原 最简单的基于FFMPEG+SDL的音频播放器：拆分-解码器和播放器

2015年07月17日 09:31:10　阅读数：13046

==========

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：

音频数据->PCM->音频设备

FFmpeg音频解码器实现了：

音频数据->PCM

SDL音频采样数据播放器实现了：

PCM->音频设备

FFmpeg音频解码器

源代码

[cpp]

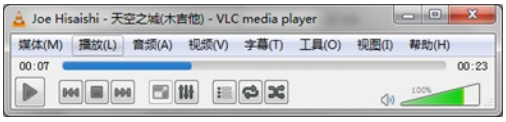
```cpp
/**
 * 最简单的基于FFmpeg的音频解码器
 * Simplest FFmpeg Audio Decoder
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序可以将音频码流（mp3，AAC等）解码为PCM采样数据。
 * 是最简单的FFmpeg音频解码方面的教程。
 * 通过学习本例子可以了解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>
```

```
#include <libavformat/avformat.h>
#include <libswresample/swresample.h>
#ifdef __cplusplus
};
#endif
#endif

#define MAX_AUDIO_FRAME_SIZE 192000 // 1 second of 48khz 32bit audio

int main(int argc, char* argv[])
{
    AVFormatContext *pFormatCtx;
    int             i, audioStream;
    AVCodecContext  *pCodecCtx;
    AVCodec         *pCodec;
    AVPacket        *packet;
    uint8_t         *out_buffer;
    AVFrame         *pFrame;
    int ret;
    uint32_t len = 0;
    int got_picture;
    int index = 0;
    int64_t in_channel_layout;
    struct SwrContext *au_convert_ctx;

    FILE *pFile=fopen("output.pcm", "wb");
    char url[]="skycity1.mp3";

    av_register_all();
    avformat_network_init();
    pFormatCtx = avformat_alloc_context();
    //Open
    if(avformat_open_input(&pFormatCtx,url,NULL,NULL)!=0){
        printf("Couldn't open input stream.\n");
        return -1;
    }
    // Retrieve stream information
    if(avformat_find_stream_info(pFormatCtx,NULL)<0){
        printf("Couldn't find stream information.\n");
        return -1;
    }
    // Dump valid information onto standard error
    av_dump_format(pFormatCtx, 0, url, false);

    // Find the first audio stream
    audioStream=-1;
    for(i=0; i < pFormatCtx->nb_streams; i++)
        if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_AUDIO){
            audioStream=i;
            break;
        }

    if(audioStream==-1){
        printf("Didn't find a audio stream.\n");
        return -1;
    }

    // Get a pointer to the codec context for the audio stream
    pCodecCtx=pFormatCtx->streams[audioStream]->codec;

    // Find the decoder for the audio stream
    pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
    if(pCodec==NULL){
        printf("Codec not found.\n");
        return -1;
    }

    // Open codec
    if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
        printf("Could not open codec.\n");
        return -1;
    }

    packet=(AVPacket *)av_malloc(sizeof(AVPacket));
    av_init_packet(packet);

    //Out Audio Param
    uint64_t out_channel_layout=AV_CH_LAYOUT_STEREO;
    //nb_samples: AAC-1024 MP3-1152
    int out_nb_samples=pCodecCtx->frame_size;
    AVSampleFormat out_sample_fmt=AV_SAMPLE_FMT_S16;
    int out_sample_rate=44100;
    int out_channels=av_get_channel_layout_nb_channels(out_channel_layout);
    //Out Buffer Size
    int out_buffer_size=av_samples_get_buffer_size(NULL,out_channels ,out_nb_samples,out_sample_fmt, 1);

    out_buffer=(uint8_t *)av_malloc(MAX_AUDIO_FRAME_SIZE*2);
    pFrame=av_frame_alloc();

    //FIX:Some Codec's Context Information is missing
    in_channel_layout=av_get_default_channel_layout(pCodecCtx->channels);
```

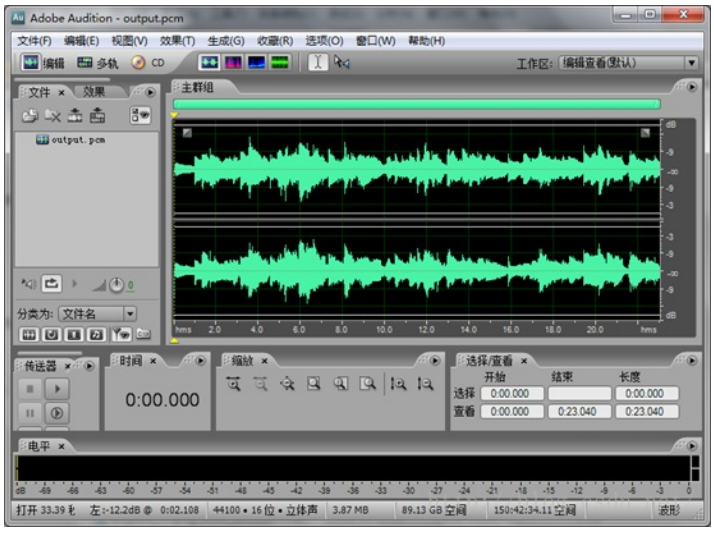
```
    //Swr
    au_convert_ctx = swr_alloc();
    au_convert_ctx=swr_alloc_set_opts(au_convert_ctx,out_channel_layout, out_sample_fmt, out_sample_rate,
        in_channel_layout,pCodecCtx->sample_fmt , pCodecCtx->sample_rate,0, NULL);
    swr_init(au_convert_ctx);

    while(av_read_frame(pFormatCtx, packet)>=0){
        if(packet->stream_index==audioStream){

            ret = avcodec_decode_audio4( pCodecCtx, pFrame,&got_picture, packet);
            if ( ret < 0 ) {
                printf("Error in decoding audio frame.\n");
                return -1;
            }
            if ( got_picture > 0 ){
                swr_convert(au_convert_ctx,&out_buffer, MAX_AUDIO_FRAME_SIZE,(const uint8_t **)pFrame->data , pFrame->nb_samples);

                printf("index:%5d\t pts:%lld\t packet size:%d\n",index,packet->pts,packet->size);
                //Write PCM
                fwrite(out_buffer, 1, out_buffer_size, pFile);
                index++;
            }
        }
        av_free_packet(packet);
    }

    swr_free(&au_convert_ctx);

    fclose(pFile);

    av_free(out_buffer);
    // Close the codec
    avcodec_close(pCodecCtx);
    // Close the video file
    avformat_close_input(&pFormatCtx);

    return 0;
}
```

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

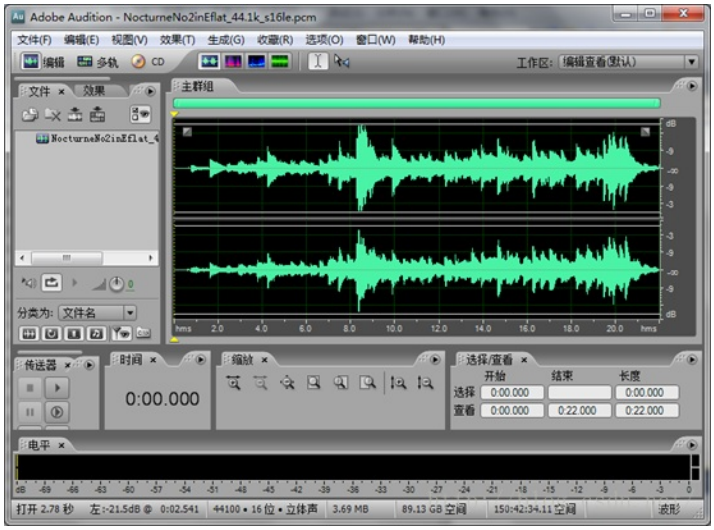
```
[cpp]
/**
 * 最简单的SDL2播放音频的例子（SDL2播放PCM）
 * Simplest Audio Play SDL2 (SDL2 play PCM)
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
```

```
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序使用SDL2播放PCM音频采样数据。SDL实际上是对底层绘图
 * API（Direct3D，OpenGL）的封装，使用起来明显简单于直接调用底层
 * API。
 *
 * 函数调用步骤如下：
 *
 * [初始化]
 * SDL_Init(): 初始化SDL。
 * SDL_OpenAudio(): 根据参数（存储于SDL_AudioSpec）打开音频设备。
 * SDL_PauseAudio(): 播放音频数据。
 *
 * [循环播放数据]
 * SDL_Delay(): 延时等待播放完成。
 *
 * This software plays PCM raw audio data using SDL2.
 * SDL is a wrapper of low-level API (DirectSound).
 * Use SDL is much easier than directly call these low-level API.
 *
 * The process is shown as follows:
 *
 * [Init]
 * SDL_Init(): Init SDL.
 * SDL_OpenAudio(): Opens the audio device with the desired
 *                  parameters (In SDL_AudioSpec).
 * SDL_PauseAudio(): Play Audio.
 *
 * [Loop to play data]
 * SDL_Delay(): Wait for completetion of playback.
 */

#include <stdio.h>
#include <tchar.h>

extern "C"
{
#include "sdl/SDL.h"
};

//Buffer:
//|-----------|-------------
//chunk-------pos---len-----|
static  Uint8  *audio_chunk;
static  Uint32  audio_len;
static  Uint8  *audio_pos;

/* Audio Callback
 * The audio function callback takes the following parameters:
 * stream: A pointer to the audio buffer to be filled
 * len: The length (in bytes) of the audio buffer
 *
*/
void  fill_audio(void *udata,Uint8 *stream,int len){
    //SDL 2.0
    SDL_memset(stream, 0, len);
    if(audio_len==0)        /*  Only  play  if  we  have  data  left  */
            return;
    len=(len>audio_len?audio_len:len);   /*  Mix  as  much  data  as  possible  */

    SDL_MixAudio(stream,audio_pos,len,SDL_MIX_MAXVOLUME);
    audio_pos += len;
    audio_len -= len;
}

int main(int argc, char* argv[])
{
    //Init
    if(SDL_Init(SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
        printf( "Could not initialize SDL - %s\n", SDL_GetError());
        return -1;
    }
    //SDL_AudioSpec
    SDL_AudioSpec wanted_spec;
    wanted_spec.freq = 44100;
    wanted_spec.format = AUDIO_S16SYS;
    wanted_spec.channels = 2;
    wanted_spec.silence = 0;
    wanted_spec.samples = 1024;
    wanted_spec.callback = fill_audio;

    if (SDL_OpenAudio(&wanted_spec, NULL)<0){
        printf("can't open audio.\n");
        return -1;
    }

    FILE *fp=fopen("NocturneNo2inEflat_44.1k_s16le.pcm","rb+");
    if(fp==NULL){
        printf("cannot open this file\n");
```

```
        return -1;
    }

    int pcm_buffer_size=4096;
    char *pcm_buffer=(char *)malloc(pcm_buffer_size);
    int data_count=0;

    //Play
    SDL_PauseAudio(0);

    while(1){
        if (fread(pcm_buffer, 1, pcm_buffer_size, fp) != pcm_buffer_size){
            // Loop
            fseek(fp, 0, SEEK_SET);
            fread(pcm_buffer, 1, pcm_buffer_size, fp);
            data_count=0;
        }
        printf("Now Playing %10d Bytes data.\n",data_count);
        data_count+=pcm_buffer_size;
        //Set audio buffer (PCM data)
        audio_chunk = (Uint8 *) pcm_buffer;
        //Audio buffer length
        audio_len =pcm_buffer_size;
        audio_pos = audio_chunk;

        while(audio_len>0)//Wait until finish
            SDL_Delay(1);
    }
    free(pcm_buffer);
    SDL_Quit();
    return 0;
}
```

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