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
/**
 * 最简单的基于FFmpeg的音频播放器  1.2
 * Simplest FFmpeg Audio Player  1.2
 *
 * 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
 *
 * 本程序实现了音频的解码和播放。
 *
 * This software decode and play audio streams.
 */

#include "stdafx.h"
#include <stdlib.h>
#include <string.h>
extern "C"
{
#include "libavcodec/avcodec.h"
#include "libavformat/avformat.h"
#include "libswresample/swresample.h"
//SDL
```

```
#include "sdl/SDL.h"
};

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


//Output PCM
#define OUTPUT_PCM 1
//Use SDL
#define USE_SDL 1

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

/* 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
 * 回调函数
*/
void  fill_audio(void *udata,Uint8 *stream,int 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[])
{
    AVFormatContext *pFormatCtx;
    int             i, audioStream;
    AVCodecContext  *pCodecCtx;
    AVCodec         *pCodec;

    char url[]="WavinFlag.aac";
    //char url[]="WavinFlag.mp3";
    //char url[]="72bian.wma";

    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(av_find_stream_info(pFormatCtx)<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;
    }

```

```
    FILE *pFile=NULL;
#if OUTPUT_PCM
    pFile=fopen("output.pcm", "wb");
#endif

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

    //Out Audio Param
    uint64_t out_channel_layout=AV_CH_LAYOUT_STEREO;
    //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);

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

    AVFrame *pFrame;
    pFrame=avcodec_alloc_frame();
//SDL----------------
#if USE_SDL
    //Init
    if(SDL_Init(SDL_INIT_VIDEO | 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 = out_sample_rate;
    wanted_spec.format = AUDIO_S16SYS;
    wanted_spec.channels = out_channels;
    wanted_spec.silence = 0;
    wanted_spec.samples = out_nb_samples;
    wanted_spec.callback = fill_audio;
    wanted_spec.userdata = pCodecCtx;

    if (SDL_OpenAudio(&wanted_spec, NULL)<0){
        printf("can't open audio.\n");
        return -1;
    }
#endif
    printf("Bitrate:\t %3d\n", pFormatCtx->bit_rate);
    printf("Decoder Name:\t %s\n", pCodecCtx->codec->long_name);
    printf("Channels:\t %d\n", pCodecCtx->channels);
    printf("Sample per Second\t %d \n", pCodecCtx->sample_rate);

    uint32_t ret,len = 0;
    int got_picture;
    int index = 0;
    //FIX:Some Codec's Context Information is missing
    int64_t in_channel_layout=av_get_default_channel_layout(pCodecCtx->channels);
    //Swr
    struct SwrContext *au_convert_ctx;
    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);

    //Play
    SDL_PauseAudio(0);

    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);

#if OUTPUT_PCM
                //Write PCM
                fwrite(out_buffer, 1, out_buffer_size, pFile);
#endif

                index++;
            }
//SDL----------------
#if USE_SDL
            //Set audio buffer (PCM data)
            audio_chunk = (Uint8 *) out_buffer;
            //Audio buffer length
            audio_len =out_buffer_size;

            audio_pos = audio_chunk;
```

```
            audio_pos = audio_chunk;

            while(audio_len>0)//Wait until finish
                SDL_Delay(1);
#endif
        }
        av_free_packet(packet);
    }

    swr_free(&au_convert_ctx);

#if USE_SDL
    SDL_CloseAudio();//Close SDL
    SDL_Quit();
#endif

#if OUTPUT_PCM
    fclose(pFile);
#endif
    av_free(out_buffer);
    avcodec_close(pCodecCtx);
    av_close_input_file(pFormatCtx);

    return 0;
}
```

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。

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_SAMPLE_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
//输出音频数据大小，一定小于输出内存。
int out_linesize;
//输出内存大小
int out_buffer_size=av_samples_get_buffer_size(&out_linesize, pCodecCtx->channels,pCodecCtx->frame_size,pCodecCtx->sample_fmt, 1);
uint8_t *out_buffer=new uint8_t[out_buffer_size];
...
au_convert_ctx = swr_alloc();
au_convert_ctx=swr_alloc_set_opts(au_convert_ctx,AV_CH_LAYOUT_STEREO, AV_SAMPLE_FMT_S16, 44100,
    pCodecCtx->channel_layout,pCodecCtx->sample_fmt , pCodecCtx->sample_rate,0, NULL);
swr_init(au_convert_ctx);

while(av_read_frame(pFormatCtx, packet)>=0){
    ......
    swr_convert(au_convert_ctx,&out_buffer, out_linesize,(const uint8_t **)pFrame->data , pFrame->nb_samples);

    ......
}
```

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
//Output PCM
#define OUTPUT_PCM 0
//Use SDL
#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相关学习资料：

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