ffmpeg的中文文档

Posted on 13 Feb 2015 ffmpegDocuments 添加评论

ffmpeg的文档，把之前三个文章直接合并而成的。没有新的内容和改动。

1. 概要

```
ffmpeg [global_options] {[input_file_options] -i INPUT_FILE} ...
{[output_file_options] OUTPUT_FILE} ...
```

2. 说明

ffmpeg 是一个非常快的视频和音频转换器，还可以抓取实时的音频/视频流。它可以在任意的采样率之间的转换和调整视频，并同时使用高品质的多相滤波器。

ffmpeg 从输入“文件”（其可以是常规文件，管道，网络流，录制装置等），由指定任意数量的读取 **-i** 选项，并写入到任意数量的输出“文件”，只需指定

一个输出的文件名。任何一个命令行中不能被解释为选项的内容都被认为是一个输出文件名。

每个输入或输出文件可以在原则上，包含任意数量的不同类型 频/字幕/附件/数据）的流。输出文件中允许流的数量和类型是由输出格 则决定的。输入流和输出流直接的映射可以自动完成也可以用 `-map` 选项给定（见流选择章节）。

引用输入文件的选项时，则必须使用他们的索引（从0开始）。例如：第一输入文件是 `0` ，第二个是 `1` 等。类似地，一个文件中的流也通过其索引指定。例如 `2:3` 指的是在第三个输入文件中的第四数据流。参见流章节。

作为一般规则，选项作用于下一个指定的文件。因此，命令的顺序是重要，你可以在命令行上多次相同的选项。每次选项的出现都将作用于下一个输入或输出文件。这条规则若有例外将会提前声明（例如冗余级别）。

不要混合输入和输出文件。首先指定所有输入文件，那么所有的输出文件。也不要混用属于不同的文件的选项。所有选项仅适用于下一个输入或输出文件，之后选项将被重置。

设置输出文件以64千比特/秒的视频比特率：

```
ffmpeg -i input.avi -b:V 64K -bufsize 64K output.avi
```

要强制输出文件为24 fps的帧速率：

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

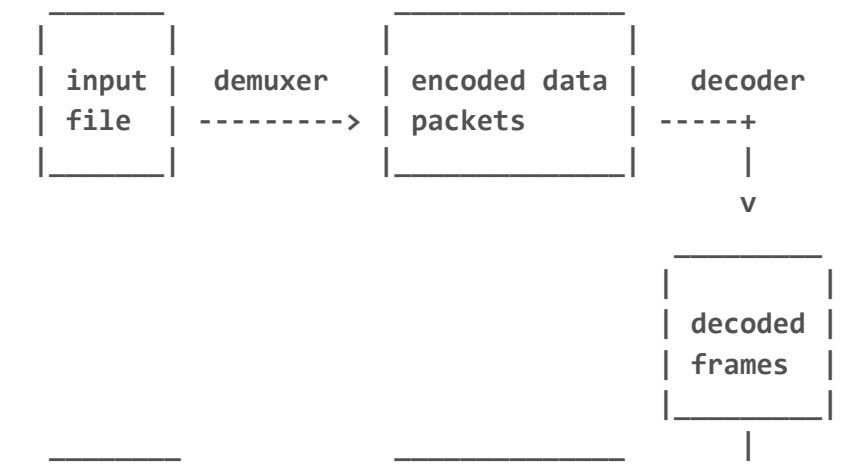
要强制输入文件的帧频（仅对原始格式有效），以1 FPS读入文件，以每秒24帧的帧速率输出：

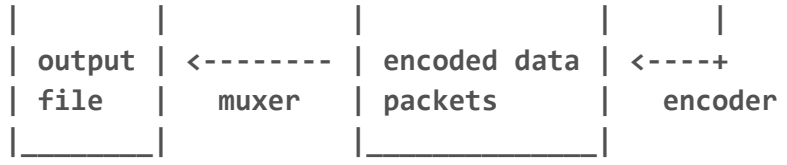
```
ffmpeg -r 1 -i input.m2v -r 24 output.avi
```

`format` 选项可能需要指定，对于原始输入文件。

3. 详细描述

在转码过程ffmpeg每个输出可以由以下图描述：





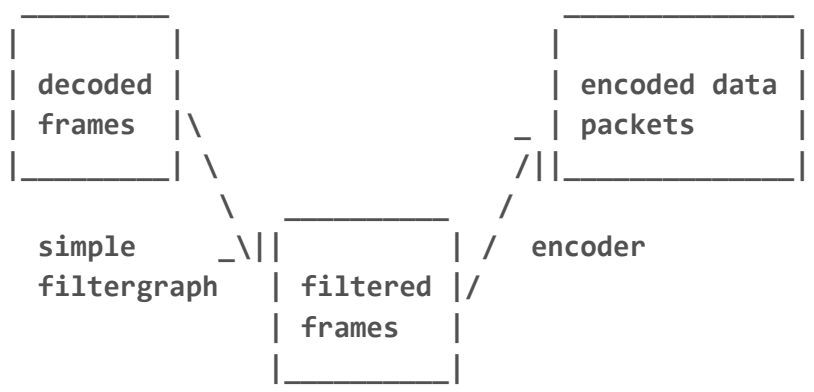
`ffmpeg` 调用libavformat库（含分流器）来读取输入文件并获得含有他们编码信息的数据包。当有多个输入文件， `ffmpeg` 将通过跟踪最小的时间戳来试图在所有活跃的输入流间同步。编码的数据包然后被传递到解码器（除非复制音频流被选择用于流，见进一步的说明）。解码器产生的未压缩的帧（原始视频/ PCM音频/ ...），它可以进一步通过滤镜进行处理（见下一节）。通过滤镜后，这些帧被传递到编码器，编码器将其编码并输出编码后的数据包。最后，这些将被传输给混合器以将编码数据写入到输出文件。

3.1 滤镜

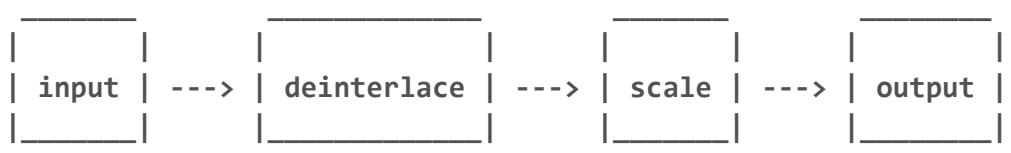
在编码之前， `ffmpeg` 可以使用libavfilter库中的滤镜处理原始的音频和视频帧。几个连接的滤镜可以形成一个滤镜组（filtergraphs）。 `ffmpeg` 有两种filtergraphs：简单和复杂。

3.1.1 简单filtergraphs

简单filtergraphs是那些具有相同的类型且正好一个输入和输出的滤镜组。另外，在上图中，他们可以由简单地在解码和编码之间插入附加步骤来表示：



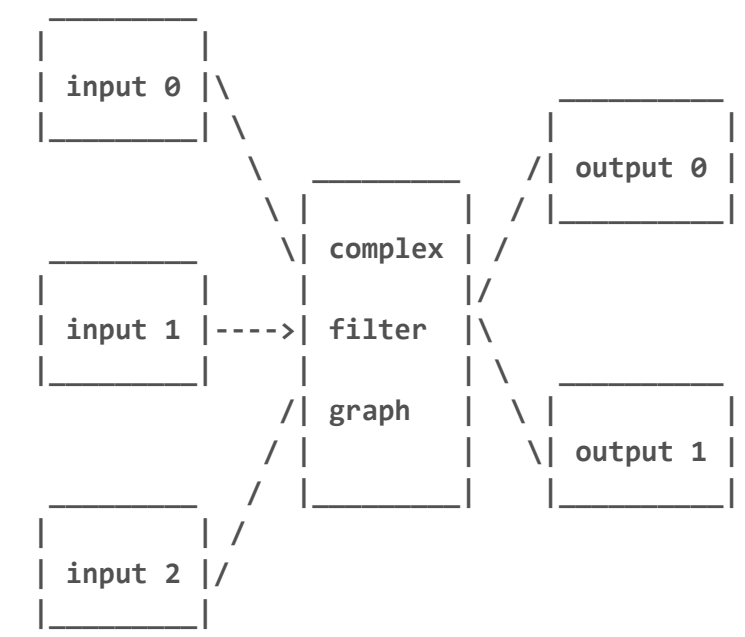
简单filtergraphs配置了每个流的筛选器选项（与视频和音频分别-vf和-af别名）。一个简单的FilterGraph动态视频可以看一下这样的例子：



需要注意的是一些滤镜改变帧属性而不是画面的内容。例如，在上例中， `fps` 改变帧的数量，但不触及帧的内容。又如 `setpts` 滤镜，其仅设置时间戳而保持帧不变。

3.1.2 复杂filtergraphs

复杂filtergraphs是那些不能被描述为简单的线性处理链的滤镜组。例如，当滤镜组具有多个输入和/或输出，或当输出流的类型是不同于输入。它们可以被表示为以下图：



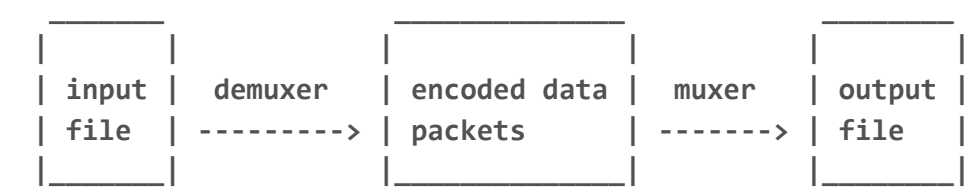
复杂filtergraphs可使用 `-filter_complex` 选项配置。注意，此选项是全局性的，因为复杂FilterGraph，就其本质，不能明确地与单个流或文件相关联。

`-lavfi` 选项相当于 `-filter_complex` 。

一个复杂FilterGraph动态的简单的例子是在 `overlay` 滤镜，它具有两个视频输入和一个视频输出，含有一个视频重叠在另一个的上面。其对应的音频滤镜是 `amix` 。

3.2 复制流

复制流是通过添加 `copy` 选项到 `-codec` 选项完成的。它使 `ffmpeg` 对指定的流忽略解码和编码步骤，所以它只能混合和拆包。它用于改变所述容器的格式或修改容器级别的元数据是有用的。在这种情况下，可以简化为这样：



由于不存在解码或编码，它是非常快，没有质量损失。然而因为许多因素的工作，它可能无法在某些情况下使用。应用滤镜显然也是不可能的，因为滤镜仅能作用在未压缩的数据上。

4 选择流

默认情况下， `ffmpeg` 只包含输入文件中每个类型流各一个（视频，音频，字

幕），并将它们添加到每个输出文件。它选择“最好”的每一个流基于以下标准：用于视频，它选择最高分辨率的流；对于音频，它使用最多声道的流；对于字幕，它是第一个字幕流。在相同类型中，参数相等的若干流中具有最低索引的流被选择。

您可以通过 `-vn/-an/-sn` 选项禁用其中的一些默认值。若需全手动控制，请使用-map选项，它将禁用刚才所描述的默认设置。

5 选项

所有的数值选项中，如果不另外指明，均表示接受数作为输入，其后可添加一个SI单位的字符串，例如： `K` ， `M` ，或 `G` 。

如果 `i` 被附加在SI单位，完整的前缀将被解释为一个单元前缀的二进制倍数，也即1024倍，而不是1000倍。追加 `B` 可使数值增加8倍。这允许使用，例如： `KB` ， `MiB` ， `G` 和 `B` 的数量后缀。

选项不带参数是布尔选项，并设置相应的值设置为 `true` 。他们可以通过在选项前添加 `no` 来将选项设置为 `false` 。例如使用 `-nofoo` 将设置名称为 `foo` 为假。

5.1 流标识符

有些选项是按流的，例如比特率或编解码器。流标识符被用来精确地指定一个给定的选项作用于哪一个数据流（多个）。

一个标识符一般是选项名称加冒号分隔的字符串。例如 `-codec:a:1 ac3` 包含流标识符 `a:1` ，它匹配第二音频流。因此，将选择AC3编解码器的第二音频流。

一个标识符可以匹配多个流，这个选项将适用于所有流。比如，流标识符 `-b:a 128k` 标识了所有的音频流。

空标识符匹配所有的流。例如， `-codec copy` 或 `-codec: copy` 会复制所有的数据流而不重新编码。

流标识符的可能形式有：

`stream_index`

匹配与该索引对应的流。例如 `-threads:1 4` 将设置第二个流的线程计数为4。

`stream_type[:stream_index]`

流类型是下列字母之一： `v` 为视频， `a` 为声音， `s` 为字幕， `d` 为数据， `t` 为附件。如果 `stream_index` 给出，则它匹配该类型的索引为 `stream_index` 的流。否则，它匹配所有这种类型的流。

`p:program_id[:stream_index]`

如果给定 `stream_index` ，那其将与在与ID为 `program_id` 的program的 `stream_index` 的流相匹配。否则，它将匹配在 `program_id` 中的所有流。

`#stream_id` 或者 `i:stream_id`

按流索引逐一匹配流（如在MPEG-TS容器中的PID）。

`m:key[:value]`

匹配流的元数据标签中具有指定 `key` 的流。如果 `value` 没有给出，将匹配包含给定标签的所有流。

请注意，在 `ffmpeg` 中，按元数据匹配仅能用于输入文件。

5.2 通用选项

这些选项当中的FF *工具共享。

`-L`

显示许可证。

`-h, -?, -help, --help [arg]`

显示帮助。一个可选参数可以被指定为打印与特定项目相关的帮助。如果没有指定参数，则只显示基本的（非高级）工具选项。

`arg` 的可能值是：

`long`
除了基本的工具选项外，打印高级的工具选项。

`full`
打印所有选项，包括编码器，解码器，分流器，混合器，滤镜等的共享和私有选项。

`decoder=decoder_name`
打印有关的解码器`decoder_name`的详细信息。使用`-decoders`选项来获得所有的解码器的列表。

`encoder=encoder_name`
打印有关指定编码器`encoder_name`的详细信息。使用`-encoders`选项来获得所有编码器的列表。

`demuxer=demuxer_name`
打印有关的分流器`demuxer_name`的详细信息。使用`-formats`选项来获取所有分流器和混合器的列表。

`muxer=muxer_name`
打印有关混合器`muxer_name`的详细信息。使用`-formats`选项来获取所有混合器和分流器的列表。

`filter=filter_name`
打印有关滤镜`filter_name`的详细信息。使用`-filters`选项来获得所有滤镜的列表。

-version

显示的版本。

-formats

显示可用的格式（包括设备）。

-devices

显示可用的设备。

-codecs

显示libavcodec已知的所有编解码器。

注意，整个文档中术语“解码器”更正确地称呼是比特流媒体格式（media bitstream format）。

-decoders

显示可用的解码器。

-encoders

显示所有可用的编码器。

-bsfs

显示可用的流滤镜。

-protocols

显示可用的协议。

-filters

显示可用的libavfilter滤镜。

-pix_fmts

显示可用的像素格式。

-sample_fmts

显示可用的采样格式。

-layouts

显示频道名称和标准的渠道布局。

-colors

显示公认的颜色名称。

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

显示自动检测到的输入设备的源。某些设备可提供不能自动检测系统相关的源名称。返回的列表不能被假定为总是完整的。

`ffmpeg -sources pulse,server=192.168.0.4`

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

自动检测显示输出设备的接收器。某些设备可提供不能自动检测系统相关的接收器名称。返回的列表不能被假定为总是完整的。

`ffmpeg -sinks pulse,server=192.168.0.4`

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

设置库使用的日志记录级别。加入 **repeat+** 表示重复日志输出不应该被压缩到所述第一条日志和“最后的日志重复n次”线将被省略。 **repeat** ，也可以单独使用。如果 **repeat** 可以单独使用，并没有预设的记录级别，默认记录级将被使用。如果给定多个日志级别参数，使用 **repeat** 不会改变日志级别。 **loglevel** 是一个字符串或数字，可为以下值之一：

`quiet, -8`
保持沉默。

`panic, 0`
只显示可能导致程序崩溃的致命错误。目前没有此类错误。

`fatal, 8`
只显示致命错误。这些错误会导致进程绝对无法继续。

`error, 16`
显示所有的错误，包括那些可以修复的。

`warning, 24`
显示所有警告和错误。将显示任何有关可能不正确或不正常事件的信息。

`info, 32`
显示处理过程中的信息。不单单是警告和错误。这是默认值。

`verbose, 40`
与**info**类似 ，但更详细。

`debug, 48`
显示一切信息，包括调试信息。

默认情况下，程序日志输出到标准错误流，如果终端支持着色，颜色用来标记错误和警告。日志着色可以被环境变量**AV_LOG_FORCE_NOCOLOR**或**NO_COLOR** ，或者可以被强制设置环境变量**AV_LOG_FORCE_COLOR** 禁用。使用环境变量**NO_COLOR**已被弃用，并在之后的**FFmpeg**的版本将被丢弃。

-report

转储完整的命令行和控制台输出到当前目录一个文件名 为 **program - YYYYMMDD - HHMMSS .log** 的文件。此文件对于错误报告非常有用。这也意味着 **-loglevel verbose** 。

将环境变量设置 **FFREPORT** 为任何值具有相同的效果。如果该值是一个’ :’ -分隔键=值序列，这些选项会影响报表;如果包含特殊字符则需要使用转义字符，或者 “:” 分隔（参见的 **ffmpeg-utils** 的手册中的 “引用与转义” 一节）。

下列选项也可使用：

`file`
设置报告使用的文件名**; %p**添加程序名， **%t**添加时间戳， **%%**添加一个普通的**%**

`level`
设置使用的数值（查看日志详细级别**`-loglevel`** ）。

例如，要输出到名为 **ffreport.log** 使用的一个日志级别文件的报告32（日志级别info的别称 ）：

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

非致命的环境变量的解析错误不会出现在报告中。

-hide_banner

不打印横幅。

所有FFmpeg的工具通常会显示一个版权声明，构建选项和库版本。此选项可以用来抑制打印此信息。

-cpuflags flags (global)

允许设置和清除CPU标志。此选项用于测试。不要使用它，除非你知道自己在做什么。

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

可能选项有：

```
`x86`
        ‘mmx’
        ‘mmxext’
        ‘sse’
        ‘sse2’
        ‘sse2slow’
        ‘sse3’
        “sse3slow”
        ‘ssse3’
        ‘atom’
        ‘sse4.1’
        ‘sse4.2’
        ‘avx’
        ‘xop’
        ‘fma4’
        ‘3dnow’
        ‘3dnowext’
        ‘cmov’
`ARM`
        ‘armv5te’
        ‘armv6’
        ‘armv6t2’
        ‘vfp’
        ‘vfpv3’
        ‘neon’
        ‘PowerPC’
        ‘altivec’
`Specific Processors`
        ‘pentium2’
        ‘pentium3’
        ‘pentium4’
        ‘k6’
        ‘k62’
        ‘athlon’
        ‘athlonxp’
        ‘k8’
```

-opencl_bench

测试所有可用的OpenCL设备并显示结果。此选项仅当FFmpeg含有–enable-opencl 编译时可用。

-opencl_options options (global)

设置的OpenCL环境选项。此选项仅当FFmpeg的已编译–enable-opencl 。

options必须是冒号分隔的key = value选项对。参见ffmpeg-utils的手册中的“OpenCL的选项”部分的内容。

5.3 AVOption选项

这些选项直接由 **libavformat** ， **libavdevice** 和 **libavcodec** 库提供。要查看可用AVOption的列表，请使用 **-help** 选项。它们被分为两类：

generic 这些选项可以为任何容器，编解码器或设备进行设置。通用的选项都列在AVFormatContext选择容器/设备和AVCodecContext选择编解码器小节。

private 这些选项是作用于给定的容器，装置或编解码器。私有选项列在其相应的容器/设备/编解码器。

比如编辑ID3v2.3头而不是默认的ID3v2.4到MP3文件，使用MP3混合器的 **id3v2_version** 私有选项：

```
ffmpeg -i input.flac -id3v2_version 3 out.mp3
```

所有编解码器AVOption选项是按流指定的，并且因此应该指定相应的流标识符。

注： **-nooption** 语法不能用于布尔AVOption选项，请使用 **-option 0/-option 1** 。

注：老的前缀v/a/s的流标识记号已经过时，将被移除。

5.4 主要选项

-f fmt (input/output)

强制指定输入或输出的文件格式。输入文件的格式通常是自动检测的，输出文件的格式由该文件的扩展名猜测，所以在大多数情况下不需要此选项。

-i filename (input)

输入的文件名

-y (global)

直接覆盖输出文件。

-n (global)

如果指定的输出文件已经存在，不要覆盖输出文件，并立即退出。

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

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

用于对一个或多个数据流指定一个编码器（一个输出文件之前使用时）或一个解码器（一个输入文件之前使用时）。 codec是一个解码器/编码器的名称或特殊值 copy （仅输出），copy表示该流不是被重新编码。

例如

`ffmpeg -i INPUT -map 0 -c:v libx264 -c:a copy OUTPUT`
使用**libx264**编码所有视频流并拷贝所有音频流。

对于每个数据流，最后匹配的 **c** 选项被应用，所以

`ffmpeg -i INPUT -map 0 -c copy -c:v:1 libx264 -c:a:137 libvorbis OUTPUT`
将复制的所有流除了第二视频流，并将用**libx264**进行编码，并且对于第**138**个音频流用**libvorbis**进行编码。

-t duration (input/output)

当作为输入选项（在 **-i** 之前 ），限制从输入文件中读取数据的 **duration** 。

当作为输出选项（输出文件名之前）使用，在达到 **duration** 后停止写入输出文件。

duration 可能是以秒为单位，或以 **hh:mm:ss[.xxx]** 形式出现。

-to 和 **-t** 是相互排斥的， **-t** 具有优先权。

-to position (output)

在 **position** 位置停止输出。 **position** 可能是一个表示秒数的数，或 **hh:mm:ss[.xxx]** 形式。

-to 和 **-t** 是相互排斥的， **-t** 具有优先权。

-fs limit_size (output)

设置文件大小限制，以字节表示。

-ss position (input/output)

当用作输入选项一起使用（在 **-i** 以前），跳转到输入文件中 **position** 位置。请注意，在大多数的格式是不可能确切定位，这样ffmpeg将寻求最接近的 **position** 位置点。当转码和 **-accurate_seek** 启用（默认设置），寻找点和 **position** 位置之间的附加段将被解码并丢弃。当进行流复制或当 **-noaccurate_seek** 被使用时，它都将被保留。

当用作输出选项（在输出文件名前），解码但丢弃输入直到时间戳到达的位置。

position 位置可以是秒数或hh:mm:ss[.xxx]形式。

-itsoffset offset (input)

设置输入时间偏移。

offset 必须是持续时间规范，请参阅（ffmpeg-utils）在FFmpeg-utils（1）手动的持续时间段的相关内容 。

偏移被添加到输入文件的时间戳。指定一个正偏移意味着相应流将延迟offset所指定的时间。

-timestamp date (output)

设置在容器内记录的时间戳。

date必须是一个规范的持续时间，请参阅（ffmpeg-utils）在FFmpeg-utils的（1）日期部分 。

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

设置元数据的键/值对。

可选 **metadata_specifier** 可以被用于设置流或章节的元数据。见 **-map_metadata** 文档的详细信息。

此选项将覆盖 **-map_metadata** 设置的元数据。另外，也可以通过使用空值来删除元数据。

例如，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`

-target type (output)

指定目标文件类型（ vcd ， svcd ， dvd ， dv ， dv50 ）。 类型可能与前缀 **pal-** ， **ntsc-** 或 **film-** 使用相应的标准。所有的格式选项（比特率，编解码器，缓冲大小）都将自动设置。你仅需键入：

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

不过，你可以指定其他选项，只需你知道他们与标准不冲突，如：

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

-dframes number (output)

设置输出数据的帧数。这是 **-frames:d** 的别名。

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

在 **framecount** 帧后停止写入流。

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

使用固定编码率（VBR）。 **q/qscale** 的意思与编解码器定义相关。如果 **qscale** 不与 **stream_specifier** 联用，那么它仅适用于视频流，这是为了保持兼容性。另外将相同的编码器参数赋给两个不同的编解码器通常并不是用户想要

的，因此若需要这样的功能，可以使用流标识符(**stream_specifier**)来指定。

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

创建由FilterGraph指定的滤镜组并使用它。

FilterGraph是作用于流的滤镜组的描述，而且必须有一个单一的输入和同一类型的数据流输出。

在滤镜组里，输入被关联到 **in** 标签 ，输出到 **out** 标签 。关于ffmpeg滤镜组的语法可参见ffmpeg-filters的手册。

如果你想创建具有多个输入或输出的滤镜组，参见 **-filter_complex** 的相关选项。

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

这个选项类似于 **-filter** ，唯一的区别是，它的参数是滤镜组所在的文件名。

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

指定匹配流（S）的预设。

-stats (global)

打印编码进度/统计数据。这是默认值，你可通过指定 **-nostats** 禁用 。

-progress url (global)

发送程序友好的进展信息到 **url**

进度信息大约每秒和编码过程结束后写入。它是由 **key = value** 行组成。 key只能包含字母和数字字符。最后一个关键字序列始终是 **progress** 。

-stdin

启用标准输入交互。这是默认设置，除非标准输入被作为输入。要明确禁用互动，你需要指定 **-nostdin** 。

在标准输入禁用相互作用是很有用的，例如，如果ffmpeg在后台进程组中。大致相同的结果可以用 **ffmpeg ... < /dev/null** 实现，但它需要一个终端。

-debug_ts (global)

打印时间戳信息。它默认是关闭的。此选项主要是用于测试和调试目的，输出格式可从一个版本切换到另一个，所以它不应该在可移植脚本中使用。

参见 **-fdebug ts** 选项。

-attach filename (output)

添加一个附件到输出文件。这仅由几个格式支持。例如Matroska格式，这个文件可以是用来渲染字幕使用的字体。附件被实现为数据流的一个特定类型的，因此该

选项将增加一个新的流的文件。因此，可以以通常的方式在此流使用每个流的选项。使用此选项创建的附件流将所有的其他流之后创建（也即那些由 **-map** 或自动映射创建的流）。

请注意，对于Matroska，你还必须设置 **mimetype** 元数据标签：

`ffmpeg -i INPUT -attach DejaVuSans.ttf -metadata:s:2 mimetype=application/x-truetype-font out.mkv`
（假设该附件流将是输出文件的第三个流）。

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

提取匹配的附件流成一个名为 **filename** 的文件。如果 **filename** 是空的，那么元数据标签中的 **filename** 标签的值将被使用。

例如，提取第一附件到**“out.ttf”**：
`ffmpeg -dump_attachment:t:0 out.ttf -i INPUT`

提取所有附件，并按照**filename** 标签命名文件：
`ffmpeg -dump_attachment:t "" -i INPUT`

技术说明 - 附件是通过编解码器额外数据实现的，所以这个选项实际上可以用来提取任意额外数据，而不仅仅是附件。

5.5 视频选项

-vframes number (output)

设置视频帧的输出数量。这是的 **-frames:v** 别名 。

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

设置帧速率（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，而是产生时间戳假设恒定的帧速率fps这与一些输入格式，诸如image2或者v4l2，中的 **-framerate** 选项是不同的（在老版本的ffmpeg中是一样的）。如有疑问，使用的输入选项 **-framerate** 代替 **-r** 。

作为输出选项，复制或删除输入帧以达到恒定的输出帧速率fps。

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

设置帧尺寸。

作为输入的选项，这是私有选项 **video_size** 的快捷方式，部分分流器可以识别该参数，这时帧大小或者未存储在文件中或不可配置,例如原始视频或视频采集卡。

作为输出的选择，这将插入 **scale** 视频滤镜到相应滤镜组的末端 。如需改变滤镜位置，请将 **scale** 滤镜直接插入到开头或其他地方。

格式是 **wxh** （默认值是使用与源相同的尺寸）。

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

设置指定视频显示的宽高比 **aspect** 。

aspect 可以是一个浮点数，或使用形如 **num: den** 的形式，其中 **num** 是分子， **den** 是分母。例如， “4:3” ， “16:9” ， “1.3333” ，和 “1.7777” 都是有效的参数值。

如果与 **-vcodec copy** 一起使用，这会影响储存在容器级别的宽高比，而不是存储在编码帧的宽高比，如果有的话。

-vn (output)

禁用视频录制。

-vcodec codec (output)

设置视频编解码器。这是-codec:v的别名。

-pass[:stream_specifier] n (output,per-stream)

选择通过次数（1或2）。它是用来做两遍视频编码。视频的统计信息记录在第一次编码时写入到日志文件中（也参见选项 **-passlogfile** ），在第二次编码时，该日志文件被用于按要求生成准确比特率的视频。在第一次编码中，你可以通过禁用音频并将输出设置为null，下面为Windows和Unix的例子：

```
`ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y NU
L`
`ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y /d
ev/null`
```

-passlogfile[:stream_specifier] prefix (output,per-stream)

设置二次编码的日志文件名 前缀为 **prefix** ，默认的文件名 前缀是 **ffmpeg2pass** 。完整的文件名 将是 **PREFIX-N.log** ，其中 **N** 是一个输出流的特定数字

-vf filtergraph (output)

创建由 **filtergraph** 指定的滤镜组，并使用它。

这是 **-filter:v** 的别名 ，参见 **-filter** 选项。

5.6 高级视频选项

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

设置的像素格式。使用 **-pix_fmts** 显示所有支持的像素格式。如果所选择的像素格式不能被选择，ffmpeg将打印警告，并选择由编码器所支持的最好的像素格式。如果 **pix_fmt** 前缀 **+** ，如果请求的像素格式不能被选中，ffmpeg会出现错误退出，同时滤镜组中的自动转换将被禁用。如果 **pix_fmt** 是一个单一的 **+** ，ffmpeg的选择与输入或者滤镜输出相同的像素格式，并将禁用自动转换。

-sws_flags flags (input/output)

设置软件缩放的标志。

-vdt n

丢弃的阈值。

-rc_override[:stream_specifier] override (output,per-stream)

覆盖特定的时间间隔内的帧率控制，格式为用斜杠分隔的“整型，整型，整型”列表。前两个值是在开始和结束帧编号，最后一个如果是正数，则为量化器；负数则为品质因数。

-ilme

支持编码器force interlacing（仅对MPEG-2和MPEG-4有效）。如果你的输入文件是隔行并要保持隔行格式最小损失，请使用此选项。另一种方法是使用 **-deinterlace** 反交错输入流，但这会引入损失。

-psnr

计算压缩帧的PSNR（伪信噪比）。

-vstats

输出视频编码统计到 **vstats_HHMMSS.log** 。

-vstats_file file

输出频编码统计到 **file** 。

-top[:stream_specifier] n (output,per-stream)

top=1/bottom=0/auto=-1 field first

-dc precision

Intra_dc_precision.

-vtag fourcc/tag (output)

强制视频 tag/ fourcc。这是 **-tag:v** 的别名。

-qphist (global)

显示QP直方图

-vbsf bitstream_filter

已抛弃，见 **-bsf**

-force_key_frames[:stream_specifier] time[,time...] (output,per-stream)

-force_key_frames[:stream_specifier] expr:expr (output,per-stream)

强制关键帧在指定的时间戳，更精确地在每个指定的时间之后的第一帧。

如果参数的前缀 **expr:** ， **expr** 将解释为一个表达式，并在每一帧执行。如果运行结果非零，一个关键帧被强制加入。

如果时间之一是 **chapters [delta]** ，它被扩展成的文件中的所有章节开始通过 **delta** 偏移的时间（时间以秒为单位）。这个选项可能是有用的，以确保存在一个搜索点位于章节标记或者在输出文件中的任何其它指定的地方。

例如，在5分钟插入一个关键帧，同时在每章开始前0.1秒插入一个关键帧：

```
`-force_key_frames 0:05:00,chapters-0.1`
```

expr 中可以包含以下常量：

`n`
当前处理的帧的数量，从**0**开始

`n_forced`
强制帧的数量

`prev_forced_n`
先前强制帧的数目，当没有强制的关键帧时，它是**NAN**

`prev_forced_t`
先前强制帧的时间，当没有强制关键帧时，它是**NAN**

`t`
当前处理的帧的时间

例如每**5**秒强制插入一个关键帧，你可以使用：
`-force_key_frames expr:gte(t,n_forced*5)`

自**13**秒开始，强制插入一个关键帧在上一强制关键帧后**5**秒：
`-1-force_key_frames expr:if(isnan(prev_forced_t),gte(t,13),gte(t,prev_forced_t+5))`

需要注意的是太多强制的关键帧对某些编码器的超前算法是非常有害的：使用固定的GOP选项或类似选项会更有效。

`-copyinkf[:stream_specifier] (output,per-stream)`

当复制流时，也复制起始处的非关键帧。

`-hwaccel[:stream_specifier] hwaccel (input,per-stream)`

使用硬件加速解码匹配流。 `hwaccel` 的允许值包括：

`none`
不要使用任何硬件加速（默认）。

`auto`
自动选择硬件加速的方法。

`vda`
使用苹果**VDA**硬件加速。

`vdpau`
使用**VDPAU**（视频解码和演示**API**对于**Unix**）硬件加速。

`dxva2`
使用**DXVA2**（**DirectX**视频加速）硬件加速。

如果选择的hwaccel不可用或不支持选择的解码器，此选项没有效果。

注意，大多数加速方法适用于播放，但并不会比现代CPU软件解码更快。此外，ffmpeg通常需要解码帧从GPU存储器复制到系统存储器，从而导致进一步的性能损失。因此此选项主要用于测试。

`-hwaccel_device[:stream_specifier] hwaccel_device (input,per-stream)`

选择一个设备使用硬件加速。

此选项仅使得同时与 `-hwaccel` 选项使用。它的确切含义取决于所选择的具体硬件加速方法。

`vdpau`
对于**VDPAU**，此选项在**X11**中使用。如果没有指定这个选项， **DISPLAY**环境变量的值被使用

`dxva2`
对于**DXVA2**，这个选项应包含在显示适配器使用的数量。如果未指定此选项，默认的适配器将被使用。

5.7 音频选项

`-aframes number (output)`

设定的音频帧输出的数目。这是 `-frames:a` 的别名。

`-ar[:stream_specifier] freq (input/output,per-stream)`

设置音频采样频率。对于输出流，将默认设置为相应的输入流的采样频率。对于输入流该选项仅对音频抓取设备和raw格式分流器和映射到相应分流器上的选项有效。

`-aq q (output)`

设置音频质量（与编解码器有关，VBR）。这是 `-q:a` 的别名。

`-ac[:stream_specifier] channels (input/output,per-stream)`

设置音频通道的数目。输出流默认将设置为输入音频信道的数目。对于输入流该选项仅对音频抓取设备和raw格式分流器和映射到相应分流器上的选项有效。

`-an (output)`

禁用录音。

`-acodec codec (input/output)`

设置音频解码器。这是 `-codec:a` 的别名。

`-sample_fmt[:stream_specifier] sample_fmt (output,per-stream)`

设置音频样本格式。使用 `-sample_fmts` 得到支持的采样格式的列表。

`-af filtergraph (output)`

创建由FilterGraph指定的滤镜组并使用它。

这是 `-filter:a` 的别名 ，参见 `-filter` 选项。

5.8 高级音频选项

`-atag fourcc/tag (output)`

强制音频 `tag/fourcc` 值。这是 `-tag:a` 的别名。

`-absf bitstream_filter`

已过时，参见 `-bsf`

`-guess_layout_max channels (input,per-stream)`

如果一些输入通道布局是未知的，试图猜测它最多的声道数量。例如，2要求ffmpeg识别1个通道为单声道和2声道立体声，6声道作为5.1。默认是总是试图去猜测。用0来禁用所有的猜测。

5.9 字幕选项

`-scodec codec (input/output)`

设置字幕的编解码器。这是 `-codec:s` 的别名。

`-sn (output)`

关闭字幕记录。

`-sbsf bitstream_filter`

已过时，参见 `-bsf`

5.10 高级字幕选项

`-fix_sub_duration`

调整字幕的持续时间。对于每个字幕，等待在相同的流中的下一个分组，并调节第一持续时间，以避免重叠。这对某些字幕编解码器是必要的，特别是数字电视广播字幕(DVB)，因为在原来的分组的持续时间仅仅是一个粗略的估计，结束标记是通过一个空的字幕帧完成的。没有使用这个选项在必要时可导致夸张的持续时间，或由于非单调时间戳混流故障。

注意，此选项将延迟所有数据的输出直到下一个字幕分组被解码：它可能会增加内存消耗和延迟。

`-canvas_size size`

设置用于呈现字幕的画布的大小。

5.11 高级选项

`-map [-]input_file_id[:stream_specifier] [,sync_file_id[:stream_specifier]] | [linklabel] (output)`

指定一个或多个输入流作为用于输出文件的来源。每个输入流由输入文件索引 `input_file_id` 和输入流索引 `input_stream_id` 标识。这两个指标都是从0开始。 `sync_file_id：stream_specifier` 可用于指定输入流作为同步参考。

在命令行上第一个 `-map` 选项指定的输出流0，第二 `-map` 选项指定的源输出流1等

一个连字符 `-` 创建一个“否定的”的映射。它禁用从已经建立映射中匹配流。

另一种 `LinkLabel` 的形式将映射从复杂滤波器组输出到输出文件（见 `-filter_complex` 选项）。 `LinkLabel` 必须对应于一个链路中已定义的输出标签。

例如，映射第一输入文件的所有流到输出
`ffmpeg -i INPUT -map 0 output`

例如，如果在第一输入文件中的两个音频流，这些流记为**“0：0”，“0：1”**。您可以使用**-map**选择哪个流输出到输出文件。例如：

`ffmpeg -i INPUT -map 0:1 out.wav`

将映射**INPUT**的输入数据流**“0:1”**到在（单）输出流**out.wav**。

例如，从输入文件**a.mov**选择索引**2**流（由识别符**“0：2”**指定的）与从输入**b.mov**索引**6**（由识别符**“1：6”**指定）流，复制到输出文件**out.mov**：

```
ffmpeg -i a.mov -i b.mov -c copy -map 0:2 -map 1:6 out.mov
```

选择所有的视频和输入文件中的第三音频流：

```
ffmpeg -i INPUT -map 0:v -map 0:a:2 OUTPUT
```

映射所有的数据流，除了第二音频，使用否定的映射

```
ffmpeg -i INPUT -map 0 -map -0:a:1 OUTPUT
```

选择英语音频流：

```
ffmpeg -i INPUT -map 0:m:language:eng OUTPUT
```

请注意，使用此选项将禁用此输出文件的默认映射。

`-map_channel [input_file_id.stream_specifier.channel_id|-1] [:output_file_id.stream_specifier]`

映射从一个给定的输入音频声道到输出。如果 `output_file_id.stream_specifier` 未设置时，声道将被映射的所有音频流。

使用 `-1` 替代 `input_file_id.stream_specifier.channel_id` 将映射一个静音声道。

例如，假设**INPUT**是一个立体声音频文件，你可以切换两个音频通道与下面的命令：

```
ffmpeg -i INPUT -map_channel 0.0.1 -map_channel 0.0.0 OUTPUT
```

如果您想要静音的第一声道，并保留第二个：

```
ffmpeg -i INPUT -map_channel -1 -map_channel 0.0.1 OUTPUT
```

`-map_channel` 选项的顺序指定在输出流中的声道的顺序。所述输出信道的布局是从映射信道数猜测的（如果只有一个 `-map_channel` ，则使用单声道，如果是2，则使用立体声，如此等等）。如果输入和输出信道的布局不匹配（例如2个 `-map_channel` 和 `-ac 6` ），联合使用 `-map_channel` 和 `-ac` 将更新声道的增益水平。

您也可以提取各个输入通道到特定的输出；以下命令中提取的两个通道的INPUT音频流（文件0，流0）到相应的 `OUTPUT_CH0` 和 `OUTPUT_CH1` 输出：

```
ffmpeg -i INPUT -map_channel 0.0.0 OUTPUT_CH0 -map_channel 0.0.1 OUTPUT_CH1
```

以下示例拆分立体声输入的通道成两个独立的数据流，其被放入同一个输出文件：

```
ffmpeg -i stereo.wav -map 0:0 -map 0:1 -map_channel 0.0.0:0.0
-map_channel 0.0.1:0.1 -y out.ogg
```

注意，目前每个输出流只能包含来自单个输入流的声道；不能使用 **-map_channel** 从不同的数据流提取的多个输入音频声道（来自相同的或不同的文件），并将它们合并成一个单一的输出流。因此，目前不可能完成例如把两个单独的单声道流成一个单一的立体声流的任务。然而，拆分立体声流分成两个单信道单声道流则是可能的。

如果你需要这个功能，一个可能的解决方法是使用 **amerge** 滤镜。例如，如果你需要用2个单声道音频融合媒体中的（在这里是input.mkv）流成一个单一的立体声声道音频流（并保持视频流），可以使用下面的命令：

```
ffmpeg -i input.mkv -filter_complex "[0:1] [0:2] amerge" -c:a
pcm_s16le -c:v copy output.mkv
```

-map_metadata[:metadata_spec_out] infile[:metadata_spec_in] (output,per-metadata)

根据 **infile** 设置下一个输出文件的元数据信息。请注意，这些都是文件索引（从零开始），而不是文件名。可选参数 **metadata_spec_in/out** 可用于指定哪些元数据进行复制。元数据说明可以有以下几种形式：

`g`
全局元数据，即元数据应用于整个文件

`s[:stream_spec]`
每个流的元数据`**stream_spec**`是一个流标识符，参见流标识符一章。在输入的元数据时，将从第一个匹配的流复制。在输出元数据时，将复制到所有匹配流。

`c:chapter_index`
每章的元数据`**chapter_index**`是从零开始的章节索引。

`p:program_index`
每个项目的元数据`**program_index**`是从零开始的的项目索引。

如果元数据说明被省略，则默认为全球性的。

缺省情况下，全局元数据是从第一输入文件复制的，每个流和每个章节的元数据与数据流/章节将被依次复制。创建相关类型的任何映射将禁用这些默认映射。否定的文件索引可以用来创建只禁用自动复制的虚拟映射。

例如，从输入文件的第一数据流复制元数据到输出文件的全局元数据：
```
ffmpeg -i in.ogg -map_metadata 0:s:0 out.mp3
```

反过来，即全局的元数据复制到所有音频流：

```
ffmpeg -i in.mkv -map_metadata:s:a 0:g out.mkv
```

需要注意的是**0**在本实例中能起到相同的效果，由于全局元数据被假定默认。

-map_chapters input_file_index (output)

从输入文件复制章节与索引 **input_file_index** 到下一个输出文件中。如果不指定章节映射，则章节从第一输入文件复制并至少有一个章节。使用负的文件索引来排除任何章节的复制。

-benchmark (global)

在编码结束显示基准信息。显示使用的CPU时间和最大内存消耗。不是所有系统都支持最大内存消耗，如果不支持，它通常会显示为0。

-benchmark_all (global)

显示编码过程中基准信息。显示各个步骤（音频/视频编码/解码）所使用的CPU时间。

-timelimit duration (global)

在已经运行 **duration** 秒后退出ffmpeg

-dump (global)

转储每个输入包到标准错误流。

-hex (global)

当dump包时，也dump有效载荷。

-re (input)

读取输入的原始帧速率。主要用于模拟抓取设备。或实时输入流（例如从文件读取时）。不应该在实际的抓取设备或实时输入流中使用（它可能会导致数据包丢失）。默认ffmpeg尝试尽可能快地读出输入端。此选项会减慢输入的本地帧速率的读取。它是用于实时输出（如直播）是有用的。

-loop_input

循环输入流。目前，它仅适用于图像流。此选项用于自动测试 **ffserver** 。此选项已被弃用，使用 **-loop 1** 。

-loop_output number_of_times

反复循环输出支持循环的格式如动画GIF（0意味着无限循环输出）。此选项已被弃用，使用 **-loop** 。

-vsync parameter

视频同步方法。出于兼容性考虑旧值可以被指定为数字。新添加的值将必须总是指定为字符串。

0, passthrough
每一帧传递从分流器到复用器的时间戳。

1, cfr
帧将被复制并下降至达到完全所需的恒定帧速率。

2, vfr
帧通过与其时间戳或下降，从而防止两帧具有相同的时间戳。

drop
与**passthrough**相同，但破坏所有时间戳，使得复用器生成基于帧速率新的时间戳。

-1, auto
根据复用器的功能选择**1**或者**2**。这是默认的方法。

注意，该时间戳在此之后可以进一步由复用器修改。例如，在格式选项 **avoid_negative_ts** 被启用时。

与 **-map** 联用，您可以选择从哪个流提取时间戳。您可以留下视频或音频不变，同步剩余流不变的。

-async samples_per_second

音频同步的方法。“伸展/挤压”音频流相匹配的时间戳，所述参数是音频发生改变所容许的每秒最大点数。 **-async 1** 是一种特殊情况，音频数据流仅在开始校正，而其后将不再校正。

注意，该时间戳在此之后可以进一步由复用器修改。例如，在格式选项 **avoid_negative_ts** 被启用时。

此选项已被弃用。使用 **aresample** 音频过滤器代替。

-copyts

不要处理输入时间戳，但保持它们的值，不尝试对它们进行sanitize。尤其是，不要删除初始启动时间偏移值。

需要注意的是，即使使用了该选项，根据不同的 **vsync** 选项或对特定复用器处理（例如格式选项 **avoid_negative_ts** 被启用）输出时间戳与输入可能不匹配时间戳。

-start_at_zero

当与 **copyts** 使用，调整输入时间戳，使他们从零开始。

这意味着使用例如 **-ss 50** 将使输出时间戳开始50秒时，不管输入文件开始处的

时间戳是多少。

-copytb mode

指定在拷贝流时如何设置编码器的时间基准，mode是一个整型，可以假定为下列值之一：

1
使用分流器的时间基准。

时间基准从相应的输入分流器复制到输出编码器。该情况下，对可变帧速率的视频流的复制，有时需要避免非单调的时间戳。

0
使用解码器时间基准。

时间基准从相应的输入解码器复制到输出编码器。

-1
尝试自动做出选择，以便产生一个合理的输出。

默认值为-1。

-shortest (output)

最短的输入流结束时完成编码。

-dts_delta_threshold

时间戳间断门槛。

-muxdelay seconds (input)

设置最大分流解码延时。

-muxpreload seconds (input)

设定初始解码分流延迟。

-streamid output-stream-index:new-value (output)

分配一个新流id值到输出流。该选项需指定到输出文件名之前。对于在多个输出文件存在的情况下，一个流id将被重新分配到不同的值。

例如，要设置流**0** 到流**33**，同时流**1**至流**36**，到**MPEGTS**格式的输出文件：
ffmpeg -i infile -streamid 0:33 -streamid 1:36 out.ts

-bsf[:stream_specifier] bitstream_filters (output,per-stream)

指定匹配流的 **bitstream_filters** 。它是一个逗号分隔的滤镜列表。使用 **-**

bsfs 选项得到的滤镜列表。

```
ffmpeg -i h264.mp4 -c:v copy -bsf:v h264_mp4toannexb -an ou
t.h264
ffmpeg -i file.mov -an -vn -bsf:s mov2textsub -c:s copy -f ra
wvideo sub.txt
```

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

指定匹配流的 **tag/fourcc** 。

-timecode hh:mm:ssSEPff

指定写入的时间码。对于non drop 时间码，分隔符SEP是 **:** 对于drop时间码是 **;** （或 **.** ）。

```
ffmpeg -i input.mpg -timecode 01:02:03.04 -r 30000/1001 -s nt
sc output.mpg
```

-filter_complex filtergraph (global)

定义复杂FilterGraph，即具有输入和/或输出任意个数的滤镜组。对于简单滤镜——那些具有一个输入和相同类型的一个输出的滤镜——参见 **-filter** 选项。FilterGraph是FilterGraph的描述，参见FFMPEG滤镜手册的“FilterGraph语法”部分中的描述。

输入链路标签必须关联到使用 **[file_index:stream_specifier]** 语法标记的输入流（即-map选项中使用的格式）。如果 **stream_specifier** 匹配多个流，第一个将被使用。未标记的输入将被连接到匹配类型的第一未使用的输入流。

输出链接标签由 **-map** 指定。未标记的输出被加到第一输出文件。

注意，使用该选项，可以只有lavfi源而没有正常的输入文件。

例如，叠加图像到视频

```
ffmpeg -i video.mkv -i image.png -filter_complex '[0:v][1:v]o
verlay[out]' -map '[out]' out.mkv
```

这里**[0:v]**指的是在第一输入文件中的第一视频流，这是与覆盖滤波器的第一（主）输入。同样，在第二输入的第一视频流链接到覆盖层的第二（覆盖）输入。

假定在每个输入文件中只有一个视频流，就可以省略输入标签，因此上述命令相当于

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay[ou
t]' -map'[out]' out.mkv
```

此外，我们可以省略输出标签和单输出的滤镜，它将被自动添加到输出文

件，所以我们可以简单地写
```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay' ou
t.mkv
```

要使用**lavfi**产生**5**秒钟的纯红色视频**color**来源：
```
ffmpeg -filter_complex 'color=c=red' -t 5 out.mkv
```

-lavfi filtergraph (global)

定义复杂FilterGraph，即具有输入和/或输出任意个数的滤镜组。相当于 **-filter_complex** 。

-filter_complex_script filename (global)

这个选项类似于 **-filter_complex** ，唯一的区别是，它的参数是包含被读取FilterGraph的文件的名称。

-accurate_seek (input)

此选项启用或禁用准确寻求输入文件，配合 **-ss** 选项使用。它默认是启用的，所以当转码时是准确的。使用 **-noaccurate_seek** 禁用它，在拷贝一些数据流和转码时，这可能是有用的。

-override_ffserver (global)

从ffserver 覆盖输入规格。使用这个选项，你可以映射任何输入流到ffserver，并从ffmpeg 控制编码的许多方面。如果没有这个选项ffmpeg将发送ffserver要求的内容。

该选项用于那些不能被指定到ffserver的功能 ，即便他们可以用于ffmpeg 。

-discard (input)

允许在分路器丢弃特定流或流帧。不是所有的分路器都支持。

none
禁止丢弃帧。

default
默认情况下，不丢弃帧。

noref
丢弃所有非参考帧。

bidir
放弃所有双向帧。

nokey
丢弃所有帧除了关键帧。

all
丢弃所有帧。

作为一个特例，可以使用一个位图字幕流作为输入：它将被转换到文件中最大的视频相同尺寸，或720×576，如果没有视频。需要注意的是，这是一个实验性和临时解决方案。它会在libavfilter有适当字幕支持后删除。

例如，硬编码存储在**MPEG-TS**格式的**DVB-T**的记录的顶部字幕，**1**秒延迟字幕：

```
ffmpeg -i input.ts -filter_complex \
  '[#0x2ef] setpts=PTS+1/TB [sub] ; [#0x2d0] [sub] overlay' \
  -sn -map '#0x2dc' output.mkv
```

（**0x2d0**，**0x2dc**和**0x2ef**分别是**MPEG-TS**的**PID**的视频，音频和字幕流**;** **0**：**0,0**：**3**和**0**：**7**可起到同样作用）

5.12 预置文件

预置文件是包含 **option = value** 的文件 ，每行一个，指定的选项需按照命令行中的顺序。’ #’开头的字符行被忽略，并用来提供注释。参见FFmpeg的源代码树中的preset目录中的例子。

Preset files are specified with the vpre, apre, spre, and fpre options. The fpre option takes the filename of the preset instead of a preset name as input and can be used for any kind of codec. For the vpre, apre, and spre options, the options specified in a preset file are applied to the currently selected codec of the same type as the preset option.

The argument passed to the vpre, apre, and spre preset options identifies the preset file to use according to the following rules:

First ffmpeg searches for a file named arg.ffpreset in the directories $FFMPEG_DATADIR(ifset), and$HOME/.ffmpeg, and in the datadir defined at configuration time (usually PREFIX/share/ffmpeg) or in a ffpresets folder along the executable on win32, in that order. For example, if the argument is libvpx-1080p, it will search for the file libvpx-1080p.ffpreset.

If no such file is found, then ffmpeg will search for a file named codec_name-arg.ffpreset in the above-mentioned directories, where codec_name is the name of the codec to which the preset file options will be applied.

6 提示

- 为在非常低的比特率流，使用低帧速率和小的GOP大小。这对于那些配置较低的Linux用户上播放RealVideo尤是如此，它可能会丢帧。例如：
  ffmpeg -g 3 -r 3 -t 10 -b:v 50k -s qcif -f rv10 /tmp/b.rm
- 编码中显示的参数“q”是当前量化器。值1表示非常良好的质量。值31表

示最差的质量。如果Q = 31出现过于频繁，这意味着该编码器无法压缩到满足您的比特率要求的码率。您必须增加比特率，降低帧率或减少帧的大小。

- 如果你的电脑不够快，可以牺牲压缩比换取速度。你可以用 `-me 0` 加快运动估计和 `-g 0` 完全禁用运动估计（你仅有I-frames，这意味着它几乎和JPEG压缩一样好）。
- 可通过降低采样频率获得非常低的音频比特率（对于MPEG音频，下降到22050Hz，对于AC-3， 使用22050或11025Hz）。
- 有一个恒定的质量（但可变比特率），使用选项 `-qscale N` 时， `N` 是1（优秀品质）和31（质量最差）之间。

7 范例

7.1 预置文件

预置文件包含option=value，一个用于每行，指定其也可以指定的命令行上的选项的序列的序列。’ #’ 开头的字符行被忽略，并用来提供注释。空行将也被忽略。检查的例子FFmpeg的源代码树中的preset目录。

预置文件中指定的pre选项，这个选项需要一个预设名称作为输入。FFmpeg在 `$AVCONV_DATADIR` 和 `$HOME/.ffmpeg` 以及编译时指定的目录（通常是 `$PREFIX/share/ffmpeg` ）的目录中搜索文件名为 `preset_name.avpreset` 的文件。例如，如果该参数是 `libx264-max` ，它会搜索文件 `libx264-max.avpreset` 。

7.2 视频和音频抓取

如果指定了输入格式和设备，ffmpeg将可以直接抓取视频和音频。 ffmpeg -f oss -i /dev/dsp -f video4linux2 -i /dev/video0 /tmp/out.mpg

或从ALSA音源（单声道输入，卡ID 1），而不是OSS： ffmpeg -f alsa -ac 1 -i hw:1 -f video4linux2 -i /dev/video0 /tmp/out.mpg

请注意，您必须在启动ffmpeg前激活正确的视频源和信道，比如的xawtv。你也必须正确设置混音器中的音频记录电平。

7.3 X11抓取

使用ffmpeg抓取X11显示器 ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0 /tmp/out.mpg 0.0是display.screen数量的X11服务器，与DISPLAY环境变量一致。

```
ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0+10,20
/tmp/out.mpg
```

0.0是**display.screen**数量的**X11**服务器，与**DISPLAY**环境变量一致。**10**是在**x**偏移和**20**的**y**偏移。

7.4 视频和音频文件格式转换

任何支持的文件格式和协议可以作为ffmpeg的输入：

范例：

您可以使用YUV文件作为输入： **ffmpeg -i /tmp/test%d.Y /tmp/out.mpg**

它将使用文件：

```
/tmp/test0.Y, /tmp/test0.U, /tmp/test0.V,/tmp/test1.Y, /tmp/t
est1.U, /tmp/test1.V, etc...Y，/ tmp目录/ TEST0。U, /tmp/test
0.V,
/tmp/test1.Y, /tmp/test1.U, /tmp/test1.V, etc...
```

Y文件使用的U和V文件的分辨率的两倍。他们是原始文件，没有头信息。他们可以通过所有视频解码器产生。您必须指定图像的大小与-s选项，如果ffmpeg的不能猜测它。

您可以从原始YUV420P文件输入： **ffmpeg -i /tmp/test.yuv /tmp/out.avi**

test.yuv是含有生YUV平面数据的文件。每个帧是由Y平面后跟U和V平面的一半的垂直和水平分辨率。

可以输出到原始文件YUV420P： **ffmpeg -i mydivx.avi hugefile.yuv**

您可以设置多个输入文件和输出文件： **ffmpeg -i /tmp/a.wav -s 640x480 -i /tmp/a.yuv /tmp/a.mpg**

转换音频文件a.wav和原始YUV视频文件a.yuv到MPEG文件a.mpg。

你也可以同时做音频和视频转换： **ffmpeg -i /tmp/a.wav -ar 22050 /tmp/a.mp2**

使用22050赫兹的采样率转换a.wav到MPEG音频。

可以同时进行多种格式的编码，并定义从输入流到输出数据流的映射： **ffmpeg -i /tmp/a.wav -map 0:a -b:a 64k /tmp/a.mp2 -map 0:a -b:a 128k /tmp/b.mp2**

转换a.wav到64千比特的a.mp2和128千比特的b.mp2。 **-map file:index** 输出数据流的定义的顺序指定了用于每一个输出流的输入流。

您可以转码解密的VOB： **ffmpeg -i snatch_1.vob -f avi -c:v mpeg4 -b:v 800k -g 300 -bf 2 -c:a libmp3lame -b:a 128k snatch.avi**

这是一个典型的DVD翻录的例子;输入是VOB文件，输出与MPEG-4视频和MP3音频的AVI文件。注意，在这个命令，我们使用B帧，以便对MPEG-4流与DivX5兼容，并且GOP大小为300，这意味着每10秒插入一帧到29.97fps的输入视频。此外，音频流是MP3编码，所以你需要启用通过传递LAME支持可使用–enable-libmp3lame配置。该映射是特别有用的用于DVD的转码，以获得所需的音频语言。

注意：使用 **ffmpeg -formats** 查看支持的输入格式。

您可以从视频中提取图像，或从图像创建视频：

从视频中提取图片： `ffmpeg -i foo.avi -r 1 -s WxH -f image2 foo-%03d.jpeg`

这将每秒提取一个视频帧，并输出到名为foo-001.jpeg，foo-002.jpeg等图片中。图片将被重新缩放到定义的尺寸。

如果你想提取有限数量的帧，你可以组合使用上面的命令与-vframes或-t选项，或与-ss开始从某一个时间点提取。

从图像创建视频： `ffmpeg -f image2 -i foo-%03d.jpeg -r 12 -s WxH foo.avi`

语法 `foo-%03d.jpeg` 指定要使用的三个数字组成的十进制数用零填充到表达的序列号。它支持C语言 `printf` 函数相同的语法，但只有格式接受整数。

当导入图像序列， `-i` 还支持Shell扩展通配符。这在内部通过image2-specific `-pattern_type glob` 选项选择。

例如，从文件名匹配 `foo-*.jpeg` 的图片创建视频 `ffmpeg -f image2 -pattern_type glob -i 'foo-*.jpeg' -r 12 -s WxH foo.avi`

可以把相同类型的许多流到输出文件： `ffmpeg -i test1.avi -i test2.avi -map 1:1 -map 1:0 -map 0:1 -map 0:0 -c copy -y test12.nut`

产生的输出文件test12.nut将包含从以相反的顺序存储的输入文件中的四个流。

要强制CBR视频输出： `ffmpeg -i myfile.avi -b 4000k -minrate 4000k -maxrate 4000k -bufsize 1835k out.m2v`

`lmin` ， `lmax` ， `mblmin` 和 `mblmax` 四个选项的单位是 `lambda` ，但你可以使用QP2LAMBDA常熟轻松地从 `q` 单位转换： `ffmpeg -i src.ext -lmax 21*QP2LAMBDA dst.ext`

8 参见

ffmpeg-all , ffplay , ffprobe , ffserver , ffmpeg-utils , ffmpeg-scaler , ffmpeg-resampler , ffmpeg-codecs , ffmpeg-bitstream-filters , ffmpeg-formats , ffmpeg-devices , ffmpeg-protocols , ffmpeg-filters

亲，给点评论吧！

Related Posts

新加坡LTA的增强JSON API 07 Apr 2016

新加坡LTA的JSON API 05 Apr 2016

新加坡NEA的JSON API 05 Apr 2016