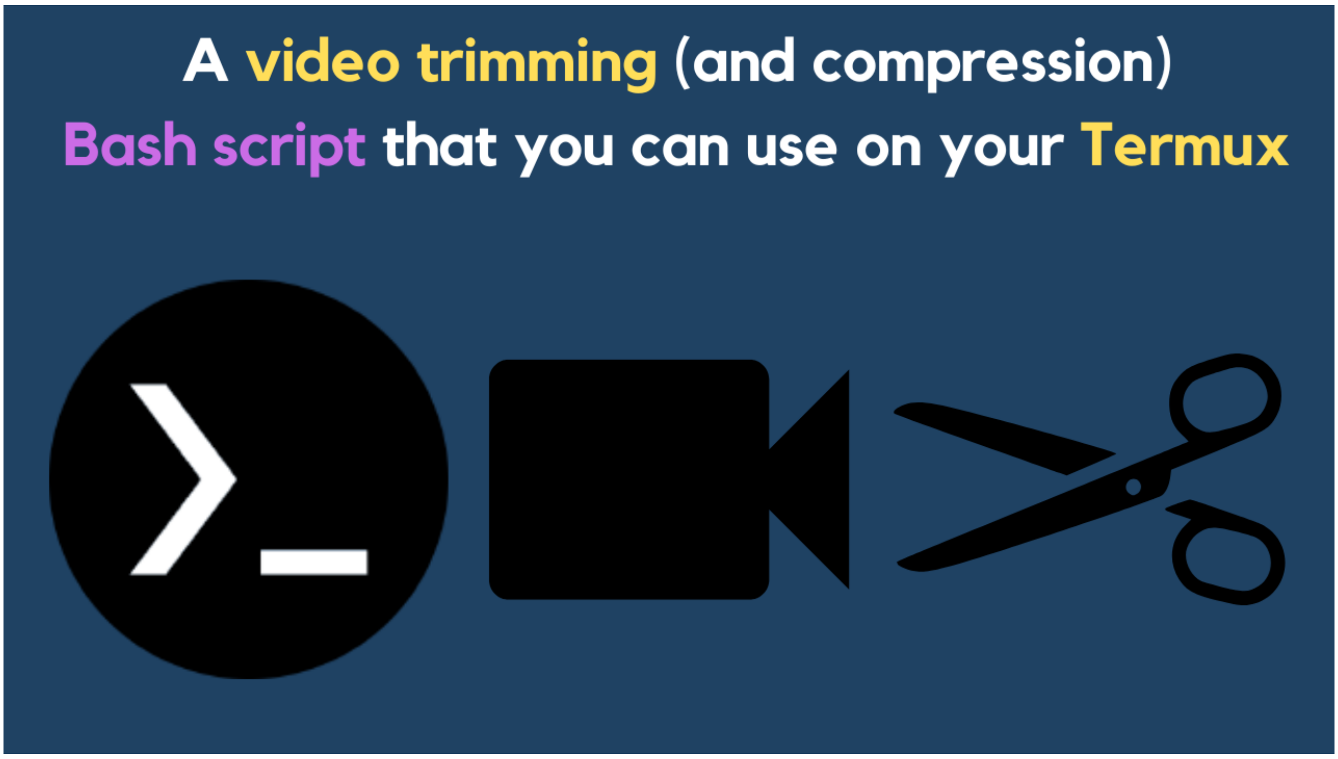
cut-video

HTML: https://paulojeronimo.com/posts/cut-video


A video trimming (and compression)
Bash script that you can use on your Termux

Paulo Jerônimo in Brasília on December 12, 2022: I created a Bash script for video trimming and compression using FFmpeg and HandBrakeCLI. It can be helpful if you are like me: I use command line tools to do everything in my daily work, including video transformations! ;)

Please, also note that I created this article post to use for recording my first video in English. I am a Brazilian, developing in Bash since the early 2000s, but I never recorded any video (in English) about this. So I decided to create a new Bash script, and a related article, to use as input for recording this video.

1. The cut-video script

The `cut-video` script is a **55 lines Bash script** that I wrote to do video cuttings and compression using FFmpeg and HandBrakeCLI.

> **55 is a "magic" Fibonacci number** and I love this kind of numbers! See another project that I developed (in JavaScript/ React): https://finisher.tech/fibonacci-app.

I will explain the code below. In order to do this, I put some tags inside the code (using comments in the format "# ref-string") and will explain all of them.

Here is the entire content of the script:

```bash
#!/usr/bin/env bash
# Author: Paulo Jerônimo (paulojeronimo.com)
# Links:
#   - Article | https://paulojeronimo.com/posts/cut-video
#   - Video | https://youtube.com/@PauloJeronimo
set -eou pipefail # ref-set
cd "$(dirname "$0")" # ref-cd-to-the-script-location
script_name=$(basename "$0" .sh)
source ./.$script_name 2> /dev/null || { \
  source ~/.$script_name 2> /dev/null || :; } # ref-source
clips_file=${clips_file:-clips.csv} # ref-clips_file
video_var=${video_var:-${PWD##*/}} # ref-video_var
mkv_compression=${mkv_compression:-false}
mkv_tool=${mkv_tool:-HandBrakeCLI} #ref-mkv_tool
$mkv_compression && \
  echo MKV compression will be made by $mkv_tool! || \
  mkv_tool=
go_ahead=true # ref-go_ahead
for t in ffmpeg $mkv_tool
do
  command -v $t > /dev/null || { # ref-command-isnt-in-path
    echo Command \"$t\" not found ...
    go_ahead=false
  }
done
for f in "$clips_file" original-video
do
  [ -f "$f" ] || { # ref-file-not-found
    echo File \"$f\" not found ...
    go_ahead=false
  }
done
$go_ahead || { # ref-abort
  echo Aborting due to the problems reported above!
  exit 1
}
log=$script_name.log; > $log # ref-log
```

```
while IFS=, read -r ss to video # ref-read-csv
do
  gen_file=$(sed "s,\$video,$video_var,g" <<< "$video") #ref-gen_file
  echo ffmpeg -nostdin -y -i original-video \
    -ss $ss -to $to -c:v copy \"$gen_file\" | tee last-cmd # ref-ffmpeg
  sed 's/^/\$ /g' last-cmd >> $log; bash < last-cmd &>> $log
  ! $mkv_compression || { # ref-mkv_compression
    [ -f ./$mkv_tool.$$ ] || echo "#!/usr/bin/env bash" > $mkv_tool.$$
    [ "$mkv_tool" = HandBrakeCLI ] && \
      cmd="$mkv_tool -i \"$gen_file\" -o \"${gen_file%.*}.mkv\"" || \
      cmd="$mkv_tool \"$gen_file\""
    echo -e "tee last-cmd <<< '$cmd'\nsed 's/^/\$ /g' last-cmd >> \
      $log\nbash < last-cmd &>> $log" >> $mkv_tool.$$
  }
done < "$clips_file"
! $mkv_compression || {
  mv -f ./$mkv_tool.$$ ./$mkv_tool; chmod +x ./$mkv_tool; ./$mkv_tool; }
# vim: tabstop=2 shiftwidth=2 colorcolumn=72
```

The more recent version of this script, maybe different and with more features, is also available in my `dotfiles` GitHub repository, here: https://github.com/paulojeronimo/dotfiles/.scripts/cut-video.

2. Details in the cut-video script

▼ *the first line*

In most of all Unix scripts, the first line contains an interpreter directive called [shebang].
It "is like a comment" in the first line but, actually, it is a character sequence `#!` ⋯ used by the program loader to interpret the rest of the script.
The loader executes the specified interpreter, passing it as an argument to the path initially used when attempting to run the script.
See more details in the references section.

▼ *the comment lines in the header*

Sometimes I like to write the header of my scripts using another structured text. In this case, it is written in YAML format. You can test this by typing the following command on your terminal (assuming you have yq installed):

```
yq -o json <(sed -n '2,5p' cut-video | sed 's,^# ,,g')
```

This will be the output:

```
{
  "Author": "Paulo Jerônimo (paulojeronimo.com)",
  "Links": [
    "Article | https://paulojeronimo.com/posts/cut-video",
    "Video | https://youtube.com/@PauloJeronimo"
  ]
}
```

▼ *ref-set*

TODO → Finish this topic.

▼ *ref-cd-to-the-script-location*

TODO → Finish this topic.

▼ *ref-source*

TODO → Finish this topic.

▼ *ref-clips_file*

The *clips_file* variable is file name, with a **Comma Separated Virgula (CSV)** content, that will be used to determine the positions where the file *original-file* (a link to the file that will be processed) will be cutted.

Here is a sample content for this file:

```
00:00:00,00:05:33,$video.1 - nothing-else-matters.mp4
00:05:34,00:12:43,$video.2 - master-of-pupets.mp4
00:12:44,00:18:09,$video.3 - fade-to-black.mp4
00:18:10,00:23:18,$video.4 - the-unforgiven.mp4
00:23:19,00:30:20,$video.5 - one.mp4
```

▼ *ref-video_var*

The *video_var* variable will be configured to get last part of the name of the current directory (indicated by the `$PWD`), and will be used to compose the name of the generated file in the cutting (made by FFmpeg) and also in the compression (made by HandBrakeCLI).

▼ *ref-mkv_tool*

TODO → Finish this topic.

▼ *ref-go_ahead*

The code below this comment will vefify if we have the conditions to run this script satisfied.

The first validation (**ref-command-isnt-in-path**) will verify if the required commands (FFmpeg and HandBrakeCLI) are available in the `PATH`.

The second validation (**ref-file-not-found**) will check if the files (`$clips_file` and `original-video`) are present on the current directory.

If one of these conditions aren't satisfied, the `go_ahead` variable will be `false`. In the **ref-abort** block, executed if `$go_ahead` is false, the script will print an aborting message and terminate its execution with an error code 1.

▼ *ref-log*

TODO → Finish this topic.

▼ *ref-read-csv*

This `while` will loop through the contents of the CSV (`$clips_file`).

TODO → Finish this topic.

3. Testing

In order to do the next tests with cut-video, you can copy and paste the following commands directly into your shell. I'll be using two environments in these tests: Termux and Linux.

3.1. Test 1

To follow the steps that I will show you, you will need the following tools installed:

1. yt-dlp
2. FFmpeg

I will demonstrate the commands below by doing this on my own mobile phone and using Termux.

To replicate the commands inside Termux, like me, first you will need to run the following script on it:

```
bash <(curl -sSL TODO(url)/termux-setup.sh)
```

Let's download a video containing some piano musics that I like (in low quality definition to speed up our test) and create a link called `original-file` to it:

```
d=/tmp/mettalica-on-the-piano; rm -rf $d; mkdir $d && cd $_ && \
yt-dlp -f 160 'https://www.youtube.com/watch?v=he_o9LmXYwg' && \
ln -sf "$(echo *.mp4)" original-video
```

After that, we'll download the cut-video script and make it executable:

```
s=cut-video; curl -sSL TODO(url)/$s -o $s; chmod +x $s
```

Before calling this script we need to create a file called `clips.csv` with the following command:

```
cat <<'EOF'>clips.csv
00:00:00,00:05:33,$video - 1.nothing-else-matters.mp4
00:05:34,00:12:43,$video - 2.master-of-pupets.mp4
00:12:44,00:18:09,$video - 3.fade-to-black.mp4
00:18:10,00:23:18,$video - 4.the-unforgiven.mp4
00:23:19,00:30:20,$video - 5.one.mp4
```

```
EOF
```

We can now call execute the cut-video script:

```
./cut-video
```

If everything goes weel, the tree of files created in this testing will be equals to the output shown in command below:

```
$ tree
.
|-- Best Metallica songs on the piano [he_o9LmXYwg].mp4
|-- clips.csv
|-- cut-video
|-- cut-video.log
|-- last-cmd
|-- metallica-on-the-piano - 1.nothing-else-matters.mp4
|-- metallica-on-the-piano - 2.master-of-pupets.mp4
|-- metallica-on-the-piano - 3.fade-to-black.mp4
|-- metallica-on-the-piano - 4.the-unforgiven.mp4
|-- metallica-on-the-piano - 5.one.mp4
`-- original-video -> Best Metallica songs on the piano [he_o9LmXYwg].mp4

0 directories, 11 files
```

3.2. Test 2

TODO → Finish this topic.

To follow this test we will need another environment and more some addional tools installed:

1. Docker
2. HandBrakeCLI

```
cat <<'EOF'>.cut-video
# vim: syntax=bash

mkv_compression=true
echo .cut-video loaded!
EOF
```

4. References

- **shebang**
  a. https://en.wikipedia.org/wiki/Shebang_(Unix)#Examples
  b. https://stackoverflow.com/questions/21612980/why-is-usr-bin-env-bash-superior-to-bin-bash
  c. https://www.baeldung.com/linux/bash-shebang-lines