# 原 最简单的基于FFmpeg的AVDevice例子（屏幕录制）

2014年10月02日 13:19:23 阅读数：49569

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

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

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
/**
 * 最简单的基于FFmpeg的AVDevice例子（屏幕录制）
 * Simplest FFmpeg Device (Screen Capture)
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 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
 * 的libavdevice类库最简单的例子。通过该例子，可以学习FFmpeg中
 * libavdevice类库的使用方法。
 * 本程序在Windows下可以使用2种方式录制屏幕：
 *  1.gdigrab: Win32下的基于GDI的屏幕录制设备。
 *             抓取桌面的时候，输入URL为“desktop”。
 *  2.dshow: 使用Directshow。注意需要安装额外的软件screen-capture-recorder
 * 在Linux下可以使用x11grab录制屏幕。
 * 在MacOS下可以使用avfoundation录制屏幕。
 *
 * This software capture screen of computer. It's the simplest example
 * about usage of FFmpeg's libavdevice Library.
 * It's suiltable for the beginner of FFmpeg.
 * This software support 2 methods to capture screen in Microsoft Windows:
 *  1.gdigrab: Win32 GDI-based screen capture device.
 *             Input URL in avformat_open_input() is "desktop".
 *  2.dshow: Use Directshow. Need to install screen-capture-recorder.
 * It use x11grab to capture screen in Linux.
 * It use avfoundation to capture screen in MacOS.
 */


#include <stdio.h>

#define __STDC_CONSTANT_MACROS

#ifdef _WIN32
//Windows
extern "C"
{
#include "libavcodec/avcodec.h"
#include "libavformat/avformat.h"
#include "libswscale/swscale.h"
#include "libavdevice/avdevice.h"
#include "SDL/SDL.h"
};
#else
//Linux...
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavcodec/avcodec.h>
#include <libavformat/avformat.h>
#include <libswscale/swscale.h>
#include <libavdevice/avdevice.h>
#include <SDL/SDL.h>
#ifdef __cplusplus
};
#endif
#endif

//Output YUV420P
#define OUTPUT_YUV420P 0
//'1' Use Dshow
//'0' Use GDIgrab
#define USE_DSHOW 0

//Refresh Event
#define SFM_REFRESH_EVENT  (SDL_USEREVENT + 1)

#define SFM_BREAK_EVENT  (SDL_USEREVENT + 2)

int thread_exit=0;

int sfp_refresh_thread(void *opaque)
{
    thread_exit=0;
    while (!thread_exit) {
        SDL_Event event;
        event.type = SFM_REFRESH_EVENT;
        SDL_PushEvent(&event);
        SDL_Delay(40);
    }
    thread_exit=0;
```

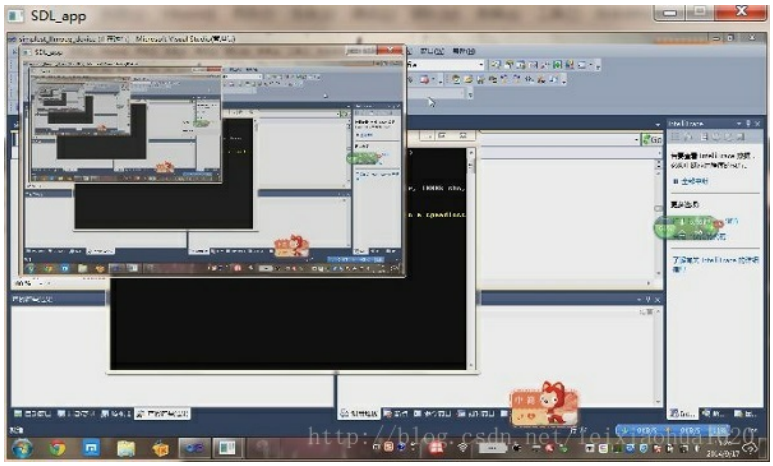
```
    thread_exit=0;
    //Break
    SDL_Event event;
    event.type = SFM_BREAK_EVENT;
    SDL_PushEvent(&event);

    return 0;
}

//Show Dshow Device
void show_dshow_device(){
    AVFormatContext *pFormatCtx = avformat_alloc_context();
    AVDictionary* options = NULL;
    av_dict_set(&options,"list_devices","true",0);
    AVInputFormat *iformat = av_find_input_format("dshow");
    printf("========Device Info=============\n");
    avformat_open_input(&pFormatCtx,"video=dummy",iformat,&options);
    printf("================================\n");
}

//Show AVFoundation Device
void show_avfoundation_device(){
    AVFormatContext *pFormatCtx = avformat_alloc_context();
    AVDictionary* options = NULL;
    av_dict_set(&options,"list_devices","true",0);
    AVInputFormat *iformat = av_find_input_format("avfoundation");
    printf("==AVFoundation Device Info===\n");
    avformat_open_input(&pFormatCtx,"",iformat,&options);
    printf("=============================\n");
}



int main(int argc, char* argv[])
{

    AVFormatContext *pFormatCtx;
    int             i, videoindex;
    AVCodecContext  *pCodecCtx;
    AVCodec         *pCodec;

    av_register_all();
    avformat_network_init();
    pFormatCtx = avformat_alloc_context();

    //Open File
    //char filepath[]="src01_480x272_22.h265";
    //avformat_open_input(&pFormatCtx,filepath,NULL,NULL)

    //Register Device
    avdevice_register_all();
    //Windows
#ifdef _WIN32
#if USE_DSHOW
    //Use dshow
    //
    //Need to Install screen-capture-recorder
    //screen-capture-recorder
    //Website: http://sourceforge.net/projects/screencapturer/
    //
    AVInputFormat *ifmt=av_find_input_format("dshow");
    if(avformat_open_input(&pFormatCtx,"video=screen-capture-recorder",ifmt,NULL)!=0){
        printf("Couldn't open input stream.\n");
        return -1;
    }
#else
    //Use gdigrab
    AVDictionary* options = NULL;
    //Set some options
    //grabbing frame rate
    //av_dict_set(&options,"framerate","5",0);
    //The distance from the left edge of the screen or desktop
    //av_dict_set(&options,"offset_x","20",0);
    //The distance from the top edge of the screen or desktop
    //av_dict_set(&options,"offset_y","40",0);
    //Video frame size. The default is to capture the full screen
    //av_dict_set(&options,"video_size","640x480",0);
    AVInputFormat *ifmt=av_find_input_format("gdigrab");
    if(avformat_open_input(&pFormatCtx,"desktop",ifmt,&options)!=0){
        printf("Couldn't open input stream.\n");
        return -1;
    }

#endif
#elif defined linux
    //Linux
    AVDictionary* options = NULL;
    //Set some options
    //grabbing frame rate
    //av_dict_set(&options,"framerate","5",0);
    //Make the grabbed area follow the mouse
    //av_dict_set(&options,"follow_mouse","centered",0);
```

```
    //Video frame size. The default is to capture the full screen
    //av_dict_set(&options,"video_size","640x480",0);
    AVInputFormat *ifmt=av_find_input_format("x11grab");
    //Grab at position 10,20
    if(avformat_open_input(&pFormatCtx,":0.0+10,20",ifmt,&options)!=0){
        printf("Couldn't open input stream.\n");
        return -1;
    }
#else
    show_avfoundation_device();
    //Mac
    AVInputFormat *ifmt=av_find_input_format("avfoundation");
    //Avfoundation
    //[video]:[audio]
    if(avformat_open_input(&pFormatCtx,"1",ifmt,NULL)!=0){
        printf("Couldn't open input stream.\n");
        return -1;
    }
#endif

    if(avformat_find_stream_info(pFormatCtx,NULL)<0)
    {
        printf("Couldn't find stream information.\n");
        return -1;
    }
    videoindex=-1;
    for(i=0; i<pFormatCtx->nb_streams; i++)
        if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO)
        {
            videoindex=i;
            break;
        }
    if(videoindex==-1)
    {
        printf("Didn't find a video stream.\n");
        return -1;
    }
    pCodecCtx=pFormatCtx->streams[videoindex]->codec;
    pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
    if(pCodec==NULL)
    {
        printf("Codec not found.\n");
        return -1;
    }
    if(avcodec_open2(pCodecCtx, pCodec,NULL)<0)
    {
        printf("Could not open codec.\n");
        return -1;
    }
    AVFrame *pFrame,*pFrameYUV;
    pFrame=av_frame_alloc();
    pFrameYUV=av_frame_alloc();
    //unsigned char *out_buffer=(unsigned char *)av_malloc(avpicture_get_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->heigh
t));
    //avpicture_fill((AVPicture *)pFrameYUV, out_buffer, AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height);
    //SDL----------------------------
    if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
        printf( "Could not initialize SDL - %s\n", SDL_GetError());
        return -1;
    }
    int screen_w=640,screen_h=360;
    const SDL_VideoInfo *vi = SDL_GetVideoInfo();
    //Half of the Desktop's width and height.
    screen_w = vi->current_w/2;
    screen_h = vi->current_h/2;
    SDL_Surface *screen;
    screen = SDL_SetVideoMode(screen_w, screen_h, 0,0);

    if(!screen) {
        printf("SDL: could not set video mode - exiting:%s\n",SDL_GetError());
        return -1;
    }
    SDL_Overlay *bmp;
    bmp = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height,SDL_YV12_OVERLAY, screen);
    SDL_Rect rect;
    rect.x = 0;
    rect.y = 0;
    rect.w = screen_w;
    rect.h = screen_h;
    //SDL End------------------------
    int ret, got_picture;

    AVPacket *packet=(AVPacket *)av_malloc(sizeof(AVPacket));

#if OUTPUT_YUV420P
    FILE *fp_yuv=fopen("output.yuv","wb+");
#endif

    struct SwsContext *img_convert_ctx;
    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);
```

```
    //------------------------------
    SDL_Thread *video_tid = SDL_CreateThread(sfp_refresh_thread,NULL);
    //
    SDL_WM_SetCaption("Simplest FFmpeg Grab Desktop",NULL);
    //Event Loop
    SDL_Event event;

    for (;;) {
        //Wait
        SDL_WaitEvent(&event);
        if(event.type==SFM_REFRESH_EVENT){
            //------------------------------
            if(av_read_frame(pFormatCtx, packet)>=0){
                if(packet->stream_index==videoindex){
                    ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
                    if(ret < 0){
                        printf("Decode Error.\n");
                        return -1;
                    }
                    if(got_picture){
                        SDL_LockYUVOverlay(bmp);
                        pFrameYUV->data[0]=bmp->pixels[0];
                        pFrameYUV->data[1]=bmp->pixels[2];
                        pFrameYUV->data[2]=bmp->pixels[1];
                        pFrameYUV->linesize[0]=bmp->pitches[0];
                        pFrameYUV->linesize[1]=bmp->pitches[2];
                        pFrameYUV->linesize[2]=bmp->pitches[1];
                        sws_scale(img_convert_ctx, (const unsigned char* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height
, pFrameYUV->data, pFrameYUV->linesize);

#if OUTPUT_YUV420P
                        int y_size=pCodecCtx->width*pCodecCtx->height;
                        fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
                        fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
                        fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
#endif
                        SDL_UnlockYUVOverlay(bmp);

                        SDL_DisplayYUVOverlay(bmp, &rect);

                    }
                }
                av_free_packet(packet);
            }else{
                //Exit Thread
                thread_exit=1;
            }
        }else if(event.type==SDL_QUIT){
            thread_exit=1;
        }else if(event.type==SFM_BREAK_EVENT){
            break;
        }

    }


    sws_freeContext(img_convert_ctx);

#if OUTPUT_YUV420P
    fclose(fp_yuv);
#endif

    SDL_Quit();

    //av_free(out_buffer);
    av_free(pFrameYUV);
    avcodec_close(pCodecCtx);
    avformat_close_input(&pFormatCtx);

    return 0;
}
```

## 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
#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' Use Dshow
//'0' Use GDIgrab
#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
::VS2010 Environment
call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
::include
@set INCLUDE=include;%INCLUDE%
::lib
@set LIB=lib;%LIB%
::compile and link
cl simplest_ffmpeg_grabdesktop.cpp /MD /link SDL.lib SDLmain.lib avcodec.lib ^
avformat.lib avutil.lib avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib ^
/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
g++ simplest_ffmpeg_grabdesktop.cpp -g -o simplest_ffmpeg_grabdesktop.exe \
-I /usr/local/include -L /usr/local/lib \
-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
gcc simplest_ffmpeg_grabdesktop.cpp -g -o simplest_ffmpeg_grabdesktop.out \
-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
gcc simplest_ffmpeg_grabdesktop.cpp -g -o simplest_ffmpeg_grabdesktop.out \
-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。



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