

# 最简单的基于FFMPEG+SDL的音频播放器

2013年08月29日 17:11:09 阅读数：63813

最简单的基于FFmpeg的音频播放器系列文章列表：

《最简单的基于FFMPEG+SDL的音频播放器》

《最简单的基于FFMPEG+SDL的音频播放器 ver2 （采用SDL2.0）》

《最简单的基于FFMPEG+SDL的音频播放器：拆分·解码器和播放器》

## 简介

FFMPEG工程浩大，可以参考的书籍又不是很多，因此很多刚学习FFMPEG的人常常感觉到无从下手。

在此我把自己做项目过程中实现的一个非常简单的音频播放器（大约200行代码）源代码传上来，以作备忘，同时方便新手学习FFMPEG。

该播放器虽然简单，但是几乎包含了使用FFMPEG播放一个音频所有必备的API，并且使用SDL输出解码出来的音频。

并且支持流媒体等多种音频输入。程序使用了新的FFMPEG类库，和早期版本的FFMPEG类库的API函数略有不同。平台使用VC2010。

### SourceForge项目主页

<https://sourceforge.net/projects/simplestffmpegaudioplayer/>

注：本版本的SDL采用了SDL1.2，采用SDL2.0的播放器可以参考：

[最简单的基于FFMPEG+SDL的音频播放器 ver2 （采用SDL2.0）](#)

注意：

- 1.程序输出的解码后PCM音频数据可以使用Audition打开播放
- 2.m4a,aac文件可以直接播放。mp3文件需要调整SDL音频帧大小为4608（默认是4096），否则播放会不流畅
- 3.也可以播放视频中的音频

## 源代码

```
[cpp]
1.  /**
2.   * 最简单的基于FFmpeg的音频播放器  1.2
3.   *  Simplest FFmpeg Audio Player  1.2
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.  *  本程序实现了音频的解码和播放。
12.  *
13.  *  This software decode and play audio streams.
14.  */
15.
16. #include "stdafx.h"
17. #include <stdlib.h>
18. #include <string.h>
19. extern "C"
20. {
21.     #include "libavcodec/avcodec.h"
22.     #include "libavformat/avformat.h"
23.     #include "libswresample/swresample.h"
24.     //SDL
```

```

25. #include "SDL/SDL.h"
26. };
27.
28. #define MAX_AUDIO_FRAME_SIZE 192000 // 1 second of 48khz 32bit audio
29.
30.
31. //Output PCM
32. #define OUTPUT_PCM 1
33. //Use SDL
34. #define USE_SDL 1
35.
36. //Buffer:
37. //|-----|-----|
38. //chunk-----pos---len-----|
39. static Uint8 *audio_chunk;
40. static Uint32 audio_len;
41. static Uint8 *audio_pos;
42.
43. /* The audio function callback takes the following parameters:
44.  * stream: A pointer to the audio buffer to be filled
45.  * len: The length (in bytes) of the audio buffer
46.  * 回调函数
47.  */
48. void fill_audio(void *udata, Uint8 *stream, int len){
49.     if(audio_len==0) /* Only play if we have data left */
50.         return;
51.     len=(len>audio_len?audio_len:len); /* Mix as much data as possible */
52.
53.     SDL_MixAudio(stream,audio_pos,len,SDL_MIX_MAXVOLUME);
54.     audio_pos += len;
55.     audio_len -= len;
56. }
57. //-----
58.
59.
60. int main(int argc, char* argv[])
61. {
62.     AVFormatContext *pFormatCtx;
63.     int i, audioStream;
64.     AVCodecContext *pCodecCtx;
65.     AVCodec *pCodec;
66.
67.     char url[]="WavinFlag.aac";
68.     //char url[]="WavinFlag.mp3";
69.     //char url[]="72bian.wma";
70.
71.     av_register_all();
72.     avformat_network_init();
73.     pFormatCtx = avformat_alloc_context();
74.     //Open
75.     if(avformat_open_input(&pFormatCtx,url,NULL,NULL)!=0){
76.         printf("Couldn't open input stream.\n");
77.         return -1;
78.     }
79.     // Retrieve stream information
80.     if(av_find_stream_info(pFormatCtx)<0){
81.         printf("Couldn't find stream information.\n");
82.         return -1;
83.     }
84.     // Dump valid information onto standard error
85.     av_dump_format(pFormatCtx, 0, url, false);
86.
87.     // Find the first audio stream
88.     audioStream=-1;
89.     for(i=0; i < pFormatCtx->nb_streams; i++){
90.         if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_AUDIO){
91.             audioStream=i;
92.             break;
93.         }
94.     }
95.
96.     if(audioStream==-1){
97.         printf("Didn't find a audio stream.\n");
98.         return -1;
99.     }
100.
101.     // Get a pointer to the codec context for the audio stream
102.     pCodecCtx=pFormatCtx->streams[audioStream]->codec;
103.
104.     // Find the decoder for the audio stream
105.     pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
106.     if(pCodec==NULL){
107.         printf("Codec not found.\n");
108.         return -1;
109.     }
110.
111.     // Open codec
112.     if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
113.         printf("Could not open codec.\n");
114.         return -1;
115.     }

```

```

116. FILE *pFile=NULL;
117. #if OUTPUT_PCM
118. pFile=fopen("output.pcm", "wb");
119. #endif
120.
121. AVPacket *packet=(AVPacket *)malloc(sizeof(AVPacket));
122. av_init_packet(packet);
123.
124. //Out Audio Param
125. uint64_t out_channel_layout=AV_CH_LAYOUT_STEREO;
126. //AAC:1024 MP3:1152
127. int out_nb_samples=pCodecCtx->frame_size;
128. AVSampleFormat out_sample_fmt=AV_SAMPLE_FMT_S16;
129. int out_sample_rate=44100;
130. int out_channels=av_get_channel_layout_nb_channels(out_channel_layout);
131. //Out Buffer Size
132. int out_buffer_size=av_samples_get_buffer_size(NULL,out_channels ,out_nb_samples,out_sample_fmt, 1);
133.
134. uint8_t *out_buffer=(uint8_t *)av_malloc(MAX_AUDIO_FRAME_SIZE*2);
135.
136. AVFrame *pFrame;
137. pFrame=avcodec_alloc_frame();
138. //SDL-----
139. #if USE_SDL
140. //Init
141. if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
142.     printf( "Could not initialize SDL - %s\n", SDL_GetError());
143.     return -1;
144. }
145. //SDL_AudioSpec
146. SDL_AudioSpec wanted_spec;
147. wanted_spec.freq = out_sample_rate;
148. wanted_spec.format = AUDIO_S16SYS;
149. wanted_spec.channels = out_channels;
150. wanted_spec.silence = 0;
151. wanted_spec.samples = out_nb_samples;
152. wanted_spec.callback = fill_audio;
153. wanted_spec.userdata = pCodecCtx;
154.
155. if (SDL_OpenAudio(&wanted_spec, NULL)<0){
156.     printf("can't open audio.\n");
157.     return -1;
158. }
159. #endif
160. printf("Bitrate:\t %3d\n", pFormatCtx->bit_rate);
161. printf("Decoder Name:\t %s\n", pCodecCtx->codec->long_name);
162. printf("Channels:\t %d\n", pCodecCtx->channels);
163. printf("Sample per Second\t %d \n", pCodecCtx->sample_rate);
164.
165. uint32_t ret,len = 0;
166. int got_picture;
167. int index = 0;
168. //FIX:Some Codec's Context Information is missing
169. int64_t in_channel_layout=av_get_default_channel_layout(pCodecCtx->channels);
170. //Swr
171. struct SwrContext *au_convert_ctx;
172. au_convert_ctx = swr_alloc();
173. au_convert_ctx=swr_alloc_set_opts(au_convert_ctx,out_channel_layout, out_sample_fmt, out_sample_rate,
174.     in_channel_layout,pCodecCtx->sample_fmt , pCodecCtx->sample_rate,0, NULL);
175. swr_init(au_convert_ctx);
176.
177. //Play
178. SDL_PauseAudio(0);
179.
180. while(av_read_frame(pFormatCtx, packet)>=0){
181.     if(packet->stream_index==audioStream){
182.
183.         ret = avcodec_decode_audio4( pCodecCtx, pFrame,&got_picture, packet);
184.         if ( ret < 0 ) {
185.             printf("Error in decoding audio frame.\n");
186.             return -1;
187.         }
188.         if ( got_picture > 0 ){
189.             swr_convert(au_convert_ctx,&out_buffer, MAX_AUDIO_FRAME_SIZE,(const uint8_t **)pFrame->data , pFrame->nb_samples);
190.
191.             printf("index:%5d\t pts:%lld\t packet size:%d\n",index,packet->pts,packet->size);
192.
193. #if OUTPUT_PCM
194.             //Write PCM
195.             fwrite(out_buffer, 1, out_buffer_size, pFile);
196. #endif
197.
198.             index++;
199.         }
200. //SDL-----
201. #if USE_SDL
202.         //Set audio buffer (PCM data)
203.         audio_chunk = (uint8_t *) out_buffer;
204.         //Audio buffer length
205.         audio_len =out_buffer_size;
206.
207.         //Fill audio buffer

```

```

207.         audio_pos = audio_chunk;
208.
209.         while(audio_len>0)//Wait until finish
210.             SDL_Delay(1);
211.     #endif
212.     }
213.     av_free_packet(packet);
214. }
215.
216.     swr_free(&au_convert_ctx);
217.
218. #if USE_SDL
219.     SDL_CloseAudio();//Close SDL
220.     SDL_Quit();
221. #endif
222.
223. #if OUTPUT_PCM
224.     fclose(pFile);
225. #endif
226.     av_free(out_buffer);
227.     avcodec_close(pCodecCtx);
228.     av_close_input_file(pFormatCtx);
229.
230.     return 0;
231. }

```

## 结果

程序会打印每一帧的信息，同时将音频输出到音频输出设备。运行截图如下所示。

```

index 220
pts 144179200
dts 144179200
packet_size 369
index 221
pts 144834560
dts 144834560
packet_size 344
index 222
pts 145489920
dts 145489920
packet_size 350
index 223
pts 146145280
dts 146145280
packet_size 359
index 224
pts 146800640
dts 146800640
packet_size 349
index 225
pts 147456000
dts 147456000
packet_size 377

```

http://blog.csdn.net/leixiaohua1020/6033893

完整工程下载地址：

<http://download.csdn.net/detail/leixiaohua1020/6033893>

## 更新列表

**更新（2014.5.8）** =====

simplest ffmpeg audio player

完整工程（更新版）下载地址：

<http://download.csdn.net/detail/leixiaohua1020/7319225>

新版本中使用了最新版本的FFMPEG类库（2014.5.7）。FFMPEG在新版本中的音频解码方面发生了比较大的变化。如果将旧版的主程序和新版的类库组合使用的话，会出现听到的都是杂音这一现象。经过研究发现，新版中avcodec\_decode\_audio4()解码后输出的音频采样数据格式为AV\_SA

MPLE\_FMT\_FLTP (float, planar) 而不再是AV\_SAMPLE\_FMT\_S16 (signed 16 bits)。因此无法直接使用SDL进行播放。

最后的解决方法是使用SwrContext对音频采样数据进行转换之后，再进行输出播放，问题就可以得到解决了。转换方面的代码如下示例：

```
[cpp]
1. //输出音频数据大小，一定小于输出内存。
2. int out_linesize;
3. //输出内存大小
4. int out_buffer_size=av_samples_get_buffer_size(&out_linesize, pCodecCtx->channels,pCodecCtx->frame_size,pCodecCtx->sample_fmt, 1);
5. uint8_t *out_buffer=new uint8_t[out_buffer_size];
6. ...
7. au_convert_ctx = swr_alloc();
8. au_convert_ctx=swr_alloc_set_opts(au_convert_ctx,AV_CH_LAYOUT_STEREO, AV_SAMPLE_FMT_S16, 44100,
9. pCodecCtx->channel_layout,pCodecCtx->sample_fmt , pCodecCtx->sample_rate,0, NULL);
10. swr_init(au_convert_ctx);
11.
12. while(av_read_frame(pFormatCtx, packet)>=0){
13.     .....
14.     swr_convert(au_convert_ctx,&out_buffer, out_linesize,(const uint8_t **)pFrame->data , pFrame->nb_samples);
15.
16.     .....
17. }
```

更新 (2014.9.1) =====

simplest ffmpeg audio player classic

完整工程 (classic) 下载地址：

<http://download.csdn.net/detail/leixiaohua1020/7849625>

本版本使用的类库编译时间为2012年的，无需经过swr\_convert()即可播放，代码简洁。

重建了工程，删掉了不必要的代码，把代码修改得更规范更易懂。

可以通过宏控制是否使用SDL，以及是否输出PCM。

```
[cpp]
1. //Output PCM
2. #define OUTPUT_PCM 0
3. //Use SDL
4. #define USE_SDL 1
```

更新 (2014.9.2) =====

simplest ffmpeg audio player 1.2

完整工程下载地址：

<http://download.csdn.net/detail/leixiaohua1020/7853199>

本版本使用新的类库（2014.5.6），解码后的音频需要经过swr\_convert()转换后方可播放。

重建了工程，删掉了不必要的代码，把代码修改得更规范更易懂。

可以通过宏控制是否使用SDL，以及是否输出PCM。

此外修改了部分地方，在原先版本的基础上，支持更多种的音频格式：AAC，MP3...

这一版本后不再修正这个音频播放器，以后改为修正基于SDL2.0的音频播放器

=====

**FFMPEG相关资料：**

SDL GUIDE 中文译本

<http://download.csdn.net/detail/leixiaohua1020/6389841>

ffdoc （FFMPEG的最完整教程）

<http://download.csdn.net/detail/leixiaohua1020/6377803>

如何用FFmpeg编写一个简单播放器

<http://download.csdn.net/detail/leixiaohua1020/6373783>

文章标签：

FFMPEG

播放器

解码

SDL

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