

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]
1.  /**
2.   * 最简单的基于FFmpeg的视频解码器
3.   * Simplest FFmpeg Decoder
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.  *
12.  * 本程序实现了视频文件解码为YUV数据。它使用了libavcodec和
13.  * libavformat。是最简单的FFmpeg视频解码方面的教程。
14.  * 通过学习本例子可以了解FFmpeg的解码流程。
15.  * This software is a simplest decoder based on FFmpeg.
16.  * It decodes video to YUV pixel data.
17.  * It uses libavcodec and libavformat.
18.  * Suitable for beginner of FFmpeg.
19.  *
20.  */
21.
22.
23.
24. #include <stdio.h>
25.
26. #define __STDC_CONSTANT_MACROS
27.
28. #ifdef _WIN32
29. //Windows
30. extern "C"
31. {
32. #include "libavcodec/avcodec.h"
33. #include "libavformat/avformat.h"
34. #include "libswscale/swscale.h"
35. #include "libavutil/imgutils.h"
36. };
37. #else
38. //linux ...
```

```

39. // extern ...
40. #ifdef __cplusplus
41. extern "C"
42. {
43. #endif
44. #include <libavcodec/avcodec.h>
45. #include <libavformat/avformat.h>
46. #include <libswscale/swscale.h>
47. #include <libavutil/imgutils.h>
48. #ifdef __cplusplus
49. };
50. #endif
51.
52.
53. int main(int argc, char* argv[])
54. {
55.     AVFormatContext *pFormatCtx;
56.     int i, videoindex;
57.     AVCodecContext *pCodecCtx;
58.     AVCodec *pCodec;
59.     AVFrame *pFrame,*pFrameYUV;
60.     unsigned char *out_buffer;
61.     AVPacket *packet;
62.     int y_size;
63.     int ret, got_picture;
64.     struct SwsContext *img_convert_ctx;
65.
66.     char filepath[]="Titanic.mkv";
67.
68.     FILE *fp_yuv=fopen("output.yuv","wb+");
69.
70.     av_register_all();
71.     avformat_network_init();
72.     pFormatCtx = avformat_alloc_context();
73.
74.     if(avformat_open_input(&pFormatCtx,filepath,NULL,NULL)!=0){
75.         printf("Couldn't open input stream.\n");
76.         return -1;
77.     }
78.     if(avformat_find_stream_info(pFormatCtx,NULL)<0){
79.         printf("Couldn't find stream information.\n");
80.         return -1;
81.     }
82.     videoindex=-1;
83.     for(i=0; i<pFormatCtx->nb_streams; i++){
84.         if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
85.             videoindex=i;
86.             break;
87.         }
88.     }
89.     if(videoindex==-1){
90.         printf("Didn't find a video stream.\n");
91.         return -1;
92.     }
93.
94.     pCodecCtx=pFormatCtx->streams[videoindex]->codec;
95.     pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
96.     if(pCodec==NULL){
97.         printf("Codec not found.\n");
98.         return -1;
99.     }
100.    if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
101.        printf("Could not open codec.\n");
102.        return -1;
103.    }
104.
105.    pFrame=av_frame_alloc();
106.    pFrameYUV=av_frame_alloc();
107.    out_buffer=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height,1));
108.    av_image_fill_arrays(pFrameYUV->data, pFrameYUV->linesize,out_buffer,
109.        AV_PIX_FMT_YUV420P,pCodecCtx->width, pCodecCtx->height,1);
110.
111.
112.
113.    packet=(AVPacket *)av_malloc(sizeof(AVPacket));
114.    //Output Info-----
115.    printf("----- File Information ----- \n");
116.    av_dump_format(pFormatCtx,0,filepath,0);
117.    printf("----- \n");
118.    img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
119.        pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
120.
121.    while(av_read_frame(pFormatCtx, packet)>=0){
122.        if(packet->stream_index==videoindex){
123.            ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
124.            if(ret < 0){
125.                printf("Decode Error.\n");
126.                return -1;
127.            }
128.            if(got_picture){
129.                sws scale(img convert ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,

```

```

130.         pFrameYUV->data, pFrameYUV->linesize);
131.
132.         y_size=pCodecCtx->width*pCodecCtx->height;
133.         fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
134.         fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
135.         fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
136.         printf("Succeed to decode 1 frame!\n");
137.
138.     }
139. }
140. av_free_packet(packet);
141. }
142. //flush decoder
143. //FIX: Flush Frames remained in Codec
144. while (1) {
145.     ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
146.     if (ret < 0)
147.         break;
148.     if (!got_picture)
149.         break;
150.     sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
151.         pFrameYUV->data, pFrameYUV->linesize);
152.
153.     int y_size=pCodecCtx->width*pCodecCtx->height;
154.     fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
155.     fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
156.     fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
157.
158.     printf("Flush Decoder: Succeed to decode 1 frame!\n");
159. }
160.
161. sws_freeContext(img_convert_ctx);
162.
163. fclose(fp_yuv);
164.
165. av_frame_free(&pFrameYUV);
166. av_frame_free(&pFrame);
167. avcodec_close(pCodecCtx);
168. avformat_close_input(&FormatCtx);
169.
170. return 0;
171. }

```

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]
1.  /**
2.   * 最简单的SDL2播放视频的例子（SDL2播放RGB/YUV）
3.   * Simplest Video Play SDL2 (SDL2 play RGB/YUV)
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RGB/YUV视频像素数据。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_CreateWindow(): 创建窗口 (Window) 。
20.  * SDL_CreateRenderer(): 基于窗口创建渲染器 (Render) 。
21.  * SDL_CreateTexture(): 创建纹理 (Texture) 。
22.  *
23.  * [循环渲染数据]
24.  * SDL_UpdateTexture(): 设置纹理的数据。
25.  * SDL_RenderCopy(): 纹理复制给渲染器。
26.  * SDL_RenderPresent(): 显示。
27.  *
28.  * This software plays RGB/YUV raw video data using SDL2

```

```

28.  * This software plays raw/rgb raw video data using SDL2.
29.  * SDL is a wrapper of low-level API (Direct3D, OpenGL).
30.  * Use SDL is much easier than directly call these low-level API.
31.  *
32.  * The process is shown as follows:
33.  *
34.  * [Init]
35.  * SDL_Init(): Init SDL.
36.  * SDL_CreateWindow(): Create a Window.
37.  * SDL_CreateRenderer(): Create a Render.
38.  * SDL_CreateTexture(): Create a Texture.
39.  *
40.  * [Loop to Render data]
41.  * SDL_UpdateTexture(): Set Texture's data.
42.  * SDL_RenderCopy(): Copy Texture to Render.
43.  * SDL_RenderPresent(): Show.
44.  */
45.
46. #include <stdio.h>
47.
48. extern "C"
49. {
50.     #include "sdl/SDL.h"
51. };
52.
53. const int bpp=12;
54.
55. int screen_w=500,screen_h=500;
56. const int pixel_w=320,pixel_h=180;
57.
58. unsigned char buffer[pixel_w*pixel_h*bpp/8];
59.
60.
61. //Refresh Event
62. #define REFRESH_EVENT (SDL_USEREVENT + 1)
63.
64. #define BREAK_EVENT (SDL_USEREVENT + 2)
65.
66. int thread_exit=0;
67.
68. int refresh_video(void *opaque){
69.     thread_exit=0;
70.     while (!thread_exit) {
71.         SDL_Event event;
72.         event.type = REFRESH_EVENT;
73.         SDL_PushEvent(&event);
74.         SDL_Delay(40);
75.     }
76.     thread_exit=0;
77.     //Break
78.     SDL_Event event;
79.     event.type = BREAK_EVENT;
80.     SDL_PushEvent(&event);
81.
82.     return 0;
83. }
84.
85. int main(int argc, char* argv[])
86. {
87.     if(SDL_Init(SDL_INIT_VIDEO)) {
88.         printf("Could not initialize SDL - %s\n", SDL_GetError());
89.         return -1;
90.     }
91.
92.     SDL_Window *screen;
93.     //SDL 2.0 Support for multiple windows
94.     screen = SDL_CreateWindow("Simplest Video Play SDL2", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
95.         screen_w, screen_h, SDL_WINDOW_OPENGL|SDL_WINDOW_RESIZABLE);
96.     if(!screen) {
97.         printf("SDL: could not create window - exiting:%s\n",SDL_GetError());
98.         return -1;
99.     }
100.    SDL_Renderer* sdlRenderer = SDL_CreateRenderer(screen, -1, 0);
101.
102.    Uint32 pixformat=0;
103.
104.    //IYUV: Y + U + V (3 planes)
105.    //YV12: Y + V + U (3 planes)
106.    pixformat= SDL_PIXELFORMAT_IYUV;
107.
108.    SDL_Texture* sdlTexture = SDL_CreateTexture(sdlRenderer,pixformat, SDL_TEXTUREACCESS_STREAMING,pixel_w,pixel_h);
109.
110.    FILE *fp=NULL;
111.    fp=fopen("test_yuv420p_320x180.yuv","rb+");
112.
113.    if(fp==NULL){
114.        printf("cannot open this file\n");
115.        return -1;
116.    }
117.
118.    SDL_Rect sdlRect;
119.

```

```

120.     SDL_Thread *refresh_thread = SDL_CreateThread(refresh_video,NULL,NULL);
121.     SDL_Event event;
122.     while(1){
123.         //Wait
124.         SDL_WaitEvent(&event);
125.         if(event.type==REFRESH_EVENT){
126.             if (fread(buffer, 1, pixel_w*pixel_h*bpp/8, fp) != pixel_w*pixel_h*bpp/8){
127.                 // Loop
128.                 fseek(fp, 0, SEEK_SET);
129.                 fread(buffer, 1, pixel_w*pixel_h*bpp/8, fp);
130.             }
131.
132.             SDL_UpdateTexture( sdlTexture, NULL, buffer, pixel_w);
133.
134.             //FIX: If window is resize
135.             sdlRect.x = 0;
136.             sdlRect.y = 0;
137.             sdlRect.w = screen_w;
138.             sdlRect.h = screen_h;
139.
140.             SDL_RenderClear( sdlRenderer );
141.             SDL_RenderCopy( sdlRenderer, sdlTexture, NULL, &sdlRect);
142.             SDL_RenderPresent( sdlRenderer );
143.
144.             }else if(event.type==SDL_WINDOWEVENT){
145.                 //If Resize
146.                 SDL_GetWindowSize(screen,&screen_w,&screen_h);
147.             }else if(event.type==SDL_QUIT){
148.                 thread_exit=1;
149.             }else if(event.type==BREAK_EVENT){
150.                 break;
151.             }
152.         }
153.     SDL_Quit();
154.     return 0;
155. }

```

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](#)

此PDF由spggy生成,请尊重原作者版权!!!

我的邮箱:liushide@163.com