

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

2014年09月01日 14:26:55 阅读数：28043

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

之前做过一个简单的音频播放器：《 [最简单的基于FFMPEG+SDL的音频播放器](#) 》，采用的是SDL1.2。前两天刚把原先做的《最简单的基于FFMPEG+SDL的视频播放器》更新采用了SDL2.0，于是顺手也把音频播放器更新成为SDL2.0。

需要注意的是，与播放视频有很大的不同，SDL2.0播放音频的函数相对于SDL1.2来说变化很小。基本上保持了不变。

除了使用SDL2.0之外，修改了如下地方：

*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。

*支持MP3,AAC等多种格式

源代码

```
[cpp]
1.  /**
2.   * 最简单的基于FFmpeg的音频播放器 2
3.   *  Simplest Ffmpeg Audio Player 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.  *  是最简单的FFmpeg音频解码方面的教程。
13.  *  通过学习本例子可以了解FFmpeg的解码流程。
14.  *
15.  *  该版本使用SDL 2.0替换了第一个版本中的SDL 1.0。
16.  *  注意：SDL 2.0中音频解码的API并无变化。唯一变化的地方在于
17.  *  其回调函数的中的Audio Buffer并没有完全初始化，需要手动初始化。
18.  *  本例子中即SDL_memset(stream, 0, len);
19.  *
20.  *  This software decode and play audio streams.
21.  *  Suitable for beginner of Ffmpeg.
22.  *
23.  *  This version use SDL 2.0 instead of SDL 1.2 in version 1
24.  *  Note:The good news for audio is that, with one exception,
25.  *  it's entirely backwards compatible with 1.2.
26.  *  That one really important exception: The audio callback
27.  *  does NOT start with a fully initialized buffer anymore.
28.  *  You must fully write to the buffer in all cases. In this
29.  *  example it is SDL_memset(stream, 0, len);
30.  *
31.  *  Version 2.0
32.  */
33. #include <stdio.h>
34. #include <stdlib.h>
35. #include <string.h>
36.
37. #define __STDC_CONSTANT_MACROS
38.
39. #ifdef _WIN32
40. //Windows
41. extern "C"
42. {
43. #include "libavcodec/avcodec.h"
```

```

44. #include "libavformat/avformat.h"
45. #include "libswresample/swresample.h"
46. #include "SDL2/SDL.h"
47. };
48. #else
49. //Linux...
50. #ifdef __cplusplus
51. extern "C"
52. {
53. #endif
54. #include <libavcodec/avcodec.h>
55. #include <libavformat/avformat.h>
56. #include <libswresample/swresample.h>
57. #include <SDL2/SDL.h>
58. #ifdef __cplusplus
59. };
60. #endif
61. #endif
62.
63. #define MAX_AUDIO_FRAME_SIZE 192000 // 1 second of 48khz 32bit audio
64.
65. //Output PCM
66. #define OUTPUT_PCM 1
67. //Use SDL
68. #define USE_SDL 1
69.
70. //Buffer:
71. //|-----|-----|
72. //chunk-----pos---len-----|
73. static Uint8 *audio_chunk;
74. static Uint32 audio_len;
75. static Uint8 *audio_pos;
76.
77. /* The audio function callback takes the following parameters:
78.  * stream: A pointer to the audio buffer to be filled
79.  * len: The length (in bytes) of the audio buffer
80.  */
81. void fill_audio(void *udata, Uint8 *stream, int len){
82.     //SDL 2.0
83.     SDL_memset(stream, 0, len);
84.     if(audio_len==0)
85.         return;
86.
87.     len=(len>audio_len?audio_len:len); /* Mix as much data as possible */
88.
89.     SDL_MixAudio(stream, audio_pos, len, SDL_MIX_MAXVOLUME);
90.     audio_pos += len;
91.     audio_len -= len;
92. }
93. //-----
94.
95.
96. int main(int argc, char* argv[])
97. {
98.     AVFormatContext *pFormatCtx;
99.     int i, audioStream;
100.     AVCodecContext *pCodecCtx;
101.     AVCodec *pCodec;
102.     AVPacket *packet;
103.     uint8_t *out_buffer;
104.     AVFrame *pFrame;
105.     SDL_AudioSpec wanted_spec;
106.     int ret;
107.     uint32_t len = 0;
108.     int got_picture;
109.     int index = 0;
110.     int64_t in_channel_layout;
111.     struct SwrContext *au_convert_ctx;
112.
113.     FILE *pFile=NULL;
114.     char url[]="xiaoqingge.mp3";
115.
116.     av_register_all();
117.     avformat_network_init();
118.     pFormatCtx = avformat_alloc_context();
119.     //Open
120.     if(avformat_open_input(&pFormatCtx, url, NULL, NULL)!=0){
121.         printf("Couldn't open input stream.\n");
122.         return -1;
123.     }
124.     // Retrieve stream information
125.     if(avformat_find_stream_info(pFormatCtx, NULL)<0){
126.         printf("Couldn't find stream information.\n");
127.         return -1;
128.     }
129.     // Dump valid information onto standard error
130.     av_dump_format(pFormatCtx, 0, url, false);
131.
132.     // Find the first audio stream
133.     audioStream=-1;
134.     for(i=0; i < pFormatCtx->nb_streams; i++)

```

```

135.         if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_AUDIO){
136.             audioStream=i;
137.             break;
138.         }
139.
140.         if(audioStream==-1){
141.             printf("Didn't find a audio stream.\n");
142.             return -1;
143.         }
144.
145.         // Get a pointer to the codec context for the audio stream
146.         pCodecCtx=pFormatCtx->streams[audioStream]->codec;
147.
148.         // Find the decoder for the audio stream
149.         pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
150.         if(pCodec==NULL){
151.             printf("Codec not found.\n");
152.             return -1;
153.         }
154.
155.         // Open codec
156.         if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
157.             printf("Could not open codec.\n");
158.             return -1;
159.         }
160.
161.
162.         #if OUTPUT_PCM
163.         pFile=fopen("output.pcm", "wb");
164.         #endif
165.
166.         packet=(AVPacket *)av_malloc(sizeof(AVPacket));
167.         av_init_packet(packet);
168.
169.         //Out Audio Param
170.         uint64_t out_channel_layout=AV_CH_LAYOUT_STEREO;
171.         //nb_samples: AAC-1024 MP3-1152
172.         int out_nb_samples=pCodecCtx->frame_size;
173.         AVSampleFormat out_sample_fmt=AV_SAMPLE_FMT_S16;
174.         int out_sample_rate=44100;
175.         int out_channels=av_get_channel_layout_nb_channels(out_channel_layout);
176.         //Out Buffer Size
177.         int out_buffer_size=av_samples_get_buffer_size(NULL,out_channels ,out_nb_samples,out_sample_fmt, 1);
178.
179.         out_buffer=(uint8_t *)av_malloc(MAX_AUDIO_FRAME_SIZE*2);
180.         pFrame=av_frame_alloc();
181.         //SDL-----
182.         #if USE_SDL
183.         //Init
184.         if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
185.             printf( "Could not initialize SDL - %s\n", SDL_GetError());
186.             return -1;
187.         }
188.         //SDL_AudioSpec
189.         wanted_spec.freq = out_sample_rate;
190.         wanted_spec.format = AUDIO_S16SYS;
191.         wanted_spec.channels = out_channels;
192.         wanted_spec.silence = 0;
193.         wanted_spec.samples = out_nb_samples;
194.         wanted_spec.callback = fill_audio;
195.         wanted_spec.userdata = pCodecCtx;
196.
197.         if (SDL_OpenAudio(&wanted_spec, NULL)<0){
198.             printf("can't open audio.\n");
199.             return -1;
200.         }
201.         #endif
202.
203.         //FIX:Some Codec's Context Information is missing
204.         in_channel_layout=av_get_default_channel_layout(pCodecCtx->channels);
205.         //Swr
206.
207.         au_convert_ctx = swr_alloc();
208.         au_convert_ctx=swr_alloc_set_opts(au_convert_ctx,out_channel_layout, out_sample_fmt, out_sample_rate,
209.             in_channel_layout,pCodecCtx->sample_fmt , pCodecCtx->sample_rate,0, NULL);
210.         swr_init(au_convert_ctx);
211.
212.         //Play
213.         SDL_PauseAudio(0);
214.
215.         while(av_read_frame(pFormatCtx, packet)>=0){
216.             if(packet->stream_index==audioStream){
217.                 ret = avcodec_decode_audio4( pCodecCtx, pFrame,&got_picture, packet);
218.                 if ( ret < 0 ) {
219.                     printf("Error in decoding audio frame.\n");
220.                     return -1;
221.                 }
222.                 if ( got_picture > 0 ){
223.                     swr_convert(au_convert_ctx,&out_buffer, MAX_AUDIO_FRAME_SIZE,(const uint8_t **)pFrame->data , pFrame->nb_samples);
224.                     #if 1
225.                     printf("index:%5d\t pts:%lld\t packet size:%d\n",index,packet->pts,packet->size);
226.                     #endif

```

```

227. #endif
228.
229. #if OUTPUT_PCM
230.     //Write PCM
231.     fwrite(out_buffer, 1, out_buffer_size, pFile);
232. #endif
233.     index++;
234. }
235.
236. #if USE_SDL
237.     while(audio_len>0)//Wait until finish
238.         SDL_Delay(1);
239.
240.     //Set audio buffer (PCM data)
241.     audio_chunk = (Uint8 *) out_buffer;
242.     //Audio buffer length
243.     audio_len =out_buffer_size;
244.     audio_pos = audio_chunk;
245.
246. #endif
247.     }
248.     av_free_packet(packet);
249. }
250.
251.     swr_free(&au_convert_ctx);
252.
253. #if USE_SDL
254.     SDL_CloseAudio();//Close SDL
255.     SDL_Quit();
256. #endif
257.
258. #if OUTPUT_PCM
259.     fclose(pFile);
260. #endif
261.     av_free(out_buffer);
262.     avcodec_close(pCodecCtx);
263.     avformat_close_input(&pFormatCtx);
264.
265.     return 0;
266. }

```

下载

Simplest FFmpeg audio player 2

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

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

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

修正版CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/7853285>

*注：修正版中又修正了以下问题：

- 1.PCM输出的fwrite()的size有错误
- 2.PCM输出的fclose()外面添加了宏定义
- 3.部分编码器（例如WMA）的AVCodecContext中的channel_layout没有进行初始化。会导致SwrContext初始化失败。改为通过channels（一定会初始化）计算channel_layout而不是直接取channel_layout的值。

更新-2.1 (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_audio_player.cpp /MD /link SDL.lib SDLmain.lib avcodec.lib ^
9.  avformat.lib avutil.lib avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib ^
10. /SUBSYSTEM:WINDOWS /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_audio_player.cpp -g -o simplest_ffmpeg_audio_player.exe \
2.  -I /usr/local/include -L /usr/local/lib \
3.  -lmingw32 -lSDL2main -lSDL2 -lavformat -lavcodec -lavutil -lswresample
```

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_audio_player.cpp -g -o simplest_ffmpeg_audio_player.out -I /usr/local/include -L /usr/local/lib \
2.  -lSDL2main -lSDL2 -lavformat -lavcodec -lavutil -lswresample
```

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/leixiaohua1020/8444761>

SourceForge、Github等上面已经更新。

更新-2.2 (2015.7.17)=

增加了下面工程：

simplest_ffmpeg_audio_decoder：音频解码器。使用了libavcodec和libavformat。

simplest_audio_play_sdl2：使用SDL2播放PCM采样数据的例子。

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8924329>

SourceForge、Github等上面已经更新。

文章标签：[ffmpeg](#) [音频](#) [sdl2](#) [播放](#) [解码](#)

个人分类：[FFMPEG](#) [我的开源项目](#)

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