FFMPEG(1)

名称

ffmpeg - ffmpeg 视频转换器

概要

ffmpeg [全局选项] {[输入文件选项] -i 输入文件} ... {[输出文件选项] 输出文件}

描述

ffmpeg 是一款能够抓取活动音视频源的快速音视频转换器。借助于一个高质量的多相滤波器，ffmpeg 能够飞速地在任意采样率与调整视频大小之间进行转换。

ffmpeg 从任意数量的输入文件中读取数据（普通文件、管道、网络流媒体，采集器），这些输入文件通过-i 选项指定，然后输出到任意数量的输出文件中，输出文件以纯输出文件名指定。命令行上任何无法被解释为选项的内容，都将被视为输出文件名。

原则上，每个输入输出文件都能包含任意数量的各种类型的流（视频/音频/字幕/附件/数据）。容器格式可能会限制流的类型和数量。选择输入中的哪一个流输出到哪一个输出，可以自动决定，也可以通过“-map”来决定（详见流的选择章节）。

为了有选择地访问输入文件，您必须使用输入文件的索引（索引从 0 开始）。例如，第一个输入文件的索引为 0，第二个是 1，等等。同样地，文件中的流也是通过索引来访问的。例如，“2:3”为第三个输入文件中的第四个流的索引，详见流描述符这一章节。

作为一个通用的规则，选项是作用于下一个指定文件的，因此，顺序就尤为重要了，不过你可以在命令行中多次使用同一个选项，每一次使用都作用于下一个输入或输出文件。不过，全局选项（例如详细级别）必须最先指定，这是个例外。

不要混淆了输入和输出文件 -- 首先指定所有的输入文件，然后是所有的输出文件。同样不要混淆了文件各自的选项。

所有的选项都仅仅作用于下一个输入输出文件，而且文件之间，选项互不影响。

- 将输出文件的视频比特率设为 64kbit/s:

  ffmpeg -i input.avi -b:v 64k -bufsize 64k output.avi
- 将输出文件的帧率设为 24fps:

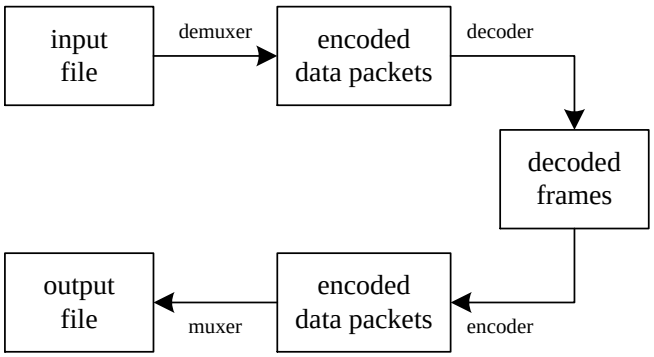
  ffmpeg -i input.avi -r 24 output.avi
- 将输入文件的帧率（仅对原始格式有效）设为 1fps，输出文件的帧率设为 24fps:

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

原始输入文件可能需要格式相关的选项。

详细描述

下面的框图表示了 ffmpeg 转码得到每一个输出文件的过程:



ffmpeg 调用 libavformat 库（包含了解复用器）读取输入文件，得到包含了已编码数据的 packet。如果有多个输入文件，ffmpeg 将试图通过跟踪任何有效流的最小时间戳，来保持所有输入文件的同步。

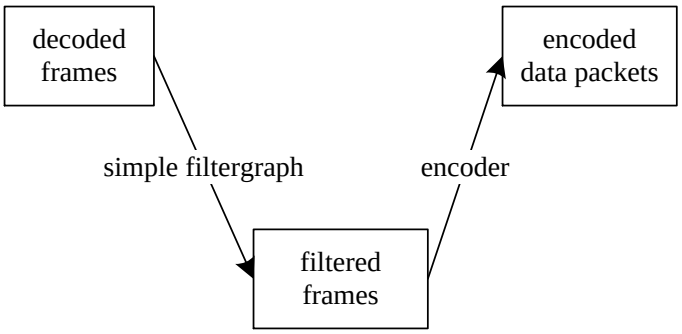
以编码的 packet 会被传送给 decoder（除非选择了流拷贝，详见后面章节）。decoder 会输出未经压缩的帧（原始的视频/PCM 音频/...），这些帧会被传送给滤波器作进一步处理。滤波后，这些帧被传送给编码器进行编码，得到编码过的 packets，然后被送往复用器，并经由复用器写入输出文件。

滤波

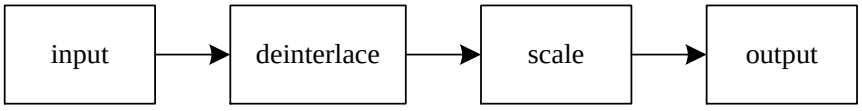
在编码之前，ffmpeg 将利用 libavfilter 库里面的滤波器对原始音视频帧进行处理。几个连在一起的滤波器组成一个滤器图。ffmpeg 内部有两种滤器图：简单滤器图和复杂滤器图。

简单滤器图

简单滤器图包括类型相同的一个输入和一个输出。在上面的图中，简单滤器图可被视为在解码和编码之间简单地插入一个额外的步骤：



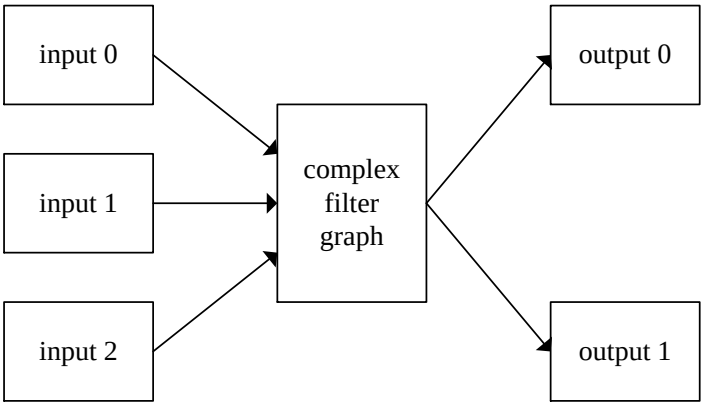
简单滤器图由 per-stream-filter 选项（-vf（视频）和-af（音频））配置。一个用于处理视频的简单滤器图可如下所示：



注意，一些滤波器会改变帧的属性，但不会改变帧的内容，例如，上面的“fps”滤波器会改变帧的数目，但不会改变帧的内容。另外一个例子是“setpts”滤波器，他仅设置时间戳，否则的话将原封不动地把帧往后传送。

复杂滤器图

复杂滤器图是那些无法简单地描述为对一个流进行简单的线性处理的滤器图。比如一个图有一个以上的输入和输出，或输入和输出的类型不一样。复杂滤器图可由下图表示：



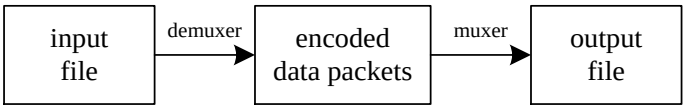
复杂滤器图由选项-filter_complex 来配置。注意，由于复杂滤器图在特性上就无法明确地与一个单独的流或文件联系起来，因此这个选项是全局的。

选项-lavfi 和选项-filter_complex 是等价的。

一个惯用的的复杂滤器图的例子是“覆盖”滤波器，它有两个视频输入和一个视频输出，其中一个视频覆盖在另一个之上。与之对应的音频部分则是“amix”滤波器

流拷贝

对-codec 选项使用“copy”参数，便选择了流拷贝模块。流拷贝使得 ffmpeg 跳过了解码和编码过程，仅仅进行了解复用和复用处理。这对更改容器格式或修改容器级别的 metadata 很有用。流拷贝可由下图表示：



由于没有解码编码过程，因此处理起来非常快，而且没有失真。然而，在很多场合下，流拷贝会由于很多因素而无法正常工作。使用滤波器也是明显不可能的，因为滤波器只能处理未压缩的数据。

流选择

默认情况下，ffmeng 只会包括输入文件中每种类型（视频、音频、字幕）各一个流，然后把他们添加到每一个输出文件中。ffmpeg 会根据下面的准则来找到每种类型各自的最佳流：对于视频，分辨率最高的流是最佳流；对于音频，声道数最多的流是最佳流；对于字幕，第一个字幕流就是最佳流。在这个规则下，如果某一类型的几个流得分一样的话，ffmpeg 将会选择索引最小的那一个。

您可以通过“-vn/-an/-sn”来禁用这个默认行为。可以使用“-map”选项来禁止刚刚所述的默认行为，以实现完全的用户控制。

选项

在没有特别说明的情况下，所有的数字类选项，都接受字符串样式的数字作为输入，都

可以在其后面跟上国际单位制词头，如：‘K’、‘M’或‘G’

如果这些国际单位制词头带上了i,就会被解释为二进制乘数词头,而且是基于1024的,而不是1000。如果带上了B，则要乘8。这意味着可以使用诸如：‘KB’，‘MiB’，‘G’和‘B’作为数字后缀。

不带参数的选项称为布尔选项。这些选项会将相应的值设成true。在布尔选项的名字前面加上“no”会将相应的值设成false，如使用“-nofoo”会将“foo”设成false。

流指定符

一些选项是作用于每一个流的，如比特率、编解码器。流指定符是用来精确指定一个给出的选项属于哪一个流。

流指定符是一个字符串，添加在选项名后面，通过冒号与选项名分割开，如，“-codec:a:1 ac3”包含了一个流指定符“a:1”,这个指定符指定了第二个音频流。因此,ffmpeg将为第二个音频流选择ac3编解码器。

流指定符可以匹配多个流，这样一来选项将作用于所有匹配的流，如，“-b:a 128k”中的流指定符会匹配所有音频流。

一个空流指定符匹配所有的流，如“-codec copy”或者“-codec: Copy“将在不重新编码的情况下拷贝所有流。

流指定符可能的形式如下：

stream_index

匹配索引为stream_index的流，如，“-threads:1 4”将设置第二个流的线程数为4。

stream_type[:stream_index]

stream_type为下面的其中之一：‘v’表示视频，‘a’表示音频，‘s’表示字幕，‘d’表示数据，'t'表示附件。如果给定stream_index，则会匹配指定类型的索引为stream_index的流，否则匹配所有该类型的流。

p:program_id[:stream_index]

如果给定stream_index，则匹配id为program_id的程序里面索引为stream_index的流，否则匹配程序里所有流。

#stream_id 或者 i:stream_id

通过stream_id来匹配流（例如，MPEG-TS容器里的PID）

m:key[:value]

匹配metadata包含指定符中键值对的流。如果value没有指定，则匹配metadata中包含key的流。

注意，在ffmpeg中，通过metadata来匹配仅对输入文件有效。

通用选项

通用选项是共享于所有ff*工具的。

-L 显示证书

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

显示帮助。可以一个可选的选项参数来打印关于某个条目的帮助。如果没有参数，仅仅显示基本（非高级）的工具选项。

可用的参数值如下：

long

在显示基本工具选项的基础上，再显示高级工具选项。

full

打印完整的选项列表，包括编码器、解码器、解复用器、复用器、滤波器等的私有和共享选项。

decoder=decoder_name

打印名为 decoder_name 的解码器的详细信息。-decoders 选项可以得到一个包含所有解码器的列表。

encoder=encoder_name

打印名为 encoder_name 的编码器的详细信息。-encoders 选项可以得到一个包含所有编码器的列表。

demuxer=demuxer_name

打印名为 demuxer_name 的解复用器的详细信息。-formats 选项可以得到一个包含所有解复用/复用器的列表。

muxer=muxer_name

打印名为 muxer_name 的复用器的详细信息。-formats 选项可以得到一个包含所有解复用/复用器的列表。

filter=filter_name

打印名为 filter_name 的滤波器的详细信息。-filters 选项可以得到一个包含所有滤波器的列表。

-version

显示版本信息

-formats

显示所有可用的格式

-codecs

显示 libavcodec 库可识别的所有编解码器。

注意，术语‘codec’贯穿整篇文档，它是一个媒体特比流格式的缩写。

-decoders

显示可用的解码器

-encoders

显示可用的编码器

-bsfs

显示可用的比特流滤波器

-protocols

显示可用的协议

-filters

显示 libavfilter 库中可用的滤波器

-pix_fmts

显示可用的像素格式

-sample_fmts

显示可用的采样格式

-layouts

显示通道名和标准通道布局

-colors

显示可识别的颜色名称

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

设置库使用的日志级别。增加“repeat+”表示不要压缩重复输出的日志到第一行。“repeat”可以单独使用，此时如果没有设定日志级别，那么将使用默认级别。如果给出了多个日志级别参数，‘repeat’不会改变日志级别。

日志级别可以是一个数字，也可以是下面的字符串：

quiet

不要显示任何日志；安静

panic

仅仅显示一些可导致程序崩溃的致命信息，例如断言失败。这个级别目前还没有被使用。

fatal

仅仅显示一些致命错误，出现这些错误后程序往往无法继续进行下去。

error

显示错误，包括那些可恢复的错误。

warning

显示警告和错误。任何有关错误或超出预期的事件的信息都将显示出来。

info

显示程序运行期间提示性的消息。这是对 error 和 warning 的补充，是默认的级别。

verbose

和“info”一样，除了更多的动作。

debug

显示所有的信息，包括调试信息。

程序默认将日志输出的标准错误输出，如果终端支持颜色，程序将对错误信息

和警告信息使用不同颜色。给日志上色可通过设置环境变量 AV_LOG_FORCE_NOCOLOR 或者 NO_COLOR 来禁用，或者可以通过设定环境变量 AV_LOG_FORCE_COLOR 启用。环境变量 NO_COLOR 在以后的 FFmpeg 版本中会被弃用。

-report

将完整的命令行和控制台输出到当前目录下的“program-YYYYMMDD-HHMMSS.log”文件中。该选项默认使用 verbose 的日志级别。

设置环境变量“FFREPORT”为任意值均可达到同样的效果。如果值为‘:’分隔开的‘key=value’序列，这些选项将影响上报；必须避免选项值里包含特殊字符或选项分隔符‘:’（详见 ffmpeg-utils 说明文档中的‘引用和逃脱’）。下面的是可识别的选项：

file

设置上报所用的文件名。%p 扩展为程序名，%t 扩展为时间戳，%%扩展为%。

level

设置日志级别

解析环境变量时的错误信息不是致命信息，因此不会出现在上报中。

-hide_banner

禁止打印标语

所有 FFmpeg 工具都会打印版权说明、编译选项和库版本。这个选项可以禁止打印这些信息

-cpuflags flags （global）

设置和清除 cpu 标志。这个选项用于测试。不要使用这个选项，除非你清楚自己在干嘛。

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

可用的一些标志：

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

用基准问题测试所有 OpenCL 设备，并显示结果。该选项仅可用于带‘--enable-opencl’编译的 FFmpeg

-opencl_options options (global)

设置 OpenCL 环境选项。可用于带‘--enable-opencl’编译的 FFmpeg。

选项必须为‘:’分割的‘key=value’列表。参见 ffmpeg-utils 说明文档中‘OpenCL Options’章节获取所有支持的选项的列表。

AVOptions

AVOptions 由 libavformat、libavdevice 和 libavcodec 库提供。通过-help 来获取所有可用的 AVOptions。结果被分为两类:

generic

该选项可被用于任意容器、编解码器和设备。容器/设备的通用选项显示在 AVFormatContext 选项下面，编解码器的选项显示在 AVCodecContext 下面。

private

这些选项是特定于某一个容器、设备和编解码器的。私有选项显示在相应容器/设备/编解码器下面。

例如，使用 MP3 复用器的私有选项 id3v2_version，写一个 ID3v2.3 头部到一个 MP3 文件，而不是默认的 ID3v2.4：

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

所有的编解码器的 AVOptions 都是作用于某一个流的，因此需要使用流指定符。

注意：-nooption 语法不能用于布尔型的 AVOptions，请使用-option 0/-option 1。

注意：老版本中使用 v/a/s 加在选项名前面来指定流的 AVOptions，这种做法被弃用。

主要选项

-f fmt (input/output)

设定输入输出文件的格式。输入文件的格式会被自动检测到，而输出文件通常通过文件扩展名来识别，因此这个选项在大部分情况下都不需要。

-i filename (input)

输入文件名

-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，不要覆盖文件，直接退出程序

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

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

为一个或多个流选择一个编码器（当用于输出文件前面）或一个解码器（当用于输入文件前面）。codec 是解码器/编码器的名称，或者一个特殊的值“copy”（仅对于输出文件）表明这个流无须重新编码。

例如：

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

用 libx264 编码所有视频流，拷贝所有音频流。

对于每一个流，最后一个匹配的“c”选项将被匹配，因此：

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

将拷贝第二个视频流以外的所有流，第二个视频流将通过 libx264 编码，第 138 个音频流将通过 libvorbis 编码。

-t duration (input/output)

当作为一个输入选项使用（在“-i”前面）时，将限制从输入文件读入的数据

的时长。

当作为一个输出选项使用（在一个输出文件名前面）时，一旦输出文件长度已经达到指定值，停止写输出文件。

时长可以为秒数，或者“hh:mm:ss[.xxx]”的格式。

-to 和-t 是互斥的，-t 优先级更高。

-to position (output)

当达到 position 位置时，停止写输出文件。position 可以是秒数，或者“hh:mm:ss[.xxx]”的格式。

-to 和-t 是互斥的，-t 优先级更高。

-fs limit_size (output)

设置文件大小，单位为字节。

-ss position (input/output)

当作为一个输入选项使用（在“-i”前面）时，将输入文件 seek 到 position 位置。注意，对于大多数媒体格式，程序不能 seek 到精确的位置，因此 ffmpeg 会 seek 到 pisition 之前最近的 seek 点。当转码时，并且-accurate_seek 被使能（默认会使能），seek 点和 position 之间的片段会被解码，然后丢弃掉。当执行流拷贝，或者使用了-noaccurate_seek 选项，这个片段会被保留。

当作为一个输出选项使用（在一个输出文件名前面）时，解码器会丢弃输入的数据，直到时间戳达到了 position 位置。

position 可以为秒数，或者“hh:mm:ss[.xxx]”的格式。

-itsoffset offset (input)

设置输入文件的时间偏移。

offset 必须是一个时间长度，详见 ffmpeg-utils 说明文当中的时间长度章节。

offset 会被添加到输入文件的时间戳。给定一个正的 offset 意味着相应的流会被推迟 offset 指定的时间。

-timestamp date (output)

设置容器中的记录时间戳。

data 必须是一个时间长度，详见 ffmpeg-utils 说明文当中的时间长度章节。

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

设置 metadata 中的 key/value 对

一个可选的 metadata 指定符可以用于设定流或 chapters 的 metadata。详细信息请参考“-map_metadata”的说明文档。

这个选项会覆盖“-map_metadata”所设定的 metadata 值。通过设置为空，可以清除 metadata 信息。

例如，设定输出文件的 title 信息：

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)

指定目标文件的类型（“vcd”，“svcd”，“dvd”，“dv”，“dv50”）。type 可加上“pal-”，“ntsc-”或者“film-”前缀，以便使用相关的标准，然后所有的格式选项（比特率、编解码器、缓冲大小）都会被自动设定，你可如下面一样输入命令：

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

不过，如果你知道某些选项不会与标准冲突，那你也可以添加这些选项，如：

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

-dframes number (output)

设置需要记录的数据帧数，等同于“-frames:d”

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

当写入的帧数据达到了 framecount，停止写入

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

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

使用固定质量标准（动态比特率）。q/qscale 的意义取决于编解码器。如果 qscale 后面没有带上流描述符，那它仅仅作用于视频流，这是为了与以前版本的行为保持兼容，另外，在没有流描述符的情况下，为音频编解码器和视频编解码器指定相同的编解码器指定值，通常都不是我们正真想要的。

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

创建 filtergraph 指定的滤器图，并用来对流进行滤波。

滤器图描述了用来处理流的滤波器，它必须拥有唯一的输入和输出，且类型和要处理的流相同。在滤器图中，“in”代表输入，“out”代表输出。关于滤器图语法的详细信息请参考 ffmpeg-filters 手册。

如果您想创建一个包含多个输出和输出的滤器图，请看考-filter_complex 选项。

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

该选项和-filter 相似，唯一的不同之处就是该选项的参数是一个文件名，在这个文件中包含了滤波器图标的描述。

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

为匹配的流指定预设。

-stats (global)

打印编码进度/统计信息。这是默认的行为，可以通过”-nostats“选项禁用此

功能。

-progress url (global)

发送程序可识别的进度信息到 url

进度信息是按秒写入的，编码结束的时候也会写入。进度信息由包含“key=alue”的行组成。key 只包含字母和数字。一段进度信息的最后一个 key 总是为“progress”

-stdin

启用标准输入的交互功能。默认只有标准输入才能作为输入。你可以通过“-nostdin”选项来明确禁用标准输入的交互功能。例如，ffmpeg 在后台进程组中。也可以通过命令“ffmpeg ... < /dev/null”来实现相同的功能，但这需要一个 shell。

-debug_ts (global)

打印时间戳信息，默认情况下是关闭的。这个选项主要用于测试和调试，而且由于不同版本之间的输出格式会有不同，因此不能用在可移植的脚本中。

参见”-fdebug ts“选项。

-attach filename (output)

在输出文件中添加附件。这个选项只被某些格式支持，如 Matroska 格式在渲染字幕时用到的字体。附件以指定类型的流实现，因此这个选项会在文件中添加一个新的流，接下来我们就可以通过常用的方式，对这个新添加的流使用“pre-stream”选项。附件会在其它的流（如通过“-map”或者自动映射创建的流）都创建之后再创建出来。

注意，对于 Matroska 你需要设置 metadata 标签 mimetype ：

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

（假设附件流是输出文件中的第三个流）。

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

提取匹配的附件流到名为 filename 的文件中。如果 filename 为空，那么就使用 metadata 中名为 filename 的 tag 所对应的值。

例如，提取第一个附件流到文件‘out.ttf’中：

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

提取所有的附件流到“filename”tag 所决定的文件中：

ffmpeg -dump_attachment:t "" -i INPUT

技术提示 --附件其实属于编解码器的额外数据，因此这个选项也可以用于任何流的额外数据，而不仅仅是附件。

视频选项

-vframes number (output)

设置将要记录的视频帧数，等价于“-frames:v”。

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

设置帧率（Hz、分数或缩写词）

作为一个输入选项，忽略文件中的时间戳，代之以通过帧率计算出的时间戳。这个选项和某些格式（image2 或 v4l2）所使用的-framerate 是不同的（在以前的版本中，他们是一样的）。如果有疑问，请使用-framerate，不要用输入选项-r。

作为一个输出选项，复制或丢弃输入帧，以得到恒定的输出帧率。

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

设置帧大小。

用作一个输入选项时，这是私用选项“video_size”的快捷方式。这个选项可被某些解复用器识别，因为帧大小信息要么没有存储在文件中，要么帧大小信息是可配置的 -- 如原始视频或视频采集器。

用作一个输出选项时，这个选项会在滤波器图标的末尾插入一个‘scale’视频滤波器。请直接使用‘scale’滤波器来插入到滤波器图标的开始或其它地方。

格式为 wxh（默认与源相同）

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

根据 aspect 来设置视频的显示比例。

aspect 可以是浮点数形式的字符串，或者形如“num:den”的字符串，其中 num 和 den 分别为显示比例的分子和分母，如“4:3”，“16:9”，“1.3333”和“1.7777”，都是合法的参数值。

如果该选项和-vcodeccopy 一起使用，将会影响到存储在容器级别的显示比例值，但不会影响到存储在已编码帧的显示比例值，如果存在这个值的话。

-vn (output)

禁止记录视频。

-vcodec codec (output)

设置视频编解码器，等价于‘-codec:v’

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

选择通道值（1 或 2）。该选项用于双通道视频编码。在通道 1 中，视频的统计信息被写入一个日志文件中（参见‘-passlogfile’选项），然后通道 2 根据日志文件生成指定比特率的视频。在通道 1 中，您可以禁用音频，将其输出到 null 设备，windows 和 unix 中的例子分别如下：

```
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y NUL
ffmpeg -i foo.mov -c:v libxvid -pass 1 -an -f rawvideo -y /dev/null
```

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

设置双通道日志文件的文件名前缀为 prefix，默认的文件名前缀为

“ffmpeg2pass”。完整的文件为 PREFIX-N.log，其中 N 为指定输出流的数字。

-vf filtergraph (output)

创建 filtergreph 指定的滤器图，并用来处理流。

等价于“-filter:v”。

高级视频选项

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

设置像素格式。“-pix_fmts”可用来显示所有支持的像素格式。如果选择的像素格式无法被选中，ffmpeg 将打印警告信息，并且选者编码器支持的最佳像素格式。如果 pix_fmt 加上了‘+’前缀，而指定的格式无法选中，且滤器图内部的自动转码功能被禁用，ffmpeg 会出错并退出。如果 pix_fmt 只是单独的一个“+”，ffmpeg 会选择和输入（或滤器图有的输出）相同的像素格式，并且禁用自动转码功能。

-sws_flags flags (input/output)

设置 SwScaler 标志。

-vdt n

丢弃阈值。

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

指定间隔内的速率控制覆盖，格式为斜线分隔开的“int,int,int”列表。前两个值未起始和终止帧序号，最后一个如果为正，表示要使用的量化器，如果为负，表示质量因数。

-ilme

强制编码器（仅仅对 MPEG-2 和 MPEG-4）支持隔行扫描。如果输入文件是隔行扫描的，并且为了最小化失真而保持隔行扫描，请使用这个选项。另一个方法是使用‘-deinterlace’对输入流去隔行扫描，但是这样会引入失真。

-psnr

计算压缩帧的 PSNR。

-vstats

将视频编码的统计信息写入 vstats_HHMMS.log。

-vstats_file file

将视频编码的信息写入文件 file 中。

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

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

-dc precision

Intra_dc_precision.

-vtag fourcc/tag (output)

强制视频标签/四字符码。等价于“-tag:v”。

-qphist (global)

显示 QP 直方图

-vbsf bitstream_filter

已弃用，请参考“-bsf”。

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

强制指定时间点的帧为关键帧，更精确的表述是在每一个指定时间点后面的第一个帧。

如果参数带上了‘expr:’前缀，前缀字符串将被解释为一个表达式，并被用来对每一帧进行评估。如果评估结果不是 0，那么该帧就是关键帧。

如果其中一个 time 形如“‘chapters”[delta]”，它将被按秒移动 delta，结果被作为文件中所有章节的起点。这个选项能够确保一个 seek 点是在一个章节处，或输出文件中任何设计的地方。

例如，为了在 5 分的地方插入一个关键帧，在每个章节的前 0.1s 出加上关键帧

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

expr 指定的表达式可以包含下面的常数：

n　当前正在处理的帧的序号，从 0 开始。

n_forced　强制帧的数目。

prev_forced_n
　前一次强制帧的数目，如果没有强制过，结果为“NAN”

prev_forced_t
　前一次强制帧的时间，如果没有强制过，结果为“nan”

t　当前处理的帧的时间。

For example to force a key frame every 5 seconds, you can specify:
例如，每隔 5s 强制一个关键帧：

```
-force_key_frames expr:gte(t,n_forced*5)
```

在上一个强制帧后 5s 强制一个关键帧，从 13s 开始：

-force_key_frames
expr:if(isnan(prev_forced_t),gte(t,13),gte(t,prev_forced_t+5))

注意，强制了过多的强制帧不利于特定编码器的搜寻算法：利用‘fixed-GOP’或相似的选项会更有效率。

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

在执行流拷贝时，同样拷贝开始处的非关键帧。

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

使用硬件加速来解码匹配的流，允许的值为：

none

不要使用任何硬件加速（默认）。

auto

自动选择硬件加速方法。

vda 使用苹果 VDA 硬件加速。

vdpau

使用 VDPAU（Video Decode and Presentation API for Unix）硬件加速。

dxva2

使用 DXVA2 硬件加速。

当选择的解码器不支持选择的硬件加速，或选择的硬件加速不可用，该选项将失效。

注意，大部分硬件加速都是用于播放的，而且不会比现代 CPU 的软件加速更快。另外，ffmpeg 会经常需要从 GPU 内存拷贝解码后的帧到系统内存，导致后续的性能下降。因此，该选项主要用于测试。

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

选择一个用于硬件加速的设备。

该选项是在-hwaccel 选项被指定的前提下才有意义。该选项的具体意义取决于所选择的硬件加速方法。

vdpau

对于 VDPAU，该选项会指定使用 X11 显示/屏幕。如果没有指定该选项，将会使用环境变量 DISPLAY 指定的设备。

dxva2

对于 DXVA2，该选项应当包括需要使用的显示适配器的数目。如果没有指定该选项，将使用默认的适配器。

音频选项

-aframes number (output)

Set the number of audio frames to record. This is an alias for "-frames:a".

设置将要记录的音频帧的数目，等价于‘-frames:a’。

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

Set the audio sampling frequency. For output streams it is set by default to the frequency of the corresponding input stream. For input streams

this option only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

设置音频采样频率。输出流的采样频率默认设置为相应输入流的输入频率。对于输入流，该选项仅对音频抓取设备和原始解复用器有意义，会被映射到相应解复用器的选项。

-aq q (output)

Set the audio quality (codec-specific, VBR). This is an alias for -q:a.

设置音频质量（codec-specific，VBR），等价于‘-q:a’。

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

Set the number of audio channels. For output streams it is set by default to the number of input audio channels. For input streams this option

only makes sense for audio grabbing devices and raw demuxers and is mapped to the corresponding demuxer options.

设置音频声道数。输出流的音频声道数默认设置为相应的输入流的音频声道数。对于输入流，该选项仅对音频抓取设备和原始解复用器有意义，会被映射到相应解复用器的选项。

-an (output)

禁用录音。

-acodec codec (input/output)

设置音频编解码器，等价于‘-codec:a’。

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

设置音频采样格式，‘-sample_fmts’可显示所支持的采样格式列表。

-af filtergraph (output)

根据 filtergraph 创建滤波器图标，并用来处理流。等价于“-filter:a”，详见-filter 选项。

高级音频选项

-atag fourcc/tag (output)

强制音频标签/四字符码，等价于“-tag:a”。

-absf bitstream_filter

丢弃，参见‘-bsf’

-guess_layout_max channels (input,per-stream)

如果一些输入通道布局未知，仅在通道数不大于 channels 情况下猜测。如，channels 为 2 时，ffmpeg 会把 1 个通道识别为 mono，2 个通道识别为 stereo，但是 6 个声道不会识别为 5.1。默认情况下总是会猜，将 channels 设为 0 可禁用此功能。

字幕选项

-scodec codec (input/output)

设置字幕编解码器，等价于‘-codec:s’。

-sn (output)

禁止字幕记录。

-sbsf bitstream_filter

弃用，参见'-bsf'

高级字幕选项

-fix_sub_duration

固定字幕长度。对于每一个字幕，等待同一个流中的下一个包，调整第一个的时长以避免重叠。

这对于一些字幕编解码器是很有必要的，尤其是 DVB 字幕，因为原始包仅仅包含了粗略的估计值，而且用空字幕帧来标记结尾。当使用该选项出现失败，会导致夸张的时长，或者由非单调时间戳导致复用失败。

注意，该选项会延迟所有的数据输出，直到下一个字幕包被解码出来：这会导致内存消耗和延迟。

-canvas_size size

设置用于渲染字幕的画布尺寸。

高级选项

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

指定一个或多个流作为输出文件的源。每一个输入流是通过输入文件的索引 input_file_id 和输入文件中的流索引 input_stream_id 来识别的。这两个索引都是从 0 起算的。如果指定了，sync_file_id:stream_specifier 将设置用于演示同步参考的流。

命令行上的第一个“-map”选项指定了第 0 个输出流的源，第二个“-map”选项指定了第 1 个输出流的源，以此类推。

在流标识符前加“-”意味着“negative（负的）”映射。它会将已创建的映射

的流映射禁用掉。

linklabe 可以将复杂滤波器图表的输出映射到输出文件。linklabel 必须对应于图表中定义的输出连接标签。

例如，将第一个输入文件的所有流映射到输出：

ffmpeg -i INPUT -map 0 output

例如，如果在第一个输入文件中有两个音频流，分别用“0:0”，“0:1”表示。我们可以通过“-map”来选择哪一个流输出到输出文件中。如：

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

将把 INPUT 文件中“0:1”所表示的输入流映射到 out.wav 中的一个输出流中。

例如，将输入文件 a.mov 中索引为 2 的流（表示为“0: 1”）和输入文件 b.mov 中索引为 6 的流（表示为“1:6”）拷贝到输出文件 out.mov 中：

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

选择输入文件中所有视频和第三个音频流：

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

映射除第二个音频以外的所有流，使用负映射：

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

注意，使用这个选项将禁用输出文件的默认映射。

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

映射输入中的一个声道到输出。如果没有指定 output_file_id.stream_specifier，将会把所有音频流的声道都映射过去。

“-1”表示将静音声道映射过去。

例如，假设 INPUT 是一个立体声音频文件，可以通过下面的命令转换两个声道：

ffmpeg -i INPUT -map_channel 0.0.1 -map_channel 0.0.0 OUTPUT

如果你想把第一个声道静音，只保留第二个声道：

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

“-map_channel”的顺序决定了输入流的声道顺序。输出声道的布局是根据映射的声道数目来定的(一个“-map_channel”选项表示单声道,两个表示立体声,以此类推)。联合使用“-ac”和“-map_channel”将在输入和输出的声道布局不匹配时更新声道增益水平（例如两个“-map_channel”和“-ac
6”)。

我们也可以提取输入中到每一个声道到指定的输出中。下面的命令提取了INPUT 音频流(第 0 个文件,第 0 个流)的两个声道到输出的 OUTPUT_CH0 和 OUTPUT_CH1 中：

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

下面的例子将一个立体声的声道分割到同一个输出文件的两个流中：

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

注意，目前每一个输出流只能包含同一个输入流。例如，我们不能通过“-map_channel”选项从不同的流中（相同或不同的文件中的流）提取多个输入音频声道，然后合并到一个输出流中。同样，我们也不能将两个单声道流合并成一个立体声流。不过，我们可以将一个立体声流分割成另个单声道流。

如果你确实需要上面的功能，一个可行的方案是使用 amerge 滤波器。例如，如果你想将 input.mkv 中的两个单声道流合并成一个立体声流（视频保持不变），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)

设置 infile 后面的输出文件的 metadata 信息。注意 infile 是文件的索引，不是文件名。metadata_spec_in/out 参数指定了要拷贝的 matadata。metadata 描述符有如下形式：

g　全局 metadata，应用于整个文件的 metadata。

s[:stream_spec]

流 metadata。stream_spec 是流描述符中所讲的流描述符。在输入 metadata

描述符中，第一个匹配的流将被拷贝。在输出 metadata 描述符中，所有匹配的流将获得拷贝。

c:chapter_index

章节 metadata。chapter_index 是基于 0 的章节索引。

p:program_index

程序 metadata。program_index 是基于 0 的程序索引。

如果忽略 metadata 描述符，默认为全局描述符。

默认情况下，全局 metadata 从第一个输入文件中拷贝，流 metadata 和章节 metadata 随着流/章节一起拷贝。创建相关类型的映射将禁用默认映射。一个负的文件索引将创建一个虚拟映射，这会禁用自动拷贝。

例如，将输入文件的第一个流的 metadata 拷贝到输出文件中的全局 metadata：

ffmpeg -i in.ogg -map_metadata 0:s:0 out.mp3

拷贝全局 metadata 到所有音频流：

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

注意，这个例子中，0 也可以实现相同功能，因为默认的类型是全局 metadata。

-map_chapters input_file_index (output)

拷贝索引为 input_file_index 的输入文件的章节到下一个输出文件。如果没有指定章节索引，那么会从第一个输出文件中拷贝至少一个章节。一个负的文件索引可以禁用任何章节拷贝。

-benchmark (global)

在编码结束时显示标记信息。显示 CPU 使用时间和最大内存消耗。并不是所有系统都支持最大内存消耗，如果不支持，将显示为 0.

-benchmark_all (global)

.

在编码过程中显示标记信息。在各个过程步骤中显示 CPU 使用时间（音频/视频 编码/解码）

-timelimit duration (global)

当 ffmpeg 运行了 duration 秒后退出 ffmpeg。

-dump (global)

将每一个输入包输出到标准错误输出。

-hex (global)

输出包时输出负荷情况。

-re (input)

以本地帧率读取输入。主要用来模拟一个采集器或实时输入流（例如从一个文件中读取）。不能和真实的采集器或实时输入流一起使用，否则将导致包的丢失。ffmpeg 默认是尽可能快地读取输入。这个选项会将读取输入的速度降低到输入的本地帧率，这对实时输出有用（如现场直播）

-loop_input

遍历输入流。目前仅对图片流有效。这个选项用于 FFserver 自动测试。该选项已被弃用，请使用-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 表示无限循环输出），该选项已被弃用，请使用-loop。

-vsync parameter

视频同步方法。处于兼容性的考虑，旧的值可以指定为数字。新的值必须总是指定为字符串。

0, passthrough

每一帧都是和它的时间戳一起从解复用器传给复用器。

1, cfr

帧将被重复并丢弃，以达到要求的帧率。

2, vfr

帧将被带着时间戳一起传送或者丢弃，以避免两个帧有相同的时间戳。

drop

传送但破坏所有时间戳，使得复用器可以基于帧率产生新的时间戳。

-1, auto

基于复用器性能，在 1 和 2 之间选择，这是默认方法。

注意，经过这个处理，时间戳可能会被复用器改变。例如，avoid_negative_ts 被使能的时候。

通过-map 选项，你可能选择从哪一个流获取时间戳。你可能使视频或音频保持不变，然后使其它的流与之同步。

-async samples_per_second

音频同步方法。“伸展/压缩”音频流来匹配时间戳，参数是音频在 1 秒内改变的最大采样数。-async 1 是一个特殊情况，只有音频开始的地方会修正，此后不会再有修

正。

注意，在这个处理之后，时间戳可能会被复用器改变。例如，avoid_negative_ts 被使能的时候。

该选项已经被弃用，请使用“aresample”音频滤波器。

-copyts

不要处理输入时间戳，保留他们的值。尤其不要删除初始时间的偏移值。

注意，取决于 sync 选项或复用器的处理（例如格式选项 avoid_negative_ts 被使能），输出时间戳可能和输入时间戳不匹配。

-copytb mode

指定流拷贝时如何设置编码器时间戳。mode 是一个整数值，并且可以假设为下面其中的一个值：

1 使用解复用器时基

拷贝相应输入解复用器的时基到输出编码器。当以可变帧率拷贝视频流时，有时需要避免非单调地增加时间戳。

0 使用解码器时基

拷贝相应输入解码器的时基到输出编码器。

-1 试图自动选择，以得到合理的输出。

默认值是-1.

-shortest (output)

当最短的输入流结束了，结束编码。

-dts_delta_threshold

时间戳不连续 delta 阈值。

-muxdelay seconds (input)

设置最大的解复用器-解码器延时。

-muxpreload seconds (input)

设置初始解复用器-解码器延时。

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

给一个输出流分配一个新的 stream-id。这个选项要在它要作用的输出文件名的前面指定。当有多个输出文件时，一个 streamid 可能被分配其它文件名。

例如，给一个输出传输串流文件的 strean0 设置 PID 为 33，stream1 设置 PID 为 36：

```
ffmpeg -i infile -streamid 0:33 -streamid 1:36 out.ts
```

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

为匹配的流设置比特流滤波器。比特流滤波器是一个由逗号分隔的比特流滤波器列表。“-bsfs 选项”可得到比特流滤波器列表。

ffmpeg -i h264.mp4 -c:v copy -bsf:v h264_mp4toannexb -an out.h264

ffmpeg -i file.mov -an -vn -bsf:s mov2textsub -c:s copy -f rawvideo sub.txt

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

为匹配的流分配一个标签/四字符码。

-timecode hh:mm:ssSEPff

为写入操作指定时间码。对于非下降时间码，SEP 是‘:’；下降时间码是“;”（或者'.')。

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

-filter_complex filtergraph (global)

定义一个复杂滤器图，即一个带有任意数量输入输出的滤波器图表。对于简单图表，即有类型相同的单输入和输出的图表，请参考‘-filter’选项。滤器图是对滤波器图表的描述，如 ffmpeg-filter 手册中‘滤波器图表语法’章节所描述的那样。

必须使用"[file_index:stream_specifier]"语法为输入流指定输入连接标签（即，和 -map 一样）。如果有多个流和流描述符匹配，将使用第一个流。没有加上标签的输入将和第一个类型匹配且未使用的流连接上。

输出连接标签通过-map 引用。未加标签的输出将被添加到第一个输出文件中。
注意，通过该选项，可以仅使用 lavfi 源，无需普通输入文件。
例如，在视频上覆盖一张图片：

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

这里，“[0:v]”引用了第一个输入文件的第一个视频流，这个流被连接到覆盖滤波器的第一个输入。同样，第二个文件的第一个视频流被连接到覆盖滤波器的第二个输入。

假设每个输入文件仅包含一个视频流，我们可以忽略输入标签，因此上面的命令等价于：

ffmpeg -i video.mkv -i image.png -filter_complex 'overlay[out]' -map '[out]' out.mkv

此外，我们可以忽略输出标签，滤波器图表的唯一输出将被自动添加到输出文件中，因此，我们可以简单写成：

```
ffmpeg -i video.mkv -i image.png -filter_complex 'overlay' out.mkv
```

为了生成 5s 的纯红色视频，使用 lavfi “color”源：

```
ffmpeg -filter_complex 'color=c=red' -t 5 out.mkv
```

-lavfi filtergraph (global)

定义一个复杂滤波器图表，即带有任意数量输入和输出的图表。等价于 -filter_complex

-filter_complex_script filename (global)

This option is similar to -filter_complex, the only difference is that its argument is the name of the file from which a complex filtergraph

description is to be read.

这个选项和-filter_complex 相似，唯一不同的地方是，该选项的参数是一个文件名，该文件包含了对复杂滤波器图表的描述。

-accurate_seek (input)

这个选项用来使能/禁用当-ss 存在时对输入文件的精确 seek 操作。默认是使能的，所以在转码过程中，seek 操作是精确的。-noaccurate_seek 可禁用精确 seek，这在拷贝一些流的过程中转码其它的流时很有用。

-override_ffserver (global)

覆盖 ffserver 的输入描述。利用这个选项，我们可以将任何输入流映射到 ffserver，并且可以控制 ffmpeg 的很多编码性能。如果没有这个选项，ffmpeg 将会把任何 ffserver 想要的东西传给它。

在某些情况下，某些特性无法指定给 ffserver，但却可以指定给 ffmpeg，该选项就是为此而设计的。

-discard (input)

允许在解复用器端丢弃指定流或流里面的帧。不是所有解复用器都支持这个选项。

none

不丢弃任何帧。

default

默认情况，不丢弃任何帧。

noref

丢弃所有未被引用的帧。

bidir
丢弃所有双向参考帧。

nokey
丢弃所有非关键帧。

all 丢弃所有帧。

有一个例外情况，我们可以将一个位图字幕流作为输入：这个流会被转化为和文件中最大视频一样尺寸的视频，或者 720x576 如果没有视频。注意，这只是临时性的试验性的方法，一旦 libavfilter 对字幕有了适当的支持，这个方法会被移除。

例如，为了硬编码 MPEG-TS 中存储的 DVB-T 记录上的字幕，将字幕延迟 1s：

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

（0x2d0, 0x2dc 和 0x2ef 分别为 MPEG-TS 中视频、音频和字幕流的 PID；0:0, 0:3 和 0:7 也可以正常工作）

预置文件

一个预置文件包含了‘option=value’对的序列，每行一个，指定了那些不适合写在命令行上的选项。以‘#’开头的行为注释。可以在 ffmpeg 源代码的 presets 目录想查看相关例子。

预置文件由"vpre", "apre","spre"和"fpre"选项指定。“fpre”选项可用于任意类型的编解码器，它以预置文件的文件名，而非预置名作为输入。预置文件中的"vpre", "apre"和"spre"选项作用于当前选择的和预置选项相同类型的编解码器。

传给预置选项"vpre", "apre"和"spre"的参数根据以下规则确定要使用的预置文件：

首先 ffmpeg 会依次在以下目录中寻找名为 arg.ffpreset 的文件：$FFMPEG_DATADIR（如果已经设置了的话）、HOME/.ffmpeg、软件配置时定义的数据目录（通常为 PREFIX/share/ffmpeg）、或者 win32 系统中的可运行程序所在的 ffpresets 目录。例如，假设参数为"libvpx-1080p"，ffmpeg 将寻找 libvpx-1080p.ffpreset 文件。

如果没有找到，ffmpeg 会再从以上目录中寻找名为 codec_name-arg.ffpreset 的文件，codec_name 是要应用预置文件选项的编解码器的名字。例如，假设我们用"-vcodec libvpx"选项选择了视频编解码器，并且指定了预置文件选项"-vpre 1080p"，那 ffmpeg 将会寻找名为 libvpx-1080p.ffpreset 的文件。

温馨提示

- 对于比特率很小的流，请使用低帧率和小尺寸的 GOP。尤其对于 linux 播放器无法快速处理的 RealVideo，播放器可以丢帧。请看下面的例子：

  ffmpeg -g 3 -r 3 -t 10 -b:v 50k -s qcif -f rv10 /tmp/b.rm

- 编码过程中显示出来的参数‘q’是当前量化器。如果值为 1，表示能够达到很高的质量，如果为 31，意味着最差的质量。如果经常出现 q=31，这表明编码器无法充分按照我们的比特率来压缩数据。此时，您要么增加比特率，减小帧率，要么减小帧的尺寸。

- 如果您的电脑不够快，您可以降低压缩率以提高压缩速度。您可以使用'-me zero'来提高运动估计速度，或者'-g 0'来完全禁用运动估计（这样将只有 I 帧，意味着和 JPEG 一样的压缩效果）

- 为了降低音频比特率，应该降低采样频率（MPEG 音频降到 22050Hz，AC-3 降到 22050 或者 11025）。

- 为了获得恒定质量（但是可变的比特率），请使用选项'-qscale n'，其中 n 介于 1（最佳质量）和 31（最差质量）之间

例子

预置文件

一个预置文件包含了‘option=value’对的序列，每行一个，指定了一些同样可以写在命令行上的选项。以‘#’开头的行为注释，会被忽略。空行也将被忽略。可以在 ffmpeg 源代码的 presets 目录想查看相关例子。

预置文件通过选项“pre”指定，这个选项以一个预置名称为输入。

ffmpeg 依次在以下目录中寻找名为 preset_name.avpreset 的文件：$AVCONV_DATADIR（如果已经设置了的话）、HOME/.ffmpeg、软件配置时定义的数据目录（通常为 PREFIX/share/ffmpeg）。例如，假设参数为"libx264-max"，ffmpeg 将寻找 libx264-max.avpreset 文件。

音视频抓取

如果我们指定了输入格式和输入设备，ffmpeg 就能直接抓取音视频。

ffmpeg -f oss -i /dev/dsp -f video4linux2 -i /dev/video0 /tmp/out.mpg

或者一个 ALSA 音频数据源（单声道输入，音频卡号 1），而不是 OSS:

ffmpeg -f alsa -ac 1 -i hw:1 -f video4linux2 -i /dev/video0 /tmp/out.mpg

注意，在电视上（例如 Gerd Knorr 的 http://linux.bytesex.org/xawtv/）启动 ffmpeg 之前，您必须已经激活了正确的视频数据源和通道。您同样需要给标准混频器设置合适的录音级别。

X11 抓取

通过 ffmpeg 抓取 X11 显示

ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0 /tmp/out.mpg

.

0.0 是您的 X11 服务器的 display.sreen，和环境变量 DISPLAY 的值相同。

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

0.0 是您的 X11 服务器的 display.sreen，和环境变量 DISPLAY 的值相同。10 是抓取时的 x-offset，20 是 y-offset。

音视频文件格式转换

任何支持的文件格式和协议都可作为 ffmpeg 的输入。

例子：

- 您可以将 YUV 文件作为输入：

  ffmpeg -i /tmp/test%d.Y /tmp/out.mpg

  ffmpeg 将使用下面的文件：

  /tmp/test0.Y, /tmp/test0.U, /tmp/test0.V,
  /tmp/test1.Y, /tmp/test1.U, /tmp/test1.V, etc...

Y 文件使用了两倍于 U 和 V 文件的分辨率。它们都是原始文件，没有文件头。所有标准视频解码器都能生成这些文件。如果 ffmpeg 无法猜出图像的尺寸，您必须用‘-s’选项来指定。

- 您可以将一个原始的 YUV420P 文件作为输入：

  ffmpeg -i /tmp/test.yuv /tmp/out.avi

test.yuv 是一个包含了原始 YUV 二维数据的文件。每一帧都由 Y 平面及后面跟着的 U 和 V 平面组成，其中 U 和 V 平面的横向和纵向分辨率都为 Y 平面的一半。

- 您可以输出到一个原始 YUV420P 文件：

  ffmpeg -i mydivx.avi hugefile.yuv

- 您可以设定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

  将音频文件 a.wav 和原始 YUV 视频文件 a.yuv 转换成 MPEG 文件 a.mpg。

- 您同样可以同时执行音频和视频转化：

  ffmpeg -i /tmp/a.wav -ar 22050 /tmp/a.mp2

  将 a.wav 以采样率 22050 转换成 MPEG 音频。

- 您可以同时编码成多种格式，并且定义一个从输入流到输出流的映射：

  ffmpeg -i /tmp/a.wav -map 0:a -b:a 64k /tmp/a.mp2 -map 0:a -b:a 128k /tmp/b.mp2

  将 a.wav 分别以 64kbits 和 128kbits 转换到 a.mp2 和 b.mp2。‘-map file:index’指定了哪一个输入流如何以输出流定义的顺序用到输出流。

- 您可以转码加密视的频对象组：

  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

  这是一个典型的分割 DVD 的例子；输入是一个 VOB 文件，输出是一个视频为 MPEG-4、音频为 MP3 的文件。注意，在这个命令中，我们使用了 B 帧，这样可以使 MPEG-4 流兼容于 DivX5，GOP 大小为 300，这表明对于 29.97fps 的输入视频，每隔 10s 一个帧。当转码 DVD 以获取想要的音频语言时，映射功能是很有用的。

  注意：想要获取支持的输入格式，请使用“ffmpeg -formats”。

- 您可以从视频中提取图像，或者用很多图片生成一个视频：

  从视频中提取图片：

  ffmpeg -i foo.avi -r 1 -s WxH -f image2 foo-%03d.jpeg

  这个命令从视频中每隔 1 秒提取一个视频帧，并输出到文件中，文件名为 foo-001.jpeg、foo-002.jpeg 等等。图片会被重新调整，以适配新的 WxH 值。

如果您只想提取一定数量的帧，可以在上面的命令加入‘-vframes’或'-t'选项，或者加入-ss 选项来指定从某一时间点开始提取。

用图片生成视频：

ffmpeg -f image2 -i foo-%03d.jpeg -r 12 -s WxH foo.avi

The syntax "foo-%03d.jpeg" specifies to use a decimal number composed of three digits padded with zeroes to express the sequence number. It is

the same syntax supported by the C printf function, but only formats accepting a normal integer are suitable.

“foo-%03d.jpeg”表明用三位整数来表示序号，不足三位的以 0 填充。这个语法和 C 语言中的 printf 函数一样，但需要文件格式接受普通整数。

当导入一个图片序列时，通过选择“-pattern_type glob”，-i 选项还支持类似于 shell 的正则表达式。

例如，从符合"foo-*.jpeg"模式的文件名生成一个视频：

ffmpeg -f image2 -pattern_type glob -i 'foo-*.jpeg' -r 12 -s WxH foo.avi

- 您可以将相同类型的多个流放入到输出文件中：

ffmpeg -i test1.avi -i test2.avi -map 1:1 -map 1:0 -map 0:1 -map 0:0 -c copy -y test12.nut

得到的输出文件 test12.nut 将以相反的顺序包含输入文件中的前 4 个流。

- 强制固定比特率的视频输出：

ffmpeg -i myfile.avi -b 4000k -minrate 4000k -maxrate 4000k -bufsize 1835k out.m2v

- lmin, lmax, mblmin 和 mblmax 四个选项使用希腊单位，但我们可以很简单地以 QP2LAMBDA 常数从‘q’单位转换得到。

ffmpeg -i src.ext -lmax 21*QP2LAMBDA dst.ext

令请参阅

ffmpeg-all(1), ffplay(1), ffprobe(1), ffserver(1), ffmpeg-utils(1), ffmpeg-scaler(1),

ffmpeg-resampler(1), ffmpeg-codecs(1),
ffmpeg-bitstream-filters(1), ffmpeg-formats(1), ffmpeg-devices(1), ffmpeg-protocols(1), ffmpeg-filters(1)

作者

FFmpeg 开发者。

关于作者身份的详细信息，请参阅工程的 Git 历史（git://source.ffmpeg.org/ffmpeg），即在 FFmpeg 源代码目录下输入 git log，或者去往<http://source.ffmpeg.org>浏览网上仓库。

特定组件的维护者都列在了源代码目录下的 MAINTAINERS 文件中。