The Magical Video Converter: ffmpeg Artist-name -metadata title artist -metadata album

Command line utility `ffmpeg`, dedicated to different transformations of video, is one of the tools which genuinly feel like magic.
`ffmpeg` download link can do a lot, so it is no surprise that the documentation is overwhelming.
Below are some usecases so that I do not have to search for them again and again.

Use cases

Play first 20 sec of multiple videos

I downloaded several lectures, but not sure how they go one after another, the sequential number of the lecture appears in first 20 sec of the video.
So I want to play the first 20 sec of all these videos. Videos are in `mkv` format and all of them are stored in one folder

```
for file in *.mkv; do ffplay -ss 1 -t 20 "$file" ; done
```

NOTE: when video stops playing, the window of the video needs to be closed manually to start playing the next one

Convert between formats

This probably cannot be easier using `ffmpeg`

- single video:

```
ffmpeg -i input_video.mp4 output_video.avi
```

- multiple videos:

```
for file in *.mp4;
do ffmpeg -i "${file}" "${file/.mp4/.avi}"
done
```

Video

How to process recordings from the meetings

Trimming

```
ffmpeg -i input.mp4 -ss 00:00:01 -to 01:09:50 -c:v copy -c:a copy output.mp4
```

Joining files

```
ffmpeg -i "concat:input1.ts|input2.ts|input3.ts" -c copy output.ts
```

Compres scale, increase audio

```
ffmpeg -i input.mp4 -vf "scale=1280x720" -filter:a "volume=8.0" -vcodec libx264 -crf 28 -preset slow -acodec
```

This makes the meeting 8x louder, compresses the recording and makes the resolution smaller
Processing on the i5 from 2021 laptop is 1/2 time of the total length of the recording
Compressess from … min video of … GB to …GB

Speed up or slow down video and remove audio

```
ffmpeg -i input.mp4 -filter:v "setpts=0.5*PTS" -an output.mp4
```

Here is a breakdown of the various parameters:

- '-i input.mp4' : Specifies the input video file that FFmpeg will process. Replace input.mp4 with the name of your video file.

- '-filter:v' : Applies a video filter to the input video stream.
- 'setpts=0.5*PTS' : It reduces the PTS by half (0.5), which makes the output video play twice as fast as the original video. You can adjust the figure to suit your use case. For example, using 0.25PTS will make the output video play four times faster.
- '-an' : The -an stands for "audio none" and it disables the audio stream in the output file.

Change the audio intensity

```
ffmpeg -i input.mp4 -filter:a "volume=2.0" -c:v copy louder.mp4
```

To test it by listening:

```
ffplay -i input.mp4 -af "volume=2.0"
```

Speed up or slow down both audio and video

```
ffmpeg -i input.mp4 -vf "setpts=0.5*PTS" -af "atempo=2.0" output.mp4
```

This is a bit convoluted, speeding up audio is the inverse of what you put for video * '-af' : This flag applies an audio filter. * 'atempo=2.0' : Doubles the speed of the audio to match the speed of the output video. This will make the audio remain in sync with the video.

Convert a folder of tiff files into video specifying the frames per second

```
ffmpeg -framerate 7 -start_number 1 -i "frame_%03d.tif" \
  -c:v libx264 -pix_fmt yuv420p -crf 23 -preset medium -movflags +faststart \
  video.mp4
```

- replace `frame_` with the common part of the name
- `-crf N` gives the quality/compression 1 is least compressed 23 is most compressed

Use only every 5-th image for the video

```
ffmpeg -framerate 30 -pattern_type glob -i "*.jpg" \
  -vf "select=not(mod(n\,5)),setpts=N/FRAME_RATE/TB" \
  -r 30 out.mp4
```

Scale down

single video:

```
ffmpeg -i input_video.mp4 -vf "scale=1280:720" output_video.mp4
```

multiple videos:

```
for file in *.mp4;
do ffmpeg -i "${file}" -vf "scale=1280:720" "${file/.mp4/_scaled.mp4}";
done
```

Compress

With h264 codec, works in windows media player, faster:

```
ffmpeg -i input.mp4 -vcodec libx264 -crf 28 -preset slow -acodec aac -b:a 96k output.mp4
```

Compressed 489MB 6 min mp4 video from capcut to 8MB mp4 video; took 2-3min

With h265 codec, does not work natively in windows media player:

```
ffmpeg -i input.mp4 -vcodec libx265 -crf 28 -preset medium -acodec aac -b:a 128k output.mp4
```

Compressed 489MB 6min mp4 video from capcut to 9MB mp4 video; took 7min

Extract audio from video

- single file:

```
ffmpeg -i input_video.mp4 -q:a 0 -map a output_audio.mp3
```

- many files:

```
for file in *.mp4;
do ffmpeg -i "${file}" -q:a 0 -map a "${file/.mp4/.mp3}";
done
```

Change audio from mono to stereo

- single file: `ffmpeg -i input_mono.mp3 -ac 2 output_stereo.mp3`
- multiple files:

```
for file  in *.mp3;
do ffmpeg -i "${file}" -ac 2 "${file/.mp3/stereo.mp3}";
done
```

Add the metadata artist and track to mp3 files so that it is recognized by media players

Single manual file:

```
ffmpeg -i input.mp3 -metadata artist="Artist-name" -metadata title="Track-title" -c copy output.mp3
```

Automated for files in folder which is a named as the album 1. Get into the folder with the mp3 files 2. Run following code replacing "artist", "album" with the appropriate text

```
for file in *.mp3;
ffmpeg -i $file -metadata artist="artist" -metadata album="album" -metadata title="${file/.mp3/}" -c copy "${
done;
```

Make gifs from video

- single file:

```
ffmpeg -i input_video.mp4 -vf "fps=10,scale=320:-1:flags=lanczos" -c:v gif output.gif`
```

- multiple files:

```
for file in *.mp4;
do ffmpeg -i "${file}" -vf "fps=10,scale=320:-1:flags=lanczos" -c:v gif "${file/.mp4/.gif}
done;
```