前言

FFMPEG 是特别强大的专门用于处理音视频的开源库。你既可以使用它的 API 对音视频进行处理，也可以使用它提供的工具，如 ffmpeg, ffplay, ffprobe，来编辑你的音视频文件。

本文将简要介绍一下 FFMPEG 库的基本目录结构及其功能，然后详细介绍一下我们在日常工作中，如何使用 ffmpeg 提供的工具来处理音视频文件。

FFMPEG 目录及作用

- libavcodec： 提供了一系列编码器的实现。
- libavformat： 实现在流协议，容器格式及其本 IO 访问。
- libavutil： 包括了 hash 器，解码器和各利工具函数。
- libavfilter： 提供了各种音视频过滤器。
- libavdevice： 提供了访问捕获设备和回放设备的接口。
- libswresample： 实现了混音和重采样。
- libswscale： 实现了色彩转换和缩放工能。

FFMPEG 基本概念

在讲解 FFMPEG 命令之前，我们先要介绍一些音视频格式的基要概念。

- 音 / 视频流

在音视频领域，我们把一路音 / 视频称为一路**流**。如我们小时候经常使用 VCD 看港片，在里边可以选择粤语或国语声音，其实就是 CD 视频文件中存放了两路音频流，用户可以选择其中一路进行播放。

- 容器

我们一般把 MP4、FLV、MOV 等文件格式称之为**容器**。也就是在这些常用格式文件中，可以存放多路音视频文件。以 MP4 为例，就可以存放一路视频流，多路音频流，多路字幕流。

- channel

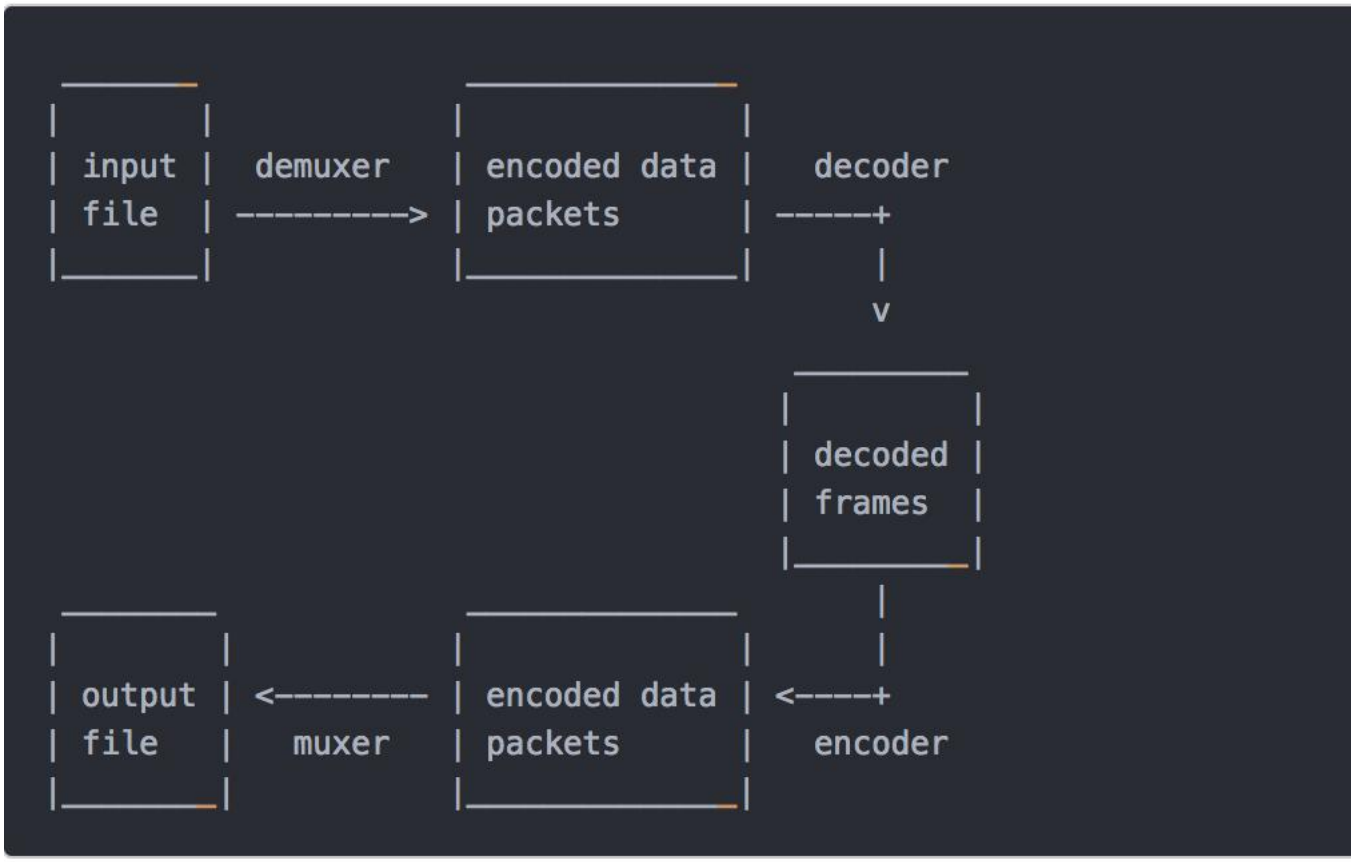
channel 是音频中的概念，称之为声道。在一路音频流中，可以有单声道，双声道或立体声。

FFMPEG 命令

我们按使用目的可以将 FFMPEG 命令分成以下几类：

- 基本信息查询命令
- 录制
- 分解/复用
- 处理原始数据
- 滤镜
- 切割与合并
- 图 / 视互转
- 直播相关

除了 FFMPEG 的基本信息查询命令外，其它命令都按下图所示的流程处理音视频。



然后将编码的数据包传送给解码器（除非为数据流选择了流拷贝，请参阅进一步描述）。 解码器产生未压缩的帧（原始视频/ PCM 音频/ ...），可以通过滤波进一步处理（见下一节）。 在过滤之后，帧被传递到编码器，编码器并输出编码的数据包。 最后，这些传递给复用器，将编码的数据包写入输出文件。

默认情况下，ffmpeg 只包含输入文件中每种类型（视频，音频，字幕）的一个流，并将其添加到每个输出文件中。 它根据以下标准挑选每一个的“最佳”：对于视频，它是具有最高分辨率的流，对于音频，它是具有最多 channel 的流，对于字幕，是第一个字幕流。 在相同类型的几个流相等的情况下，选择具有最低索引的流。

您可以通过使用-vn / -an / -sn / -dn 选项来禁用某些默认设置。 要进行全面的手动控制，请使用-map 选项，该选项禁用刚描述的默认设置。

下面我们就来详细介绍一下这些命令。

基本信息查询命令

FFMPEG 可以使用下面的参数进行基本信息查询。例如，想查询一下现在使用的 FFMPEG 都支持哪些 filter，就可以用 `ffmpeg -filters` 来查询。详细参数说明如下：

参数	说明
-version	显示版本。
-formats	显示可用的格式（包括设备）。
-demuxers	显示可用的 demuxers。
-muxers	显示可用的 muxers。
-devices	显示可用的设备。
-codecs	显示 libavcodec 已知的所有编解码器。

参数	说明
-decoders	显示可用的解码器。
-encoders	显示所有可用的编码器。
-bsfs	显示可用的比特流 filter。
-protocols	显示可用的协议。
-filters	显示可用的 libavfilter 过滤器。
-pix_fmts	显示可用的像素格式。
-sample_fmts	显示可用的采样格式。
-layouts	显示 channel 名称和标准 channel 布局。
-colors	显示识别的颜色名称。

接下来介绍的是 FFMPEG 处理音视频时使用的命令格式与参数。

命令基本格式及参数

下面是 FFMPEG 的基本命令格式：

```
ffmpeg [global_options] {[input_file_options] -i input_url} ...

                        {[output_file_options] output_url} ...
```

ffmpeg 通过 -i 选项读取输任意数量的输入“文件”（可以是常规文件，管道，网络流，抓取设备等，并写入任意数量的输出“文件”。

原则上，每个输入/输出“文件”都可以包含任意数量的不同类型的视频流（视频/音频/字幕/附件/数据）。 流的数量和/或类型是由容器格式来限制。 选择从哪个输入进入到哪个输出将自动完成或使用 -map 选项。

要引用选项中的输入文件，您必须使用它们的索引（从 0 开始）。 例如。 第一个输入文件是 0，第二个输入文件是 1，等等。类似地，文件内的流被它们的索引引用。 例如。 2：3 是指第三个输入文件中的第四个流。

上面就是 FFMPEG 处理音视频的常用命令，下面是一些常用参数：

主要参数

参数	说明
-f fmt（输入/输出）	强制输入或输出文件格式。 格式通常是自动检测输入文件，并从输出文件的文件扩展名中猜测出来，所以在大多数情况下这个选项是不需要的。
-i url（输入）	输入文件的网址
-y（全局参数）	覆盖输出文件而不询问。
-n（全局参数）	不要覆盖输出文件，如果指定的输出文件已经存在，请立即退出。
-c [：stream_specifier] codec（输入/输出，每个流）	选择一个编码器（当在输出文件之前使用）或解码器（当在输入文件之前使用时）用于一个或多个流。codec 是解码器/编码器的名称或 copy（仅输出）以指示该流不被重新编码。如：ffmpeg -i INPUT -map 0 -c:v libx264 -c:a copy OUTPUT
-codec [：stream_specifier]编解码器（输入/输出，每个流）	同 -c

参数	说明
-t duration（输入/输出）	当用作输入选项（在-i 之前）时，限制从输入文件读取的数据的持续时间。当用作输出选项时（在输出 url 之前），在持续时间到达持续时间之后停止输出。
-ss 位置（输入/输出）	当用作输入选项时（在-i 之前），在这个输入文件中寻找位置。请注意，在大多数格式中，不可能精确搜索，因此 ffmpeg 将在位置之前寻找最近的搜索点。 当转码和-accurate_seek 被启用时（默认），搜索点和位置之间的这个额外的分段将被解码和丢弃。 当进行流式复制或使用-noaccurate_seek 时，它将被保留。当用作输出选项（在输出 url 之前）时，解码但丢弃输入，直到时间戳到达位置。
-frames [：stream_specifier] framecount（output，per-stream）	停止在帧计数帧之后写入流。
-filter [：stream_specifier] filtergraph（output，per-stream）	创建由 filtergraph 指定的过滤器图，并使用它来过滤流。filtergraph 是应用于流的 filtergraph 的描述，并且必须具有相同类型的流的单个输入和单个输出。在过滤器图形中，输入与标签中的标签相关联，标签中的输出与标签相关联。有关 filtergraph 语法的更多信息，请参阅 ffmpeg-filters 手册。

视频参数

参数	说明
-vframes num（输出）	设置要输出的视频帧的数量。对于-frames：v，这是一个过时的别名，您应该使用它。
-r [：stream_specifier] fps（输入/输出，每个流）	设置帧率（Hz 值，分数或缩写）。作为输入选项，忽略存储在文件中的任何时间戳，根据速率生成新的时间戳。这与用于-framerate 选项不同（它在 FFmpeg 的旧版本中使用的是相同的）。如果有疑问，请使用-framerate 而不是输入选项-r。

参数	说明
	作为输出选项，复制或丢弃输入帧以实现恒定输出帧频 fps。
-s [：stream_specifier] 大小（输入/输出，每个流）	设置窗口大小。作为输入选项，这是 video_size 专用选项的快捷方式，由某些分帧器识别，其帧尺寸未被存储在文件中。作为输出选项，这会将缩放视频过滤器插入到相应过滤器图形的末尾。请直接使用比例过滤器将其插入到开头或其他地方。格式是'wxh'（默认 - 与源相同）。
-aspect [：stream_specifier] 宽高比（输出，每个流）	设置方面指定的视频显示宽高比。aspect 可以是浮点数字符串，也可以是 num：den 形式的字符串，其中 num 和 den 是宽高比的分子和分母。例如“4：3”，“16：9”，“1.3333”和“1.7777”是有效的参数值。如果与-vcodec 副本一起使用，则会影响存储在容器级别的宽高比，但不会影响存储在编码帧中的宽高比（如果存在）。
-vn（输出）	禁用视频录制。
-vcodec 编解码器（输出）	设置视频编解码器。这是-codec：v 的别名。
-vf filtergraph（输出）	创建由 filtergraph 指定的过滤器图，并使用它来过滤流。

音频参数

参数	说明
-aframes（输出）	设置要输出的音频帧的数量。这是-frames：a 的一个过时的别名。
-ar [：stream_specifier] freq（输入/输出，每个流）	设置音频采样频率。对于输出流，它默认设置为相应输入流的频率。对于输入流，此选项仅适用于音频捕获设备和原始分路器，并映射到相应的分路器选件。

参数	说明
-ac [：stream_specifier]通道（输入/输出，每个流）	设置音频通道的数量。对于输出流，它默认设置为输入音频通道的数量。对于输入流，此选项仅适用于音频捕获设备和原始分路器，并映射到相应的分路器选件。
-an（输出）	禁用录音。
-acodec 编解码器（输入/输出）	设置音频编解码器。这是-codec 的别名：a。
-sample_fmt [：stream_specifier] sample_fmt（输出，每个流）	设置音频采样格式。使用-sample_fmts 获取支持的样本格式列表。
-af filtergraph（输出）	创建由 filtergraph 指定的过滤器图，并使用它来过滤流。

了解了这些基本信息后，接下来我们看看 FFMPEG 具体都能干些什么吧。

录制

首先通过下面的命令查看一下 mac 上都有哪些设备。

```
ffmpeg -f avfoundation -list_devices true -i ""
```

录屏

```
ffmpeg -f avfoundation -i 1 -r 30 out.yuv
```

- -f 指定使用 avfoundation 采集数据。
- -i 指定从哪儿采集数据，它是一个文件索引号。在我的 MAC 上，1 代表桌面（可以通过上面的命令查询设备索引号）。

- -r 指定帧率。按 ffmpeg 官方文档说-r 与-framerate 作用相同，但实际测试时发现不同。-framerate 用于限制输入，而-r 用于限制输出。

注意，桌面的输入对帧率没有要求，所以不用限制桌面的帧率。其实限制了也没用。

录屏+声音

```
ffmpeg  -f avfoundation -i 1:0  -r 29.97 -c:v libx264 -crf 0 -c:a libfdk_aac -profile:a aac_he_v2 -b:a 32k  out.flv
```

- -i 1:0 冒号前面的 "1" 代表的屏幕索引号。冒号后面的"0"代表的声音索相号。
- -c:v 与参数 -vcodec 一样，表示视频编码器。c 是 codec 的缩写，v 是 video 的缩写。
- -crf 是 x264 的参数。 0 表式无损压缩。
- -c:a 与参数 -acodec 一样，表示音频编码器。
- -profile 是 fdk_aac 的参数。aac_he_v2 表式使用 AAC_HE v2 压缩数据。
- -b:a 指定音频码率。 b 是 bitrate 的缩写, a 是 audio 的缩与。

录视频

```
ffmpeg -framerate 30 -f avfoundation -i 0 out.mp4
```

- -framerate 限制视频的采集帧率。这个必须要根据提示要求进行设置，如果不设置就会报错。
- -f 指定使用 avfoundation 采集数据。
- -i 指定视频设备的索引号。

视频+音频

```
ffmpeg -framerate 30 -f avfoundation -i 0:0 out.mp4
```

录音

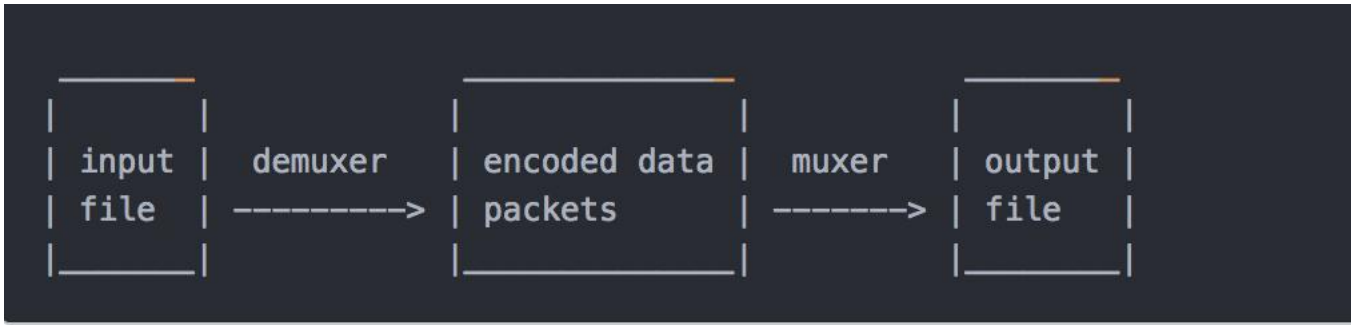
```
ffmpeg -f avfoundation -i :0 out.wav
```

录制音频裸数据

```
ffmpeg  -f avfoundation -i :0 -ar 44100 -f s16le out.pcm
```

分解与复用

流拷贝是通过将 copy 参数提供给-codec 选项来选择流的模式。它使得 ffmpeg 省略了指定流的解码和编码步骤，所以它只能进行多路分解和多路复用。 这对于更改容器格式或修改容器级元数据很有用。 在这种情况下，上图将简化为：



由于没有解码或编码，速度非常快，没有质量损失。 但是，由于许多因素，在某些情况下可能无法正常工作。 应用过滤器显然也是不可能的，因为过滤器处理未压缩的数据。

抽取音频流

```
ffmpeg -i input.mp4 -acodec copy -vn out.aac
```

- acodec: 指定音频编码器，copy 指明只拷贝，不做编解码。
- vn: v 代表视频，n 代表 no 也就是无视频的意思。

抽取视频流

```
ffmpeg -i input.mp4 -vcodec copy -an out.h264
```

- vcodec: 指定视频编码器，copy 指明只拷贝，不做编解码。
- an: a 代表视频，n 代表 no 也就是无音频的意思。

转格式

```
ffmpeg -i out.mp4 -vcodec copy -acodec copy out.flv
```

上面的命令表式的是音频、视频都直接 copy，只是将 mp4 的封装格式转成了 flv。

音视频合并

```
ffmpeg -i out.h264 -i out.aac -vcodec copy -acodec copy out.mp4
```

处理原始数据

提取 YUV 数据

```
ffmpeg -i input.mp4 -an -c:v rawvideo -pixel_format yuv420p out.yuv

ffplay -s wxh out.yuv
```

- -c:v rawvideo 指定将视频转成原始数据
- -pixel_format yuv420p 指定转换格式为 yuv420p

YUV 转 H264

```
ffmpeg -f rawvideo -pix_fmt yuv420p -s 320x240 -r 30 -i out.yuv -c:v libx264 -f

rawvideo out.h264
```

提取 PCM 数据

```
ffmpeg -i out.mp4 -vn -ar 44100 -ac 2 -f s16le out.pcm


ffplay -ar 44100 -ac 2 -f s16le -i out.pcm
```

PCM 转 WAV

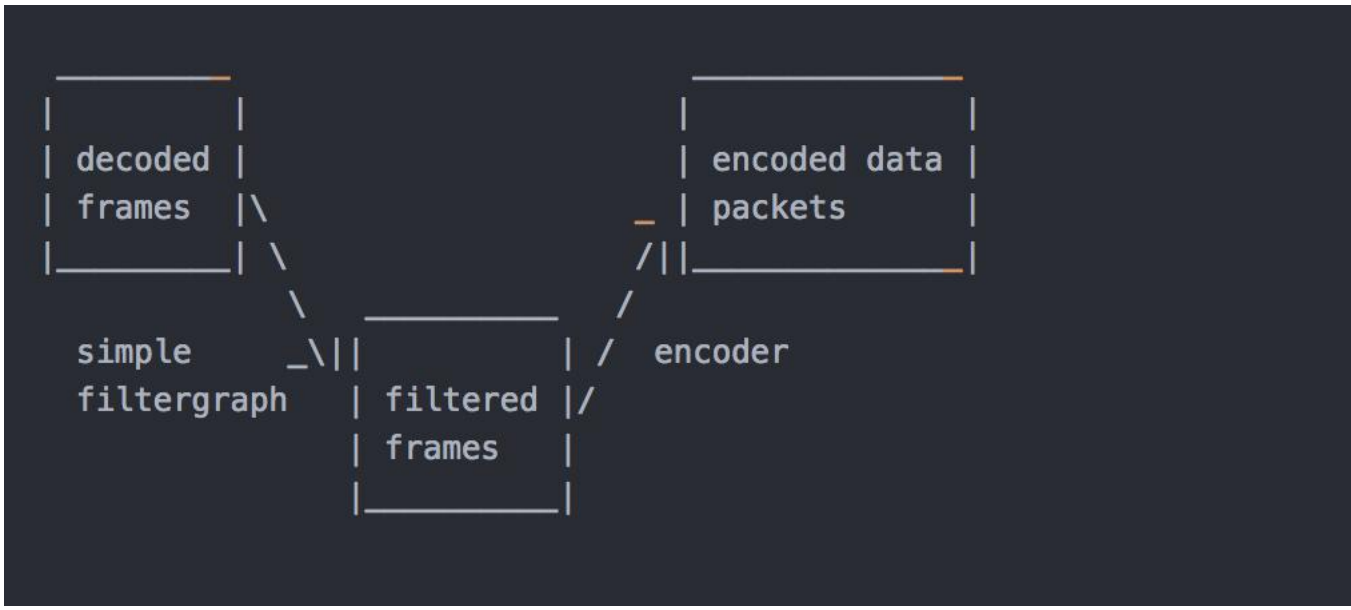
```
ffmpeg -f s16be -ar 8000 -ac 2 -acodec pcm_s16be -i input.raw output.wav
```

滤镜

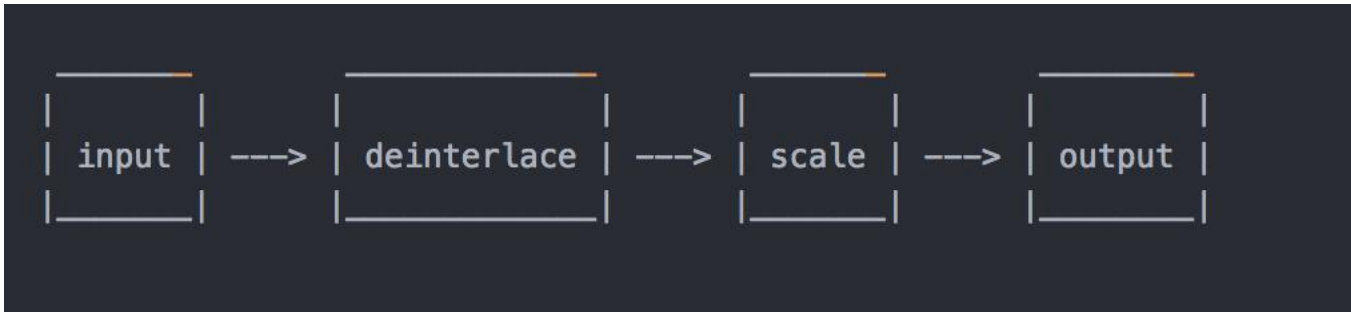
在编码之前，ffmpeg 可以使用 libavfilter 库中的过滤器处理原始音频和视频帧。几个链式过滤器形成一个过滤器图形。 ffmpeg 区分两种类型的过滤器图形：简单和复杂。

简单滤镜

简单的过滤器图是那些只有一个输入和输出，都是相同的类型。 在上面的图中，它们可以通过在解码和编码之间插入一个额外的步骤来表示：



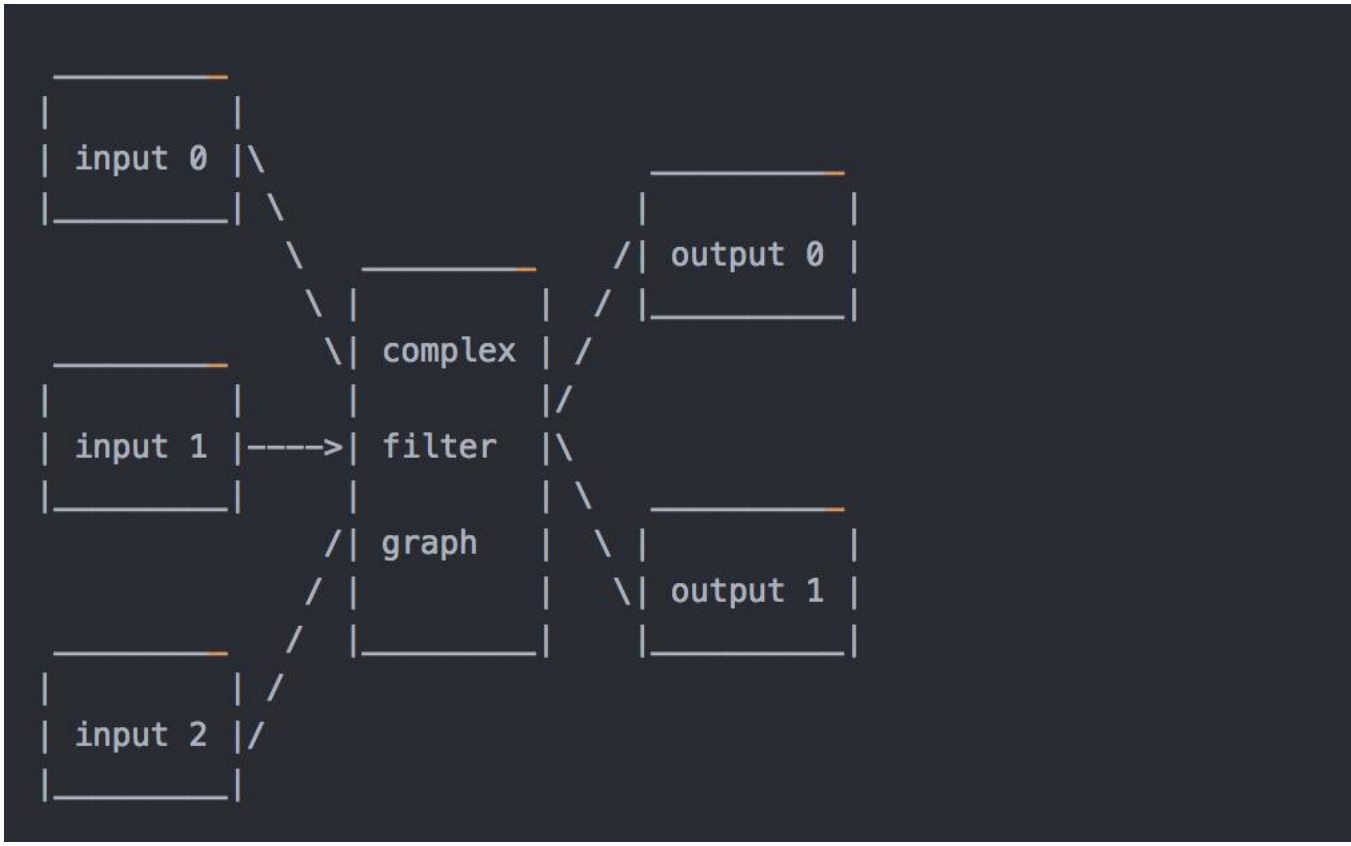
简单的 filtergraphs 配置了 per-stream-filter 选项（分别为视频和音频使用-vf 和 -af 别名）。 一个简单的视频 filtergraph 可以看起来像这样的例子：



请注意，某些滤镜会更改帧属性，但不会改变帧内容。 例如。 上例中的 fps 过滤器会改变帧数，但不会触及帧内容。 另一个例子是 setpts 过滤器，它只设置时间戳，否则不改变帧。

复杂滤镜

复杂的过滤器图是那些不能简单描述为应用于一个流的线性处理链的过滤器图。 例如，当图形有多个输入和/或输出，或者当输出流类型与输入不同时，就是这种情况。 他们可以用下图来表示：



复杂的过滤器图使用-filter_complex 选项进行配置。 请注意，此选项是全局性的，因为复杂的过滤器图形本质上不能与单个流或文件明确关联。

-lavfi 选项等同于-filter_complex。

一个复杂的过滤器图的一个简单的例子是覆盖过滤器，它有两个视频输入和一个视频输出，包含一个视频叠加在另一个上面。 它的音频对应是 amix 滤波器。

添加水印

```
ffmpeg -i out.mp4  -vf "movie=logo.png,scale=64:48[watermask];[in][watermask]

overlay=30:10 [out]" water.mp4
```

- -vf 中的 movie 指定 logo 位置。scale 指定 logo 大小。overlay 指定 logo 摆放的位置。

删除水印
先通过 ffplay 找到要删除 LOGO 的位置

```
ffplay -i test.flv -vf delogo=x=806:y=20:w=70:h=80:show=1
```

使用 delogo 滤镜删除 LOGO

```
ffmpeg -i test.flv -vf delogo=x=806:y=20:w=70:h=80 output.flv
```

视频缩小一倍

```
ffmpeg -i out.mp4 -vf scale=iw/2:-1 scale.mp4
```

- -vf scale 指定使用简单过滤器 scale，iw/2:-1 中的 iw 指定按整型取视频的宽度。 -1 表示高度随宽度一起变化。

视频裁剪

```
ffmpeg -i VR.mov  -vf crop=in_w-200:in_h-200 -c:v libx264 -c:a copy -video_size

1280x720 vr_new.mp4
```

crop 格式：crop=out_w:out_h:x:y

- out_w: 输出的宽度。可以使用 in_w 表式输入视频的宽度。
- out_h: 输出的高度。可以使用 in_h 表式输入视频的高度。
- x : X 坐标
- y : Y 坐标

如果 x 和 y 设置为 0,说明从左上角开始裁剪。如果不写是从中心点裁剪。
倍速播放

```
ffmpeg -i out.mp4 -filter_complex "[0:v]setpts=0.5*PTS[v];[0:a]atempo=2.0[a]" -map "[v]"
-map "[a]" speed2.0.mp4
```

- -filter_complex 复杂滤镜，[0:v]表示第一个（文件索引号是 0）文件的视频作为输入。setpts=0.5*PTS 表示每帧视频的 pts 时间戳都乘 0.5 ，也就是差少一半。[v]表示输出的别名。音频同理就不详述了。
- map 可用于处理复杂输出，如可以将指定的多路流输出到一个输出文件，也可以指定输出到多个文件。"[v]" 复杂滤镜输出的别名作为输出文件的一路流。上面 map 的用法是将复杂滤镜输出的视频和音频输出到指定文件中。

对称视频

```
ffmpeg  -i out.mp4 -filter_complex "[0:v]pad=w=2*iw[a];[0:v]hflip[b];[a][b]overlay=x=w"
duicheng.mp4
```

- hflip 水平翻转

如果要修改为垂直翻转可以用 vflip。

画中画

```
ffmpeg -i out.mp4 -i out1.mp4 -filter_complex
"[1:v]scale=w=176:h=144:force_original_aspect_ratio=decrease[ckout];[0:v][ckout]overlay=x
=W-w-10:y=0[out]" -map "[out]" -movflags faststart new.mp4
```

录制画中画

```
ffmpeg  -f avfoundation -i "1" -framerate 30 -f avfoundation -i "0:0"
```

```
-r 30 -c:v libx264 -preset ultrafast

-c:a libfdk_aac -profile:a aac_he_v2 -ar 44100 -ac 2

-filter_complex

"[1:v]scale=w=176:h=144:force_original_aspect_ratio=decrease[a];[0:v][a]overlay=x=W-w-

10:y=0[out]"

-map "[out]" -movflags faststart -map 1:a b.mp4
```

多路视频拼接

```
ffmpeg  -f avfoundation -i "1" -framerate 30 -f avfoundation    -i "0:0" -r 30 -c:v libx264

-preset ultrafast -c:a libfdk_aac -profile:a aac_he_v2 -ar 44100 -ac 2 -filter_complex

"[0:v]scale=320:240[a];[a]pad=640:240[b];[b][1:v]overlay=320:0[out]" -map "[out]"

-movflags faststart  -map 1:a  c.mp4
```

音视频的拼接与裁剪

裁剪

```
ffmpeg -i out.mp4 -ss 00:00:00 -t 10 out1.mp4
```

- -ss 指定裁剪的开始时间，精确到秒
- -t 被裁剪后的时长。

合并

首先创建一个 inputs.txt 文件，文件内容如下：

```
file '1.flv'

file '2.flv'

file '3.flv'
```

然后执行下面的命令：

```
ffmpeg -f concat -i inputs.txt -c copy output.flv
```

hls 切片

```
ffmpeg -i out.mp4 -c:v libx264 -c:a libfdk_aac -strict -2 -f hls  out.m3u8
```

- -strict -2 指明音频使有 AAC。
- -f hls 转成 m3u8 格式。

视频图片互转

视频转 JPEG

```
ffmpeg -i test.flv -r 1 -f image2 image-%3d.jpeg
```

视频转 gif

```
ffmpeg -i out.mp4 -ss 00:00:00 -t 10 out.gif
```

图片转视频

```
ffmpeg  -f image2 -i image-%3d.jpeg images.mp4
```

直播相关

推流

```
ffmpeg -re -i out.mp4 -c copy -f flv rtmp://server/live/streamName
```

拉流保存

```
ffmpeg -i rtmp://server/live/streamName -c copy dump.flv
```

转流

```
ffmpeg -i rtmp://server/live/originalStream -c:a copy -c:v copy -f flv
rtmp://server/live/h264Stream
```

实时推流

```
ffmpeg -framerate 15 -f avfoundation -i "1" -s 1280x720 -c:v libx264  -f  flv
rtmp://localhost:1935/live/room
```

ffplay

播放 YUV 数据

```
ffplay -pix_fmt nv12 -s 192x144 1.yuv
```

播放 YUV 中的 Y 平面

```
ffplay -pix_fmt nv21 -s 640x480 -vf extractplanes='y' 1.yuv
```