_write_trailer(ofmt_ctx);
end:
    avformat_close_input(&ifmt_ctx);
    /* close output */
    if (ofmt_ctx && !(ofmt->flags & AVFMT_NOFILE))
        avio_close(ofmt_ctx->pb);
    avformat_free_context(ofmt_ctx);
    if (ret < 0 && ret != AVERROR_EOF) {
        LOGE( "Error occurred.\n");
        return -1;
    }
    return 0;
}
```

Android.mk文件位于jni/Android.mk，如下所示。

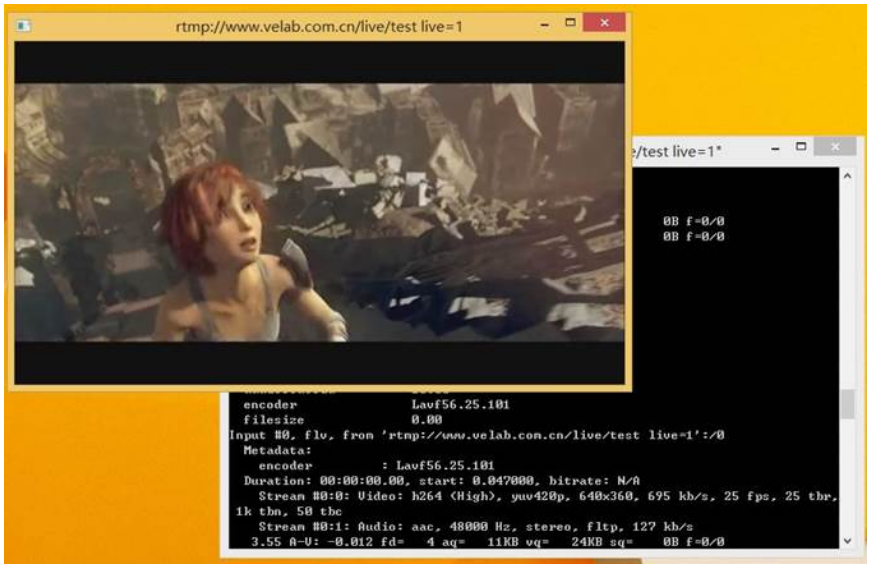
```
[plain]
# Android.mk for FFmpeg
#
# Lei Xiaohua 雷霄骅
# leixiaohua1020@126.com
# http://blog.csdn.net/leixiaohua1020
#

LOCAL_PATH := $(call my-dir)

# FFmpeg library
include $(CLEAR_VARS)
LOCAL_MODULE := avcodec
LOCAL_SRC_FILES := libavcodec-56.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := avdevice
LOCAL_SRC_FILES := libavdevice-56.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := avfilter
LOCAL_SRC_FILES := libavfilter-5.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := avformat
LOCAL_SRC_FILES := libavformat-56.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := avutil
LOCAL_SRC_FILES := libavutil-54.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := postproc
LOCAL_SRC_FILES := libpostproc-53.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := swresample
LOCAL_SRC_FILES := libswresample-1.so
include $(PREBUILT_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := swscale
LOCAL_SRC_FILES := libswscale-3.so
include $(PREBUILT_SHARED_LIBRARY)

# Program
include $(CLEAR_VARS)
LOCAL_MODULE := sffstreamer
LOCAL_SRC_FILES :=simplest_ffmpeg_streamer.c
LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
LOCAL_LDLIBS := -llog -lz
LOCAL_SHARED_LIBRARIES := avcodec avdevice avfilter avformat avutil postproc swresample swscale
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

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