

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

2014年08月31日 01:20:14 阅读数：79214

=====

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](#)

=====

### 简介

之前做过一个FFMPEG+SDL的简单播放器：《 [100行代码实现最简单的基于FFMPEG+SDL的视频播放器](#) 》。该播放器采用SDL1.2显示视频。最近有不少人反映SDL已经升级到2.0版本了，甚至官网的Wiki上都只有SDL2.0的文档了，因此下载了SDL 2.0 并且进行了简单的研究。随后对此前的播放器进行了修改，将SDL1.2换成了SDL2.0。

注：《 [100行代码实现最简单的基于FFMPEG+SDL的视频播放器](#) 》文章中提到的很多知识这里不再重复记录。本文重点记录SDL1.2与SDL2.0的不同。

平台使用了VC2010，FFmpeg类库使用了最近的版本，SDL使用2.0版本。

### Simplest FFmpeg Player 2

#### 项目主页

SourceForge：<https://sourceforge.net/projects/simplestffmpegplayer/>

Github：[https://github.com/leixiaohua1020/simplest\\_ffmpeg\\_player](https://github.com/leixiaohua1020/simplest_ffmpeg_player)

开源中国：[http://git.oschina.net/leixiaohua1020/simplest\\_ffmpeg\\_player](http://git.oschina.net/leixiaohua1020/simplest_ffmpeg_player)

### 流程图

FFmpeg解码一个视频流程如下图所示：

□

SDL2.0显示YUV的流程图：

□

对比SDL1.2的流程图，发现变化还是很大的。几乎所有的API都发生了变化。但是函数和变量有一定的对应关系：

SDL\_SetVideoMode()——SDL\_CreateWindow()

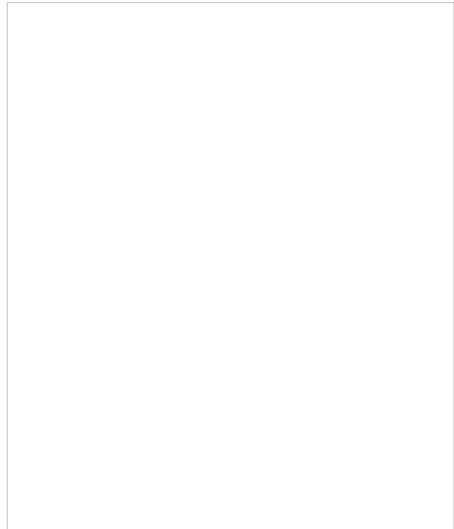
SDL\_Surface——SDL\_Window

SDL\_CreateYUVOverlay()——SDL\_CreateTexture()

SDL\_Overlay——SDL\_Texture

不再一一例举。

下图为SDL1.x显示YUV的流程图。



简单解释各个变量的作用：

SDL\_Window就是使用SDL的时候弹出的那个窗口。在SDL1.x版本中，只可以创建一个一个窗口。在SDL2.0版本中，可以创建多个窗口。

SDL\_Texture用于显示YUV数据。一个SDL\_Texture对应一帧YUV数据。

SDL\_Renderer用于渲染SDL\_Texture至SDL\_Window。

SDL\_Rect用于确定SDL\_Texture显示的位置。注意：一个SDL\_Texture可以指定多个不同的SDL\_Rect，这样就可以在SDL\_Window不同位置显示相同的内容（使用SDL\_RenderCopy()函数）。

它们的关系如下图所示：

□

下图举了个例子，指定了4个SDL\_Rect，可以实现4分屏的显示。

□

## simplest\_ffmpeg\_player（标准版）代码

最基础的版本，学习的开始。

```
[cpp]
1.  /**
2.   * 最简单的基于FFmpeg的视频播放器 2
3.   * Simplest FFmpeg Player 2
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.  * 第2版使用SDL2.0取代了第一版中的SDL1.2
12.  * Version 2 use SDL 2.0 instead of SDL 1.2 in version 1.
13.  *
14.  * 本程序实现了视频文件的解码和显示(支持HEVC, H.264, MPEG2等)。
15.  * 是最简单的FFmpeg视频解码方面的教程。
16.  * 通过学习本例子可以了解FFmpeg的解码流程。
17.  * This software is a simplest video player based on FFmpeg.
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. #include "SDL2/SDL.h"
37. };
38. #else
39. //Linux...
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 <SDL2/SDL.h>
48. #include <libavutil/imgutils.h>
49. #ifdef __cplusplus
50. };
51. #endif
52. #endif
53.
54. //Output YUV420P data as a file
55. #define OUTPUT_YUV420P 0
56.
57. int main(int argc, char* argv[])
58. {
59.     AVFormatContext *pFormatCtx;
60.     int i, videoindex;
61.     AVCodecContext *pCodecCtx;
62.     AVCodec *pCodec;
63.     AVFrame *pFrame,*pFrameYUV;
64.     unsigned char *out_buffer;
65.     AVPacket *packet;
66.     int y_size;
67.     int ret, got_picture;
68.     struct SwsContext *img_convert_ctx;
69.
70.     char filepath[]="bigbuckbunny_480x272.h265";
71.     //SDL-----
72.     int screen_w=0,screen_h=0;
73.     SDL_Window *screen;
74.     SDL_Renderer* sdlRenderer;
75.     SDL_Texture* sdlTexture;
76.     SDL_Rect sdlRect;
77.
78.     FILE *fp_yuv;
79.
80.     av_register_all();
81.     avformat_network_init();
82.     pFormatCtx = avformat_alloc_context();
83.
84.     if(avformat_open_input(&pFormatCtx,filepath,NULL,NULL)!=0){
85.         printf("Couldn't open input stream.\n");
86.         return -1;
87.     }
88.     if(avformat_find_stream_info(pFormatCtx,NULL)<0){
89.         printf("Couldn't find stream information.\n");
90.         return -1;
91.     }
92.     videoindex=-1;
93.     for(i=0; i<pFormatCtx->nb_streams; i++){
94.         if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
95.             videoindex=i;
96.             break;
97.         }
98.     }
99.     if(videoindex==-1){
100.         printf("Didn't find a video stream.\n");
101.         return -1;
102.     }
103.     pCodecCtx=pFormatCtx->streams[videoindex]->codec;
104.     pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
105.     if(pCodec==NULL){
106.         printf("Codec not found.\n");
107.         return -1;
108.     }
109.     if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
110.         printf("Could not open codec.\n");
111.         return -1;
112.     }
113.
114.     pFrame=av_frame_alloc();
115.     pFrameYUV=av_frame_alloc();
116.     out_buffer=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height,1));
117.     av_image_fill_arrays(pFrameYUV->data, pFrameYUV->linesize,out_buffer,
118.         AV_PIX_FMT_YUV420P,pCodecCtx->width, pCodecCtx->height,1);
119.
120.     packet=(AVPacket *)av_malloc(sizeof(AVPacket));
121.     //Output Info-----
122.     printf("----- File Information ----- \n");
123.     av_dump_format(pFormatCtx,0,filepath,0);
124.     printf("----- \n");
125.     img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,

```

```

126.         pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
127.
128. #if OUTPUT_YUV420P
129.     fp_yuv=fopen("output.yuv","wb+");
130. #endif
131.
132.     if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
133.         printf("Could not initialize SDL - %s\n", SDL_GetError());
134.         return -1;
135.     }
136.
137.     screen_w = pCodecCtx->width;
138.     screen_h = pCodecCtx->height;
139.     //SDL 2.0 Support for multiple windows
140.     screen = SDL_CreateWindow("Simplest ffmpeg player's Window", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
141.         screen_w, screen_h,
142.         SDL_WINDOW_OPENGL);
143.
144.     if(!screen) {
145.         printf("SDL: could not create window - exiting:%s\n",SDL_GetError());
146.         return -1;
147.     }
148.
149.     sdlRenderer = SDL_CreateRenderer(screen, -1, 0);
150.     //IYUV: Y + U + V (3 planes)
151.     //YV12: Y + V + U (3 planes)
152.     sdlTexture = SDL_CreateTexture(sdlRenderer, SDL_PIXELFORMAT_IYUV, SDL_TEXTUREACCESS_STREAMING,pCodecCtx->width,pCodecCtx->height
    );
153.
154.     sdlRect.x=0;
155.     sdlRect.y=0;
156.     sdlRect.w=screen_w;
157.     sdlRect.h=screen_h;
158.
159.     //SDL End-----
160.     while(av_read_frame(pFormatCtx, packet)>=0){
161.         if(packet->stream_index==videoindex){
162.             ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
163.             if(ret < 0){
164.                 printf("Decode Error.\n");
165.                 return -1;
166.             }
167.             if(got_picture){
168.                 sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
169.                     pFrameYUV->data, pFrameYUV->linesize);
170.
171. #if OUTPUT_YUV420P
172.                 y_size=pCodecCtx->width*pCodecCtx->height;
173.                 fwrite(pFrameYUV->data[0],1,y_size,fp_yuv); //Y
174.                 fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv); //U
175.                 fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv); //V
176. #endif
177.                 //SDL-----
178. #if 0
179.                 SDL_UpdateTexture( sdlTexture, NULL, pFrameYUV->data[0], pFrameYUV->linesize[0] );
180. #else
181.                 SDL_UpdateYUVTexture(sdlTexture, &sdlRect,
182.                     pFrameYUV->data[0], pFrameYUV->linesize[0],
183.                     pFrameYUV->data[1], pFrameYUV->linesize[1],
184.                     pFrameYUV->data[2], pFrameYUV->linesize[2]);
185. #endif
186.
187.                 SDL_RenderClear( sdlRenderer );
188.                 SDL_RenderCopy( sdlRenderer, sdlTexture, NULL, &sdlRect);
189.                 SDL_RenderPresent( sdlRenderer );
190.                 //SDL End-----
191.                 //Delay 40ms
192.                 SDL_Delay(40);
193.             }
194.         }
195.         av_free_packet(packet);
196.     }
197.     //flush decoder
198.     //FIX: Flush Frames remained in Codec
199.     while (1) {
200.         ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
201.         if (ret < 0)
202.             break;
203.         if (!got_picture)
204.             break;
205.         sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
206.             pFrameYUV->data, pFrameYUV->linesize);
207. #if OUTPUT_YUV420P
208.         int y_size=pCodecCtx->width*pCodecCtx->height;
209.         fwrite(pFrameYUV->data[0],1,y_size,fp_yuv); //Y
210.         fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv); //U
211.         fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv); //V
212. #endif
213.         //SDL-----
214.         SDL_UpdateTexture( sdlTexture, &sdlRect, pFrameYUV->data[0], pFrameYUV->linesize[0] );
215.         SDL_RenderClear( sdlRenderer );
216.         SDL_RenderCopy( sdlRenderer, sdlTexture, NULL, &sdlRect);

```

```

217.         SDL_RenderCopy( sRenderer, sTexture, NULL, sRectSrc );
218.         SDL_RenderPresent( sRenderer );
219.         //SDL_End-----
220.         //Delay 40ms
221.         SDL_Delay(40);
222.     }
223.
224.     sws_freeContext(img_convert_ctx);
225.
226. #if OUTPUT_YUV420P
227.     fclose(fp_yuv);
228. #endif
229.
230.     SDL_Quit();
231.
232.     av_frame_free(&FrameYUV);
233.     av_frame_free(&Frame);
234.     avcodec_close(pCodecCtx);
235.     avformat_close_input(&FormatCtx);
236.
237.     return 0;
238. }

```

## simplest\_ffmpeg\_player\_su (SU版) 代码

标准版的基础之上引入了SDL的Event。

效果如下：

- (1) SDL弹出的窗口可以移动了
- (2) 画面显示是严格的40ms一帧

```

1.  /**
2.   * 最简单的基于FFmpeg的视频播放器2(SDL升级版)
3.   * Simplest Ffmpeg Player 2(SDL Update)
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.  * 第2版使用SDL2.0取代了第一版中的SDL1.2
12.  * Version 2 use SDL 2.0 instead of SDL 1.2 in version 1.
13.  *
14.  * 本程序实现了视频文件的解码和显示(支持HEVC, H.264, MPEG2等)。
15.  * 是最简单的FFmpeg视频解码方面的教程。
16.  * 通过学习本例子可以了解FFmpeg的解码流程。
17.  * 本版本中使用SDL消息机制刷新视频画面。
18.  * This software is a simplest video player based on Ffmpeg.
19.  * Suitable for beginner of Ffmpeg.
20.  *
21.  * 备注:
22.  * 标准版在播放视频的时候, 画面显示使用延时40ms的方式。这么做有两个后果:
23.  * (1) SDL弹出的窗口无法移动, 一直显示是忙碌状态
24.  * (2) 画面显示并不是严格的40ms一帧, 因为还没有考虑解码的时间。
25.  * SU (SDL Update) 版在视频解码的过程中, 不再使用延时40ms的方式, 而是创建了
26.  * 一个线程, 每隔40ms发送一个自定义的消息, 告知主函数进行解码显示。这样做之后:
27.  * (1) SDL弹出的窗口可以移动了
28.  * (2) 画面显示是严格的40ms一帧
29.  * Remark:
30.  * Standard Version use's SDL_Delay() to control video's frame rate, it has 2
31.  * disadvantages:
32.  * (1)SDL's Screen can't be moved and always "Busy".
33.  * (2)Frame rate can't be accurate because it doesn't consider the time consumed
34.  * by avcodec_decode_video2()
35.  * SU (SDL Update) Version solved 2 problems above. It create a thread to send SDL
36.  * Event every 40ms to tell the main loop to decode and show video frames.
37.  */
38.
39. #include <stdio.h>
40.
41. #define __STDC_CONSTANT_MACROS
42.
43. #ifdef _WIN32
44. //Windows
45. extern "C"
46. {
47. #include "libavcodec/avcodec.h"
48. #include "libavformat/avformat.h"
49. #include "libswscale/swscale.h"
50. #include "libavutil/imgutils.h"
51. #include "SDL2/SDL.h"

```

```

52. };
53. #else
54. //Linux...
55. #ifdef __cplusplus
56. extern "C"
57. {
58. #endif
59. #include <libavcodec/avcodec.h>
60. #include <libavformat/avformat.h>
61. #include <libswscale/swscale.h>
62. #include <libavutil/imgutils.h>
63. #include <SDL2/SDL.h>
64. #ifdef __cplusplus
65. };
66. #endif
67. #endif
68.
69. //Refresh Event
70. #define SFM_REFRESH_EVENT (SDL_USEREVENT + 1)
71.
72. #define SFM_BREAK_EVENT (SDL_USEREVENT + 2)
73.
74. int thread_exit=0;
75. int thread_pause=0;
76.
77. int sfp_refresh_thread(void *opaque){
78.     thread_exit=0;
79.     thread_pause=0;
80.
81.     while (!thread_exit) {
82.         if(!thread_pause){
83.             SDL_Event event;
84.             event.type = SFM_REFRESH_EVENT;
85.             SDL_PushEvent(&event);
86.         }
87.         SDL_Delay(40);
88.     }
89.     thread_exit=0;
90.     thread_pause=0;
91.     //Break
92.     SDL_Event event;
93.     event.type = SFM_BREAK_EVENT;
94.     SDL_PushEvent(&event);
95.
96.     return 0;
97. }
98.
99.
100. int main(int argc, char* argv[])
101. {
102.
103.     AVFormatContext *pFormatCtx;
104.     int i, videoindex;
105.     AVCodecContext *pCodecCtx;
106.     AVCodec *pCodec;
107.     AVFrame *pFrame,*pFrameYUV;
108.     unsigned char *out_buffer;
109.     AVPacket *packet;
110.     int ret, got_picture;
111.
112.     //-----SDL-----
113.     int screen_w,screen_h;
114.     SDL_Window *screen;
115.     SDL_Renderer* sdlRenderer;
116.     SDL_Texture* sdlTexture;
117.     SDL_Rect sdlRect;
118.     SDL_Thread *video_tid;
119.     SDL_Event event;
120.
121.     struct SwsContext *img_convert_ctx;
122.
123.     //char filepath[]="bigbuckbunny_480x272.h265";
124.     char filepath[]="Titanic.ts";
125.
126.     av_register_all();
127.     avformat_network_init();
128.     pFormatCtx = avformat_alloc_context();
129.
130.     if(avformat_open_input(&pFormatCtx,filepath,NULL,NULL)!=0){
131.         printf("Couldn't open input stream.\n");
132.         return -1;
133.     }
134.     if(avformat_find_stream_info(pFormatCtx,NULL)<0){
135.         printf("Couldn't find stream information.\n");
136.         return -1;
137.     }
138.     videoindex=-1;
139.     for(i=0; i<pFormatCtx->nb_streams; i++){
140.         if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
141.             videoindex=i;
142.             break;

```

```

143.     }
144.     if(videoindex==-1){
145.         printf("Didn't find a video stream.\n");
146.         return -1;
147.     }
148.     pCodecCtx=pFormatCtx->streams[videoindex]->codec;
149.     pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
150.     if(pCodec==NULL){
151.         printf("Codec not found.\n");
152.         return -1;
153.     }
154.     if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
155.         printf("Could not open codec.\n");
156.         return -1;
157.     }
158.     pFrame=av_frame_alloc();
159.     pFrameYUV=av_frame_alloc();
160.
161.     out_buffer=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height,1));
162.     av_image_fill_arrays(pFrameYUV->data, pFrameYUV->linesize,out_buffer,
163.         AV_PIX_FMT_YUV420P,pCodecCtx->width, pCodecCtx->height,1);
164.
165.     //Output Info-----
166.     printf("----- File Information ----- \n");
167.     av_dump_format(pFormatCtx,0,filepath,0);
168.     printf("----- \r");
169.
170.     img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
171.         pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
172.
173.
174.     if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
175.         printf( "Could not initialize SDL - %s\n", SDL_GetError());
176.         return -1;
177.     }
178.     //SDL 2.0 Support for multiple windows
179.     screen_w = pCodecCtx->width;
180.     screen_h = pCodecCtx->height;
181.     screen = SDL_CreateWindow("Simplest ffmpeg player's Window", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
182.         screen_w, screen_h,SDL_WINDOW_OPENGL);
183.
184.     if(!screen) {
185.         printf("SDL: could not create window - exiting:%s\n",SDL_GetError());
186.         return -1;
187.     }
188.     sdlRenderer = SDL_CreateRenderer(screen, -1, 0);
189.     //IYUV: Y + U + V (3 planes)
190.     //YV12: Y + V + U (3 planes)
191.     sdlTexture = SDL_CreateTexture(sdlRenderer, SDL_PIXELFORMAT_IYUV, SDL_TEXTUREACCESS_STREAMING,pCodecCtx->width,pCodecCtx->height
192. );
193.
194.     sdlRect.x=0;
195.     sdlRect.y=0;
196.     sdlRect.w=screen_w;
197.     sdlRect.h=screen_h;
198.
199.     packet=(AVPacket *)av_malloc(sizeof(AVPacket));
200.
201.     video_tid = SDL_CreateThread(sfp_refresh_thread,NULL,NULL);
202.     //-----SDL End-----
203.     //Event Loop
204.     for (;;) {
205.         //Wait
206.         SDL_WaitEvent(&event);
207.         if(event.type==SFM_REFRESH_EVENT){
208.             while(1){
209.                 if(av_read_frame(pFormatCtx, packet)<0)
210.                     thread_exit=1;
211.
212.                 if(packet->stream_index==videoindex)
213.                     break;
214.             }
215.             ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
216.             if(ret < 0){
217.                 printf("Decode Error.\n");
218.                 return -1;
219.             }
220.             if(got_picture){
221.                 sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height, pFrame
YUV->data, pFrameYUV->linesize);
222.                 //SDL-----
223.                 SDL_UpdateTexture( sdlTexture, NULL, pFrameYUV->data[0], pFrameYUV->linesize[0] );
224.                 SDL_RenderClear( sdlRenderer );
225.                 //SDL_RenderCopy( sdlRenderer, sdlTexture, &sdlRect, &sdlRect );
226.                 SDL_RenderCopy( sdlRenderer, sdlTexture, NULL, NULL);
227.                 SDL_RenderPresent( sdlRenderer );
228.                 //SDL End-----
229.             }
230.             av_free_packet(packet);
231.             }else if(event.type==SDL_KEYDOWN){

```

```

232.         //Pause
233.         if(event.key.keysym.sym==SDLK_SPACE)
234.             thread_pause=!thread_pause;
235.         }else if(event.type==SDL_QUIT){
236.             thread_exit=1;
237.         }else if(event.type==SFM_BREAK_EVENT){
238.             break;
239.         }
240.
241.     }
242.
243.     sws_freeContext(img_convert_ctx);
244.
245.     SDL_Quit();
246.     //-----
247.     av_frame_free(&pFrameYUV);
248.     av_frame_free(&pFrame);
249.     avcodec_close(pCodecCtx);
250.     avformat_close_input(&pFormatCtx);
251.
252.     return 0;
253. }

```

## 运行结果

程序运行后，会在命令行窗口打印一些视频信息，同时会弹出一个窗口播放视频内容。

□

## 下载

CSDN完整工程下载地址：

<http://download.csdn.net/detail/leixiaohua1020/7826277>

更新（2014.10.5）=====

版本升级至2.2。

1.新版本在原版本的基础上增加了“flush\_decoder”功能。当av\_read\_frame()循环退出的时候，实际上解码器中可能还包含剩余的几帧数据。因此需要通过“flush\_decoder”将这帧数据输出。“flush\_decoder”功能简而言之即直接调用avcodec\_decode\_video2()获得AVFrame，而不再向解码器传递AVPacket。参考代码如下：

```

[cpp]
1. //FIX: Flush Frames remained in Codec
2. while (1) {
3.     ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
4.     if (ret < 0)
5.         break;
6.     if (!got_picture)
7.         break;
8.     sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height, pFrameYUV->data, pFrameYUV->linesize);
9.     //处理...
10. }

```

2.为了更好地适应Linux等其他操作系统，做到可以跨平台，去除了VC特有的一些函数。比如“#include "stdafx.h"”，“\_tmain()”等等。

具体信息参见文章：[avcodec\\_decode\\_video2\(\)解码视频后丢帧的问题解决](#)

2.2版下载地址：<http://download.csdn.net/detail/leixiaohua1020/8002337>

更新-2.3（2015.1.03）=====

新增一个工程：[最简单的基于FFmpeg的解码器-纯净版（不包含libavformat）](#)

2.3版CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8322307>

更新-2.4（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。

```
[plain]
1.  ::VS2010 Environment
2.  call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
3.  ::include
4.  @set INCLUDE=include;%INCLUDE%
5.  ::lib
6.  @set LIB=lib;%LIB%
7.  ::compile and link
8.  cl simplest_ffmpeg_player.cpp /MD /link SDL2.lib SDL2main.lib avcodec.lib ^
9.  avformat.lib avutil.lib avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib ^
10. /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。

```
[plain]
1.  g++ simplest_ffmpeg_player.cpp -g -o simplest_ffmpeg_player.exe \
2.  -I /usr/local/include -L /usr/local/lib \
3.  -lmingw32 -lSDL2main -lSDL2 -lavformat -lavcodec -lavutil -lswscale
```

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。

```
[plain]
1.  gcc simplest_ffmpeg_player.cpp -g -o simplest_ffmpeg_player.out \
2.  -I /usr/local/include -L /usr/local/lib -lSDL2main -lSDL2 -lavformat -lavcodec -lavutil -lswscale
```

PS：相关的编译命令已经保存到了工程文件夹中

CSDN项目下载地址：<http://download.csdn.net/detail/leixiaohua1020/8443943>

SourceForge、Github等上面已经更新。

**更新-2.5 (2015.7.17) =====**

增加了下列工程：

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\_video\_play\_sdl2：使用SDL2播放YUV的例子。

simplest\_ffmpeg\_helloworld：输出FFmpeg类库的信息。

CSDN项目下载地址：<http://download.csdn.net/detail/leixiaohua1020/8924321>

SourceForge、Github等上面已经更新。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/38868499>

文章标签：[ffmpeg](#) [视频](#) [解码](#) [播放器](#)

个人分类：[FFMPEG](#) [我的开源项目](#)

所属专栏：[FFmpeg](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com