目录

强大的音视频处理工具：FFmpeg

- 最新版本： `v1.0`
- 更新时间： `20210914`

简介

介绍音视频处理工具FFmpeg有哪些强大的功能。先对ffmpeg进行概览，包括可以用来干什么，与之相关的ffprobe、ffplay、ffserver等工具；再介绍如何安装ffmpeg；如何用ffmpeg处理音频，比如从音频中提取某段音频片段；以及各种视频处理，包括视频属性的获取和调整，包括调整视频宽高尺寸大小；以及动图gif处理，包括视频转动图、动图转视频；以及水印处理，包括去除视频水印；从视频中提取完整音频和音频片段；字幕相关处理，包括字幕的背景知识，包括软字幕和硬字幕、常见字幕格式ass和srt；以及如何用Aegisub编辑字幕；从视频中提取字幕、从srt转换出ass字幕；嵌入字幕，包括用流拷贝模式嵌入软字幕、用vf模式烧录嵌入硬字幕、且可以指定字幕位置、指定字幕文字属性等；整理ffmpeg使用的心得和常见问题；以及其他有哪些工具软件用到了ffmpeg、如何用Python调用ffmpeg；最后给出附录内容，包括help语法、文档资料等。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/media_process_ffmpeg: 强大的音视频处理工具：FFmpeg

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 强大的音视频处理工具：FFmpeg book.crifan.com
- 强大的音视频处理工具：FFmpeg crifan.github.io

离线下载阅读

- 强大的音视频处理工具：FFmpeg PDF
- 强大的音视频处理工具：FFmpeg ePub

- 强大的音视频处理工具：FFmpeg Mobi

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

FFmpeg概览

- `FFmpeg`
  - 是什么：一个，功能极其强大的，音视频处理相关的库

ffmpeg能用来干什么？

可以用 `ffmpeg` 做很多和音视频相关的处理。

绝大多数和音频视频字幕等相关的操作，ffmpeg都支持。

列举我之前遇到过的一些，供参考：

- 解析出视频的信息
  - 举例
    - 查看字幕属性信息
      - `ffmpeg -i xxx.mp4`
        - 输出字幕信息：
          - `Stream #0:2(zho): Subtitle: mov_text (tx3g / 0x67337874), 1280x108, 0 kb/s (default)`
    - 分辨率=宽x高
      - 用ffmpeg相关的ffprobe
        - `ffprobe -v error -select_streams v:0 -show_entries stream=height,width -of csv=s=x:p=0 input.mp4`
        - 输出：`1280x720`
- 视频去水印
  - `ffmpeg -i input.mp4 -vf "delogo=x=324:y=28:w=140:h=53" -c:a copy input_removedWatermarked.mp4`
- 从视频中提取音频并分割成多个音频片段
  - 前提：
    - 首先要有字幕文件：可以指定多个时间段
  - 对于每个时间段，用ffmpeg提取指定时间段的音频
    - `ffmpeg -i input.mp4 -ss 00:00:11.270 -to 00:00:14.550 -b:a 128k output_audio_000011270_000014550.mp3`
- 给视频加字幕=把字幕嵌入合并到视频中
  - srt字幕
    - 软字幕 内挂字幕
      - `ffmpeg -i video_no_subtitle.mp4 -i subtitle.srt -codec copy -map 0`

        - `video_with_soft_subtitle.mp4`
    - ass字幕
        - 硬字幕 内嵌字幕=嵌入ass字幕到视频中
            - `ffmpeg -i input_video.mp4 -vf ass=subtitle.ass input_video_with_ass_subtitle.mp4`
- 从视频中提取出字幕
    - `ffmpeg -i video_with_soft_subtitle.mp4 -map 0:s:0 extracted_subtitle.srt`
- 字幕类型转换
    - srt转换为ass
        - `ffmpeg -i subtitle.srt subtitle.ass`

另外还有：

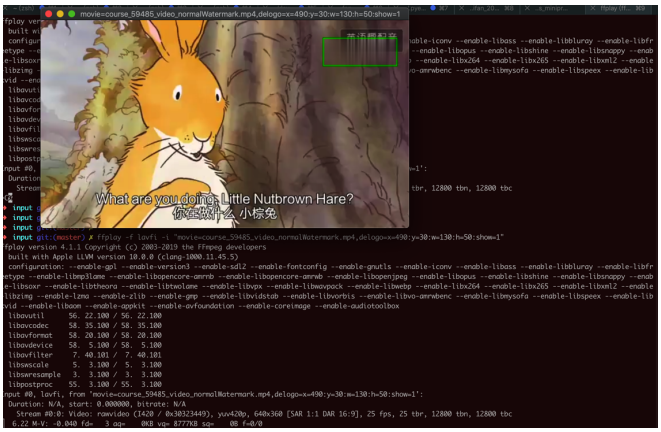
- `ffmpeg` 被其他工具调用：用于解析和操作音视频
    - Python的音频处理库：`pydub`
        - https://github.com/jiaaro/pydub
    - Python的音频解析库：`audioread`
        - https://github.com/beetbox/audioread

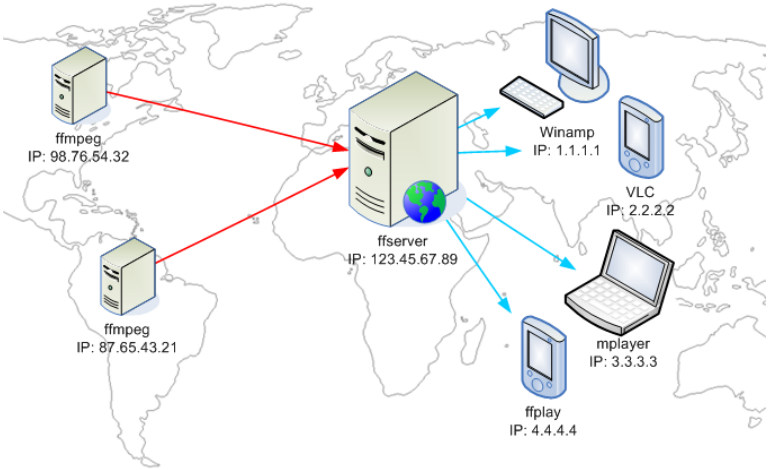
FFmpeg相关

ffmpeg与相关的ffprobe，ffplay，ffserver的区别

- ffmpeg：处理音视频
- ffprobe：检测音视频 -》 获取相关文件的属性和信息
- ffplay：播放音视频及其他一些辅助功能
  - 辅助功能举例：
    - 播放视频并显示水印位置：便于发现水印位置是否准确

```
ffplay -f lavfi -i "movie=course_59485_video_no
```

    - 播放效果：
    - 
    - -》方便看出要去除的水印的位置有偏差
      - 可以后续再调整参数值，让去除水印的区域更加准确
- ffserver：搭建流媒体服务器 -》用来支持其他端去播放音视频



FFmpeg安装

Mac

只安装ffmpeg

```
brew install ffmpeg
```

如果想要加上很多其他 `filter = 功能模块` ，则可以加上对应参数：

```
brew reinstall ffmpeg \
    --with-tools \
    --with-fdk-aac \
    --with-freetype \
    --with-fontconfig \
    --with-libass \
    --with-libvorbis \
    --with-libvpx \
    --with-opus \
    --with-x265
```

安装ffmpeg，和相关ffprobe，ffplay，甚至ffserver

去官网

Download FFmpeg

https://ffmpeg.org/download.html

提到的：

static FFmpeg binaries for macOS 64-bit

去下载最新版本：

- `ffmpeg` ：https://evermeet.cx/ffmpeg/ffmpeg-4.2.7z
- `ffprobe` ：https://evermeet.cx/ffmpeg/ffprobe-4.2.7z
- `ffplay` ：https://evermeet.cx/ffmpeg/ffplay-4.2.7z
- `ffserver` ：https://evermeet.cx/ffmpeg/ffserver-3.4.2.7z

分别下载后，解压得到二进制文件，再拷贝到合适目录。

比如Mac中的：

`/usr/local/bin`

即可。

问：如何确认已安装好?

用which去确认

```
which ffmpeg
which ffprobe
which ffplay
which ffserver
```

去确认能找到

以及顺带去看看版本

```
ffmpeg -version
ffprobe -version
ffplay -version
ffserver -version
```

音频处理

此处整理用ffmpeg处理音频文件。

提取音频片段

此处整理，从完整的音频文件中，提取其中一段，即提取音频片段。

从mp3中提取某个时间段的mp3

```
ffmpeg -i show_14322648_audio.mp3 -acodec copy -ss 00:00:0
```

- 参数解释
  - `-i`：input 输入文件
  - `-acodec copy`：
    - `-acodec` = `audio codec`：音频编码器
    - == `-c copy`
      - 等价于：

        ```
        ffmpeg -i show_14322648_audio.mp3 -c copy
        ```

官网文档

- Stream-copy ffmpeg Documentation

  > **3.2 Stream copy**
  >
  > Stream copy is a mode selected by supplying the copy parameter to the -codec option. It makes ffmpeg omit the decoding and encoding step for the specified stream, so it does only demuxing and muxing. It is useful for changing the container format or modifying container-level metadata

- toc-Main-options

> **-t duration (input/output)**
>
> When used as an input option (before -i), limit the duration of data read from the input file. When used as an output option (before an output url), stop writing the output after its duration reaches duration. duration must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual. -to and -t are mutually exclusive and -t has priority.
>
> **-to position (input/output)**
>
> Stop writing the output or reading the input at position. position must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.
>
> -to and -t are mutually exclusive and -t has priority.
>
> **-ss position (input/output)**
>
> When used as an input option (before -i), seeks in this input file to position. Note that in most formats it is not possible to seek exactly, so ffmpeg will seek to the closest seek point before position. When transcoding and -accurate_seek is enabled (the default), this extra segment between the seek point and position will be decoded and discarded. When doing stream copy or when -noaccurate_seek is used, it will be preserved.
>
> When used as an output option (before an output url), decodes but discards input until the timestamps reach position.
>
> position must be a time duration specification, see (ffmpeg-utils)the Time duration section in the ffmpeg-utils(1) manual.

输出举例

```
➜  splitAudio git:(master) ffmpeg -i show_14322648_audio.mp
ffmpeg version 4.0.2 Copyright (c) 2000-2018 the FFmpeg dev
  built with Apple LLVM version 10.0.0 (clang-1000.11.45.2)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/4.0.2 --
  libavutil      56. 14.100 / 56. 14.100
  libavcodec     58. 18.100 / 58. 18.100
  libavformat    58. 12.100 / 58. 12.100
  libavdevice    58.  3.100 / 58.  3.100
  libavfilter     7. 16.100 /  7. 16.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  1.100 /  5.  1.100
  libswresample   3.  1.100 /  3.  1.100
  libpostproc    55.  1.100 / 55.  1.100
Input #0, mp3, from 'show_14322648_audio.mp3':
  Metadata:
    major_brand     : isom
    minor_version   : 0
    compatible_brands: isomiso2avc1
    encoder         : Lavf58.12.100
  Duration: 00:00:48.27, start: 0.025057, bitrate: 128 kb/s
    Stream #0:0: Audio: mp3, 44100 Hz, stereo, fltp, 128 kb
    Metadata:
      encoder         : Lavc58.18
Output #0, mp3, to 'show_14322648_audio_000003110_00000611(
  Metadata:
    major_brand     : isom
    minor_version   : 0
    compatible_brands: isomiso2avc1
    TSSE            : Lavf58.12.100
    Stream #0:0: Audio: mp3, 44100 Hz, stereo, fltp, 128 kb
    Metadata:
      encoder         : Lavc58.18
Stream mapping:
  Stream #0:0 -> #0:0 (copy)
Press [q] to stop, [?] for help
size=      47kB time=00:00:02.97 bitrate= 129.5kbits/s spee
video:0kB audio:47kB subtitle:0kB other streams:0kB global
```

视频处理

此处整理用ffmpeg处理视频文件。

视频属性

获取视频属性

用ffprobe获取视频分辨率 宽度和高度

```
ffprobe -v error -select_streams v:0 -show_entries stream=v
```

举例：

```
# ffprobe -v error -select_streams v:0 -show_entries strear
480x360
```

查看视频基本信息

通过：

```
ffmpeg -i xxx.mp4
```

即可看到mp4视频的信息，其中包括了字幕的信息：

```
    Stream #0:2(zho): Subtitle: mov_text (tx3g / 0x6733787
    Metadata:
      creation_time   : 2018-10-13T03:27:31.000000Z
      handler_name    : SubtitleHandler
```

其中：

- `Stream #0:2`：字幕也是一个stream流，index是0:2
- `zho`：(应该)表示是中文
- `mov_text`：后来才听说，好像指的是`Apple Mov Text`格式的?

举例1

```
➜  ffmpeg_edit_subtitle ffmpeg -i CTT_Folge_01_CH_Subs_Defa
ffmpeg version 3.4.2 Copyright (c) 2000-2018 the FFmpeg dev
  built with Apple LLVM version 9.0.0 (clang-900.0.39.2)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/3.4.2 --
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'CTT_Folge_01_CH_Su
  Metadata:
    major_brand     : mp42
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    creation_time   : 2018-10-13T03:27:31.000000Z
    encoder         : HandBrake 1.1.2 2018090500
  Duration: 00:26:52.03, start: -0.001333, bitrate: 1762 kb
    Stream #0:0(und): Video: h264 (Main) (avc1 / 0x3163766
    Metadata:
      creation_time   : 2018-10-13T03:27:31.000000Z
      handler_name    : VideoHandler
    Stream #0:1(eng): Audio: aac (LC) (mp4a / 0x6134706D),
    Metadata:
      creation_time   : 2018-10-13T03:27:31.000000Z
      handler_name    : Stereo
    Stream #0:2(zho): Subtitle: mov_text (tx3g / 0x6733787
    Metadata:
      creation_time   : 2018-10-13T03:27:31.000000Z
      handler_name    : SubtitleHandler
At least one output file must be specified
```

看到字幕了：`Stream #0:2(zho)`

zho应该是中文

且显示已经是default了，即默认是中文字幕（而不是之前的英文字幕）

举例2

```
ffmpeg -i 1xx-我来保证你们的安全.mp4
ffmpeg version 3.4.2 Copyright (c) 2000-2018 the FFmpeg dev
  built with Apple LLVM version 9.0.0 (clang-900.0.39.2)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/3.4.2 --
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from '1xxx-我来保证你们的安
  Metadata:
    major_brand     : isom
    minor_version   : 0
    compatible_brands: isomiso2avc1
    creation_time   : 2018-01-28T05:46:37.000000Z
  Duration: 00:01:26.24, start: 0.000000, bitrate: 606 kb/s
    Stream #0:0(eng): Video: h264 (High) (avc1 / 0x31636661
    Stream #0:1(eng): Audio: aac (LC) (mp4a / 0x6134706D),
    Metadata:
      creation_time   : 2018-01-28T05:46:35.000000Z
At least one output file must be specified
```

中并没有 `Subtitle` 的信息

-> 看来这个视频本身就不包含独立的字幕

-> 没法用ffmpeg去提取字幕了

视频属性调整

播放速率

加倍速播放视频

```
ffmpeg -i input.mov -filter:v "setpts=0.5*PTS" output.mov
```

慢倍速播放视频

```
ffmpeg -i input.mov -filter:v "setpts=2.0*PTS" output.mov
```

定义帧率 16fps

```
ffmpeg -i input.mov -r 16 -filter:v "setpts=0.125*PTS" -an
```

静音视频（移除视频中的音频）

```
ffmpeg -i input.mov -an mute-output.mov
```

- 说明
  - `-an` 就是禁止音频输出

尺寸调整

缩放视频尺寸

```
ffmpeg -i big.mov -vf scale=360:-1  small.mov
```

- 注意
  - `scale` 值必须是偶数，这里的 `-1` 表示保持长宽比，根据宽度值自适应高度
  - 如果要求压缩出来的视频尺寸长宽都保持为偶数，可以使用 `-2`

extend扩大视频高度（和宽度）

- 注意：此处改变视频宽高，是 `调整` = `extend` = `enlarge` = `扩大` 或 `缩小` ，不是 `等比例缩放` = `resize`

思路：用 `ffmpeg` 的 `pad` 参数去指定要扩大的宽度和高度

参数详解：

- `pad` = `padding` ：增加视频区域，即宽度和（或）高度
  - `width` 和 `height` ：（增加了padding后的）输出的视频的宽度和高度
    - 值的方式
      - 固定的数值
      - 表达式
        - 可以借用内置支持的常量或变量
          - 比如
            - 输入的原始视频的
              - 宽度： `in_w` = `iw`
              - 高度： `in_h` = `ih`
            - 输出的padding后的视频的
              - 宽度： `out_w` = `ow`
              - 高度： `out_h` = `oh`
  - x和y：
    - 原始视频，放在输出后的视频的位置
    - 默认： `0` 和 `0`
      - 特殊
        - 如果（ `x` 或 `y` ）是负数，则效果是（水平方面或垂直方向）自动居中

用法举例

- 输入：原始视频 480x360
- 输出

希望：宽度不变，高度在下面增加100，背景色是白色

命令：

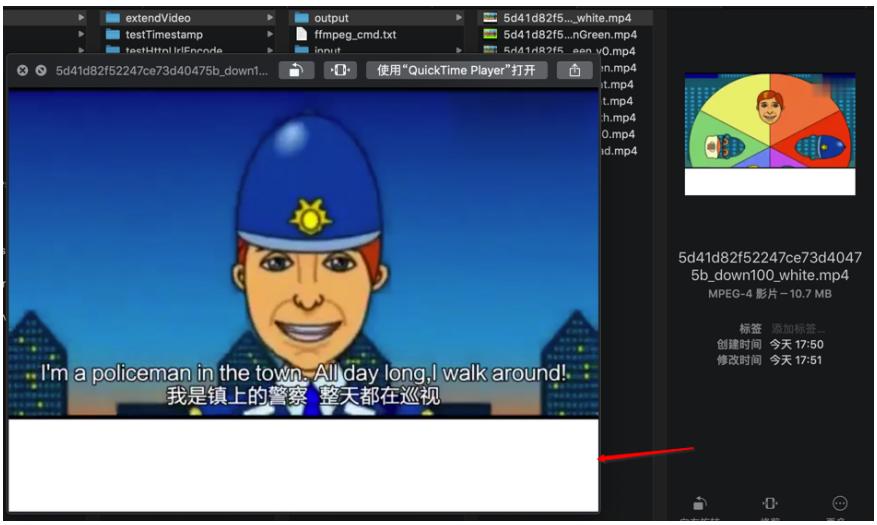
```
ffmpeg -i input.mp4 -vf "pad=width=0:height=460:x=0:y=0:co
```

参数说明：

- 高度
  - 可以用数值：`460`
  - 也可以用表达式：`ih+100`

    ```
    ffmpeg -i input.mp4 -vf "pad=width=0:height=ih+100:
    ```

效果：



希望：调整多个属性

希望：

- 高度：上面增加 `50` ，下面增加 `100`
  - 总增加高度= `50+100` = `150`
- 宽度：左右都增加 `30` ，原视频居中
  - 总增加宽度= `30*2` = `60`
- 背景色：`LawnGreen`
- 透明度：`0.2`

命令：

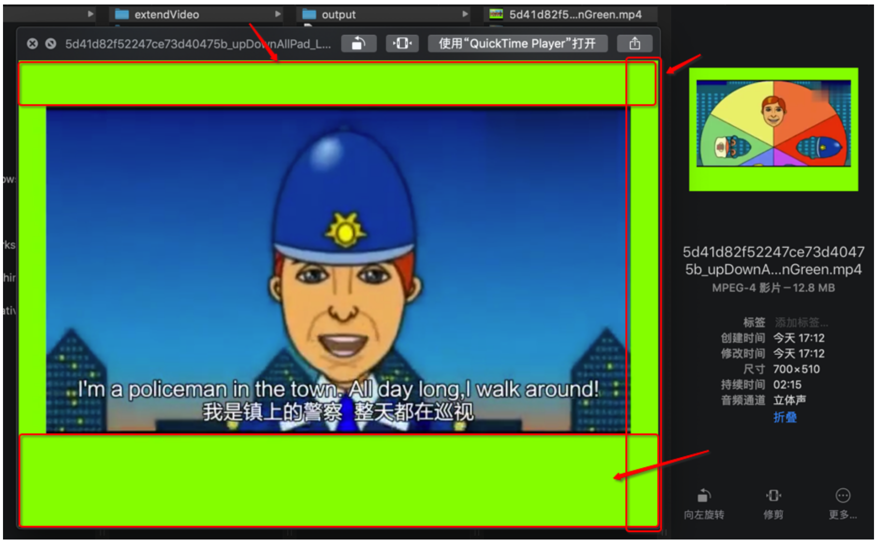
- pad中的值用手动计算出的值：

```
ffmpeg -i input.mp4 -vf "pad=width=540:height=510:x=3
```

- 或：pad中的值尽量用表达式：

```
ffmpeg -i input.mp4 -vf "pad=width=iw+60:height=ih+15
```

效果：



附录：

前面折腾期间的命令记录，包括出错的命令：

```
ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl

ffmpeg -i input/5d41d82f52247ce73d40475b.mp4 -vf "pad=widtl
```

供参考。

动图gif

视频转动图gif

视频转成动图(gif)

```
ffmpeg -i small.mp4 small.gif
```

转化视频中的一部分为 GIF

从视频中第二秒开始，截取时长为3秒的片段转化为 gif

```
ffmpeg -t 3 -ss 00:00:02 -i small.webm small-clip.gif
```

转化高质量 GIF

默认转化是中等质量模式，若要转化出高质量的 gif，可以修改比特率

```
ffmpeg -i small.mp4 -b 2048k small.gif
```

动图转视频

gif转mp4

```
ffmpeg -f gif -i animation.gif animation.mp4
```

特殊：gif转出来的mp4播放不了？

有些gif转化出来的mp4不能被Mac的 `QuickTime Player.app` 播放，需要添加 `pixel formal` 参数

```
ffmpeg -i input.gif -vf scale=420:-2,format=yuv420p out.mp4
```

- 说明
  - 使用 yunv420p 需要保证长宽为偶数，这里同时使用了 `scale=420:-2`
    - wiki中解释： QuickTime Player 对 H.264 视频只支持 YUV 色域 4:2:0 方式的二次插值算法

gif转其他视频格式

```
ffmpeg -f gif -i animation.gif animation.mpeg
ffmpeg -f gif -i animation.gif animation.webm
```

水印

去除视频水印

用ffmpeg去除视频水印

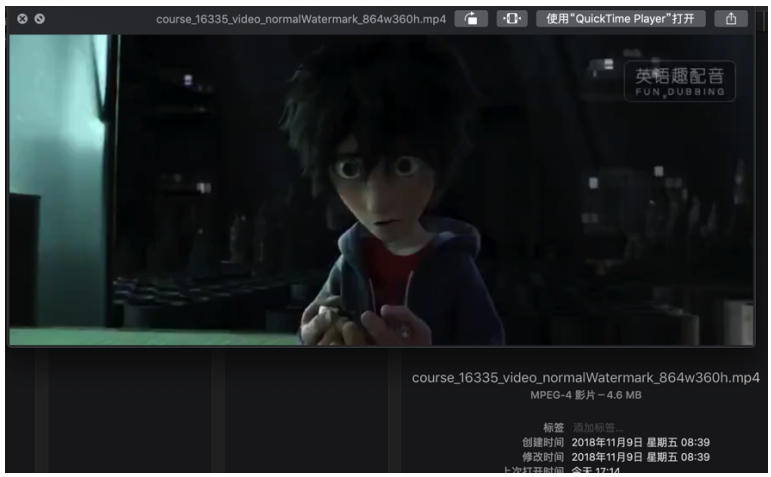
举例：

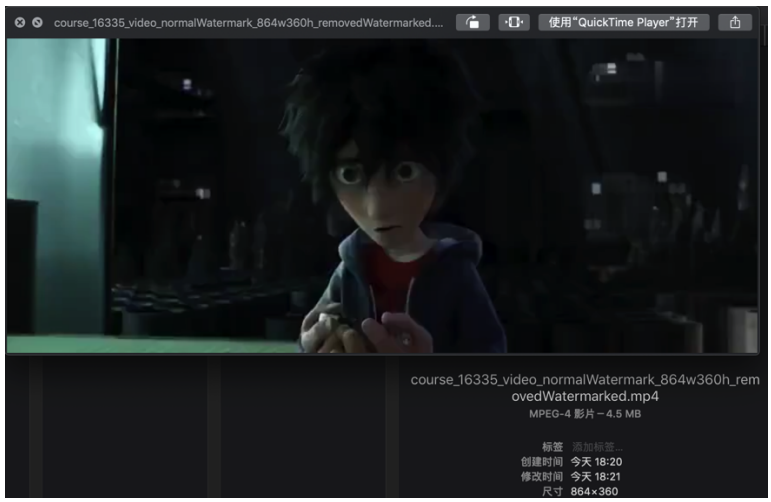
```
ffmpegCmd=ffmpeg -i course_16335_video_normalWatermark_864w
```

参数解释：

- `x` 和 `y` 是以**左上角**为原点的坐标
- `w` 和 `h` 是水印的矩形区域的宽度和高度

效果：

- 去水印前
  - 

- 去水印后
  - 

调试：如何确定水印的区域

另外，在去除水印之前，可以借助于ffplay去播放，给水印区域加上绿色框：

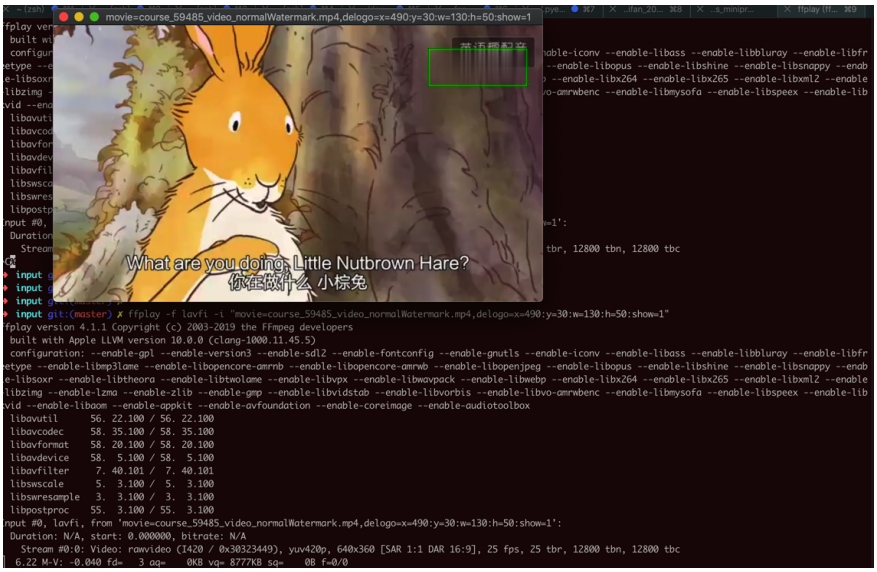
```
ffplay -f lavfi -i "movie=video_file.mp4,delogo=x=11:y=22:
```

便于识别水印位置对不对。

举例：

```
ffplay -f lavfi -i "movie=course_59485_video_normalWaterma
```

可看到右上角的区域，加上了绿色边框：



可以看出水印矩形区域的位置明显不对，和希望的位置有偏差，然后继续去调整 `x` 、 `y` 、 `w` 、 `h` 参数即可。

从视频中提取音频

此处整理，从视频文件中提取音频相关内容。

从mp4中提取音频mp3

提取整个音频

```
ffmpeg -i show_14322648_video.mp4 -b:a 128k show_14322648_a
```

提取音频片段

```
ffmpeg -i show_157712932_video.mp4 -ss 00:00:11.270 -to 00:
```

- 参数解释

```
usage: ffmpeg [options] [[infile options] -i infile]...

-ss time_off        set the start time offset
-to time_stop       record or transcode stop time

-ab bitrate         audio bitrate (please use -b:a)
```

字幕处理

此处介绍，视频中的字幕相关处理，比如合并、嵌入、烧录，转换等等。

字幕背景知识

字幕分类

根据字幕信息嵌入到视频中的方式，可以把字幕分为：

- 软字幕
  - 特点
    - 过程可逆
      - 可以提取出字幕文件，查看字幕源代码
  - 包含
    - 内挂字幕
      - 一般指字幕文件与视频一同封装在MKV文件中，播放时需经过播放器处理解析显示（=VSFilter渲染）
    - 外挂字幕
      - 以单独的字幕文件形式存在，播放时经播放器处理解析显示（=VSFilter渲染）到视频上
- 硬字幕=内嵌字幕
  - 指字幕被以图形方式硬编码到视频中
    - 变成视频数据本身=视频数据的一部分
  - 特点
    - 过程不可逆
      - 无法再把字幕提取出来
    - 播放时不需要额外的播放器读取解析显示（=VSFilter渲染）字幕

字幕格式

ASS

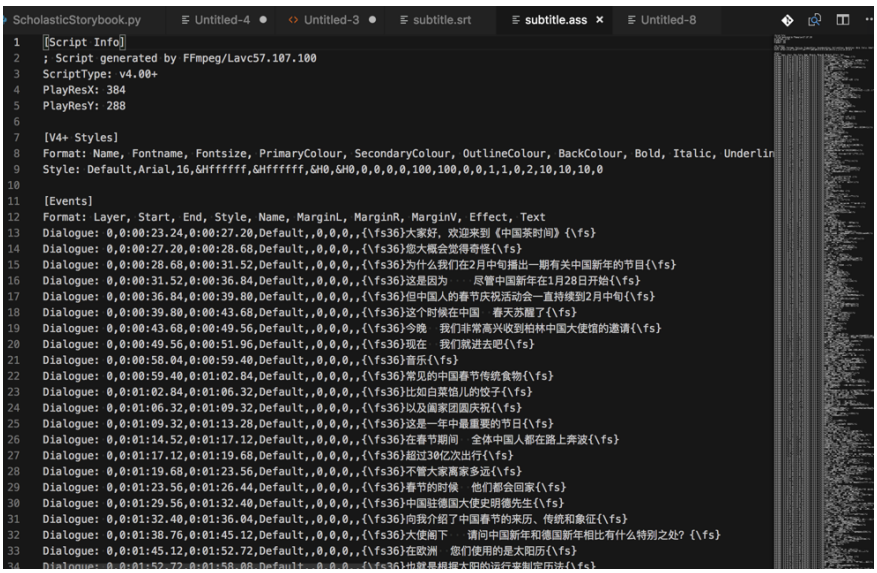
- Advanced SubStation Alpha=ASS
  - 一句话描述：一种比SSA更先进的字幕脚本格式
    - 之前的格式 SSA
      - 全称：SubStation Alpha
        - 或 Sub Station Alpha
      - 由 CS Low（亦作 Kotus）创建
      - 特点
        - 比传统字幕（如SRT）更加强大先进的字幕文件格式。
        - 该格式在Windows平台上可以经播放器由VSFilter渲染进行播放。这是一款广受欢迎且已停止开发的字幕编辑制作工具。
      - 最新版本：V4.00
  - 基于：SSA 4.00+编码构建
  - 特点：
    - 在SSA编写风格的基础上增添更多的效果和指令
  - 使用现状
    - 该字幕格式常被字幕组所应用
      - 目前有很多播放器支持渲染SSA、ASS字幕
  - 最新版本：V4.00+
  - ASS字幕分类
    - 软字幕
      - 特点
        - 过程可逆
          - 可以提取、取出和查看源代码的
      - 包含
        - 内挂字幕
          - 一般指字幕文件与视频一同封装在MKV文件中，播放时需经过VSFilter渲染
        - 外挂字幕
          - 字幕文件以单独形式存在，播放时经VSFilter渲染到视频上
    - 内嵌字幕
      - 指字幕被以图形方式硬编码到视频中
        - 变成视频数据本身=视频数据的一部分
      - 特点
        - 过程不可逆
          - 无法再把字幕提取出来

        - 播放时不需VSFilter等渲染
- 常用字幕制作软件
    - Aegisub
    - Jubler
    - VisualSubSync
    - subtitleeditor
    - Sabbu
    - Sub Station Alpha tool
    - Subtitle Workshop
    - Subtitle Processor
    - Miyu
    - Gaupol
- ASS格式详解
    - 脚本说明
        - Script Info: 脚本的一般全局信息:
            - Title：标题
            - Original Script：脚本原作
            - Script Updated By：脚本优化
            - Script Type：类型
                - 用于兼容性设置
                    - SSA=4.00
                    - ASS=4.00+
            - PlayResX & PlayResY：屏幕宽高
            - PlayDepth：决定颜色数量
            - Timer：定时器
        - V4 Styles: 定义文字样式，在events部分可以直接调用这些样式。
            - PrimaryColour：第一颜色，即文字本身的颜色
            - SecondaryColour：第二颜色
                - 在卡拉OK时使用，卡拉OK指令会用第一颜色填充第二颜色。
            - TertiaryColour：边框色，文字边框的颜色
                - ASS中称为OutlineColour
            - BackColour：阴影颜色，文字阴影的颜色
            - MarginL, MarginR. 左右缩进。文字距离视频最左最右的最小距离。
            - MarginV： 上下缩进。
                - 文字热点（对齐点）距离视频上下的最小间距离。这取决于文字对齐方式，如果该文字使用中对齐，则上下缩进值无效。需要时文字可用\n或\N命令换行。ASS文件中，Script Info部分的一个参数“WrapStyle”也可以改变此值
            - Outline：边框样式

        - 文字可以设定为有边框，无边框，或拥有不透明边框（矩形背景）
      - Shadow：阴影距离
        - 文字的阴影到文字的距离大小
      - AlphaLevel.：透明度（0至255有效）
      - Encoding：ASCII码中的文字编码方式
      - Alignment：对齐方式
        - 含义：画面中位置的对齐方式，按照数字键盘区布局
        - 取值范围：1-9
          - 具体效果，参考此图： *
        - 默认：2
          - 底部居中
  - Events: 字幕的主体部分
    - 字幕的出现时间和样式、对样式的修改和特效
      - 语法格式为：
        - {\命令(参数，多个参数用逗号隔开)}
          - 例如
            - {\move(80,80,200,200,150,300)}
            - 只有一个参数的命令不需要括号 (如. {\kf89})
    - 注意
      - 一对大括号内可以放置多个语句
        - 如
          - {\bord8\be1}
      - \N 和 \n 命令不需要在两边加大括号就可使用

ass字幕文件举例

举例1：



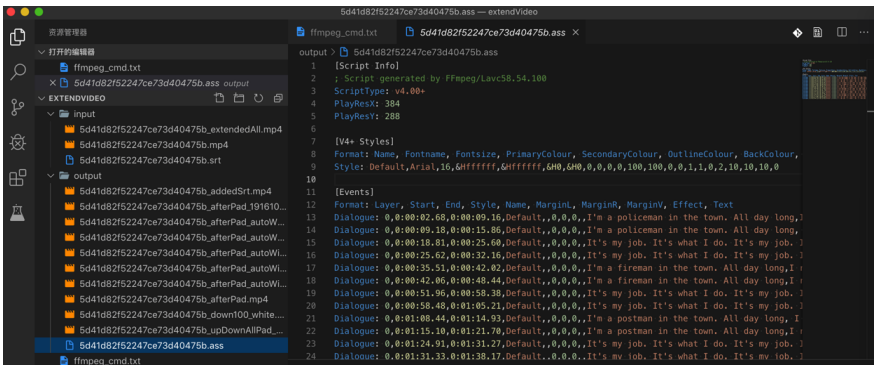
举例2：`input/5d41d82f52247ce73d40475b.ass`

内容是

```
[Script Info]
; Script generated by FFmpeg/Lavc58.54.100
ScriptType: v4.00+
PlayResX: 384
PlayResY: 288

[V4+ Styles]
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryC
Style: Default,Arial,16,&Hffffff,&Hffffff,&H0,&H0,0,0,0,0,1

[Events]
Format: Layer, Start, End, Style, Name, MarginL, MarginR, M
Dialogue: 0,0:00:02.68,0:00:09.16,Default,,0,0,0,,I'm a po
Dialogue: 0,0:00:09.18,0:00:15.86,Default,,0,0,0,,I'm a po
...
Dialogue: 0,0:02:04.21,0:02:10.82,Default,,0,0,0,,It's my
```



srt

- `srt` = `subtitle`

举例

中英文双字幕：

```
1
00:00:02,000 --> 00:00:06,700
Careful now, I don't want to hurt you.
现在要小心了 我可不想伤到你啊

2
00:00:10,500 --> 00:00:14,550
So Mr. Teacher guy, as the real Dragon Warrior,
那么 这个作为神龙斗士老师的你

3
00:00:14,560 --> 00:00:17,950
I say to you, Shakabooey!
我想对你说 滚你的

4
00:00:24,500 --> 00:00:28,030
So, guess you can start planning my parade now.
那 我想你们可以开始我的游行了是吧
```

英文（单字幕）：

```
1
00:00:02,310 --> 00:00:04,677
I am a little turtle

2
00:00:04,752 --> 00:00:07,540
I crawl so slow

3
00:00:07,670 --> 00:00:12,120
I carry my house wherever I go.

4
00:00:12,210 --> 00:00:16,927
When I get tired, I put in my head,
```

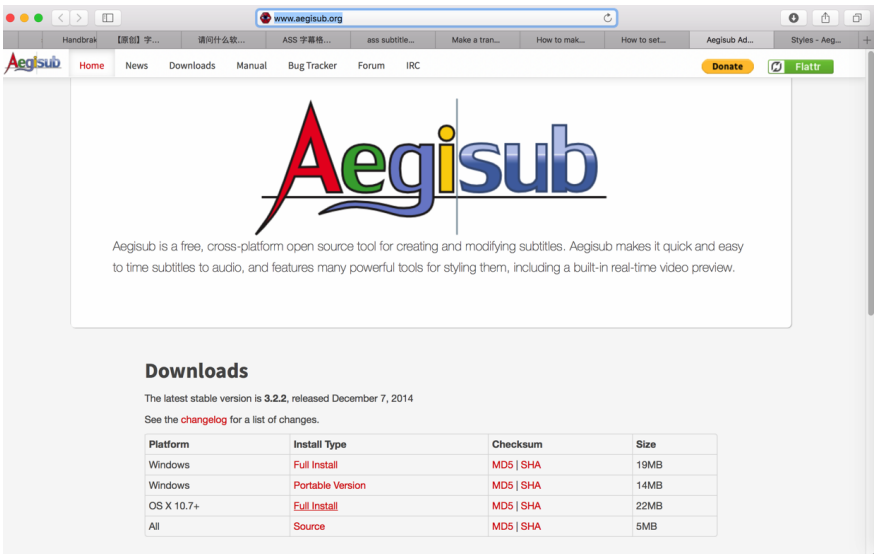
编辑字幕

Aegisub

- `Aegisub`
  - 概述：字幕编辑工具
  - 主页
    - Aegisub Advanced Subtitle Editor
      - http://www.aegisub.org

用Aegisub编辑字幕

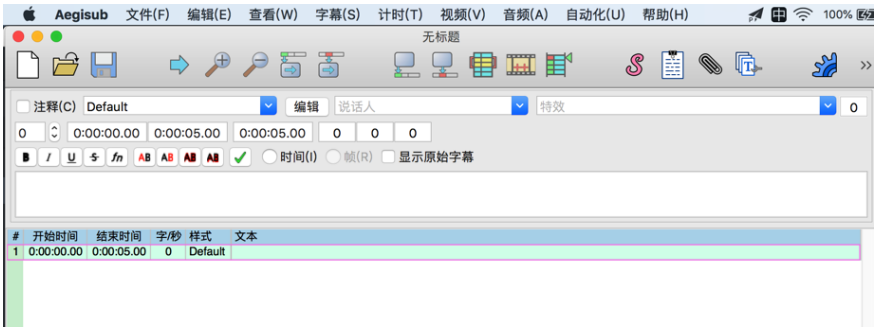
从官网



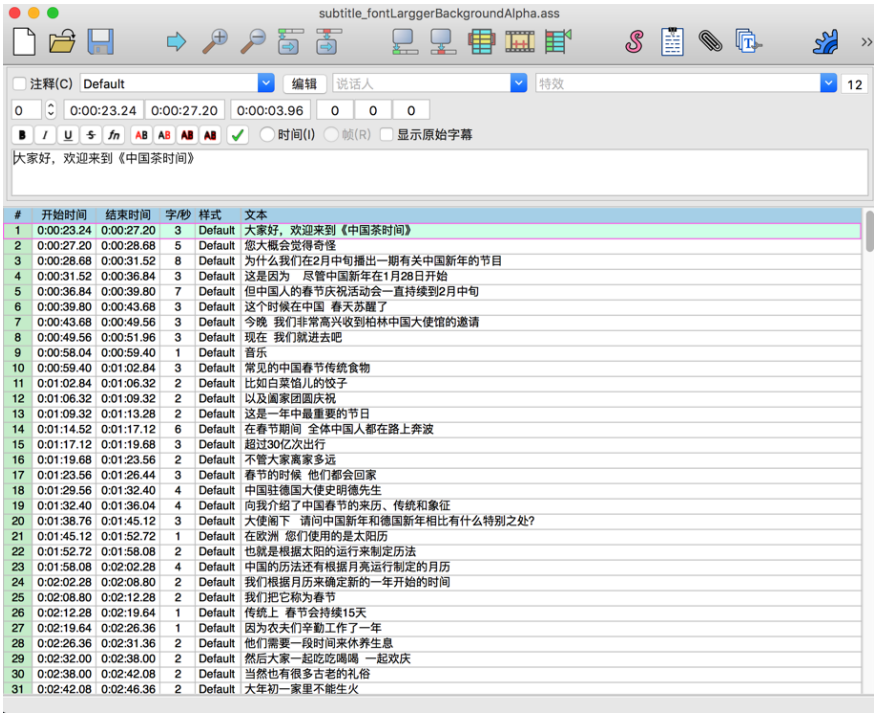
下载对应版本：

`Aegisub-3.2.2.dmg`

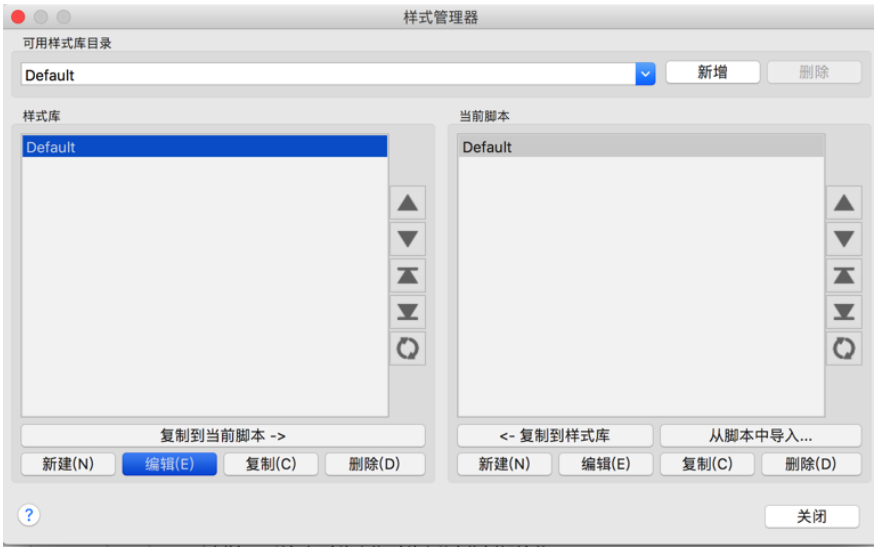
去安装，然后打开



先去打开你的字幕文件，此处的ass文件：



字幕-》样式管理器-》去编辑 当前脚本 中的，比如默认有个：Default 的样式

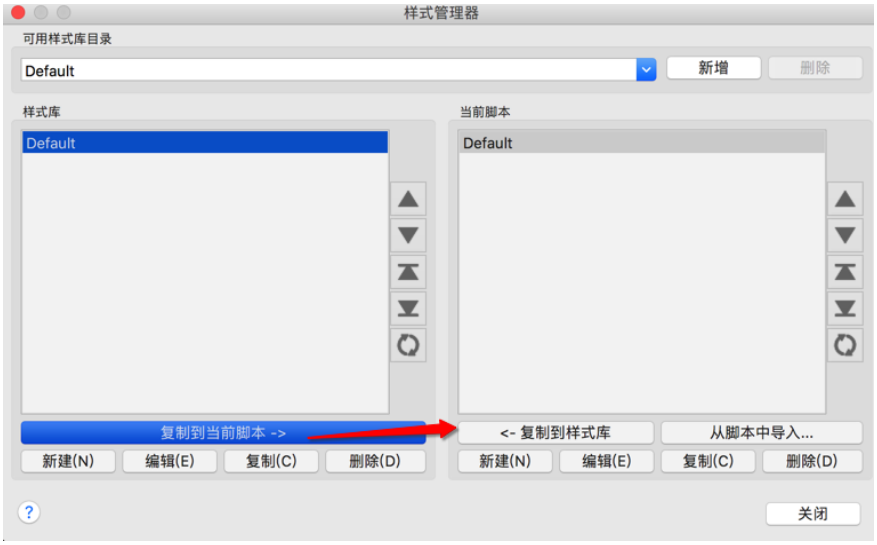


就出来字幕预览了：

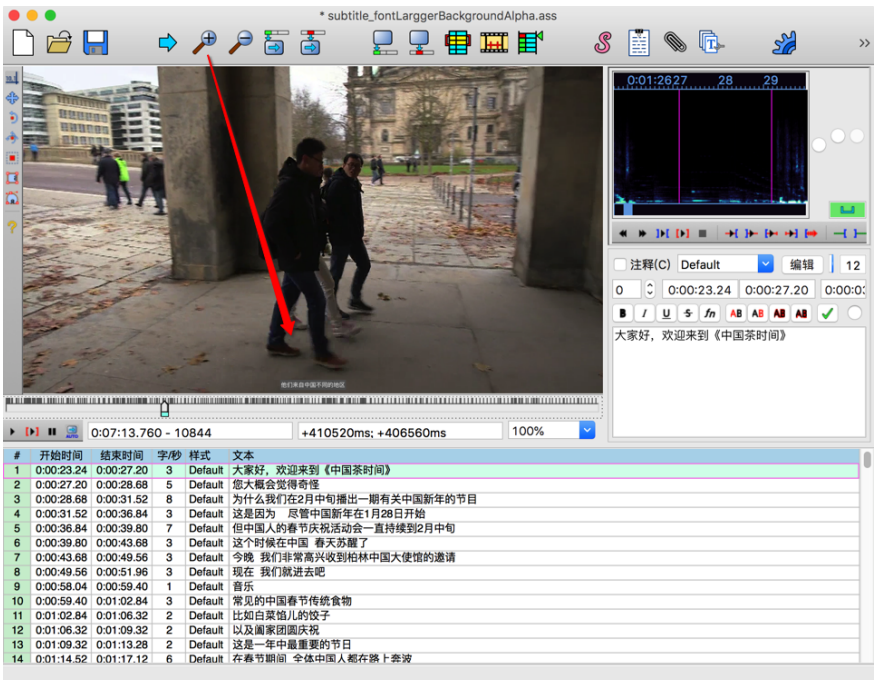
参数解释：

- 半透明效果
  - 先勾选 边框-》不透明效果
  - 再去：颜色-》 点击：边框 或 阴影，弹出设置框，改动你要的颜色，尤其是调整 透明度

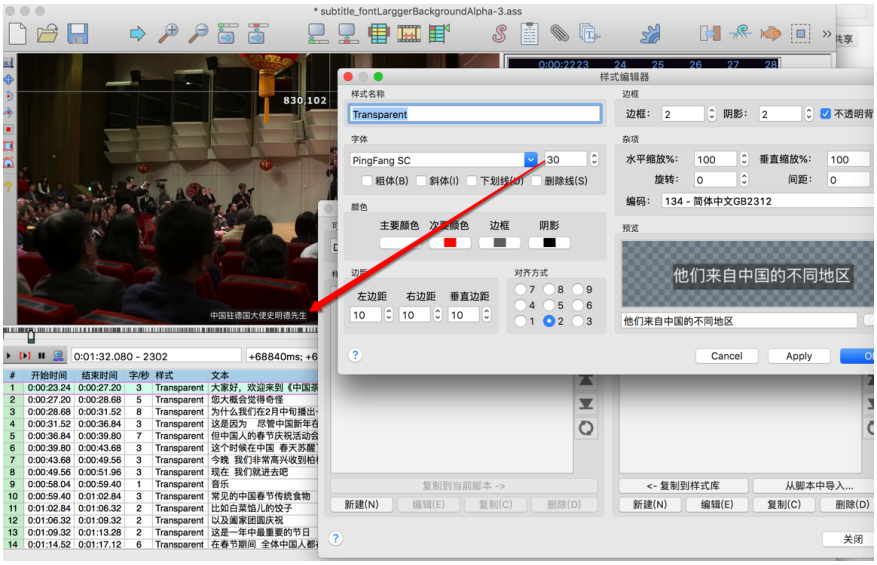
从左边的 样式库 中 默认的样式：Default，选中，点击下面的 复制到当前脚本



然后再去放大，即可以看到效果了：



继续编辑字幕



直到调节出你要的效果。

编辑好ass后，另存为，得到最终的ass文件。

具体过程详见：

【已解决】用Aegisub字幕编辑器去调整字体大小和字幕背景半透明效果

举例

- 输入=编辑前

此处之前从某mp4视频中用ffmpeg提取出srt，在用ffmpeg从srt转换出ass字幕

```
[Script Info]
; Script generated by FFmpeg/Lavc58.18.100
ScriptType: v4.00+
PlayResX: 384
PlayResY: 288

[V4+ Styles]
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryC
Style: Default,Arial,36,&Hffffff,&Hffffff,&H0,&H0,0,0,0,0,1
```

- 编辑字幕

用Aegisub编辑字幕，，调整出我要的效果后，另存为保存出的ass文件

把其配置：

```
Style: Transparent,PingFang SC,20,&H00FFFFFF,&H000000FF,&HE
```

换进来即可

- 输出=编辑后

```
[Script Info]
; Script generated by FFmpeg/Lavc58.18.100
ScriptType: v4.00+
PlayResX: 384
PlayResY: 288

[V4+ Styles]
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryC
Style: Transparent,PingFang SC,20,&H00FFFFFF,&H000000FF,&HE

[Events]
Format: Layer, Start, End, Style, Name, MarginL, MarginR, M
Dialogue: 0,0:00:23.24,0:00:27.20,Transparent,,0,0,0,,大家好
```

- 核心配置

就一句：

```
Style: Transparent,PingFang SC,20,&H00FFFFFF,&H000000FF,&HE
```

起到了配置字幕属性，实现了效果：

- 字体：PingFang SC
- 字体大小：20
- 字幕的背景半透明效果：后面很多参数组合的效果

常见问题

当导入视频时会提示：载入视频的分辨率与当前脚本指定的分辨率不匹配

- 不要用默认选择：直接设为视频分辨率
  - 会导致预览看到的视频很小，字幕很小
    - 后续调整字幕的大小后，预览中视频里的字幕仍旧很小，导致字幕字体大小无法生效
- 重新选这个：重设脚本分辨率
  - 但是也会导致另存为的ass文件中，有视频方面的设置
    - 我是后来自己参考原始的配置，改回去为原始的配置的

提取字幕

从视频中提取字幕文件

- 前提
  - 原视频中有字幕（的流）= 有soft `软字幕`
    - 特殊情况：
      - 有种字幕是被合并编码成视频数据本身的情况，是没有字幕的
        - 这种视频 表面上看有字幕显示出来，但是无法提取出字幕的

从视频中提取srt字幕

命令：

```
ffmpeg -i video_file.mp4 -map 0:s:0 subtitle.srt
```

参数含义：

- `-i video_file.mp4`：
  - input输入文件是video_file.mp4
- `-map 0:s:0 subtitle.srt` 的含义：
  - `-map`：高级参数
  - `0:s:0`：
    - `0` -> `input_file_id` = `文件id`，输入的文件索引编号
      - 此处就一个文件，所以 `0` 表示此处输入的mp4视频：video_file.mp4
    - `:s` -> `:stream_specifier`，流stream选择的是字幕subtitle
    - `:0` -> `:stream_specifier` 第 `0` 个字幕
      - 此处只有一个字幕，就是这个唯一的字幕
  - `subtitle.srt`：输出的字幕文件名

转换字幕

从srt转换出ass字幕文件

```
ffmpeg -i subtitle.srt subtitle.ass
```

举例：

```
➜  ffmpeg_edit_subtitle ffmpeg -i subtitle.srt subtitle.ass
ffmpeg version 3.4.2 Copyright (c) 2000-2018 the FFmpeg dev
  built with Apple LLVM version 9.0.0 (clang-900.0.39.2)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/3.4.2 --
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Input #0, srt, from 'subtitle.srt':
  Duration: N/A, bitrate: N/A
    Stream #0:0: Subtitle: subrip
Output #0, ass, to 'subtitle.ass':
  Metadata:
    encoder         : Lavf57.83.100
    Stream #0:0: Subtitle: ass (ssa)
    Metadata:
      encoder         : Lavc57.107.100 ssa
Stream mapping:
  Stream #0:0 -> #0:0 (subrip (srt) -> ass (ssa))
Press [q] to stop, [?] for help


size=      33kB time=00:26:17.56 bitrate=   0.2kbits/s spee
video:0kB audio:0kB subtitle:23kB other streams:0kB global
```

嵌入字幕

用ffmpeg去 `嵌入字幕` = `写入字幕` = `合并字幕` = `烧录字幕`

分2类：

软字幕=内挂字幕

- 别称： `软字幕` = `soft subs` = `soft subtitles`
- ffmpeg处理逻辑： `stream copy` **流拷贝**模式
- 举例：

```
ffmpeg -i input.mkv -i subtitles.ass -codec copy -map

ffmpeg -i infile.mp4 -f srt -i infile.srt -c:v copy -
```

硬字幕=内嵌字幕

- 别称： `硬字幕` = `hardsubs` = `hard subs` = `hard subtitles` = `burn in subtitles`
- ffmpeg处理逻辑：用 `-vf` 加上**字幕文件**
- 前提：ffmpeg支持ass
  - 即：ffmpeg编译参数中包含： `--enable-libass`
    - 举例

```
# ffmpeg -version
ffmpeg version 4.2-tessus  https://evermeet.c
built with Apple LLVM version 10.0.1 (clang-1
configuration: --cc=/usr/bin/clang --prefix=/
```

- 举例
  1. 嵌入srt字幕

```
ffmpeg -i input.mp4 -vf "subtitles=subtitle.srt" ou
```

    - 注
      - (1)-vf后面参数，可以不加双引号：

```
ffmpeg -i input.mp4 -vf subtitles=subtitle.s
```

        - 不过为了区别后续其他参数，最好像上面一样加上双引号，比较容易和其他参数区别开
      - (2)如果srt字幕内在在视频文件中

- 比如视频是mkv的容器，内挂有srt字幕，则可以：

```
ffmpeg -i input.mkv -vf subtitles=input
```

2. 嵌入ass字幕

```
ffmpeg -i input.mp4 -vf "ass=subtitle.ass" output.m
```

- 举例

```
ffmpeg -i CTT_Folge_01_CH_Subs_DefaultZhcnButNo
```

- 特殊
  - 另外ffmpeg也支持：`Picture-based subtitles`=基于图片的字幕
    - 命令举例：

```
ffmpeg -i input.mkv -filter_complex "[0
```

    - 参数说明：
      - 如果是多个stream流，可以把`[0:s]`换成对应的`[0:s:0]`或`[0:s:1]`
- 注意：
  - （1）当视频有多个音频流，此种方式可能会破坏编码，则用下面方式去修复：

```
ffmpeg -i input.ts -filter_complex "[0:v][0
```

  - （2）Windows环境：注意设置字体的路径

指定字幕位置

- 最简单但常见的需求：无需操心字幕的具体位置，只需要保证字幕在视频底部
  - 则可以直接嵌入字幕，其中字幕文件是srt或ass均可

```
ffmpeg -i input.mp4 -vf subtitles=input.srt output.
ffmpeg -i input.mp4 -vf ass=subtitle.ass output.mp4
```

- 高级需求：指定字幕的具体的位置（不同区域，具体边距等）
  - 前提：必须是ass文件（才能用参数指定字幕位置），srt无法指定字幕位置
    - 如果是srt文件，则需要先去转换成ass文件

```
ffmpeg -i input.srt output.ass
```

嵌入ass字幕到视频中

```
ffmpeg -i input.mp4 -vf "ass=input.ass" output.mp4
```

- 其中：
  - ass字幕文件input.ass中，有对应的位置的参数配置

```
[V4+ Styles]
Format: Name, Fontname, Fontsize, PrimaryColour, Se

Style: Default,Arial,16,&Hffffff,&Hffffff,&H0,&H0,0
```

    - 其中：
      - `Alignment`：默认为`2` = 底部居中
        - 就满足了我们希望的：字幕在底部居中的位置

微调左右间距和底部间距

- 再去微调左右间距和底部间距时
  - 再去改动：
    - `MarginL, MarginR, MarginV`
  - 比如：

```
MarginL, MarginR, MarginV
20,20,10
```

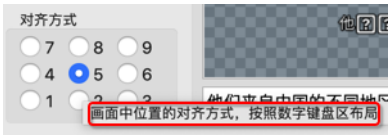
    - 即可实现：
        - `MarginL` = `MarginR` ：左右间距 `20`
        - `MarginV` ：底部向上间距 `10`

嵌入ass字幕举例

- 字幕文件：`input/5d41d82f52247ce73d40475b_cfgPosition.ass`
    - ass文件中相关参数

```
Format: Name, Fontname, Fontsize, PrimaryColour, Se

Style: Default,Arial,16,&Hffffff,&Hffffff,&H0,&H0,0
```

        - 参数说明
            - 字幕位置主要影响因素
                - 字幕主体所在区域
                    - `Alignment`
                        - 默认为 `2` = 底部居中
                        - 其中值：`1-9`
                            - 逻辑：和键盘中数字的排布一致
                            - 见图：
                            - 

            - 字幕位置次要影响因素
                - 字幕的边距
                    - （字幕距离）左右的边距：`MarginL` 和 `MarginR` ： `20`
                    - （字幕距离）底部的边距：`10`
- 命令

```
ffmpeg -i input/5d41d82f52247ce73d40475b_extendedAll.mp
```

- 效果

- 

指定字幕文字属性

此处介绍嵌入字幕时，指定字幕文字的各种属性，比如 `字体大小` 、 `字体类型` 、 `颜色` 、 `透明度` 等

- srt字幕：加force_style参数

```
ffmpeg -i video.mp4 -vf "subtitles=subs.srt:force_sty
```

- ass字幕：在ass字幕中设置参数

```
Fontname, Fontsize, PrimaryColour, SecondaryColour, C
```

具体设置成什么值，以及效果如何，可借助于软件Aegisub去设置和预览

举例1

```
Style: Transparent,PingFang SC,20,&H00FFFFFF,&H000000FF,&HB
```

实现了字幕效果：

- 字体：PingFang SC
- 字体大小：20
- 字幕的背景半透明效果：后面很多参数组合的效果

如图：

举例2：ass设置半透明的背景

```
Style: Default,Arial,16,&H00FFFFFF,&H000000FF,&H80000000,&H
```

即可达到要的字幕有个半透明的背景色了。

其中 `16` 指的是 `字体大小` ，可以根据需要更改为自己要的值。

ffmpeg使用心得

此处整理ffmpeg的一些使用心得。

报错：[AVFilterGraph @ 0x7fa224608740] No such filter: 'ass'

```
ffmpeg -i CTT_Folge_01_CH_Subs_DefaultZhcnButNotShow.mp4 -v
```

报错：

```
[AVFilterGraph @ 0x7fa224608740] No such filter: 'ass'
```

原因：此处ffmpeg（没有subtitles，也）没有ass的filter

办法：

重新安装支持了ass的ffmpeg

具体步骤：

Mac

```
brew reinstall ffmpeg --with-libass
```

报错：Error: Calling keg_only :provided_pre_mountain_lion is disabled! There is no replacement

重新安装ffmpeg期间：

```
ffmpeg_edit_subtitle brew reinstall ffmpeg \
    --with-tools \
    --with-fdk-aac \
    --with-freetype \
    --with-fontconfig \
    --with-libass \
    --with-libvorbis \
    --with-libvpx \
    --with-opus \
    --with-x265
```

报错：

```
Error: Calling keg_only :provided_pre_mountain_lion is disa
```

解决办法：

先去更新brew再升级ffmpeg

```
brew update && brew upgrade ffmpeg
```

再去：

```
brew reinstall ffmpeg 加上你的要的参数
```

即：

```
brew reinstall ffmpeg \
    --with-tools \
    --with-fdk-aac \
    --with-freetype \
    --with-fontconfig \
    --with-libass \
    --with-libvorbis \
    --with-libvpx \
    --with-opus \
    --with-x265
```

ffmpeg不显示头部信息

用ffmpeg去处理视频等过程中，每次运行命令，都会输出ffmpeg版本等信息

举例：

```
# ffmpeg  -nostdin -i /root/xxx/course/32145/course_32145_v
ffmpeg version 4.2-static https://johnvansickle.com/ffmpeg/
  built with gcc 6.3.0 (Debian 6.3.0-18+deb9u1) 20170516
  configuration: --enable-gpl --enable-version3 --enable-st
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libswscale      5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from '/root/xxx/course/3
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    creation_time   : 2015-11-16T08:09:13.000000Z
    encoder         : Lavf57.71.100
  Duration: 00:00:46.00, start: 0.000000, bitrate: 678 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x3163766
    Metadata:
      creation_time   : 2015-11-16T08:09:13.000000Z
      handler_name    : VideoHandler
    Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D),
    Metadata:
      creation_time   : 2015-11-16T08:09:13.000000Z
      handler_name    : SoundHandler
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> h264 (libx264))
  Stream #0:1 -> #0:1 (copy)
[libx264 @ 0xce0e540] using SAR=1/1
[libx264 @ 0xce0e540] using cpu capabilities: MMX2 SSE2Fast
[libx264 @ 0xce0e540] profile Progressive High, level 3.0,
[libx264 @ 0xce0e540] 264 - core 157 r2969 d4099dd - H.264/
Output #0, mp4, to
......
```

中的：

```
ffmpeg version 4.2-static https://johnvansickle.com/ffmpeg/
  built with gcc 6.3.0 (Debian 6.3.0-18+deb9u1) 20170516
  configuration: --enable-gpl --enable-version3 --enable-st
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libswscale      5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
```

这部分内容。

如果觉得很麻烦，想要禁止其输出，可以加上参数：

`-hide_banner`

用于禁止显示banner，即可。

ffmpeg不显示输出的众多的普通的信息

另外，如果对于处理期间输出的众多内容，不想要显示，则可以加：

`-loglevel error`

用于设置当输出的信息，超过error，才显示

而上面信息都是属于普通的info级别，低于error，所以就不显示

起到控制不显示普通的输出的信息的效果了

注：

ffmpeg输出信息的等级：

- `quiet`
- `panic`
- `fatal`
- `error`
- `warning`
- `info`
- `verbose`
- `debug`

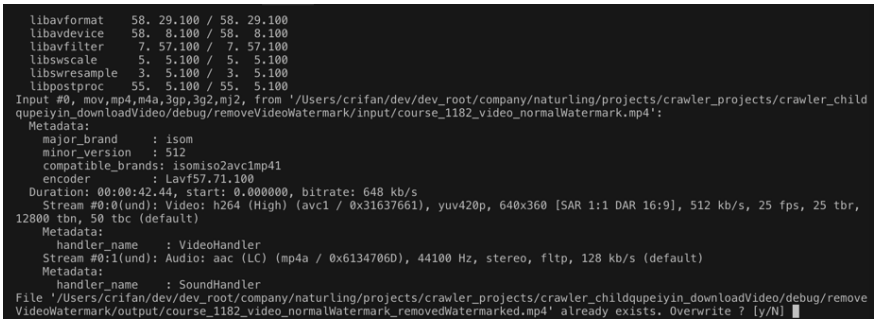
ffmpeg强制覆盖输出文件

ffmpeg强制覆盖输出文件 = 不需要当每次检测到已存在文件再提示你是否要覆盖

用ffmpeg时，如果输出文件已存在，默认不会覆盖，会提示你：

```
File xxx.mp4 already exists. Overwrite ? [y/N]
```

要输入y，才可以继续：



如果不想要提示，而是强制覆盖源文件，则可以加上参数：

```
-y
```

相关参数含义：

```
# ffmpeg --help
...
-y                  overwrite output files
```

用到ffmpeg的

此处整理，内部用到了ffmpeg的一些相关内容：

- `pydub`：Python的音频处理库
  - jiaaro/pydub: Manipulate audio with a simple and easy high level interface
    - https://github.com/jiaaro/pydub
      - 底层是用ffmpeg去实现音视频的处理的
- `audioread`：Python的音频解析库
  - beetbox/audioread: cross-library (GStreamer + Core Audio + MAD + FFmpeg) audio decoding for Python
    - https://github.com/beetbox/audioread
      - 底层是通过ffmpeg去解析出音频信息
- `Gifski`
  - GitHub
    - sindresorhus/Gifski: 🌈 Convert videos to high-quality GIFs on your Mac
      - https://github.com/sindresorhus/Gifski
  - 举例
    - 把多张PNG图片，转换成gif动图：

```
TMPFILE="$(mktemp /tmp/XXXXXXXXXX).mov"; \
    ffmpeg -f image2 -framerate 30 -i image_%
    && open -a Gifski "$TMPFILE"
```

Python

此处整理，用Python调用ffmpeg，实现音视频处理期间的相关内容。

ffmpeg相关库函数

之前已把相关的，Python中调用ffmpeg去处理音视频相关的功能，封装成函数：

https://github.com/crifan/crifanLibPython/blob/master/python3/crifanLib/thirdParty/crifanFfmpeg.py

其中就有：

- `removeVideoWatermark`：去除视频水印
- `detectVideoDimension`：获取视频属性

需要的，可以直接拿去用。

用Python调用ffmpeg

用python代码调用ffmpeg去从mp4中（根据字幕信息）截图mp3音频片段

```
import subprocess
# extract audio segment from video
# ffmpeg -i show_157712932_video.mp4 -ss 00:00:11.270 -to (
if not os.path.exists(curAudioSegmentFullpath):
  ffmpegCmd = "ffmpeg -i %s -ss %s -to %s -b:a 128k %s" % (
  subprocess.call(ffmpegCmd, shell=True)
```

心得：加 `-nostdin` 避免后台模式运行时卡死

Python代码中调用ffmpeg去处理视频，比如：

```
ffmpeg -i input_video.mp4 -vf "delogo=x=474:y=6:w=162:h=90"
```

然后正常运行Python时：

```
python3 processCourseVideo.py
```

没问题。

后台运行时：

```
python3 processCourseVideo.py &
```

结果会卡死。

后来发现：需要给 `ffmpeg` 加 `-nostdin` 参数：

```
ffmpeg -nostdin -i input_video.mp4 -vf "delogo=x=474:y=6:w=
```

能避免卡死。

详见：

【已解决】后台运行Python代码导致subprocess.check_call调用ffmpeg卡死

附录

下面列出相关参考资料。

help语法

贴出（后来重新安装的支持了ass等库的）ffmpeg的help帮助信息：

```
➜  ~ ffmpeg --help
ffmpeg version 4.0.2 Copyright (c) 2000-2018 the FFmpeg dev
  built with Apple LLVM version 10.0.0 (clang-1000.11.45.2)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/4.0.2 --
  libavutil      56. 14.100 / 56. 14.100
  libavcodec     58. 18.100 / 58. 18.100
  libavformat    58. 12.100 / 58. 12.100
  libavdevice    58.  3.100 / 58.  3.100
  libavfilter     7. 16.100 /  7. 16.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  1.100 /  5.  1.100
  libswresample   3.  1.100 /  3.  1.100
  libpostproc    55.  1.100 / 55.  1.100
Hyper fast Audio and Video encoder
usage: ffmpeg [options] [[infile options] -i infile]... {[o

Getting help:
    -h      -- print basic options
    -h long -- print more options
    -h full -- print all options (including all format and
    -h type=name -- print all options for the named decoder
    See man ffmpeg for detailed description of the options.

Print help / information / capabilities:
-L                  show license
-h topic            show help
-? topic            show help
-help topic         show help
--help topic        show help
-version            show version
-buildconf          show build configuration
-formats            show available formats
-muxers             show available muxers
-demuxers           show available demuxers
-devices            show available devices
-codecs             show available codecs
-decoders           show available decoders
-encoders           show available encoders
-bsfs               show available bit stream filters
-protocols          show available protocols
-filters            show available filters
-pix_fmts           show available pixel formats
-layouts            show standard channel layouts
-sample_fmts        show available audio sample formats
-colors             show available color names
-sources device     list sources of the input device
-sinks device       list sinks of the output device
-hwaccels           show available HW acceleration methods

Global options (affect whole program instead of just one fi
```

```
-loglevel loglevel  set logging level
-v loglevel         set logging level
-report             generate a report
-max_alloc bytes    set maximum size of a single allocated
-y                  overwrite output files
-n                  never overwrite output files
-ignore_unknown     Ignore unknown stream types
-filter_threads     number of non-complex filter threads
-filter_complex_threads  number of threads for -filter_comp
-stats              print progress report during encoding
-max_error_rate maximum error rate  ratio of errors (0.0: r
-bits_per_raw_sample number  set the number of bits per raw
-vol volume         change audio volume (256=normal)

Per-file main options:
-f fmt              force format
-c codec            codec name
-codec codec        codec name
-pre preset         preset name
-map_metadata outfile[,metadata]:infile[,metadata]  set met
-t duration         record or transcode "duration" seconds
-to time_stop       record or transcode stop time
-fs limit_size      set the limit file size in bytes
-ss time_off        set the start time offset
-sseof time_off     set the start time offset relative to E
-seek_timestamp     enable/disable seeking by timestamp wit
-timestamp time     set the recording timestamp ('now' to s
-metadata string=string  add metadata
-program title=string:st=number...  add program with specit
-target type        specify target file type ("vcd", "svcd"
-apad               audio pad
-frames number      set the number of frames to output
-filter filter_graph  set stream filtergraph
-filter_script filename  read stream filtergraph descriptio
-reinit_filter      reinit filtergraph on input parameter c
-discard            discard
-disposition        disposition

Video options:
-vframes number     set the number of video frames to outpu
-r rate             set frame rate (Hz value, fraction or a
-s size             set frame size (WxH or abbreviation)
-aspect aspect      set aspect ratio (4:3, 16:9 or 1.3333,
-bits_per_raw_sample number  set the number of bits per raw
-vn                 disable video
-vcodec codec       force video codec ('copy' to copy strea
-timecode hh:mm:ss[:;.]ff  set initial TimeCode value.
-pass n             select the pass number (1 to 3)
-vf filter_graph    set video filters
-ab bitrate         audio bitrate (please use -b:a)
-b bitrate          video bitrate (please use -b:v)
```

```
-dn                 disable data

Audio options:
-aframes number     set the number of audio frames to outpu
-aq quality         set audio quality (codec-specific)
-ar rate            set audio sampling rate (in Hz)
-ac channels        set number of audio channels
-an                 disable audio
-acodec codec       force audio codec ('copy' to copy strea
-vol volume         change audio volume (256=normal)
-af filter_graph    set audio filters

Subtitle options:
-s size             set frame size (WxH or abbreviation)
-sn                 disable subtitle
-scodec codec       force subtitle codec ('copy' to copy st
-stag fourcc/tag    force subtitle tag/fourcc
-fix_sub_duration   fix subtitles duration
-canvas_size size   set canvas size (WxH or abbreviation)
-spre preset        set the subtitle options to the indicat
```

供参考。

文档

- ffmpeg的官网文档
  - ffmpeg Documentation
    - https://ffmpeg.org/ffmpeg.html
  - Documentation
    - https://www.ffmpeg.org/documentation.html
  - ffmpeg Documentation
    - https://www.ffmpeg.org/ffmpeg-all.html
  - FFmpeg Utilities Documentation
    - https://ffmpeg.org/ffmpeg-utils.html
  - FFmpeg Codecs Documentation
    - https://www.ffmpeg.org/ffmpeg-codecs.html

参考资料

- 【已解决】嵌入ASS字幕到mp4视频中如何指定字幕的位置
- 【已解决】把视频画面增加高度再合并字幕到指定位置
- 【已解决】ffmpeg保持视频宽度不变去增加扩大视频高度
- 【已解决】ffmpeg合并字幕到视频下方指定位置
- 【整理】ASS字幕文件格式详解
- 【已解决】ffprobe作用及用法和其于ffmpeg的区别
- 【已解决】ffmpeg查看视频的基本信息包括字幕信息
- 【已解决】Mac中用ffmpeg调整mp4默认字幕为中文
- 【已解决】Mac中用ffmpeg从mp4视频中提取mp3音频
- 【已解决】Mac中安装ffmpeg及相关ffplay和ffprobe等工具
- 【已解决】用ffmpeg去除mp4视频水印
- 【已解决】Python代码其中可利用ffmpeg检测视频真实分辨率即宽度和高度
- 【已解决】mac中下载和安装最新版的ffmpeg
- 【已解决】ffmpeg从mp4视频提取出srt和ass字幕文件
- 【已解决】用Aegisub字幕编辑器去调整字体大小和字幕背景半透明效果
- 【已解决】ass字幕文件中如何设置字幕的半透明背景色
- 【已解决】ffmpeg集成srt字幕到视频的字幕流中即软字幕
- 【已解决】调节ass字幕配置字幕字体大小和背景色再用ffmpeg嵌入视频中
- 【已解决】后台运行Python代码导致subprocess.check_call调用ffmpeg卡死
- 【已解决】用Python代码从视频中提取出音频mp3文件
-
- 【基本解决】Mac中提取mp4视频中的字幕
- 【已解决】Mac中brew reinstall ffmpeg出错：Error: Calling keg_only :provided_pre_mountain_lion is disabled! There is no replacement
-
- 【基本解决】Python中把wma、wav等格式音频转换为mp3
- 【已解决】Mac中播放某mp4视频默认没有字幕
- 【基本解决】Handbrake压制mp4时添加SRT字幕文件
- 【已解决】ffmpeg嵌入硬编码烧录ass字幕到视频中即字幕成为视频内容本身
- 【已解决】Mac中尝试给mp4内嵌字幕出错：AVFilterGraph No such filter ass
- 【已解决】Mac中用ffmpeg调整mp4默认字幕为中文
- 【已解决】ffmpeg集成srt字幕到视频的字幕流中即软字幕
- 【已解决】Mac中找软件编辑mp4视频的默认字幕为中文

- 【已解决】用ffmpeg从mp4视频中提取出整个mp3以及根据时间段去分割mp3
-
- Stream-copy ffmpeg Documentation
- toc-Main-options
- sindresorhus/Gifski: 🌈 Convert videos to high-quality GIFs on your Mac
- 使用 ffmpeg 实现 MP4 与 GIF 的互转 - 任平生的学习笔记
- HowToBurnSubtitlesIntoVideo – FFmpeg
- FFmpeg Filters Documentation - subtitles
- FFmpeg Filters Documentation - ass
- enable libass - Page 3 - FFmpeg
- SubStation Alpha - 维基百科，自由的百科全书
- ffserver – FFmpeg
- ffmpeg——ffserver的一个简单demo - Eyrane的博客 - CSDN博客
- ffmpeg+ffserver搭建流媒体服务器 - 静之深 - 博客园
- ffmpeg Documentation
- Documentation
- ffmpeg Documentation
- FFmpeg Utilities Documentation
- FFmpeg Codecs Documentation
-