

原 最简单的基于FFmpeg的封装格式处理：视音频分离器（demuxer）

2014年10月08日 00:58:56 阅读数：23377

最简单的基于FFmpeg的封装格式处理系列文章列表：

[最简单的基于FFmpeg的封装格式处理：视音频分离器简化版（demuxer-simple）](#)

[最简单的基于FFmpeg的封装格式处理：视音频分离器（demuxer）](#)

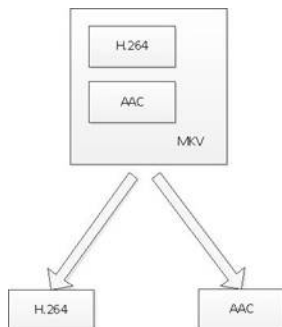
[最简单的基于FFmpeg的封装格式处理：视音频复用器（muxer）](#)

[最简单的基于FFMPEG的封装格式处理：封装格式转换（remuxer）](#)

简介

打算记录一下基于FFmpeg的封装格式处理方面的例子。包括了视音频分离，复用，封装格式转换。这是第2篇。

本文记录一个基于FFmpeg的视音频分离器(Simplest FFmpeg demuxer)。视音频分离器(Demuxer)即是将封装格式数据(例如MKV)中的视频压缩数据(例如H.264)和音频压缩数据(例如AAC)分离开。如图所示。在这个过程中并不涉及到编码和解码。

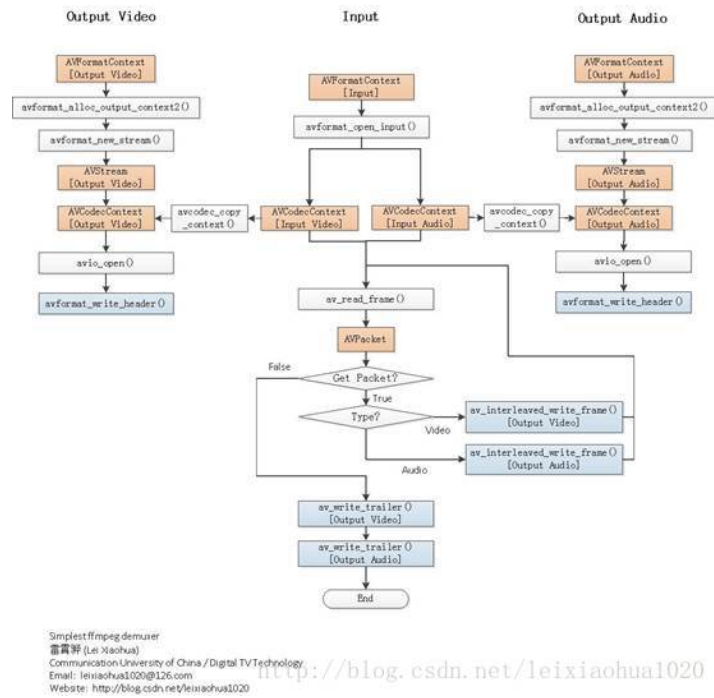


本文记录的程序可以将一个MPEG2TS封装的视频文件（其中视频编码为H.264，音频编码为AAC）分离成为两个文件：一个H.264编码的视频码流文件，一个AAC编码的音频码流文件。

前一篇文章中，记录一个简单版的视音频分离器。相比于前一篇文中的分离器，本篇文章记录的分离器复杂了很多。相比于简单版的分离器，学习的难度大了一些。但是该分离器可以很好地处理FFmpeg支持的各种格式（例如分离AAC音频流），拥有更好的实用性。

流程图

程序的流程如下图所示。从流程图中可以看出，一共初始化了3个AVFormatContext，其中1个用于输入，另外2个分别用于视频输出和音频输出。3个AVFormatContext初始化之后，通过avcodec_copy_context()函数可以将输入视频/音频的参数拷贝至输出视频/音频的AVCodecContext结构体。最后，通过av_read_frame()获取AVPacket，根据AVPacket类型的不同，分别使用av_interleaved_write_frame()写入不同的输出文件中即可。



[单击查看更清晰的图片](#)

PS：对于某些封装格式（例如MP4/FLV/MKV等）中的H.264，需要用到名称为"h264_mp4toannexb"的bitstream filter。这一点在前一篇文章《最简单的基于FFmpeg的封装格式处理：音视频分离器简化版（demuxer-simple）》中，已经有过详细叙述，这里不再重复。

简单介绍一下流程中各个重要函数的意义：

- avformat_open_input()：打开输入文件。
- avcodec_copy_context()：赋值AVCodecContext的参数。
- avformat_alloc_output_context2()：初始化输出文件。
- avio_open()：打开输出文件。
- avformat_write_header()：写入文件头。
- av_read_frame()：从输入文件读取一个AVPacket。
- av_interleaved_write_frame()：写入一个AVPacket到输出文件。
- av_write_trailer()：写入文件尾。

代码

下面贴上代码：

```
[cpp]
1.  /**
2.   * 最简单的基于FFmpeg的音视频分离器
3.   * Simplest FFMpeg Demuxer
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.  * 本程序可以将封装格式中的视频码流数据和音频码流数据分离出来。
12.  * 在该例子中，将MPEG2TS的文件分离得到H.264视频码流文件和AAC
13.  * 音频码流文件。
14.  *
15.  * This software split a media file (in Container such as
16.  * MKV, FLV, AVI...) to video and audio bitstream.
17.  * In this example, it demux a MPEG2TS file to H.264 bitstream
18.  * and AAC bitstream.
19.  */
20.
21. #include <stdio.h>
22.
23. #define __STDC_CONSTANT_MACROS
24.
25. #ifdef WIN32
26. //Windows
27. extern "C"
28. {
29. #include "libavformat/avformat.h"
30. };
31. #else
32. //Linux...
```

```

33. #ifdef __cplusplus
34. extern "C"
35. {
36. #endif
37. #include <libavformat/avformat.h>
38. #ifdef __cplusplus
39. };
40. #endif
41. #endif
42.
43. /*
44. FIX: H.264 in some container format (FLV, MP4, MKV etc.) need
45. "h264_mp4toannexb" bitstream filter (BSF)
46. *Add SPS,PPS in front of IDR frame
47. *Add start code ("0,0,0,1") in front of NALU
48. H.264 in some container (MPEG2TS) don't need this BSF.
49. */
50. //'1': Use H.264 Bitstream Filter
51. #define USE_H264BSF 0
52.
53. int main(int argc, char* argv[])
54. {
55.     AVOutputFormat *ofmt_a = NULL,*ofmt_v = NULL;
56.     // (Input AVFormatContext and Output AVFormatContext)
57.     AVFormatContext *ifmt_ctx = NULL, *ofmt_ctx_a = NULL, *ofmt_ctx_v = NULL;
58.     AVPacket pkt;
59.     int ret, i;
60.     int videoindex=-1, audioindex=-1;
61.     int frame_index=0;
62.
63.     const char *in_filename = "cuc_ieschool.ts";//Input file URL
64.     //char *in_filename = "cuc_ieschool.mkv";
65.     const char *out_filename_v = "cuc_ieschool.h264";//Output file URL
66.     //char *out_filename_a = "cuc_ieschool.mp3";
67.     const char *out_filename_a = "cuc_ieschool.aac";
68.
69.     av_register_all();
70.     //Input
71.     if ((ret = avformat_open_input(&ifmt_ctx, in_filename, 0, 0)) < 0) {
72.         printf( "Could not open input file.");
73.         goto end;
74.     }
75.     if ((ret = avformat_find_stream_info(ifmt_ctx, 0)) < 0) {
76.         printf( "Failed to retrieve input stream information");
77.         goto end;
78.     }
79.
80.     //Output
81.     avformat_alloc_output_context2(&ofmt_ctx_v, NULL, NULL, out_filename_v);
82.     if (!ofmt_ctx_v) {
83.         printf( "Could not create output context\n");
84.         ret = AERROR_UNKNOWN;
85.         goto end;
86.     }
87.     ofmt_v = ofmt_ctx_v->oformat;
88.
89.     avformat_alloc_output_context2(&ofmt_ctx_a, NULL, NULL, out_filename_a);
90.     if (!ofmt_ctx_a) {
91.         printf( "Could not create output context\n");
92.         ret = AERROR_UNKNOWN;
93.         goto end;
94.     }
95.     ofmt_a = ofmt_ctx_a->oformat;
96.
97.     for (i = 0; i < ifmt_ctx->nb_streams; i++) {
98.         //Create output AVStream according to input AVStream
99.         AVFormatContext *ofmt_ctx;
100.         AVStream *in_stream = ifmt_ctx->streams[i];
101.         AVStream *out_stream = NULL;
102.
103.         if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
104.             videoindex=i;
105.             out_stream=avformat_new_stream(ofmt_ctx_v, in_stream->codec->codec);
106.             ofmt_ctx=ofmt_ctx_v;
107.         }else if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_AUDIO){
108.             audioindex=i;
109.             out_stream=avformat_new_stream(ofmt_ctx_a, in_stream->codec->codec);
110.             ofmt_ctx=ofmt_ctx_a;
111.         }else{
112.             break;
113.         }
114.
115.         if (!out_stream) {
116.             printf( "Failed allocating output stream\n");
117.             ret = AERROR_UNKNOWN;
118.             goto end;
119.         }
120.         //Copy the settings of AVCodecContext
121.         if (avcodec_copy_context(out_stream->codec, in_stream->codec) < 0) {
122.             printf( "Failed to copy context from input to output stream codec context\n");
123.             goto end;

```

```

124.     }
125.     out_stream->codec->codec_tag = 0;
126.
127.     if (ofmt_ctx->oformat->flags & AVFMT_GLOBALHEADER)
128.         out_stream->codec->flags |= CODEC_FLAG_GLOBAL_HEADER;
129. }
130.
131. //Dump Format-----
132. printf("\n=====Input Video=====\\n");
133. av_dump_format(ifmt_ctx, 0, in_filename, 0);
134. printf("\n=====Output Video=====\\n");
135. av_dump_format(ofmt_ctx_v, 0, out_filename_v, 1);
136. printf("\n=====Output Audio=====\\n");
137. av_dump_format(ofmt_ctx_a, 0, out_filename_a, 1);
138. printf("\n=====\\n");
139. //Open output file
140. if (!(ofmt_v->flags & AVFMT_NOFILE)) {
141.     if (avio_open(&ofmt_ctx_v->pb, out_filename_v, AVIO_FLAG_WRITE) < 0) {
142.         printf( "Could not open output file '%s'", out_filename_v);
143.         goto end;
144.     }
145. }
146.
147. if (!(ofmt_a->flags & AVFMT_NOFILE)) {
148.     if (avio_open(&ofmt_ctx_a->pb, out_filename_a, AVIO_FLAG_WRITE) < 0) {
149.         printf( "Could not open output file '%s'", out_filename_a);
150.         goto end;
151.     }
152. }
153.
154. //Write file header
155. if (avformat_write_header(ofmt_ctx_v, NULL) < 0) {
156.     printf( "Error occurred when opening video output file\\n");
157.     goto end;
158. }
159. if (avformat_write_header(ofmt_ctx_a, NULL) < 0) {
160.     printf( "Error occurred when opening audio output file\\n");
161.     goto end;
162. }
163.
164. #if USE_H264BSF
165.     AVBitStreamFilterContext* h264bsfc = av_bitstream_filter_init("h264_mp4toannexb");
166. #endif
167.
168. while (1) {
169.     AVFormatContext *ofmt_ctx;
170.     AVStream *in_stream, *out_stream;
171.     //Get an AVPacket
172.     if (av_read_frame(ifmt_ctx, &pkt) < 0)
173.         break;
174.     in_stream = ifmt_ctx->streams[pkt.stream_index];
175.
176.
177.     if(pkt.stream_index==videoindex){
178.         out_stream = ofmt_ctx_v->streams[0];
179.         ofmt_ctx=ofmt_ctx_v;
180.         printf("Write Video Packet. size:%d\\tpts:%lld\\n",pkt.size,pkt.pts);
181.         #if USE_H264BSF
182.             av_bitstream_filter_filter(h264bsfc, in_stream->codec, NULL, &pkt.data, &pkt.size, pkt.data, pkt.size, 0);
183.         #endif
184.     }else if(pkt.stream_index==audioindex){
185.         out_stream = ofmt_ctx_a->streams[0];
186.         ofmt_ctx=ofmt_ctx_a;
187.         printf("Write Audio Packet. size:%d\\tpts:%lld\\n",pkt.size,pkt.pts);
188.     }else{
189.         continue;
190.     }
191.
192.
193.     //Convert PTS/DTS
194.     pkt.pts = av_rescale_q_rnd(pkt.pts, in_stream->time_base, out_stream->time_base, (AVRounding)
(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
195.     pkt.dts = av_rescale_q_rnd(pkt.dts, in_stream->time_base, out_stream->time_base, (AVRounding)
(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
196.     pkt.duration = av_rescale_q(pkt.duration, in_stream->time_base, out_stream->time_base);
197.     pkt.pos = -1;
198.     pkt.stream_index=0;
199.     //Write
200.     if (av_interleaved_write_frame(ofmt_ctx, &pkt) < 0) {
201.         printf( "Error muxing packet\\n");
202.         break;
203.     }
204.     //printf("Write %d frames to output file\\n",frame_index);
205.     av_free_packet(&pkt);
206.     frame_index++;
207. }
208.
209. #if USE_H264BSF
210.     av_bitstream_filter_close(h264bsfc);
211. #endif
212.
213. //Write file trailer

```

```

213.     //write file trailer
214.     av_write_trailer(ofmt_ctx_a);
215.     av_write_trailer(ofmt_ctx_v);
216. end:
217.     avformat_close_input(&ifmt_ctx);
218.     /* close output */
219.     if (ofmt_ctx_a && !(ofmt_a->flags & AVFMT_NOFILE))
220.         avio_close(ofmt_ctx_a->pb);
221.
222.     if (ofmt_ctx_v && !(ofmt_v->flags & AVFMT_NOFILE))
223.         avio_close(ofmt_ctx_v->pb);
224.
225.     avformat_free_context(ofmt_ctx_a);
226.     avformat_free_context(ofmt_ctx_v);
227.
228.
229.     if (ret < 0 && ret != AVEERROR_EOF) {
230.         printf( "Error occurred.\n");
231.         return -1;
232.     }
233.     return 0;
234. }

```

结果

输入文件为：

cuc_ieschool.ts：MPEG2TS封装格式数据。

输出文件为：

cuc_ieschool.h264：H.264视频码流数据。

cuc_ieschool.aac：AAC音频码流数据。

下载

simplest ffmpeg format

项目主页

SourceForge：<https://sourceforge.net/projects/simplestffmpegformat/>

Github：https://github.com/leixiaohua1020/simplest_ffmpeg_format

开源中国：http://git.oschina.net/leixiaohua1020/simplest_ffmpeg_format

CSDN下载地址：

<http://download.csdn.net/detail/leixiaohua1020/8005317>

工程中包含4个例子：

simplest_ffmpeg_demuxer_simple：视音频分离器（简化版）。

simplest_ffmpeg_demuxer：视音频分离器。

simplest_ffmpeg_muxer：视音频复用器。

simplest_ffmpeg_remuxer：封装格式转换器。

更新-1.1=====

修复了以下问题：

(1)Release版本下的运行问题

(2)simplest_ffmpeg_muxer分装H.264裸流的时候丢失声音的错误

CSDN下载地址：

<http://download.csdn.net/detail/leixiaohua1020/8284309>

更新-1.2 (2015.2.13)=====

这次考虑到了跨平台的要求，调整了源代码。经过这次调整之后，源代码可以在以下平台编译通过：

VC++：打开sln文件即可编译，无需配置。

cl.exe：打开compile_cl.bat即可命令行下使用cl.exe进行编译，注意可能需要按照VC的安装路径调整脚本里面的参数。编译命令如下。

```
[plain]
1.  ::VS2010 Environment
2.  call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
3.  ::include
4.  @set INCLUDE=include;%INCLUDE%
5.  ::lib
6.  @set LIB=lib;%LIB%
7.  ::compile and link
8.  cl simplest_ffmpeg_demuxer.cpp /link avcodec.lib avformat.lib avutil.lib ^
9.  avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib /OPT:NOREF
```

MinGW：MinGW命令行下运行compile_mingw.sh即可使用MinGW的g++进行编译。编译命令如下。

```
[plain]
1.  g++ simplest_ffmpeg_demuxer.cpp -g -o simplest_ffmpeg_demuxer.exe \
2.  -I /usr/local/include -L /usr/local/lib -lavformat -lavcodec -lavutil
```

GCC：Linux或者MacOS命令行下运行compile_gcc.sh即可使用GCC进行编译。编译命令如下。

```
[plain]
1.  gcc simplest_ffmpeg_demuxer.cpp -g -o simplest_ffmpeg_demuxer.out \
2.  -I /usr/local/include -L /usr/local/lib -lavformat -lavcodec -lavutil
```

PS：相关的编译命令已经保存到了工程文件夹中

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8445303>

SourceForge上已经更新。

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/39802819>

文章标签：[ffmpeg](#) [demux](#) [分离](#) [AAC](#) [封装格式](#)

个人分类：[我的开源项目](#) [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