FFmpeg最全命令合辑

FFmpeg/WebRTC/RTMP/RTSP/HLS/播放器-音视频流媒体高级开发 https://ke.qq.com/course/468797?tuin=137bb271

官方文档：

https://www.ffmpeg.org/ffplay-all.html

https://www.ffmpeg.org/ffmpeg-all.html

01-Windows FFMPEG命令行环境搭建

1. 到ffmpeg官方下载已经编译好的Windows shared库；

2. 将执行文件ffmpeg.exe ffplay.exe ffprobe.exe拷贝到C:\Windows目录;

3. 将相应的动态库拷贝到C:\Windows\SysWOW64目录；

注：WOW64 (Windows-on-Windows 64-bit)

4. 在命令行窗口输入ffmpeg -version 查看版本，以却确定环境是否搭建成功。

02-FFMPEG如何查询命令帮助文档

ffmpeg/ffplay/ffprobe区别

- ffmpeg:
  Hyper fast Audio and Video encoder

  超快音视频编码器（类似爱剪辑）
- ffplay:
  Simple media player简单媒体播放器
- ffprobe:
  Simple multimedia streams analyzer

  简单多媒体流分析器

ffmpeg命令查看帮助文档

基本信息：ffmpeg –h

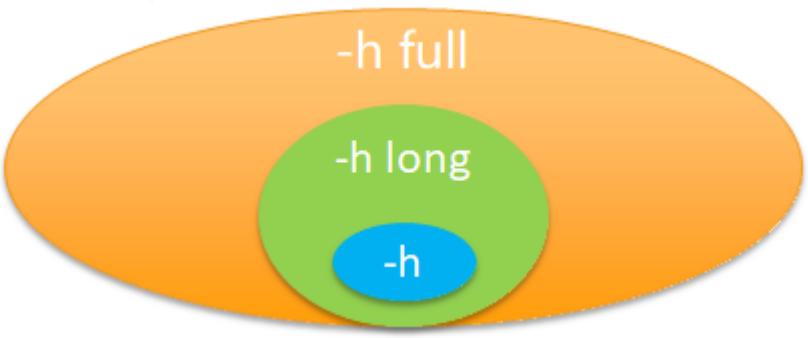
高级信息：ffmpeg –h long

所有信息：ffmpeg –h full

若嫌打印过多： ffmepg –h full > ffmpeg_h_full.log，然后再打开ffmpeg_h_full.log文件查看

usage:

ffmpeg [options] [[infile options] –i infile]... {[outfile options] outfile}...



ffplay命令查看帮助文档

所有信息：ffplay –h

usage:

ffplay [options] input_file

ffprobe命令查看帮助文档

所有信息：ffprobe –h

usage:

ffprobe [OPTIONS] [INPUT_FILE]

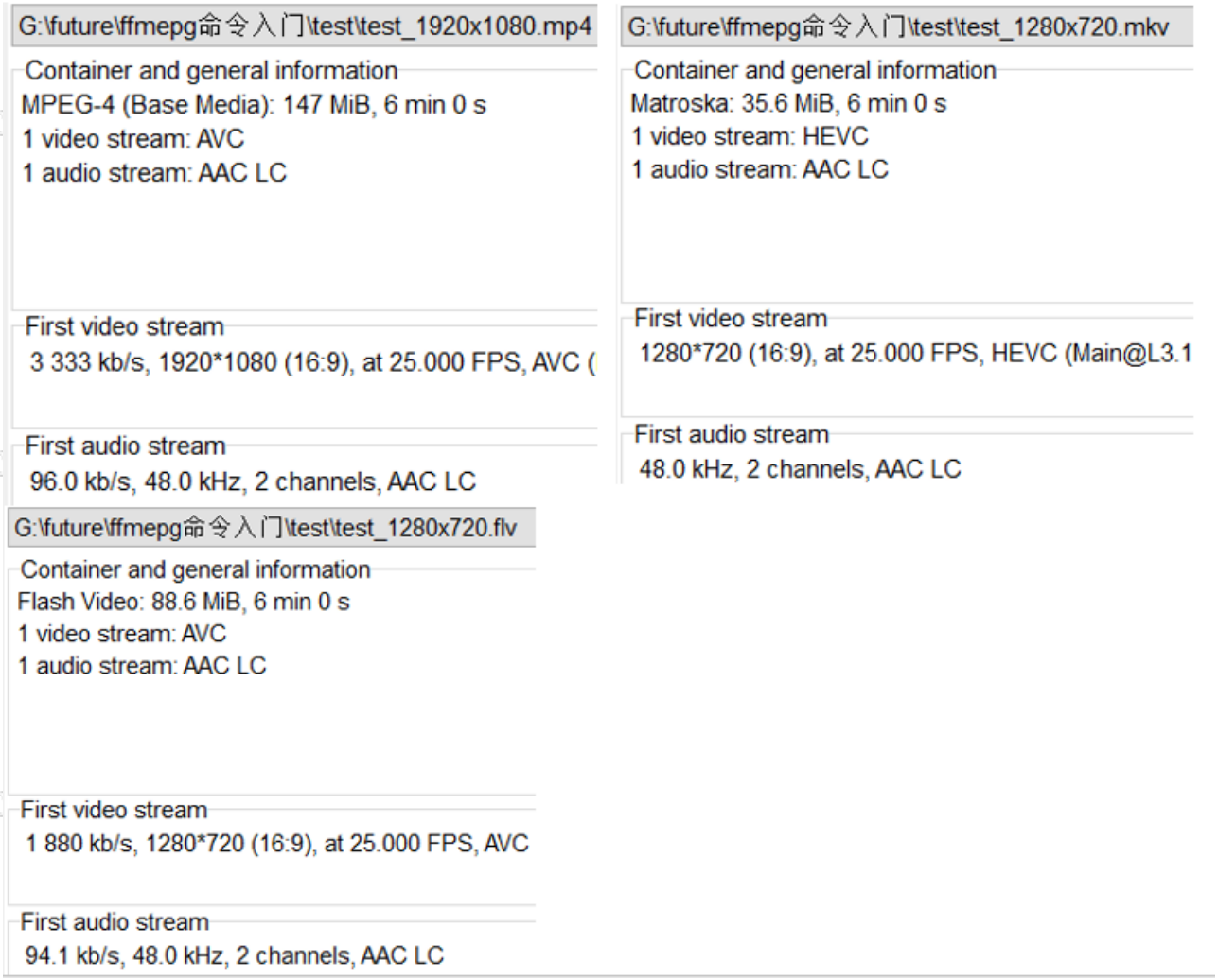
ffmpeg/ffplay/ffprobe部分参数通用，部分参数不通用，在使用时需要注意。

03-ffmpeg音视频处理流程

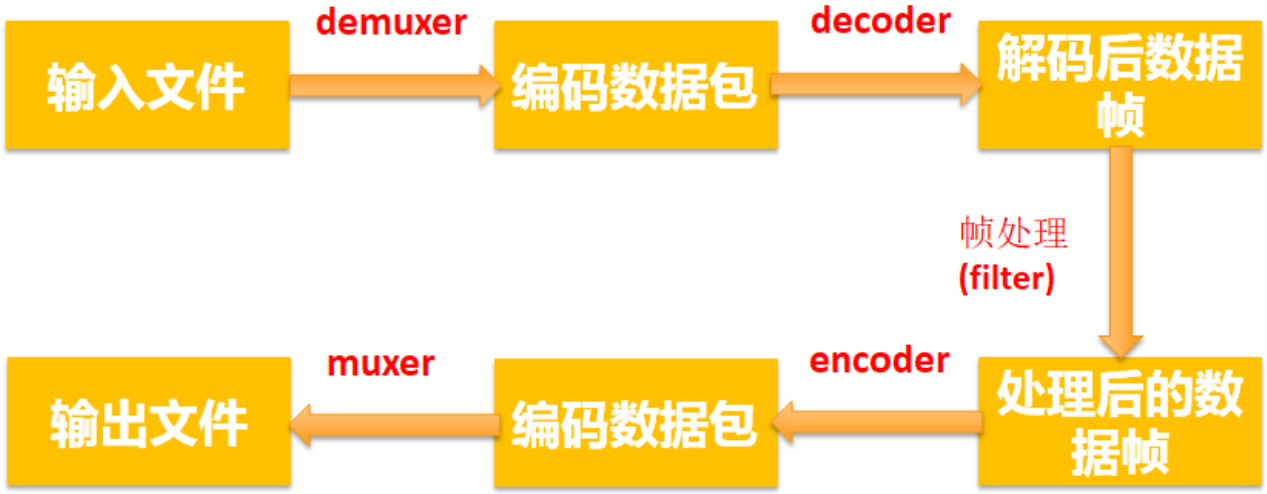
先看两条命令

ffmpeg -i test_1920x1080.mp4 -acodeccopy -vcodeclibx264 -s **1280x720**test_1280x720.**flv**

ffmpeg -i test_1920x1080.mp4 -acodeccopy -vcodeclibx265 -s**1280x720**test_1280x720.**mkv**



ffmpeg音视频处理流程



04–ffmpeg命令分类查询

ffmpeg命令分类查询

命令参数	内容	命令参数	内容
–version	显示版本	–bsfs	显示可用比特流filter
–buildconf	显示编译配置	–protocols	显示可用的协议
–formats	显示可用格式(muxers+demuxers)	–filters	显示可用的过滤器
–muxers	显示可用复用器	–pix_fmts	显示可用的像素格式
–demuxers	显示可用解复用器	–layouts	显示标准声道名称
–codecs	显示可用编解码器(decoders+encoders)	–sample_fmts	显示可用的音频采样格式
–decoders	显示可用解码器	–colors	显示可用的颜色名称
–encoders	显示可用编码器		

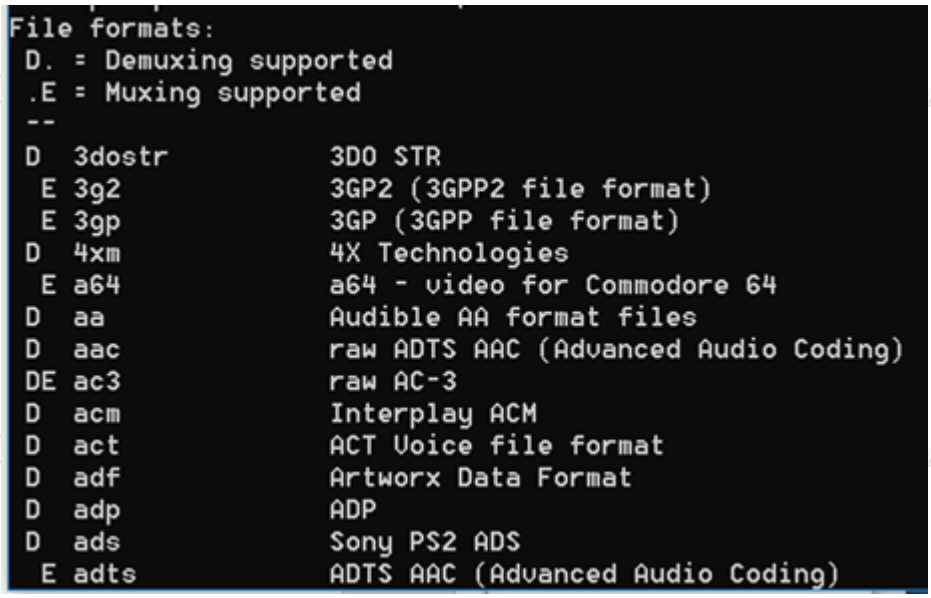
ffmpeg –version

```
G:\future\ffmepg命令入门\test>ffmpeg -version
ffmpeg version 4.1 Copyright (c) 2000-2018 the FFmpeg developers
built with gcc 8.2.1 (GCC) 20181017
configuration: --disable-static --enable-shared --enable-gpl --enable-version3 --enable-sdl2 --enable
e-gnutls --enable-iconv --enable-libass --enable-libbluray --enable-libfreetype --enable-libmp3lame -
-amrnb --enable-libopencore-amrwb --enable-libopenjpeg --enable-libopus --enable-libshine --enable-li
bsoxr --enable-libtheora --enable-libtwolame --enable-libvpx --enable-libwavpack --enable-libwebp --e
ble-libx265 --enable-libxml2 --enable-libzimg --enable-lzma --enable-zlib --enable-gmp --enable-libvi
orbis --enable-libvo-amrwbenc --enable-libmysofa --enable-libspeex --enable-libxvid --enable-libaom -
able-amf --enable-ffnvcodec --enable-cuvid --enable-d3d11va --enable-nvenc --enable-nvdec --enable-dx
th
libavutil      56. 22.100 / 56. 22.100
libavcodec     58. 35.100 / 58. 35.100
libavformat    58. 20.100 / 58. 20.100
libavdevice    58.  5.100 / 58.  5.100
libavfilter     7. 40.101 /  7. 40.101
libswscale      5.  3.100 /  5.  3.100
libswresample   3.  3.100 /  3.  3.100
libpostproc    55.  3.100 / 55.  3.100
```

ffmpeg –buildconf

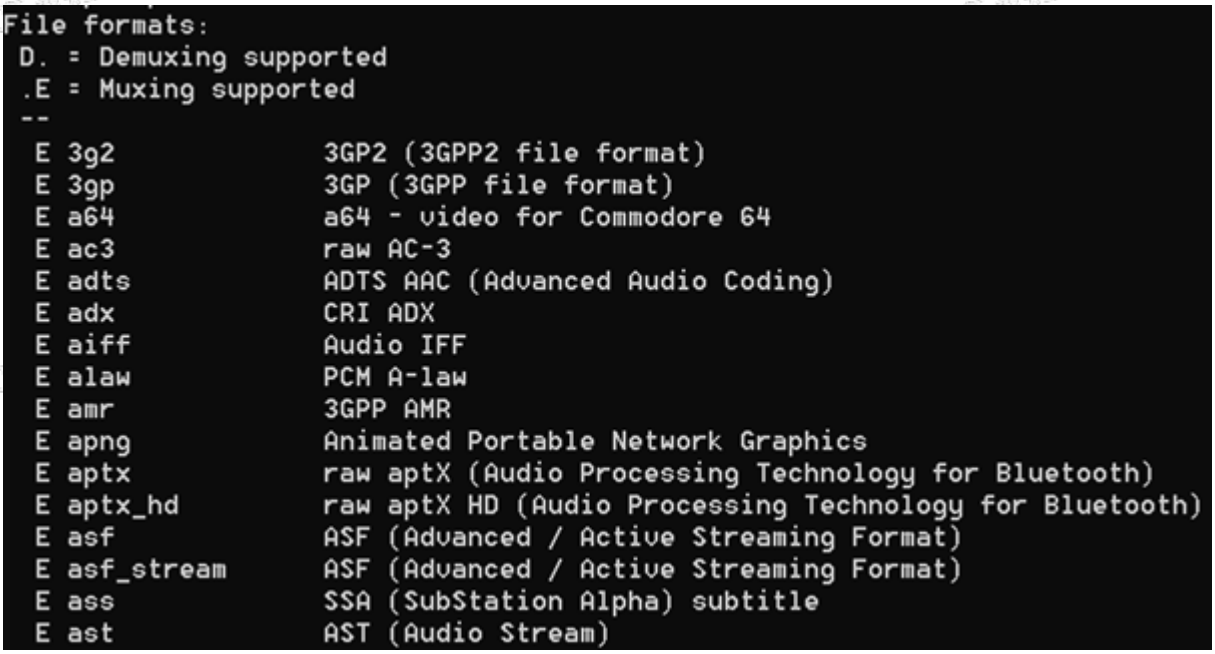
```
G:\future\ffmepg命令入门\test>ffmpeg -buildconf
ffmpeg version 4.1 Copyright (c) 2000-2018 the FFmpeg developers
  built with gcc 8.2.1 (GCC) 20181017
  configuration: --disable-static --enable-shared --enable-gpl --enable-version3 --enable-sdl2 --enab
ble-gnutls --enable-iconv --enable-libass --enable-libbluray --enable-libfreetype --enable-libmp3lame
re-amrnb --enable-libopencore-amrwb --enable-libopenjpeg --enable-libopus --enable-libshine --enable-
libsoxr --enable-libtheora --enable-libtwolame --enable-libvpx --enable-libwavpack --enable-libwebp -
nable-libx265 --enable-libxml2 --enable-libzimg --enable-lzma --enable-zlib --enable-gmp --enable-lib
bvorbis --enable-libvo-amrwbenc --enable-libmysofa --enable-libspeex --enable-libxvid --enable-libaom
enable-amf --enable-ffnvcodec --enable-cuvid --enable-d3d11va --enable-nvenc --enable-nvdec --enable-
ynth
  libavutil      56. 22.100 / 56. 22.100
  libavcodec     58. 35.100 / 58. 35.100
  libavformat    58. 20.100 / 58. 20.100
  libavdevice    58.  5.100 / 58.  5.100
  libavfilter     7. 40.101 /  7. 40.101
  libswscale      5.  3.100 /  5.  3.100
  libswresample   3.  3.100 /  3.  3.100
  libpostproc    55.  3.100 / 55.  3.100

  configuration:
    --disable-static
    --enable-shared
    --enable-gpl
    --enable-version3
    --enable-sdl2
    --enable-fontconfig
    --enable-gnutls
    --enable-iconv
    --enable-libass
    --enable-libbluray
    --enable-libfreetype
    --enable-libmp3lame
    --enable-libopencore-amrnb
    --enable-libopencore-amrwb
    --enable-libopenjpeg
    --enable-libopus
```

ffmpeg –formats



```
File formats:
 D. = Demuxing supported
 .E = Muxing supported
 --
 D  3dostr          3DO STR
  E 3g2             3GP2 (3GPP2 file format)
  E 3gp             3GP (3GPP file format)
 D  4xm             4X Technologies
  E a64             a64 - video for Commodore 64
 D  aa              Audible AA format files
 D  aac             raw ADTS AAC (Advanced Audio Coding)
 DE ac3             raw AC-3
 D  acm             Interplay ACM
 D  act             ACT Voice file format
 D  adf             Artworx Data Format
 D  adp             ADP
 D  ads             Sony PS2 ADS
  E adts            ADTS AAC (Advanced Audio Coding)
```

ffmpeg –muxers



```
File formats:
 D. = Demuxing supported
 .E = Muxing supported
 --
  E 3g2             3GP2 (3GPP2 file format)
  E 3gp             3GP (3GPP file format)
  E a64             a64 - video for Commodore 64
  E ac3             raw AC-3
  E adts            ADTS AAC (Advanced Audio Coding)
  E adx             CRI ADX
  E aiff            Audio IFF
  E alaw            PCM A-law
  E amr             3GPP AMR
  E apng            Animated Portable Network Graphics
  E aptx            raw aptX (Audio Processing Technology for Bluetooth)
  E aptx_hd         raw aptX HD (Audio Processing Technology for Bluetooth)
  E asf             ASF (Advanced / Active Streaming Format)
  E asf_stream      ASF (Advanced / Active Streaming Format)
  E ass             SSA (SubStation Alpha) subtitle
  E ast             AST (Audio Stream)
```

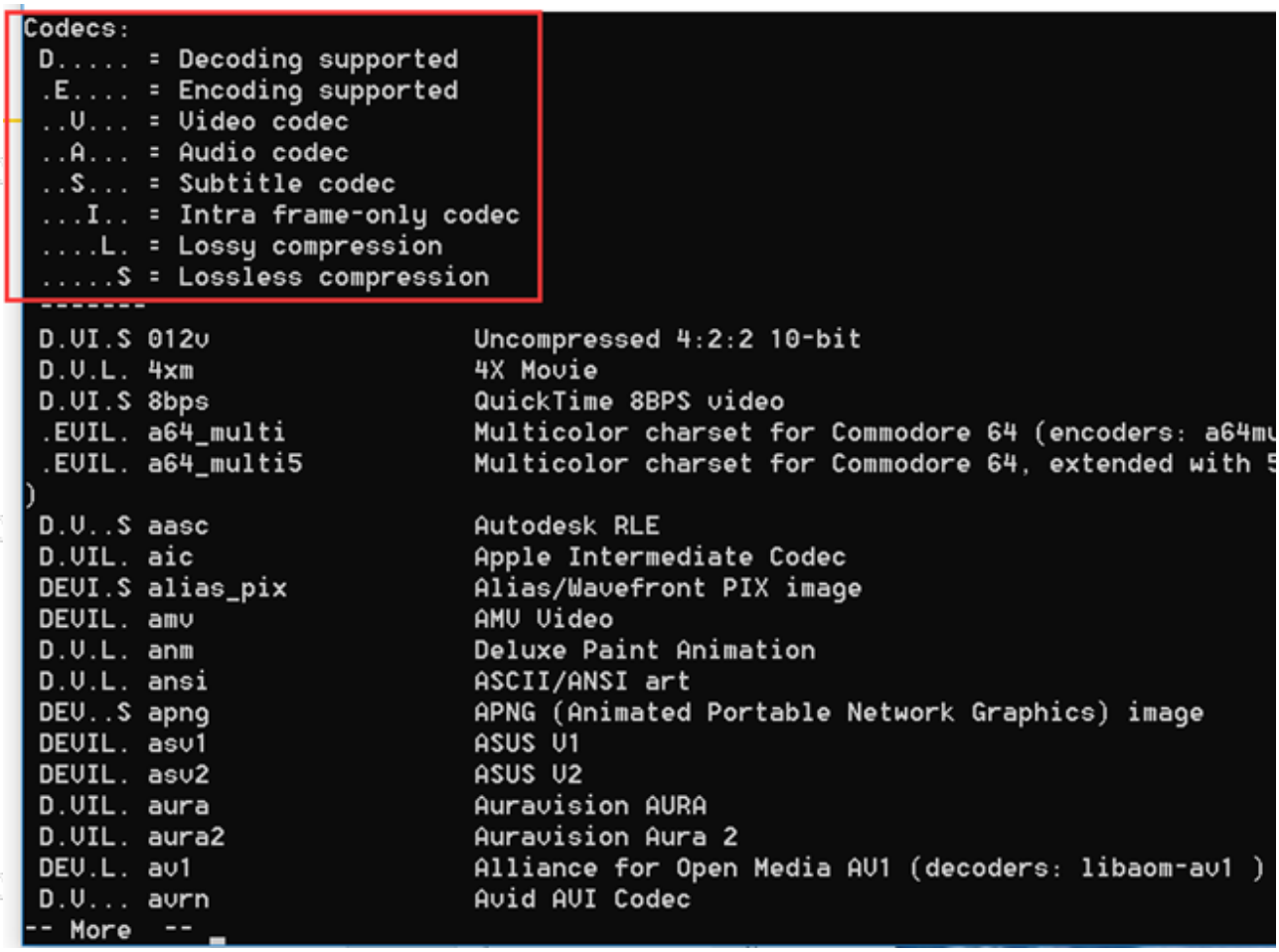
ffmpeg –demuxers

```
File formats:
 D. = Demuxing supported
 .E = Muxing supported
 --
 D  3dostr          3DO STR
 D  4xm             4X Technologies
 D  aa              Audible AA format files
 D  aac             raw ADTS AAC (Advanced Audio Coding)
 D  ac3             raw AC-3
 D  acm             Interplay ACM
 D  act             ACT Voice file format
 D  adf             Artworx Data Format
 D  adp             ADP
 D  ads             Sony PS2 ADS
 D  adx             CRI ADX
 D  aea             MD STUDIO audio
 D  afc             AFC
 D  aiff            Audio IFF
 D  aix             CRI AIX
 D  alaw            PCM A-law
 D  alias_pix       Alias/Wavefront PIX image
 D  amr             3GPP AMR
```

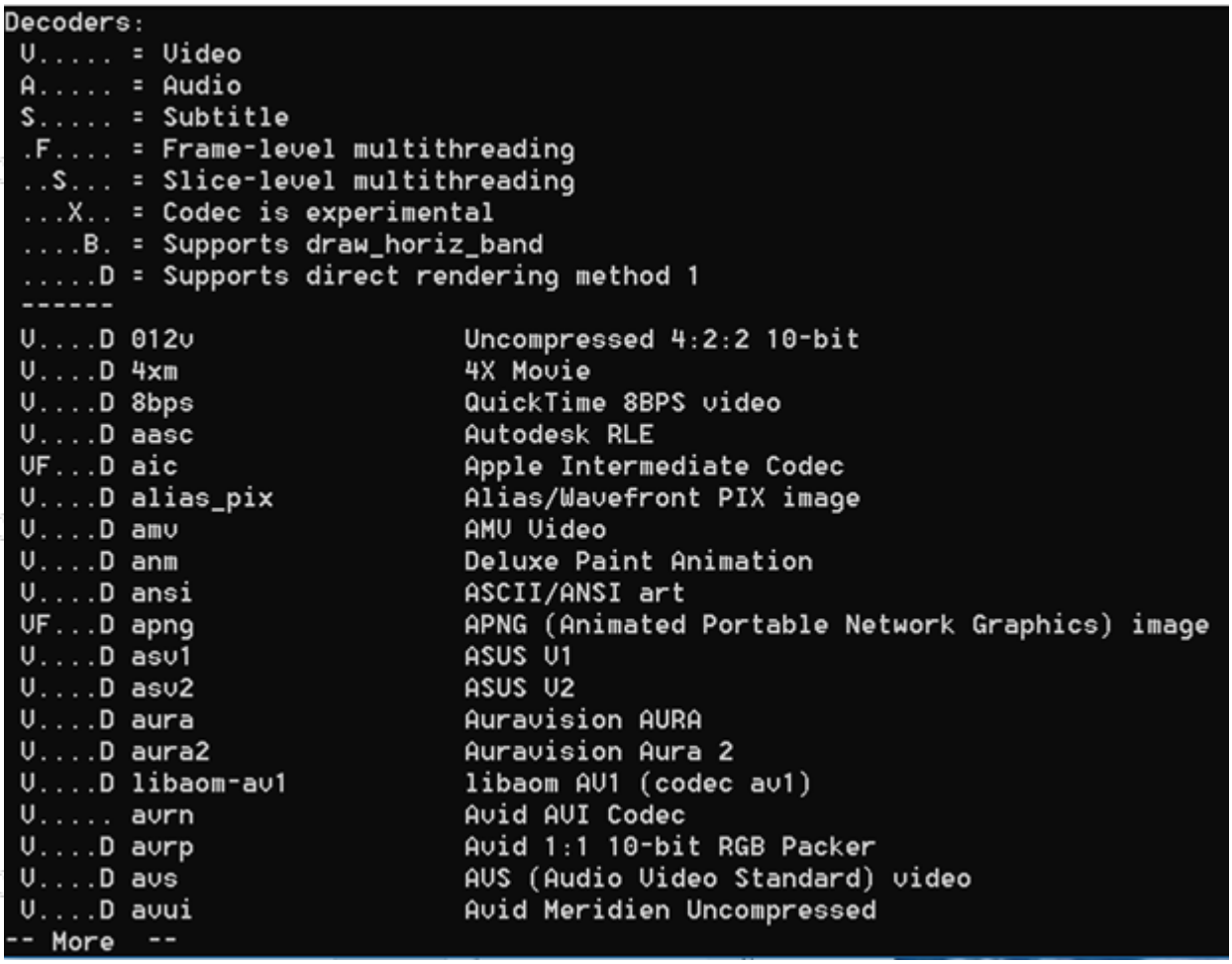
ffmpeg –devices

```
Devices:
 D. = Demuxing supported
 .E = Muxing supported
 --
 D  dshow           DirectShow capture
 D  lavfi           Libavfilter virtual input device
  E sdl,sdl2        SDL2 output device
 D  vfwcap          VfW video capture
```

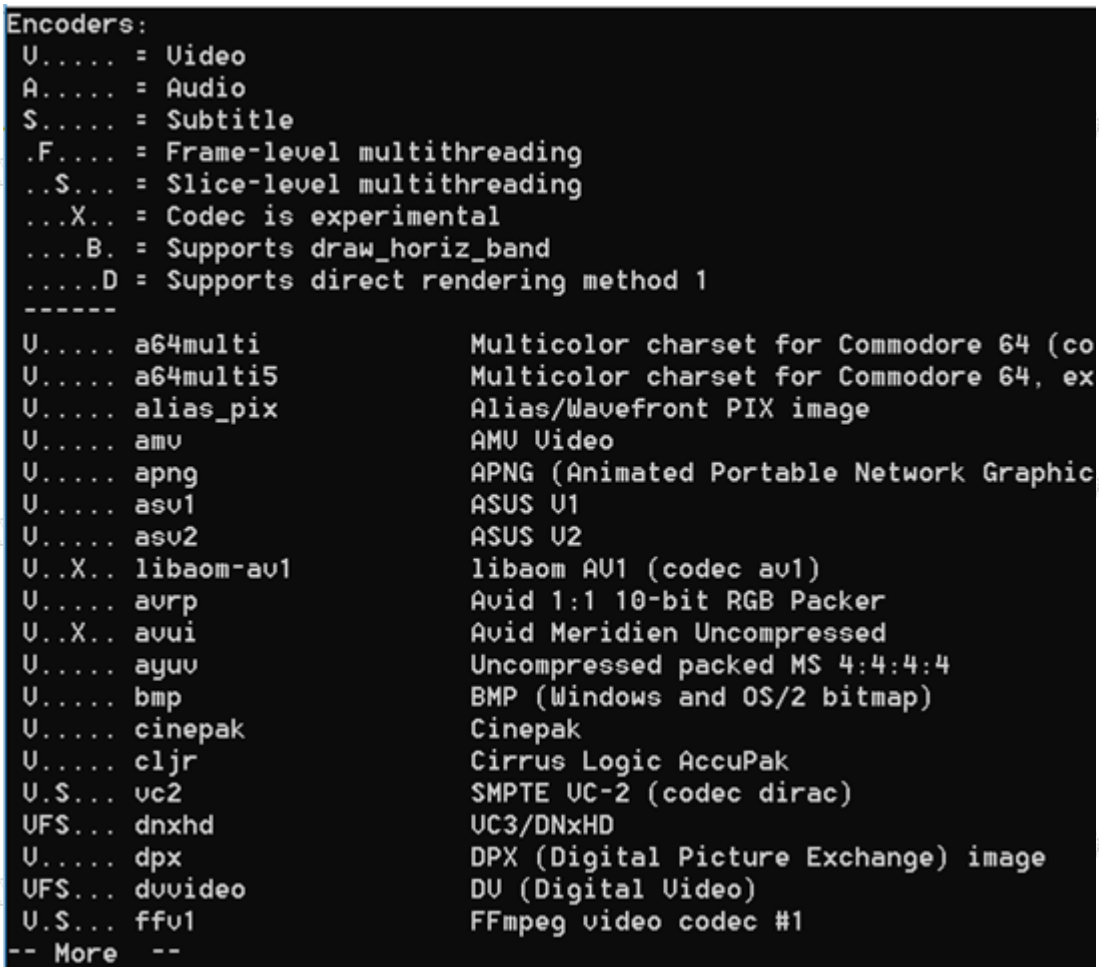
ffmpeg –codecs



ffmpeg –decoders



ffmpeg –encoders

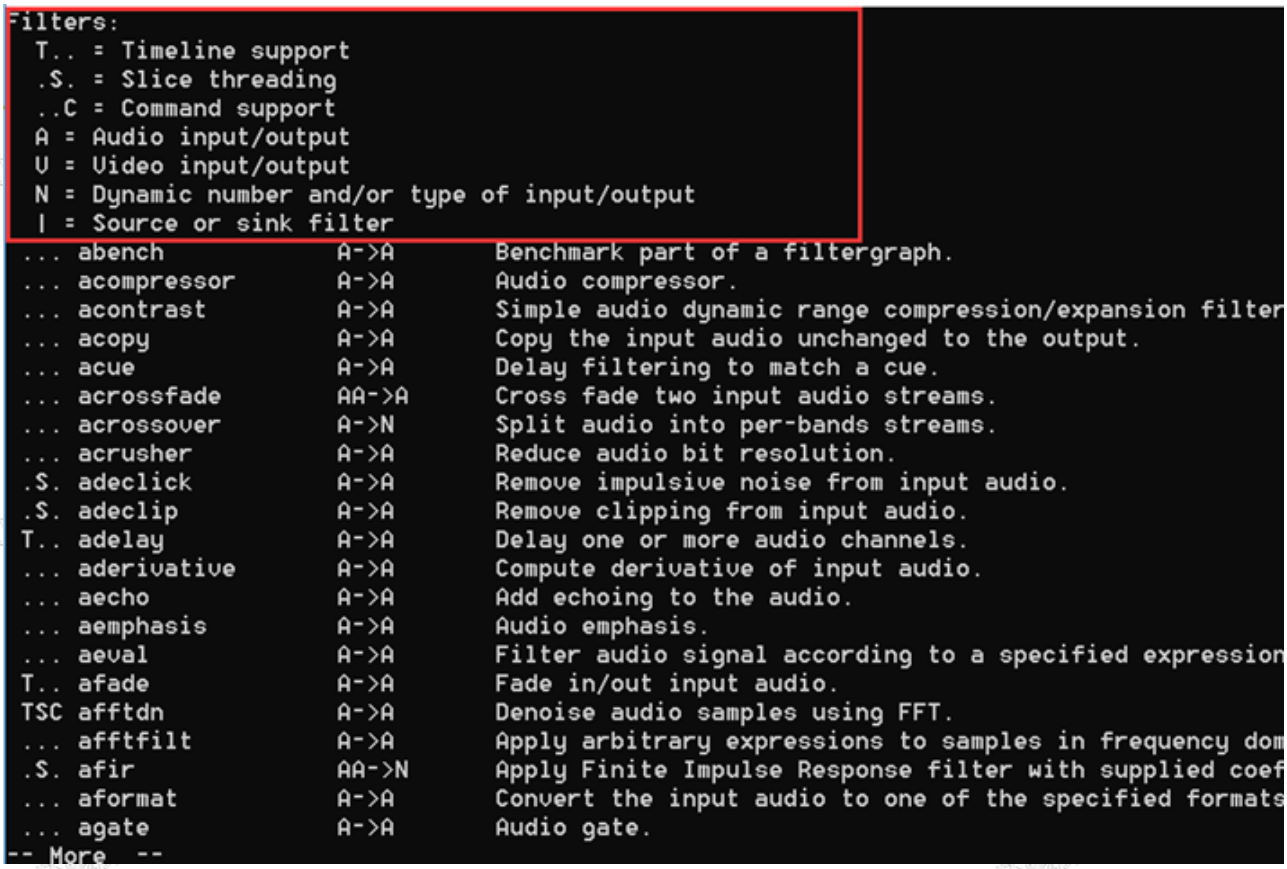

Encoders:
V..... = Video
A..... = Audio
S..... = Subtitle
.F.... = Frame-level multithreading
..S... = Slice-level multithreading
...X.. = Codec is experimental
....B. = Supports draw_horiz_band
.....D = Supports direct rendering method 1

V..... a64multi Multicolor charset for Commodore 64 (co
V..... a64multi5 Multicolor charset for Commodore 64, ex
V..... alias_pix Alias/Wavefront PIX image
V..... amv AMV Video
V..... apng APNG (Animated Portable Network Graphic
V..... asv1 ASUS V1
V..... asv2 ASUS V2
V..X.. libaom-av1 libaom AV1 (codec av1)
V..... avrp Avid 1:1 10-bit RGB Packer
V..X.. avui Avid Meridien Uncompressed
V..... ayuv Uncompressed packed MS 4:4:4:4
V..... bmp BMP (Windows and OS/2 bitmap)
V..... cinepak Cinepak
V..... cljr Cirrus Logic AccuPak
V.S... vc2 SMPTE VC-2 (codec dirac)
VFS... dnxhd VC3/DNxHD
V..... dpx DPX (Digital Picture Exchange) image
VFS... dvvideo DV (Digital Video)
V.S... ffv1 FFmpeg video codec #1
-- More --

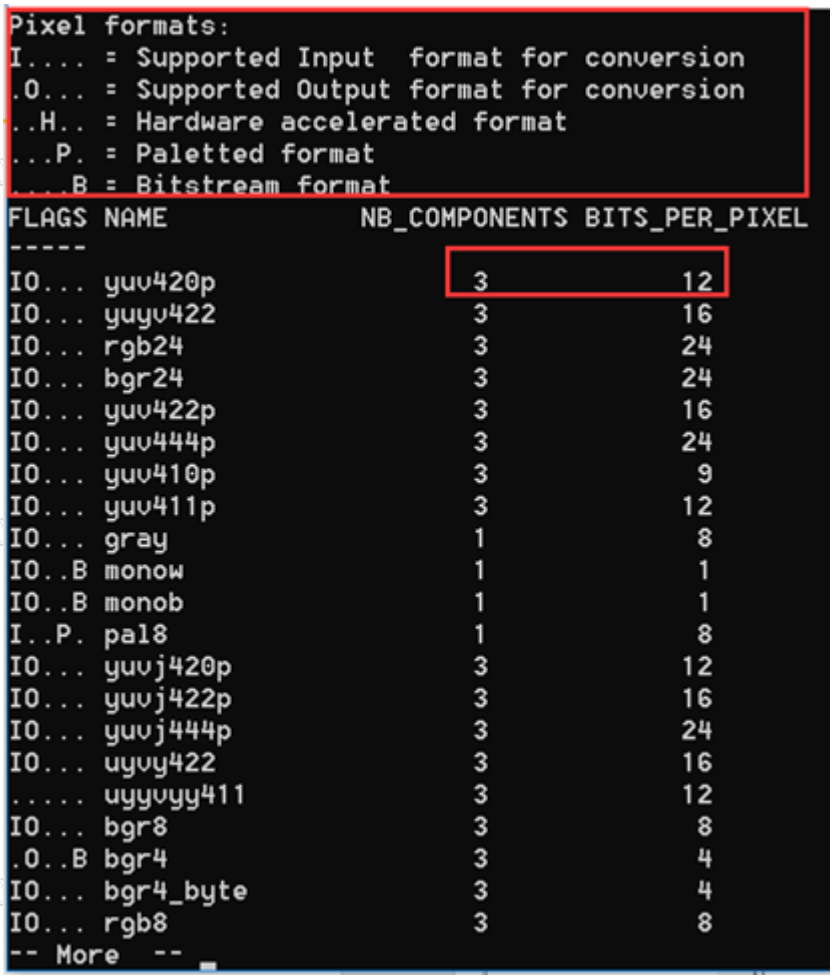
ffmpeg –bsfs

ffmpeg –protocols

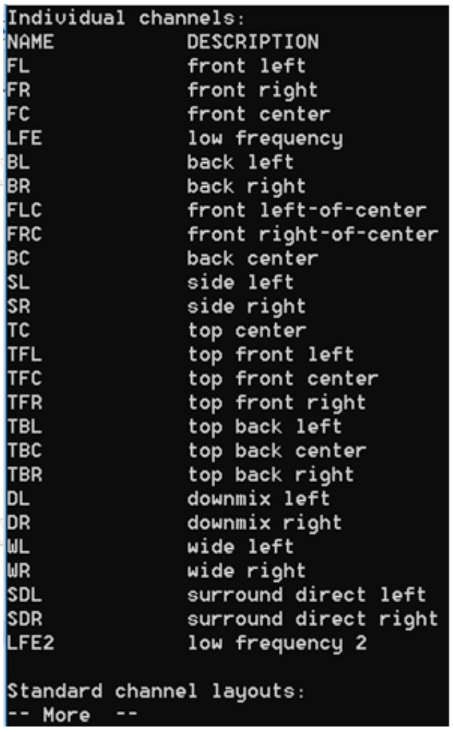
ffmpeg –filters



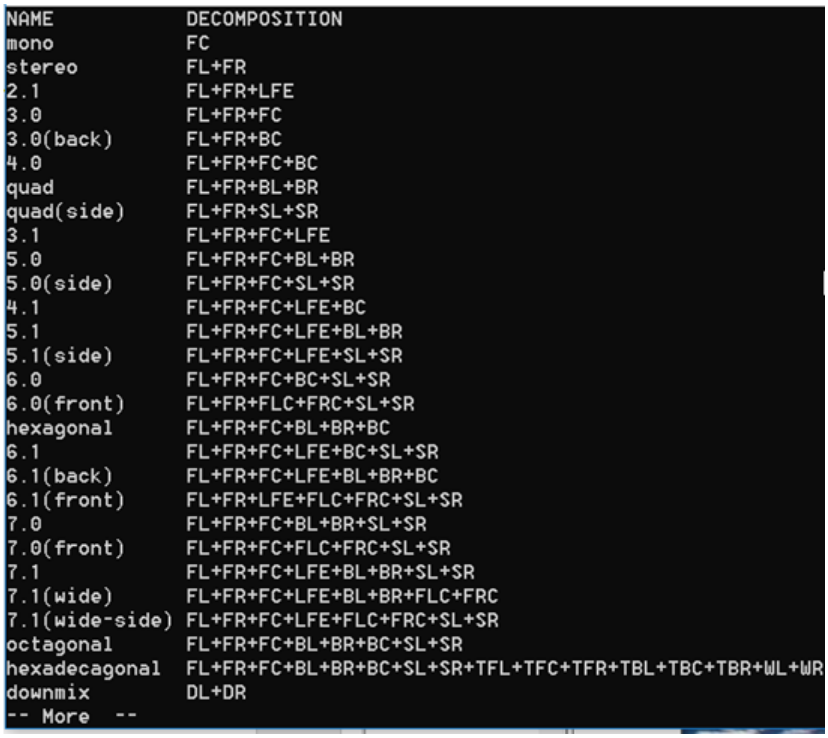
ffmpeg –pix_fmts



ffmpeg –layouts



```
Individual channels:
NAME           DESCRIPTION
FL             front left
FR             front right
FC             front center
LFE            low frequency
BL             back left
BR             back right
FLC            front left-of-center
FRC            front right-of-center
BC             back center
SL             side left
SR             side right
TC             top center
TFL            top front left
TFC            top front center
TFR            top front right
TBL            top back left
TBC            top back center
TBR            top back right
DL             downmix left
DR             downmix right
WL             wide left
WR             wide right
SDL            surround direct left
SDR            surround direct right
LFE2           low frequency 2

Standard channel layouts:
-- More  --
```



```
NAME           DECOMPOSITION
mono           FC
stereo         FL+FR
2.1            FL+FR+LFE
3.0            FL+FR+FC
3.0(back)      FL+FR+BC
4.0            FL+FR+FC+BC
quad           FL+FR+BL+BR
quad(side)     FL+FR+SL+SR
3.1            FL+FR+FC+LFE
5.0            FL+FR+FC+BL+BR
5.0(side)      FL+FR+FC+SL+SR
4.1            FL+FR+FC+LFE+BC
5.1            FL+FR+FC+LFE+BL+BR
5.1(side)      FL+FR+FC+LFE+SL+SR
6.0            FL+FR+FC+BC+SL+SR
6.0(front)     FL+FR+FLC+FRC+SL+SR
hexagonal      FL+FR+FC+BL+BR+BC
6.1            FL+FR+FC+LFE+BC+SL+SR
6.1(back)      FL+FR+FC+LFE+BL+BR+BC
6.1(front)     FL+FR+LFE+FLC+FRC+SL+SR
7.0            FL+FR+FC+BL+BR+SL+SR
7.0(front)     FL+FR+FC+FLC+FRC+SL+SR
7.1            FL+FR+FC+LFE+BL+BR+SL+SR
7.1(wide)      FL+FR+FC+LFE+BL+BR+FLC+FRC
7.1(wide-side) FL+FR+FC+LFE+FLC+FRC+SL+SR
octagonal      FL+FR+FC+BL+BR+BC+SL+SR
hexadecagonal  FL+FR+FC+BL+BR+BC+SL+SR+TFL+TFC+TFR+TBL+TBC+TBR+WL+WR
downmix        DL+DR
-- More  --
```

ffmpeg –sample_fmts

```
name   depth
u8         8
s16       16
s32       32
flt       32
dbl       64
u8p        8
s16p      16
s32p      32
fltp      32
dblp      64
s64       64
s64p      64
```

为什么在ffplay播放PCM的时候需要–f f32le这种模式而不是–f flt呢？比如

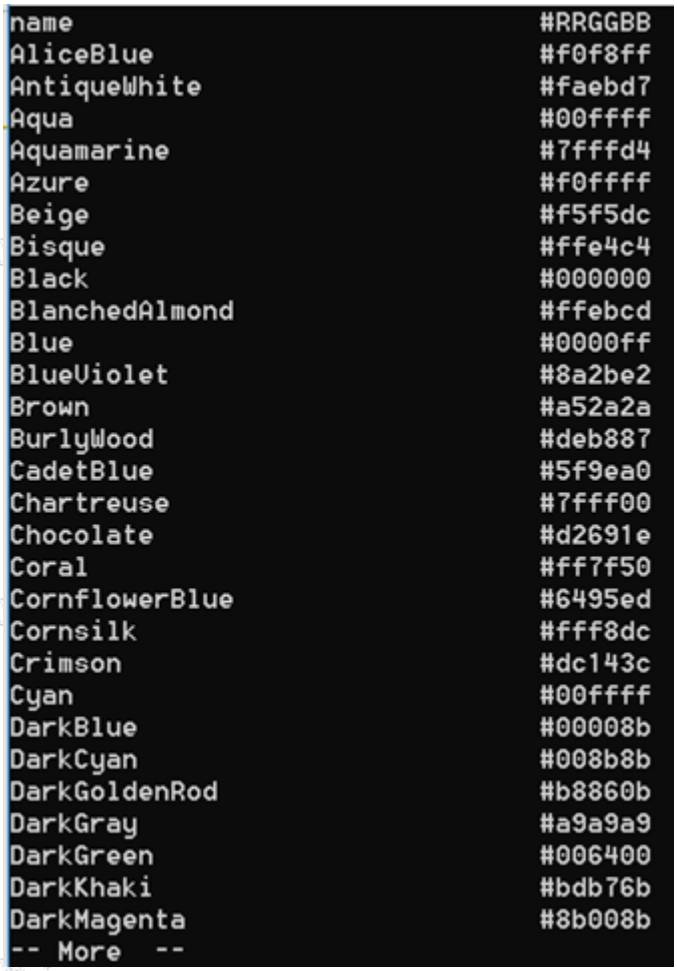
> ffplay –ar 48000 –ac 2 –f f32le believe.pcm

是因为此时实际–f对应的是decoder，这里解PCM则对应pcmdec.c(ffmpeg–4.2.1\libavformat)的PCM解码器，这里的PCM decoder实际只是按照大小端、浮点还是整数、占用的bit数等参数来做解析。比如–f f32le对应为：

> PCMDEF(f32le, ”PCM 32–bit floating–point little–endian”, NULL,
> AV_CODEC_ID_PCM_F32LE)

AV_CODEC_ID_PCM_F32LE属于CODEC ID。

ffmpeg –colors



查看具体分类所支持的参数

语法：ffmepg –h type=name

比如：

- ffmpeg –h muxer=flv
- ffmpeg –h filter=atempo　　(atempo调整音频播放速率)
- ffmpeg –h encoder=libx264

05-ffplay播放控制

选项	说明	选项	说明
q, ESC	退出播放	t	循环切换字幕流
f	全屏切换	c	循环切换节目
p, SPC	暂停	w	循环切换过滤器或显示模式
m	静音切换	s	逐帧播放
9, 0	9减少音量，0增加音量	left/right	向后/向前拖动10秒
~~/, *~~	~~/减少音量，*增加音量~~	down/up	向后/向前拖动1分钟
a	循环切换音频流	鼠标右键单击	拖动与显示宽度对应百分比的文件进行播放
v	循环切换视频流	鼠标左键双击	全屏切换

06-ffplay命令选项

ffplay命令-主要选项

选项	说明	备注
-x width	强制显示宽带。	
-y height	强制显示高度。	
-video_size size	帧尺寸 设置显示帧存储(WxH格式)，仅适用于类似原始YUV等没有包含帧大小(WxH)的视频。 比如: ffplay -pixel_format yuv420p -video_size 320x240 -framerate 5 yuv420p_320x240.yuv	
-pixel_format format	格式设置像素格式	

–fs	以全屏模式启动	
–an	禁用音频（不播放声音）	
–vn	禁用视频（不播放视频）	
–sn	禁用字幕（不显示字幕）	
–ss pos	根据设置的秒进行定位拖动，注意时间单位：比如’55’ 55 seconds, ’12:03:45’ ,12 hours, 03 minutes and 45 seconds, ’23.189’ 23.189 second	
–t duration	设置播放视频/音频长度，时间单位如 –ss选项	
–bytes	按字节进行定位拖动（0=off 1=on –1=auto）	
–seek_interval interval	自定义左/右键定位拖动间隔（以秒为单位），默认值为10秒	
–nodisp	关闭图形化显示窗口，视频将不显示	
–noborder	无边框窗口	
–volume vol	设置起始音量。音量范围[0 ~100]	
–f fmt	强制使用设置的格式进行解析。比如–f s16le	
–window_title title	设置窗口标题（默认为输入文件名）	
–loop number	设置播放循环次数	
–showmode mode	设置显示模式，可用的模式值：0 显示视频，1 显示音频波形，2 显示音频频谱。缺省为0，如果视频不存在则自动选择2	
–vf filtergraph	设置视频滤镜	
–af filtergraph	设置音频滤镜	

ffplay命令–高级选项

选项	说明	备注
–stats	打印多个回放统计信息，包括显示流持续时间，编解码器参数，流中的当前位置，以及音频/视频同步差值。默认情况下处于启用状态，要显式禁用它则需要指定–nostats。。	
–fast	非标准化规范的多媒体兼容优化	
–genpts	生成pts	
–sync type	同步类型 将主时钟设置为audio（type=audio），video（type=video）或external（type=ext），默认是audio为主时钟。	
–ast audio_stream_specifier	指定音频流索引，比如–ast 3，播放流索引为3的音频流	
–vst video_stream_specifier	指定视频流索引，比如–vst 4，播放流索引为4的视频流	
–sst subtitle_stream_specifier	指定字幕流索引，比如–sst 5，播放流索引为5的字幕流	
–autoexit	视频播放完毕后退出。	
–exitonkeydown	键盘按下任何键退出播放	
–exitonmousedown	鼠标按下任何键退出播放	
–codec:media_specifier codec_name	强制使用设置的多媒体解码器，media_specifier可用值为a（音频）， v（视频）和s字幕。比如–codec:v h264_qsv 强制视频采用h264_qsv解码	
–acodec codec_name	强制使用设置的音频解码器进行音频解码	
–vcodec codec_name	强制使用设置的视频解码器进行视频解码	
–scodec codec_name	强制使用设置的字幕解码器进行字幕解码	
–autorotate	根据文件元数据自动旋转视频。值为0或1 ，默认为1	
–framedrop	如果视频不同步则丢弃视频帧。当主时钟非视频时钟时默认开启。若需禁用则使用 –noframedrop	
–infbuf	不限制输入缓冲区大小。尽可能快地从输入中读取尽可能多的数据。播放实时流时默认启用，如果未及时读取数据，则可能会丢弃数据。此选项将不限制缓冲区的大小。若需禁用则使用–noinfbuf	

更多参考

http://www.ffmpeg.org/ffplay.html

07-ffplay命令播放媒体

播放本地文件

ffplay -window_title "test time" -ss 2 -t 10 -autoexit test.mp4

ffplay buweishui.mp3

播放网络流

ffplay -window_title "rtmp stream" rtmp://202.69.69.180:443/webcast/bshdlive-pc

强制解码器

mpeg4解码器：ffplay -vcodec mpeg4 test.mp4

h264解码器：ffplay -vcodec h264 test.mp4

禁用音频或视频

禁用音频：ffplay test.mp4 -an

禁用视频：ffplay test.mp4 -vn

播放YUV数据

ffplay -pixel_format yuv420p -video_size 320x240 -framerate 5 yuv420p_320x240.yuv

播放RGB数据

ffplay -pixel_format rgb24 -video_size 320x240 -i rgb24_320x240.rgb

ffplay -pixel_format rgb24 -video_size 320x240 -framerate 5 -i rgb24_320x240.rgb

播放PCM数据

ffplay –ar 48000 –ac 2 –f f32le 48000_2_f32le.pcm

–ar　　set audio sampling rate (in Hz) (from 0 to INT_MAX) (default 0)

–ac　　set number of audio channels (from 0 to INT_MAX) (default 0)

08–ffplay简单过滤器

视频旋转

ffplay –i test.mp4 –vf transpose=1

视频反转

ffplay test.mp4 –vf hflip

ffplay test.mp4 –vf vflip

视频旋转和反转

ffplay test.mp4 –vf hflip,transpose=1

音频变速播放

ffplay –i test.mp4 –af atempo=2

视频变速播放

ffplay –i test.mp4 –vf setpts=PTS/2

音视频同时变速

ffplay –i test.mp4 –vf setpts=PTS/2 –af atempo=2

更多参考

http://www.ffmpeg.org/ffmpeg–filters.html

09–ffmpeg命令参数说明

主要参数

- –i 设定输入流
- –f 设定输出格式(format)
- –ss 开始时间
- –t 时间长度

音频参数

- –aframes 设置要输出的音频帧数
- –b:a 音频码率
- –ar 设定采样率
- –ac 设定声音的Channel数
- –acodec 设定声音编解码器，如果用copy表示原始编解码数据必须被拷贝。
- –an 不处理音频
- –af 音频过滤器

ffmpeg –i test.mp4 –b:a 192k –ar 48000 –ac 2 –acodec libmp3lame –aframes 200 out2.mp3

视频参数

- –vframes 设置要输出的视频帧数
- –b 设定视频码率
- –b:v 视频码率
- –r 设定帧速率
- –s 设定画面的宽与高
- –vn 不处理视频
- –aspect aspect 设置横纵比 4:3 16:9 或 1.3333 1.7777
- –vcodec 设定视频编解码器，如果用copy表示原始编解码数据必须被拷贝。
- –vf 视频过滤器

ffmpeg –i test.mp4 –vframes 300 –b:v 300k –r 30 –s 640x480 –aspect 16:9 –vcodec libx265

10–ffmpeg命令提取音视频数据

保留封装格式

ffmpeg –i test.mp4 –acodec copy –vn audio.mp4

ffmpeg –i test.mp4 –vcodec copy –an video.mp4

提取视频

保留编码格式：ffmpeg –i test.mp4 –vcodec copy –an test_copy.h264

强制格式：ffmpeg –i test.mp4 –vcodec libx264 –an test.h264

提取音频

保留编码格式：ffmpeg –i test.mp4 –acodec copy –vn test.aac

强制格式：ffmpeg –i test.mp4 –acodec libmp3lame –vn test.mp3

11 提取像素格式和PCM数据

提取像素格式

提取YUV

提取3秒数据，分辨率和源视频一致

ffmpeg –i test_1280x720.mp4 –t 3 –pix_fmt yuv420p yuv420p_orig.yuv

提取3秒数据，分辨率转为320x240

ffmpeg –i test_1280x720.mp4 –t 3 –pix_fmt yuv420p –s 320x240 yuv420p_320x240.yuv

提取RGB

提取3秒数据，分辨率转为320x240

ffmpeg –i test.mp4 –t 3 –pix_fmt rgb24 –s 320x240 rgb24_320x240.rgb

RGB和YUV之间的转换

ffmpeg –s 320x240 –pix_fmt yuv420p –i yuv420p_320x240.yuv –pix_fmt rgb24 rgb24_320x240_2.rgb

提取PCM数据

ffmpeg –i buweishui.mp3 –ar 48000 –ac 2 –f s16le 48000_2_s16le.pcm

ffmpeg –i buweishui.mp3 –ar 48000 –ac 2 –sample_fmt s16 out_s16.wav

ffmpeg –i buweishui.mp3 –ar 48000 –ac 2 –codec:a pcm_s16le out2_s16le.wav

ffmpeg –i buweishui.mp3 –ar 48000 –ac 2 –f f32le 48000_2_f32le.pcm

ffmpeg –i test.mp4 –t 10 –vn –ar 48000 –ac 2 –f f32le 48000_2_f32le_2.pcm

12–ffmpeg命令转封装

保持编码格式

ffmpeg –i test.mp4 –vcodec copy –acodec copy test_copy.ts

ffmpeg –i test.mp4 –codec copy test_copy2.ts

改变编码格式

ffmpeg –i test.mp4 –vcodec libx265 –acodec libmp3lame out_h265_mp3.mkv

修改帧率

ffmpeg –i test.mp4 –r 15 –codec copy output.mp4 (错误命令)

ffmpeg –i test.mp4 –r 15 output2.mp4

修改视频码率

ffmpeg –i test.mp4 –b 400k output_b.mkv （此时音频也被重新编码）

修改视频码率

ffmpeg –i test.mp4 –b:v 400k output_bv.mkv

修改音频码率

ffmpeg –i test.mp4 –b:a 192k output_ba.mp4

如果不想重新编码video，需要加上–vcodec copy

修改音视频码率

ffmpeg –i test.mp4 –b:v 400k –b:a 192k output_bva.mp4

修改视频分辨率

ffmpeg –i test.mp4 –s 480x270 output_480x270.mp4

修改音频采样率:

ffmpeg –i test.mp4 –ar 44100 output_44100hz.mp4

13–ffmpeg命令过滤器

生成测试文件

找三个不同的视频每个视频截取10秒内容

- ffmpeg –i 沙海02.mp4 –ss 00:05:00 –t 10 –codec copy 1.mp4
- ffmpeg –i 复仇者联盟3.mp4 –ss 00:05:00 –t 10 –codec copy 2.mp4
- ffmpeg –i 红海行动.mp4 –ss 00:05:00 –t 10 –codec copy 3.mp4
  如果音视频格式不统一则强制统一为 –vcodec libx264 –acodec aac

将上述1.mp4/2.mp4/3.mp4转成ts格式

- ffmpeg –i 1.mp4 –codec copy –vbsf h264_mp4toannexb 1.ts
- ffmpeg –i 2.mp4 –codec copy –vbsf h264_mp4toannexb 2.ts
- ffmpeg –i 3.mp4 –codec copy –vbsf h264_mp4toannexb 3.ts

转成flv格式

- ffmpeg –i 1.mp4 –codec copy 1.flv
- ffmpeg –i 2.mp4 –codec copy 2.flv
- ffmpeg –i 3.mp4 –codec copy 3.flv

开始拼接文件

以MP4格式进行拼接

~~方法1：ffmpeg –i ”concat:1.mp4|2.mp4|3.mp4” –codec copy out_mp4.mp4~~

方法2：ffmpeg –f concat –i mp4list.txt–codec copy out_mp42.mp4

以TS格式进行拼接

方法1：ffmpeg –i ”concat:1.ts|2.ts|3.ts” –codec copy out_ts.mp4

方法2：ffmpeg –f concat –i tslist.txt –codec copy out_ts2.mp4

以FLV格式进行拼接

~~方法1：ffmpeg –i ”concat:1.flv|2.flv|3.flv” –codec copy out_flv.mp4~~

方法2：ffmpeg –f concat –i flvlist.txt –codec copy out_flv2.mp4

方法1只适用部分封装格式，比如TS

建议：

（1）使用方法2进行拼接

（2）转成TS格式再进行拼接

测试不同编码拼接

修改音频编码

- ffmpeg –i 2.mp4 –vcodeccopy –acodec ac3 –vbsfh264_mp4toannexb 2.ts
- ffmpeg –i ”concat:1.ts|2.ts|3.ts” –codec copy out1.mp4 结果第二段没有声音

修改音频采样率

- ffmpeg –i 2.mp4 –vcodec copy –acodec aac –ar 96000 –vbsf h264_mp4toannexb 2.ts
- ffmpeg –i ”concat:1.ts|2.ts|3.ts” –codec copy out2.mp4 第二段播放异常

修改视频编码格式

- ffmpeg –i 1.mp4 –acodec copy –vcodec libx265 1.ts
- ffmpeg –i ”concat:1.ts|2.ts|3.ts” –codec copy out3.mp4

修改视频分辨率

- ffmpeg –i 1.mp4 –acodec copy –vcodec libx264 –s 800x472 –vbsfh264_mp4toannexb 1.ts
- ffmpeg –i ”concat:1.ts|2.ts|3.ts” –codec copy out4.**mp4**

注意：

- 把每个视频封装格式也统一为**ts**，拼接输出的时候再输出你需要的封装格式，比如MP4
- **视频分辨率可以不同，但是编码格式需要统一**
- **音频编码格式需要统一，音频参数(采样率/声道等)也需要统一**

14–fmpeg命令图片与视频互转

截取一张图片

- ffmpeg –i test.mp4 –y –f image2 –ss 00:00:02 –vframes 1 –s 640x360 test.jpg
- ffmpeg –i test.mp4 –y –f image2 –ss 00:00:02 –vframes 1 –s 640x360 test.bmp

参数说明：

- –i 输入
- –y 覆盖
- –f 格式 image2 一种格式
- –ss 起始值

- –vframes 帧 如果大于1 那么 输出加%03d test%03d.jpg
- –s 格式大小size

转换视频为图片（每帧一张图):

- ffmpeg –i test.mp4 –t 5 –s 640x360 –r 15frame%03d.jpg

图片转换为视频:

- ffmpeg –f image2 –i frame%03d.jpg–r 25 video.mp4

从视频中生成GIF图片

- ffmpeg –i test.mp4 –t 5 –r1 image1.gif
- ffmpeg –i test.mp4 –t 5 –r 25 –s 640x360 image2.gif

将 GIF 转化为 视频

- ffmpeg –f gif –i image2.gif image2.mp4

15–ffmpeg命令视频录制

ffmpeg命令视频录制(Windows)

先安装dshow软件Screen Capturer Recorder

- 项目地址：https://sourceforge.net/projects/screencapturer/files/
- 然后查看可用设备名字：ffmpeg –list_devices true –f dshow –i dummy

[dshow @ 0509d6c0] **DirectShow video devices** (some may be both video and audio devices)

[dshow @ 0509d6c0] ”Integrated Webcam” //笔记本摄像头

[dshow @ 0509d6c0] Alternative name ”@device_pnp_\\?
\usb#vid_0bda&pid_5689&mi_00#6&233dd6c7&0&0000#{65e8773d–8f56–11

d0–a3b9–00a0c9223196}\global”

[dshow @ 0509d6c0] ”e2eSoft VCam”

[dshow @ 0509d6c0] Alternative name ”@device_sw_{860BB310–5D01–11D0–BD3B–
00A0C911CE86}\e2eSoft VCam”

[dshow @ 0509d6c0] **”screen–capture–recorder”**

[dshow @ 0509d6c0] Alternative name ”@device_sw_{860BB310–5D01–11D0–BD3B–
00A0C911CE86}\{4EA6930A–2C8A–4AE6–A561–56E4

B5044439}”

[dshow @ 0509d6c0] **DirectShow audio devices**

[dshow @ 0509d6c0] **"楹﹀厠椋?(Realtek Audio)"**

[dshow @ 0509d6c0] Alternative name "@device_cm_{33D9A762-90C8-11D0-BD43-00A0C911CE86}\wave_{8B8892E5-D3E5-47EC-8B5E

-CEEBF54014E7}"

[dshow @ 0509d6c0] **"virtual-audio-capturer"**

[dshow @ 0509d6c0] Alternative name "@device_sw_{33D9A762-90C8-11D0-BD43-00A0C911CE86}\{8E14549B-DB61-4309-AFA1-3578

E927E935}"

dummy: Immediate exit requested

如果出现乱码的时候在命令行输入：chcp 65001

chcp 就是change code page
65001是unicode字符集 支持中文

音视频录制

录制视频（默认参数）

- **桌面**：ffmpeg -f dshow -i video="screen-capture-recorder" v-out.mp4
- **摄像头**： ffmpeg -f dshow -i video="Integrated Webcam" -y v-out2.flv (**要根据自己摄像头名称**)

录制声音（默认参数）

- **系统声音：**ffmpeg -f dshow -i audio="virtual-audio-capturer" a-out.aac
- **系统+麦克风声音：**ffmpeg -f dshow -i audio="麦克风 (Realtek Audio)"-f dshow -i audio="virtual-audio-capturer" -filter_complex amix=inputs=2:duration=first:dropout_transition=2 a-out2.aac

同时录制声音和视频（默认参数）

- ffmpeg –f dshow –i audio=”麦克风(Realtek Audio)” –f dshow –i audio=”virtual–audio–capturer” –filter_complex amix=inputs=2:duration=first:dropout_transition=2 –f dshow –i video=”screen–capture–recorder” –y av–out.flv

查看视频录制的可选参数

查看视频录制的可选参数

- ffmpeg –f dshow –list_options true –i video=”screen–capture–recorder”

> [dshow @ 02f0d6c0] DirectShow video device options (from video devices)
>
> [dshow @ 02f0d6c0] Pin ”Capture” (alternative pin name ”1”)
>
> [dshow @ 02f0d6c0] pixel_format=bgr0 min s=1x1 fps=0.02 max s=1920x1080 fps=30
>
> [dshow @ 02f0d6c0] pixel_format=bgr0 min s=1x1 fps=0.02 max s=1920x1080 fps=30
>
> [dshow @ 02f0d6c0] pixel_format=bgr24 min s=1x1 fps=0.02 max s=1920x1080 fps=30
>
> [dshow @ 02f0d6c0] pixel_format=rgb555le min s=1x1 fps=0.02 max s=1920x1080 fps=30
>
> [dshow @ 02f0d6c0] pixel_format=rgb555le min s=1x1 fps=0.02 max s=1920x1080 fps=30
>
> [dshow @ 02f0d6c0] pixel_format=rgb8 min s=1x1 fps=0.02 max s=1920x1080 fps=30
>
> [dshow @ 02f0d6c0] pixel_format=yuv420p min s=1x1 fps=0.02 max s=1920x1080 fps=30

查看音频录制的可选参数

- ffmpeg –f dshow–list_optionstrue –i audio=”virtual–audio–capturer”

> [dshow @ 05a2d6c0] DirectShow audio only device options (from audio devices)
>
> [dshow @ 05a2d6c0] Pin ”Capture Virtual Audio Pin” (alternative pin name ”1”)
>
> [dshow @ 05a2d6c0] min ch=2 bits=16 rate= 48000 max ch=2 bits=16 rate= 48000

- ffmpeg –f dshow –list_options true –i audio=”麦克风 (Realtek Audio)”

指定参数录制音视频

- ffmpeg –f dshow –i audio=”麦克风(Realtek Audio)” –f dshow –i audio=”virtual–audio–capturer” –filter_complex amix=inputs=2:duration=first:dropout_transition=2 **–fdshow** –video_size1920x1080 –framerate 15 –pixel_format yuv420p –i video=”screen–capture–recorder” –vcodec h264_qsv –b:v 3M –y av–out.flv
- ffmpeg –fdshow –i audio=”麦克风(Realtek Audio)” –f dshow –i audio=”virtual–audio–capturer” –filter_complex amix=inputs=2:duration=first:dropout_transition=2 –f dshow –i video=”screen–capture–recorder” –vcodec h264_qsv –b:v 3M–r 15 –y av–out2.mp4
- ffmpeg –fdshow –i audio=”麦克风(Realtek Audio)” –f dshow –i audio=”virtual–audio–capturer” –filter_complex amix=inputs=2:duration=first:dropout_transition=2 –f dshow–framerate 15 –pixel_format yuv420p –i video=”screen–capture–recorder” –vcodec h264_qsv –b:v 3M–r 15 –y av–out3.mp4

桌面+摄像头+麦克风

ffmpeg –f dshow –framerate 15 –i video=”screen–capture–recorder” –f dshow –framerate 10 –i video=”Integrated Webcam” –filter_complex ”[1]scale=iw/2:ih/2[pip];[0][pip]overlay=main_w–overlay_w–10:main_h–overlay_h–10” –f dshow –i audio=”麦克风 (Realtek Audio)” –c:a aac –c:v h264_qsv –r 10 –b 3M –f flv rtmp://111.229.231.225/live/33

rtp推流

先用ffplay检测摄像头是否正常，比如：

ffplay –f dshow –i video=”USB2.0 PC CAMERA”

推流：ffmpeg –f dshow –i video=”USB2.0 PC CAMERA” –vcodec libx264 –f rtp rtp://192.168.2.208:6970 > test.sdp

拉流：ffplay –protocol_whitelist ”file,udp,rtp” –i test.sdp

16–ffmpeg命令直播

直播拉流

- ffplay rtmp://server/live/streamName
- ffmpeg –i rtmp://server/live/streamName –c copy dump.flv

对于不是rtmp的协议 –c copy要谨慎使用

可用地址

- HKS：rtmp://live.hkstv.hk.lxdns.com/live/hks2
- 大熊兔(点播)：rtsp://184.72.239.149/vod/mp4://BigBuckBunny_175k.mov
- CCTV1高清：http://ivi.bupt.edu.cn/hls/cctv1hd.m3u8
    - ffmpeg –i http://ivi.bupt.edu.cn/hls/cctv1hd.m3u8 –c copy cctv1.**ts**
    - ffmpeg –i http://ivi.bupt.edu.cn/hls/cctv1hd.m3u8 cctv1.flv
    - ffmpeg –i http://ivi.bupt.edu.cn/hls/cctv1hd.m3u8 –acodec aac –vcodec libx264 cctv1–2.flv
- CCTV3高清：http://ivi.bupt.edu.cn/hls/cctv3hd.m3u8
- CCTV5高清：http://ivi.bupt.edu.cn/hls/cctv5hd.m3u8
- CCTV5+高清：http://ivi.bupt.edu.cn/hls/cctv5phd.m3u8
- CCTV6高清：http://ivi.bupt.edu.cn/hls/cctv6hd.m3u8

ffmpeg推流

直播推流

- ffmpeg –re –i out.mp4 –c copy flvrtmp://server/live/streamName
  参数：–re,表示按时间戳读取文件

参考：Nginx搭建rtmp流媒体服务器(Ubuntu 16.04)

https://www.jianshu.com/p/16741e363a77

17 FFmpeg滤镜

17.1 filter的分类

按照处理数据的类型，通常多媒体的filter分为：

- 音频filter
- 视频filter
- 字幕filter

另一种按照处于编解码器的位置划分：

- prefilters: used before encoding
- intrafilters: used while encoding (and are thus an integral part of a video codec)
- postfilters: used after decoding

FFmpeg中filter分为：

- source filter （只有输出）
- audio filter
- video filter
- Multimedia filter
- sink filter （只有输入）

除了source和sink filter，其他filter都至少有一个输入、至少一个输出。

17.2 视频裁剪

视频过滤器（滤镜）:裁剪

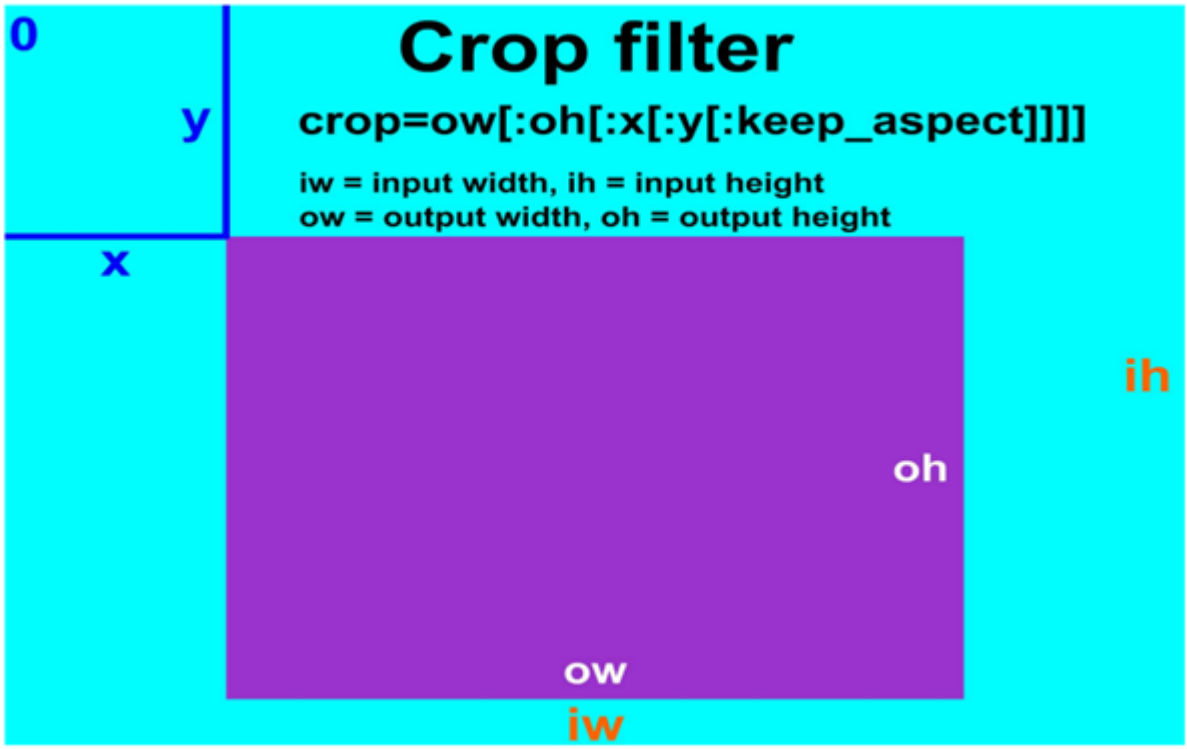


图 1–1 裁剪图

描述	将输入视频帧的宽度和高度从x和y值表示的位置裁剪到指定的宽度和高度; x和y是输出的左上角坐标，协调系统的中心是输入视频帧的左上角。 如果使用了可选的keep_aspect参数，将会改变输出SAR(样本宽比)以补偿新的DAR(显示长宽比)
语法	crop=ow[:oh[:x[:y[:keep_aspect]]]]

变量	**用于ow和oh参数的表达式中的可用变量**
x, y	对x的计算值(从左上角水平方向的像素个数)和y(垂直像素的数量)，对每个帧进行评估，x的默认值为(iw – ow)/2, y的默认值为(ih – oh)/2
in_w, iw	输入的宽度
in_h, ih	输入的高度
out_w, ow	输出(裁剪)宽度，默认值= iw
out_h, oh	输出(裁剪)高度，默认值= ih

a	**纵横比，与iw/ih相同**
sar	输入样本比例
dar	输入显示宽比，等于表达式a*sar
hsub, vsub	水平和垂直的色度子样本值，对于像素格式yuv422p, hsub的值为2,vsub为1
n	输入帧的数目，从0开始
pos	位置在输入框的文件中，如果不知道NAN
t	时间戳以秒表示，如果输入时间戳未知

ow的值可以从oh得到，反之亦然，但不能从x和y中得到，因为这些值是在ow和oh之后进行的。x的值可以从y的值中得到，反之亦然。例如，在输入框的左三、中三和右三，我们可以使用命令:

```
ffmpeg –i input –vf crop=iw/3:ih:0:0 output

ffmpeg –i input –vf crop=iw/3:ih:iw/3:0 output

ffmpeg –i input –vf crop=iw/3:ih:iw/3*2:0 output
```

练习题:

（1）裁剪100x100的区域，起点为(12,34).

crop=100:100:12:34

相同效果:

crop=w=100:h=100:x=12:y=34

（2）裁剪中心区域，大小为100x100

crop=100:100

（3）裁剪中心区域，大小为输入视频的2/3

crop=2/3*in_w:2/3*in_h

（4）裁剪中心区域的正方形，高度为输入视频的高

crop=out_w=in_h

crop=in_h

（5）裁剪偏移左上角100像素

crop=in_w−100:in_h−100:100:100

（6）裁剪掉左右10像素，上下20像素

crop=in_w−2*10:in_h−2*20

（7）裁剪右下角区域

crop=in_w/2:in_h/2:in_w/2:in_h/2

17.3 FFmpeg滤镜Filter内置变量

在使用Filter时，经常会用到根据时间轴进行操作的需求，在使用FFmpeg的Filter时可以使用Filter的时间相关的内置变量，下面先来了解一下这些相关的变量，见下表。

表17−3 FFmpeg滤镜filter基本内置变量

变量	说明
t	以秒表示的时间戳，如果输入的时间是未知的则是NAN
n	输入帧的顺序编号，从0开始
pos	输入帧的位置，如果未知的则是NAN
w	输入视频帧的宽度
h	输入视频帧的高度

17.4 添加水印

17.4.1 文字水印

在视频中增加文字水印需要准备的条件比较多，需要有文字字库处理的相关文件，在编译FFmpeg时需要支持FreeType、FontConfig、iconv，系统中需要有相关的字库，在FFmpeg中增加纯字母水印可以使用**drawtext**滤镜进行支持，下面就来看一下**drawtext**的滤镜参数，具体见表1-4。

表1-4 FFmpeg文字滤镜参数

参数	**类型**	**说明**
text	字符串	文字
textfile	字符串	文字文件
box	布尔	文字区域背景框(缺省false)
boxcolor	色彩	展示字体区域块的颜色
font	字符串	字体名称（默认为Sans字体）
fontsize	整数	显示字体的大小
x	字符串	缺省为0
y	字符串	缺省为0
alpha	浮点数	透明度(默认为1)，值从0~1

举例

（1）将文字的水印加在视频的左上角：

ffplay –i input.mp4 –vf ”drawtext=fontsize=100:fontfile=FreeSerif.ttf:text=’hello world’:x=20:y=20”

将**字体的颜色设置为绿色**：

ffplay –i input.mp4 –vf ”drawtext=fontsize=100:fontfile=FreeSerif.ttf:text=’hello world’:fontcolor=green”

如果想调整文字水印显示的位置，调整x与y参数的数值即可。

ffplay –i input.mp4 –vf ”drawtext=fontsize=100:fontfile=FreeSerif.ttf:text=’hello world’:fontcolor=green:x=400:y=200”

修改透明度

ffplay –i input.mp4 –vf ”drawtext=fontsize=100:fontfile=FreeSerif.ttf:text=’hello world’:fontcolor=green:x=400:y=200:alpha=0.5”

（2）文字水印还可以增加一个框，然后给框加上背景颜色：

ffplay –i input.mp4 –vf ”drawtext=fontsize=100:fontfile=FreeSerif.ttf:text=’hello world’:fontcolor=green:box=1:boxcolor=yellow”

至此，**文字水印的基础功能已经添加完成。**

（3）有些时候文字水印希望以本地时间作为水印内容，可以在drawtext滤镜中配合一些特殊用法来完成，在text中显示本地当前时间，格式为年月日时分秒的方式，

ffplay –i input.mp4 –vf ”drawtext=fontsize=60:fontfile=FreeSerif.ttf:text=’%{localtime\:%Y\–%m\–%d %H–%M–%S}’:fontcolor=green:box=1:boxcolor=yellow”

在使用ffmpeg转码存储到文件时需要加上–re，否则时间不对。

ffmpeg –re –i input.mp4 –vf ”drawtext=fontsize=60:fontfile=FreeSerif.ttf:text=’%{localtime\:%Y\–%m\–%d %H–%M–%S}’:fontcolor=green:box=1:boxcolor=yellow” out.mp4

（4）在个别场景中，需要定时显示水印，定时不显示水印，这种方式同样可以配合drawtext滤镜进行处理，使用drawtext与enable配合即可，例如每3秒钟显示一次文字水印：

ffplay –i input.mp4 –vf ”drawtext=fontsize=60:fontfile=FreeSerif.ttf:text=’test’:fontcolor=green:box=1:boxcolor=yellow:enable=lt(mod(t\,3)\,1)”

在使用ffmpeg转码存储到文件时需要加上–re，否则时间不对。

表达式参考：http://www.ffmpeg.org/ffmpeg–utils.html 3 Expression Evaluation

lt(x, y) Return 1 if x is lesser than y, 0 otherwise.

mod(x, y) Compute the remainder of division of x by y.

（5）跑马灯效果

ffplay –i input.mp4 –vf ”drawtext=fontsize=100:fontfile=FreeSerif.ttf:text=’helloworld’:x=mod(100*t\,w):y=abs(sin(t))*h*0.7”

修改字体透明度，修改字体颜色

ffplay –i input.mp4 –vf ”drawtext=fontsize=40:fontfile=FreeSerif.ttf:text=’liaoqingfu’:x=mod(50*t\,w):y=abs(sin(t))*h*0.7:alpha=0.5:fontcolor=white:enable=lt(mod(t\,3)\,1)”

17.4.2 图片水印

FFmpeg除了可以向视频添加文字水印之外，还可以向视频添加图片水印、视频跑马灯等，本节将重点介绍如何为视频添加图片水印；为视频添加图片水印可以使用movie滤镜，下面就来熟悉一下**movie**滤镜的参数，如表1-5所示。

表1-5 FFmpeg movie滤镜的参数

参数	类型	说明
filename	字符串	输入的文件名，可以是文件，协议，设备
format_name, f	字符串	输入的封装格式
stream_index, si	整数	输入的流索引编号
seek_point, sp	浮点数	Seek输入流的时间位置
streams, s	字符串	输入的多个流的流信息
loop	整数	循环次数
discontinuity	时间差值	支持跳动的时间戳差值

ffmpeg –i input.mp4 –vf ”movie=logo.png[watermark];[in][watermark]overlay=x=10:y=10[out]” output.mp4

Ø 原始视频文件路径：input.mp4

Ø 水印图片路径：logo.png

Ø 水印位置：(x,y)=(10,10)<=(left,top)距离左侧、顶部各10像素；

Ø 输出文件路径：output.mp4

overlay过滤器

描述：前景窗口(第二输入)覆盖在背景窗口(第一输入)的指定位置。

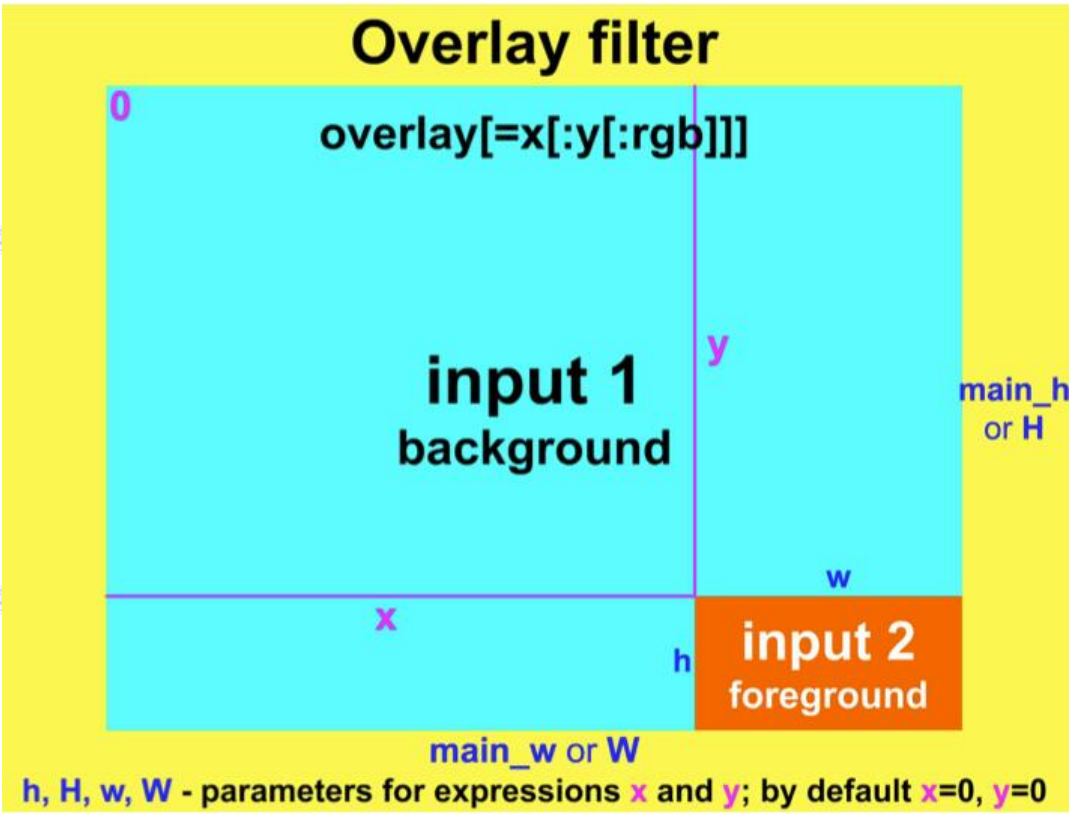
语法：overlay[=x:y[[:rgb={0, 1}]]
参数 x 和 y 是可选的，默认为 0。
参数 rgb 参数也是可选的，其值为 0 或 1，默认为 0。

参数说明：
x 从左上角的水平坐标，默认值为 0
y 从左上角的垂直坐标，默认值为 0
rgb 值为 0 表示输入颜色空间不改变，默认为 0；值为 1 表示将输入的颜色空间设置为 RGB

参数	说明
main_w 或 W	视频单帧图像宽度
main_h 或 H	视频单帧图像高度
overlay_w	水印图片的宽度
overlay_h	水印图片的高度

对应地可以将overlay参数设置成如下值来改变水印图片的位置：

水印图片位置	overlay值
左上角	10:10
右上角	main_w–overlay_w–10:10
左下角	10:main_h–overlay_h–10
右下角	main_w–overlay_w–10:main_h–overlay_h–10



在FFmpeg中加入图片水印有两种方式，一种是通过movie指定水印文件路径，另外一种方式是通过filter读取输入文件的流并指定为水印，这里重点介绍如何读取movie图片文件作为水印。

（1）图片logo.png将会打入到input.mp4视频中，显示在x坐标50、y坐标20的位置

ffplay –i input.mp4 –vf ”movie=logo.png[logo];[in][logo]overlay=50:10[out]”

由于logo.png图片的背景色是白色，所以显示起来比较生硬，如果水印图片是透明背景的，效果会更好，下面找一张透明背景色的图片试一下：

ffplay –i input.mp4 –vf ”movie=logo2.png[watermark];[in][watermark]overlay=50:10[out]”

（2）显示位置

ffplay –i input.mp4 –vf ”movie=logo.png[watermark];[in][watermark]overlay=10:10[out]”

ffplay –i input.mp4 –vf ”movie=logo.png[watermark];[in][watermark]overlay=main_w–overlay_w–10:10[out]”

ffplay –i input.mp4 –vf ”movie=logo.png[watermark];[in][watermark]overlay=10:main_h–overlay_h–10[out]”

ffplay –i input.mp4 –vf ”movie=logo.png[watermark];[in][watermark]overlay=main_w–overlay_w–10:main_h–overlay_h–10[out]”

（3）跑马灯效果

ffplay –i input.mp4 –vf ”movie=logo.png[watermark];[in][watermark]overlay=x=mod(50*t\,main_w):y=abs(sin(t))*h*0.7[out]”

17.4.3 FFmpeg生成画中画

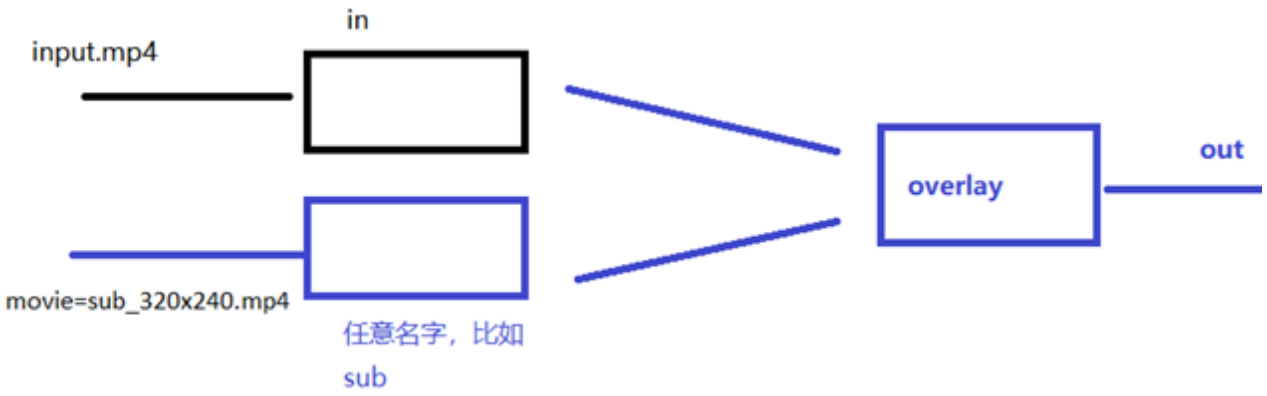
在使用FFmpeg处理流媒体文件时，有时需要使用画中画的效果。在FFmpeg中，可以通过overlay将多个视频流、多个多媒体采集设备、多个视频文件合并到一个界面中，生成画中画的效果。在前面的滤镜使用中，以至于以后的滤镜使用中，与视频操作相关的处理，大多数都会与overlay滤镜配合使用，尤其是用在图层处理与合并场景中，下面就来了解一下overlay的参数，具体见表1–6。

表1–6 FFmpeg滤镜overlay基本参数

参数	类型	说明
x	字符串	X坐标
y	字符串	Y坐标
eof_action	整数	遇到eof表示时的处理方式，默认为重复 Ø repeat(值为0)：重复前一帧 Ø endcall(值为1)：停止所有的流 Ø pass(值为2)：保留主图层
shortest	布尔	终止最短的视频时全部终止（默认false）

format	整数	设置output的像素格式，默认为yuv420 Ø yuv420 (值为0) Ø yuv422 (值为1) Ø yuv444 (值为2) Ø rgb (值为3)

从参数列表中可以看到，主要参数并不多，但实际上在overlay滤镜使用中，还有很多组合的参数可以使用，可以使用一些内部变量，例如overlay图层的宽、高、坐标等。



（1）显示画中画效果

ffplay –i input.mp4 –vf ”movie=sub_320x240.mp4[sub];[in][sub]overlay=x=20:y=20[out]”

ffplay –i input.mp4 –vf ”movie=sub_320x240.mp4[sub];[in][sub]overlay=x=20:y=20:eof_action=1[out]”

ffplay –i input.mp4 –vf ”movie=sub_320x240.mp4[sub];[in][sub]overlay=x=20:y=20:shortest =1[out]”

缩放子画面尺寸

ffplay –i input.mp4 –vf ”movie=sub_320x240.mp4,scale=640x480[sub];[in][sub]overlay=x=20:y=20[out]”

（2）跑马灯

ffplay –i input.mp4 –vf ”movie=sub_320x240.mp4[test];[in][test]overlay= x=mod(50*t\,main_w):y=abs(sin(t))*main_h*0.7[out]”

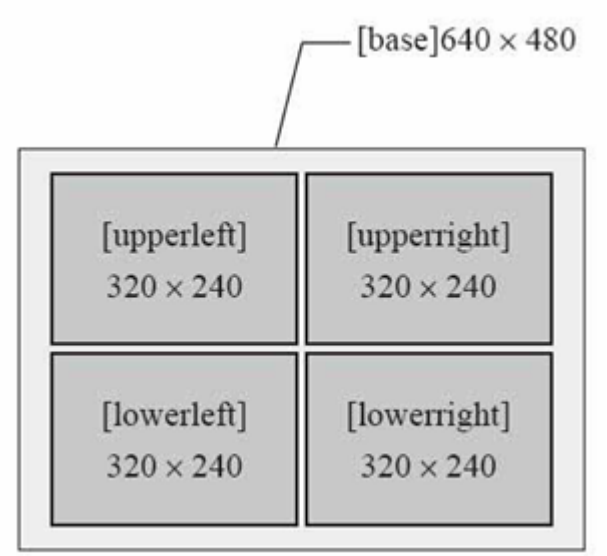
17.4.4 FFmpeg视频多宫格处理

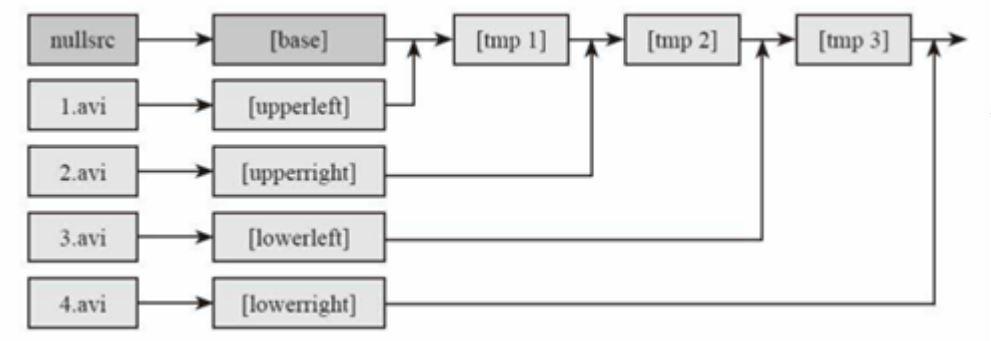
视频除了画中画显示，还有一种场景为以多宫格的方式呈现出来，除了可以输入视频文件，还可以输入视频流、采集设备等。从前文中可以看出进行视频图像处理时，overlay滤镜为关键画布，可以通过FFmpeg建立一个画布，也可以使用默认的画布。如果想以多宫格的方式展现，则可以自己建立一个足够大的画布，下面就来看一下多宫格展示的例子：

ffmpeg –i 1.mp4 –i 2.mp4 –i 3.mp4 –i 4.mp4 –filter_complex ”nullsrc=size=640x480[base];[0:v] setpts=PTS–STARTPTS,scale=320x240[upperleft];[1:v]setpts=PTS–STARTPTS,scale=320x240[upperright];[2:v]setpts=PTS–STARTPTS, scale=320x240[lowerleft];[3:v]setpts=PTS–STARTPTS,scale=320x240[lowerright];[base][upperleft]overlay=shortest=1[tmp1];[tmp1][upperright]overlay=shortest=1:x=320[tmp2];[tmp2][lowerleft]overlay=shortest=1:y=240[tmp3];[tmp3][lowerright]overlay=shortest=1:x=320:y=240” out.mp4

1.2.3.4.mp4为文件路径，out.MP4为输出文件路径，通过nullsrc创建overlay画布，画布大小640:480,

使用[0:v][1:v][2:v][3:v]将输入的4个视频流去除，分别进行缩放处理，然后基于nullsrc生成的画布进行视频平铺，命令中自定义upperleft,upperright,lowerleft,lowerright进行不同位置平铺。





只叠加左上右上的命令：

ffmpeg –i 1.mp4 –i 2.mp4 –i 3.mp4 –i 4.mp4 –filter_complex ”nullsrc=size=640x480[base];[0:v]setpts=PTS–STARTPTS,scale=320x240[upperleft];[1:v]setpts=PTS–STARTPTS,scale=320x240[upperright];[base][upperleft]overlay=shortest=1[tmp1];[tmp1][upperright]overlay=shortest=1:x=320” out2.mp4