# 最简单的基于FFMPEG+SDL的视频播放器：拆分-解码器和播放器

2015年07月16日 21:57:56　阅读数：15970

==========

最简单的基于FFmpeg的视频播放器系列文章列表：

100行代码实现最简单的基于FFMPEG+SDL的视频播放器（SDL1.x）

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

最简单的基于FFmpeg的解码器-纯净版（不包含libavformat）

最简单的基于FFMPEG+SDL的视频播放器：拆分-解码器和播放器

最简单的基于FFMPEG的Helloworld程序

==========

本文补充记录《最简单的基于FFMPEG+SDL的视频播放器》中的两个例子：FFmpeg视频解码器和SDL像素数据播放器。这两个部分是从视频播放器中拆分出来的两个例子。FFmpeg视频解码器实现了视频数据到YUV数据的解码，而SDL像素数据播放器实现了YUV数据的显示。简而言之，原先的FFmpeg+SDL视频播放器实现了：

**视频数据->YUV->显示器**

FFmpeg视频解码器实现了：

**视频数据->YUV**

SDL像素数据播放器实现了：

**YUV->显示器**

## FFmpeg视频解码器

### 源代码

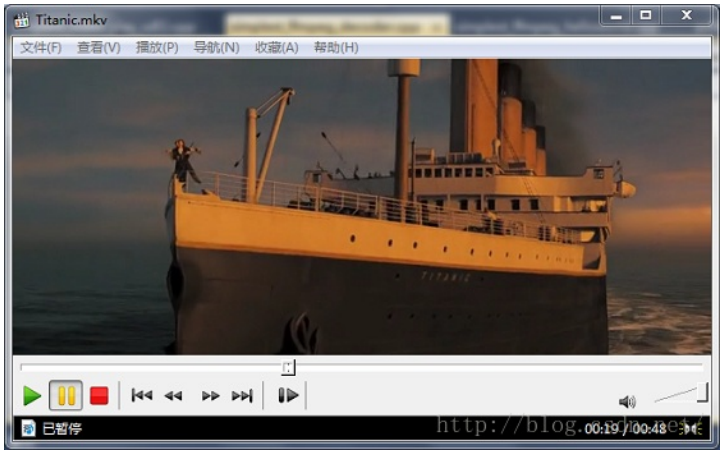
```cpp
/**
 * 最简单的基于FFmpeg的视频解码器
 * Simplest FFmpeg Decoder
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 *
 * 本程序实现了视频文件解码为YUV数据。它使用了libavcodec和
 * libavformat。是最简单的FFmpeg视频解码方面的教程。
 * 通过学习本例子可以了解FFmpeg的解码流程。
 * This software is a simplest decoder based on FFmpeg.
 * It decodes video to YUV pixel data.
 * It uses libavcodec and libavformat.
 * Suitable for beginner of FFmpeg.
 *
 */



#include <stdio.h>

#define __STDC_CONSTANT_MACROS

#ifdef _WIN32
//Windows
extern "C"
{
#include "libavcodec/avcodec.h"
#include "libavformat/avformat.h"
#include "libswscale/swscale.h"
#include "libavutil/imgutils.h"
};
#else
//Linux...
```

```
//Linux...
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavcodec/avcodec.h>
#include <libavformat/avformat.h>
#include <libswscale/swscale.h>
#include <libavutil/imgutils.h>
#ifdef __cplusplus
};
#endif
#endif


int main(int argc, char* argv[])
{
    AVFormatContext *pFormatCtx;
    int             i, videoindex;
    AVCodecContext  *pCodecCtx;
    AVCodec         *pCodec;
    AVFrame *pFrame,*pFrameYUV;
    unsigned char *out_buffer;
    AVPacket *packet;
    int y_size;
    int ret, got_picture;
    struct SwsContext *img_convert_ctx;

    char filepath[]="Titanic.mkv";

    FILE *fp_yuv=fopen("output.yuv","wb+");

    av_register_all();
    avformat_network_init();
    pFormatCtx = avformat_alloc_context();

    if(avformat_open_input(&pFormatCtx,filepath,NULL,NULL)!=0){
        printf("Couldn't open input stream.\n");
        return -1;
    }
    if(avformat_find_stream_info(pFormatCtx,NULL)<0){
        printf("Couldn't find stream information.\n");
        return -1;
    }
    videoindex=-1;
    for(i=0; i<pFormatCtx->nb_streams; i++)
        if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
            videoindex=i;
            break;
        }

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

    pCodecCtx=pFormatCtx->streams[videoindex]->codec;
    pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
    if(pCodec==NULL){
        printf("Codec not found.\n");
        return -1;
    }
    if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
        printf("Could not open codec.\n");
        return -1;
    }

    pFrame=av_frame_alloc();
    pFrameYUV=av_frame_alloc();
    out_buffer=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P,  pCodecCtx->width, pCodecCtx->height,1));
    av_image_fill_arrays(pFrameYUV->data, pFrameYUV->linesize,out_buffer,
        AV_PIX_FMT_YUV420P,pCodecCtx->width, pCodecCtx->height,1);



    packet=(AVPacket *)av_malloc(sizeof(AVPacket));
    //Output Info-----------------------------
    printf("--------------- File Information ----------------\n");
    av_dump_format(pFormatCtx,0,filepath,0);
    printf("-------------------------------------------------\r");
    img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
        pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);

    while(av_read_frame(pFormatCtx, packet)>=0){
        if(packet->stream_index==videoindex){
            ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
            if(ret < 0){
                printf("Decode Error.\n");
                return -1;
            }
            if(got_picture){
                sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
```

```
                    pFrameYUV->data, pFrameYUV->linesize);

                y_size=pCodecCtx->width*pCodecCtx->height;
                fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
                fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
                fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
                printf("Succeed to decode 1 frame!\n");

            }
        }
        av_free_packet(packet);
    }
    //flush decoder
    //FIX: Flush Frames remained in Codec
    while (1) {
        ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
        if (ret < 0)
            break;
        if (!got_picture)
            break;
        sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
            pFrameYUV->data, pFrameYUV->linesize);

        int y_size=pCodecCtx->width*pCodecCtx->height;
        fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
        fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
        fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V

        printf("Flush Decoder: Succeed to decode 1 frame!\n");
    }

    sws_freeContext(img_convert_ctx);

    fclose(fp_yuv);

    av_frame_free(&pFrameYUV);
    av_frame_free(&pFrame);
    avcodec_close(pCodecCtx);
    avformat_close_input(&pFormatCtx);

    return 0;
}
```

## 运行结果

程序运行后，会解码下面的视频文件。




解码后的YUV420P数据被保存成了一个文件。使用YUV播放器设置宽高之后可以查看YUV内容。


## SDL像素数据播放器

### 源代码

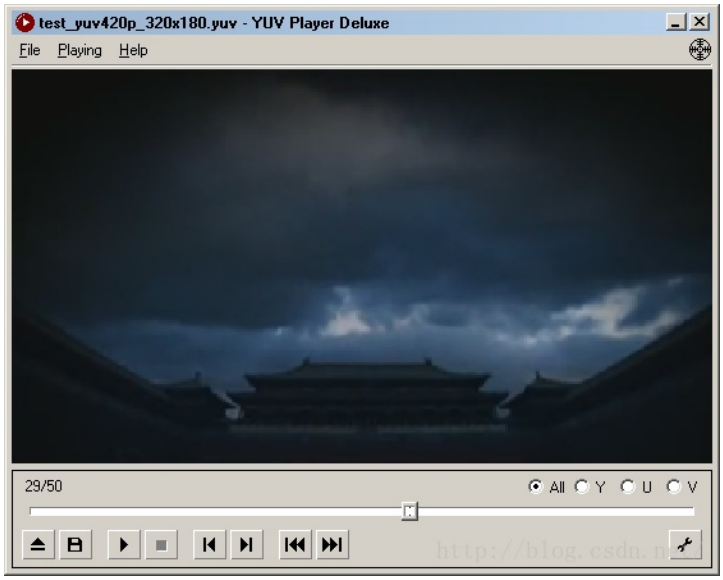
```cpp
/**
 * 最简单的SDL2播放视频的例子（SDL2播放RGB/YUV）
 * Simplest Video Play SDL2 (SDL2 play RGB/YUV)
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序使用SDL2播放RGB/YUV视频像素数据。SDL实际上是对底层绘图
 * 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_CreateWindow(): 创建窗口（Window）。
 * SDL_CreateRenderer(): 基于窗口创建渲染器（Render）。
 * SDL_CreateTexture(): 创建纹理（Texture）。
 *
 * [循环渲染数据]
 * SDL_UpdateTexture(): 设置纹理的数据。
 * SDL_RenderCopy(): 纹理复制给渲染器。
 * SDL_RenderPresent(): 显示。
 *
 * This software plays RGB/YUV raw video data using SDL2.
 * SDL is a wrapper of low-level API (Direct3D, OpenGL).
 * Use SDL is much easier than directly call these low-level API.
 *
 * The process is shown as follows:
 *
 * [Init]
 * SDL_Init(): Init SDL.
 * SDL_CreateWindow(): Create a Window.
 * SDL_CreateRenderer(): Create a Render.
 * SDL_CreateTexture(): Create a Texture.
 *
 * [Loop to Render data]
 * SDL_UpdateTexture(): Set Texture's data.
 * SDL_RenderCopy(): Copy Texture to Render.
 * SDL_RenderPresent(): Show.
 */

#include <stdio.h>

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

const int bpp=12;

int screen_w=500,screen_h=500;
const int pixel_w=320,pixel_h=180;

unsigned char buffer[pixel_w*pixel_h*bpp/8];


```

```
//Refresh Event
#define REFRESH_EVENT  (SDL_USEREVENT + 1)

#define BREAK_EVENT  (SDL_USEREVENT + 2)

int thread_exit=0;

int refresh_video(void *opaque){
    thread_exit=0;
    while (!thread_exit) {
        SDL_Event event;
        event.type = REFRESH_EVENT;
        SDL_PushEvent(&event);
        SDL_Delay(40);
    }
    thread_exit=0;
    //Break
    SDL_Event event;
    event.type = BREAK_EVENT;
    SDL_PushEvent(&event);

    return 0;
}

int main(int argc, char* argv[])
{
    if(SDL_Init(SDL_INIT_VIDEO)) {
        printf( "Could not initialize SDL - %s\n", SDL_GetError());
        return -1;
    }

    SDL_Window *screen;
    //SDL 2.0 Support for multiple windows
    screen = SDL_CreateWindow("Simplest Video Play SDL2", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
        screen_w, screen_h,SDL_WINDOW_OPENGL|SDL_WINDOW_RESIZABLE);
    if(!screen) {
        printf("SDL: could not create window - exiting:%s\n",SDL_GetError());
        return -1;
    }
    SDL_Renderer* sdlRenderer = SDL_CreateRenderer(screen, -1, 0);

    Uint32 pixformat=0;

    //IYUV: Y + U + V  (3 planes)
    //YV12: Y + V + U  (3 planes)
    pixformat= SDL_PIXELFORMAT_IYUV;

    SDL_Texture* sdlTexture = SDL_CreateTexture(sdlRenderer,pixformat, SDL_TEXTUREACCESS_STREAMING,pixel_w,pixel_h);

    FILE *fp=NULL;
    fp=fopen("test_yuv420p_320x180.yuv","rb+");

    if(fp==NULL){
        printf("cannot open this file\n");
        return -1;
    }

    SDL_Rect sdlRect;

    SDL_Thread *refresh_thread = SDL_CreateThread(refresh_video,NULL,NULL);
    SDL_Event event;
    while(1){
        //Wait
        SDL_WaitEvent(&event);
        if(event.type==REFRESH_EVENT){
            if (fread(buffer, 1, pixel_w*pixel_h*bpp/8, fp) != pixel_w*pixel_h*bpp/8){
                // Loop
                fseek(fp, 0, SEEK_SET);
                fread(buffer, 1, pixel_w*pixel_h*bpp/8, fp);
            }

            SDL_UpdateTexture( sdlTexture, NULL, buffer, pixel_w);

            //FIX: If window is resize
            sdlRect.x = 0;
            sdlRect.y = 0;
            sdlRect.w = screen_w;
            sdlRect.h = screen_h;

            SDL_RenderClear( sdlRenderer );
            SDL_RenderCopy( sdlRenderer, sdlTexture, NULL, &sdlRect);
            SDL_RenderPresent( sdlRenderer );

        }else if(event.type==SDL_WINDOWEVENT){
            //If Resize
            SDL_GetWindowSize(screen,&screen_w,&screen_h);
        }else if(event.type==SDL_QUIT){
            thread_exit=1;
        }else if(event.type==BREAK_EVENT){
            break;
        }
```

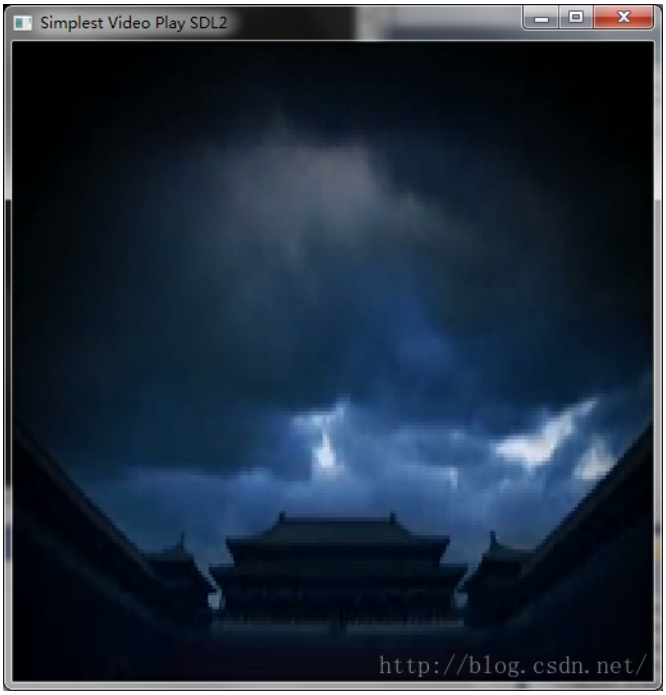
```
    }
    SDL_Quit();
    return 0;
}
```

## 运行结果

程序运行后，会读取程序文件夹下的一个YUV420P文件，内容如下所示。




接下来会将YUV内容绘制在弹出的窗口中。




## 下载

**Simplest FFmpeg Player**

**项目主页**

SourceForge：https://sourceforge.net/projects/simplestffmpegplayer/
Github：https://github.com/leixiaohua1020/simplest_ffmpeg_player

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

CSDN下载地址：http://download.csdn.net/detail/leixiaohua1020/8924321

本程序实现了视频文件的解码和显示（支持HEVC，H.264，MPEG2等）。
是最简单的FFmpeg视频解码方面的教程。
通过学习本例子可以了解FFmpeg的解码流程。
项目包含6个工程：

simplest_ffmpeg_player：标准版，FFmpeg学习的开始。
simplest_ffmpeg_player_su：SU（SDL Update）版，加入了简单的SDL的Event。
simplest_ffmpeg_decoder：一个包含了封装格式处理功能的解码器。使用了libavcodec和libavformat。
simplest_ffmpeg_decoder_pure：一个纯净的解码器。只使用libavcodec（没有使用libavformat）。
simplest_video_play_sdl2：使用SDL2播放YUV的例子。
simplest_ffmpeg_helloworld：输出FFmpeg类库的信息。

文章标签： FFmpeg SDL 解码 播放 YUV
个人分类： FFMPEG
所属专栏： FFmpeg