图像、视频、算法、Linux

Ideas worth spreading

目录视图 摘要视图 RSS 订阅

个人资料

arvik

关注 发私信

访问： 294768次

积分： 3313

等级： BLOG 5

排名： 第7894名

原创： 80篇 转载： 5篇

译文： 1篇 评论： 23条

最新动态

ffmpeg 官方文档 上篇 （译）

标签： ffmpeg 音视频

2016-01-14 17:31 3362人阅读 评论(0) 收藏 举报

本文章已收录于： 直播技术知识库

分类： 图像、音视频（2） ffmpeg（1）

目录(?) [+]

说明：本篇博客为翻译文档，原英文文档地址为：http://ffmpeg.org/ffmpeg.html

译者：arvik

翻译时间：2016/01/18至2016/01/19

耗时：2天

内容：音视频编解码——ffmpeg文档

由于原文较长，故分两篇博客进行翻译。水平有限，部分翻译内容可能较为晦涩，遇到翻译绕口、难懂的地方建议对比原英文文档对应的地方理解。

概要

```
ffmpeg [global_options] {[input_file_options] -i input_file} ... {[output_file_options] output_file} ...
```

简介

ffmpeg是一个非常快速的音视频转换器，它可以直接从音视频直播流中获取输入源。他还可以利用高质量的多相滤波器来转换任意采样率和动态调整视频大小。

ffmpeg读取从任意数量的输入“文件”(可常规文件、管道、网络流、抓设备,等等),由- i选项指定,并写入任意数量的输出“文件”,由普通的指定输出文件名。任何一个在命令行上不能被解释执行的选项都被认为是一个输出文件名。

原则上,每个输入或输出文件可以包含任意数量的不同类型的流(视频/音频/字幕/附件/数据)。所允许的流的数量和类型可能会受到容器格式的限制。选择哪个输入流将进入哪个输出流将被自动执行或者由 -map 选项指定。

指输入文件的选项，你必须使用他们的索引(基于0)。如第一个输入文件是0，第二个是1，等等。同样，流在一个文件被称为索引。如。2:3指第三个文件的第四个流。Also see the Stream specifiers chapter。

对于一般规则，选项被应用到下一个指定的文件中，因此，顺序非常重要，你可以在命令行上重复多次用一个选项，每次出现然后应用到下一个输入或输出文件。当然，全局选项例外！

不要混合输入和输出文件——首先指定所有输入文件，然后指定所有输出文件，也不要混合不同文件的选项。所有的选项仅适用于下一个输入文件或输出文件。

设置输出文件的视频比特率为64 kbit/s：

```
ffmpeg -i input.avi -b:v 64k -bufsize 64k output.avi
```

强制输出文件的帧率为每秒24帧：

调试经验 (5)
ffmpeg (2)
STM32 (7)
MSP430 (5)
ucos (7)
nginx (7)
opencv (0)
web (8)
小设计 (4)
SDL (0)
shell脚本 (0)
Qt (0)
MFC (1)
keil (2)
IAR (1)
ucGUI (1)

文章存档

2016年12月 (8)
2016年10月 (4)
2016年09月 (5)
2016年07月 (5)
2016年06月 (6)

展开

阅读排行

【智能路由器】基于netfilter... (17767)
【智能路由器】设备流量、网... (17659)
【智能路由器】源码追踪路由... (17177)
【智能路由器】轻量级web服... (16384)
【智能路由器】动态域名（基... (16021)
【智能路由器】开篇 (15676)
【智能路由器】视频监控 (15561)

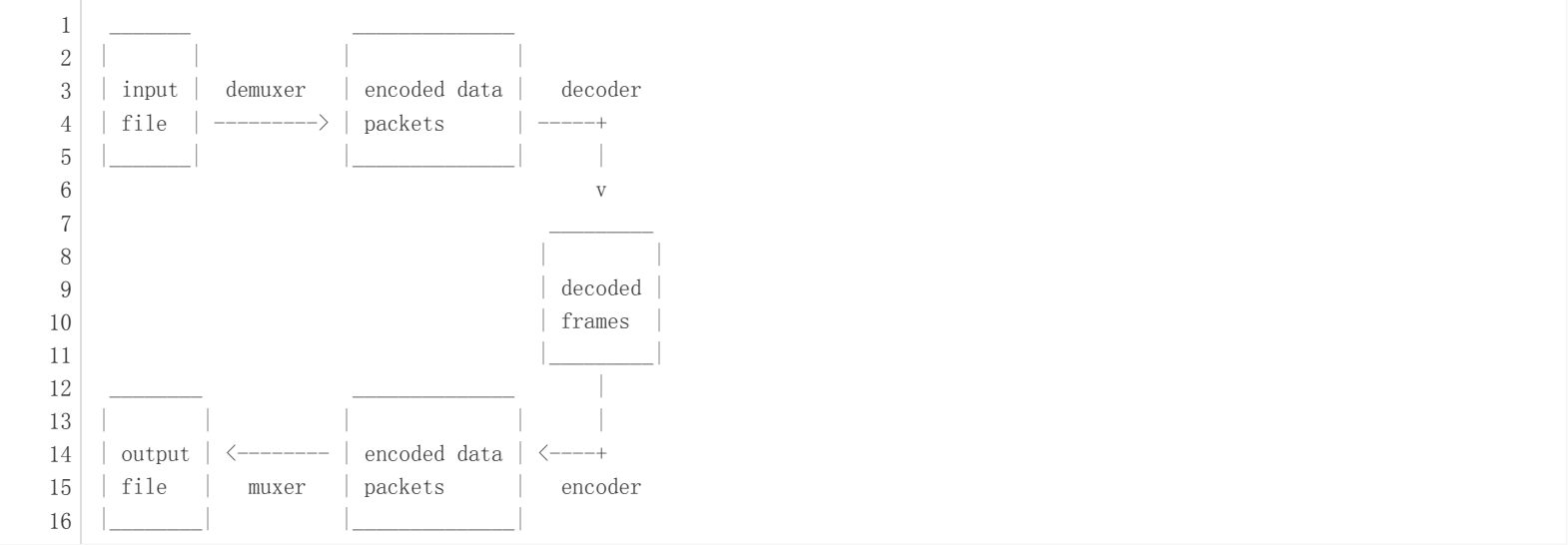
```
ffmpeg -i input.avi -r 24 output.avi
```

强制输入文件（仅对raw格式文件有效）的帧率为1帧/秒，且设置输出文件的帧率为24帧/秒：

```
ffmpeg -r 1 -i input.m2v -r 24 output.avi
```

详细描述

ffmpeg的每个输出的转化过程如下图所示：



ffmpeg调用libavformat库(包含demuxers)来读取输入文件，并从他们那里得到包含编码数据的数据包。当有多个输入文件时，ffmpeg通过在所有的有用的输入流之中跟踪最小时间戳来使它们保持同步。

编码的数据被传递到解码器（除非streamcopy被用来做进一步描述）。解码器输出解压后的数据帧（原始视频/音频PCM /...），这些数据帧可以被滤镜进一步处理（见下一步）。过滤后，这些数据帧被传递到编码器，编码器对这些帧进行编码并且输出编码后的数据包。最后，这些数据包被送给复用器（muxer），复用器将这些压缩后的帧写入到文件。（arvik注释： 复用器就是把这些压缩后的音频帧，视频帧，字幕打包起来再写入文件）

滤镜

在编码之前，ffmpeg可以使用libavfilter库里的滤镜处理原始音频和视频帧，几个链接滤镜形成一个filtergraph。ffmpeg分辨两种类型的滤镜：简单的和复杂的。

简单的filtergraphs

简单filtergraphs是那些恰好有一个输入和输出，且输入和输出类型相同。在上面的图中他们可以被表示为在简单的解码和编码之间插入一个额外的步骤:

```
 _________                    _______________
|         |                  |               |
| decoded |                  | encoded data  |
| frames  |\                _ | packets       |
|_________| \               /||_______________|
           \   __________   /
 simple     _\||          | /  encoder
 filtergraph  | filtered  |/
              | frames    |
              |___________|
```

简单filtergraphs配置了制定 -filter 选项(-vf和-af分别为视频和音频的别名)。一个视频的简单filtergraph看起来像下面这样：

```
 _______        _____________        _______        ________
|       |      |             |      |       |      |        |
| input | ---> | deinterlace | ---> | scale | ---> | output |
|_______|      |_____________|      |_______|      |________|
```

注意，一些滤镜改变帧属性而不是帧的内容，例如在上面的示例中的fps滤镜改变帧的数目，但不改变帧内容。另一个例子是setpts滤镜，它只设置时间戳，否则通过的帧就不会变化。

复杂的filtergraphs

复杂filtergraphs是那些不能被描述为应用于一帧的一个简单的线性处理链。这种情况，例如，当这个filter表有多个输入和/或输出时，或者当输出流类型不同于输入流类型时。他们可以用下图表示：

```
 _________
|         |
| input 0 |\                    _________
```

```
 _________ \                    |          |
            \   _________     /| output 0 |
             \ |         |  / |__________|
 _________    \| complex | /
|         |    |         |/
| input 1 |---->| filter  |\
|_________|    |         | \   __________
               /| graph   |  \ |          |
              / |         |   \| output 1 |
 _________   /  |_________|    |__________|
|         | /
| input 2 |/
|_________|
```

复杂的filtergraphs由-filter_complex选项进行配置。注意，这个选项是全局选项，因为复杂的filtergraph，自然不能明确地关联单个流或文件。

选项 -lavfi 相当于 -filter_complex 。

一个复杂filtergraphs的小例子就是滤镜覆盖，即两个输入文件一个输出文件，所包含的一个视频覆盖在另一个视频上面。

流复制

流复制是一个提供复制参数给-codec选项的模式选择。它使ffmpeg省略指定流的解码和编码过程，它所做的只是复用和解复用，对于改变封装格式或修改封装元数据很有用。上述中，在这种情况下，简化如下：

```
 _______              _______________           ________
|       |            |               |         |        |
| input |  demuxer   | encoded data  |  muxer  | output |
| file  | ---------> | packets       | ------> | file   |
|_______|            |_______________|         |________|
```

由于没有解码和编码过程，它非常快且没有质量损失。然而，某些情况下由于许多因素它可能不会正常工作。流复制的情况下，显然不可能用滤镜，因为滤镜是作用在解压后的数据上。

流选择

默认情况下，ffmpeg在某一时刻只包含输入文件的一种类型（视频，音频，字幕）的一个流 并且把它加到每个输出文件。他会根据条件自动为每个流挑选“最佳”方式，对于视频，它是拥有最高分辨率的流，对于音频，他是拥有最多通道数的流，对于字幕，它就是第一个字幕流。假如几个流有同样的类型和相等的码率，那么它会挑选索引值最小的那个流。

你可以使用-vn/-an/-sn选项来禁用它们。对于全手动控制来说，使用-map选项来禁用上述的默认设置。

选项

所有的数值选项，如果为指定，可以接受一个字符串作为输入，字符串末尾可能是一个单位，例如：“K”、“M”,或“G”。

如果“i”被加到国际单位制前缀，那么完整的前缀将被解释为一个二进制倍数的单位前缀，它基于1024的倍数而不是1000。附加“B”到国际单位制前缀，则会将该值乘以8。这被允许使用，例如：“KB”，“MiB”，“G”和“B”作为数字后缀。

（ps 这段感觉压根没翻译正确，原文如下： —arvik）

```
If ’i’ is appended to the SI unit prefix, the complete prefix will be interpreted as a unit prefix for binary multiples, whic
```

没有带参数布尔值选项，它的值被默认设置为真。它们可以通过在选项名字前面加前缀“no”来设置其值为假。例如用“-nofoo”将设置布尔选项名称“foo”为假。

流说明符

某些选项被应用于每帧，例如，bitrate或者codec。流说明符用来精确的指定一个选项属于哪个流。

流说明符通常是一个附加到选项名字后面的字符串，它们之间用冒号分开。如： -codec:a:1 ac3包含a:1流说明符，它匹配第二个音频流，因此，他会为第二个音频流选择ac3解码器。

一个空的流说明符将匹配所有的流。例如：-codec copy 或 -codec: copy 将复制所有没有重编码的流。

流说明符的可能形式如下：

stream_index

用这个索引匹配流，例如： -threads:1 4 将设置第二个流的 thread 为4

stream_type[:stream_index]

stream_type 是下列之一： ‘v’ 或 ‘V’ 代表 video， ‘a’ 代表 audio，’ s’ 代表 subtitle，’ d’ 代表 data，’ t’ 代表 attachments。 ‘v’ 匹配所有的视频流，, ‘V’ 只匹配不附加图片,视频缩略图或封面艺术的视频流。如果strem_index被给出，它将匹配这种类型的流对应索引（ stream_index ）的流，否则，它匹配所有这种类型的流。

p:program_id[:stream_index]

如果给出了参数 strem_index，它将在拥有 program_id 的program中匹配这些带有 strem_index 数字的的流，否则它将匹配program中的所有流。

#stream_id or i:stream_id

通过流id匹配流。（例子：MPEG-TS容器里的PID）

m:key[:value]

匹配有指定value的元标记key的流，如果给出了value，则匹配那些包含给出标记的任何value。

u

匹配可用配置的流，此时解码器必须定义并且必不可少的信息 如视频尺寸或音频采样率必须存在。

注意：在ffmpeg中，以元数据作为匹配的将只会对输入文件起作用。

通用选项

这些选项适用于所有的 ff* 工具。

-L

显示许可证

-h, -?, -help, –help [arg]

显示帮助。一个典型的参数可能用来打印指定条目的帮助信息。如果没有指定参数，则只会显示基本的（非高级）工具选项。

```
可能的选项如下：
long
打印除基本工具选项外的高级工具选项

full
打印完整的选项列表，包括编码器、解码器、复用器、解复用器、滤镜等共享或私有选项

decoder=decoder_name
打印名字为decoder_name的解码器的详细信息，使用 -decoders 选项来获取所有解码器列表。

encoder=encoder_name
打印解码器encoder_name的编码器详细信息，使用 -encoder_name 选项来获取所有编码器列表。

demuxer=demuxer_name
打印解复用器demuxer_name的详细信息，使用 -formats 选项来获取所有复用与解复用信息列表。

muxer=muxer_name
打印复用器muxer_name的详细信息，使用 -formats 选项来获取所有复用与解复用信息列表。

filter=filter_name
打印滤镜filter_name的详细信息，使用 -filters 选项来获取所有滤镜列表。
```

-version

显示软件版本

-formats

显示可识别的格式（包括设备）

-devices

显示可识别设备

-codecs

显示libavcodec里所有编解码器

-decoders

显示所有可用解码器

-encoders

显示所有可用编码器

-bsfs

显示可用比特流滤镜

-protocols

显示可用协议

-filters

显示libavfilter中可用滤镜

-pix_fmts

显示可用像素格式

-sample_fmts

Show available sample formats.

-layouts

显示通道名称和标准通道布局。

-colors

显示公认颜色名称

-sources device[,opt1=val1[,opt2=val2]...]

显示输入文件识别的数据源。有些设备可能提供系统独立的名称，这些名称不能被识别。返回的列表不能总是认为是完整的。

```
ffmpeg -sources pulse,server=192.168.0.4
```

-sinks device[,opt1=val1[,opt2=val2]...]

显示可识别的输出设备。

```
ffmpeg -sinks pulse,server=192.168.0.4
```

-loglevel [repeat+]loglevel | -v [repeat+]loglevel

使用库设置日志级别，添加 “repeat+”选项指明重复的日志输出不应被压缩到第一行且 “Last message repeated n times”这一行可能被忽略。“repeat”也可以被单独使用，如果单独使用，且没有设置日志等级，则默认日志等级被设置。如果多个日志等级被设置，则使用“repeat”将不会改变日志等级。日志等级是一个包含下列值得字符串或值：

```
‘quiet, -8’
Show nothing at all; be silent.

‘panic, 0’
Only show fatal errors which could lead the process to crash, such as and assert failure. This is not currently used for anything.

‘fatal, 8’
Only show fatal errors. These are errors after which the process absolutely cannot continue after.

‘error, 16’
Show all errors, including ones which can be recovered from.

‘warning, 24’
Show all warnings and errors. Any message related to possibly incorrect or unexpected events will be shown.

‘info, 32’
Show informative messages during processing. This is in addition to warnings and errors. This is the default value.

‘verbose, 40’
Same as info, except more verbose.

‘debug, 48’
Show everything, including debugging information.

‘trace, 56’
```

默认情况下，程序日志输出到stderr，如果终端支持着色，则颜色备用来标记错误和警告，可以通过设置环境变量AV_LOG_FORCE_NOCOLOR 或者NO_COLOR来禁止着色，或者强制设置环境变量AV_LOG_FORCE_COLOR。下一个ffmpeg版本的的环境变量NO_COLOR将被分离且被弃用。

-report

将完整的命令行和控制台输出到一个名为program-YYYYMMDD-HHMMSS的文件。登录当前目录，这个文件可以用于错误报告，它还暗含着冗长的日志。

设置环境变量FFREPORT为任何值都具有相同的效果，该值是一个被“：”分离的键值对，这些选项会影响输出。如过包含特殊的选项分隔符“：”的话这些选项必须被分离，（参见ffmpeg工具手册章节“Quoting and escaping” ）

下列选项被识别：

file

存放输出报告的文件名， %p 是扩展程序的名称。%t 是时间戳扩展，%%代表一个%。

level

使用数值设置日志的详细级别（见 -loglevel ）

例如：使用日志等级为32输出一个文件名为ffreport.log：

```
1 FFREPORT=file=ffreport.log:level=32 ffmpeg -i input output
```

环境变量解析错误并不致命，也不会出现在报告中。

-hide_banner

隐藏打印信息

FFmpeg所有工具通常会显示版权声明,构建选项和库版本。这个选项可以用来隐藏打印这个信息。

-cpuflags flags (global)

允许设置和清除CPU标志位，该选项用于测试，如果你不清楚它是用来干嘛的，请不要使用它。

```
ffmpeg -cpuflags -sse+mmx ...
ffmpeg -cpuflags mmx ...
ffmpeg -cpuflags 0 ...
```

该选项可能的值如下：

‘x86’

‘mmx’

‘mmxext’

‘sse’

‘sse2’

‘sse2slow’

‘sse3’

‘sse3slow’

‘ssse3’

‘atom’

‘sse4.1’

‘sse4.2’

‘avx’

‘avx2’

‘xop’

‘fma3’

‘fma4’

‘3dnow’

‘3dnowext’

‘bmi1’

‘bmi2’

‘cmov’

‘ARM’

‘armv5te’

‘armv6’

‘armv6t2’

‘vfp’

‘vfpv3’

‘neon’

‘setend’

‘AArch64’

‘armv8’

‘vfp’

‘neon’

‘PowerPC’

‘altivec’

‘Specific Processors’

‘pentium2’

‘pentium3’

‘pentium4’

‘k6’

‘k62’

‘athlon’

‘athlonxp’

‘k8’

-opencl_bench

这个选项用于校准所有可用OpenCL设备并打印结果，该选项仅当FFmpeg编译时配置 –enable-opencl参数后才可使用。

当ffmpeg配置 –enable-opencl时，该选项对于OpenCL上下文设置来说是全局选项，请参见“OpenCL选项”部分ffmpeg-utils手册的支持选项的完整列表。另外，这些选项包括选择一个特定的平台且有能力运行OpenCL代码的设备。默认情况下，ffmpeg将运行第一个平台上的第一个设备。用户在选择OpenCL设备时全局OpenCL上下文选项提供选择的灵活性，大多数用户可能想要选择运行最快的OpenCL设备系统。

典型的使用最快的OpenCL设备用法的步骤如下。

运行命令：

```
ffmpeg -opencl_bench
```

在最快的设备列表中记下这个平台ID(pidx)和设备ID(didx)的第一个，选择平台和设备使用命令:

```
ffmpeg -opencl_options platform_idx=pidx:device_idx=didx ...
```

-opencl_options options (global)

设置OpenCL环境选项。该选项仅当FFmpeg配置 –enable-opencl有效。

AVOptions

这些选项由libavformat,libavdevice和libavcodec库直接提供。查看可用AVOptions列表,使用- help选项。他们分为两类:

generic

这些选项可以为任何容器,编解码器或设备设置。通用选项列出了在AVFormatContext选项下的容器/设备和AVCodecContext编解码器的选项。

private

这些选项是特定于给定的容器,设备或编解码器，相应的容器/设备/编解码器有自己的私有选项。

主要选项

-f fmt (input/output)

强制输入或输出文件格式，输入文件格式通常是由ffmpeg自动检测，输出文件格式通常由文件名后缀猜测出来，所以该选项大多数情况下是不必要的。

-i filename (input)

输入文件名

-y (global)

覆盖输出文件时不必询问。

-n (global)

如果指定的输出文件已经存在，不覆盖输出文件，并立即退出。

-stream_loop number (input)

设置输入流的循环次数，loop 0 表示不循环， loop 1 表示无限循环。

-c[:stream_specifier] codec (input/output,per-stream)

-codec[:stream_specifier] codec (input/output,per-stream)

选择一个编码器(在输出文件之前)或解码器(在输入文件之前)用于一个或多个流。codec是一个解码器/编码器的名字或一个特殊的值，它指明 这个流不应该被重新解码和编码。

举个例子：

```
ffmpeg -i INPUT -map 0 -c:v libx264 -c:a copy OUTPUT
```

用 libx264 编码所有视频流且复制所有音频流

对于每个流，最后应用 c 选项匹配，因此

```
ffmpeg -i INPUT -map 0 -c copy -c:v:1 libx264 -c:a:137 libvorbis OUTPUT
```

将复制所有的流，除了第二个视频将用 libx264编码和第138音频将用libvorbis编码以外。

-t duration (input/output)

当使用时作为输入选项（在 -i 选项前），限制从输入文件读取数据的时间。

当使用时作为输出选项（在输出文件名之前），在到达 duration 时间后，停止向输出文件写数据。

时间必须是一个时间标准格式，见(ffmpeg-utils)中的时间部分ffmpeg-utils(1)手册。

-to 和 -t 相互排斥，且 -t 优先。

-to position (output)

停止向输出文件写数据的位置，position必须是一个时间标准格式，见(ffmpeg-utils)中的时间部分ffmpeg-utils(1)手册。

-to 和 -t 相互排斥，且 -t 优先

-fs limit_size (output)

设置文件大小限制，以字节表示。

-ss position (input/output)

当作为输入选项时（在 -i 之前），查询在这个输入文件位置。注意，在大多数格式是不可能查询准确，所以ffmpeg将查询最接近查询之前的位置。当 transcoding 和 -accurate_seek 开启（默认），这种额外的在寻求点和位置之间的段将解码和丢弃。当进行流复制或使用 -noaccurate_seek 时,它将被保留下来。

当用作输出选项(在输出文件名之前)，解码 但会丢弃 时间戳以前的位置 的输入数据。

-sseof position (input/output)

类似 -ss 但和 “end of file”相关。在以前的文件里是负值，0代表EOF。

-itsoffset offset (input)

设置输入时间偏移量。

offset 必须是一个时间标准格式。见(ffmpeg-utils)中的时间部分ffmpeg-utils(1)。

这个 offset 是添加到输入文件时间戳的偏移量，指定一个正的 offset 意味着相应的流将被延迟 offset 指定的标准格式的时间。

-timestamp date (output)

设置容器中的时间戳记录。

data 必须是一个日期标准格式，见ffmpeg-utils(ffmpeg-utils)日期部分(1)手册。

-metadata[:metadata_specifier] key=value (output,per-metadata)

设置一个元数据键/值对。

一个可选的metadata_specifier可能被给出用来设置元数据。详细内容见map_metadata文档。

设置 -map_metadata 将会重写源数据，也可以置空来删除一个元数据。

例如，对于设置输出文件的标题：

```
ffmpeg -i in.avi -metadata title="my title" out.flv
```

设置第一个音频流的语言：

```
ffmpeg -i INPUT -metadata:s:a:0 language=eng OUTPUT
```

-program [title=title:][program_num=program_num:]st=stream[:st=stream...] (output)

以指定的 title 、program_num 创建一个程序，并且将指定的 stream 添加到上面。

-target type (output)

指定目标文件类型(svcd vcd,dvd,dv,dv50)。类型可能与前缀 pal-, ntsc- 或 -film混合来使用相应的标准。所有的格式选项(比特率,编解码器缓冲区大小)然后自动设置。你可以敲如下命令:

```
ffmpeg -i myfile.avi -target vcd /tmp/vcd.mpg
```

不过您可以指定附加选项，只要你知道他们与标准不冲突，如:

```
1 ffmpeg -i myfile.avi -target vcd -bf 2 /tmp/vcd.mpg
```

-dframes number (output)

设置数据帧的输出数量，这是 -frames:d 的别名。

-frames[:stream_specifier] framecount (output,per-stream)

framecount帧后面的内容停止写入到流。

-q[:stream_specifier] q (output,per-stream)

-qscale[:stream_specifier] q (output,per-stream)

使用固定质量大小。q/qscale的意义就是codec-dependent。如果使用qscale不带stream_specifier，则它只应用于视频流。

-filter[:stream_specifier] filtergraph (output,per-stream)

通过指定的 filtergraph 创建滤镜表并且用来过滤流。

filtergraph是对一个应用于流的滤镜表的描述，这个流必须有单个输入和单个输出且类型相同。 在滤镜表中，输入和

输入标签相对应，输出和输出标签相对应，有关更多信息,请参见ffmpeg-filters手册关于filtergraph语法。

如果你想在多个输入和输出上创建滤镜表，参看 -filter_complex 选项。

-filter_script[:stream_specifier] filename (output,per-stream)

这个选项类似于 -filter，唯一的区别是它的参数名称，是一个从滤镜表描述中读出的文件名。

-pre[:stream_specifier] preset_name (output,per-stream)

为匹配流指定预设值。

-stats (global)

打印编码进度/统计数据。在默认情况下,明确你需要设定-nostats禁用它。

-progress url (global)

发送 -progress url 进程信息到url。

-stdin

使交互标准输入。在默认情况下,除非使用标准输入作为输入，否则您需要指定-nostdin显式禁用交互。

-debug_ts (global)

打印时间戳信息。它在默认情况下是关闭的。这个选项主要是用于测试和调试的目的，且输出格式可能会从一个版本到另一个版本发生变化，所以它不应该采用便携式脚本。

参见选择-fdebug ts。

-attach filename (output)

添加一个附件到输出文件，这被少数几个格式支持，像用于呈现字幕的Matroska 。附件是实现为一个特定类型的流,所以这个选项将添加一个新的流到文件，然后可以以常规方式来定制这个流选项，以这个选项创建的附加流将会在其他流之后创建。（例如带有 -map 或者 automatic mappings ）

注意，对于Matroska您还必须设置mimetype元数据标签:

```
ffmpeg -i INPUT -attach DejaVuSans.ttf -metadata:s:2 mimetype=application/x-truetype-font out.mkv
```

假设附件流是第三输出文件。

-dump_attachment[:stream_specifier] filename (input,per-stream)

从filename为名的文件中提取附加流，如果文件名是空的，那么文件名的值将使用元数据标记。

如提取第一个附件文件名为“out.ttf”：

```
ffmpeg -dump_attachment:t:0 out.ttf -i INPUT
```

提取所有由标签filename决定的附件:

```
ffmpeg -dump_attachment:t "" -i INPUT
```

注意：附件被实现为编解码器的额外数据，故该选项实际可以被用来实现任何流的额外数据，而不仅仅是附件。

-noautorotate

基于文件的元数据禁用自动旋转视频功能。

上篇至此结束，下篇接着翻译 Video Options

- 上一篇 新手初次使用Git实战经历
- 下一篇 【智能路由器】新手openwrt平台搭建

我的同类文章

图像、音视频（2） ffmpeg（1）

- 【ffmpeg】常用结构体集合 2015-12-16 阅读 1799
- JPEG数据格式分析 2015-06-18 阅读 3569

猜你在找

iOS8开发技术（Swift版）：音频…
apiDoc自动化文档同步工具
iOS8开发视频教程Swift语言版-P…
JSP分页标签—DisplayTag实战视…
从三星官方内核开始移植-uboot…
cocos2d-iPhone官方文档泰然网译
java中的反射 20 类@译自Oracle…
译TestNG官方文档中文版04 运行…
Docker安全性官方文档译
译TestNG官方文档中文版12 Test…

查看评论

暂无评论

发表评论

用 户 名： NB_vol_1

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN Spark ERP
IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache .NET API
HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE
Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular
Cloud Foundry Redis Scala Django Bootstrap