原 **最简单的基于FFMPEG+SDL的音频播放器：拆分-解码器和播放器**

==

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》

==

　　本文补充记录《最简单的基于FFMPEG+SDL的音频播放器》中的两个例子：FFmpeg音频解码器和SDL音频采样数据播放器。这两个部分是从音频播放器中拆分出来的两个例子。FFmpeg音频解码器实现了视频数据到PCM采样数据的解码，而SDL音频采样数据播放器实现了PCM数据到音频设备的播放。简而言之，原先的FFmpeg+SDL音频播放器实现了：

<div align="center">**音频数据->PCM->音频设备**</div>

FFmpeg音频解码器实现了：

<div align="center">**音频数据->PCM**</div>

SDL音频采样数据播放器实现了：

<div align="center">**PCM->音频设备**</div>

FFmpeg音频解码器

源代码

```cpp
/**
 * 最简单的基于FFmpeg的音频解码器
 * Simplest FFmpeg Audio Decoder
 *
 * 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
 *
 * 本程序可以将音频码流（mp3，AAC等）解码为PCM采样数据。
 * 是最简单的FFmpeg音频解码方面的教程。
 * 通过学习本例子可以了解FFmpeg的解码流程。
 *
 * This software decode audio streams (AAC,MP3 ...) to PCM data.
 * Suitable for beginner of FFmpeg.
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define __STDC_CONSTANT_MACROS

#ifdef _WIN32
//Windows
extern "C"
{
#include "libavcodec/avcodec.h"
#include "libavformat/avformat.h"
#include "libswresample/swresample.h"
};
#else
//Linux...
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavcodec/avcodec.h>
```

```c
40.    #include <libavformat/avformat.h>
41.    #include <libswresample/swresample.h>
42.    #ifdef __cplusplus
43.    };
44.    #endif
45.    #endif
46.
47.    #define MAX_AUDIO_FRAME_SIZE 192000 // 1 second of 48khz 32bit audio
48.
49.    int main(int argc, char* argv[])
50.    {
51.        AVFormatContext *pFormatCtx;
52.        int             i, audioStream;
53.        AVCodecContext  *pCodecCtx;
54.        AVCodec         *pCodec;
55.        AVPacket        *packet;
56.        uint8_t         *out_buffer;
57.        AVFrame         *pFrame;
58.        int ret;
59.        uint32_t len = 0;
60.        int got_picture;
61.        int index = 0;
62.        int64_t in_channel_layout;
63.        struct SwrContext *au_convert_ctx;
64.
65.        FILE *pFile=fopen("output.pcm", "wb");
66.        char url[]="skycity1.mp3";
67.
68.        av_register_all();
69.        avformat_network_init();
70.        pFormatCtx = avformat_alloc_context();
71.        //Open
72.        if(avformat_open_input(&pFormatCtx,url,NULL,NULL)!=0){
73.            printf("Couldn't open input stream.\n");
74.            return -1;
75.        }
76.        // Retrieve stream information
77.        if(avformat_find_stream_info(pFormatCtx,NULL)<0){
78.            printf("Couldn't find stream information.\n");
79.            return -1;
80.        }
81.        // Dump valid information onto standard error
82.        av_dump_format(pFormatCtx, 0, url, false);
83.
84.        // Find the first audio stream
85.        audioStream=-1;
86.        for(i=0; i < pFormatCtx->nb_streams; i++)
87.            if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_AUDIO){
88.                audioStream=i;
89.                break;
90.            }
91.
92.        if(audioStream==-1){
93.            printf("Didn't find a audio stream.\n");
94.            return -1;
95.        }
96.
97.        // Get a pointer to the codec context for the audio stream
98.        pCodecCtx=pFormatCtx->streams[audioStream]->codec;
99.
100.       // Find the decoder for the audio stream
101.       pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
102.       if(pCodec==NULL){
103.           printf("Codec not found.\n");
104.           return -1;
105.       }
106.
107.       // Open codec
108.       if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
109.           printf("Could not open codec.\n");
110.           return -1;
111.       }
112.
113.       packet=(AVPacket *)av_malloc(sizeof(AVPacket));
114.       av_init_packet(packet);
115.
116.       //Out Audio Param
117.       uint64_t out_channel_layout=AV_CH_LAYOUT_STEREO;
118.       //nb_samples: AAC-1024 MP3-1152
119.       int out_nb_samples=pCodecCtx->frame_size;
120.       AVSampleFormat out_sample_fmt=AV_SAMPLE_FMT_S16;
121.       int out_sample_rate=44100;
122.       int out_channels=av_get_channel_layout_nb_channels(out_channel_layout);
123.       //Out Buffer Size
124.       int out_buffer_size=av_samples_get_buffer_size(NULL,out_channels ,out_nb_samples,out_sample_fmt, 1);
125.
126.       out_buffer=(uint8_t *)av_malloc(MAX_AUDIO_FRAME_SIZE*2);
127.       pFrame=av_frame_alloc();
128.
129.       //FIX:Some Codec's Context Information is missing
130.       in_channel_layout=av_get_default_channel_layout(pCodecCtx->channels);
```

```
131.        //Swr
132.        au_convert_ctx = swr_alloc();
133.        au_convert_ctx=swr_alloc_set_opts(au_convert_ctx,out_channel_layout, out_sample_fmt, out_sample_rate,
134.            in_channel_layout,pCodecCtx->sample_fmt , pCodecCtx->sample_rate,0, NULL);
135.        swr_init(au_convert_ctx);
136.
137.        while(av_read_frame(pFormatCtx, packet)>=0){
138.            if(packet->stream_index==audioStream){
139.
140.                ret = avcodec_decode_audio4( pCodecCtx, pFrame,&got_picture, packet);
141.                if ( ret < 0 ) {
142.                    printf("Error in decoding audio frame.\n");
143.                    return -1;
144.                }
145.                if ( got_picture > 0 ){
146.                    swr_convert(au_convert_ctx,&out_buffer, MAX_AUDIO_FRAME_SIZE,(const uint8_t **)pFrame->data , pFrame->nb_samples);
147.
148.                    printf("index:%5d\t pts:%lld\t packet size:%d\n",index,packet->pts,packet->size);
149.                    //Write PCM
150.                    fwrite(out_buffer, 1, out_buffer_size, pFile);
151.                    index++;
152.                }
153.            }
154.            av_free_packet(packet);
155.        }
156.
157.        swr_free(&au_convert_ctx);
158.
159.        fclose(pFile);
160.
161.        av_free(out_buffer);
162.        // Close the codec
163.        avcodec_close(pCodecCtx);
164.        // Close the video file
165.        avformat_close_input(&pFormatCtx);
166.
167.        return 0;
168.    }
```
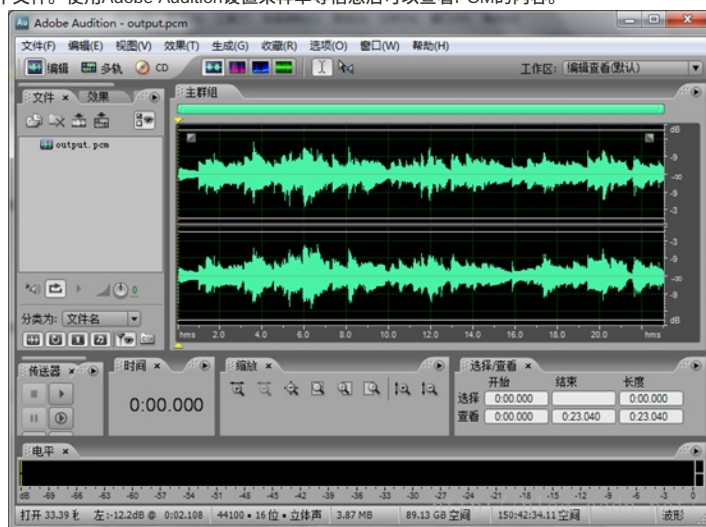
运行结果

程序运行后，会解码下面的音频文件。
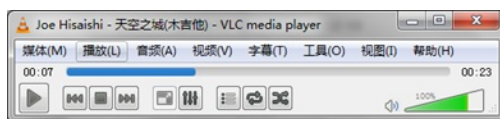
解码后的PCM采样数据被保存成了一个文件。使用Adobe Audition设置采样率等信息后可以查看PCM的内容。

SDL音频采样数据播放器

源代码

```cpp
1.  /**
2.   * 最简单的SDL2播放音频的例子 (SDL2播放PCM)
3.   * Simplest Audio Play SDL2 (SDL2 play PCM)
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.     * 本程序使用SDL2播放PCM音频采样数据。SDL实际上是对底层绘图
12.     * API（Direct3D，OpenGL）的封装，使用起来明显简单于直接调用底层
13.     * API。
14.     *
15.     * 函数调用步骤如下:
16.     *
17.     * [初始化]
18.     * SDL_Init(): 初始化SDL。
19.     * SDL_OpenAudio(): 根据参数（存储于SDL_AudioSpec）打开音频设备。
20.     * SDL_PauseAudio(): 播放音频数据。
21.     *
22.     * [循环播放数据]
23.     * SDL_Delay(): 延时等待播放完成。
24.     *
25.     * This software plays PCM raw audio data using SDL2.
26.     * SDL is a wrapper of low-level API (DirectSound).
27.     * Use SDL is much easier than directly call these low-level API.
28.     *
29.     * The process is shown as follows:
30.     *
31.     * [Init]
32.     * SDL_Init(): Init SDL.
33.     * SDL_OpenAudio(): Opens the audio device with the desired
34.     *                  parameters (In SDL_AudioSpec).
35.     * SDL_PauseAudio(): Play Audio.
36.     *
37.     * [Loop to play data]
38.     * SDL_Delay(): Wait for completetion of playback.
39.     */
40.
41.     #include <stdio.h>
42.     #include <tchar.h>
43.
44.     extern "C"
45.     {
46.     #include "sdl/SDL.h"
47.     };
48.
49.     //Buffer:
50.     //|-----------|-------------
51.     //chunk-------pos---len-----|
52.     static  Uint8  *audio_chunk;
53.     static  Uint32  audio_len;
54.     static  Uint8  *audio_pos;
55.
56.     /* Audio Callback
57.      * The audio function callback takes the following parameters:
58.      * stream: A pointer to the audio buffer to be filled
59.      * len: The length (in bytes) of the audio buffer
60.      *
61.     */
62.     void  fill_audio(void *udata,Uint8 *stream,int len){
63.         //SDL 2.0
64.         SDL_memset(stream, 0, len);
65.         if(audio_len==0)        /*  Only  play  if  we  have  data  left  */
66.             return;
67.         len=(len>audio_len?audio_len:len);   /*  Mix  as  much  data  as  possible  */
68.
69.         SDL_MixAudio(stream,audio_pos,len,SDL_MIX_MAXVOLUME);
70.         audio_pos += len;
71.         audio_len -= len;
72.     }
73.
74.     int main(int argc, char* argv[])
75.     {
76.         //Init
77.         if(SDL_Init(SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
78.             printf( "Could not initialize SDL - %s\n", SDL_GetError());
79.             return -1;
80.         }
81.         //SDL_AudioSpec
82.         SDL_AudioSpec wanted_spec;
83.         wanted_spec.freq = 44100;
84.         wanted_spec.format = AUDIO_S16SYS;
85.         wanted_spec.channels = 2;
86.         wanted_spec.silence = 0;
87.         wanted_spec.samples = 1024;
88.         wanted_spec.callback = fill_audio;
89.
90.         if (SDL_OpenAudio(&wanted_spec, NULL)<0){
91.             printf("can't open audio.\n");
92.             return -1;
93.         }
94.
95.         FILE *fp=fopen("NocturneNo2inEflat_44.1k_s16le.pcm","rb+");
96.         if(fp==NULL){
97.             printf("cannot open this file\n");
```

```
 98.            return -1;
 99.        }
100.
101.        int pcm_buffer_size=4096;
102.        char *pcm_buffer=(char *)malloc(pcm_buffer_size);
103.        int data_count=0;
104.
105.        //Play
106.        SDL_PauseAudio(0);
107.
108.        while(1){
109.            if (fread(pcm_buffer, 1, pcm_buffer_size, fp) != pcm_buffer_size){
110.                // Loop
111.                fseek(fp, 0, SEEK_SET);
112.                fread(pcm_buffer, 1, pcm_buffer_size, fp);
113.                data_count=0;
114.            }
115.            printf("Now Playing %10d Bytes data.\n",data_count);
116.            data_count+=pcm_buffer_size;
117.            //Set audio buffer (PCM data)
118.            audio_chunk = (Uint8 *) pcm_buffer;
119.            //Audio buffer length
120.            audio_len =pcm_buffer_size;
121.            audio_pos = audio_chunk;
122.
123.            while(audio_len>0)//Wait until finish
124.                SDL_Delay(1);
125.        }
126.        free(pcm_buffer);
127.        SDL_Quit();
128.        return 0;
129.    }
```

运行结果

程序运行后，会读取程序文件夹下的一个PCM采样数据文件，内容如下所示。
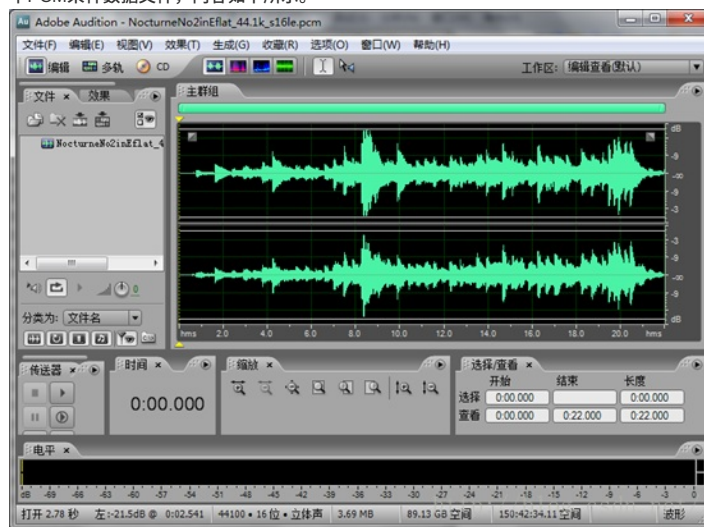
接下来会将PCM数据输出到系统的音频设备上（音响、耳机）。

下载

Simplest FFmpeg Audio Player

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

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

本程序实现了音频的解码和播放。是最简单的FFmpeg音频解码方面的教程。
通过学习本例子可以了解FFmpeg的解码流程。
项目包含3个工程：

simplest_ffmpeg_audio_player：基于FFmpeg+SDL的音频解码器

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。

文章标签： FFmpeg 音频 SDL PCM 解码

个人分类： FFMPEG

所属专栏： FFmpeg