ffmpeg et avconv

```
sudo apt-get install ffmpeg
```

avconv était un fork de ffmpeg sous Ubuntu.

Liens directs vers builds Windows :

- https://ffmpeg.zeranoe.com/builds/win32/static/ffmpeg-latest-win32-static.7z
- https://ffmpeg.zeranoe.com/builds/win64/static/ffmpeg-latest-win64-static.7z

liste les formats disponibles en import/export

```
ffmpeg -formats
```

liste les encoders disponibles

```
ffmpeg -encoders
```

Encodage hardware avec carte nVidia

Sous Windows, vérifier que la build ffmpeg est fonctionnelle avec l'encodage hardware :

```
ffmpeg -encoders | findstr /R /I /C:"nvidia"
```

Linux :

```
ffmpeg -encoders | grep -i nvidia
```

Dans tous les cas, on doit avoir un retour du style :

```
 V..... h264_nvenc           NVIDIA NVENC H.264 encoder (codec h264)
 V..... nvenc                NVIDIA NVENC H.264 encoder (codec h264)
 V..... nvenc_h264           NVIDIA NVENC H.264 encoder (codec h264)
 V..... nvenc_hevc           NVIDIA NVENC hevc encoder (codec hevc)
 V..... hevc_nvenc           NVIDIA NVENC hevc encoder (codec hevc)
```

On a donc bien les encoders nVidia, pour x264 et HEVC/h265. Il ne reste plus qu'à modifier les commandes pour profiter de l'encodage hardware :

- `-c:v libx264` sera remplacé par `-c:v h264_nvenc`,
- `-c:v libx265` sera lui remplacé par `-c:v hevc_nvenc`.

Petite comparaison de durée, sur une base core i7 4790k GeForce 980, 16Go RAM, sur une vidéo mpeg2 (.ts) :

```
ffmpeg -i test.ts -c:v h264_nvenc -crf 23 -acodec copy h264_nvenc.mp4
ffmpeg -i test.ts -c:v hevc_nvenc -crf 23 -acodec copy hevc_nvenc.mp4
ffmpeg -i test.ts -c:v libx264 -crf 23 -acodec copy libx264.mp4
ffmpeg -i test.ts -c:v libx265 -crf 23 -acodec copy libx265.mp4
```

On se retrouve avec :

```
h264_nvenc : 00:00:37.1873675 (37s),
hevc_nvenc : 00:01:28.3609427 (88s),
libx264 : 00:09:04.4235522 (544s, ~x15),
libx265 : 00:25:45.6305339 (1545s, ~x18).
```

infos d'un fichier

```
ffmpeg -i input.mp4 -hide_banner
```

bitrate constant

-minrate, **-maxrate** and the bitrate **-b:v** sont obligatoires pour le bitrate constant.

```
ffmpeg -i input.mp4 -c:v libvpx -minrate 1M -maxrate 1M -b:v 1M output.webm
```

bitrate variable

- **-qmin** : minimum quantization parameter (default 4)
- **-qmax** : maximum quantization parameter (default 63)
- **-b:v** : target bit rate setting. If not set, the encoder will choose ~1000 kBit/s as a default, but only when the -crf option is used.
- **-crf** : overall quality setting. If not set, the encoder will do “normal” VBR, trying to reach the target bitrate in within the qmin/qmax bounds. If set, the encoder will use CQ mode, and the target bitrate will become the maximum allowed bitrate. The CRF value is 10 by default.

```
ffmpeg -i input.mp4 -c:v libvpx -qmin 0 -qmax 50 -crf 10 -b:v 2M output.webm
```

conserver la qualité

qscale 0

```
ffmpeg -i input.mp4 -qscale 0 output.avi
```

copier seulement un passage d'une vidéo

- **-ss** starting time
- **-t** durée

```
ffmpeg -i input.avi -ss 00:15:30.00 -t 00:00:30.00 -c:v copy -c:a copy
ouput.avi
```

rip dvd

Via : http://kaocode.fr/sauvegarder-ses-dvd-video-avec-ffmpeg/

h264 + aac

```
ffmpeg -i concat:VTS_01_1.VOB\|VTS_01_2.VOB\|VTS_01_3.VOB -map 0:v -map 0:a
-c:v libx264 -crf 18 -maxrate 4000k -vf yadif -c:a libfdk_aac -b:a 320k
MonDVD.mkv
```

h265 + aac

```
ffmpeg -i concat:VTS_01_1.VOB\|VTS_01_2.VOB\|VTS_01_3.VOB -map 0:v -map 0:a
-c:v libx265 -crf 18 -maxrate 4000k -vf yadif -c:a libfdk_aac -b:a 320k
MonDVD-h265.mkv
```

VP9 + OPUS

```
ffmpeg -i concat:VTS_01_1.VOB\|VTS_01_2.VOB\|VTS_01_3.VOB -map 0:v -map 0:a
-c:v libvpx-vp9 -threads 8 -crf 14 -vb 6M -vf yadif -acodec libopus -ab 256k
MonDVD-VP9.mkv
```

video rapide à partir d'une image et d'une piste audio

```
ffmpeg -loop 1 -i img.jpg -i music.mp3 -shortest -c:v libx264 -c:a copy output.mp4
```

Si on veut affiner un peu les pas de 'seek' (option -g : par défaut 250, diminuer pour augmenter la fréquence des pas, mais augmente la taille finale de la vidéo) :

```
ffmpeg -loop 1 -i img.jpg -i music.mp3 -shortest -g 100 -c:v libx264 -c:a copy output.mp4
```

accélérer / ralentir vidéo

Via : https://trac.ffmpeg.org/wiki/How%20to%20speed%20up%20/%20slow%20down%20a%20video

```
ffmpeg -i input.mkv -filter:v "setpts=0.5*PTS" output.mkv
```

The filter works by changing the presentation timestamp (PTS) of each video frame. For example, if there are two succesive frames shown at timestamps 1 and 2, and you want to speed up the video, those timestamps need to become 0.5 and 1, respectively. Thus, we have to multiply them by 0.5.

Note that this method will drop frames to achieve the desired speed. You can avoid dropped frames by specifying a higher output frame rate than the input. For example, to go from an input of 4 FPS to one that is sped up to 4x that (16 FPS):

```
ffmpeg -i input.mkv -r 16 -filter:v "setpts=0.25*PTS" output.mkv
```

To slow down your video, you have to use a multiplier greater than 1:

```
ffmpeg -i input.mkv -filter:v "setpts=2.0*PTS" output.mkv
```

Speeding up/slowing down audio

You can speed up or slow down audio with the atempo audio filter. To double the speed of audio:

```
ffmpeg -i input.mkv -filter:a "atempo=2.0" -vn output.mkv
```

The atempo filter is limited to using values between 0.5 and 2.0 (so it can slow it down to no less than half the original speed, and speed up to no more than double the input). If you need to, you can get around this limitation by stringing multiple atempo filters together. The following with quadruple the audio speed:

```
ffmpeg -i input.mkv -filter:a "atempo=2.0,atempo=2.0" -vn output.mkv
```

Using a complex filtergraph, you can speed up video and audio at the same time:

```
ffmpeg -i input.mkv -filter_complex
"[0:v]setpts=0.5*PTS[v];[0:a]atempo=2.0[a]" -map "[v]" -map "[a]" output.mkv
```

convertir un FLAC en MP3

```
ffmpeg -i mon.flac -ab 192k mon.mp3
```

Et en gardant les metadatas :

```
ffmpeg -i mon.flac -ab 320k -map_metadata 0 -id3v2_version 3 mon.mp3
```

extraire audio

```
ffmpeg -i input.mp4 -vn -ac 2 -ar 44100 -ab 128k -f mp3 output.mp3
```

convertir vidéo en GIF

Via : http://mwholt.blogspot.com/2014/08/convert-video-to-animated-gif-with.html

- **-r** : fps du GIF
- **-delay** : 100/x fps : 100/5 ⇒ 20 fps

```
ffmpeg -i input.mp4 -r 20 -f image2pipe -vcodec ppm - | convert -delay 5 -
gif:- | convert -layers Optimize - output.gif
```

créer miniature d'une vidéo

Via : http://superuser.com/questions/538112/meaningful-thumbnails-for-a-video-using-ffmpeg

```
ffmpeg -ss 3 -i input.mp4 -vf "select=gt(scene\,0.5)" -frames:v 5 -vsync vfr
out%02d.jpg
```

```
ffmpeg -itsoffset -4 -i input.mp4 -vcodec png -vframes 1 -an -f rawvideo -s
120x90 -y output.png
```

Extraire toute la vidéo en images

```
ffmpeg -i input.mp4 -r 25 output_%04d.png
```

Inverse : compiler une série d'images et les animer

```
ffmpeg -r 60 -f image2 -s 1920x1080 -i pic%04d.png -vcodec libx264 -crf 25
-pix_fmt yuv420p output.mp4
```

Avec :

- `-r` : les FPS de la sortie,
- `-s` : la résolution de sortie,
- `%04d` : marge entre images, ici, 04 pour un compteur numérique à 4 chiffres en complétant avec des `0`, donc de `pic0001.png` à `pic WXYZ.pg`.

Ajouter un overlay

```
ffmpeg -i input.mp4 -i image.png -filter_complex "[0:v][1:v]
overlay=25:25:enable='between(t,0,20)'" -pix_fmt yuv420p -c:a copy
output.mp4
```

- `overlay=25:25` : Position de l'overlay : l'origine 0:0 est le coin supérieur gauche, donc à 25px à droite et 25px vers le bas,
- `enable='between(t,0,20)'` : Afficher entre les secondes 0 et 20, donc pendant les 20 premières secondes de la vidéo,
- `[0:v][1:v]` : Le premier fichier (ID:0) en entrée (`-i`) de type vidéo (`:v`), sur lequel on appose le seconde fichier (ID:1) toujours en entrée (`-i`) et encore aussi de type vidéo (`:v`),
- `-c:a copy` : copier la piste audio sans réencoder.

sous-titres

Via : https://en.wikibooks.org/wiki/FFMPEG_An_Intermediate_Guide/subtitle_options

- **-metadata:s:s:0** : that means to set the metadata for ... Stream:Subtitle:Number of stream, starting with 0
- **language code** : eng,,fre ou fra,ger,ita,spa

Copie vidéo :

```
ffmpeg -i input.mkv -sub_charenc CP1252 -i subtitle.srt -vcodec copy -acodec copy -scodec srt -metadata:s:s:0 language=fre output.mkv
```

Réencoder vidéo :

```
ffmpeg -i input.avi -sub_charenc ISO-8859-1 -i subtitle.srt -vcodec h264 -acodec ac3 -scodec srt -metadata:s:s:0 language=fre output2.mkv
```

Insertion brute depuis fichier source identique (`.mkv` contenant vidéo et sous-titres) :

```
ffmpeg -i input.mkv -vf subtitles=input.mkv output.mp4
```

Insertion brute depuis fichiers source distincts (vidéo `.mp4` et sous-titres externes en `.srt` par exemple) :

```
ffmpeg -i input.mp4 -vf subtitles=input.srt output.mp4
```

La même, mais en utilisant le second sous-titres d'un fichier `.mkv` :

```
ffmpeg -i input.mkv -vf subtitles=input.mkv:si=1 output.mp4
```

Forcer la police d'un sous-titre externe `.srt` :

```
ffmpeg -i input.mp4 -vf subtitles=input.srt:force_style='FontName=DejaVu Serif,PrimaryColour=&HAA00FF00' output.mp4
```

Scaling

Via : https://trac.ffmpeg.org/wiki/Scaling%20(resizing)%20with%20ffmpeg Forcer une taille :

```
ffmpeg -i input.avi -vf scale=320:240 output.avi
```

Garder le ratio :

```
ffmpeg -i input.avi -vf scale=320:-1 output.avi
```

En fonction de la taille originale :

```
ffmpeg -i input.avi -vf scale=iw*2:ih*2 output.avi
```

Forcer à respecter la taille maximale tout en conservant ratio :

```
ffmpeg -i input.avi -vf scale=w=320:h=240:force_original_aspect_ratio=decrease output.avi
```

Mapper plusieurs pistes audio

```
ffmpeg -i input.m2ts -map 0:0 -c:v libx265 -preset ultrafast -x265-params crf=23 -map 0:1 -c:a libopus -map 0:2 -c:a libopus -map 0:3 -c:a libopus -t 120 output.mkv
```

Ajouter piste audio

```
ffmpeg -i input.mp4 -i input.mp3 -c copy -map 0:0 -map 1:0 output.mp4
```

Exemples MP4 x265

x265 scale / crf / variable bitrate

```
ffmpeg -i input.mp4 -c:v libx265 -s 1280x720 -x265-params crf=30 -preset veryslow -c:a copy output.mkv
```

2pass / x265 / windows

```
ffmpeg -i "input.avi" -vf pad=608:360:0:4:black -c:v libx265 -preset slow -b:v 250k -x265-params pass=1 -c:a libfdk_aac -vbr 4 -f mp4 NUL && ffmpeg -i "input.avi" -vf pad=608:360:0:4:black -c:v libx265 -preset slow -b:v 250k -x265-params pass=2 -c:a libfdk_aac -vbr 4 "output.mkv"
```

gntm x265 crf

```
ffmpeg -i "input.mp4"  -c:v libx265 -x265-params crf=27 -c:a libfdk_aac -profile:a aac_he -b:a 48k  "output.mp4"
```

Exemples MP4 x264

scaling forced x264

```
ffmpeg -i "input.avi" -c:v libx264 -preset veryslow -b:v 2000k -aspect 16:9 -vf scale=1280x720,pad=1280:720:0:0:black output.mp4
```

720p x264 crf

```
ffmpeg -i "input.wmv" -c:v libx264 -preset slow -crf 28 -s 1280:720  -c:a libfdk_aac -profile:a aac_he_v2 -b:a 24k "output.mp4"
```

2pass x264

```
ffmpeg -i "input.mp4"  -c:v libx264 -b:v 10k  -pass 1 -c:a copy -f mp4 NUL && ffmpeg -i "input.mp4" -c:v libx264  -b:v 10k -pass 2 -c:a copy "output.mp4"
```

slowing down flip rotate slow mo framerate 24fps

```
ffmpeg -i "input.mp4" -c:v libx264 -filter:v "setpts=5.0*PTS" -filter:v "transpose=1" -r 24 -map 0:0 "output.mp4"
```

slowing down flip rotate slow mo framerate 15fps

```
ffmpeg -i "input.mp4" -c:v libx264 -filter:v "setpts=8.0*PTS" -filter:v "transpose=1" -r 15 -map 0:0 "output.mp4"
```

trim trimming cutting to (from 60s to 180s)

```
ffmpeg -i input.mp4 -ss 60 -to 180 -c:a copy -c:v copy output.mp4
```

Exemples MP4 AAC

aac low bitrate 48k 64k v1

```
ffmpeg -i "input.mp4"  -c:v copy -c:a libfdk_aac -profile:a aac_he -b:a 48k "output.mp4"
```

aac low bitrate 32k v2

```
ffmpeg -i "input.mp4"  -c:v copy -c:a libfdk_aac -profile:a aac_he_v2 -b:a 32k  "output.mp4"
```

aac vbr 4

```
ffmpeg "input.mp3" -c:a libfdk_aac -vbr 4   "output.m4a"
```

Optimisations Web

Utiliser l'attribut “faststart” :

```
ffmpeg -i input.mp4 -movflags faststart -acodec copy -vcodec copy output.mp4
```

Avoir un x264 (`libx264`) qui ne plante pas avec Firefox : forcer le format de pixel en `yuv420p` et pas en `yuv444` :

```
ffmpeg -i input.mp4 -acodec copy -vcodec libx264 -pix_fmt yuv420p output.mp4
```

Presets & Tune

Liste des presets (`-preset X`) disponibles en x264 :

- ultrafast,
- superfast,
- veryfast,
- faster,
- fast,
- medium,
- slow,
- slower,
- veryslow,
- placebo.

Liste des “tune” (`-tune X`) disponibles :

- film,
- animation,
- grain,
- stillimage,
- psnr,ssim,
- fastdecode,
- zerolatency.

Paramètres basiques

- **-i [fichier source]** – Nom et chemin du fichier source,
- **-codec:v** – indique l'encodeur vidéo utilisé, pour nous c'est la librairie libvpx pour du webm VP8,
- **-b:v [bitrate]** – indique le débit vidéo souhaité,
- **-quality good** – indique la qualité de l'encodeur vidéo. les choix disponibles sont “best” / “good” / “realtime”. (La choix “best” apporte un gain de qualité insignifiant),
- **-cpu-used [0-5]** – indique la vitesse de l'encodeur vidéo. (Une valeur faible indique une bonne qualité),
- **-r [25]** – indique à l'encodeur, le nombre d'image par seconde (fps) du fichier de sortie,
- **-maxrate** – Indique la limite haute sur le flux vidéo à ne pas dépasser,
- **-bufsize** – indique à l'encodeur le niveau auquel il est possible d'aller de façon ponctuel en terme de débit en cas de besoin. de façon générale le double du maxrate est utilisé afin de tenir sur 2 secondes,
- **-qmin 10** -qmax 42 – Minimum et maximum des valeurs de quantification (un qmax bas entraine une qualité élevée),
- **-threads [num]** – indique le nombre de fil d'exécution à utiliser. Indiquez le nombre de CPU disponibles,
- **-vf scale=[width:height]** – Changer de résolution. (La valeur “-1” signifie “conserver le ratio”, exemple: “-1:360” produira un fichier de 360p),
- **-codec:a libvorbis** – indique à l'encodeur libvorbis de produire un fichier audio vorbis,
- **-b:a [bitrate]** – Débit du flux audio,
- **-f webm** – indique à FFmpeg le format de sortie (si l'extension webm est présente sur le fichier de sortie, ce paramètre devient inutile),
- **-pass [1|2]** – indique à FFmpeg de faire du multiple passes et indique quelle passe est à traiter,
- **-an** – désactive l'encodage audio, pour gagner du temps au premier à la première passe d'encodage.

webm VP8

Forcer le téléchargement progressif :

http://sourceforge.net/projects/matroska/files/mkclean/

```
mkclean --optimize --remux input.webm output.webm
```

Erreur Type MIME pour serveur web :

```
AddType video/webm .webm
```

WEBM 1 passe:

```
ffmpeg -i input.mp4 -c:a libvorbis -b:a 128k -c:v libvpx -quality good -cpu-used 0 -slices 4 -qmax 30 -threads 8 output.webm
```

WEBM 1 passe avec resize max 1280x720 en gardant le ratio:

```
ffmpeg -i input.mp4 -vf scale=w=1280:h=720:force_original_aspect_ratio=decrease -c:a libvorbis -b:a 128k -c:v libvpx -quality good -cpu-used 0 -slices 4 -qmax 30 -threads 8 output.webm
```

WEBM 2 passes (Linux : "/dev/null" | Windows : "NUL"):

```
ffmpeg -i input.mp4 -codec:v libvpx -quality good -cpu-used 0 -b:v 600k -qmin 10 -qmax 42 -maxrate 500k -bufsize 1000k -threads 2 -vf scale=-1:480 -an -pass 1 -f webm /dev/null
ffmpeg -i input.mp4 -codec:v libvpx -quality good -cpu-used 0 -b:v 600k -qmin 10 -qmax 42 -maxrate 500k -bufsize 1000k -threads 2 -vf scale=-1:480 -codec:a libvorbis -b:a 128k -pass 2 -f webm output.webm
```

WEBM 2 passes:

```
ffmpeg -i input.mp4 -c:v libvpx -quality good -cpu-used 0 -slices 4 -qmax 30 -threads 8 -an -f webm -pass 1 NUL
ffmpeg -i input.mp4 -c:a libvorbis -b:a 128k -c:v libvpx -quality good -cpu-used 0 -slices 4 -qmax 30 -threads 8 -pass 2 output.webm
```

webm VP9

VOD

```
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 1 -b:v 1000K -threads 8 -speed 4 -tile-columns 6 -frame-parallel 1 -an -f webm /dev/null
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 2 -b:v 1000K -threads 8 -speed 1 -tile-columns 6 -frame-parallel 1 -auto-alt-ref 1 -lag-in-frames 25 -c:a libopus -b:a 64k -f webm output.webm
```

Best Quality (Slowest) Recommended Settings

```
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 1 -b:v 1000K -threads 1 -speed 4 -tile-columns 0 -frame-parallel 0 -g 9999 -aq-mode 0 -an -f webm /dev/null
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 2 -b:v 1000K -threads 1 -speed 0 -tile-columns 0 -frame-parallel 0 -auto-alt-ref 1 -lag-in-frames 25 -g 9999 -
```

```
aq-mode 0 -c:a libopus -b:a 64k -f webm output.webm
```

Constant Quality Recommended Settings

```
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 1 -b:v 0 -crf 33 -threads 8 -speed
4 -tile-columns 6 -frame-parallel 1 -an -f webm /dev/null
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 2 -b:v 0 -crf 33 -threads 8 -speed
2 -tile-columns 6 -frame-parallel 1 -auto-alt-ref 1 -lag-in-frames 25 -c:a
libopus -b:a 64k -f webm output.webm
```

Recommandation Google (best quality)

```
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 1 -b:v 1000K -threads 1 -speed 4
-tile-columns 0 -frame-parallel 0 -auto-alt-ref 1 -lag-in-frames 25 -g 9999
-aq-mode 0 -an -f webm /dev/null
ffmpeg -i input.mp4 -c:v libvpx-vp9 -pass 2 -b:v 1000K -threads 1 -speed 0
-tile-columns 0 -frame-parallel 0 -auto-alt-ref 1 -lag-in-frames 25 -g 9999
-aq-mode 0 -c:a libopus -b:a 64k -f webm out.webm
```

Sources

- Webm Project,
- ffmpeg VP9.

DNxHD (aka AVID)

Via : http://hd3g.tv/b/2011/08/transcoder-des-fichiers-videos-avec-ffmpeg-en-dnxhd/,
Doc codec : http://www.avid.com/fr/products/Avid-DNxHR-and-DNxHD.

```
ffmpeg -i input.mov -vcodec dnxhd -b <bitrate> -acodec pcm_s16le -f mov
output.mov
```

Exemples de bitrates disponibles et compatibles pour DNxHD :

- 1080p/25fps et 1080i/50fps : 36M, 120M, et 185M.
- 1080p/24fps : 36M, 115M, et 175M.
- 1080p/29,97fps et 1080i/59,9fps4 : 36M, 145M, et 220M.

Bonus : normalisation du son

Normaliser le son avec ffmpeg-normalize (Via
https://lepouf.info/normaliser-le-son-avec-ffmpeg-normalize/) :

On installe les dépendances, puis le paquet voulu (`ffmpeg-normalize`) :

```
apt-get install python python-pip ffmpeg
pip install ffmpeg-normalize
```

Exemples d'utilisation de `ffmpeg-normalize` :
Normalisation de volume :

```
ffmpeg-normalize -vu input.mp4
```

Normalisation de crête :

```
ffmpeg-normalize -vum input.mp4
```

Normalisation de crête à un volume spécifique (ici -10db) :

```
ffmpeg-normalize -vum -l -10 input.mp4
```

Normalisation de volume, avec codec et bitrate spécifique (les codecs disponibles sont ceux disponibles par `ffmpeg`) :

```
ffmpeg-normalize -vu -a libvorbis -e "-b:a 192k" input.mp4
```

Source de `ffmpeg-normalize` : https://github.com/slhck/ffmpeg-normalize

Bonus 2 : faire du reverse

Ou comment inverser le sens de lecture vidéo/audio.
Ici, les commandes simples pour inverser le sens, ne pas oublier d'y rajouter les options d'encodage voulues.

Pour inverser la vidéo (`-vf reverse`) :

```
ffmpeg -i input.mp4 -vf reverse output.mp4
```

Pour inverser l'audio (`-af areverse`) :

```
ffmpeg -i input.mp4 -af areverse output.mp4
```

Et pour les 2 en même temps (`-vf reverse` + `-af areverse`) :

```
ffmpeg -i input.mp4 -vf reverse -af areverse output.mp4
```

Pages Supplémentaires

P

- Paramètres H264_NVENC

P (suite)

- Paramètres HEVC_NVENC

webm sources

- https://www.virag.si/2012/01/webm-web-video-encoding-tutorial-with-ffmpeg-0-9/
- https://trac.ffmpeg.org/wiki/Encode/VP8
- http://www.webmproject.org/docs/encoder-parameters/
- http://wiki.webmproject.org/ffmpeg
- https://doc.ubuntu-fr.org/webm

A ajouter

https://www.ostechnix.com/20-ffmpeg-commands-beginners/