Using ffmpeg to split video files by size [closed]

Asked 5 years, 11 months ago Modified 2 years, 5 months ago Viewed 17k times

16

7

Closed. This question does not meet Stack Overflow guidelines. It is not currently accepting answers.

We don't allow questions about general computing hardware and software on Stack Overflow. You can edit the question so it's on-topic for Stack Overflow.

Closed 1 year ago.

Improve this question

I'm trying to write a batch file using ffmpeg to automate the redundant daily task of taking footage from work that's recorded in 4gb blocks (which is standard in most DSLR cameras & GoPro's), and split the clips into 2gb files for streaming purposes. The idea is to have the script check external drive FOOTAGE's folder @import and split files after 2gb (since the max size is 4gb, this will alleviate the need for more than one split).

I'm also trying to amend the filenames of the split files, so FILE1 is 4gb, it splits into FILE1_1 and FILE1_2 which are 2gb each, respectively. Everything I've tried has just copied the original file into two new, identical files - no split or anything.

After doing some Googling and reading some of the answers here, I found this post, but it's based on duration, not size (recording video footage at varying levels of quality makes this pointless): Split into equal parts and convert many mp4 videos using ffmpeg

Can someone help me with this? I haven't come across any usable solutions utilizing what I understand to be the method, using -fs limit_size, and I really want to understand how this works.

UPDATE: Also found this, but it hasn't been updated in four years and I don't see anything in there regarding splitting that will prove helpful:

https://github.com/kcm1700/VideoSplitter/blob/master/

bash video split ffmpeg clip

Share Improve this question Follow

edited May 23, 2017 at 12:01
Community Bot
1 • 1

asked Jul 8, 2016 at 5:37
user2548469
377 • 2 • 5 • 9

3 Answers

Sorted by: Highest score (default)

27

I just had the same problem, and after I didn't find a solution here, I wrote a quick bash script for that job.

Features:

- Use ffmpeg's `-fs`-flag to limit filesize
- Check length of resulting video part and calculate where to start next part
- Enumerate video parts
- proceed with as many parts as needed to contain the whole source file.
- Allow for custom ffmpeg arguments to also reduce resolution and quality in one pass.

It takes only three arguments: The source filename, the desired filesize of each part and the desired arguments for ffmpeg.

Example call to split a file into 64MB parts:

```
./split-video.sh huge-video.mov 64000000 "-c:v libx264 -crf 23 -c:a copy -vf
scale=960:-1"
```

Finally, the source code of the script:

```
#!/bin/bash
# Short script to split videos by filesize using ffmpeg by LukeLR

if [ $# -ne 3 ]; then
    echo 'Illegal number of parameters. Needs 3 parameters:'
    echo 'Usage:'
    echo './split-video.sh FILE SIZELIMIT "FFMPEG_ARGS'
    echo
    echo 'Parameters:'
    echo '    - FILE:        Name of the video file to split'
```

```
    echo '    - SIZELIMIT:   Maximum file size of each part (in bytes)'
    echo '    - FFMPEG_ARGS: Additional arguments to pass to each ffmpeg-call'
    echo '                   (video format and quality options etc.)'
    exit 1
fi

FILE="$1"
SIZELIMIT="$2"
FFMPEG_ARGS="$3"

# Duration of the source video
DURATION=$(ffprobe -i "$FILE" -show_entries format=duration -v quiet -of
default=noprint_wrappers=1:nokey=1|cut -d. -f1)

# Duration that has been encoded so far
CUR_DURATION=0

# Filename of the source video (without extension)
BASENAME="${FILE%.*}"

# Extension for the video parts
#EXTENSION="${FILE##*.}"
EXTENSION="mp4"

# Number of the current video part
i=1

# Filename of the next video part
NEXTFILENAME="$BASENAME-$i.$EXTENSION"

echo "Duration of source video: $DURATION"

# Until the duration of all partial videos has reached the duration of the
source video
while [[ $CUR_DURATION -lt $DURATION ]]; do
    # Encode next part
    echo ffmpeg -i "$FILE" -ss "$CUR_DURATION" -fs "$SIZELIMIT" $FFMPEG_ARGS
"$NEXTFILENAME"
    ffmpeg -ss "$CUR_DURATION" -i "$FILE" -fs "$SIZELIMIT" $FFMPEG_ARGS
"$NEXTFILENAME"

    # Duration of the new part
    NEW_DURATION=$(ffprobe -i "$NEXTFILENAME" -show_entries format=duration -v
quiet -of default=noprint_wrappers=1:nokey=1|cut -d. -f1)

    # Total duration encoded so far
    CUR_DURATION=$((CUR_DURATION + NEW_DURATION))

    i=$((i + 1))

    echo "Duration of $NEXTFILENAME: $NEW_DURATION"
    echo "Part No. $i starts at $CUR_DURATION"

    NEXTFILENAME="$BASENAME-$i.$EXTENSION"
done
```

Hope, that helps :)

Share Improve this answer Follow

edited Sep 4, 2018 at 2:59

answered Sep 4, 2018 at 2:50

LukeLR
1,039 ● 1 ● 12 ● 24

2 May I know if this solution as you are providing it reduces video quality for chunks? ffmpeg noob here – Giorgi Aptsiauri Jun 27, 2020 at 17:36

1 That entirely depends on what you pass as `FFMPEG_ARGS` (third argument). If you pass nothing, ffmpeg will choose the settings automatically depending on your output container, which most likely means it will re-compress the content, causing unnecessary CPU load and quality loss. Therefore it is advised to pass at least `-c copy` as the third argument to this script, as this will instruct ffmpeg to copy the video and audio data without copying it. This makes conversion much faster (only limit is hdd speed) while keeping the original quality at the same time. – LukeLR Aug 28, 2020 at 11:25

Thank you. I've learned much more about FFMPEG since I posted this comment. I have used this script at the time but I had some trouble with it, I don't remember, I will look through it again. – Giorgi Aptsiauri Aug 28, 2020 at 13:19

2 Your script is wonderful! I think the default behavior should be unmodified copy and nothing re-code. – peterh Dec 12, 2020 at 19:50

This script only make the first part and fails. The `ffprobe` command to get duration returns same duration for the source file and first part file. Because of that while loop only executes once. – rusty Dec 9, 2021 at 11:52

16

You can do this in one command using mp4box.

```
mp4box -splits 2000000 filename.mp4
```

where `splits` argument takes value in kilobytes. Due to presence of keyframes, segments won't be exactly 2GB.

Share Improve this answer Follow

answered Jul 8, 2016 at 8:08

Gyan
74.8k ● 7 ● 138 ● 171

1 I thought it was in bytes? Won't that be 2MB? – Benubird Sep 6, 2016 at 19:12

No, it's in kB. – Gyan Sep 6, 2016 at 19:21

1 When I tried this tool, it put the videos into 3 parts when it only needed 2 parts. The last piece was 120kb. The previous part had over 100 MB of free space left over. – Daniel Kaplan Aug 2, 2020 at 23:45

this command just dont work. It just create 1 file with the size set on parameter – Natan Lotério Nov 11, 2020 at 17:27

Depends on where the keyframes are. – Gyan Nov 12, 2020 at 3:59

1

One idea would be to use ffmpeg with the option -fs. This wil limit the file size. The size of the output file is slightly more than the requested file size.

This will only create one file. But you can build a loop until the whole file is split.

First create a part of the file, then check how long it is with:

```
ffprobe -i input.file -show_format -v quiet | sed -n 's/duration=//p'
```

Then start another file with the offset by using -ss

After that do another round of encoding until the whole file is split. You can also use a stream copy for audio and video

Share Improve this answer Follow

answered Jul 8, 2016 at 6:50

mm2knet

66 • 9

3 Can you post an example of script utilizing -fs and -ss? – user2548469 Jul 8, 2016 at 7:05