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：28107

==========

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]

```cpp
/**
 * 最简单的基于FFmpeg的音频播放器 2
 * Simplest FFmpeg Audio Player 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。
 * 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。
 *
 * 该版本使用SDL 2.0替换了第一个版本中的SDL 1.0。
 * 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
 * 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。
 * 本例子中即SDL_memset(stream, 0, len);
 *
 * This software decode and play audio streams.
 * Suitable for beginner of FFmpeg.
 *
 * This version use SDL 2.0 instead of SDL 1.2 in version 1
 * Note:The good news for audio is that, with one exception,
 * it's entirely backwards compatible with 1.2.
 * That one really important exception: The audio callback
 * does NOT start with a fully initialized buffer anymore.
 * You must fully write to the buffer in all cases. In this
 * example it is SDL_memset(stream, 0, len);
 *
 * Version 2.0
 */
#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"
#include "SDL2/SDL.h"
};
#else
//Linux...
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavcodec/avcodec.h>
#include <libavformat/avformat.h>
#include <libswresample/swresample.h>
#include <SDL2/SDL.h>
#ifdef __cplusplus
};
#endif
#endif

#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
*/
void  fill_audio(void *udata,Uint8 *stream,int len){
    //SDL 2.0
    SDL_memset(stream, 0, len);
    if(audio_len==0)
        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;
    AVPacket        *packet;
    uint8_t         *out_buffer;
    AVFrame         *pFrame;
    SDL_AudioSpec wanted_spec;
    int ret;
    uint32_t len = 0;
    int got_picture;
    int index = 0;
    int64_t in_channel_layout;
    struct SwrContext *au_convert_ctx;

    FILE *pFile=NULL;
    char url[]="xiaoqingge.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;
    }


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

    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();
//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
    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

    //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);

    //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);
#if 1
                printf("index:%5d\t pts:%lld\t packet size:%d\n",index,packet->pts,packet->size);
#endif
```

```
#endif


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

#if USE_SDL
            while(audio_len>0)//Wait until finish
                SDL_Delay(1);

            //Set audio buffer (PCM data)
            audio_chunk = (Uint8 *) out_buffer;
            //Audio buffer length
            audio_len =out_buffer_size;
            audio_pos = audio_chunk;

#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);
    avformat_close_input(&pFormatCtx);

    return 0;
}
```

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。

```
::VS2010 Environment
call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
::include
@set INCLUDE=include;%INCLUDE%
::lib
@set LIB=lib;%LIB%
::compile and link
cl simplest_ffmpeg_audio_player.cpp /MD /link SDL.lib SDLmain.lib avcodec.lib ^
avformat.lib avutil.lib avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib ^
/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。

```
g++ simplest_ffmpeg_audio_player.cpp -g -o simplest_ffmpeg_audio_player.exe \
-I /usr/local/include -L /usr/local/lib \
-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。

```
gcc simplest_ffmpeg_audio_player.cpp -g -o simplest_ffmpeg_audio_player.out -I /usr/local/include -L /usr/local/lib \
-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