ChatGPT

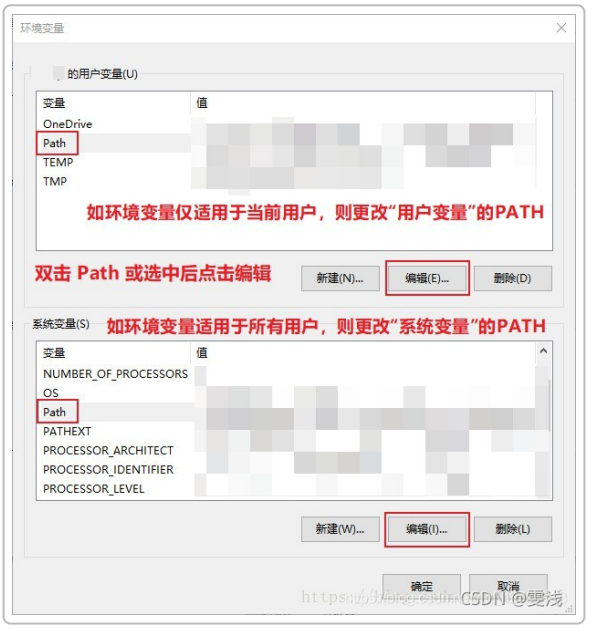
FFmpeg Windows 平台入门教程

FFmpeg（Fast Forward MPEG）是一款开源、跨平台的音视频处理工具，几乎支持所有主流格式的转码、剪辑、合并、抽帧、推流等功能 [1]。它提供了功能强大的命令行程序，可以录制、转换以及流化音视频。本教程将以 Windows 平台为例，从安装配置到核心命令，系统介绍 FFmpeg 的基础用法，帮助初学者快速上手处理常见的音视频任务。

安装与环境配置

步骤 1 – 下载 FFmpeg： 访问 FFmpeg 官网的下载页面，根据系统选择 Windows 64 位版本（一般选择 `win64-gpl` 静态版本，不要选开发者用的 shared 版本） [2]。例如，可下载 `ffmpeg-n7.1-latest-win64-gpl-7.1.zip` 压缩包，然后将其解压。解压后会得到包含 **ffmpeg.exe**, **ffplay.exe**, **ffprobe.exe** 等文件的文件夹，可将该文件夹重命名为 “ffmpeg”，并放置到如 `C:\ffmpeg\` 这样的路径下，便于管理 [3] [4]。

步骤 2 – 配置环境变量： 为了在任意路径使用 FFmpeg，需要将其 **bin** 目录加入系统 PATH 环境变量 [5]。打开 Windows 的“系统属性” > “高级” > “环境变量”，在“系统变量”中找到 **Path** 项并编辑，新建一行填写 FFmpeg 的 `bin` 路径（例如 `C:\ffmpeg\bin`），保存即可 [6]。完成后，打开新的命令提示符 (CMD) 窗口，直接输入 `ffmpeg` 并回车，系统应能找到并运行 FFmpeg 程序。下面两图演示了在 Windows 环境变量中添加 FFmpeg 路径的步骤：



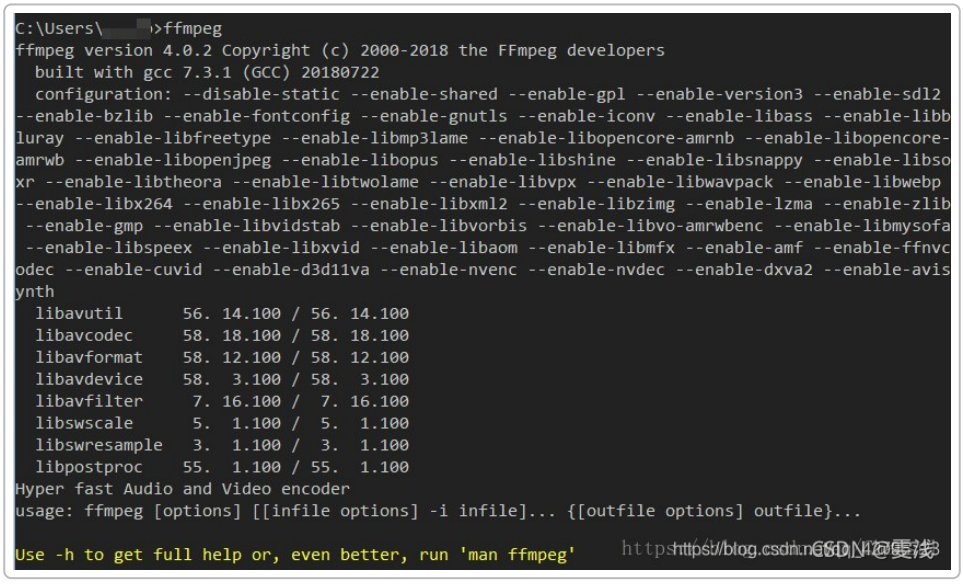
在 Windows 系统的环境变量设置中，将 FFmpeg 的 `/bin` 目录添加进 Path 列表，以便在命令行中随处调用 FFmpeg [7] [8]。

环境变量 Path 添加 FFmpeg 路径示例：新建条目 `D:\Program Files\ffmpeg\bin` 并保存 [9] [10]。完成设置后即可在命令行使用 FFmpeg 命令。

步骤 3 – 验证安装： 打开命令提示符，输入以下命令查看 FFmpeg 是否正常工作：

```
ffmpeg -version
```

若成功，会显示 FFmpeg 版本和编译配置信息等 [11]。例如，下图所示为 FFmpeg 4.0.2 在 Windows 上运行时的输出片段：



在 CMD 中运行 `ffmpeg`，输出显示版本号和支持的编解码等信息，表示 FFmpeg 已安装成功 [12] [13]。

完成以上步骤后，您已经在 Windows 平台成功安装并配置了 FFmpeg，可以在任意目录通过命令行调用 **`ffmpeg`** 来处理多媒体文件了。

基础语法结构

FFmpeg 命令行的基本构成可以表示为：

```
ffmpeg [全局参数] -i <输入文件> [输入/输出参数] <输出文件>
```

如上所示，`ffmpeg` 后接若干选项，再用 `-i` 指定一个输入文件或流，然后可以添加多个参数对媒体进行处理，最后指定输出文件名[14]。FFmpeg 支持同时读取多个输入和输出多个文件（例如为同一输入视频生成不同格式的输出）[15]。对于大多数常见操作，我们只需一个输入一个输出，其基本用法可以总结为：“指定输入 -> 设置处理参数 -> 指定输出”。

例如，将一个 MP4 视频转换为 AVI 格式的基本命令如下：

```
ffmpeg -i input.mp4 output.avi
```

这条命令中，`-i input.mp4` 指定输入文件为 input.mp4，没有额外参数则表示使用默认编码设置，将其转换封装为 AVI 格式输出[16]。再如，将一段 MP3 音频转码为 AAC 音频，可以执行：

```
ffmpeg -i input.mp3 output.aac
```

同理，FFmpeg 会自动根据输出文件扩展名选择合适的编码格式完成转码。需要注意，FFmpeg 有大量可选参数，可以在输入和输出之前分别指定，用于控制编码格式（`-c:v`，`-c:a` 等）、比特率（`-b:v`，`-b:a`）、分辨率（`-s`）、帧率（`-r`）以及各种滤镜效果（`-vf` 视频滤镜，`-af` 音频滤镜）等。在不了解高级参数时，可以先使用默认设置完成任务，然后逐步学习调整参数以满足需求。

核心功能介绍

下面介绍 FFmpeg 最常用的一些核心功能模块，包括格式转码、剪辑拼接、滤镜特效、抽取帧/音频，以及流媒体处理等。每个部分都给出常用命令示例和简要说明，帮助快速掌握基本用法。

转码（格式转换）

转码是指在不同音视频格式之间进行转换。借助 FFmpeg，几乎任何主流的媒体格式都可以相互转化。最简单的格式转换只需指定输入文件和输出文件，FFmpeg 将根据文件扩展名自动选用默认编码器：

- 视频格式转换示例：将 MP4 视频转换为 AVI 视频（视频编码将从 H.264 转为 MPEG-4，音频编码从 AAC 转为 MP3 等，由 FFmpeg 默认配置决定）[16]：

```
ffmpeg -i input.mp4 output.avi
```

类似地，可以将 AVI 转为 MP4，或 MKV 转为 MP4 等，只需更改输出文件名后缀即可，FFmpeg 会自动完成解码和重新编码。

- 音频格式转换示例：将一首 MP3 歌曲转换为 AAC 格式：

```
ffmpeg -i input.mp3 output.aac
```

如果不特别指定音频编码器，FFmpeg 默认使用内置的 AAC 编码器输出 AAC 格式 [17]。也可以通过参数 `-c:a libmp3lame` 等指定特定编码器。

在转换过程中，可以通过参数控制质量和编码方式。例如，使用 `-b:v` 指定视频比特率，`-b:a` 指定音频比特率，或使用 `-c:v libx264 -crf 23` 以恒定质量模式转码等。但对于简单的格式更改，上述基本命令已经足够。**注意：**有些格式容器对编码有要求，例如 MP4 容器通常使用 H.264/AVC 视频和 AAC 音频，如果直接更改后缀不指定编码，FFmpeg 默认设置通常会符合要求 [18]。但若遇到不兼容的情况，可手动指定编码参数使之匹配。

剪切与拼接

剪切（裁剪）和**拼接**是视频编辑中的基本操作。FFmpeg 提供了多种方法来截取片段或合并文件：

- **截取视频片段：** 使用参数 `-ss`（开始时间）和 `-t`（持续时间）可以从输入媒体中裁剪出一段。例如，截取 input.mp4 从第 00:00:10 开始的 30 秒视频片段 [19]：

```
ffmpeg -ss 00:00:10 -i input.mp4 -t 30 -c copy output_clip.mp4
```

以上命令在输入前使用 `-ss` 定位，`-t 30` 指定长度 30 秒，`-c copy` 表示直接复制原编码而不重新编码，以保持质量且处理更快 [20]（注意直接复制要求剪辑点在关键帧上才能精确剪辑，否则可能略有偏差）。如需从视频结尾往前截取，也可使用 `-to` 指定结束时间点。

- **合并多个文件：** FFmpeg 提供 **concat** 功能将多个媒体顺序拼接。常用方法是使用 “concat 模式” 列出文件清单，然后一次性合并 [21] [22]。步骤如下：

- 新建一个文本文件（如 list.txt），内容按行列出待合并文件：

  ```
  file 'part1.mp4'
  file 'part2.mp4'
  file 'part3.mp4'
  ```

- 执行 concat demuxer 合并：

  ```
  ffmpeg -f concat -safe 0 -i list.txt -c copy output.mp4
  ```

  上述命令指定输入格式为 concat（`-f concat`），读取文件列表 list.txt，`-c copy` 直接合并流而不重编码 [23]。**注意：**使用此方式要求待拼接的视频编码参数一致（如分辨率、编码格式），否则应先转码各段使之一致再合并。如果源文件编码不同或需过渡效果，可考虑使用 FFmpeg 的 concat 滤镜 [24]（相对复杂，这里不展开）。

- **音频合并：** 对于音频文件，操作与视频类似。可以用 concat 列表法合并多段音频，或用 `-filter_complex amerge` 混音叠加（区别在于串联 vs 混合）。串联多个音频只需将它们像视频一样列入列表，用 concat 方式合并即可。

通过以上方式，您可以快速剪辑出需要的片段，或将多个片段无缝拼接成一个完整文件。剪辑时若不指定 `-c` 参数，FFmpeg 默认会重新编码输出；如无特殊需要，推荐使用复制模式以避免画质损失和耗时的重新编码[25]。

添加滤镜

FFmpeg 拥有强大的滤镜（filter）系统，可对视频音频流应用各种效果。下面介绍几类常用滤镜操作，包括添加字幕、水印，以及常见视频特效等。

添加字幕

字幕可以分为**外挂字幕**（独立的字幕文件，播放时由播放器加载）和**内嵌字幕**。内嵌字幕又分两种：一种是软字幕（**内封字幕**），将字幕流封装在视频容器中，可选择开关；另一种是硬字幕（**硬嵌字幕**），直接把字幕渲染到视频画面上[26]。FFmpeg 可以实现字幕封装或烧录。

- **内封（软）字幕流：**如果已有视频和字幕文件（如 `.srt` 或 `.ass`），可使用 FFmpeg 将字幕**封装**进视频容器。例如，将 input.mp4 和外挂字幕 input.srt 合成为带字幕轨的 MP4 文件，可以命令[27]：

```
ffmpeg -i input.mp4 -i input.srt -c copy -c:s mov_text output.mp4
```

这里第二个 `-i` 指定字幕文件，`-c copy` 复制视频音频流，`-c:s mov_text` 将字幕流转换为 MP4 支持的 mov_text 格式封装进输出[27]。处理后生成的 output.mp4 包含软字幕轨，播放时可选择显示或隐藏字幕（需使用支持字幕的播放器）。**注意：**如封装 MKV 容器，则应使用 `-c:s srt` 或 `-c:s ass` 对应封装字幕格式[28]。

- **硬字幕（烧录）：**将字幕直接绘制到视频图像上的方法是使用 **subtitles** 滤镜。此方式会把文字渲染到每帧画面中，成为视频内容的一部分。要求 FFmpeg 编译时开启了 `--enable-libass` 支持 ASS/SSA 字体渲染（大多数预编译的 Windows FFmpeg 已支持）。基本用法示例：

```
ffmpeg -i input.mp4 -vf "subtitles=movie.srt:force_style='FontName=SimHei,FontSize=24'"
output_hardsub.mp4
```

上述命令对 input.mp4 添加硬字幕，使用字幕滤镜加载 `movie.srt` 字幕文件，并指定了字体、字号等样式。**subtitles** 滤镜会自动解析 SRT/ASS 字幕并进行渲染[29]。输出的视频 output_hardsub.mp4 中字幕直接嵌在画面上，无法关掉。硬字幕适合在设备不支持软字幕时使用，但需要重新编码视频，过程可能较耗时。

以上两种方式各有优劣：软字幕不损伤画质且可开关，但兼容性取决于播放器；硬字幕则**万能通用**（任何播放器都能看到字幕）但需要占用视频码流。根据需求选择即可。

添加水印

给视频增加水印通常通过 **overlay** 滤镜实现。水印可以是静态图片（水印logo）或动态素材（例如GIF动画、视频片段）。下面分别介绍：

- **静态图片水印：**将一张PNG等格式的图片叠加到视频指定位置。命令格式如下[30]：

```
ffmpeg -i input.mp4 -i logo.png -filter_complex "overlay=x=10:y=10" output.mp4
```

这里第二个 `-i` 导入水印图，`-filter_complex` 使用 **overlay** 滤镜。参数 `x=10:y=10` 指定水印相对主视频左上角的偏移像素[31]（此例表示在距离左边和上边各 10px 位置放置水印）。未指定其他参数时，水印将持续覆盖整个视频时长。如需只出现于特定时间段，可以给 overlay 加 **enable** 表达式控制，例如 `overlay=enable='between(t,5,15)':x=10:y=10` 表示让水印在第5秒到15秒之间显示。

如果希望给水印图设置透明度，可以预先在PNG带透明通道，或使用 FFmpeg 的 `format=rgba, colorchannelmixer=...` 滤镜调整不透明度。针对文本水印，FFmpeg 也提供了 **drawtext** 滤镜，可以直接在视频上绘制文字[32]。

- **动态图像水印：**可以叠加动画 GIF 或视频片段作为水印。方法仍然是 overlay，但需要处理动画的循环和帧率。例如将一个 GIF 动画作为水印：

```
ffmpeg -i main.mp4 -ignore_loop 0 -i anim.gif -filter_complex "overlay=x=W-w-10:y=H-h-10:shortest=1" output.mp4
```

参数解析：`-ignore_loop 0` 使 GIF 无限循环播放[33]（否则默认只播放一次就停止，剩余时间静止[34]）；`overlay` 中用了 `x=W-w-10, y=H-h-10` 将水印放在画面右下角（W 和 H 代表主画面宽高，w 和 h 是水印宽高），`shortest=1` 则保证当主视频结束时停止输出（防止 GIF 无限循环导致输出无尽）[35][36]。这样，动画水印会在整个视频过程中不停播放循环。如果想让水印动起来（位置或旋转变化），可以利用 overlay 的时间变量 t 结合表达式，例如 `x='mod(t,10)*20'` 实现每秒水平移动等[37]。总之，借助 FFmpeg 强大的表达式功能，可以制作出各种动态水印效果[38][39]。

无论静态或动态水印，都可以通过调整叠加次序和滤镜参数实现**多重水印**。比如叠加多个图片只需多次 overlay：`-filter_complex "[0:v][1:v]overlay=... [tmp];[tmp][2:v]overlay=..."` 依次叠加即可[40]。添加文字水印则用 drawtext 滤镜多次叠加不同文本[41]。利用这些方法，可以方便地为视频添加版权标识、台标或者其他提示信息。

视频特效滤镜

FFmpeg 提供了丰富的滤镜来实现各种特效处理，比如模糊、锐化、翻转、调速等。以下举几种常见效果：

- **模糊与锐化：**使用 **boxblur** 或 **gaussian blur** 滤镜可模糊画面，使用 **unsharp** 滤镜可锐化细节。示例：将视频画面进行轻微模糊处理[42]：

```
ffmpeg -i input.mp4 -vf "boxblur=1.5:1" output_blur.mp4
```

上述命令应用 boxblur 滤镜，`luma_radius=1.5`，`luma_power=1`，对亮度通道进行一定程度模糊[43]。相反，要锐化画面，可以用：

```
ffmpeg -i input.mp4 -vf "unsharp" output_sharp.mp4
```

默认参数的 unsharp 滤镜会使用一个 5x5 矩阵对亮度进行锐化处理[44][45]。可以通过参数调整锐化强度或甚至设置负值来实现**高斯模糊**效果[46]。例如 `unsharp=6:6:-2` 相当于对亮度应用 radius=6 的模糊[46]。模糊滤镜常用于美化画面或打马赛克背景，锐化则用于提升细节轮廓。

- **镜像翻转：**使用 **hflip** 或 **vflip** 滤镜，可以水平或垂直翻转视频（镜像效果）。例如水平镜像一个视频：

```
ffmpeg -i input.mp4 -vf hflip output_flip.mp4
```

这会将视频左右调换，相当于沿垂直中线翻转 [47]。同理，`-vf vflip` 则上下翻转画面 [48]。翻转滤镜没有额外参数，应用后输出看起来就像原画在镜子中的映像。

- **快放与慢放（速度调整）：** 调整播放速度涉及同时处理视频和音频。视频速度由帧时间戳 (PTS) 控制，音频速度通过采样播放速率控制。FFmpeg 提供 **setpts** 滤镜修改视频帧时间，**atempo** 滤镜调整音频速度。基本思路：**setpts** 新PTS = 原PTS * 倍率的倒数，**atempo** 直接使用速度倍率（支持0.5～2.0之间，超过范围可串联滤镜叠加 [49] [50]）。例如，将视频加速为原来的2倍（快进） [51] [52]：

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

上述命令对视频流应用 `setpts=0.5*PTS`（使帧间隔减半），对音频流应用 `atempo=2.0`（播放速度加倍），从而整体提速2×输出 [51] [52]。相反，若要放慢一半速度，可以将 `setpts=2.0*PTS` 并使用 `atempo=0.5`（注意 atempo 最小支持0.5，即音频半速，再更慢需多次 atempo 级联 [53]）。例如慢放0.5倍：

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

调速后的输出会改变时长和节奏。如果只想改变视频不改变音频，可简单地对视频使用 setpts 而音频复制不变（或反之）；另外也可以用 `-r` 更改帧率实现倍速但那会丢帧，不太精确 [54]。setpts/atempo 滤镜的方法更为准确和灵活，是 FFmpeg 推荐的变速方案 [55] [56]。

FFmpeg 还有数百种滤镜，例如调色滤镜（hue、eq等用于调整亮度对比饱和度）、降噪滤镜（hqdn3d 等用于去除噪点）、淡入淡出滤镜（fade）、叠加画中画（overlay）等等，这里不一一展开。可以通过 `ffmpeg -filters` 查看支持的滤镜列表。合理组合滤镜，几乎可以实现**各种视频特效**处理。

音频波形与频谱可视化

FFmpeg 不仅能处理视频画面，也能将音频信号可视化为波形图或频谱图等，这对于制作音乐可视化视频或分析音频特征很有用。常用的滤镜有 **showwaves**（绘制波形）和 **showspectrum**（绘制频谱）等 [57]。下面介绍如何使用：

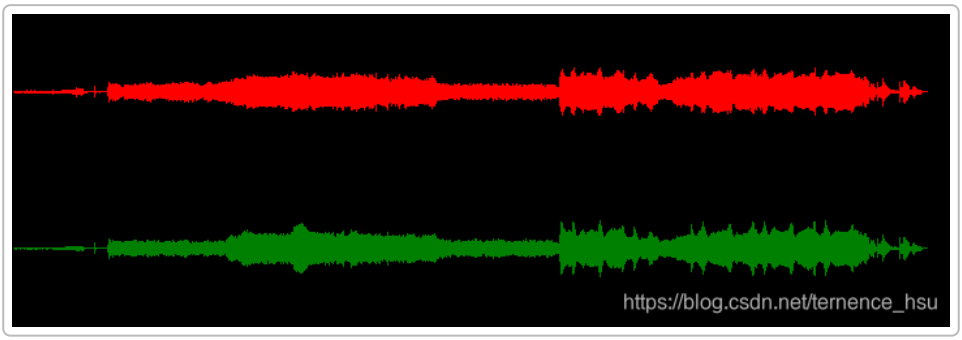
- **音频波形图（Audio Waveform）：** 使用 `showwaves` 或更高分辨率的 `showwavespic` 滤镜可以将音频振幅随时间变化绘制成曲线。例子：将一段音频生成波形视频 [58]：

```
ffmpeg -i song.mp3 -filter_complex "showwaves=s=1280x720:mode=line:rate=25,format=yuv420p" \
 -pix_fmt yuv420p output_waveform.mp4
```

该命令读取 song.mp3，用 **showwaves** 生成1280x720大小、25帧每秒的波形动画，mode=line 表示波形为线条而非山峰填充，最后指定输出为 yuv420p 像素格式以兼容大多数播放器 [58] [59]。输出的视频将显示随着音乐起伏的波形曲线。如果想要静态波形图片，可使用 **showwavespic** 滤镜并输出单帧即可 [60] [61]。例如：

```
ffmpeg -i song.mp3 -filter_complex "showwavespic=split_channels=1:s=720x240" -frames:v 1 waveform.png
```

上面命令生成了一张包含左右声道波形的 PNG 图片，分辨率 720x240 [62] [63]。下图展示了双声道音频的波形图示例，其中红色和绿色曲线分别代表左右声道的振幅：



使用 FFmpeg 绘制的双声道音频波形图，每个声道的音量随时间变化趋势清晰可见 [63] [64]。

- **音频频谱图（Audio Spectrum）：** 使用 **showspectrum** 或 **showcqt** 滤镜可以得到声音频谱的可视化。频谱图通常以时间为横轴、频率为纵轴、颜色亮度代表不同频率成分的强度。简单示例：将音频生成滚动频谱视频：

```
ffmpeg -i song.mp3 -filter_complex "showspectrum=s=1280x720:mode=combined:slide=scroll" \
 -pix_fmt yuv420p spectrum.mp4
```

这条命令使用 **showspectrum** 滤镜，将音频信号转换为 1280x720 的频谱图 [65]。`mode=combined` 表示将左右声道混合显示（可选 left/right 分开显示模式），`slide=scroll` 则生成滚动的实时频谱（像专业播放器的频谱动画）。输出 spectrum.mp4 即为音乐的实时频谱视频。如果要获得静态的频谱图片，可以使用 **showspectrumpic** 滤镜，或指定 `-frames:v 1` 截取某一帧频谱画面。

通过以上方法，可以制作**音乐可视化**视频：比如将歌曲的波形或频谱作为背景动态效果。同时这些可视化也可帮助分析音频的频率分布特点。在实际应用中，常将波形/频谱叠加到原视频或与专辑封面合成，实现内容和效果的融合。

抽帧、截图、提取音频

FFmpeg 可以方便地从视频中提取静态图像或者音频。以下是几个常见任务：

- **视频抽帧为序列图像：** 利用 FFmpeg 可以按指定帧率导出视频的逐帧图片。例如，将视频每帧都保存成 JPEG 图像：

```
ffmpeg -i input.mp4 -r 1 -f image2 frame_%03d.jpg
```

参数解释：`-r 1` 表示每秒取1帧，`-f image2` 指定输出格式为图像序列，`frame_%03d.jpg` 会按序号命名输出文件（例如 frame_001.jpg, frame_002.jpg 等）。如果想取更高频率的帧，可以把 `-r` 改大，比如 `-r 24`

则按视频帧率输出所有帧[66][67]。也可以在 `-i` 前使用 `-ss` 定位时间，截取从某时间开始的一系列帧。**注意：**抽帧导出时最好指定图片格式和命名模板，FFmpeg 将自动循环写出序列帧。

- **视频截图（特定时间帧）：**有时只需从视频中抓取特定时刻的一帧作为截图。可以利用 `-ss` 精确寻址然后输出单帧。例如，截取视频第 10 秒处的画面为图片：

```
ffmpeg -ss 10 -i input.mp4 -frames:v 1 snapshot.png
```

该命令在打开输入时直接跳到 00:00:10 处（加快定位），然后通过 `-frames:v 1` 只输出一帧视频到 snapshot.png。得到的即是第10秒时视频画面的静态截图[68]。可以更改 `-ss` 时间来抓取不同位置的帧；如需多张指定时间的截图，可以多次调用或使用复杂滤镜（如 select）。这里直接使用单次 ffmpeg 命令就能快捷截屏。

- **提取音频：**从视频中分离出音轨保存为独立音频文件也是常见需求。使用 `-vn`（no video）选项可以丢弃视频，只保留音频流。例如，将一个 MP4 视频的声音导出为 MP3：

```
ffmpeg -i input.mp4 -vn -c:a libmp3lame -q:a 2 output.mp3
```

这里 `-vn` 去掉视频，`-c:a libmp3lame` 指定使用 MP3 编码，`-q:a 2` 设置音质（2 大致对应 192kbps 高品质）。如果希望不重新编码而是无损提取，可以使用 `-c:a copy` 直接拷贝原音频流。例如原视频是 AAC 音频：`ffmpeg -i input.mp4 -vn -c:a copy output.aac` 将提取出原始 AAC 音轨而无需编码[69]。同理，可以提取视频的某一段音频，加上 `-ss` 和 `-t` 截取时间即可。

通过以上命令，我们可以方便地把视频**转换为静态图像**（生成预览图、Gif素材等），或**分离出音频**（获取背景音乐、对白音轨等）。这些操作不会影响原文件，只是复制或转换出新内容，在后续编辑中非常实用。

推流、录制、直播

除了本地文件处理，FFmpeg 也是强大的流媒体工具，可以采集音视频设备并推送直播流，以及在不同流协议之间进行转换。下面介绍在 Windows 下利用 FFmpeg 进行**摄像头/屏幕录制**、**推送 RTMP 流**以及**直播流转发**的基本示例。

摄像头/屏幕录制

录制摄像头：FFmpeg 可直接调用系统的摄像头和麦克风设备进行录音录像。在 Windows 上通常使用 DirectShow 接口（格式 `-f dshow`）。首先用命令列出可用设备名称：

```
ffmpeg -list_devices true -f dshow -i dummy
```

在输出中找到摄像头和音频输入的名称[70]（例如摄像头可能显示为“Integrated Camera”，麦克风可能是“Microphone (Realtek...)”）。获取名称后，即可执行录制。例如录制 5 分钟摄像头画面和麦克风声音，保存为文件：

```
ffmpeg -f dshow -i video="Integrated Camera" -f dshow -i audio="麦克风阵列 (Realtek(R) Audio)" \
 -t 00:05:00 -c:v libx264 -preset ultrafast -c:a aac output.flv
```

上述命令指定了视频输入为名为“Integrated Camera”的摄像头，音频输入为麦克风阵列[71]。`-t 00:05:00` 限制录制时长5分钟，视频编码用快速的 H.264 (`libx264` ultrafast)，音频编码为 AAC。输出封装为 FLV 文件（flv在直播和录制中很常用，封装开销小）[72]。录制过程中在控制台按 **Ctrl+C** 可提前终止。录制得到的 output.flv 可用播放器打开查看。若需要镜像修正摄像头画面，可以加上滤镜 `-vf hflip` 实时水平翻转[72]。

录制屏幕：FFmpeg 可以抓取屏幕内容进行录制或直播。在 Windows 下使用 GDIGrab (`-f gdigrab`) 来捕获屏幕。最简单的命令：

```
ffmpeg -f gdigrab -framerate 30 -i desktop screen.mp4
```

此命令以 30fps 帧率录制整个桌面屏幕，保存为 screen.mp4[73][74]。如果有多个显示器，`desktop` 默认抓取主屏；也可以通过 `-i desktop` 来指定全桌面。如要捕获某个窗口，可以使用 `-i title="窗口名称"` 指定窗口标题[74]。例如：

```
ffmpeg -f gdigrab -framerate 20 -i title="Untitled - Notepad" window.mp4
```

以上将20fps录制标题包含“Untitled - Notepad”的窗口画面（需窗口标题匹配）。屏幕录制默认不包含系统声音，如果需要录制系统或麦克风声音，可以再添加一个音频输入（如 `-f dshow -i audio="Stereo Mix (Realtek Audio)"`）并混流。

屏幕录制在制作教程、录制游戏等场景很实用。需要注意帧率和分辨率设置，帧率越高、分辨率越大，对机器性能和输出文件体积要求越高。可以根据需求调整 `-framerate` 以及输出编码设置（如降低 `-crf` 提高压缩）。

推送 RTMP 流

推流是指将音视频数据发送到流媒体服务器，实现直播分发。FFmpeg 常用作推流客户端，将本地文件或采集设备编码后推送到 RTMP 服务器地址。

典型命令格式如下（以 RTMP 为例）：

```
ffmpeg -re -i input.mp4 -c:v libx264 -c:a aac -f flv rtmp://<服务器地址>/<流路径>
```

参数说明：`-re` 让读取按实时速度进行（用于文件推流，防止一下子推完），`-c:v libx264 -c:a aac` 指定视频编码H.264、音频编码AAC（多数直播平台支持此组合），`-f flv` 指定输出封装为 FLV（RTMP 协议规定用 flv封装），后面的 RTMP URL 是由直播服务器提供的推流地址。例如，我们有一个本地视频文件，需要推送到 RTMP 测试服务器：

```
ffmpeg -re -i trailer.mp4 -c:v libx264 -preset veryfast -b:v 1200k -c:a aac -b:a 128k \
 -f flv rtmp://live.example.com/app/streamKey
```

这会将 trailer.mp4 以约 1.2Mbps 视频码率、128kbps 音频码率重新编码后推送到指定 RTMP 地址。许多直播平台要求推流采用特定编码和码率，请根据要求调整参数。若源已经是 H.264/AAC 等符合要求的格式，也可以直接 `-c copy` 复制流以降低CPU消耗[75]。

此外，可以推送**实时采集**的画面。例如前述摄像头采集直接推流，只需将输出改为 RTMP 地址：

```
ffmpeg -f dshow -i video="Integrated Camera" -vcodec libx264 -acodec copy -preset:v ultrafast \
 -tune:v zerolatency -f flv rtmp://<server>/live/webcam
```

上面命令将摄像头视频编码后推到给定 RTMP 流 [76] [77]（如无麦克风输入则用 `-acodec copy` 但实际上无音频流）。我们也可以把**屏幕**作为输入推送直播，例如：

```
ffmpeg -f gdigrab -i desktop -vcodec libx264 -preset ultrafast -tune zerolatency \
 -f flv rtmp://<server>/live/desktop
```

这将屏幕录制实时编码后推流 [78]。使用 `ultrafast` 预设和 `zerolatency` 调优可以降低编码延迟，适合直播场景 [79]。实际上，大部分直播软件的原理也是利用 FFmpeg 底层库进行采集和推流。

小提示：推流前确保网络带宽足够，并根据网络情况设置合适的码率，否则码率过高可能导致卡顿。使用 `-b:v` 和 `-b:a` 控制总码率，或使用恒定质量并辅以 `-maxrate` 限制峰值。FFmpeg 推流不会有 UI 界面，但可以从控制台日志观察推流是否正常（帧率、速度、丢帧情况等）。

简单直播转发

FFmpeg 可以充当“流媒体中继”，即拉取一个流再推送到另一个地址，实现协议转换或分发转发。例如将一个 RTSP 安防摄像头流转发为 RTMP 直播流：

```
ffmpeg -i rtsp://camera_ip/stream -rtsp_transport tcp -c:v libx264 -c:a aac -f flv rtmp://
live.example.com/app/streamKey
```

上述命令拉取 RTSP 源，用 TCP 方式防止丢包 [80]，转码为 H.264/AAC 并通过 RTMP 推出 [81]。这样就把 RTSP 转成了 RTMP，可方便地供网页播放器播放。若源已经是 H.264 视频和 AAC 音频，也可直接 `-c:v copy -c:a copy` 无需转码 [82] [83]。

类似地，FFmpeg 可以实现各种**转流转协议**：如接收 UDP 流输出为 HLS，或接收 RTMP 再推送另一路 RTMP 等等。 [84] 节选几个实用场景命令：

- **文件作为直播源：** `ffmpeg -re -i localvideo.mp4 -c:v copy -c:a copy -f flv rtmp://...` 将本地文件当实时流循环发送 [75]。
- **拉流保存：** `ffmpeg -i rtmp://server/live/stream -c:v copy -c:a copy output.mp4` 把直播流无损录制保存 [85]。
- **协议转换：** 上文的 RTSP→RTMP 是一例，亦可实现类似 **HTTP-FLV 转 HLS**、**RTP 转 RTMP** 等，只需调整输入输出和相应选项 [86]。

值得一提的是，FFmpeg 转发直播时也可同时**处理滤镜**或**转码压缩**，比如在中继时叠加水印或改变分辨率 [87]。这使得 FFmpeg 成为一个灵活的流媒体处理工具，在实际工程中可用它搭建自定义的转码转发服务。

通过本教程的分模块讲解，初学者应当对 FFmpeg 在 Windows 下的**基本用法**有了系统的了解。从安装配置、命令语法，到各类核心功能示例（格式转换、剪辑拼接、滤镜特效、音频处理、流媒体推流等），这些覆盖了日常音视频处理的主要场景。在使用过程中，建议多尝试这些命令并查看 FFmpeg 的控制台输出，从中学习更

多参数细节。同时可以参考官方文档和社区资源，进一步发掘 FFmpeg 更强大的功能。相信通过实践，您很快能熟练运用 FFmpeg 来应对各种音视频处理任务！ 88 89

---

1 2 3 4 5 6 7 8 11 16 19 69 Windows 安装 FFmpeg 新手教程（附环境变量配置）_ffmpeg环境变量配置-CSDN博客
https://blog.csdn.net/qq_34707272/article/details/148159336

9 10 12 13 win系统安装 FFmpeg 并设置环境变量_ffmpeg-20200311-36aaee2-win64-static-CSDN博客
https://blog.csdn.net/weixin_43876729/article/details/120218227

14 15 18 88 FFmpeg命令详解 - 瞎搞的富哥 - 博客园
https://www.cnblogs.com/liangjingfu/p/12857624.html

17 ffmpeg实现MP3转ACC格式功能原创 - CSDN博客
https://blog.csdn.net/tong5956/article/details/82852166

20 25 66 67 68 FFmpeg 常用命令 | 飞翔的荷兰人
https://ekozhan.com/summary/2022/05/25/ffmpeg.html

21 FFmpeg 使用指南之concat demuxer 串联多个文件 - Verne in GitHub
https://blog.einverne.info/post/2022/07/ffmpeg-concat-demuxer.html

22 23 FFmpeg 合并视频常用命令 - 冷雨Justin
https://qysit.com/ffmpeg-%E5%90%88%E5%B9%B6%E8%A7%86%E9%A2%91%E5%B8%B8%E7%94%A8%E5%91%BD%E4%BB%A4/

24 ffmpeg 合并多个视频文件 - 知乎专栏
https://zhuanlan.zhihu.com/p/677538831

26 29 使用FFMPEG加载外挂或内封字幕小记_ffmpeg内嵌字幕-CSDN博客
https://blog.csdn.net/yunxiaobaobei/article/details/130668234

27 28 使用 FFMPEG 命令将视频的〖外挂字幕〗转为 〖嵌入视频〗 - 刘江龙 - 博客园
https://www.cnblogs.com/qq-757617012/p/13967017.html

30 31 32 40 41 ffmpeg水印 静态_overlay水印-CSDN博客
https://blog.csdn.net/2301_78684949/article/details/131311367

33 34 37 ffmpeg第4篇：为视频添加动态水印 | 暖宝宝官方认证铲屎官
https://misland.github.io/2021/02/24/ffmpeg-watermark-2/

35 36 38 39 使用ffmpeg实现动态水印效果,-CSDN博客
https://blog.csdn.net/2301_78684949/article/details/131355706

42 43 44 45 46 《ffmpeg basics》中文版 -- 8.模糊、锐化和去噪_ffmpeg 锐化-CSDN博客
https://blog.csdn.net/qq_34305316/article/details/103935713

47 48 7.翻转和旋转视频 - 《FFmpeg Basics 中文文档》 - 技术池(jishuchi.com)
https://www.jishuchi.com/read/ffmpeg-basics/12478

49 FFmpeg命令行工具学习(五)：FFmpeg 调整音视频播放速度 - 知乎专栏
https://zhuanlan.zhihu.com/p/568798638

50 ffmpeg音视频倍速控制原创 - CSDN博客
https://blog.csdn.net/qq_22633333/article/details/104967304

51 52 54 55 56 使用ffmpeg 调整音视频方法速度_ffmpeg改变帧率后播放速度变慢-CSDN博客
https://blog.csdn.net/u011573853/article/details/103221783

53 FFmpeg命令行之ffmpeg调整音视频播放速度- 咸鱼Jay - 博客园
https://www.cnblogs.com/zuojie/p/16356827.html

57 89 FFmpeg Basic学习笔记（4） - Tocy - 博客园
https://www.cnblogs.com/tocy/p/ffmpeg-basic-learning-4.html

58 59 如何用ffmpeg根据音mp3音频生成频谱视频？ - Laziko - 博客园
https://www.cnblogs.com/Laziko/p/18240206

60 61 62 63 64 ffmpeg 绘制音频波形图_ffmpeg 生成音频波形-CSDN博客
https://blog.csdn.net/ternence_hsu/article/details/90575518

65 `showspectrum' — ffmpeg examples - Bitbucket
https://hhsprings.bitbucket.io/docs/programming/examples/ffmpeg/audio_visualization/_showspectrum_.html

70 71 72 使用ffmpeg调用本地摄像头并录音录像_ffmpeg调用摄像头-CSDN博客
https://blog.csdn.net/qq_37474779/article/details/137235700

73 74 FFmpeg | 录制摄像头和桌面数据windows | Zzz记忆
https://zfunnily.github.io/2020/09/ffmpegvideow/

75 76 77 78 79 80 81 82 83 84 85 86 87 FFMPEG采集摄像头推流方法说明 - China Soft - 博客园
https://www.cnblogs.com/chinasoft/p/16240495.html