@girish1729

ffmpeg cheatsheet

Power your multimedia skills on Linux

ffmpeg cheatsheet

Girish Venkatachalam

ffmpeg cheatsheet

- FFmpeg cheat sheet
  - Basic conversion
  - Remux an MKV file into MP4
  - High-quality encoding
  - Trimming
  - Mux video and audio from another video
  - Concat demuxer
  - Delay audio/video
  - Burn subtitles
  - Extract the frames from a video
  - Rotate a video
  - Download “Transport Stream” video streams

Mute some of the audio

Deinterlace

Create a video slideshow from images

Extract images from a video

Metadata: Change the title

Numbered images to video:

Concatenate multiple videos into one

Speed up a video

Extract audio from .MKV to .MP3

Recording

Cropping

Extracting Audio Stream

Handling id3 tags

Resample/Convert Audio

FFmpeg cheat sheet

Basic conversion

```
$ ffmpeg -i in.mp4 out.avi
```

Remux an MKV file into MP4

```
$ ffmpeg -i in.mkv -c:v copy -c:a copy out.mp4
```

High-quality encoding

Use the `crf` (Constant Rate Factor) parameter to control the output quality. The lower crf, the higher the quality (range: 0-51). The default value is 23, and visually lossless compression corresponds to `-crf`

`18`. Use the `preset` parameter to control the speed of the compression process.

```
$ ffmpeg -i in.mp4 -preset slower -crf 18 out.mp4
```

Trimming

Without re-encoding:

```
$ ffmpeg -ss [start] -i in.mp4 -t [duration] -c copy out.mp4
```

- [`-ss`] specifies the start time, e.g. `00:01:23.000` or `83` (in seconds)
- [`-t`] specifies the duration of the clip (same format).
- Recent `ffmpeg` also has a flag to supply the end time with `-to`.
- [`-c`] copy copies the first video, audio, and subtitle bitstream from the input to the output file without re-encoding them. This

won’t harm the quality and make the command run within seconds.

- With re-encoding:

If you leave out the `-c copy` option, `ffmpeg` will automatically re-encode the output video and audio according to the format you chose.

For example:

```
$ ffmpeg -ss [start] -i in.mp4 -t [duration] -c:v libx264 -c:a aac -strict experimental -b:a 128k out.mp4
```

Mux video and audio from another video

To copy the video from in0.mp4 and audio from in1.mp4:

```
 $ ffmpeg -i in0.mp4 -i in1.mp4 -c copy -map 0:0 -map 1:1 -shortest out.mp4
```

- With [-c copy] the streams will be `stream copied`, not re-encoded, so there will be no quality loss.
- The `-shortest` option will cause the output duration to match the duration of the shortest input stream.

Concat demuxer

First, make a text file.

```
$ cat > list.txt

file 'in1.mp4'

file 'in2.mp4'

file 'in3.mp4'

file 'in4.mp4'
```

Then, run `ffmpeg`:

```
ffmpeg -f concat -i list.txt -c copy out.mp4
```

Delay audio/video

- Delay video by 3.84 seconds:

```
$ ffmpeg -i in.mp4 -itsoffset 3.84 -i in.mp4 -map 1:v -map 0:a
-vcodec copy -acodec copy out.mp4
```

- Delay audio by 3.84 seconds:

```
$ ffmpeg -i in.mp4 -itsoffset 3.84 -i in.mp4 -map 0:v -map 1:a
-vcodec copy -acodec copy out.mp4
```

Burn subtitles

Use the `libass` library

Ensure your ffmpeg install has the library in the configuration `--enable-libass`.

First convert the subtitles to .ass format:

```
$ ffmpeg -i sub.srt sub.ass
```

Then add them using a video filter:

```
$ ffmpeg -i in.mp4 -vf ass=sub.ass out.mp4
```

Extract the frames from a video

To extract all frames from between 1 and 5 seconds, and also between 11 and 15 seconds:

```
$ ffmpeg -i in.mp4 -vf select='between(t,1,5)+between(t,11,15)'
-vsync 0 out%d.png
```

To extract one frame per second only:

```
$ ffmpeg -i in.mp4 -fps=1 -vsync 0 out%d.png
```

Rotate a video

- Rotate 90 clockwise:

```
$ ffmpeg -i in.mov -vf "transpose=1" out.mov
```

For the transpose parameter you can pass:

0 = 90CounterCLockwise and Vertical Flip (default) 1 = 90Clockwise

2 = 90CounterClockwise 3 = 90Clockwise and Vertical Flip

Use `-vf "transpose=2,transpose=2"` for 180 degrees.

Download “Transport Stream” video streams

1. Locate the playlist file, e.g. using Chrome > F12 > Network > Filter: m3u8
2. Download and concatenate the video fragments:

```
$ ffmpeg -i "path_to_playlist.m3u8" -c copy -bsf:a
aac_adtstoasc out.mp4
```

If you get a “Protocol ‘https not on whitelist ’file,crypto’!” error, add the `protocol_whitelist` option:

```
$ ffmpeg -protocol_whitelist "file,http,https,tcp,tls" -i

"path_to_playlist.m3u8" -c copy -bsf:a aac_adtstoasc out.mp4
```

Mute some of the audio

To replace the first 90 seconds of audio with silence:

```
ffmpeg -i in.mp4 -vcodec copy -af

"volume=enable='lte(t,90)':volume=0" out.mp4
```

To replace all audio between 1’20" and 1’30" with silence:

```
$ ffmpeg -i in.mp4 -vcodec copy -af

"volume=enable='between(t,80,90)':volume=0" out.mp4
```

Deinterlace

- Deinterlacing using “yet another deinterlacing filter”.

```
$ ffmpeg -i in.mp4 -vf yadif out.mp4
```

Create a video slideshow from images

Parameters: - `-r` marks the image framerate (inverse time of each image) -`vf fps=25` marks the true framerate of the output.

```
$ ffmpeg -r 1/5 -i img%03d.png -c:v libx264 -vf fps=25 -pix_fmt yuv420p out.mp4
```

Extract images from a video

- Extract all frames:

```
$ ffmpeg -i input.mp4 thumb%04d.jpg -hide_banner
```

- Extract a frame each second:

```
$ ffmpeg -i input.mp4 -vf fps=1 thumb%04d.jpg -hide_banner`
```

- Extract only one frame:

```
$ ffmpeg -i input.mp4 -ss 00:00:10.000 -vframes 1 thumb.jpg
```

Metadata: Change the title

```
$ ffmpeg -i in.mp4 -map_metadata -1 -metadata title="My Title"
-c:v copy -c:a copy out.mp4
```

- Variable bit rate 1080p MP3:

```
 $ ffmpeg -i input_video -vcodec libx264 -crf 25 -preset medium
-vf scale=-2:1080 -acodec libmp3lame -q:a 4 -ar 48000 -ac 2
output_video.mp4
```

- Fixed bit rate 1080p MP2:

```
 $ ffmpeg -i input_video -vcodec libx264 -b:v 1000k -vf
scale=-2:1080 -acodec mp2 -b:a 256k -ar 48000 -ac 2
output_video.mp4
```

- No audio:

```
 $ ffmpeg -i input_video -vcodec libx264 -b:v 1000k -vf
scale=-2:1080 -an output_video.mp4
```

- Crop size (width:height:xoffset:yoffset):

```
 $ ffmpeg -i input_video -vf crop=1500:800:200:100 -vcodec
libx264 -b:v 1000k -an output_video.mp4
```

- Trim time (-ss start time, -t duration):

```
 $ ffmpeg -i input_video -vcodec libx264 -b:v 1000k -an -ss
00:00:10 -t 00:00:10 output_video.mp4
```

- Mix audio and video:

```
   ffmpeg -i input_video -vcodec libx264 -b:v 1000k -vf
crop=1120:876:0:100 -i input_audio -acodec mp2 -b:a 256k -ar
48000 -ac 2 -ss 00:00:20 -t 00:00:20 output_video.mp4
```

- Crop, pan, composite:

```
  $ ffmpeg -i input_video_1 -i input_video_2 -filter_complex
'[1:v]crop=175:95:930:860[cropout];
[cropout]scale=350:190[scaleout];[0:v]
[scaleout]overlay=10:10[outv]' -map '[outv]' -vcodec libx264 -
```

```
b:v 1000k -map 0:a -acodec mp2 -b:a 256k -ac 2 -t 00:00:05
output_video.mp4
```

Numbered images to video:

```
 $  ffmpeg -r 30 -i %04d.jpg -vcodec libx264 -b:v 1000k -vf
scale=-2:1080 -an output_video.mp4
```

- FFmpeg export audio from any video to mp3

```
$ ffmpeg -i "$video" -vn -c:a libmp3lame -y "$audio";
```

- FFmpeg export frames from video to images

```
$ ffmpeg -i "$video" "$frames_folder/%08d.ppm";
```

- Retrieve the frame rate from the input video

- To view it on screen

```
$ ffprobe -v 0 -of csv=p=0 -select_streams v:0 -show_entries
stream=r_frame_rate "$video";
```

- To create a video out of a folder with frames/images and an audio file.

```
$ ffmpeg -framerate "$frame_rate" -i "$frames_folder/%08d.ppm"
-i "$audio" -pix_fmt yuv420p -acodec copy -y "$output_video";
```

- To set a custom starting index for the frames you can use the -start_number argument

```
$ ffmpeg -start_number 62 -framerate "$frame_rate" -i
"$frames_folder/%08d.ppm" -i "$audio" -pix_fmt yuv420p -acodec
copy -y "$output_video";
```

- To use the MP4 coded use -vcodec libx264

```
$ ffmpeg -framerate "$frame_rate" -i "$frames_folder/%08d.ppm"
-i "$audio" -vcodec libx264 -pix_fmt yuv420p -acodec copy -y
"$output_video";
```

- To merge an audio less video with an audio file

```
$ ffmpeg -i "$no_audio_video" -i "$audio" -shortest -vcodec
copy -acodec copy "$output_video";
```

- To change the frame rate of a video

```
$ ffmpeg -i "$video" -filter:v fps=20 "$output_video";
```

To merge two videos side by side

```
$ ffmpeg -i "$left_video" -i "$right_video" -filter_complex
hstack "$output_video"
```

Concatenate multiple videos into one

The easiest way without writing huge commands is the following:

First, create a file named `parts.txt` and add content similar to what we list below:

```
$ cat > parts.txt
#Lines starting with # will be ignored
```

```
file 'part00-03.mp4'

file 'part04.mp4'

file 'part05-07.mp4'

file 'part08-09.mp4'

file 'part10.mp4'

file 'part11-13.mp4'
```

Then execute the following command to concatenate all those videos into one:

```
 $ ffmpeg -f concat -safe 0 -i parts.txt -c copy

"$output_video";
```

Speed up a video

Using the following command, you can speed up a video by dropping excess frames:

```
 $ ffmpeg -i "$video" -filter:v "setpts=0.5*PTS"

"$output_video";
```

- The above example will double the speed (the value 0.5 controls it.)

To speed the video up without losing frames, you can increase the FPS value of the output video. To retrieve the frame rate, please see the command that was posted earlier.

```
$ ffmpeg -i "$video" -r 80 -filter:v "setpts=0.25*PTS"
"$output_video";
```

In the second example, we assumed that the input video had 20 frames per second. Using the 0.25 value, we decided to speed the video up by a factor of 4. To preserve the input frames, we increased the frame rate from 20 to 80 using the parameter -r.

Extract audio from .MKV to .MP3

The following command will find all mkv files that are in the current directory and in all sub-folders and extract the audio to mp3 format.

```
$ find . -type f -name "*.mkv" -exec bash -c 'FILE="$1"
$ ffmpeg -i "${FILE}" -vn -c:a libmp3lame -y
"${FILE%.mkv}.mp3";' _ '{}' \;
```

- Extract a 5 seconds video from 3-8

```
$ ffmpeg -ss 00:00:03 -t 00:00:08 -i input.mp4 -async 1 cut.mp4
```

- Cut first X seconds from a video

```
$ ffmpeg -ss 10 -i input.mp4 -async 1 cut.mp4
```

- Mix a video with an audio

```
$ ffmpeg -i original_video.mkv -i clean_audio.mp3 -c:v copy -
c:a flac -map 0:v:0 -map 1:a:0 out.mkv
```

Recording

- Screencasting (software encoding) with dynamic screen size

```
$ ffmpeg -y -f x11grab -s `xdpyinfo | grep 'dimensions:'| awk
'{print $2}'` -i :0.0 -f pulse -i default -c:v libx264 -r 48 -
c:a flac out.mkv
```

- Screencasting (hard encoding) with provided screen size

```
$ ffmpeg -f alsa -i default -c:a flac \
    -vaapi_device /dev/dri/renderD128 -y -f x11grab -s
1920x1080  -i :0.0+1366,0 \
    -vf 'format=nv12|vaapi,hwupload' -c:v h264_vaapi -preset
ultrafast -crf 0 \
    -tune film -r 60 out.mkv
```

- Podcast recording

```
$ ffmpeg -f pulse -i 2 -ac 2 -acodec libmp3lame -ab 320k
out.mp3
```

Cropping

The following will create a 640x480 sized output video by copying a corresponding window at offset x=100px y=25px from the input video

```
$ ffmpeg -i <input> -filter:v "crop=640:480:100:25" <output>
```

- Scaling

```
$ ffmpeg -i <input> -vf scale=640:480 <output>
```

- Cutting a video part

```
$ ffmpeg -i <input> -ss 00:01:45 -t 00:02:35 -vcodec copy -
acodec copy <output>
```

```
$ ffmpeg -ss 00:00:30 -i orginalfile.mpg -t 00:00:05 -vcodec
copy -acodec copy newfile.mpg
```

- Fixing rotation

Do not recode for rotation but simple add a video metadate field for the rotation angle

```
$ ffmpeg -i <input> -c copy -metadata:s:v:0 rotate=90 <output>
```

- H265 2-pass encoding

For H265 2-pass encoding you need to combine 2 ffmpeg calls.

```
$ ffmpeg -y -i input -c:v libx265 -b:v 2600k -x265-params
pass=1 -an -f mp4 /dev/null && \
$ ffmpeg    -i input -c:v libx265 -b:v 2600k -x265-params
pass=2 -c:a aac -b:a 128k output.mp4
```

Extracting Audio Stream

Combine `-vn` (for no video) with `-acodec copy`.

Note that the output file extension must match the audio codec in the input file for “-acodec copy” to work.

```
$ ffmpeg -i file.mp4 -vn -acodec copy output.aac
```

- Creating Thumbnails

To create a single thumb at 10s

```
$ ffmpeg -ss 10 -i <input file> -vframes 1 -vcodec png -an
thumb.png
```

- To create thumbnails every n seconds use “-vf fps=1/n” for example

```
$ ffmpeg -i <input file> -vf fps=1/60 thumbnails/thumb%03d.png
```

Handling id3 tags

- Extracting

```
$ ffmpeg -i file.mp3 -f ffmetadata metadata.txt
```

- Setting metadata.txt

```
 $ ffmpeg -i file.mp3 -acodec copy -metadata title="<title>" -
metadata artist="<artist>" -metadata album="<album>" out.mp3
```

Resample/Convert Audio

```
$ ffmpeg -i file.aac -acodec mp3 -ar 44100 -ab 128000
output.mp3
```

- Change container from MKV to MP4

```
$ ffmpeg -i file.mkv -acodec copy -vcodec copy file.mp4
```

- Video from Images

If you have multiple numbered images image1.jpg, image2.jpg…

create a video from them like this

```
$ ffmpeg -f image2 -i image%d.jpg video.mp4
```

- Split Video to Images

```
$ ffmpeg -i video.mp4 image%d.jpg
```

- Mute some of the audio
    - To replace the first 90 seconds of audio with silence:

```
$ ffmpeg -i in.mp4 -vcodec copy -af

"volume=enable='lte(t,90)':volume=0" out.mp4
```

- To replace all audio between 1:20 and 1:30 with silence:

```
$ ffmpeg -i in.mp4 -vcodec copy -af

"volume=enable='between(t,80,90)':volume=0" out.mp4
```