superuser

Using ffmpeg to cut up video

Asked 13 years, 7 months ago Modified 1 year, 1 month ago Viewed 799k times

580

I am using `ffmpeg` to cut out a section of a large file like this:

```
ffmpeg -i input.wmv -ss 60 -t 60 -acodec copy -vcodec copy output.wmv
```

The `-ss` part works fine but the `-t` is ignored. It correctly removes the initial specified seconds specified with `-ss` but then keeps going to the end of the input with the copy.

Is there a way to use `ffmpeg` to cut off the end of a video without recoding it?

command-line video ffmpeg processing

Share Improve this question Follow

edited Feb 14, 2021 at 9:27
Hastur
18.9k • 9 • 53 • 98

asked May 6, 2010 at 14:12
Neil
5,929 • 3 • 16 • 3

9 Answers

Sorted by: Highest score (default)

669

You can use the `-ss` option to specify a start timestamp, and the `-t` option to specify the encoding duration.

The following would skip the first 30 seconds, and then extract the next 10 seconds to a file called `output.wmv`:

```
ffmpeg -ss 30 -i input.wmv -c copy -t 10 output.wmv
```

In the above command, the timestamps are in seconds (`s.msec`), but timestamps can also be in `HH:MM:SS.xxx` format. The following is equivalent:

```
ffmpeg -ss 00:00:30.0 -i input.wmv -c copy -t 00:00:10.0 output.wmv
```

Note that `-t` is an output option and always needs to be specified *after* `-i`.

Some tips:

- For older ffmpeg versions, if you use `-ss` after `-i`, you get more accurate seeking at the expense of a slower execution altogether. See also: Seeking with FFmpeg
- You can use `-to` instead of `-t` to specify the timestamp to which you want to cut. So, instead of `-i <input> -ss 30 -t 10` you could also do `-i <input> -ss 30 -to 40` to achieve the same thing.
- If your ffmpeg does not support `-c`, or `-to`, it is likely very outdated. Compile a new version yourself or download a static build from their homepage. It's really not complicated.

Share Improve this answer Follow

edited Jul 19, 2022 at 15:40
rgov
599 ● 2 ● 4 ● 17

answered May 15, 2010 at 0:03
user37242

10 @Mondain Actually, you get more accuracy putting the -ss after. And slhck mentions this here blog.superuser.com/2012/02/24/… also in ffmpeg documentation for -ss it mentions a difference between putting it before or after. – barlop Oct 11, 2013 at 0:38

4 From the docs, the part you said about -t being only an output option is incorrect: `When used as an input option (before -i), limit the duration of data read from the input file. When used as an output option (before an output filename), stop writing the output after its duration reaches duration.` – deweydb Aug 30, 2016 at 2:52

@barlop sorry to summon you 10 years after, but I just witnessed the exact opposite: I was trying to cut a video from the beginning (00:00 seconds) and, by a strange astral coincidence of keyframes and PTS, by putting `-ss 00:00:00.0` after the `-i <input.mp4>` the output video was starting a few tenths of a second after the actual zero, while putting it before the `-i <input.mp4>` the output video started precisely at the "true zero". – Avio Oct 24 at 16:11

@Avio I think both the ss before or after -i, are meant to be "frame accurate" now post 2015 post ffmpeg 2.1, ..and the ss before -i is recommended 'cos it's faster. I spoke to an ffmpeg expert and he suggested that you might have a "negative timeframe". I looked that up and it showed stackoverflow.com/questions/41032079/… What if you do `ffprobe -i test.mp4 -show_format` you could see what it says after "start_time=" ? – barlop Oct 26 at 4:41

1 @Avio ah I don't know but If you're curious there is an ffmpeg irc chat on on /server irc.libera.chat and some experts there , you could get an answer there then post it here as a Q and A. – barlop Oct 27 at 13:04

176

As other people mentioned, putting `-ss` before (much faster) or after (more accurate) the `-i` makes a big difference. The section "Fast And Accurate Seeking" on the ffmpeg seek page tells you how to get both, and I have used it, and it makes a big difference. Basically you put `-ss` before AND after the `-i`, just make sure to leave enough time before where you want to start cutting to have another key frame. Example: If you want to make a 1-minute clip, from 9min0sec to 10min 0sec in Video.mp4, you could do it both quickly and accurately using:

```
ffmpeg -ss 00:08:00 -i Video.mp4 -ss 00:01:00 -t 00:01:00 -c copy VideoClip.mp4
```

The first `-ss` seeks fast to (approximately) 8min0sec, and then the second `-ss` seeks accurately to 9min0sec, and the `-t 00:01:00` takes out a 1min0sec clip.

Also note this important point from that page: "If you use `-ss` with `-c:v copy`, the resulting bitstream might end up being choppy, not playable, or out of sync with the audio stream, since ffmpeg is forced to only use/split on i-frames."

This means you need to re-encode the video, even if you want to just copy it, or risk it being choppy and out of sync. You could try just `-c copy` first, but if the video sucks you'll need to re-do it.

Share Improve this answer Follow

edited Aug 25, 2014 at 21:07
gronostaj
56.4k 20 123 180

answered Jan 20, 2014 at 4:07
seriesoftubes
1,861 1 11 3

6 -ss as an input option is both fast and accurate as of FFmpeg 2.1, so there's no need to include it as an output option too: trac.ffmpeg.org/wiki/Seeking – cgenco Jun 19, 2021 at 15:56

4 Is it possible to clarify: This answer says "You could try just `-c copy` first, but if the video sucks you'll need to re-do it." However, only `-c copy` is provided in the answer, there is no discussion of what options to add if the result is not cut correctly. What options should be added? – MRule Sep 16, 2021 at 12:27

1 @MRule if you leave out the `-c copy`, it'll reencode, which is computationally expensive compared to copying, but is more accurate because it doesn't rely on keyframes being the same in the input and output. – BallpointBen Jun 22 at 18:54

Is there a way to use the original timestamps with `-ss` before and after the `-i`? Ex have `ffmpeg -ss 30 -i input.mp4 -ss 40 -to 50 output.mp4` create a clip from 40sec to 50sec in the original vid, but “fast forward” 30 seconds to make it go faster? This way I wouldn't have to do the math to figure out the new timestamps. As written, this command would instead take the clip from 70sec to 80sec. – BallpointBen Oct 18 at 21:30

49

I found that `-ss` combined with `-c copy` resulted in a half-second chop at the start.

To avoid that, you have to remove the `-c copy` (which admittedly will do a transcode).

Share Improve this answer Follow

edited Feb 8, 2015 at 10:29
umläute
409 3 18

answered May 9, 2014 at 2:24
Chris
1,794 3 18 23

4 It added 7 seconds at the beginning of the output when I used `-c copy` with `-ss`. – Phani Rithvij Mar 24, 2020 at 7:23

1 I found when removing the -c option, then there's no need to add a second -ss before -t option. – Harry Apr 30, 2021 at 1:59

15

MANUALLY

Open the file in a media player that will frame by frame advance and play an *AVISynth file* with data such as:

```
DirectShowSource(("C:\Downloads\Video\Do you want him.flv"),
Pixel_Type="yuy2").Crop(0,0,-0,-0)
Subtitle("C:\Downloads\Video\Do you want him.flv", font="Arial", size=24,
text_color=$ff0000, align=3)
ShowFrameNumber(scroll=true, x=336, y=27, font="Arial", size=24, text_color=$ff0000)
ShowTime(x=398, y=44, font="Arial", size=24, text_color=$ff0000)
```

Then cut with the EXACT time format:

```
ffmpeg -i "Path\do you want him.flv"        \
       -ss 00:00:05.240 -to 00:00:08.360    \
       -vcodec libx264 -acodec libvo_aacenc \
       "Path\Do you want him1.flv"
```

and

```
ffmpeg -i "Path\do you want him.flv"        \
       -ss 00:00:10.240 -to 00:00:14.360    \
       -vcodec libx264 -acodec libvo_aacenc \
       "Path\Do you want him2.flv"
```

Now make a txt file with the video files with contents like:

```
file 'C:\Downloads\Video\Do you want him1.flv'
file 'C:\Downloads\Video\Do you want him2.flv'
```

Run ffmpeg:

```
ffmpeg -f concat -i FileList.txt -c copy "Path\NewName_joined.flv"
```

Share Improve this answer Follow

edited Mar 27, 2018 at 17:48
DavidPostill ♦
155k • 77 • 356 • 398

answered Sep 4, 2014 at 5:48
budman1
300 • 2 • 4

If you use -c copy it will maintain the original keyframes and their position. Your videos will be cut to those keyframes, so no, the time will not be exact. – user1323995 Oct 16, 2019 at 15:46

11

use this format:

```
ffmpeg <start time> <input file> <cut duration> <out file>
```

eg. cut 60 second clip after 1 minute of video

```
ffmpeg.exe -ss 00:01:00 -i "in file.mp4" -to 00:01:00 -c copy out.mp4
```

Notes:
start time before *input file* is faster.
start time after *input file* is accurate/precise.

Share Improve this answer Follow edited Jun 20, 2021 at 19:44 answered Feb 12, 2020 at 2:20 Zimba 1,081 11 15

Start time **before** input option is accurate/precise, too, for FFmpeg 2.1+. – MarianD Jan 23, 2022 at 2:00

It's accurate enough. In either case, if you're looking at frame level accuracy, then you'll need processing at frame level, requiring more lines of code, eg. removing watermarks or embedded subtitles. – Zimba Aug 16 at 4:24

8

For me `-t` option didn't work, but `-vframes` worked. I prefer using `#frames`, since I would rather cut at I-Frames and I found out GOP for video using `ffprobe`.

The command line that worked for me is:

```
ffmpeg -ss 60s -i input.wmv -vframes 1800 -acodec copy -vcodec copy output.wmv
```

By the way, putting `-ss` in the front of `-i` makes a big difference in execution time.

Share Improve this answer Follow edited May 19, 2013 at 6:40 slm 10.1k 10 50 57 answered May 19, 2013 at 3:14 Ben 81 1 1

2 Actually `-vframes` (or `-frames:v`) should come after `-i` because it's an output option. – slhck Nov 20, 2013 at 12:55

5

As with user225366, the -t option doesn't work for short videos, but it does for longer videos. For short videos it seems that -frames:v is better. This is what worked for me.

```
ffmpeg -ss 4 -i input.mp4 -frames:v 200 -vcodec copy output.mp4
```

`-acodec copy` needs to added if the video has audio, as the other answers show.

Share Improve this answer Follow

edited Jan 23, 2022 at 1:56
MarianD
2,686 1 17 26

answered Dec 8, 2016 at 11:24
VectorVortec
181 1 5

4

I see not many mention this (I'm no expert so maybe there is a catch), but if your file has other streams like subtitles and other metadata like chapters and so on, it's possible to cut/trim and keep all streams with the following command

```
ffmpeg -to 60 -i input.mkv -map 0 -c copy output.mkv
```

With `-map 0` you take all the streams in the file, and with `-c copy` you copy all them as they are.
Using `-to` omitting the start via `-ss` will cut the input video from start to second 60.

Fiddling with `map` is useful also if you want only specific streams to be kept in the cut (maybe you don't need all the audio sources in the file, or only some subtitles).

I use this when I need to split big MKV files that can't be stored in FAT32 storage.

Share Improve this answer Follow

edited Jun 12, 2020 at 13:48
Community Bot
1

answered Sep 1, 2019 at 5:29
Gruber
449 6 16

I added a start time and a end time, and it seems working well: ffmpeg -ss 18:50 -to 35:35 -i in.mp4" -map 0 -c copy out.mp4" – Harry Apr 30, 2021 at 3:41

0

Building on top of already great answers. What is missing there is preservation of **metadata**.

I am using following script to `cut_video`, which also **preserves metadata**:

```
#!/usr/bin/env bash

INPUT="$1"
START="$2"
DURATION="$3"

OUTPUT="${INPUT%.*}.cut.${INPUT##*.}"

ffmpeg -i "$INPUT" -ss "$START" -t "$DURATION" -c copy -movflags
use_metadata_tags -map_metadata 0 "$OUTPUT"
```

Explanation:

- `${INPUT%.*}` - the `%` removes the shortest matching suffix `.*`, i.e. deletes extension,
- `${INPUT##*.}` - the `##` removes the longest matching prefix `.*`, i.e. extracts extension.

See also following answer about metadata - [How to prevent FFmpeg from dropping metadata?](#).

Share Improve this answer Follow

answered Nov 10, 2022 at 18:36

kravemir
2,664 6 27 38

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.