FFMPEG Quick Hacks

Quickly learn to use the free command-line video-editing utility FFMPEG - cut, copy, record, edit, tag, convert, rotate, flip, resize, crop, combine, compose, blur, sharpen, smoothen, side-by-side split, PIP inset, fade in/out... Also learn to use subtitles, sound, images, animations and metadata with video.

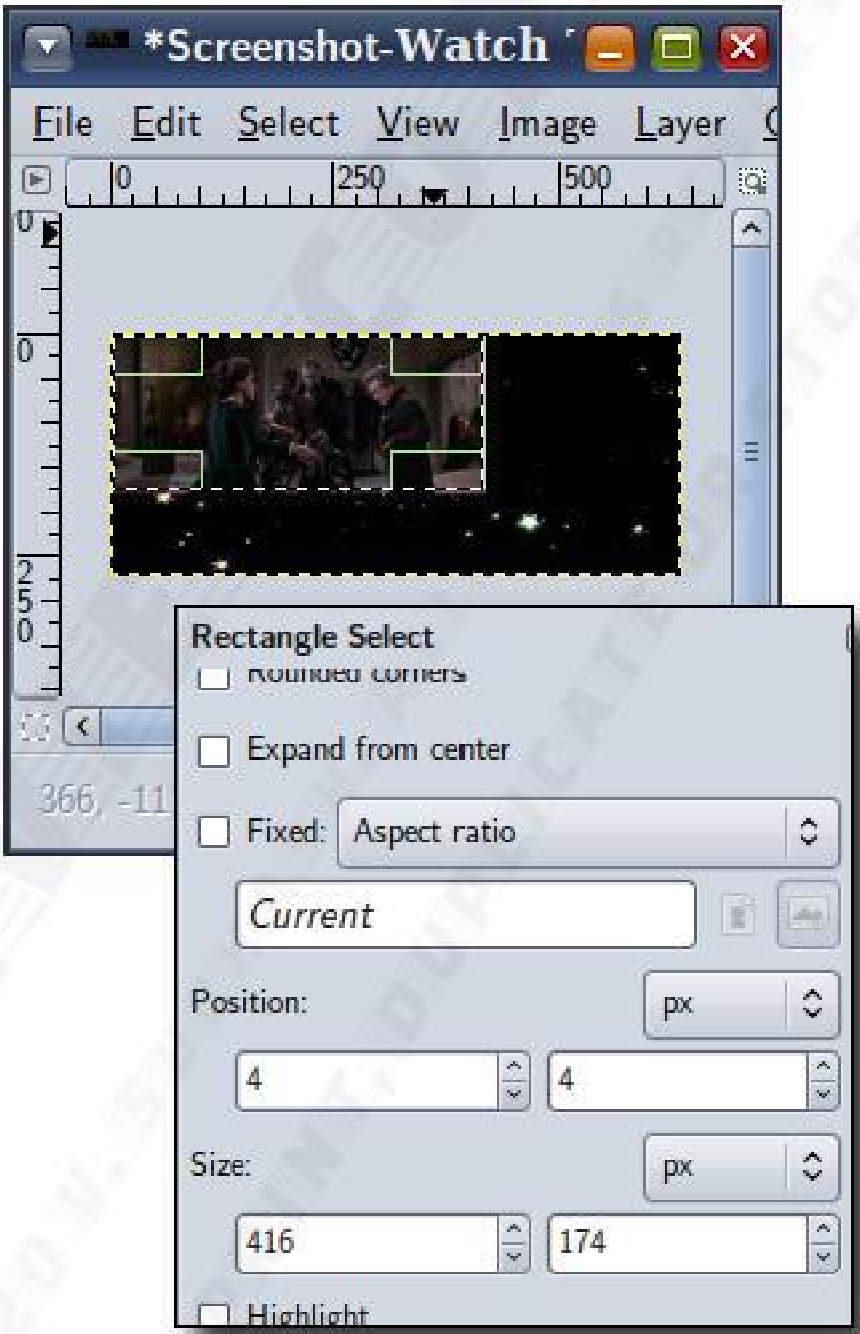
V. Subhash

Contents

Chapters

```
    -filter:v "eq=brightness=-0.3:enable='between(t,6,12)'" \
    jackcc.mp4
```

Crop a video



Fixed: Aspect ratio

Current

Position: px

4 4

Size: px

416 174

Some other video uploaders place the original video on a larger background to fool content identification techniques. If you need to get rid of the background, you need to crop the video.

To know the dimensions and position of the original video on the background, play the video on a video player (Totem or VLC) and take a still (snapshot).

[illegible]

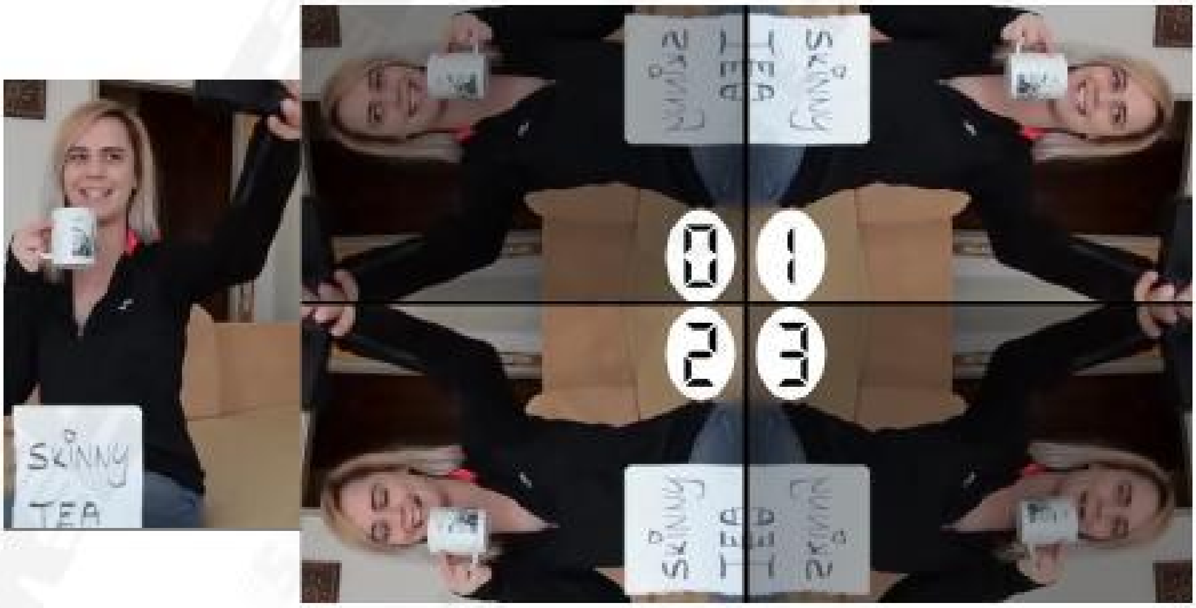
the original video. This will help you find the dimensions and position of the area of the video that needs to be cropped.

```
ffmpeg -i Hunted-Palace.mp4 \
       -filter "crop=416:174:4:4" \
       Haunted-Palace.mp4
```

Rotate a video

Sometimes, the videos that people take from a mobile phone are rotated by 90 or 180 degrees from normal. You can fix it by specifying a `transpose=dir:passthrough` filter.

```
ffmpeg -i barb.mp4 \
       -filter:video "transpose=1" \
       barb-rotated.mp4
```



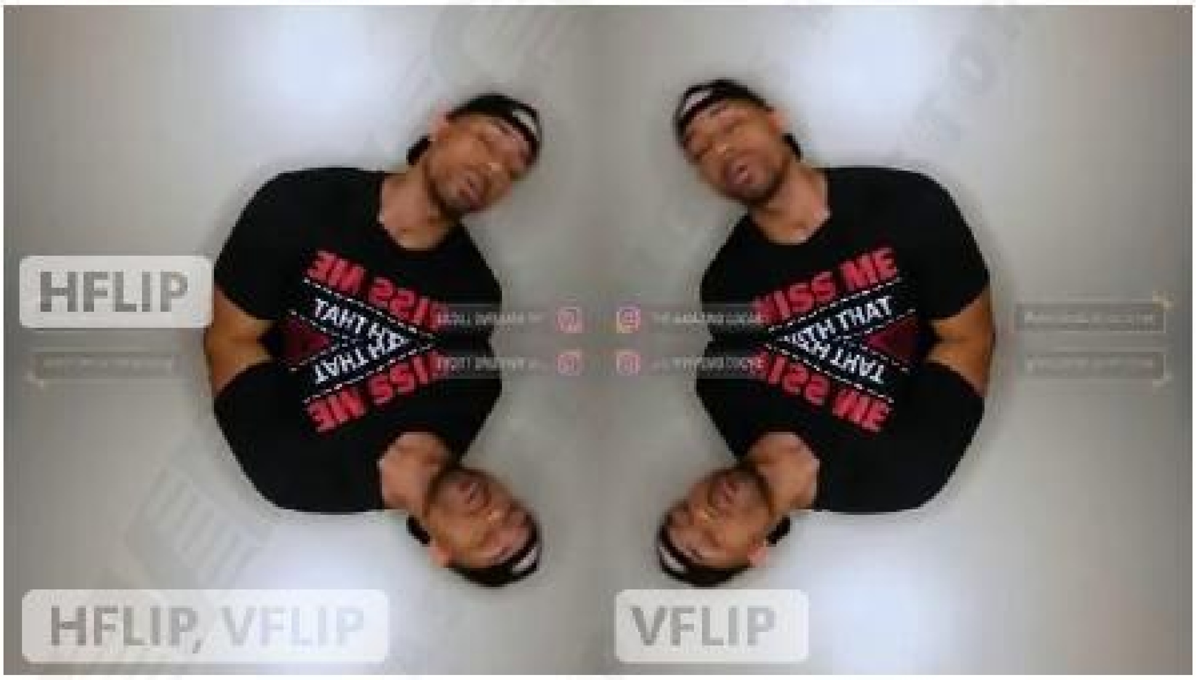
For `dir`, 1 or 2 turns the video 90 degrees **right or left**. Values 0 and 3 turns the video **left or right** and also vertically flips them.

The `passthrough` key can have values `none`, `portrait` and `landscape`. It can be set to one of the last two options to retain the orientation and prevent unnecessary rotation.

Flip a video

Some uploaders flip the videos to get past piracy filters. Use `vflip` or `hflip` to flip them back to their original.

```
ffmpeg -i lucas.mp4 \
       -filter:v "vflip" \
       lucas-upside-down.mp4
ffmpeg -i lucas.mp4 \
       -filter:v "hflip" \
       lucas-half-crazy.mp4
ffmpeg -i lucas.mp4 \
       -filter:v "hflip,vflip" \
       lucas-totally-flipped.mp4
```



Append videos using a filter

In the chapter 'Editing videos', you learned to concatenate videos using a text file. FFMPEG concat filter provides more control if you have only a handful of input files.

```
ffmpeg -i lucas.mp4 -i sarah.mp4 \
  -filter_complex \
    '[0:0][0:1][1:0][1:1]concat=n=2:v=1:a=1[v][a]' \
  -map '[v]' -map '[a]' \
  -vcodec libx264 -r 24 -b:v 266k -s qvga \
  -acodec libmp3lame -b:a 64k -ac 2 \
  -f mp4 train.mp4
```

☞ This will re-encode the input files, as will any other filter.

Delete a portion of a video in the middle

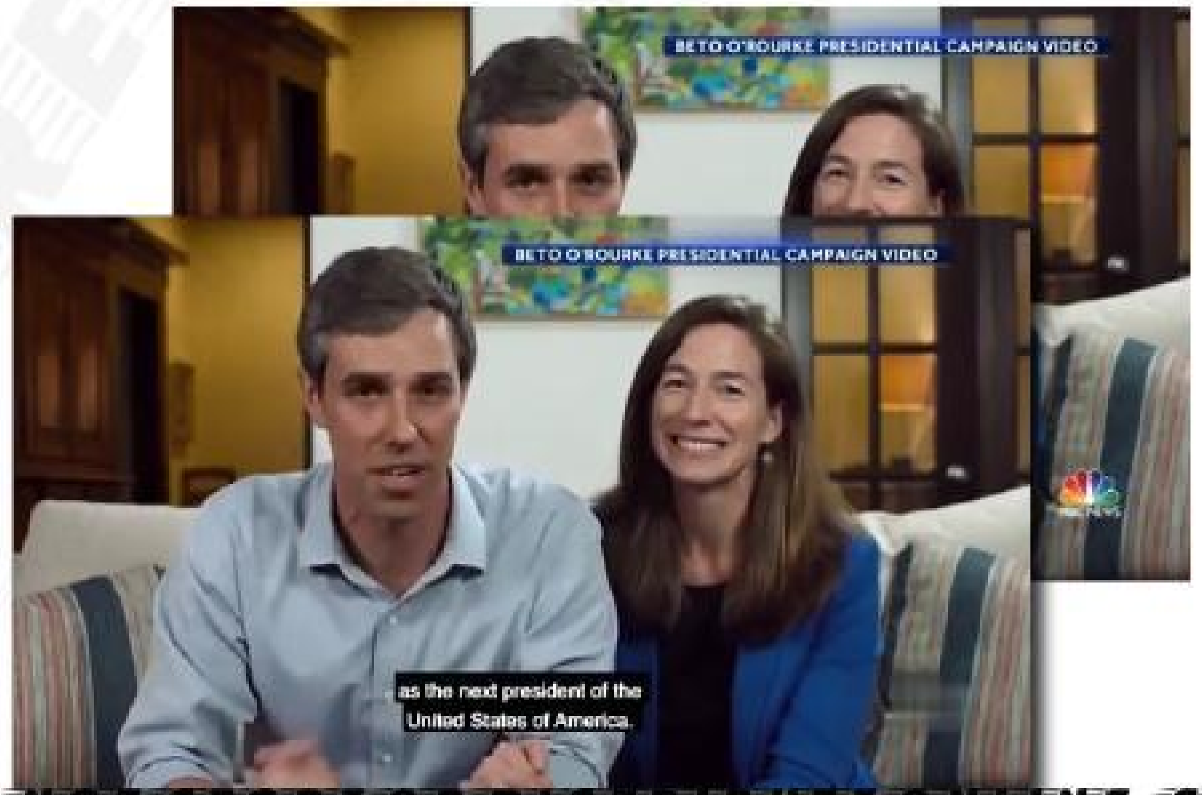
Sometimes, you need to delete a scene from a video. For that, you can use the `trim`, `atrim` and `concat` filters. In this command, the first four filters cut the portion of the video and audio that is required in the output file. In other words, I omitted the scene between 16 and 32 seconds.

```
ffmpeg -y -i barbara.mp4 \
  -filter_complex \
    "[0:v:0]trim=start=0:end=15[lv];
     [0:v:0]trim=start=33:end=50[rv];
     [0:a:0]atrim=start=0:end=15[la];
     [0:a:0]atrim=start=33:end=50[ra];
     [lv][rv]concat=n=2:v=1:a=0,setpts=N/FRAME_RATE/TB[v];
     [la][ra]concat=n=2:v=0:a=1,asetpts=N/SR/TB[a]" \
  -map '[v]' -map '[a]' barb-cut.mp4
```

☞ I have used seconds instead of timestamps because the 'hh:mm:ss' format requires a lot of unintuitive escaping.

☞ The concat filter is prone to timestamp errors. The `setpts` and `asetpts` filters recalibrate the internal timestamps with actual content.

Remove logo


BETO O'ROURKE PRESIDENTIAL CAMPAIGN VIDEO
BETO O'ROURKE PRESIDENTIAL CAMPAIGN VIDEO
as the next president of the
United States of America.

I do not know who needs this but if it floats your boat, then here it is.

```
ffmpeg -f lavfi \
  -i "testsrc[out0];aevalsrc=random(0)/20[out1]" \
  -t 0:0:30 -s 320x260 -pix_fmt yuv420p \
  test.mp4
```

The video has a colour pattern, a scrolling gradient and a changing timestamp. The audio is a low white noise.

☞ This command uses filter as a pseudo file source. I have named the filter output labels in this way to get around a bug.

Speed up a video (fast-forward)

When you fast-forward a video, its duration decreases. When you slow down a video, its duration increases. There is no one filter that fast-forwards both the audio and the video. You need to use to two different filters (one for video and one for audio) to fast-forward a media file. The two filters do not work in the same way and their range is also different. The two need to be calibrated correctly so that the same effect is achieved on the both the audio and video.

For video, you need to set the `setpts` video filter to a fraction of the PTS filter variable. If you want to double the speed of the video, divide PTS by 2. If you want the video to be four times fast, then divide PTS by 4. For audio, you need to use the `tempo` filter. The range of this filter is from half to double the speed. No more or no less. If you want go beyond that limit, you need daisy-chain multiple `atempo` filters.

This command fast-forwards a video by 4x. The `atempo` filter is set at a value of 2 and then used twice.

```
ffmpeg -y -i barb.mp4 \
  -filter_complex \
   "[0:v]setpts=PTS/4[v];
    [0:a]atempo=2, atempo=2[a2]" \
  -map '[v]' -map '[a2]' \
  barb-speed.mp4
```

Slow down a video

[illegible]

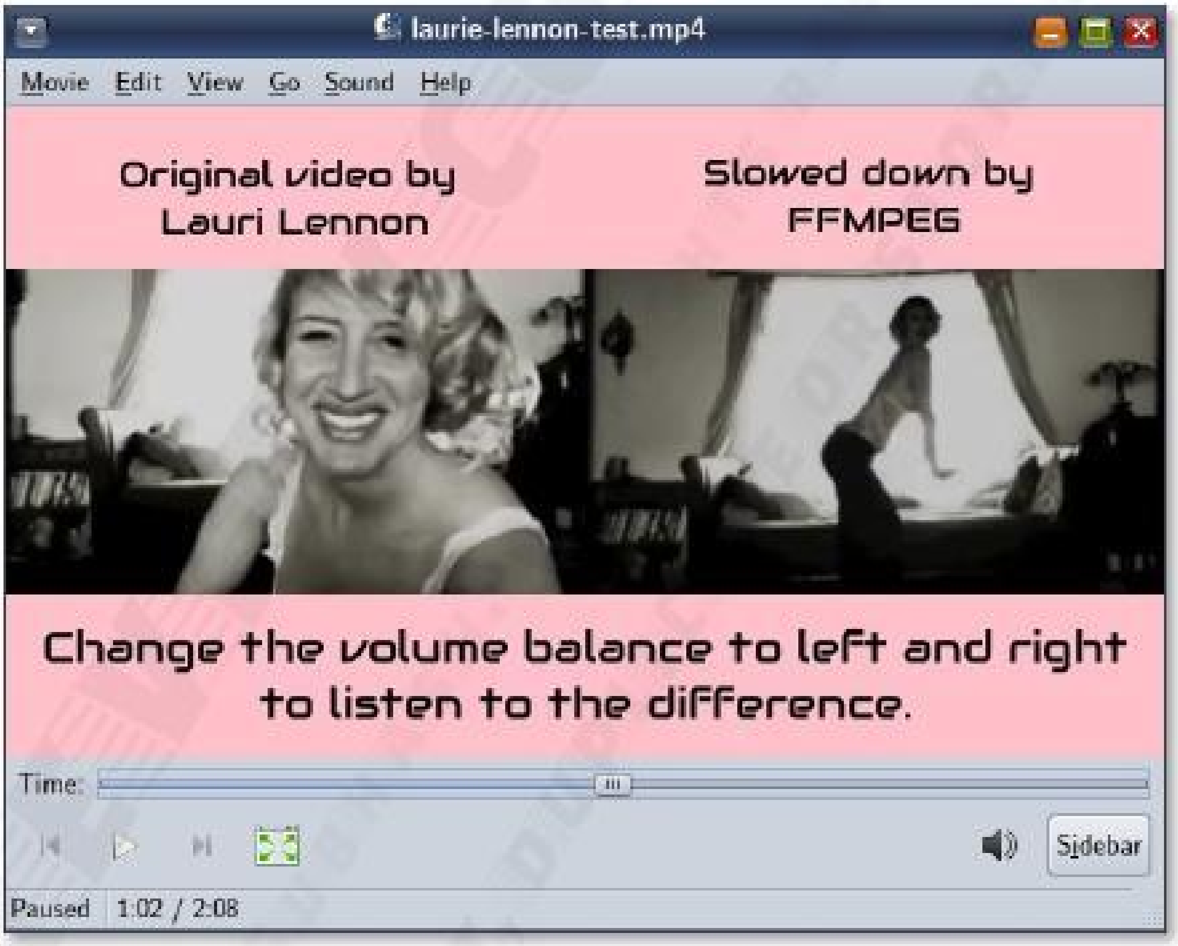
section but the multipliers will have to be different. This command slows down the video by 4x.

```
ffmpeg -y -i tom.mp4 \
  -filter_complex \
    "[0:v]setpts=PTS*4[v];
     [0:a]atempo=0.5[a1];[a1]atempo=0.5[a2]" \
  -map '[v]' -map '[a2]' \
  possessed-doll.mp4
```

In the *Tom & Jerry* video *Baby Puss*, one of the alley cats tries to dance with a doll. In the middle of it, the doll seems to become animated (!) and throws the cat overhead! Oh, the humanity! For a split-second, even the cat looks surprised. Even more creepily, the doll becomes aware of this and slumps back like a normal doll. Anybody can observe it if you slow down the video. It is a possessed doll!

Laurie Lennon, from the famed Lennon Sisters family, had published a tribute video for the *Merrie Melodies* number "*Oh, Wolfie!*". The song had high tempo and some years ago I slowed it down in Audacity. For this book, I tried to do the same using FFMPEG. My calculation became easier when I used seconds. The original video was 114 seconds and my slowed-down audio was 128 seconds.

```
# 128/114 = 1.1228 and 114/128 = 0.8906
ffmpeg -y -i Laurie-Lennon-Original.mp4 \
    -filter_complex \
    "[0:v]setpts=PTS*1.1228[v];
     [0:a]atempo=0.8906[a]" \
  -map '[v]' -map '[a]' \
  Laurie-Lennon-Slow.mp4
```



Original video by Lauri Lennon

Slowed down by FFMPEG

Change the volume balance to left and right to listen to the difference.

All About Audio

While it is convenient to have a separate chapter just for audio, you will find some information repeated.

Convert from one audio format to another

```
ffmpeg -i alarm.ogg \
       -c:a libmp3lame \
       -ac 2 \
       -b:a 128K \
       alarm.mp3              # Ogg to MP3
```

Extract audio from a video

```
ffmpeg -i music-video.mp4 \
       -vn \
       -c:a libmp3lame \
       -ac 2 \
       -b:a 128K \
       music-video.mp3      #Video audio saved as MP3
```

Convert a MIDI file to MP3 or Ogg

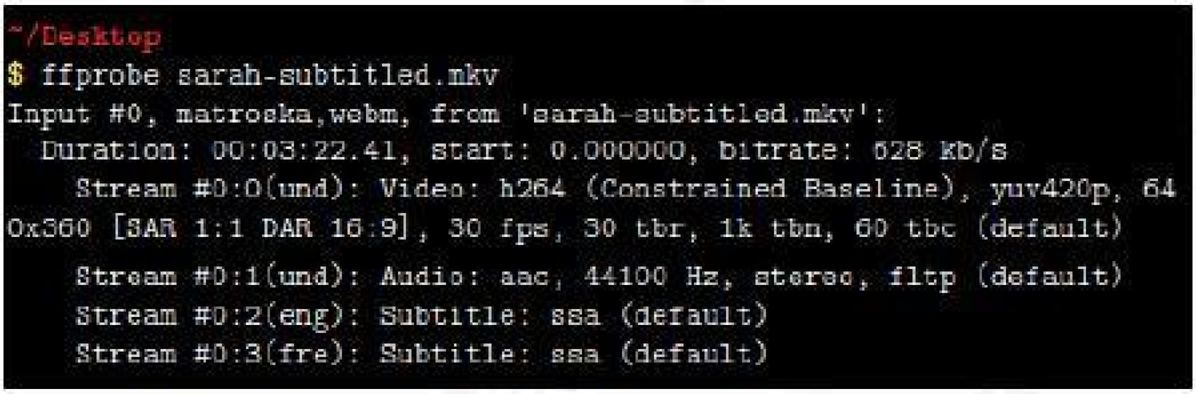
You may have noted that there are no codecs for MIDI. That is because MIDI files are quite different from ordinary sound files. Ordinary sound files contain the wave form in a predefined format. In contrast, MIDI files are merely a collection of references to a common sound bank. For this reason, it is different from other audio files.

Timidity is the Linux way of playing MIDI files. The MIDI playback can be output as WAVE files to the standard output and FFMPEG can be made to capture it.

```
timidity yamaha.midi -Ow -o - | ffmpeg -i - -b:a 128k \
                                         yamaha.ogg
```

The `-Ow` makes Timidity to output the playback in WAVE format. The `-o` option is used to specify the output file. Instead of an output file, we use `-` to make it output to the standard output. This output is then piped to an FFMPEG command, where it is captured as an input file with yet another `-`.

```
ffprobe sarah-subtitled.mkv
```



```
~/Desktop
$ ffprobe sarah-subtitled.mkv
Input #0, matroska,webm, from 'sarah-subtitled.mkv':
  Duration: 00:03:22.41, start: 0.000000, bitrate: 628 kb/s
    Stream #0:0(und): Video: h264 (Constrained Baseline), yuv420p, 64
0x360 [SAR 1:1 DAR 16:9], 30 fps, 30 tbr, 1k tbn, 60 tbc (default)
    Stream #0:1(und): Audio: aac, 44100 Hz, stereo, fltp (default)
    Stream #0:2(eng): Subtitle: ssa (default)
    Stream #0:3(fre): Subtitle: ssa (default)
```

If it has only one subtitle stream, you can extract it using FFMPEG by specifying the correct format.

```
ffmpeg -i sarah-subtitled.mkv \
       -vn -an -scodec ssa \
       subtitle.ass
```

If the video has multiple subtitle streams, you need to specify mapping. This commands saves the second subtitle stream in the input file to an SSA file.

```
ffmpeg -i sarah-subtitled.mkv \
       -vn -an -scodec ssa \
       -map 0:s:1 \
       second-subtitle.ass
```

Extract subtitles from a DVD

The files in a DVD are usually encrypted or obfuscated to prevent bootlegging. There are several free DVD-ripping applications that will decrypt the DVD and extract VOB and subtitle files. Forcing FFPROBE to find subtitle streams on big VOB files is not worth it.

About the Substation Alpha (SSA/ASS) subtitle format

Although SRT is the popular subtitle format, I prefer the Substation Alpha (.ass or .ssa) because it supports fonts. You can convert SRT to SSA using FFMPEG.

```
ffmpeg -i duffy.srt duffy.ass
```

However, I prefer not to do that. I open the SRT file manually in a GUI program called Gnome Subtitles and save it as a SSA file. After this, I run a BASH script on the .ass file to change its style statement. The style statement generated by FFMPEG and Gnome Subtitles refer to Windows fonts. These fonts are not available in Linux and the resultant subtitles look ugly. My script uses a better style statement with a custom font.

FFMPEG's version:

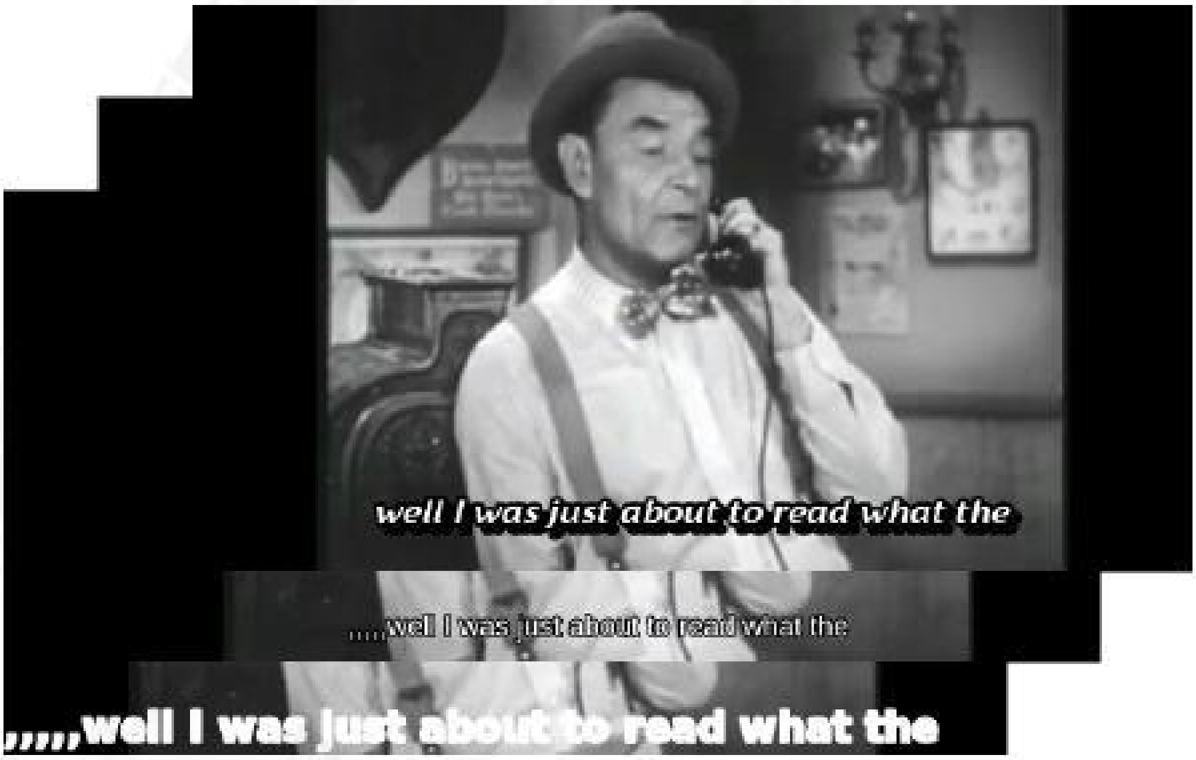
```
Style: Default,Arial,16,&Hffffff,&Hffffff,&H0,&H0,0,0,0,1,1
,0,2,10,10,10,0,0
```

Gnome Subtitles version:

```
Style: Default,Tahoma,24,&H00FFFFFF,&H00FFFFFF,&H00FFFFFF,&
H00C0C0C0,-1,0,0,0,100,100,0,0.00,1,2,3,2,20,20,20,1
```

My version:

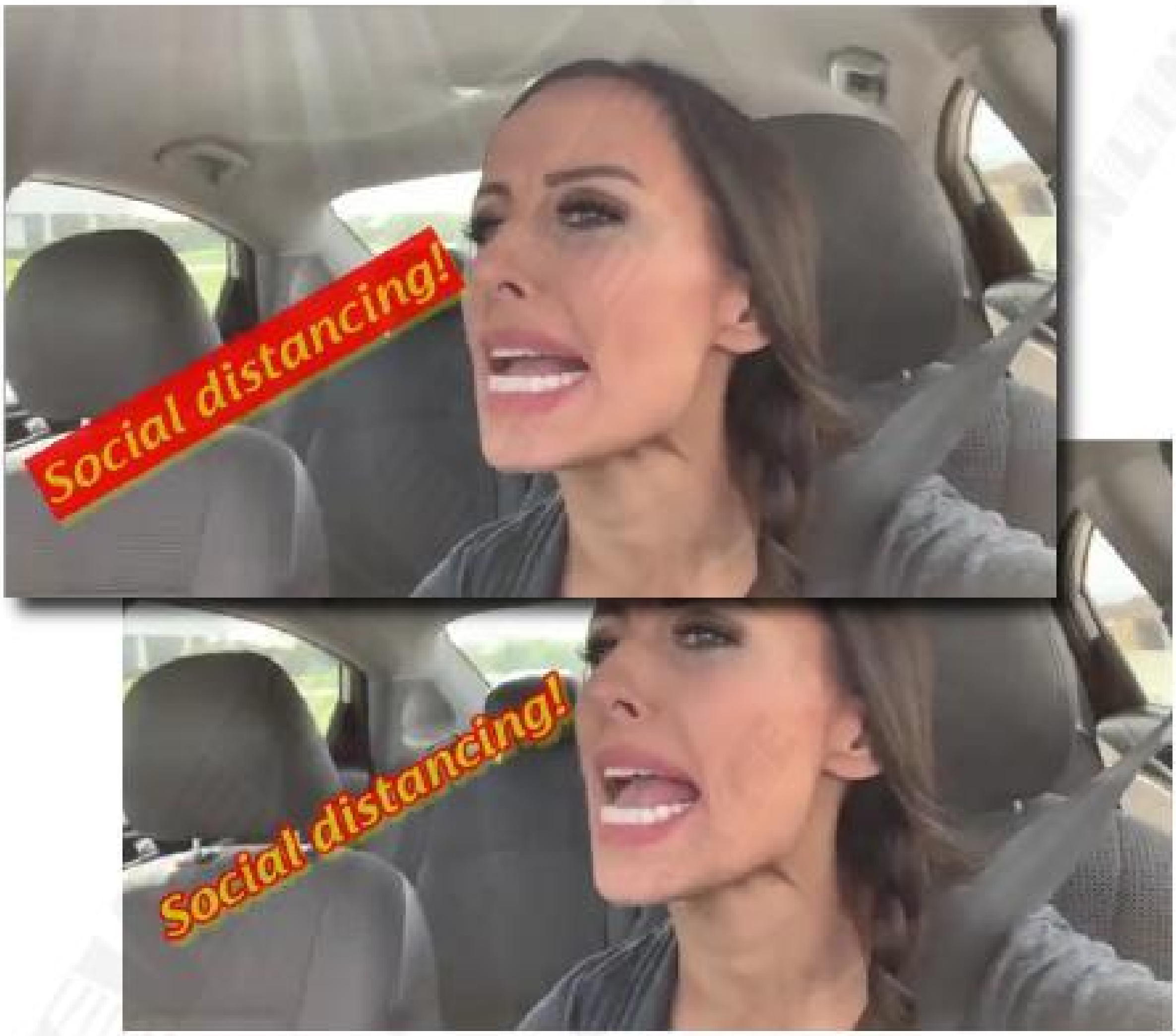
```
Style: Default,Florentia,20,&H00FFFFFF,&H000000FF,&H0000000
0,&AAFFFFFF,-1,-1,0,0,100,100,0,0.00,1,2,4,2,20,20,20,1
```


well I was just about to read what the
,,,,,well I was just about to read what the
,,,,,well I was just about to read what the

The specification of the wonderfully useful but screwed-up SSA format is available on the **https://www.matroska.org** website (*Technical Info » Subtitles » SSA*). However, I will risk a description here for the style statement.

```
Style: Name, Fontname, Fontsize, PrimaryColour, SecondaryCo
lour, OutlineColour, BackColour, Bold, Italic, Underline, S
trikeOut, ScaleX, ScaleY, Spacing, Angle, BorderStyle, Outl
```

```
ine, Shadow, Alignment, MarginL, MarginR, MarginV, Encoding
```



`Name` refers to a subtitle display style. You can define and use many different styles, not just the `Default`. The colours are in hexadecimal AABBGGRR format. `PrimaryColour` is the colour of the subtitle text. `OutlineColour` is for the outline of the text. `BackColour` is the colour of the shadow behind the text. `SecondaryColour` and `OutlineColour` may be used when timestamps collide. `Bold`, `italic` et al are -1 for true and 0 for false. (Yeah, I know.) `ScaleX` and `ScaleY` specify magnification (1-100). `Spacing` is additional pixel space between letters. `Angle` is about rotation (0-360) and controlled by `Alignment`. `BorderStyle` uses 1 (outline and drop-shadow) and 3 (outline box). If `BorderStyle` is 1, then `Outline` represents pixel space width (0-4) of its outline. In the same case, `Shadow` represents pixel space (0-4) below the text and shadow. `Alignment` takes 1 (left), 2 (center) and 3 (right). If you add 4 to them, the subtitle appears at the top of the screen. If you add 8, it goes to the middle. Then, we have margin from the left, right and bottom edges of the screen. `Encoding` is 0 for ANSI Latin and 1 for Unicode (I think).

```
Style: Default,Florentia,30,&H2200CCCC,&H000000FF,&H220000E
E,&HAA00CCCC,-1,-1,0,0,100,100,0,30.00,3,2,3,1,20,20,40,1
```

Burn subtitles into a video

After I have the SSA subtitle file modified, I usually create a MKV file with the file and a custom font added as additional streams. Sometimes, however, I burn the subtitles into the video because my WD HD media player uses a very small built-in font (with no outline or shadow) to display subtitles.

```
ffmpeg -y -i duffy.mp4 \
       -filter_complex "subtitles=duffy.ass" \
       -acodec copy \
       -t 0:1:0  \
       duffys.mp4
```

☞ Once burned into a video, the subtitles cannot be removed or turned off. They become part of the video.

All About Metadata

Metadata is data about data. In audio and video files, metadata can include information such as title, artist, album, subject, genre, year, copyright, producer, software creator, comments, lyrics and even album-art images. Not all media formats support all of these metadata.

An audio or video file can have global metadata (that is, at the file-level) and also stream-specific metadata too. You can use `ffprobe` and `ffmpeg -i` commands to display any metadata that a file has.

Set MP3 tags

MP3 tags are specified in the ID3v2 standard.

```
ffmpeg -i lucas.mp3 \
       -metadata title="Oh, my goootness" \
       -metadata artist="Lucas" \
       -metadata subject="Troomp" \
       -metadata album="Upload" \
       -metadata date="2020-12-02" \
       -metadata genre="Speech" \
       -metadata comments="Tell me what you think" \
       -id3v2_version 3 \
       -codec copy \
       lucast.mp3
```

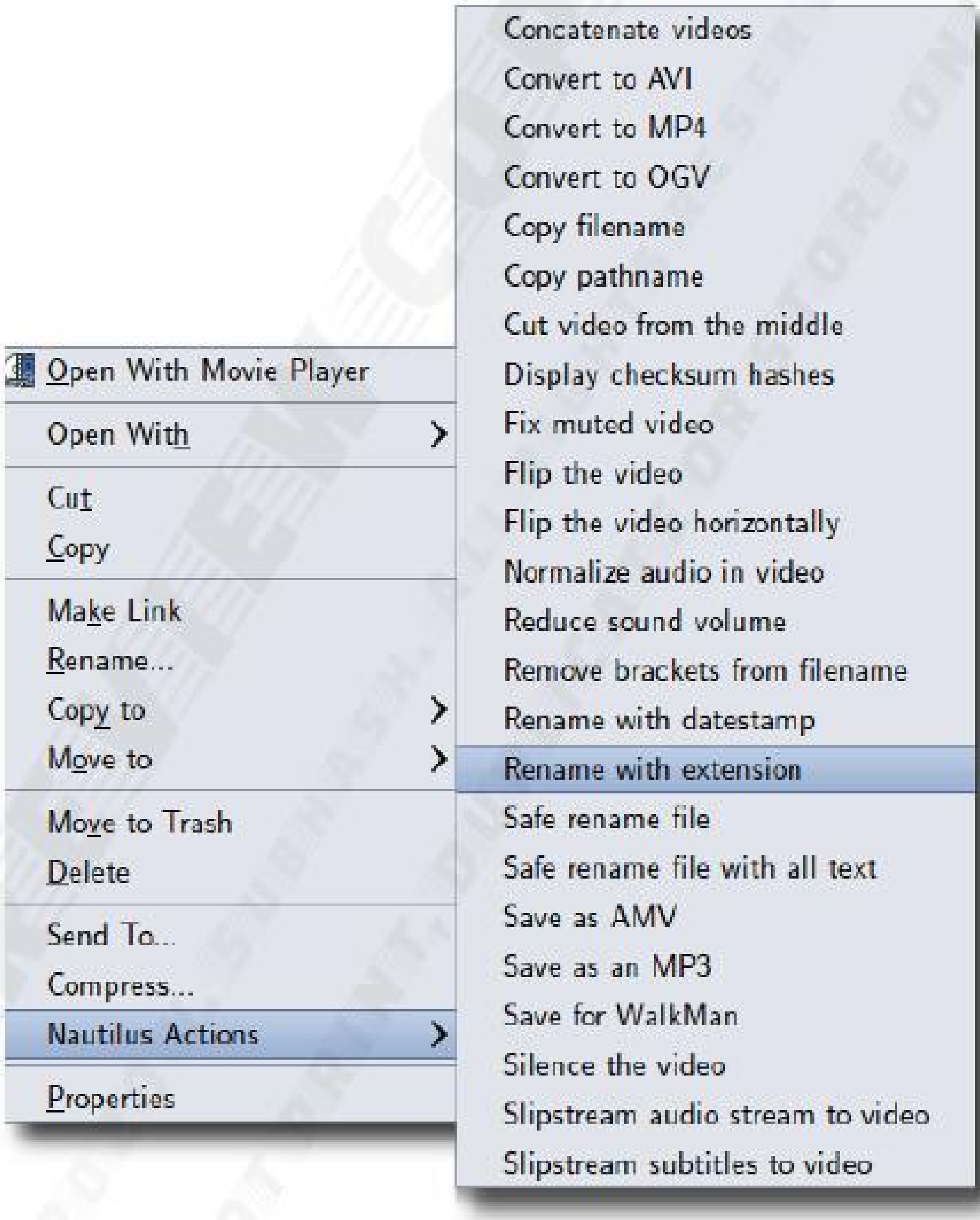
Add album art to MP3

An album art image is stored in the file as a video stream. You can add an album art like this:

```
ffmpeg -y \
  -i duffy.mp3 -i archie.png -i duffys-tavern.jpg  \
  -map 0 -map 1 -map 2 \
  -metadata:s:1 title="Archie.png" \
  -metadata:s:1 comment="Cover (front)" \
  -metadata:s:2 title="DuffysTavern.jpg" \
  -metadata:s:2 comment="Cover (back)" \
  -codec copy \
  -f mp3 \
  duffys.mp3
```

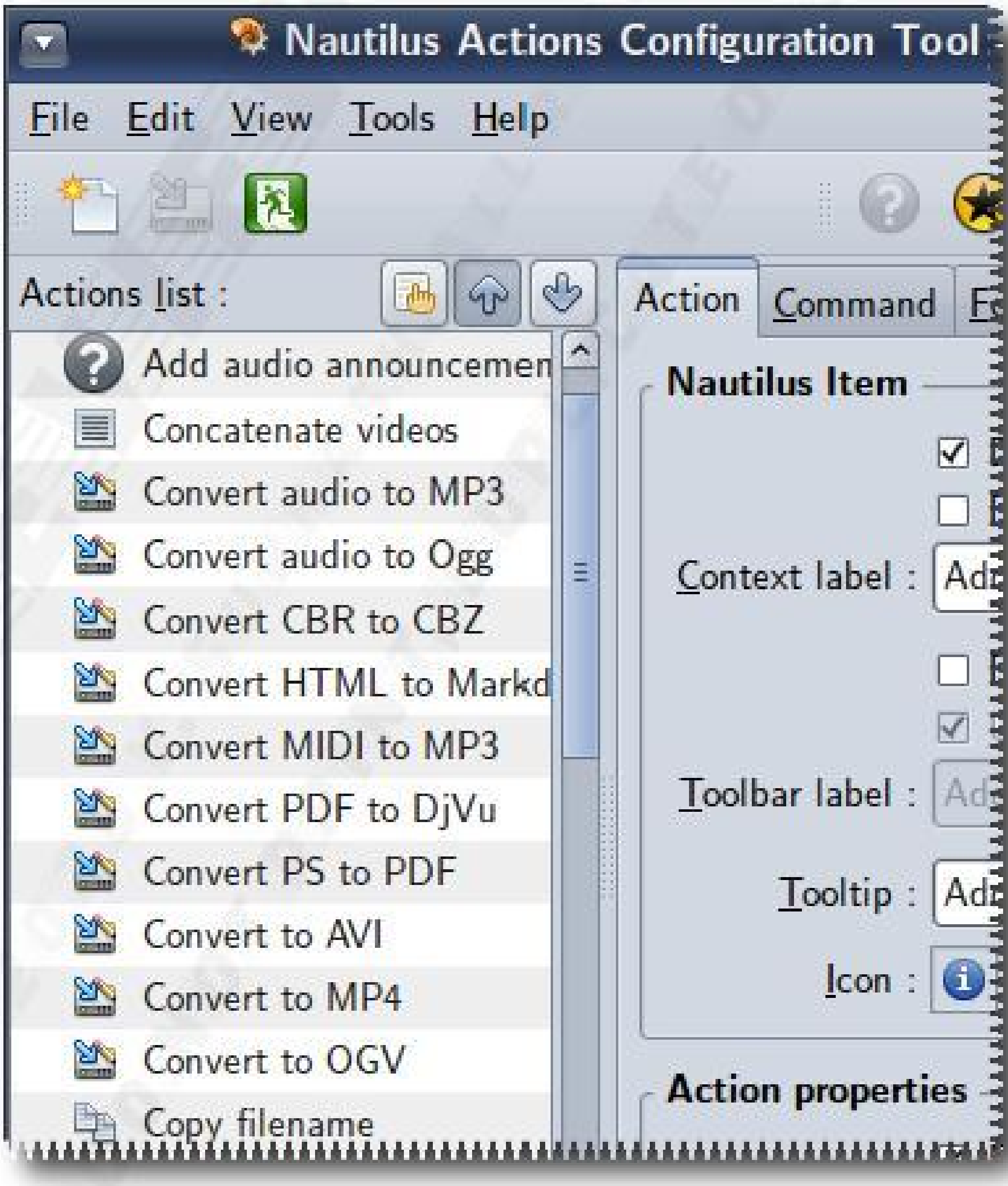
Tips

File manager automation



The first rule of software development is 'Give what the user wants'. This rule implies another thing - 'Do not give what the user does not want'. But, it was not to be... When the Chrome browser was released, it lacked options to change important settings because (it was alleged) users allegedly did not know about them or need them. This was no accident. For some years, there was a shift away from 'More power to users' to 'The user is stupid'. In the open-source community, Gnome 3 desktop environment project became plagued with this attitude. They decided that users **should not be allowed** to customise or personalise a desktop that the developers had carefully designed and curated. (Customisation and changing default settings was compared to the tragedy that was known as Myspace personal pages.) Like Fascists, they said users did not need any extra features. Windows 8 developers also justified the destruction of their desktop in similar terms. They removed features and claimed it was an improvement.

Fortunately, not all developers succumbed to this malady. Some of them developed the Mate ('ma-tay') desktop environment that continued support for the intuitive and user-friendly Gnome 2.



[illegible]

context menu options to *Nautilus*, the default file and desktop manager. In Gnome 3, Nautilus lost most of its features and became frustratingly useless. Thankfully, however, the **Mate desktop project** continued support for Nautilus and renamed it as Caja. Caja has a project that provides the same functionality of Nautilus Actions Configuration. It is called **Caja Actions Configuration**.

Now, stop composing long FFMPEG poems each time you want something done with it. Switch to Ma-Tay first. And, then use Caja Actions Configurations.

Add an `espeak` intro to you audio file

I have lots of vintage radio shows as MP3 files. My **DIY boombox** does not display any tags or filenames - only numbers. So, I use `espeak` to read out the file name, save it as a wave file and prefix it to the mp3 file. This way, I know what is the title name and show name when a new MP3 file is played on the boombox.

```
cd "$1" // Moves to a directory with MP3 files

for sFile in *.mp3
do
  sFileName=${sFile%.*}
  espeak -w "$sFileName.wav" "$sFileName"

  ffmpeg -i "$sFileName.wav" -i "$sFile" \
      -filter_complex \
        "[0:a:0][1:a:0]concat=n=2:v=0:a=1[outa]" \
      -map_metadata 1 -id3v2_version 3 \
      -map "[outa]" \
      -c:a libmp3lame  \
      "$sFileName [ann].mp3"

  rm "$sFileName.wav"
done
```

This script is useful if you have neatly tagged and renamed your MP3 files. Although FFMPEG can be used to tag MP3 ile, I prefer to use *EasyTAG* instead. I also use *pyRenamer* to create meaningfully named files.

Best MP4 (H.264 or MPEG-4) conversion settings

This is from the FFMPEG FAQ.

```
-mbd rd -flags +mv4+aic -trellis 2 -cmp 2 -subcmp 2 -g 300
```

Best VCD (MPEG-1) and DVD (MPEG-2) conversion settings

This is also from the FFMPEG FAQ.

```
-mbd rd -trellis 2 -cmp 2 -subcmp 2 -g 100
```

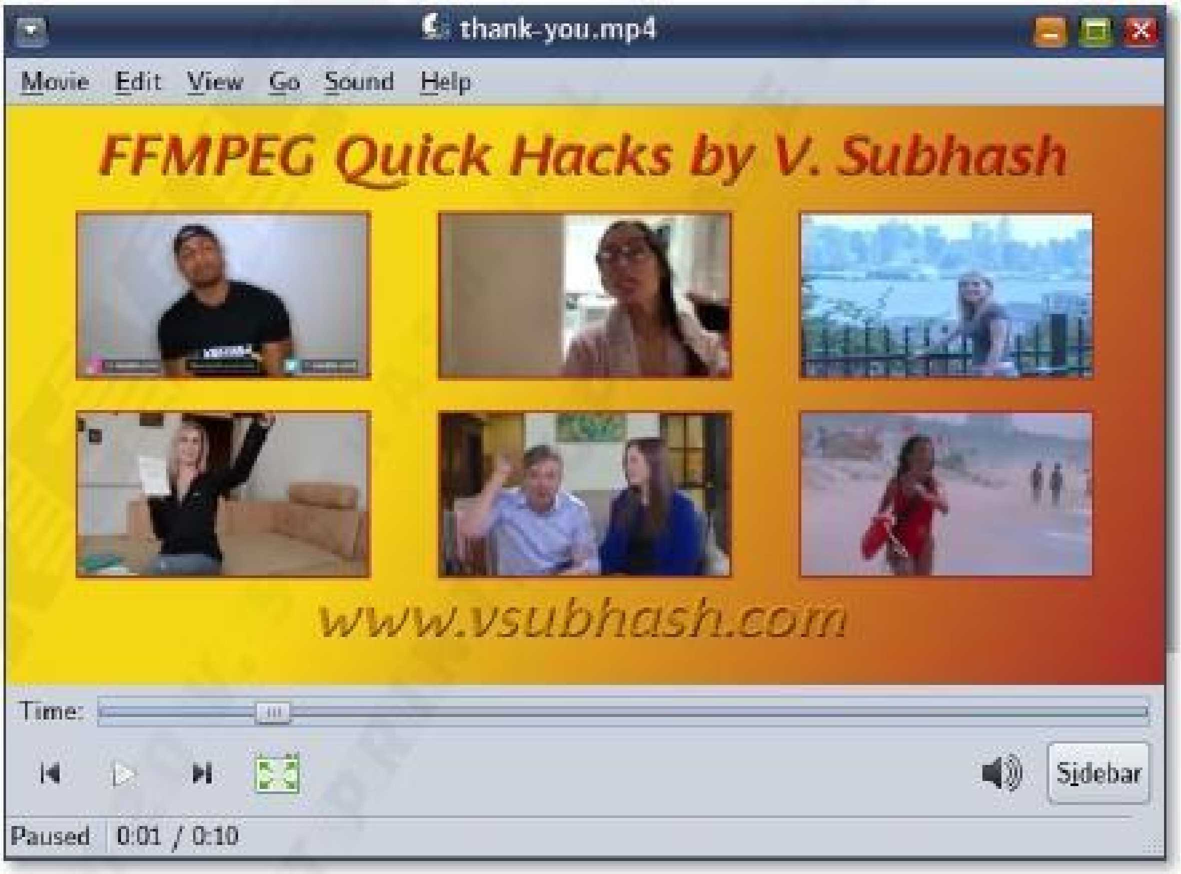
Best MP3 (MPEG 2 Audio Layer 3) conversion settings

MP3 is a lossy compression scheme for audio. When the bitrate is higher, the loss is minimized. This applies only when the source is of high quality. However, most people are unable to discern the difference between a song encoded at bitrates equal to one-eigth of the width of the screen. For a 1280x720 video, I would choose a bitrate of 1024 kbps. For a 640x480 video, I would choose 512 kbps. (I have what is known as a 'near-HD' TV and this ratio is good enough for me.) FFMPEG presets are much more generous than this.

On my system, OGV encoding has the best quality. WebM seems to have the best compression. YouTube allows WebM uploads. My internet connection is very poor so I use WebM for uploading to that site. On other sites, MP4 seems to have the most acceptance.

Thank you

This video has six downscaled videos playing simultaneously on a background image. The audio from the six input files was also downmixed to stereo. Even the text banners on the image was written using FFMPEG.


thank-you.mp4
Movie Edit View Go Sound Help
FFMPEG Quick Hacks by V. Subhash
www.vsubhash.com
Time:
Sidebar
Paused 0:01 / 0:10

```
ffmpeg -y \
  -f image2 -loop 1 -pix_fmt yuv420p -i BG-Collage.png \
  -f mp4 -i lucas.mp4 -i sara.mp4 -i jokingdotcom.mp4 -i \
            barbara.mp4 -i fallon.mp4 -i baywatch.mpg \
  -filter_complex \
    "[0:v:0]drawtext=x=50:y=15:fontcolor=red:alpha=0.6:shad
```

```
owx=1:shadowy=2:text='FFMPEG Quick Hacks by V. Subhash':fon
tsize=30:fontfile=Florentia.ttf[banner1];

     [banner1]drawtext=x=170:y=270:fontcolor=white:alpha=0.
6:shadowx=1:shadowy=2:text='www.vsubhash.com':fontsize=30:f
ontfile=Florentia.ttf[banner];

     [1:v:0]scale=160:90[scale1];
         [banner][scale1]overlay=40:60[over1];
     [2:v:0]scale=160:90[scale2];
         [over1][scale2]overlay=240:60[over2];
     [3:v:0]scale=160:90[scale3];
         [over2][scale3]overlay=440:60[over3];
     [4:v:0]scale=160:90[scale4];
         [over3][scale4]overlay=40:170[over4];
     [5:v:0]scale=160:90[scale5];
         [over4][scale5]overlay=240:170[over5];
     [6:v:0]scale=160:90[scale6];
         [over5][scale6]overlay=440:170[video];

     [1:a:0][2:a:0][3:a:0][4:a:0][5:a:0][6:a:0]amerge=input
s=6[audio]" \
  -map '[video]' -map '[audio]' \
  -ac 2 \
  -t 0:0:10 \
  thank-you.mp4; totem thank-you.mp4
```

Finally, you have reached the end of this book. I hope it was useful to you. Have a fantastic time with FFMPEG. And, spread the word about this book, FFMPEG, Linux and the free software movement. Bye.

FFMPEG Quick Hacks by V. Subhash

About FFMPEG

FFMPEG is the ultimate command-line tool to edit, enhance and convert video files. It is a FREEly downloadable program for Linux, Mac and Windows. FFMPEG is very easy to use and does not require a lot of multimedia expertise. Most users find it versatile and sophisticated enough for their needs.

About *FFMPEG Quick Hacks*

FFMPEG Quick Hacks is a novel attempt to bring this great software program to the masses. This book uses powerful learning cues to make concepts clear and interesting to the reader. It is also useful as a good desk-side reference.

What to expect from *FFMPEG Quick Hacks*

☞ A simple introduction to FFMPEG and related multimedia concepts - containers, streams, channels, maps, metadata…
☞ Learn to convert from one format to another - video to video, video to audio, video to image, image to video, audio to video…
☞ Get ready to edit video - cut videos with and without re-encoding, append (concatenate) videos, resize videos, change contrast,…
☞ Go bonkers with filters - rotate, flip, crop, side-by-side split, PIP inset, remove logo, blur, smoothen/sharpen, draw box, draw text, speed up, slow down, fade in/out…
☞ Get all in with audio - convert, change volume, mix channels, detect silence, display waveforms…
☞ Go subversive with subtitles - place them anywhere on the screen, use custom fonts and colors, specify languages, burn them into the video…
☞ Get mental with metadata - add MP3 tags including album art, set global and stream-specific metadata, remove metadata…
☞ Learn several useful tips that makes tough tasks easy

What not to expect in *FFMPEG Quick Hacks*

☞ Information about FFStream
☞ Information about FFMPEG, the software library
☞ High-level concepts and in-depth information on multimedia formats
☞ Substitute for online documentation