

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](#)

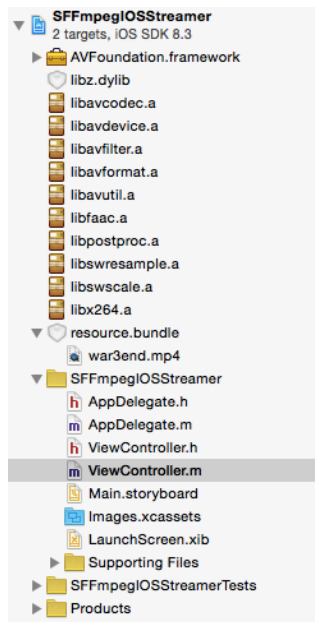
=====

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]  
1.  /**
2.   * 最简单的基于FFmpeg的推流器-IOS
3.   *  Simplest FFmpeg IOS 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。
12.  *
13.  *  This software is the simplest streamer in IOS.
14.  *  It can stream local media files to streaming media server.
15.  */
16.
17.  #import "ViewController.h"
18.  #include <libavformat/avformat.h>
19.  #include <libavutil/mathematics.h>
20.  #include <libavutil/time.h>
21.
22.  @interface ViewController ()
23.
24.  @end
25.
26.  @implementation ViewController
27.
28.  - (void)viewDidLoad {
29.      [super viewDidLoad];
30.      // Do any additional setup after loading the view, typically from a nib.
31.  }
32.
33.  - (void)didReceiveMemoryWarning {
34.      [super didReceiveMemoryWarning];
35.      // Dispose of any resources that can be recreated.
36.  }
37.
38.  - (IBAction)clickStreamButton:(id)sender {
39.
40.      char input_str_full[500]={0};
41.      char output_str_full[500]={0};
42.
43.      NSString *input_str= [NSString stringWithFormat:@"resource.bundle/%@",self.input.text];
44.      NSString *input_nsstr=[[NSBundle mainBundle]resourcePath] stringByAppendingPathComponent:input_str];
45.
46.      sprintf(input_str_full,"%s",[input_nsstr UTF8String]);
47.      sprintf(output_str_full,"%s",[self.output.text UTF8String]);
48.
49.      printf("Input Path:%s\n",input_str_full);
50.      printf("Output Path:%s\n",output_str_full);
51.
52.      AVOutputFormat *ofmt = NULL;
53.      //Input AVFormatContext and Output AVFormatContext
54.      AVFormatContext *ifmt_ctx = NULL, *ofmt_ctx = NULL;
55.      AVPacket pkt;
56.      char in_filename[500]={0};
57.      char out_filename[500]={0};
58.      int ret, i;
59.      int videoindex=-1;
```

```

60.     int frame_index=0;
61.     int64_t start_time=0;
62.     //in_filename = "cuc_ieschool.mov";
63.     //in_filename = "cuc_ieschool.h264";
64.     //in_filename = "cuc_ieschool.flv";//Input file URL
65.     //out_filename = "rtmp://localhost/publishlive/livestream";//Output URL[RTMP]
66.     //out_filename = "rtp://233.233.233.233:6666";//Output URL[UDP]
67.
68.     strcpy(in_filename,input_str_full);
69.     strcpy(out_filename,output_str_full);
70.
71.     av_register_all();
72.     //Network
73.     avformat_network_init();
74.     //Input
75.     if ((ret = avformat_open_input(&ifmt_ctx, in_filename, 0, 0)) < 0) {
76.         printf( "Could not open input file.");
77.         goto end;
78.     }
79.     if ((ret = avformat_find_stream_info(ifmt_ctx, 0)) < 0) {
80.         printf( "Failed to retrieve input stream information");
81.         goto end;
82.     }
83.
84.     for(i=0; i<ifmt_ctx->nb_streams; i++)
85.         if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
86.             videoindex=i;
87.             break;
88.         }
89.
90.     av_dump_format(ifmt_ctx, 0, in_filename, 0);
91.
92.     //Output
93.
94.     avformat_alloc_output_context2(&ofmt_ctx, NULL, "flv", out_filename); //RTMP
95.     //avformat_alloc_output_context2(&ofmt_ctx, NULL, "mpegts", out_filename);//UDP
96.
97.     if (!ofmt_ctx) {
98.         printf( "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.
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.            printf( "Failed allocating output stream\n");
109.            ret = AERROR_UNKNOWN;
110.            goto end;
111.        }
112.
113.        ret = avcodec_copy_context(out_stream->codec, in_stream->codec);
114.        if (ret < 0) {
115.            printf( "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.    //Dump Format-----
123.    av_dump_format(ofmt_ctx, 0, out_filename, 1);
124.    //Open output URL
125.    if (!(ofmt->flags & AVFMT_NOFILE)) {
126.        ret = avio_open(&ofmt_ctx->pb, out_filename, AVIO_FLAG_WRITE);
127.        if (ret < 0) {
128.            printf( "Could not open output URL '%s'", out_filename);
129.            goto end;
130.        }
131.    }
132.
133.    ret = avformat_write_header(ofmt_ctx, NULL);
134.    if (ret < 0) {
135.        printf( "Error occurred when opening output URL\n");
136.        goto end;
137.    }
138.
139.    start_time=av_gettime();
140.    while (1) {
141.        AVStream *in_stream, *out_stream;
142.        //Get an AVPacket
143.        ret = av_read_frame(ifmt_ctx, &pkt);
144.        if (ret < 0)
145.            break;
146.        //FIX:No PTS (Example: Raw H.264)
147.        //Simple Write PTS
148.        if(pkt.pts==AV_NOPTS_VALUE){
149.            //Write PTS
150.            AVRational time_base1=ifmt_ctx->streams[videoindex]->time_base;

```

```

151.         //Duration between 2 frames (us)
152.         int64_t calc_duration=(double)AV_TIME_BASE/av_q2d(ifmt_ctx->streams[videoindex]->r_frame_rate);
153.         //Parameters
154.         pkt.pts=(double)(frame_index*calc_duration)/(double)(av_q2d(time_base1)*AV_TIME_BASE);
155.         pkt.dts=pkt.pts;
156.         pkt.duration=(double)calc_duration/(double)(av_q2d(time_base1)*AV_TIME_BASE);
157.     }
158.     //Important:Delay
159.     if(pkt.stream_index==videoindex){
160.         AVRational time_base=ifmt_ctx->streams[videoindex]->time_base;
161.         AVRational time_base_q={1,AV_TIME_BASE};
162.         int64_t pts_time = av_rescale_q(pkt.dts, time_base, time_base_q);
163.         int64_t now_time = av_gettime() - start_time;
164.         if (pts_time > now_time)
165.             av_usleep(pts_time - now_time);
166.
167.     }
168.
169.     in_stream = ifmt_ctx->streams[pkt.stream_index];
170.     out_stream = ofmt_ctx->streams[pkt.stream_index];
171.     /* copy packet */
172.     //Convert PTS/DTS
173.     pkt.pts = av_rescale_q_rnd(pkt.pts, in_stream->time_base, out_stream-
>time_base, (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
174.     pkt.dts = av_rescale_q_rnd(pkt.dts, in_stream->time_base, out_stream-
>time_base, (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
175.     pkt.duration = av_rescale_q(pkt.duration, in_stream->time_base, out_stream->time_base);
176.     pkt.pos = -1;
177.     //Print to Screen
178.     if(pkt.stream_index==videoindex){
179.         printf("Send %8d video frames to output URL\n",frame_index);
180.         frame_index++;
181.     }
182.     //ret = av_write_frame(ofmt_ctx, &pkt);
183.     ret = av_interleaved_write_frame(ofmt_ctx, &pkt);
184.
185.     if (ret < 0) {
186.         printf( "Error muxing packet\n");
187.         break;
188.     }
189.
190.     av_free_packet(&pkt);
191.
192. }
193. //写文件尾 (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 != AVERROR_EOF) {
202.     printf( "Error occurred.\n");
203.     return;
204. }
205. return;
206.
207. }
208. @end

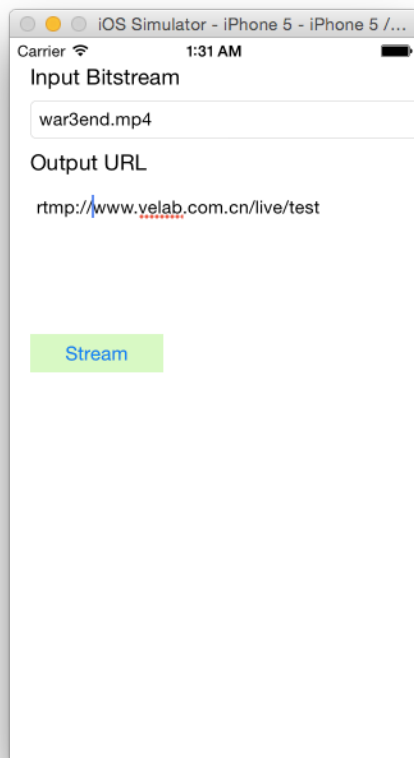
```

[objc]  

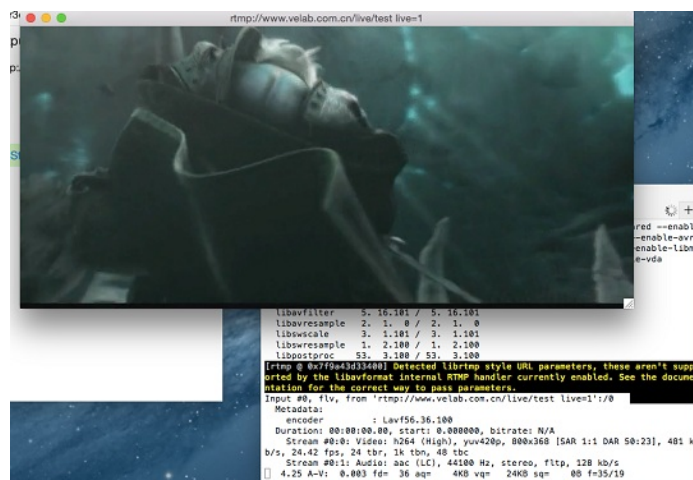
1.

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