原 最简单的基于FFmpeg的移动端例子：IOS 推流器

2015年07月29日 12:57:35　阅读数：29541

==========

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

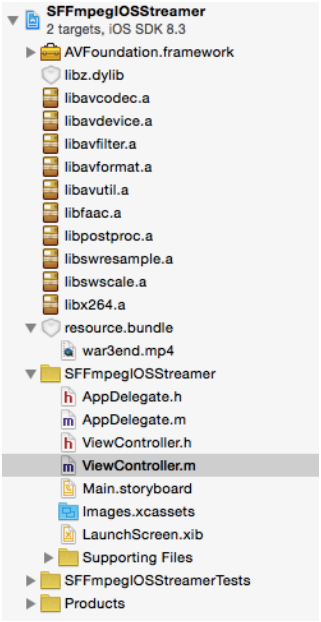
最简单的基于FFmpeg的移动端例子：Windows Phone HelloWorld

==========

本文记录IOS平台下基于FFmpeg的推流器。该示例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。



C代码位于ViewController.m文件中，内容如下所示。

```objc
/**
 * 最简单的基于FFmpeg的推流器-IOS
 * Simplest FFmpeg IOS 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IOS平台下的推流器。它可以将本地文件以流媒体的形式推送出去。
 *
 * This software is the simplest streamer in IOS.
 * It can stream local media files to streaming media server.
 */

#import "ViewController.h"
#include <libavformat/avformat.h>
#include <libavutil/mathematics.h>
#include <libavutil/time.h>

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)clickStreamButton:(id)sender {

    char input_str_full[500]={0};
    char output_str_full[500]={0};

    NSString *input_str= [NSString stringWithFormat:@"resource.bundle/%@",self.input.text];
    NSString *input_nsstr=[[[NSBundle mainBundle]resourcePath] stringByAppendingPathComponent:input_str];

    sprintf(input_str_full,"%s",[input_nsstr UTF8String]);
    sprintf(output_str_full,"%s",[self.output.text UTF8String]);

    printf("Input Path:%s\n",input_str_full);
    printf("Output Path:%s\n",output_str_full);

    AVOutputFormat *ofmt = NULL;
    //Input AVFormatContext and Output AVFormatContext
    AVFormatContext *ifmt_ctx = NULL, *ofmt_ctx = NULL;
    AVPacket pkt;
    char in_filename[500]={0};
    char out_filename[500]={0};
    int ret, i;
    int videoindex=-1;
```

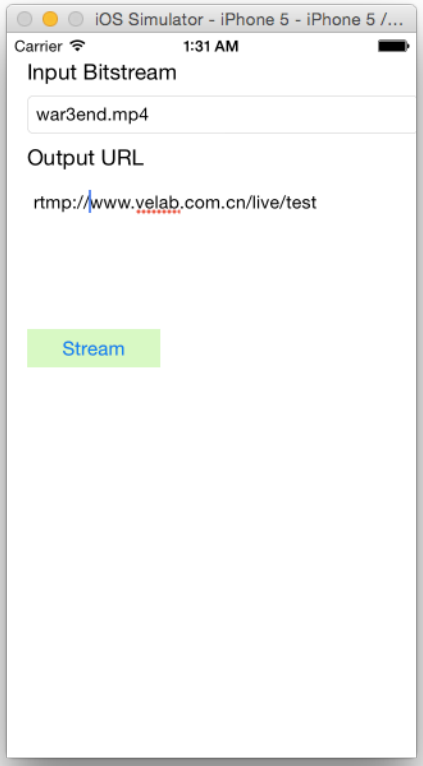
```
    int frame_index=0;
    int64_t start_time=0;
    //in_filename  = "cuc_ieschool.mov";
    //in_filename  = "cuc_ieschool.h264";
    //in_filename  = "cuc_ieschool.flv";//Input file URL
    //out_filename = "rtmp://localhost/publishlive/livestream";//Output URL[RTMP]
    //out_filename = "rtp://233.233.233.233:6666";//Output URL[UDP]

    strcpy(in_filename,input_str_full);
    strcpy(out_filename,output_str_full);

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

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

    av_dump_format(ifmt_ctx, 0, in_filename, 0);

    //Output

    avformat_alloc_output_context2(&ofmt_ctx, NULL, "flv", out_filename); //RTMP
    //avformat_alloc_output_context2(&ofmt_ctx, NULL, "mpegts", out_filename);//UDP

    if (!ofmt_ctx) {
        printf( "Could not create output context\n");
        ret = AVERROR_UNKNOWN;
        goto end;
    }
    ofmt = ofmt_ctx->oformat;
    for (i = 0; i < ifmt_ctx->nb_streams; i++) {

        AVStream *in_stream = ifmt_ctx->streams[i];
        AVStream *out_stream = avformat_new_stream(ofmt_ctx, in_stream->codec->codec);
        if (!out_stream) {
            printf( "Failed allocating output stream\n");
            ret = AVERROR_UNKNOWN;
            goto end;
        }

        ret = avcodec_copy_context(out_stream->codec, in_stream->codec);
        if (ret < 0) {
            printf( "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;
    }
    //Dump Format------------------
    av_dump_format(ofmt_ctx, 0, out_filename, 1);
    //Open output URL
    if (!(ofmt->flags & AVFMT_NOFILE)) {
        ret = avio_open(&ofmt_ctx->pb, out_filename, AVIO_FLAG_WRITE);
        if (ret < 0) {
            printf( "Could not open output URL '%s'", out_filename);
            goto end;
        }
    }

    ret = avformat_write_header(ofmt_ctx, NULL);
    if (ret < 0) {
        printf( "Error occurred when opening output URL\n");
        goto end;
    }

    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){
            printf("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) {
            printf( "Error muxing packet\n");
            break;
        }

        av_free_packet(&pkt);

    }
    //写文件尾（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) {
        printf( "Error occurred.\n");
        return;
    }
    return;

}
@end
```

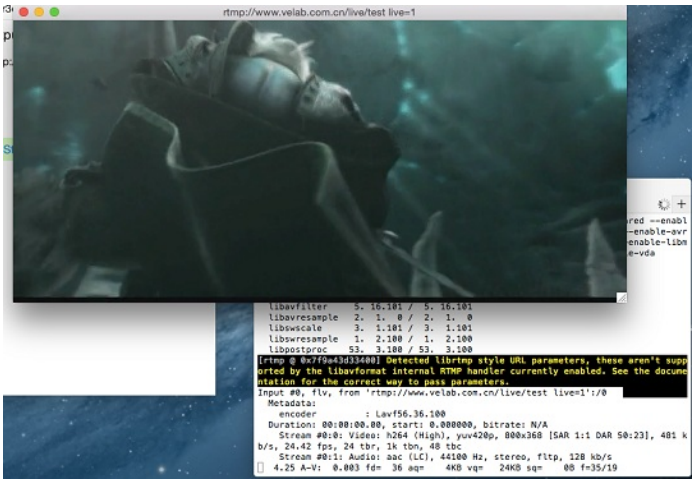
```
[objc]
1.
```

运行结果

App在手机上运行后的结果如下图所示。单击“Stream”，将会把位于resource.bundle中的“war3end.mp4”文件推送到“rtmp://www.velab.com.cn/live/test”的URL上。



使用视频播放器（在这里使用ffplay）可以查看推送的实时流，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/47072519

文章标签： FFmpeg IOS 流媒体 推流 RTMP
个人分类： FFMPEG IOS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