

最简单的基于FFmpeg的AVDevice例子（屏幕录制）

2014年10月02日 13:19:23 阅读量：49685

最简单的基于FFmpeg的AVDevice例子文章列表：

[最简单的基于FFmpeg的AVDevice例子（读取摄像头）](#)

[最简单的基于FFmpeg的AVDevice例子（屏幕录制）](#)

FFmpeg中有一个和多媒体设备交互的类库：Libavdevice。使用这个库可以读取电脑的多媒体设备的数据，或者输出数据到指定的多媒体设备上。计划写2个有关FFmpeg的libavdevice类库的例子。上篇文章记录了一个基于FFmpeg的Libavdevice类库读取摄像头数据的例子。本篇文章记录一个基于FFmpeg的Libavdevice类库录制屏幕的例子。本文程序录制当前桌面内容并且解码显示出来。有关解码显示方面的代码本文不再详述，可以参考文章：

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

抓屏方法

上篇文章记录了libavdevice的使用方法，本文不再重复。在Windows系统使用libavdevice抓取屏幕数据有两种方法：gdigrab和dshow。下文分别介绍。

1. gdigrab

gdigrab是FFmpeg专门用于抓取Windows桌面的设备。非常适合用于屏幕录制。它通过不同的输入URL支持两种方式的抓取：

- （1）“desktop”：抓取整张桌面。或者抓取桌面中的一个特定的区域。
- （2）“title={窗口名称}”：抓取屏幕中特定的一个窗口（目前中文窗口还有乱码问题）。

gdigrab另外还支持一些参数，用于设定抓屏的位置：

offset_x：抓屏起始点横坐标。

offset_y：抓屏起始点纵坐标。

video_size：抓屏的大小。

framerate：抓屏的帧率。

参考的代码如下：

```
[cpp]
1. //Use gdigrab
2. AVDictionary* options = NULL;
3. //Set some options
4. //grabbing frame rate
5. //av_dict_set(&options,"framerate","5",0);
6. //The distance from the left edge of the screen or desktop
7. //av_dict_set(&options,"offset_x","20",0);
8. //The distance from the top edge of the screen or desktop
9. //av_dict_set(&options,"offset_y","40",0);
10. //Video frame size. The default is to capture the full screen
11. //av_dict_set(&options,"video_size","640x480",0);
12. AVInputFormat *ifmt=av_find_input_format("gdigrab");
13. if(avformat_open_input(&formatCtx,"desktop",ifmt,&options)!=0){
14.     printf("Couldn't open input stream. (无法打开输入流) \n");
15.     return -1;
16. }
```

2. dshow

使用dshow抓屏需要安装抓屏软件：screen-capture-recorder

软件地址：<http://sourceforge.net/projects/screencapturer/>

下载软件安装完成后，可以指定dshow的输入设备为“screen-capture-recorder”即可。有关dshow设备的使用方法在上一篇文章中已经有详细叙述，这里不再重复。参考的代码如下：

```
[cpp]
1. AVInputFormat *ifmt=av_find_input_format("dshow");
2. if(avformat_open_input(&formatCtx,"video=screen-capture-recorder",ifmt,NULL)!=0){
3.     printf("Couldn't open input stream. (无法打开输入流) \n");
4.     return -1;
5. }
```


注：上述两种抓屏方法也可以直接使用ffmpeg.exe的命令行完成，可以参考文章：

[FFmpeg获取DirectShow设备数据（摄像头，录屏）](#)

在Linux下可以使用x11grab抓屏，在MacOS下可以使用avfoundation抓屏，在这里不再详细叙述。

代码

下面直接贴上程序代码：

```
[cpp]    
1.  /**  
2.  * 最简单的基于FFmpeg的AVDevice例子（屏幕录制）  
3.  * Simplest FFmpeg Device (Screen Capture)  
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. * 本程序实现了屏幕录制功能。可以录制并播放桌面数据。是基于FFmpeg  
12. * 的libavdevice类库最简单的例子。通过该例子，可以学习FFmpeg中  
13. * libavdevice类库的使用方法。  
14. * 本程序在Windows下可以使用2种方式录制屏幕：  
15. * 1.gdigrab: Win32下的基于GDI的屏幕录制设备。  
16. * 抓取桌面的时候，输入URL为“desktop”。  
17. * 2.dshow: 使用Directshow。注意需要安装额外的软件screen-capture-recorder  
18. * 在Linux下可以使用x11grab录制屏幕。  
19. * 在MacOS下可以使用avfoundation录制屏幕。  
20. *  
21. * This software capture screen of computer. It's the simplest example  
22. * about usage of FFmpeg's libavdevice Library.  
23. * It's suitable for the beginner of FFmpeg.  
24. * This software support 2 methods to capture screen in Microsoft Windows:  
25. * 1.gdigrab: Win32 GDI-based screen capture device.  
26. * Input URL in avformat_open_input() is "desktop".  
27. * 2.dshow: Use Directshow. Need to install screen-capture-recorder.  
28. * It use x11grab to capture screen in Linux.  
29. * It use avfoundation to capture screen in MacOS.  
30. */  
31.  
32.  
33. #include <stdio.h>  
34.  
35. #define __STDC_CONSTANT_MACROS  
36.  
37. #ifdef _WIN32  
38. //Windows  
39. extern "C"  
40. {  
41. #include "libavcodec/avcodec.h"  
42. #include "libavformat/avformat.h"  
43. #include "libswscale/swscale.h"  
44. #include "libavdevice/avdevice.h"  
45. #include "SDL/SDL.h"  
46. };  
47. #else  
48. //Linux...  
49. #ifdef __cplusplus  
50. extern "C"  
51. {  
52. #endif  
53. #include <libavcodec/avcodec.h>  
54. #include <libavformat/avformat.h>  
55. #include <libswscale/swscale.h>  
56. #include <libavdevice/avdevice.h>  
57. #include <SDL/SDL.h>  
58. #ifdef __cplusplus  
59. };  
60. #endif  
61. #endif  
62.  
63. //Output YUV420P  
64. #define OUTPUT_YUV420P 0  
65. //'1' Use Dshow  
66. //'0' Use GDIGrab  
67. #define USE_DSHOW 0  
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.  
76. int sfp_refresh_thread(void *opaque)  
77. {  
78.     thread_exit=0;  
79.     while (!thread_exit) {  
80.         SDL_Event event;  
81.         event.type = SFM_REFRESH_EVENT;  
82.         SDL_PushEvent(&event);  
83.         SDL_Delay(40);  
84.     }  
85.     thread_exit=0;
```

```

85.         linebuf_exit=0;
86.         //Break
87.         SDL_Event event;
88.         event.type = SFM_BREAK_EVENT;
89.         SDL_PushEvent(&event);
90.
91.         return 0;
92.     }
93.
94.     //Show Dshow Device
95.     void show_dshow_device(){
96.         AVFormatContext *pFormatCtx = avformat_alloc_context();
97.         AVDictionary* options = NULL;
98.         av_dict_set(&options,"list_devices","true",0);
99.         AVInputFormat *iformat = av_find_input_format("dshow");
100.        printf("=====Device Info=====\n");
101.        avformat_open_input(&pFormatCtx,"video=dummy",iformat,&options);
102.        printf("=====\n");
103.    }
104.
105.    //Show AVFoundation Device
106.    void show_avfoundation_device(){
107.        AVFormatContext *pFormatCtx = avformat_alloc_context();
108.        AVDictionary* options = NULL;
109.        av_dict_set(&options,"list_devices","true",0);
110.        AVInputFormat *iformat = av_find_input_format("avfoundation");
111.        printf("==AVFoundation Device Info==\n");
112.        avformat_open_input(&pFormatCtx,"",iformat,&options);
113.        printf("=====\n");
114.    }
115.
116.
117.
118.    int main(int argc, char* argv[])
119.    {
120.
121.        AVFormatContext *pFormatCtx;
122.        int i, videoindex;
123.        AVCodecContext *pCodecCtx;
124.        AVCodec *pCodec;
125.
126.        av_register_all();
127.        avformat_network_init();
128.        pFormatCtx = avformat_alloc_context();
129.
130.        //Open File
131.        //char filepath[]="src01_480x272_22.h265";
132.        //avformat_open_input(&pFormatCtx,filepath,NULL,NULL)
133.
134.        //Register Device
135.        avdevice_register_all();
136.        //Windows
137.    #ifdef _WIN32
138.    #if USE_DSHOW
139.        //Use dshow
140.        //
141.        //Need to Install screen-capture-recorder
142.        //screen-capture-recorder
143.        //Website: http://sourceforge.net/projects/screencapturer/
144.        //
145.        AVInputFormat *ifmt=av_find_input_format("dshow");
146.        if(avformat_open_input(&pFormatCtx,"video=screen-capture-recorder",ifmt,NULL)!=0){
147.            printf("Couldn't open input stream.\n");
148.            return -1;
149.        }
150.    #else
151.        //Use gdigrab
152.        AVDictionary* options = NULL;
153.        //Set some options
154.        //grabbing frame rate
155.        //av_dict_set(&options,"framerate","5",0);
156.        //The distance from the left edge of the screen or desktop
157.        //av_dict_set(&options,"offset_x","20",0);
158.        //The distance from the top edge of the screen or desktop
159.        //av_dict_set(&options,"offset_y","40",0);
160.        //Video frame size. The default is to capture the full screen
161.        //av_dict_set(&options,"video_size","640x480",0);
162.        AVInputFormat *ifmt=av_find_input_format("gdigrab");
163.        if(avformat_open_input(&pFormatCtx,"desktop",ifmt,&options)!=0){
164.            printf("Couldn't open input stream.\n");
165.            return -1;
166.        }
167.    #endif
168.
169.    #elif defined linux
170.        //Linux
171.        AVDictionary* options = NULL;
172.        //Set some options
173.        //grabbing frame rate
174.        //av_dict_set(&options,"framerate","5",0);
175.        //Make the grabbed area follow the mouse
176.        //av_dict_set(&options,"follow mouse","centered",0);

```

```

177. //Video frame size. The default is to capture the full screen
178. //av_dict_set(&options,"video_size","640x480",0);
179. AVInputFormat *ifmt=av_find_input_format("x11grab");
180. //Grab at position 10,20
181. if(avformat_open_input(&pFormatCtx,":0.0+10,20",ifmt,&options)!=0){
182.     printf("Couldn't open input stream.\n");
183.     return -1;
184. }
185. #else
186. show_avfoundation_device();
187. //Mac
188. AVInputFormat *ifmt=av_find_input_format("avfoundation");
189. //Avfoundation
190. //[[video]:[audio]
191. if(avformat_open_input(&pFormatCtx,"1",ifmt,NULL)!=0){
192.     printf("Couldn't open input stream.\n");
193.     return -1;
194. }
195. #endif
196.
197. if(avformat_find_stream_info(pFormatCtx,NULL)<0)
198. {
199.     printf("Couldn't find stream information.\n");
200.     return -1;
201. }
202. videoindex=-1;
203. for(i=0; i<pFormatCtx->nb_streams; i++)
204.     if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO)
205.     {
206.         videoindex=i;
207.         break;
208.     }
209. if(videoindex==-1)
210. {
211.     printf("Didn't find a video stream.\n");
212.     return -1;
213. }
214. pCodecCtx=pFormatCtx->streams[videoindex]->codec;
215. pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
216. if(pCodec==NULL)
217. {
218.     printf("Codec not found.\n");
219.     return -1;
220. }
221. if(avcodec_open2(pCodecCtx, pCodec,NULL)<0)
222. {
223.     printf("Could not open codec.\n");
224.     return -1;
225. }
226. AVFrame *pFrame,*pFrameYUV;
227. pFrame=av_frame_alloc();
228. pFrameYUV=av_frame_alloc();
229. //unsigned char *out_buffer=(unsigned char *)av_malloc(avpicture_get_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height
230. t));
231. //avpicture_fill((AVPicture *)pFrameYUV, out_buffer, AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height);
232. //SDL-----
233. if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
234.     printf("Could not initialize SDL - %s\n", SDL_GetError());
235.     return -1;
236. }
237. int screen_w=640,screen_h=360;
238. const SDL_VideoInfo *vi = SDL_GetVideoInfo();
239. //Half of the Desktop's width and height.
240. screen_w = vi->current_w/2;
241. screen_h = vi->current_h/2;
242. SDL_Surface *screen;
243. screen = SDL_SetVideoMode(screen_w, screen_h, 0,0);
244.
245. if(!screen) {
246.     printf("SDL: could not set video mode - exiting:%s\n",SDL_GetError());
247.     return -1;
248. }
249. SDL_Overlay *bmp;
250. bmp = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height,SDL_YV12_OVERLAY, screen);
251. SDL_Rect rect;
252. rect.x = 0;
253. rect.y = 0;
254. rect.w = screen_w;
255. rect.h = screen_h;
256. //SDL End-----
257. int ret, got_picture;
258.
259. AVPacket *packet=(AVPacket *)av_malloc(sizeof(AVPacket));
260.
261. #if OUTPUT_YUV420P
262. FILE *fp_yuv=fopen("output.yuv","wb+");
263. #endif
264.
265. struct SwsContext *img_convert_ctx;
266. 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);

```

```

266. //-----
267. SDL_Thread *video_tid = SDL_CreateThread(sfp_refresh_thread,NULL);
268. //
269. SDL_WM_SetCaption("Simplest Ffmpeg Grab Desktop",NULL);
270. //Event Loop
271. SDL_Event event;
272.
273. for (;;) {
274.     //Wait
275.     SDL_WaitEvent(&event);
276.     if(event.type==SFM_REFRESH_EVENT){
277.         //-----
278.         if(av_read_frame(pFormatCtx, packet)>=0){
279.             if(packet->stream_index==videoindex){
280.                 ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
281.                 if(ret < 0){
282.                     printf("Decode Error.\n");
283.                     return -1;
284.                 }
285.                 if(got_picture){
286.                     SDL_LockYUVOverlay(bmp);
287.                     pFrameYUV->data[0]=bmp->pixels[0];
288.                     pFrameYUV->data[1]=bmp->pixels[2];
289.                     pFrameYUV->data[2]=bmp->pixels[1];
290.                     pFrameYUV->linesize[0]=bmp->pitches[0];
291.                     pFrameYUV->linesize[1]=bmp->pitches[2];
292.                     pFrameYUV->linesize[2]=bmp->pitches[1];
293.                     sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height
, pFrameYUV->data, pFrameYUV->linesize);
294.
295. #if OUTPUT_YUV420P
296.                     int y_size=pCodecCtx->width*pCodecCtx->height;
297.                     fwrite(pFrameYUV->data[0],1,y_size,fp_yuv); //Y
298.                     fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv); //U
299.                     fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv); //V
300. #endif
301.                     SDL_UnlockYUVOverlay(bmp);
302.
303.                     SDL_DisplayYUVOverlay(bmp, &rect);
304.
305.                 }
306.             }
307.             av_free_packet(packet);
308.         }else{
309.             //Exit Thread
310.             thread_exit=1;
311.         }
312.     }else if(event.type==SDL_QUIT){
313.         thread_exit=1;
314.     }else if(event.type==SFM_BREAK_EVENT){
315.         break;
316.     }
317. }
318.
319.
320.
321. sws_freeContext(img_convert_ctx);
322.
323. #if OUTPUT_YUV420P
324.     fclose(fp_yuv);
325. #endif
326.
327. SDL_Quit();
328.
329. //av_free(out_buffer);
330. av_free(pFrameYUV);
331. avcodec_close(pCodecCtx);
332. avformat_close_input(&pFormatCtx);
333.
334. return 0;
335. }

```

结果

程序的运行效果如下。这个运行结果还是十分有趣的，会出现一个屏幕“嵌套”在另一个屏幕里面的现象，环环相套。

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：

[cpp]  

```
1. #define OUTPUT_YUV420P 0
```

可以通过下面的宏定义来确定使用GDIGrab或者是Dshow打开摄像头：

```
[cpp]
1.  // '1' Use Dshow
2.  // '0' Use GDIgrab
3.  #define USE_DSHOW 0
```

下载

Simplest Ffmpeg Device

项目主页

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

Github：https://github.com/leixiaohua1020/simplest_ffmpeg_device

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

CSDN下载地址：

<http://download.csdn.net/detail/leixiaohua1020/7994049>

注：

本工程包含两个基于Ffmpeg的libavdevice的例子：

simplest_ffmpeg_grabdesktop：屏幕录制。

simplest_ffmpeg_readcamera：读取摄像头

更新-1.1 (2015.1.9) =====

该版本中，修改了SDL的显示方式，弹出的窗口可以移动了。

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8344695>

更新-1.2 (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_grabdesktop.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_grabdesktop.cpp -g -o simplest_ffmpeg_grabdesktop.exe \
2.  -I /usr/local/include -L /usr/local/lib \
3.  -lmingw32 -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lavdevice -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_grabdesktop.cpp -g -o simplest_ffmpeg_grabdesktop.out \
2.  -I /usr/local/include -L /usr/local/lib -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lavdevice -lswscale
```

GCC(MacOS) : MacOS命令行下运行`compile_gcc_mac.sh`即可使用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_grabdesktop.cpp -g -o simplest_ffmpeg_grabdesktop.out \  
2. -framework Cocoa -I /usr/local/include -L /usr/local/lib -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lavdevice -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/leixiaohua1020/8445747>

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/39706721>

文章标签：[ffmpeg](#) [屏幕录制](#) [libavdevice](#) [VC](#) [directshow](#)

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