

What is ffmpeg?

Why use ffmpeg over pre-made programs?

My Workflows

VHS Tapes

1. Rip VHS to AVI via VirtualDub
2. Upscale, resample, and deinterlace raw file in VirtualDub (Break out large video into separate pieces here using batch operations)
3. Compress video(s) with Handbrake
4. Touch up file(s) with ffmpeg commands

https://www.youtube.com/watch?v=sn_TDa9zY1c

Mini DV Tapes

- Use WinDV for ingestion. It will break out each stop/start on the tape into new files.
- Touch up for YT spec with all inclusive script, if audio is off, switch to mono.

DVDs

1. Use Handbrake
2. Touch up files with ffmpeg commands if home videos

Blu-Rays

1. Use MakeMKV
2. If looking to compress, use Handbrake post-MakeMKV

Useful Commands & Scripts

<https://ffmpeg.org/ffmpeg.html#Description>

<http://ankitshah009.blogspot.com/2015/02/useful-ffmpeg-commands.html>

Template for YouTube Specs

<https://support.google.com/youtube/answer/1722171>

<https://support.google.com/youtube/answer/4603579?hl=en>

```
ffmpeg -i input -c:v libx264 -preset slow -profile:v high -crf 18 -coder 1 -
pix_fmt yuv420p -movflags +faststart -g 30 -bf 2 -c:a aac -b:a 384k -profile:a
aac_low output
```

Scripting

Bash Scripting

When building out bash scripting, you will need to run your ffmpeg commands with `-nostdin` if you are piping in variables with paths¹⁾. ffmpeg uses stdin for its commands, which will break your 'while read' commands.

`-stdin`



Enable interaction on standard input. On by default unless standard input is used as an input. To explicitly disable interaction you need to specify `-nostdin`. Disabling interaction on standard input is useful, for example, if ffmpeg is in the background process group. Roughly the same result can be achieved with `ffmpeg ... < /dev/null` but it requires a shell.

Batch Filing

```
for %a in (*.fileformat) do //insert ffmpeg command here//
```

Instead of the filepath or filename in your ffmpeg command use these shortcuts:

```
For %%A in ("%filename%") do (
  echo full path: %%~fA
  echo drive: %%~dA
  echo path: %%~pA
  echo file name only: %%~nA
  echo extension only: %%~xA
  echo expanded path with short names: %%~sA
  echo attributes: %%~aA
  echo date and time: %%~tA
  echo size: %%~zA
  echo drive + path: %%~dpA
```

```
echo name.ext: %%~nxA  
echo full path + short name: %%~fsA)
```

So, for example, this script would take every m4v file inside of a folder, offset the audio back 1/4 of a second, and output it as a copy of the original file while keeping the same filename:

```
for %%a in (*.m4v) do filepath-to-ffmpeg.exe -i "%%a" -itsoffset -0.25 -i  
"%%a" -map 0:v -map 1:a -c copy "filepath-to-other-folder\%%~na.m4v"
```

<https://forum.videohelp.com/threads/356314-How-to-batch-convert-multiplex-any-files-with-ffmpeg>
<https://stackoverflow.com/questions/9252980/how-to-split-the-filename-from-a-full-path-in-batch>

Powershell Coding

Frequent Commands

<https://www.videoproc.com/resource/ffmpeg-commands.htm>

Listing out features

<https://dev.to/drsensor/list-of-helpful-ffmpeg-command-for-checking-capabilities-1110>

Watermarking

<http://ksloan.net/watermarking-videos-from-the-command-line-using-ffmpeg-filters/>

Example using Batch Script:

```
for %%a IN (foldername) DO ffmpeg -i %%a -i "Watermark.png" -filter_complex  
"overlay=0:0" -c:v copy -c:a copy outputfolder\%%a
```

Example using Powershell:

```
gci "videofilefolder" | foreach-object {ffmpeg -i $_.FullName -i  
"Watermark.png" -filter_complex "overlay=0:0" -c:v copy -c:a copy  
"outputfolder\$_"}>
```

Cropping a video

- <https://stackoverflow.com/questions/18444194/cutting-the-videos-based-on-start-and-end-time-using-ffmpeg>
- <https://unix.stackexchange.com/questions/1670/how-can-i-use-ffmpeg-to-split-mpeg-video-into-10->

minute-chunks

Example using Powershell:

```
ffmpeg -ss 00:53:00 -i 'input' -avoid_negative_ts 1 -to 07:19:17 -c copy  
'output'
```

Caveats to cropping

When you do cut with -c copy (no re-encode), and the cut point is not exactly on a keyframe, and/or the video uses B-frames (re-ordered frames), ffmpeg can't rewrite frames to make a clean boundary. It copies packets but the decode/display timestamps (DTS/PTS) can become inconsistent around the cut boundary. Then when you concat, ffmpeg sees timestamps going "backwards" and prints: "Non-monotonic DTS... This may result in incorrect timestamps."

Players often still play linearly, but seeking near the splice can fail hard (especially around the crop point). mp4 is particularly sensitive because it stores timing/indexing in a way that makes sloppy cuts more painful than in some other workflows.

Workaround is to use a copy-cut and regenerate timestamps

```
ffmpeg -ss 0 -to 02:03:12 -i ogvideo.mp4 \  
-map 0 -c copy \  
-fflags +genpts -avoid_negative_ts make_zero \  
croppedvid.mp4
```

Then, "remux normalize" the video to give clean metadata. This is a nice to do on all the videos that you're working on if concat-ing them together

```
ffmpeg -i croppedvid.mp4 -map 0 -c copy -movflags +faststart fixed1.mp4
```

Joining video files together (Concatenate)

<https://trac.ffmpeg.org/wiki/Concatenate>

<https://stackoverflow.com/questions/7333232/how-to-concatenate-two-mp4-files-using-ffmpeg>

<https://stackoverflow.com/questions/18474312/concatenate-join-mp4-files-using-ffmpeg-and-windows-command-line-batch-not-lin>

Make a text file in the same directory. Each video needs the word 'file' at the beginning of the line to delineate.

```
file videofile.mp4  
file videofile2.mp4
```

Here's how I've scripted this. Note, this pertains more to macOS with the sed statement since the single

quotes are needed, but the tweak for linux is to just remove the single quotes and the -e

```
find . -type f > concat.txt && sed -i '' -e '/^\.\.\.DS_Store$/d' -e  
'/^\.\./concat\.txt$/d' -e 's/^\.\.///' -e 's/^/file /' concat.txt
```

Then, reference the text file as your input.

```
ffmpeg -f concat -safe 0 -i concat.txt -c copy output.mp4
```

Splitting a large video into smaller clips

<https://trac.ffmpeg.org/wiki/Seeking#Cuttingsmallsections>

Example in Batch:

```
for %i in (*.mkv) do  
ffmpeg -i video.mp4 -ss 00:00:00.000 -to 00:01:00.251 -c copy clip1.mp4  
ffmpeg -i video.mp4 -ss 00:01:01.251 -to 00:03:30.684 -c copy clip2.mp4  
ffmpeg -i video.mp4 -ss 00:05:00.112 -to 00:07:00.254 -c copy clip3.mp4  
pause
```

Moving Audio Offset (adjusting audio delay)

-itsoffset x.xx time in seconds. so, 0.25 is 1/4 of a second.

Example for adjusting audio, and compressing video:

```
#Adjusts audio and compresses file to output of your choice  
ffmpeg -i 'input' -itsoffset 1.00 -i 'input' -map 0:v -map 1:a 'output'
```

Example for copying video only:

```
#Adjust audio, but just copy the file instead of compressing. Can not use an  
MKV container for this.  
ffmpeg -i 'input' -itsoffset 0.25 -i 'input' -map 0:v -map 1:a -c copy  
'output'
```

Muting specific spots in your video

Example from the link. This mutes audio at the 59 second mark down to 0 and then back up at 1:01

```
ffmpeg -i input -vcodec copy -af "volume=enable='between(t,59,61)':volume=0"  
output
```

<https://stackoverflow.com/questions/29215197/mute-specified-sections-of-an-audio-file-using-ffmpeg>

When entering your time, it must be in seconds only. If you have a very long video, be sure to calculate out the exact seconds and enter that in. Example, four hours is 14400 seconds.

Fading out audio in a specific spot

<https://dev.to/dak425/add-fade-in-and-fade-out-effects-with-ffmpeg-2bj7>

Use `afade` for audio, `fade` for video

- `t=in` → in point
- `t=out` → out point
- `st=xx` → start at xx (seconds into video)
- `d=x` → duration of fade (in seconds)

Below is an example of fading in audio at the 59 second mark, in 1 second, and fading out at 61 seconds for a second. Video remains un-encoded.

```
ffmpeg -i video.mp4 -vcodec copy -af  
"afade=t=out:st=59:d=1,afade=t=in:st=61:d=1" out.mp4
```

Adjusting audio to broadcast spec

https://www.reddit.com/r/ffmpeg/comments/b2ygzl/increase_volume_on_multiple_files/

Converting from one file type to another

Issues that you may run into

Trying to adjust MKV files. Safe bet, do all adjustments to MP4 file first, then MKV when permanent. -

<https://superuser.com/questions/1411133/ffmpeg-invalid-length-0x59f0-0x8bd7fe90-in-parent-when-trying-to-do-anything>

Limiting Cores/Threads

<https://superuser.com/questions/155305/how-many-threads-does-ffmpeg-use-by-default>

Hardware Acceleration

<https://trac.ffmpeg.org/wiki/HWAccelIntro> <https://www.rickmakes.com/ffmpeg-notes/>

Each OS has its own Platform APIs of hardware acceleration and additionally, Intel, AMD, and Nvidia GPUs have their own hardware acceleration protocols.

Nvidia on Windows/Linux has h264_nvenc, hevc_nvenc(h265), and av1_nvenc Intel on Windows/Linux has h264_qsv, hevc_qsv, etc...

macOS supports VideoToolBox for Intel QuickSync, such as h264_videotoolbox, hevc_videotoolbox, and prores_videotoolbox

Unorganized Links

<https://dev.to/drsensor/list-of-helpful-ffmpeg-command-for-checking-capabilities-1110>

<https://stackoverflow.com/questions/68977300/ffmpeg-more-than-1000-frames-duplicated-and-input-height-doesnt-match>

1)

<https://stackoverflow.com/questions/21634088/execute-ffmpeg-command-in-a-loop>

From:

<https://wiki.jonathandurand.com/> - **Jon's Repository of Info**

Permanent link:

<https://wiki.jonathandurand.com/doku.php?id=computers:ffmpeg>

Last update: **2025/12/15 16:40**

