FFmpeg获取DirectShow设备数据（摄像头，录屏）

2014年08月02日 00:57:27　阅读数：89168

这两天研究了FFmpeg获取DirectShow设备数据的方法，在此简单记录一下以作备忘。本文所述的方法主要是对应Windows平台的。

1.　列设备

```
[plain]
1. ffmpeg -list_devices true -f dshow -i dummy
```

命令执行后输出的结果如下（注：中文的设备会出现乱码的情况）。列表显示设备的名称很重要，输入的时候都是使用“-f dshow -i video="{设备名}"”的方式。

我自己的机器上列出了以下设备：

```
[plain]
1. [dshow @0388f5e0] DirectShow video devices
2. [dshow @0388f5e0]  "Integrated Camera"
3. [dshow @0388f5e0] "screen-capture-recorder"
4. [dshow @0388f5e0] DirectShow audio devices
5. [dshow @0388f5e0]  "鑻呭□榄﹀簱椋?(Conexant20672 SmartAudi"
6. [dshow @0388f5e0]  "virtual-audio-capturer"
```

下文的测试中，使用其中的两个视频输入："Integrated Camera"和"screen-capture-recorder"。

注：音频设备出现乱码，这个问题的解决方法会随后提到。

2.　获取摄像头数据（保存为本地文件或者发送实时流）

2.1. 编码为H.264，保存为本地文件

下面这条命令，实现了从摄像头读取数据并编码为H.264，最后保存成mycamera.mkv。

```
[plain]
1. ffmpeg -f dshow -i video="Integrated Camera" -vcodec libx264 mycamera.mkv
```

2.2. 直接播放摄像头的数据

使用ffplay可以直接播放摄像头的数据，命令如下：

```
[plain]
1. ffplay -f dshow -i video="Integrated Camera"
```

如果设备名称正确的话，会直接打开本机的摄像头，如图所示。

注：除了使用DirectShow作为输入外，使用VFW也可以读取到摄像头的数据，例如下述命令可以播放摄像头数据：

```
ffplay -f vfwcap -i 0
```

此外，可以使用FFmpeg的list_options查看设备的选项：

```
ffmpeg -list_options true -f dshow -i video="Integrated Camera"
```

输出如下：

```
[dshow @ 03845420] DirectShow video device options
[dshow @ 03845420]  Pin "鏈疊幏"
[dshow @ 03845420]   pixel_format=bgr24  min s=640x480 fps=15 max s=640x480 fps=30
[dshow @ 03845420]   pixel_format=bgr24  min s=640x360 fps=15 max s=640x360 fps=30
[dshow @ 03845420]   pixel_format=bgr24  min s=352x288 fps=15 max s=352x288 fps=30
[dshow @ 03845420]   pixel_format=bgr24  min s=320x240 fps=15 max s=320x240 fps=30
[dshow @ 03845420]   pixel_format=bgr24  min s=800x448 fps=1 max s=800x448 fps=15
[dshow @ 03845420]   pixel_format=bgr24  min s=960x544 fps=1 max s=960x544 fps=10
[dshow @ 03845420]   pixel_format=bgr24  min s=1280x720 fps=1 max s=1280x720 fps=10
[dshow @ 03845420]   pixel_format=bgr24  min s=424x240 fps=15 max s=424x240 fps=30
[dshow @ 03845420]   pixel_format=yuyv422  min s=640x480 fps=15 max s=640x480 fps=30
[dshow @ 03845420]   pixel_format=yuyv422  min s=640x360 fps=15 max s=640x360 fps=30
[dshow @ 03845420]   pixel_format=yuyv422  min s=352x288 fps=15 max s=352x288 fps=30
[dshow @ 03845420]   pixel_format=yuyv422  min s=320x240 fps=15 max s=320x240 fps=30
[dshow @ 03845420]   pixel_format=yuyv422  min s=800x448 fps=1 max s=800x448 fps=15
[dshow @ 03845420]   pixel_format=yuyv422  min s=960x544 fps=1 max s=960x544 fps=10
[dshow @ 03845420]   pixel_format=yuyv422  min s=1280x720 fps=1 max s=1280x720 fps=10
[dshow @ 03845420]   pixel_format=yuyv422  min s=424x240 fps=15 max s=424x240 fps=30
[dshow @ 03845420]   vcodec=mjpeg  min s=640x480 fps=15 max s=640x480 fps=30
[dshow @ 03845420]   vcodec=mjpeg  min s=640x360 fps=15 max s=640x360 fps=30
[dshow @ 03845420]   vcodec=mjpeg  min s=352x288 fps=15 max s=352x288 fps=30
[dshow @ 03845420]   vcodec=mjpeg  min s=320x240 fps=15 max s=320x240 fps=30
[dshow @ 03845420]   vcodec=mjpeg  min s=800x448 fps=15 max s=800x448 fps=30
[dshow @ 03845420]   vcodec=mjpeg  min s=960x544 fps=15 max s=960x544 fps=30
[dshow @ 03845420]   vcodec=mjpeg  min s=1280x720 fps=15 max s=1280x720 fps=30
```

可以通过输出信息设置摄像头的参数。

例如，设置摄像头分辨率为1280x720

```
ffplay -s 1280x720 -f dshow -i video="Integrated Camera"
```

设置分辨率为424x240

```
ffplay -s 424x240 -f dshow -i video="Integrated Camera"
```

2.3. 编码为H.264，发布UDP

下面这条命令，实现了：获取摄像头数据->编码为H.264->封装为UDP并发送至组播地址。

```
ffmpeg -f dshow -i video="Integrated Camera" -vcodec libx264 -preset:v ultrafast -tune:v zerolatency -
f h264 udp://233.233.233.223:6666
```

注1：考虑到提高libx264的编码速度，添加了-preset:v ultrafast和-tune:v zerolatency两个选项。

注2：高分辨率的情况下，使用UDP可能出现丢包的情况。为了避免这种情况，可以添加–s 参数（例如-s 320x240）调小分辨率。

2.4. 编码为H.264，发布RTP

下面这条命令，实现了：获取摄像头数据->编码为H.264->封装为RTP并发送至组播地址。

```
ffmpeg -f dshow -i video="Integrated Camera" -vcodec libx264 -preset:v ultrafast -tune:v zerolatency -
f rtp rtp://233.233.233.223:6666>test.sdp
```

注1：考虑到提高libx264的编码速度，添加了-preset:v ultrafast和-tune:v zerolatency两个选项。

注2：结尾添加“>test.sdp”可以在发布的同时生成sdp文件。该文件可以用于该视频流的播放。

2.5. 编码为H.264，发布RTMP

下面这条命令，实现了：获取摄像头数据->编码为H.264->并发送至RTMP服务器。

```
ffmpeg -f dshow -i video="Integrated Camera" -vcodec libx264 -preset:v ultrafast -tune:v zerolatency -
f flv rtmp://localhost/oflaDemo/livestream
```

2.6. 编码为MPEG2，发布UDP

与编码为H.264类似，指明-vcodec即可。

```
ffmpeg -f dshow -i video="Integrated Camera" -vcodec mpeg2video -f mpeg2video udp://233.233.233.223:6666
```

播放MPEG2的UDP流如下。指明-vcodec为mpeg2video即可

```
ffplay -vcodec mpeg2video udp://233.233.233.223:6666
```

3. 屏幕录制（Windows平台下保存为本地文件或者发送实时流）

Linux下使用FFmpeg进行屏幕录制相对比较方便，可以使用x11grab，使用如下的命令：

```
ffmpeg -f x11grab -s 1600x900 -r 50 -vcodec libx264 –preset:v ultrafast –tune:v zerolatency -crf 18 -f mpegts udp://localhost:1234
```

详细时使用方式可以参考这篇文章： DesktopStreaming With FFmpeg for Lower Latency

Linux录屏在这里不再赘述。在Windows平台下屏幕录像则要稍微复杂一些。在Windows平台下，使用-dshow取代x11grab。一句话介绍：注册录屏dshow滤镜（例如screen-capture-recorder），然后通过dshow获取录屏图像然后编码处理。

因此，在使用FFmpeg屏幕录像之前，需要先安装dshow滤镜。在这里推荐一个软件：screen capture recorder。安装这个软件之后，就可以通过FFmpeg屏幕录像了。

screen capture recorder项目主页：

http://sourceforge.net/projects/screencapturer/

下载地址：

http://sourceforge.net/projects/screencapturer/files

下载完后，一路“Next”即可安装完毕。注意，需要Java运行环境（Java Runtime Environment），如果没有的话下载一个就行。

screen capture recorder本身就可以录屏，不过这里我们使用FFmpeg进行录屏。

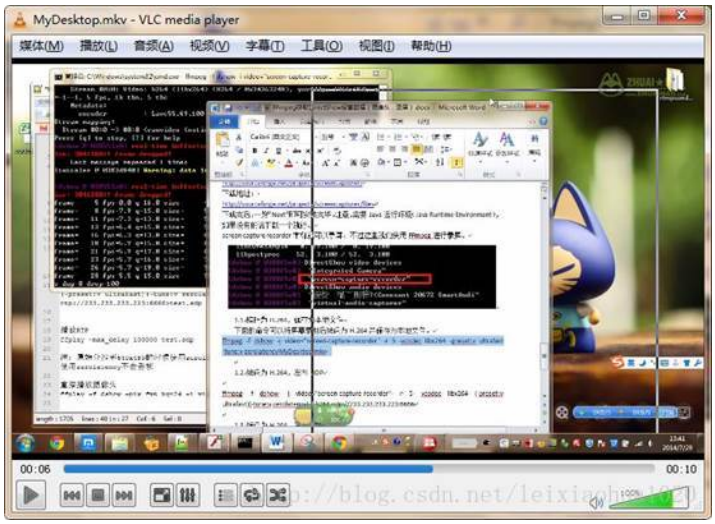
3.1. 编码为H.264，保存为本地文件

下面的命令可以将屏幕录制后编码为H.264并保存为本地文件。

```
ffmpeg -f dshow -i video="screen-capture-recorder" -r 5 -vcodec libx264 -preset:v ultrafast -tune:v zerolatency MyDesktop.mkv
```

注：“-r 5”的意思是把帧率设置成5。

最后得到的效果如下图。



此外，也可以录声音，声音输入可以分成两种：一种是真人说话的声音，通过话筒输入；一种是虚拟的声音，即录屏的时候电脑耳机里的声音。下面两条命令可以分别录制话筒的声音和电脑耳机里的声音。

录屏，伴随话筒输入的声音

```
[plain]
1. ffmpeg -f dshow -i video="screen-capture-recorder" -f dshow -i audio="鏁呴□榈﹀厠椋?(Conexant 20672 SmartAudi" -r 5 -vcodec libx264 -p
reset:v ultrafast -tune:v zerolatency -acodec libmp3lame MyDesktop.mkv
```

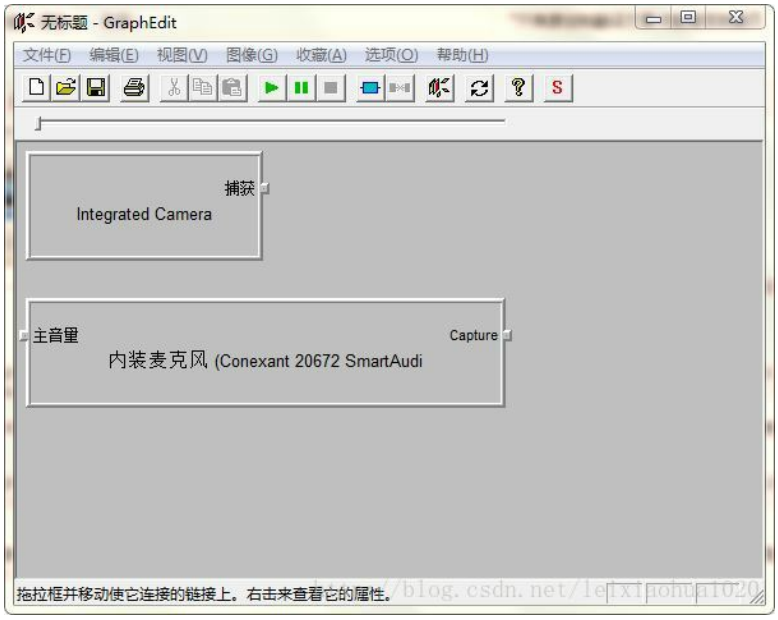
上述命令有问题：audio那里有乱码，把乱码ANSI转UTF-8之后，开始测试不行，后来发现是自己疏忽大意，乱码部分转码后为“内装麦克风 ”，然后接可以正常使用了。因此，命令应该如下图所示：

```
[plain]
1. ffmpeg -f dshow -i video="screen-capture-recorder" -f dshow -i audio="内装麦克风 (Conexant 20672 SmartAudi" -r 5 -vcodec libx264 -
preset:v ultrafast -tune:v zerolatency -acodec libmp3lame MyDesktop.mkv
```

注：

如果不熟悉ANSI转码UTF-8的话，还有一种更简单的方式查看设备的名称。即不使用FFmpeg查看系统DirectShow输入设备的名称，而使用DirectShow SDK自带的工具GraphEdit（或者网上下一个GraphStudioNext）查看输入名称。

打开GraphEdit选择“图像->插入滤镜”



然后就可以通过查看Audio Capture Sources来查看音频输入设备的简体中文名称了。从图中可以看出是“内装麦克风 (Conexant 20672 SmartAudi”。

PS：感觉这条命令适合做讲座之类的时候使用

录屏，伴随耳机输入的声音

```
[plain]
1. ffmpeg -f dshow -i video="screen-capture-recorder" -f dshow -i audio="virtual-audio-capturer" -r 5 -vcodec libx264 -preset:v ultrafa
st -tune:v zerolatency -acodec libmp3lame MyDesktop.mkv
```

PS：测这条命令的时候，这在听歌，因此录制的视频中的音频就是那首歌曲。

3.2. 编码为H.264，发布UDP

下面的命令可以将屏幕录制后编码为H.264并封装成UDP发送到组播地址

```
[plain]
1. ffmpeg -f dshow -i video="screen-capture-recorder" -r 5 -vcodec libx264 -preset:v ultrafast -tune:v zerolatency -f h264 udp://233.23
3.233.223:6666
```

注1：考虑到提高libx264的编码速度，添加了-preset:v ultrafast和-tune:v zerolatency两个选项。

注2：高分辨率的情况下，使用UDP可能出现丢包的情况。为了避免这种情况，可以添加–s 参数（例如-s 320x240）调小分辨率。

3.3. 编码为H.264，发布RTP

下面的命令可以将屏幕录制后编码为H.264并封装成RTP并发送到组播地址

```
[plain]
1. ffmpeg -f dshow -i video="screen-capture-recorder" -vcodec libx264 -preset:v ultrafast -tune:v zerolatency -
f rtp rtp://233.233.233.223:6666>test.sdp
```

注1：考虑到提高libx264的编码速度，添加了-preset:v ultrafast和-tune:v zerolatency两个选项。

注2：结尾添加“>test.sdp”可以在发布的同时生成sdp文件。该文件可以用于该视频流的播放。如下命令即可播放：

```
[plain]
1. ffplay test.sdp
```

3.4. 编码为H.264，发布RTMP

原理同上，不再赘述。

```
[plain]
1. ffmpeg -f dshow -i video="Integrated Camera" -vcodec libx264 -preset:v ultrafast -tune:v zerolatency -
f flv rtmp://localhost/oflaDemo/livestream
```

注意：播放RTMP的时候，-max_delay参数会比较明显的影响延迟，将此参数值设定小一些，有利于降低延时。

```
[plain]
1. ffplay -max_delay 100000 "rtmp://localhost/oflaDemo/livestream live=1"
```

4.另一种屏幕录制的方式（2014.10.1更新）

最近发现FFmpeg还有一个专门用于Windows下屏幕录制的设备：gdigrab。
gdigrab是基于GDI的抓屏设备，可以用于抓取屏幕的特定区域。在这里记录一下gdigrab的用法。
gdigrab通过设定不同的输入URL，支持两种方式的屏幕抓取：
（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。
下面举几个例子。
最简单的抓屏：

```
[plain]
1. ffmpeg -f gdigrab -i desktop out.mpg
```

从屏幕的（10,20）点处开始，抓取640x480的屏幕，设定帧率为5

```
[plain]
1. ffmpeg -f gdigrab -framerate 5 -offset_x 10 -offset_y 20 -video_size 640x480 -i desktop out.mpg
```


文章标签： ffmpeg directshow 录屏 摄像头 编码

个人分类： FFMPEG

所属专栏： FFmpeg