

## 原 最简单的基于FFmpeg的内存读写的例子：内存播放器

2014年10月05日 12:15:44 阅读数：24085

=====

最简单的基于FFmpeg的内存读写的例子系列文章列表：

[最简单的基于FFmpeg的内存读写的例子：内存播放器](#)

[最简单的基于FFmpeg的内存读写的例子：内存转码器](#)

=====

打算记录两个最简单的FFmpeg进行内存读写的例子。之前的所有有关FFmpeg的例子都是对文件进行操作的。例如《[100行代码实现最简单的基于FFMPEG+SDL的视频播放器](#)》播放的是一个视频的文件。而《[最简单的基于FFMPEG的转码程序](#)》也是将一个视频文件转换为另一个视频文件。《[最简单的基于FFmpeg的视频编码器\(YUV编码为H.264\)](#)》也是最后编码得到一个H.264视频文件。实际上，并不是所有视频的编码，解码都是针对文件进行处理的。有的时候需要的解码的视频数据在一段内存中。例如，通过其他系统送来的视频数据。同样，有的时候编码后的视频数据也未必必要保存成一个文件。例如，要求将编码后的视频数据送给其他的系统进行下一步的处理。以上两种情况就要求FFmpeg不仅仅是对文件进行“读，写”操作，而是要对内存进行“读，写”操作。因此打算记录的两个例子就是使用FFmpeg对内存进行读写的例子。

有关FFmpeg读写内存的例子已经在文章《[ffmpeg 从内存中读取数据\(或将数据输出到内存\)](#)》中有过叙述，但是一直没有做完整代码的工程。本文记录《最简单的基于FFmpeg内存播放器》。该例子中，首先将文件中的视频数据通过fread()读取到内存中，然后使用FFmpeg播放内存中的数据。

下篇文章计划记录的第二个例子是《最简单的基于FFmpeg内存转码器》。该例子中，首先将文件中的视频数据通过fread()读取到内存中，然后使用FFmpeg读取该数据并进行转码，接着将转码后的数据输出到另一块内存中，最后将该数据通过fwrite()写入成文件。

关于如何从内存中读取数据在这里不再详述，可以参考文章：

[ffmpeg 从内存中读取数据\(或将数据输出到内存\)](#)

### 关键点

关键点就两个：

1. 初始化自定义的AVIOContext，指定自定义的回调函数。示例代码如下：

```
[cpp]
1. //AVIOContext中的缓存
2. unsigned char *aviobuffer=(unsigned char*)av_malloc(32768);
3. AVIOContext *avio=avio_alloc_context(aviobuffer, 32768,0,NULL,read_buffer,NULL,NULL);
4. pFormatCtx->pb=avio;
5.
6. if(avformat_open_input(&pFormatCtx,NULL,NULL,NULL)!=0){
7.     printf("Couldn't open inputstream. (无法打开输入流) \n");
8.     return -1;
9. }
```

上述代码中，自定义了回调函数read\_buffer()。在使用avformat\_open\_input()打开媒体数据的时候，就可以不指定文件的URL了，即其第2个参数为NULL（因为数据不是靠文件读取，而是由read\_buffer()提供）

2. 自己写回调函数。示例代码如下：

```
[cpp]
1. //Callback
2. int read_buffer(void *opaque, uint8_t *buf, int buf_size){
3.     if(!feof(fp_open)){
4.         inttrue_size=fread(buf,1,buf_size,fp_open);
5.         return true_size;
6.     }else{
7.         return -1;
8.     }
9. }
```

当系统需要数据的时候，会自动调用该回调函数以获取数据。这个例子为了简单，直接使用fread()读取数据至内存。回调函数需要格外注意它的参数和返回值。

### 源代码

下面直接贴上程序的源代码：

```
[cpp]  
1.  /**
2.   * 最简单的基于FFmpeg的内存读写例子（内存播放器）
3.   * Simplest FFmpeg mem Player
4.   *
5.   * 雷霄骅
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.  * 本程序实现了对内存中的视频数据的播放。
12.  * 是最简单的使用FFmpeg读内存的例子。
13.  *
14.  * This software play video data in memory (not a file).
15.  * It's the simplest example to use FFmpeg to read from memory.
16.  *
17.  */
18.
19.
20. #include <stdio.h>
21.
22. #define __STDC_CONSTANT_MACROS
23.
24. #ifdef _WIN32
25. //Windows
26. extern "C"
27. {
28. #include "libavcodec/avcodec.h"
29. #include "libavformat/avformat.h"
30. #include "libswscale/swscale.h"
31. #include "SDL/SDL.h"
32. };
33. #else
34. //Linux...
35. #ifdef __cplusplus
36. extern "C"
37. {
38. #endif
39. #include <libavcodec/avcodec.h>
40. #include <libavformat/avformat.h>
41. #include <libswscale/swscale.h>
42. #include <SDL/SDL.h>
43. #ifdef __cplusplus
44. };
45. #endif
46. #endif
47.
48. //Output YUV420P
49. #define OUTPUT_YUV420P 0
50. FILE *fp_open=NULL;
51.
52. //Callback
53. int read_buffer(void *opaque, uint8_t *buf, int buf_size){
54.     if(!feof(fp_open)){
55.         int true_size=fread(buf,1,buf_size,fp_open);
56.         return true_size;
57.     }else{
58.         return -1;
59.     }
60. }
61.
62.
63.
64. int main(int argc, char* argv[])
65. {
66.
67.     AVFormatContext *pFormatCtx;
68.     int i, videoindex;
69.     AVCodecContext *pCodecCtx;
70.     AVCodec *pCodec;
71.     char filepath[]="cuc60anniversary_start.mkv";
72.
73.     av_register_all();
74.     avformat_network_init();
75.     pFormatCtx = avformat_alloc_context();
76.
77.     fp_open=fopen(filepath,"rb+");
78.     //Init AVIOContext
79.     unsigned char *aviobuffer=(unsigned char *)av_malloc(32768);
80.     AVIOContext *avio =avio_alloc_context(aviobuffer, 32768,0,NULL,read_buffer,NULL,NULL);
81.     pFormatCtx->pb=avio;
82.
83.     if(avformat_open_input(&pFormatCtx,NULL,NULL,NULL)!=0){
84.         printf("Couldn't open input stream.\n");
85.         return -1;
86.     }
87.     if(avformat_find_stream_info(pFormatCtx,NULL)<0){
```

```

88.     printf("Couldn't find stream information.\n");
89.     return -1;
90. }
91. videoindex=-1;
92. for(i=0; i<pFormatCtx->nb_streams; i++)
93.     if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
94.         videoindex=i;
95.         break;
96.     }
97. if(videoindex==-1){
98.     printf("Didn't find a video stream.\n");
99.     return -1;
100. }
101. pCodecCtx=pFormatCtx->streams[videoindex]->codec;
102. pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
103. if(pCodec==NULL){
104.     printf("Codec not found.\n");
105.     return -1;
106. }
107. if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
108.     printf("Could not open codec.\n");
109.     return -1;
110. }
111. AVFrame *pFrame,*pFrameYUV;
112. pFrame=av_frame_alloc();
113. pFrameYUV=av_frame_alloc();
114. //uint8_t *out_buffer=(uint8_t *)av_malloc(avpicture_get_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height));
115. //avpicture_fill((AVPicture *)pFrameYUV, out_buffer, AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height);
116. //SDL-----
117. if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
118.     printf("Could not initialize SDL - %s\n", SDL_GetError());
119.     return -1;
120. }
121.
122. int screen_w=0,screen_h=0;
123. SDL_Surface *screen;
124. screen_w = pCodecCtx->width;
125. screen_h = pCodecCtx->height;
126. screen = SDL_SetVideoMode(screen_w, screen_h, 0,0);
127.
128. if(!screen) {
129.     printf("SDL: could not set video mode - exiting:%s\n",SDL_GetError());
130.     return -1;
131. }
132. SDL_Overlay *bmp;
133. bmp = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height,SDL_YV12_OVERLAY, screen);
134. SDL_Rect rect;
135. rect.x = 0;
136. rect.y = 0;
137. rect.w = screen_w;
138. rect.h = screen_h;
139. //SDL End-----
140. int ret, got_picture;
141.
142. AVPacket *packet=(AVPacket *)av_malloc(sizeof(AVPacket));
143.
144. #if OUTPUT_YUV420P
145. FILE *fp_yuv=fopen("output.yuv","wb+");
146. #endif
147. SDL_WM_SetCaption("Simplest FFmpeg Mem Player",NULL);
148.
149. struct SwsContext *img_convert_ctx;
150. img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height, A
V_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
151. //-----
152. while(av_read_frame(pFormatCtx, packet)>=0){
153.     if(packet->stream_index==videoindex){
154.         ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
155.         if(ret < 0){
156.             printf("Decode Error.\n");
157.             return -1;
158.         }
159.         if(got_picture){
160.             SDL_LockYUVOverlay(bmp);
161.             pFrameYUV->data[0]=bmp->pixels[0];
162.             pFrameYUV->data[1]=bmp->pixels[2];
163.             pFrameYUV->data[2]=bmp->pixels[1];
164.             pFrameYUV->linesize[0]=bmp->pitches[0];
165.             pFrameYUV->linesize[1]=bmp->pitches[2];
166.             pFrameYUV->linesize[2]=bmp->pitches[1];
167.             sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height, pFrameYUV->d
ata, pFrameYUV->linesize);
168. #if OUTPUT_YUV420P
169.             int y_size=pCodecCtx->width*pCodecCtx->height;
170.             fwrite(pFrameYUV->data[0],1,y_size,fp_yuv); //Y
171.             fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv); //U
172.             fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv); //V
173. #endif
174.             SDL_UnlockYUVOverlay(bmp);
175.
176.             SDL_DisplayYUVOverlay(bmp, &rect);

```

```

177.         //Delay 40ms
178.         SDL_Delay(40);
179.     }
180. }
181.     av_free_packet(packet);
182. }
183.     sws_freeContext(img_convert_ctx);
184.
185. #if OUTPUT_YUV420P
186.     fclose(fp_yuv);
187. #endif
188.
189.     fclose(fp_open);
190.
191.     SDL_Quit();
192.
193.     //av_free(out_buffer);
194.     av_free(pFrameYUV);
195.     avcodec_close(pCodecCtx);
196.     avformat_close_input(&pFormatCtx);
197.
198.     return 0;
199. }

```

可以通过代码定义的宏来确定是否将解码后的YUV420P数据输出成文件：

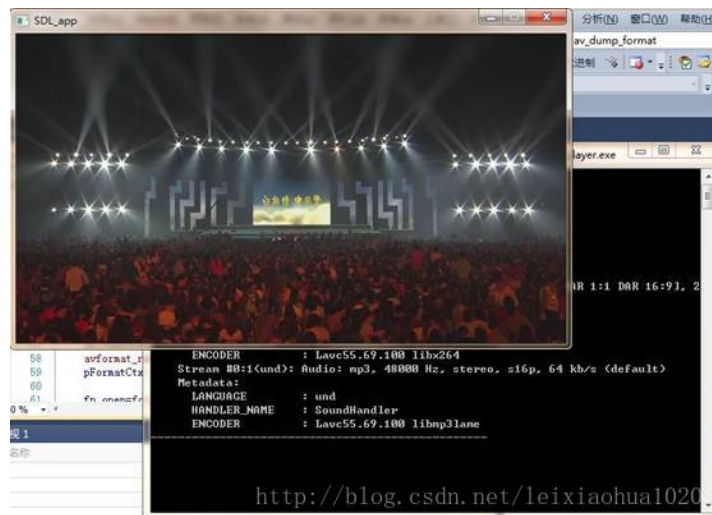
```

1. #define OUTPUT_YUV420P 0

```

## 结果

程序的运行结果如下。可以解码播放测试视频。适逢60周年校庆，因此截取了一小段校庆晚会的开场画面作为测试视频，给母校庆生~



## 下载

simplest ffmpeg mem handler

### 项目主页

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

Github：[https://github.com/leixiaohua1020/simplest\\_ffmpeg\\_mem\\_handler](https://github.com/leixiaohua1020/simplest_ffmpeg_mem_handler)

开源中国：[http://git.oschina.net/leixiaohua1020/simplest\\_ffmpeg\\_mem\\_handler](http://git.oschina.net/leixiaohua1020/simplest_ffmpeg_mem_handler)

CSDN下载地址：

<http://download.csdn.net/detail/leixiaohua1020/8003731>

本工程包含两个FFmpeg读写内存的例子：

simplest\_ffmpeg\_mem\_player：基于FFmpeg的内存播放器。

simplest\_ffmpeg\_mem\_transcoder：基于FFmpeg的内存转码器（下篇文章记录）。

## 更新-1.1 (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。

```
[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_mem_player.cpp /MD /link SDL.lib SDLmain.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_mem_player.cpp -g -o simplest_ffmpeg_mem_player.exe \
2.  -I /usr/local/include -L /usr/local/lib \
3.  -lmingw32 -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lswscale
```

GCC(Linux)：Linux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_mem_player.cpp -g -o simplest_ffmpeg_mem_player.out -lstdc++ \
2.  -I /usr/local/include -L /usr/local/lib -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lswscale
```

GCC(MacOS)：Mac终端下运行compile\_gcc\_mac.sh即可使用Mac 的GCC进行编译，Mac的GCC和Linux的GCC差别不大，但是使用SDL1.2的时候，必须加上“-framework Cocoa”参数，否则编译无法通过。编译命令如下。

```
[plain]
1.  gcc simplest_ffmpeg_mem_player.cpp -g -o simplest_ffmpeg_mem_player.out -lstdc++ \
2.  -framework Cocoa -I /usr/local/include -L /usr/local/lib -lSDLmain -lSDL -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下载地址：<http://download.csdn.net/detail/leixiaohua1020/8445795>

SourceForge上已经更新。

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/39759163>

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

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