# [FFmpeg Basics学习笔记（1）ffmpeg基础](http://www.cnblogs.com/tocy/p/ffmpeg-basic-learning-1.html)

## FFmpeg的由来

FFmpeg缩写中，FF指的是Fast Forward，mpeg是 Moving Pictures Experts Group的缩写。官网：ffmpeg.org

编译好的可执行程序也可以从官网上下载。当然你也可以自己搭建编译环境编译。

## FFmpeg命令行工具

主要包括：

* ffmpeg 快速的音频、视频编码器/解码器
* ffplay 多媒体播放器
* ffprobe 多媒体文件特征解析

同时FFmpeg编译之后包含libavcodec、libavformat、libavdevice、libavfilter、libavutil、libpostproc、libswresample、libswscale。

## ffmpeg命令行格式

*ffmpeg.exe [global options] [input file options] -i input\_file [output file options] output\_files*  
[]表示可选的参数

## ffplay命令行格式

ffplay和ffmpeg命令行参数一致，只是ffplay没有输出相关的参数。  
比如： *ffmpeg -i input\_file ... test\_options ... output\_files*  
使用ffplay可以直接预览  
*ffplay -i input\_file ... test\_options*

## ffmpeg中数值前缀或后缀语法

ffmpeg中表示数字可以使用K、M、G形式的后缀，分别表示千、兆、吉；默认是10进制的，可以使用后缀B表示以2为单位的KB、MB、GB等。比如在设置转码率的时候可以使用下面几种形式表示视频编码码率为1.5Mb：

*ffmpeg -i input.avi -b:v 1500000 output.mp4*

*ffmpeg -i input.avi -b:v 1500K output.mp4*

*ffmpeg -i input.avi -b:v 1.5M output.mp4*

*ffmpeg -i input.avi -b:v 0.015G output.mp4*

设置转码码率为10MB，形式如下：

ffmpeg -i input.mpg -b:v 10MB output.mp4

## ffmpeg转码及filter

 ffmpeg转码的基本逻辑是如下图所示：

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

| | | |

| input | demuxer | encoded data | decoder

| file | ---------> | packets | -----+

|\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

v

\_\_\_\_\_\_\_\_\_

| |

| decoded |

| frames |

|\_\_\_\_\_\_\_\_\_|

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

| | | | |

| output | <-------- | encoded data | <----+

| file | muxer | packets | encoder

|\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

ffmpeg从输入文件（input\_file）中通过解复用器（demuxer，包含在libavformat库中）读取到压缩之后的编码数据（encoded data）；压缩的数据通过解码器（decoder）的解码，还原为原始的数据（decoded frame，音频PCM、视频YUV），这些数据可作为filter的输入；原始音视频数据通常可以直接传递给编码器（encoder），输出编码后的数据（encoded packet，重新编码），这个过程也可以省略；通过复用器（muxer）的处理输出指定的封装格式。

这里提到filter的概念，在ffmpeg的转码过程中filter是最复杂的。多个filter可以构成filterchain，使用逗号分隔；多个filterchain可以构成filtergraph，用分号分隔。filtergraph分为简单和复杂两种。简单的filtergraph通常是可以线形顺序描述的filtergraph，比如下图：

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

| | | | | | | |

| input | ---> | deinterlace | ---> | scale | ---> | output |

|\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_|

复杂的filtergraph通常是具有多个输入输出文件，并有多条执行路径。比如下面的例子，

\_\_\_\_\_\_\_\_\_

| |

| input 0 |\ \_\_\_\_\_\_\_\_\_\_

|\_\_\_\_\_\_\_\_\_| \ | |

\ \_\_\_\_\_\_\_\_\_ /| output 0 |

\ | | / |\_\_\_\_\_\_\_\_\_\_|

\_\_\_\_\_\_\_\_\_ \| complex | /

| | | |/

| input 1 |---->| filter |\

|\_\_\_\_\_\_\_\_\_| | | \ \_\_\_\_\_\_\_\_\_\_

/| graph | \ | |

/ | | \| output 1 |

\_\_\_\_\_\_\_\_\_ / |\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_|

| | /

| input 2 |/

|\_\_\_\_\_\_\_\_\_|

 filter在实际使用中语法如下：（注意区分，前面是输出标签，后面是输出标签）

[in\_label1][in\_label2][...] filter\_name=parameters [out\_label1][out\_label2][...]

举例说明下，先看下面语法：  
**ffplay -i i.mpg -vf split[a][b];[a]pad=2\*iw[A];[b]hqdn3d[B];[A][B]overlay=w**

最终的filter是这样的：

　　[a]　　[a]pad=2\*iw[A]　　[A]

split　　　　　　　　　　　　　　　　overlay=w

　　[b]　　[b]hqdn3d[B]　　　 [B]

（后续部分内容会介绍不同的filter语法）

## Lavfi virtual device

-f表示使用filter，而lavfi是libavfilter virtual input device的别名。其支持的命令行格式如下：

-f lavfi [-graph [-graph\_file]]

-graph是作为输入的filtergraph，graph\_file表示filtergraph文件的。

比如下面输入显示SMPTE测试图：

ffplay -f lavfi -i smptebars

## 颜色的表示方法

* 固定的名字（可以查看颜色表），比如red、blue；
* 十六进制的颜色描述0xRRGGBB[@AA]，注意透明度A必须是浮点数，范围[0.0,1.0]或者十六进制描述（0x前缀）；
* #RRGGBB@AA，网页中描述颜色的格式，AA使用十六进制；
* random，系统随机计算一个颜色；

可以用下面语句验证下颜色的描述：

ffplay -f lavfi -i color=c=0xff0000@0.5

ffplay -f lavfi -i color=c=blue

ffplay -f lavfi -i color=c=#0000ff@0x80

ffplay -f lavfi -i color=c=random

## ffmpeg帮助文档查询

ffmpeg.exe支持以下命令后缀 -h、-h long、-h topic  
*ffmpeg -h decoder=decoder\_name*  
*ffmpeg -h encoder=encoder\_name*

* 支持的码流filter -bsfs
* 可用的编解码器 -codecs
* 可用的解码器 -decoders
* 可用解码器 -encoders
* 可用filter -filters
* 支持的音视频格式（后缀或文件格式） -formats
* 支持的音频声道layout -layouts
* 支持的像素格式 -pix\_fmts
* 支持的协议类型 -protocols
* 支持的音频数字化精度 -sample\_fmts

## 附加说明

本部分内容主要涉及《Ffmpeg Basic》的前两章，建议认真阅读下，这将是后续学习和扩展的基础，不需要全部搞明白，但是需要理解ffmpeg基本的转码、和libavfilter的工作原理。  
至于ffmpeg的帮助文档，在需要的时候知道哪里可以找到，可以参考什么就可以了。

参考：<http://ffmpeg.org/ffmpeg-filters.html#Filtering-Introduction>

# [FFmpeg Basics学习笔记（2）](http://www.cnblogs.com/tocy/p/ffmpeg-basic-learning-2.html)

## 帧率 fps的概念

帧率，单位FPS（frame per second）, 用于衡量视频每秒的处理帧数，对于编码器而言说明编码器在1s的编码的速度，通常可以使用一帧的编码时间倒数简单计算；对于解码器而言，帧率表示1秒内解码帧数。  
电视节目中比较常用的帧率制式有NTSC、PAL。另外，帧率可能会因为视频是否是交织（interlaved，**i**）或逐行（progressive，**p**）的，描述上有所不同。对于交织的视频，一帧数据包含两场：顶场和底场。  
NTSC标准使用60i fps，意思是每秒60场，亦即30帧。（60 fields），30帧  
PAL标准使用50i fps，意思是每秒50场，亦即25帧。

## 如何设置转码帧率

ffmpeg提供了-r用于设置转码之后的帧率，命令行如下：

ffmpeg -i input.avi -r 30 output.mp4

### 使用fps filter

这里介绍第一个filter，名字是fps，可用于设置输出视频的帧率，语法如下：

fps=fps=number\_of\_frames

比如下面命令行将input.avi的帧率处理成25帧，输出到output.mp4中

ffmpeg -i input.avi -vf fps=fps=25 output.mp4

### 预定义的帧率

ffmpeg提供了几种常用的帧率：

缩写 帧率 准确值

ntsc-film 23.97fps 24000/1001

film 24fps 24/1

pal、qpal、spal 25fps 25/1

ntsc、qntsc、sntsc 29.97fps 30000/1001

在使用-r参数是可以用上面的缩写指代特定帧率。

## 码率 bitrate

码率是指单位时间内处理的音视频数据的比特数，单位bps。在同样的编码条件下，码率决定音视频数据的质量。  
比较常用的码率控制策略有下面三种：

* ABR：平均码率，固定编码大小的情况下较常用
* CBR：固定码率，多数用于流媒体串流或直播，用于保存不实用。
* VBR：自适应码率，简单场景使用低码率编码，复杂场景使用高码率编码。同等大小情况下，编码质量比VBR和CBR要好。

ffmpeg中设置码率的参数在[FFmpeg Basics学习笔记（1）ffmpeg基础](http://www.cnblogs.com/tocy/p/ffmpeg-basic-learning-1.html)已经介绍了，有兴趣可以看看。

## CBR设置

设置为CBR码率输出时，需要指定minrate和maxrate参数，并保证-b和二者的参数相同即可。

ffmpeg -i in.avi -b 500K -minrate 500K -maxrate 500K -bufsize 1M out.mkv

## 限制输出文件的大小

可以使用-fs命令

ffmpeg -i in.avi -fs 10MB out.mp4

## 视频缩放

视频缩放可分为两种，保持宽高比的缩放（Scale）、任意比例缩放（Resize）。前者通常不破坏原始图像的显示比例，视觉效果通常比较好，类似数学中的等比例变换；后者比较灵活，可以任意变换。

比如下面的缩放处理，将源视频缩放为320x240大小

ffmpeg -i in.mp4 -s 320x240 out.mp4

ffmpeg中预先定义了一些分辨率，比如：（注意描述视频时通常是宽x高）

* qcif、cif、4cif、16cif,宽高都是2倍递增的关系。只需要记住cif：352x288
* vga 表示640x480分辨率
* hd720 表示1280x720分辨率，通常说的720p
* hd1080 表示1920x1080分辨率，通常说的1080p

### scale video filter

scale视频filter支持视频的缩放。其基础语法如下：

scale=width:height[inter1={1|-1}]

其中width和height表示缩放之后的视频宽高，inter表示是否是自动启用去交织的功能。在实际命令函中还可以出现以下参数：

* iw、ih 输入源的视频宽高
* ow、oh 输出的视频宽高
* a、sar、dar（a 宽高比=iw/ih； sar 输入源的宽高比； dar 输入源的显示宽高比=a\*sar）
* hsub、vsub 水平和垂直方向上色度分量的采样步长，比如对YUV422p的格式，hsub=2、vsub=1

具体使用可以参考下面命令行：

ffmpeg -i in.mp4 -vf sacle=320:240 out.mp4

下面两条语句分别实现等比例缩放、指定宽高的等比缩放

ffmpeg -i in.mp4 -vf scale=iw*0.6:ih*0.6 out.mp4

ffmpeg -i in.mp4 -vf scale=200:200/a out.mp4

## 视频裁剪 Crop

视频裁剪指的是从指定视频的某个区域（通常是矩形）取出部分画面。

### crop video filter

crop filter的语法如下：

crop=ow[:oh[❌[y:[:keep\_aspect]]]]

除了能使用scale filter提供的参数外，还可以使用用x、y、n、pos、t等参数，具体含义如下：

* x,y 裁剪区域起始点坐标，默认为((iw-ow)/2, (ih-oh)/2)
* n、pos、t分别用于标识当前帧数、位置及时间戳

只截取源视频的中心区域（长宽取一半）

ffmpeg -i in.avi -vf crop=iw/2:ih/2 out.mp4

1. **cropdetect video filter**

自动截取非黑色的区域。在4:3和16:9视频显示时通常填充黑边，cropdetect filter会比较有用。其语法如下：

cropdetect[=limit[:round[:reset]]]

各字段含义：

* limit: 黑色判定门限，取值范围[0,255]，默认为24
* round，向上取整参数，整数对齐，比如必须要求输出宽高是16的倍数、偶数。
* reset，重新计算裁剪区域的间隔

ffmpeg命令行格式，

ffmpeg -i input.mpg -vf cropdetect=limit=0 output.mp4

1. **pad video filter**

区域填充特殊颜色，通常色彩填充多用于视频分辨率和屏幕分辨率不一致的情况下，为了保证视频的效果，保持拉伸的宽高比，在左右填充或者上下填充。pad filter语法如下：

pad=width[:height[:x[:y[:color]]]]

除了可以使用crop filer中参数，这里提供了color，用于指定填充颜色；width和height为输出视频的分辨率；x和y表示输入源的左上角在输出画面上的偏移。

下面语句的功能是在800x640的图片外围填充30个像素的粉红色边框。

ffmpeg -i photo.jpg -vf pad=860:660:30:30:pink framed\_photo.jpg

1. **视频的镜像**
2. **hflip video filter**

hflip实现对视频的水平翻转。语法如下：

-vf hflip

翻转效果类似下面的示例：(翻转之后图像互为水平镜像)

ABC | CBA

DEF | FED

1. **vflip video filter**

vflip实现对视频的垂直翻转。语法如下：

-vf vflip

翻转效果类似下面（垂直镜像）

000 | 888

111 | 111

888 | 000

1. **transpose video filter**

transpose filter用于实现特定角度旋转和翻转配合的效果。其语法如下：

-vf transpose={0,1,2,3}

0 逆时针旋转90°+垂直翻转

1 顺时针旋转90°

2 逆时针旋转90°

3 顺时针旋转90°+垂直翻转

类型0和3是旋转和垂直翻转配合的处理。下面两个输出是一样的：

ffplay -f lavfi -i smptebars -vf transpose=0

ffplay -f lavfi -i smptebars -vf transpose=2,vflip

顺时针旋转90°的效果是这样的（左边是原图，右边是transpose之后的图）

AB CA

CD DB

顺时针旋转90°+垂直翻转的效果如下：

AB DB

CD CA

对比下图即可。

1. **附加说明**

本部分内容主要涉及《Ffmpeg Basic》的第3章到第7章（包含第7章），介绍了比较多的视频编码参数和处理，比如帧率、码率、缩放、裁剪、填充和旋转。  
概念性内容比较多，建议可以简单了解下有关内容，实际需要用到的时候再深入研究下。

1. [FFmpeg Basic学习笔记（3）](http://www.cnblogs.com/tocy/p/ffmpeg-basic-learning-3.html)
2. **视频叠加——画中画效果 overlay video filter**

视频叠加比较常见的应用场景是台标的叠加，在画面左上角、右上角、左下角或右下角叠加台标。  
ffmpeg提供的overlay filter支持视频叠加。基本语法如下：

overlay[=x:y[[:rgb={0,1}]]]

其中x、y是嵌入图像的位置，默认为0；rgb表示输入源的格式，0表示不修改，1表示设置输入源格式为RGB。可使用下面四个参数：

* main\_w或W、main\_h或H，主输入（被嵌入的视频）的宽度和高度
* overlay\_w或w、 overlay\_h或h，叠加输入（嵌入视频）的宽高

下面的命令行可以实现视频叠加：

ffmpeg -i input1 -i input2 -filter\_complex overlay=x:y output

1. **台标叠加测试**

在左上角叠加可以用下面命令行：

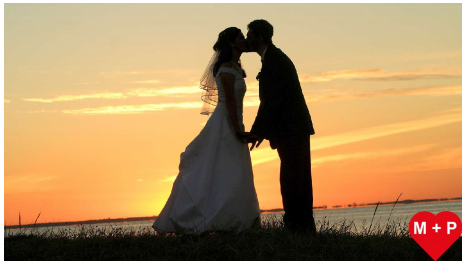
ffmpeg -i pair.mp4 -i logo.png -filter\_complex overlay pair1.mp4

叠加效果如下图：

右下角叠加可以使用下面命令行：

ffmpeg -i pair.mp4 -i logo.png -filter\_complex overlay=W-w:H-h pair3.mp4

叠加效果如下：



1. **画面上的字体显示 drawtext video filter**

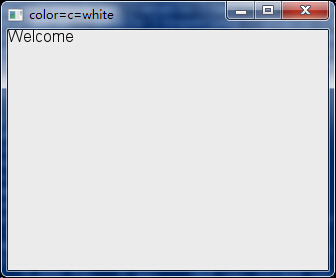
drawtext filter的语法如下：

drawtext=fontfile=font\_f:text=text1[:p3=v3[:p4=v4[...]]]

p3, p4 ... means parameter #3, parameter #4, etc.

由于具体参数比较多，有兴趣的可以阅读FFmpeg Basics的Ch10或者参考ffmpeg的在线文档[drawtext filter](http://ffmpeg.org/ffmpeg-filters.html#drawtext-1)。  
下面是具体的例子：

ffplay -f lavfi -i color=c=white -vf drawtext=fontfile=arial.ttf:text=Welcome

这会在白色窗口的左上角显示Welcome字样。效果如下：  


显示位置可以通过x、y设置，命令如下：

ffplay -f lavfi -i color=c=white -vf  
drawtext="fontfile=arial.ttf:text='Good day':x=(w-tw)/2:y=(h-th)/2"

字体大小和颜色可以通过fontcolor、fontsize设置，命令如下：

ffplay -f lavfi -i color=c=white -vf drawtext=  
"fontfile=arial.ttf:text='Happy Holidays':x=(w-tw)/2:y=(h-th)/2:  
fontcolor=green:fontsize=30"

1. **滚动字幕的实现**

drawtext filter支持t，表示当前运行的描述。配合起来可以实现滚动字幕的效果。  
可参考下面命令行：

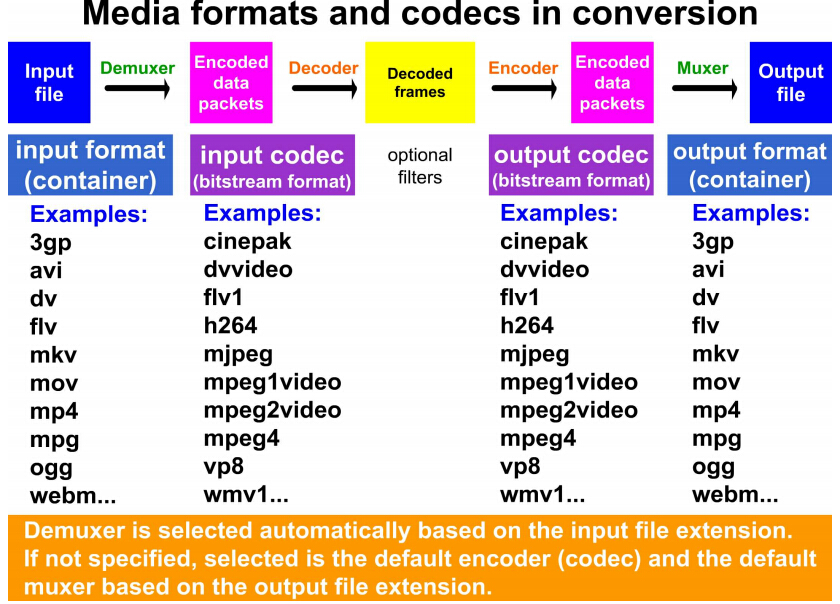
ffmpeg -f lavfi -i color=c=#abcdef -vf drawtext=  
"fontfile=arial.ttf:text='Dynamic RTL text':x=w-t\*50:  
fontcolor=darkorange:fontsize=30" output

水平滚动，每秒滚动50 pixel。

1. **不同格式之间的转换——转码/转封装**

媒体文件格式是指用于存储特定音频或视频的标准。容器格式是指可以存储多个音频或视频流的标准。ffmpeg支持的文件格式可以通过-formats查看。

通常容器支持多种音视频格式的封装，比如AVI、TS、MKV、MP4都是容器，可以支持H.264、MPEG-2、AAC等音视频格式。在实际应用中如果仅仅想更换封装格式，可以考虑使用-c copy、-c:a copy、-c:v copy。

下图说明了实际转码过程中涉及到的封装格式（容器格式）、编解码格式（比特流格式）  


转码过程中每个容器或codec都有私有的参数，可以使用-h encoder=encoder\_name查看。具体参考[FFmpeg Basic学习笔记1](http://www.cnblogs.com/tocy/p/ffmpeg-basic-learning-1.html)。

1. **ffmpeg支持的时间格式**

多媒体处理涉及修改输入源的总长度、延时指定时间或者截取特定时间等操作。ffmpeg中支持的时间格式有两种：

* [-]HH:MM:SS[.m...]
* [-]S+[.m...]

1. **按照时间设置输出流长度**

使用-t命令参数即可。下面语句从输入源中截取三分钟输出：

ffmpeg -i music.mp3 -t 180 music\_3\_minutes.mp3

1. **使用帧数限定输出长度**

* 音频使用 -aframes number或-frames:a number
* 视频使用 -vframes number或-frames:v number

下面命令行指定输出15000帧视频，大约10分钟视频长度（按25fps计算）

ffmpeg -i video.avi -vframes 15000 video\_10\_minutes.avi

1. **指定延时时间**

可使用-ss命令项（seek from start）。调用命令如下：

ffmpeg -i input.avi -ss 10 output.mp4

表示从第10秒开始输出数据。

将-ss和-t配合使用可以截取任意时长的画面，比如下面命令截取输入的第五分钟(240秒~300秒)的视频。

ffmpeg -i video.mpg -ss 240 -t 60 clip\_5th\_minute.mpg

1. **限制处理时间**

使用-timelimit命令可以限制ffmpeg处理的总时间。命令如下：

ffmpeg -i input.mpg -timelimit 600 output.mkv

不管ffmpeg有没有处理完，600s之后程序退出。

1. **播放速度改变的filter（注意修改的是时间戳）**

视频可以使用setpts filter；音频可以使用atempo filter。

1. **Metadata和Subtitle 元数据和字幕流**

Metadata通常用于记录一些标签信息，比如发布者、作者、发布日期、专题名等。字幕流在配音电影中比较常见，通常以独立的字幕文件存在，需要的时候可以加载进来播放。

1. **MetaData**

MetaData在MP3文件中比较常见，在mp3播放时可以看到比较多的类似信息，使用下面命令可以查看metadata：

ffplay -i "Kalimba.mp3"

**创建metadata**

可以使用-metadata key=value的形式创建metadata，比如下面命令行：

ffmpeg -i input -metadata artist=FFmpeg -metadata title="Test 1" output

**删除metadata**

使用-map\_metadata -1可以删除metadata

1. **subtitle**

subtitle有两种，一种是外置的，可以独立编辑发行；一种是内嵌的，嵌入到视频文件中。

1. **subtitles video filter内嵌字幕流**

语法如下：

subtitles=filename[:original\_size]

下面命令行可以实现字幕嵌入：

ffmpeg -i video.avi -vf subtitles=titles.srt video.mp4

1. **附加说明**

本部分内容主要涉及《Ffmpeg Basic》的第8章到第14章（包含第14章），主要是关于视频叠加、字符叠加、subtitle、时间操作、多媒体文件的容器格式和文件格式。

一部分内容没有做详细介绍，有些参数建议参考[FFmpeg的官网](http://ffmpeg.org/)。

# [FFmpeg Basic学习笔记（4）](http://www.cnblogs.com/tocy/p/ffmpeg-basic-learning-4.html)

## 图像处理

常见的图片格式包括YUV、BMP、JPG、GIF、PNG。

### 图像的创建

可以使用下面命令从输入源中截取图像

ffmpeg -i input -ss t image.type

从videoclip.avi中的1：23：45截一张图，输出为JPG。

ffmpeg -i videoclip.avi -ss 01:23:45 image.jpg

### 从视频中生成GIF

GIF是比较常见的动图的格式，在网络上普遍使用。需要注意的GIF中的帧是未压缩的，帧数不能太多，否则可能导致生成文件过大。使用下面命令行可以从.flv中生成GIF。

ffmpeg -i promotion.swf -pix\_fmt rgb24 promotion.gif

### ffmpeg提供的测试输入源

* color 提供纯色的画面生成机制，参数中可指定显示颜色
* smptebars 方形的电视节目测试图片，彩条
* testsrc 圆形的电视节目测试图片，有一个变化的渐变进度条和右侧的一个时间戳。

上面几个输出源的分辨率是320x240。可以用下面命令测试下具体效果：

ffplay -f lavfi -i smptebars

ffplay -f lavfi -i testsrc

ffplay -f lavfi -i color=random

### 视频转换为图片

可以使用下面命令将clip.avi中的视频帧存储为JPG图片（运行的时候建议找个小点的视频，否则你的硬盘可能废掉）

ffmpeg -i clip.avi frame%d.jpg

### 不同图片格式之间的转换

可以使用下面的命令

ffmpeg -i image.type1 image.type2

实际调用，可以将BMP转JPG

ffmpeg -i in.bmp out.jpg

### 使用图片构建视频

视频本来就是由图片附加上时间信息构成的。  
可以使用一张图重复构成一个视频。参考下面命令：

ffmpeg -loop 1 -i photo.jpg -t 10 photo.mp4

输出的视频长度为10s，重复播放一张图片。

也可以将一些列图片合成到一个视频中。比如下面命令，将图片img1.jpg、img2.jpg、……、img100.jpg按照25fps输出为4s的视频。

ffmpeg -f image2 -i img%d.jpg -r 25 video.mp4

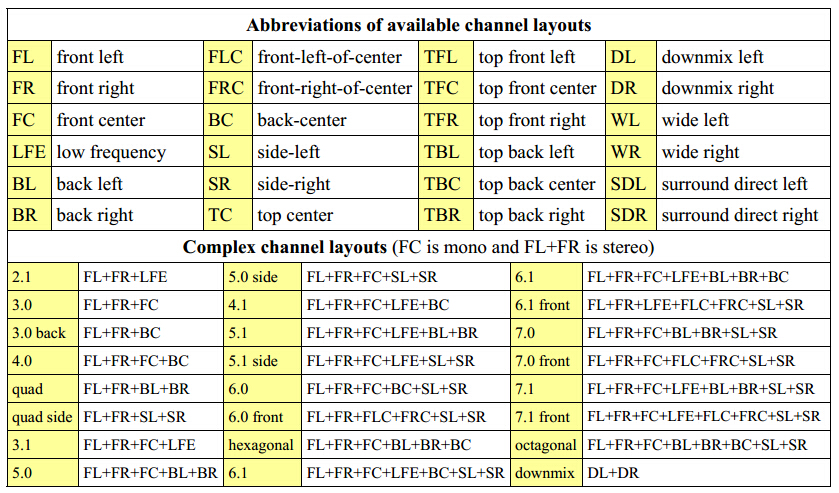
## 数字音频

数字音频是将模拟音频信号通过量化和采样实现的，其中必要重要的概念是量化位数和采样率。量化位数通常有8位、16位、32位，量化位数用于表示音频幅度信号的刻画精细程度；采样率表示单位时间内需要采样的音频点数，比较常用的是8KHz（电话、语音）、44.1kHz（CD）、48kHz（DVD、TV）。

### 常见文件格式

音频常见的文件格式有WAV、MP3、PCM、AAC、WMA等。

### 声道和layout

音频中的声道指的是音频采样设备，对于多声道的音频，通常是多个音频采样点获取的音量值。下图是关于声道的缩写，以及不同的声道布局-layout。  


单声道mono指的是FC，立体声stereo指代FL+FR。

### 设置输出音量大小

可以使用-vol命令来指定输出音量大小，范围为[0,255]。命令行如下：

ffmpeg -i sound.wav -vol 180 sound\_middle\_loud.wav

也可以使用volume audio filter处理音量大小。

## Codec支持的预设 Preset

比如-apre、-vpre、-spre、-fpre分别用于指定音频、视频、字幕和codec的预设。

举例说明，一个简单的 mpeg2.ffpreset 可能只包含一个指令：

vcodec=mpeg2video

我们可以用下面语句使用mpeg2.ffpreset，

ffmpeg -i input -fpre mpeg2.ffpreset -q 1 MPEG2\_video.mpg

再比如可以定义下面flv.ffpreset预设文件（括号内的表示注释）

vcodec=flv (video codec)

b:v=300k (video bitrate)

g=160 (group of picture size)

mbd=2 (macroblock decision algorithm)

flags=+aic+mv0+mv4 (aic - h263 advanced intra coding; always try a mb with

mv=<0,0>; mv4 - use 4 motion vector by macroblock)

trellis=1 (rate-distortion optimal quantization)

ac=1 (number of audio channels)

ar=22050 (audio sampling rate)

b:a=56k (audio bitrate)

使用下面命令调用

ffmpeg -i input.avi -f flv -r 29.97 -vf scale=320:240 -aspect 4:3 -cmp dct -subcmp dct -fpre flv.ffpreset output.flv

通下面语句等价：

ffmpeg -i input.avi -vcodec flv -f flv -r 29.97 -vf scale=320:240  
-aspect 4:3 -b:v 300k -g 160 -cmp dct -subcmp dct -mbd 2 -flags  
+aic+mv0+mv4 -trellis 1 -ac 1 -ar 22050 -b:a 56k output.flv

## 输入设备

windows下可以使用dshow来访问输入设备。比如下面输入可以列出系统中的输入设备(麦克和摄像头)：

ffmpeg -list\_devices 1 -f dshow -i dummy

如果你的主机上有个摄像头，上面的语句输出的名字为“HP Webcam”，那么你就可以使用该设备录像了。命令行如下：

ffplay -f dshow -i video="HP Webcam"

ffmpeg -f dshow -i video="HP Webcam" -f sdl "webcam via ffmpeg"

当然也可以使用-list\_options查看设备支持的参数。

同理，可以使用音频设备。

## 高级话题

* delogo video filter，用于去除视频上的logo。
* deshake video filter，用于去抖动。
* drawbox video filter，用于在视频上绘制颜色矩形框。
* 使用ffmpeg -i input.mpg -f null /dev/null获取视频帧数。
* blackdetect/blackframe video filter，用于黑场检测。
* showinfo video filter，可显示每一帧的信息，包括pts、宽高比等。
* showspectrum multimedia filter，显示音频频谱图。
* showwaves multimedia filter，显示音频波形图。
* flite audio source，用于语音合成。
* ffmpeg支持-debug[:specifier]用于调试输出信息，比如量化参数、pts、运动矢量等。
* 提供-err\_detect命令，用于检测流中可能的错误。
* 显示cpu和内存占用情况，可使用-benchmark。

## 附加说明

本部分内容是Ffmpeg Basic第15章之后的内容，主要整理了下我感兴趣的、不是很熟悉的内容。比如ffmpeg中的图片处理、数字音频处理、Preset定义和使用、输入设备获取和使用、高级应用等。更为详细的内容建议参考ffmpeg官网。