ffmpeg/ffprobe

1. Trim video
   `ffmpeg -i input.mp4  -ss 00:00:00 -to 00:00:15 -c:v copy -c:a copy output.mp4`
   This command trim video from 00:00:00 to 00:00:15 (final duration = 15 sec).
   The `-c:v copy -c:a copy` commands copy the original audio and video without re-encoding.
   CAUTION! This way may cause add black frames in the head of a video.
   How to deal with the black screen
   The black screen at the beginning of your video is bound to spoil this lovely story. Here's how to avoid a black screen
   Use the "-avoid_negative_ts make_zero" parameter
   What you do here is to pass that part that reads "zero" to avoid any negative section. i.e "make_zero" to "avoid_negative_ts"
   Hence, the video timestamp will be shifted to "0". So, if the time you specified didn't fall into the rightful place, it will automatically be corrected to begin at the "start" point.
   `ffmpeg -i input.mp4  -ss 00:00:15 -to 00:00:20 -c:v copy -c:a copy -avoid_negative_ts make_zero output.mp4`

   Another way to deal with black screen problem is replace seek parameter
   `ffmpeg -ss 00:00:15 -i input.mov -to 00:00:20 -c copy -avoid_negative_ts make_zero output.mov`
   So, to workaround this effect run a command with no copy video and audio. For example:
   `ffmpeg -i input.mp4 -ss 00:00:00 -to 00:00:15 output.mp4`
2. **Removing the first seconds** `ffmpeg -ss 00:05 -i input.mp4 -c:v libx264 -c:a aac output.mp4`

Removing the first seconds without re-encoding `ffmpeg -ss 00:05 -i input.mp4 -c:v copy -c:a copy output.mp4`

3. Speed up video
   `ffmpeg -i input.mp4 -r 60 -vf "setpts=0.5*PTS" output.mp4`
4. Shrink big MP4 video file to smaller sizes
   - **3.6 Gb** to **556 Mb**, great quality □□
   `ffmpeg -i input.mp4 -vcodec h264 -acodec mp2 output.mp4`
   - **3.6 Gb** to **62 Mb**, quality "good enough"/acceptable □□
   `ffmpeg -i input.mp4 -s 1280x720 -acodec copy -y output.mp4`
   - **3.6 Gb** to **30 Mb**, very shitty quality □□
   `ffmpeg -i input.mp4 -vcodec h264 -b:v 1000k -acodec mp3 output.mp4`
   - CPU transcoding to 720p with hevc codec
   `ffmpeg -i input.mov -s 1280x720 -vcodec hevc -acodec copy -y output_720p.mov`
   - GPU transcoding to 720p with hevc codec
   `ffmpeg -y -vsync 0 -hwaccel cuda -hwaccel_output_format cuda -i input.mov -vf scale_cuda=1280:720 -c:a copy -c:v hevc_nvenc output_720p_hevc_nvenc.mov`
5. Create video from images

   MacOS: Add pixel format.
   `ffmpeg -framerate 10 -i filename-%03d.jpg -pix_fmt yuv420p output.mp4`

`ffmpeg -framerate 10 -pattern_type  glob -i "filename-*.jpg" -pix_fmt yuv420p output.mp4`

6. Concatenate videos
   Create txt file with list of videos:
   `for f in *.mov; do echo "file '$f'" >> videos.txt; done`
   Concat videos:
   `ffmpeg -f concat -i videos.txt -c copy output_[concated.mov](http://concated.mov/)`
7. Get 1 frame from video
   1. With the seek option
      `ffmpeg -ss 00:00:01 -i input.mov -frames:v 1 output.jpeg`
   2. With select filter
      `ffmpeg -i input.mov -vf "select=eq(n\,34)" -frames:v 1 output.jpeg`
8. Get every N frame from video
   - It will get `1 frame every 10 seconds` instead of `1 frame every 1 second ffmpeg -i out1.avi -r 0.1 -f image2 image-%3d.jpeg`
   - If you really what every 10th frame from video then you can use `select` with `modulo 10` `ffmpeg -i out1.mp4 -vf "select=not(mod(n\,10))" -vsync vfr image_%03d.jpg`
9. Get fps from video
   `ffprobe -v error -select_streams v:0 -show_entries stream=r_frame_rate -of default=noprint_wrappers=1:nokey=1 input/training_sessions/140048_15sec/140048_4_2023-05-24_15.14.42_15sec_sync.mov | bc -l`
10. Convert mov to mp4 `ffmpeg -i input.mov -crf 0 output.mp4`
11. Obtain certain information about video file `ffprobe -v error -select_streams v:0 -show_entries stream=width,height,duration,bit_rate -of default=noprint_wrappers=1 input.mp4`
12. Convert mp3 to wav `ffmpeg -i file.mp3 -acodec pcm_s16le -ac 1 -ar 22050 file.wav` This command generate wav file with shorter duration
13. Show audio file info `ffmpeg -i file.wav -hide_banner`
14. **How to generate a sine wave with ffmpeg?** `ffmpeg -f lavfi -i "sine=frequency=15000:sample_rate=48000:duration=0.5" -c:a mp3 -b:a 192k -af "volume=40dB" 15kHz40.mp3`
15. **Changing volume**

To change the audio volume, you may use FFmpeg's volume audio filter.

If we want our volume to be half of the input volume:

`ffmpeg -i input.wav -filter:a "volume=0.5" output.wav`

150% of current volume:

`ffmpeg -i input.wav -filter:a "volume=1.5" output.wav`

You can also use decibel measures. To increase the volume by 10dB:

`ffmpeg -i input.wav -filter:a "volume=10dB" output.wav`

To reduce the volume, use a negative value:

`ffmpeg -i input.wav -filter:a "volume=-5dB" output.wav`

Note that the `volume` filter only **adjusts** the volume. It does not **set** the volume. To set or otherwise normalize the volume of a stream, see the sections below.

16. Concat two audio files `ffmpeg -i file1.mp3 -i file2.mp3 -filter_complex "[0:a][1:a]concat=n=2:v=0:a=1" final.mp3`
17. Encode a high quality MP3 or MP4 audio from a movie file `ffmpeg -i sample.avi -q:a 0 -map a sample.mp3`
18. Trim audio file
    `ffmpeg -i a.mp3 -ss 00:00:00 -to 00:05:00 b.mp3`

Revision #1
Created 7 October 2025 08:27:35 by annndruha
Updated 20 March 2026 21:46:23 by annndruha