

原 最简单的基于FFmpeg的AVDevice例子（读取摄像头）

2014年10月01日 00:09:19 阅读数：60100

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（或者其他设备上）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。

Libavdevice支持以下设备作为输入端：

- alsa
- avfoundation
- bktr
- dshow
- dv1394
- fbdev
- gdigrab
- iec61883
- jack
- lavfi
- libcdio
- libdc1394
- openal
- oss
- pulse
- qtkit
- sndio
- video4linux2, v4l2
- vfwcap
- x11grab
- decklink

Libavdevice支持以下设备作为输出端：

- alsa
- caca
- decklink
- fbdev
- opengl
- oss
- pulse
- sdl
- sndio
- xv

libavdevice使用

计划记录两个基于FFmpeg的libavdevice类库的例子，分成两篇文章写。本文记录一个基于FFmpeg的Libavdevice类库读取摄像头数据的例子。下一篇文章记录一个基于FFmpeg的Libavdevice类库录制屏幕的例子。本文程序读取计算机上的摄像头的数据并且解码显示出来。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）](#)》

本文主要记录使用libavdevice需要注意的步骤。

首先，使用libavdevice的时候需要包含其头文件：

```
1. #include "libavdevice/avdevice.h"
```

然后，在程序中需要注册libavdevice：

```
1. avdevice_register_all();
```

接下来就可以使用libavdevice的功能了。

使用libavdevice读取数据和直接打开视频文件比较类似。因为系统的设备也被FFmpeg认为是一种输入的格式（即AVInputFormat）。使用FFmpeg打开一个普通的视频文件使用如下函数：

```
1. AVFormatContext *pFormatCtx = avformat_alloc_context();
2. avformat_open_input(&pFormatCtx, "test.h265", NULL, NULL);
```

使用libavdevice的时候，唯一的不同在于需要首先查找用于输入的设备。在这里使用av_find_input_format()完成：

```
1. AVFormatContext *pFormatCtx = avformat_alloc_context();
2. AVInputFormat *ifmt=av_find_input_format("vfwcap");
3. avformat_open_input(&pFormatCtx, 0, ifmt, NULL);
```

上述代码首先指定了vfw设备作为输入设备，然后在URL中指定打开第0个设备（在我自己计算机上即是摄像头设备）。

在Windows平台上除了使用vfw设备作为输入设备之外，还可以使用DirectShow作为输入设备：

```
1. AVFormatContext *pFormatCtx = avformat_alloc_context();
2. AVInputFormat *ifmt=av_find_input_format("dshow");
3. avformat_open_input(&pFormatCtx, "video=Integrated Camera", ifmt, NULL);
```

使用ffmpeg.exe打开vfw设备和Directshow设备的方法可以参考文章：

[FFmpeg获取DirectShow设备数据（摄像头，录屏）](#)

注意事项

1. URL的格式是"video={设备名称}"，但是设备名称外面不能加引号。例如在上述例子中URL是"video=Integrated Camera"，而不能写成"video=\"Integrated Camera\""，否则就无法打开设备。这与直接使用ffmpeg.exe打开dshow设备（命令为：ffmpeg -list_options true -f dshow -i video="Integrated Camera"）有很大的不同。

2. Dshow的设备名称必须要提前获取，在这里有两种方法：

(1)

通过FFmpeg编程实现。使用如下代码：

```
1. //Show Device
2. void show_dshow_device(){
3.     AVFormatContext *pFormatCtx = avformat_alloc_context();
4.     AVDictionary* options = NULL;
5.     av_dict_set(&options, "list_devices", "true", 0);
6.     AVInputFormat *iformat = av_find_input_format("dshow");
7.     printf("Device Info=====\n");
8.     avformat_open_input(&pFormatCtx, "video=dummy", iformat, &options);
9.     printf("=====\n");
10. }
```

上述代码实际上相当于输入了下面一条命令：

```
1. ffmpeg -list_devices true -f dshow -i dummy
```

执行的结果如下图所示：

```
ffmpeg @ 0388f5e0] DirectShow video devices
ffmpeg @ 0388f5e0] "Integrated Camera"
ffmpeg @ 0388f5e0] "screen-capture-recorder"
ffmpeg @ 0388f5e0] DirectShow audio devices
ffmpeg @ 0388f5e0] "内置麦克风 (Conexant 20672 SmartAudio)"
ffmpeg @ 0388f5e0] "virtual-audio-capturer"
```

该方法好处是可以使用程序自动获取名称。但是当设备名称中包含中文字符的时候，会出现设备名称为乱码的情况。如果直接把乱码的设备名作为输入的话，是无法打开该设备的。这时候需要把乱码ANSI转换为UTF-8。例如上图中的第一个音频设备显示为“内置麦克风 (Conexant 20672 SmartAudio)”，转码之后即为“内置麦克风 (Conexant 20672 SmartAudio)”。使用转码之后的名称即可打开该设备。


```

15.  * 1.VFW: Video for Windows 屏幕捕捉设备。注意输入URL是设备的序号，
16.  * 从0至9。
17.  * 2.dshow: 使用Directshow。注意作者机器上的摄像头设备名称是
18.  * "Integrated Camera"，使用的时候需要改成自己电脑上摄像头设
19.  * 备的名称。
20.  * 在Linux下可以使用video4linux2读取摄像头设备。
21.  * 在MacOS下可以使用avfoundation读取摄像头设备。
22.  *
23.  * This software read data from Computer's Camera and play it.
24.  * It's the simplest example about usage of FFmpeg's libavdevice Library.
25.  * It's suitable for the beginner of FFmpeg.
26.  * This software support 2 methods to read camera in Microsoft Windows:
27.  * 1.gdigrab: Vfw (Video for Windows) capture input device.
28.  * The filename passed as input is the capture driver number,
29.  * ranging from 0 to 9.
30.  * 2.dshow: Use Directshow. Camera's name in author's computer is
31.  * "Integrated Camera".
32.  * It use video4linux2 to read Camera in Linux.
33.  * It use avfoundation to read Camera in MacOS.
34.  *
35.  */
36.
37.
38. #include <stdio.h>
39.
40. #define __STDC_CONSTANT_MACROS
41.
42. #ifdef _WIN32
43. //Windows
44. extern "C"
45. {
46. #include "libavcodec/avcodec.h"
47. #include "libavformat/avformat.h"
48. #include "libswscale/swscale.h"
49. #include "libavdevice/avdevice.h"
50. #include "SDL/SDL.h"
51. };
52. #else
53. //Linux...
54. #ifdef __cplusplus
55. extern "C"
56. {
57. #endif
58. #include <libavcodec/avcodec.h>
59. #include <libavformat/avformat.h>
60. #include <libswscale/swscale.h>
61. #include <libavdevice/avdevice.h>
62. #include <SDL/SDL.h>
63. #ifdef __cplusplus
64. };
65. #endif
66. #endif
67.
68. //Output YUV420P
69. #define OUTPUT_YUV420P 0
70. // '1' Use Dshow
71. // '0' Use VFW
72. #define USE_DSHOW 0
73.
74.
75. //Refresh Event
76. #define SFM_REFRESH_EVENT (SDL_USEREVENT + 1)
77.
78. #define SFM_BREAK_EVENT (SDL_USEREVENT + 2)
79.
80. int thread_exit=0;
81.
82. int sfp_refresh_thread(void *opaque)
83. {
84.     thread_exit=0;
85.     while (!thread_exit) {
86.         SDL_Event event;
87.         event.type = SFM_REFRESH_EVENT;
88.         SDL_PushEvent(&event);
89.         SDL_Delay(40);
90.     }
91.     thread_exit=0;
92.     //Break
93.     SDL_Event event;
94.     event.type = SFM_BREAK_EVENT;
95.     SDL_PushEvent(&event);
96.
97.     return 0;
98. }
99.
100.
101. //Show Dshow Device
102. void show_dshow_device(){
103.     AVFormatContext *pFormatCtx = avformat_alloc_context();
104.     AVDictionary* options = NULL;
105.     av_dict_set(&options, "list_devices", "true", 0);
106.     AVInputFormat *iformat = av_find_input_format("dshow");

```

```

100.     AVInputFormat *iformat = av_find_input_format( "dshow" );
101.     printf("=====Device Info=====\n");
102.     avformat_open_input(&pFormatCtx, "video=dummy", iformat, &options);
103.     printf("=====\n");
104. }
105.
106. //Show Dshow Device Option
107. void show_dshow_device_option(){
108.     AVFormatContext *pFormatCtx = avformat_alloc_context();
109.     AVDictionary* options = NULL;
110.     av_dict_set(&options, "list_options", "true", 0);
111.     AVInputFormat *iformat = av_find_input_format("dshow");
112.     printf("=====Device Option Info=====\n");
113.     avformat_open_input(&pFormatCtx, "video=Integrated Camera", iformat, &options);
114.     printf("=====\n");
115. }
116.
117. //Show VFW Device
118. void show_vfw_device(){
119.     AVFormatContext *pFormatCtx = avformat_alloc_context();
120.     AVInputFormat *iformat = av_find_input_format("vfwcap");
121.     printf("=====VFW Device Info=====\n");
122.     avformat_open_input(&pFormatCtx, "list", iformat, NULL);
123.     printf("=====\n");
124. }
125.
126. //Show AVFoundation Device
127. void show_avfoundation_device(){
128.     AVFormatContext *pFormatCtx = avformat_alloc_context();
129.     AVDictionary* options = NULL;
130.     av_dict_set(&options, "list_devices", "true", 0);
131.     AVInputFormat *iformat = av_find_input_format("avfoundation");
132.     printf("==AVFoundation Device Info==\n");
133.     avformat_open_input(&pFormatCtx, "", iformat, &options);
134.     printf("=====\n");
135. }
136.
137. int main(int argc, char* argv[])
138. {
139.     AVFormatContext *pFormatCtx;
140.     int i, videoindex;
141.     AVCodecContext *pCodecCtx;
142.     AVCodec *pCodec;
143.
144.     av_register_all();
145.     avformat_network_init();
146.     pFormatCtx = avformat_alloc_context();
147.
148.     //Open File
149.     //char filepath[]="src01_480x272_22.h265";
150.     //avformat_open_input(&pFormatCtx, filepath, NULL, NULL)
151.
152.     //Register Device
153.     avdevice_register_all();
154.
155.     //Windows
156. #ifdef _WIN32
157.
158.     //Show Dshow Device
159.     show_dshow_device();
160.     //Show Device Options
161.     show_dshow_device_option();
162.     //Show VFW Options
163.     show_vfw_device();
164.
165. #if USE_DSHOW
166.     AVInputFormat *ifmt=av_find_input_format("dshow");
167.     //Set own video device's name
168.     if(avformat_open_input(&pFormatCtx, "video=Integrated Camera", ifmt, NULL)!=0){
169.         printf("Couldn't open input stream.\n");
170.         return -1;
171.     }
172. #else
173.     AVInputFormat *ifmt=av_find_input_format("vfwcap");
174.     if(avformat_open_input(&pFormatCtx, "0", ifmt, NULL)!=0){
175.         printf("Couldn't open input stream.\n");
176.         return -1;
177.     }
178. #endif
179. #elif defined linux
180.     //Linux
181.     AVInputFormat *ifmt=av_find_input_format("video4linux2");
182.     if(avformat_open_input(&pFormatCtx, "/dev/video0", ifmt, NULL)!=0){
183.         printf("Couldn't open input stream.\n");
184.         return -1;
185.     }
186. #else
187.     show_avfoundation_device();
188.     //Mac
189.     AVInputFormat *ifmt=av_find_input_format("avfoundation");

```

```

207. //Avfoundation
208. //[[video]:[audio]]
209. if(avformat_open_input(&pFormatCtx,"0",ifmt,NULL)!=0){
210.     printf("Couldn't open input stream.\n");
211.     return -1;
212. }
213. #endif
214.
215. if(avformat_find_stream_info(pFormatCtx,NULL)<0)
216. {
217.     printf("Couldn't find stream information.\n");
218.     return -1;
219. }
220. videoindex=-1;
221. for(i=0; i<pFormatCtx->nb_streams; i++)
222.     if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO)
223.     {
224.         videoindex=i;
225.         break;
226.     }
227. if(videoindex==-1)
228. {
229.     printf("Couldn't find a video stream.\n");
230.     return -1;
231. }
232. pCodecCtx=pFormatCtx->streams[videoindex]->codec;
233. pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
234. if(pCodec==NULL)
235. {
236.     printf("Codec not found.\n");
237.     return -1;
238. }
239. if(avcodec_open2(pCodecCtx, pCodec,NULL)<0)
240. {
241.     printf("Could not open codec.\n");
242.     return -1;
243. }
244. AVFrame *pFrame,*pFrameYUV;
245. pFrame=av_frame_alloc();
246. pFrameYUV=av_frame_alloc();
247. //unsigned char *out_buffer=(unsigned char *)av_malloc(avpicture_get_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height
248. t));
249. //avpicture_fill((AVPicture *)pFrameYUV, out_buffer, AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height);
250. //SDL-----
251. if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
252.     printf("Could not initialize SDL - %s\n", SDL_GetError());
253.     return -1;
254. }
255. int screen_w=0,screen_h=0;
256. SDL_Surface *screen;
257. screen_w = pCodecCtx->width;
258. screen_h = pCodecCtx->height;
259. screen = SDL_SetVideoMode(screen_w, screen_h, 0,0);
260.
261. if(!screen) {
262.     printf("SDL: could not set video mode - exiting:%s\n",SDL_GetError());
263.     return -1;
264. }
265. SDL_Overlay *bmp;
266. bmp = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height,SDL_YV12_OVERLAY, screen);
267. SDL_Rect rect;
268. rect.x = 0;
269. rect.y = 0;
270. rect.w = screen_w;
271. rect.h = screen_h;
272. //SDL End-----
273. int ret, got_picture;
274. AVPacket *packet=(AVPacket *)av_malloc(sizeof(AVPacket));
275.
276. #if OUTPUT_YUV420P
277. FILE *fp_yuv=fopen("output.yuv", "wb+");
278. #endif
279.
280. struct SwsContext *img_convert_ctx;
281. img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height, A
282. V_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
283. //-----
284. SDL_Thread *video_tid = SDL_CreateThread(sfp_refresh_thread,NULL);
285. //
286. SDL_WM_SetCaption("Simplest FFmpeg Read Camera",NULL);
287. //Event Loop
288. SDL_Event event;
289.
290. for (;;) {
291.     //Wait
292.     SDL_WaitEvent(&event);
293.     if(event.type==SFM_REFRESH_EVENT){
294.         //-----
295.         if(av_read_frame(pFormatCtx, packet)>=0){

```

```

287.         if(packet->stream_index==videoindex){
288.             ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
289.             if(ret < 0){
290.                 printf("Decode Error.\n");
291.                 return -1;
292.             }
293.             if(got_picture){
294.                 SDL_LockYUVOverlay(bmp);
295.                 pFrameYUV->data[0]=bmp->pixels[0];
296.                 pFrameYUV->data[1]=bmp->pixels[2];
297.                 pFrameYUV->data[2]=bmp->pixels[1];
298.                 pFrameYUV->linesize[0]=bmp->pitches[0];
299.                 pFrameYUV->linesize[1]=bmp->pitches[2];
300.                 pFrameYUV->linesize[2]=bmp->pitches[1];
301.                 sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height
, pFrameYUV->data, pFrameYUV->linesize);
302.
303.             #if OUTPUT_YUV420P
304.                 int y_size=pCodecCtx->width*pCodecCtx->height;
305.                 fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
306.                 fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
307.                 fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
308.             #endif
309.
310.                 SDL_UnlockYUVOverlay(bmp);
311.
312.                 SDL_DisplayYUVOverlay(bmp, &rect);
313.
314.             }
315.         }
316.         av_free_packet(packet);
317.     }else{
318.         //Exit Thread
319.         thread_exit=1;
320.     }
321. }else if(event.type==SDL_QUIT){
322.     thread_exit=1;
323. }else if(event.type==SFM_BREAK_EVENT){
324.     break;
325. }
326.
327. }
328.
329.
330.     sws_freeContext(img_convert_ctx);
331.
332. #if OUTPUT_YUV420P
333.     fclose(fp_yuv);
334. #endif
335.
336.     SDL_Quit();
337.
338.     //av_free(out_buffer);
339.     av_free(pFrameYUV);
340.     avcodec_close(pCodecCtx);
341.     avformat_close_input(&pFormatCtx);
342.
343.     return 0;
344. }

```

结果

程序的运行效果如下。输出了摄像头的数据。



可以通过下面的宏定义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VFW或者是Dshow打开摄像头：

```
[cpp]
1. // '1' Use Dshow
2. // '0' Use VFW
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_readcamera.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_readcamera.cpp -g -o simplest_ffmpeg_readcamera.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_readcamera.cpp -g -o simplest_ffmpeg_readcamera.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_readcamera.cpp -g -o simplest_ffmpeg_readcamera.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

此外，增加了MacOS下使用avfoundation读取摄像头的代码。

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/39702113>

文章标签：[ffmpeg](#) [libavdevice](#) [编程](#) [摄像头](#) [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