



```
ora, yuv420p, 320x176, 25 fps, 25 tbr, 25 tbn, 25 tbc
8 |
ais, 48000 Hz, stereo, fltp, 160 kb/s
```



MPEG传输流解复用为单独的基本(音频和视频)流，并将它们保存在保存编解码器的MP4容器中(注意这里如何使用列表作为选项):

分离，音频放入了audio.mp4文件，视频在video.mp4文件

```
{
    'out.ts': None,
    'video.mp4': ['-map', '0:0', '-c:a', 'copy', '-f', 'mp4'],
    'audio.mp4': ['-map', '0:1', '-c:a', 'copy', '-f', 'mp4']}
}
```

```
24      媒体文件(.mp4)      468 KB  00:00:09
24      媒体文件(.mp4)      296 KB  00:00:09
```

```
0:0 -c:a copy -f mp4 video.mp4 -map 0:1 -c:a copy -f mp4 audio.mp4
```

```
.o.mp4':
```

```
(mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 384 kb/s
```

```
o.mp4':
```

```
1 (libx264) (avc1 / 0x31637661), yuv420p, 320x176 [SAR 1:1 DAR 20:11], q=-1--1, 25 fps, 12800 tbn, 25 tbc
```

式格式，也就是说，不能有包含空格字符串的列表，除非不含有列表，将命令行作为一个整体字符串，即：['-map', '0:0', '-c:a', 'copy', '-f', 'mp4'] 或 '-map 0

回MPEG传输流，即音频与视频的合成：

```
{
    'video.mp4': None, 'audio.mp4': None,
    'input.ts': ['-c:v h264 -c:a ac3']}
}
```

```
14 (native) -> h264 (libx264))
```

```
1 (mp3float) -> ac3 (native))
```

入和输出的顺序(例如使用FFmpeg -map选项时)。在这些情况下，使用常规Python字典将不起作用，因为它不能保持顺序。相反,使用OrderedDict。例如，编解码器对两个音频流进行重新编码。这里，我们使用OrderedDict来保存输入的顺序，以便它们匹配输出选项中的流的顺序：

```
{
    OrderedDict(
        video.mp4', None), ('audio_1.mp3', None), ('audio_2.mp3', None),
        '-map 0 -c:v h264 -map 1 -c:a:0 ac3 -map 2 -c:a:1 mp3')
}
```



Jumping boy

关注

8



60



6



专栏

```
:s, outputs=outputs) 6 | print(ff.cmd)

audio_1.mp3 -i audio_2.mp3 -map 0 -c:v h264 -map 1 -c:a:0 ac3 -map 2 -c:a:1 mp2 output.ts

p,3g2,mj2, from 'video.mp4':

    h264 (High) (avc1 / 0x31637661), yuv420p, 320x176 [SAR 1:1 DAR 20:11], 241 kb/s, 25 fps, 25 tbr, 12800 tbn, 50 tbc (default)

p,3g2,mj2, from 'audio_1.mp3':

    mp3 (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 384 kb/s (default)

p,3g2,mj2, from 'audio_2.mp3':

    mp3 (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 384 kb/s (default)

i4 (native) -> h264 (libx264))

} (mp3float) -> ac3 (native))

} (mp3float) -> mp2 (native))

output.ts':

    h264 (libx264), yuv420p(progressive), 320x176 [SAR 1:1 DAR 20:11], q=-1--1, 25 fps, 90k tbn, 25 tbc (default)

    ac3, 48000 Hz, stereo, fltp, 192 kb/s (default)

    mp2, 48000 Hz, stereo, s16, 384 kb/s (default)

ird input)读取输入，并将输出写入STDOUT(standard output)。这可以通过使用FFmpeg管道协议来实现。下面的示例从包含RGB格式原始视频帧的文件中
台帧数据，并将其打包到一个MP4容器中，将输出传递给STDOUT(注意，必须使用子进程将STDOUT重定向到管道中。管道为stdout值，否则输出将丢失):

'g

re:0': '-f rawvideo -pix_fmt rgb24 -s:v 640x480'},
.re:1': '-c:v h264 -f mp4'})

input_data=open('rawvideo', 'rb').read(), stdout=subprocess.PIPE)

望运行FFmpeg并阻塞等待结果，或者在应用程序中引入多线程。在这种情况下，使用asyncio进行异步执行是可能的。

塞的情况下处理FFmpeg输出。
```

```
'g
out.ts': None},
itput.ts': ['-vf', 'adif=0:-1:0, scale=iw/2:-1'])})

out.ts': None},
itput.ts': ['-vf "adif=0:-1:0, scale=iw/2:-1"'])})

'adif=0:-1:0, scale=iw/2:-1" output.ts
'adif=0:-1:0, scale=iw/2:-1" output.ts

:it code 0
```

， 它将时间码烧录成视频:

```
rtext=fontfile=/Library/Fonts/Verdana.ttf: timecode='09\:57\:00\:00': r=25: x=(w-tw)/2: y=h-(2*lh): fontcolor=white: box=1: boxcolor=0x000000

'g
out.ts': None},
itput.ts': ['-vf', "drawtext=fontfile=/Library/Fonts/Verdana.ttf: timecode='09\:57\:00\:00': r=25: x=(w-tw)/2: y=h-(2*lh): fontcolor=white: b
```

传递输出选项作为一个字符串，同时保留引号:

```
'g
out.ts': None},
itput.ts': "-vf \"drawtext=fontfile=/Library/Fonts/Verdana.ttf: timecode='09\:57\:00\:00': r=25: x=(w-tw)/2: y=h-(2*lh): fontcolor=white: box
```

这不就是把文档翻译了一下吗？英文变成了中文，没有任何其他东西啊

中将视频转换为音频\_ffmyy

简单的 FFmpeg 命令行包装程序。ffmpeg 实现了一个 Pythonic 接口,用于通过命令行执行 FFmpeg,并使用 Python 的子进程模块进行同步执行。使用 Python 的 asyncio.subproce

.ffmpeg.ffexecutablenotfounderror: executa...

Executable'ffmpeg'notfound 解决办法: 解决方法一:解压ffmpeg文件,将ffmpeg文件中的可执行文件ffmpeg.exe复制到当前项目文件目录下 解决方法二:打开ffmpeg3.py文件,将 ...

y3的使用

令, 以及python和ffmpeg的结合ffmpeg3的使用

3的错误

的错误

eg下载m3u8转换为mp4过程中, CPU占用过高的问题

载m3u8转换为mp4过程中, CPU占用过高的问题

法使用pip工具安装“ffmpeg”包的问题

ffmpeg安装版本, 但是可能是嵌入版和安装版的文件有差异, 导致这段代码会出现异常, 导致失败(以上是猜测, 没具体测试过, 这里只说解决方案)3、修改其中的两个位置的代码

安装

例如: C:\Program Files\Python37\Scripts>pip install ffmpeg==0.2.2 Collecting ffmpeg==0.2.2 Using cached https://files.pythonhosted.org/packages/18/e6/4309f4c02d...

python中使用ffmpeg (保姆级图文) 热门推荐

py (保姆级图文)

使用ffmpeg+python中使用ffmpeg

中的ffmpeg, ffmpeg的使用又会用到ffmpeg, 接下来就按照顺序依次介绍ffmpeg的安装和在Python中使用ffmpeg。1.ffmpeg下载安装官方下载地址: https://ffmpeg.org/download.

频转化为mp3格式

化为mp3格式



Jumping boy

关注

8



60



6



专栏

ipeg/builds/在中下载。

以钟将视频转换为音频

目 (ID: PythonCircle) 最近, 有读者微信上私聊我, 想让我写一篇视频批量转换成音频的文章, 我答应了, 周末宅家里把这个小工具做出来了。这样, 对于有些视频学习文件,

发现无法直接导出成电脑可以播放的视频, 因为这个格式其实只是类似一个播放列表, 引用了一些一大堆的视频片段, 所以这里写一个使用开源软件ffmpeg合并的教程, 就可以

ke代码中实现wav格式音频转为mp3音频格式。

用方法解析

工具使用 方法解析,文中通过示例代码介绍的非常详细, 对大家的学习或者工作具有一定的参考学习价值,需要的朋友可以参考下

mp3

MP3格式, 可以使用FFmpeg工具。以下是使用命令行的示例: `` ffmpeg -i input.mp4 -vn -acodec libmp3lame -q:a 2 output.mp3 `` 请确保你已经安装了FFmpeg, 并将`input.m

文件异常问题及解决方法

3音频文件异常问题及解决方法,本文给大家介绍的非常详细, 对大家的工作或学习具有一定的参考借鉴价值, 需要的朋友可以参考下

新发布

旨在帮助开发者实现特定的编程任务, 无需从零开始编写代码。这些库可以包括各种功能, 如数学运算、文件操作、数据分析和网络编程等。Python社区提供了大量的第三方

学资源库.zip

原库.zip

ffmpeg.FFmpeg(inputs={'input\_file': 'path/to/input\_file'}, outputs={'output\_file': 'path/to/output\_file'})` 在这个例子中, `input\_file`是输入文件的路径, `output\_file`...

“相关推荐”对你有帮助么？

- 
- 非常有帮助
- 
- 没帮助
- 
- 一般
- 
- 有帮助
- 
- 非常有帮助

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照

©1999-2024北京创新乐知网络技术有限公司



等级

553

收藏



复



Jumping boy

关注

8



60



6



专栏