ffmpeg

- Objet : ffmpeg
- Niveau requis :
  débutant, avisé
- Commentaires : *ffmpeg est une suite de logiciels libres en ligne de commande qui permet de traiter des flux vidéos ou audio.*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
- Suivi :
  à-placer
  - Création par Pititux le 22-04-2010
  - Mis à jour le 29-04-2012
- Commentaires sur le forum : Lien vers le forum concernant ce tuto[1)]

Introduction

ffmpeg est une suite de logiciels libres en ligne de commande qui permet de traiter des flux vidéos ou audio. Avec ffmpeg, on peut déclencher des enregistrements, comme des lectures, appliquer des corrections à l'aide de filtres, ou transcoder des médias d'un format à un autre.

D'autres outils sont disponibles dans ce paquet et possèdent leur page propre:

- ffplay : lecteur multimédia
- ffprobe : analyseur de flux

Installation

```
apt install ffmpeg libavcodec-extra
```

Utilisation

FFmpeg peut être manipulé à l'aide de différentes interfaces graphique:

- VLC
- ...

Ou directement en ligne de commande comme on va le détailler ci-dessous.

Une documentation est disponible directement dans l'application, ou sur le site officiel : https://ffmpeg.org/

```
ffmpeg -h
```

Préambule

Une petite introduction aux différentes terminologies de la vidéo peut être utile pour bien comprendre la syntaxe de ffmpeg.

Les formats:

Le format est le container qui permet le transport de la vidéo, du son et des sous-titres soit sous forme de fichier (mkv, mov…) soit sous forme de flux (MPEG TS). A l'intérieur d'un container on peut insérer (muxer) ou extraire (demuxer):

- un ou plusieurs flux vidéo (un film, ou des chaînes de télévision….)
- un ou plusieurs flux audio (la version original, la version française, …)
- un ou plusieurs flux de sous-titres. (Français, Sourd et malentendant, …)
- plus des métadonnées (titre, nom de l'artiste par exemple)

On parle de multiplexer les différentes pistes (flux ou stream) dans un format.

FFmpeg fournit une liste des formats qu'il supporte:

```
ffmpeg -formats
```

retour de la commande

```
DE avi             AVI (Audio Video Interleaved)
DE ogg             Ogg
D  matroska,webm   Matroska / WebM
 E mov             QuickTime / MOV
D  mov,mp4,m4a,3gp,3g2,mj2 QuickTime / MOV
 E webm            WebM
```

Le D signifie la capacité à le lire, et E la possibilité d'encapsuler dans le format.

Il est possible de voir les options du muxer ou demuxer disponibles pour un format spécifique comme par exemple avec *matroka* (.mkv) :

```
ffmpeg -h muxer=matroska
```

Les codecs

Le codec est un algorithme qui permet d'encoder la vidéo ou le son afin de l'adapter au protocole de transport (IP,DVB,fichier…) notamment en réduisant le débit(Kbits/s). Selon les codecs, la compression peut s'accompagner d'une perte de qualité dans l'image ou le son plus ou moins importante.
De la même manière que pour les formats, ffmpeg liste les codecs qu'il est capable de gérer:

```
ffmpeg -codecs
```

Il est possible de voir les options disponibles d'un encoder ou décoder pour un codec spécifique comme par exemple *vp9*:

```
ffmpeg -h encoder=vp9
```

Les filtres

ffmpeg dispose aussi d'une base importante de filtres qui permettent de modifier le contenu de chaque flux, comme changer la résolution, modifier le volume d'une piste, incruster un logo etc....

```
ffmpeg -filters
```

Il est possible de voir les options disponibles d'un filtre spécifique comme par exemple avec *scale* :

```
ffmpeg -h filter=scale
```

Formats + Codecs + filtres

Si on résume, ffmpeg permet de multiplexer ou de-multiplexer dans différents formats:

- des flux vidéos compressés (ou pas),
- des flux audio compressés (ou pas),
- des sous-titres dans différents formats.

Et de modifier à l'aide de filtres le contenu de chaque flux indépendamment.

Mais si ffmpeg peut gérer des tonnes de formats et des tonnes de codecs différents, toutes les combinaisons ne sont pas possible, comme le montre ce tableau: http://www.videolan.org/streaming-features.html.

En effet, chaque codec ou format comporte sa propre norme avec plus ou moins de licences restrictives, toutes les combinaisons ne sont donc pas possibles.
Heureusement le libre fournit plusieurs formats ainsi que plusieurs codecs libre de droit :

- Dans les formats citons: mkv, webm, ogv, ogg...
- Dans les codecs vidéos citons: AV1,vp9, vp8, theora(vp3), dirac...
- Dans les codecs audio citons: flac, opus, vorbis et bien d'autres...

Connaître le contenu d'un fichier

Avant de commencer tout encodage il est bon de connaître son contenu, ffmpeg permet de lire l'entête du “format”:

```
ffmpeg -i tears_of_steel.mkv
```

- -i permet de déclarer l'entrée à démuxer.

Ce qui nous retourne ceci:

retour de la commande

```
Input #0, matroska,webm, from 'tears_of_steel.mkv':
Metadata:
  title           :
  ARTIST          :
  COMPOSER        :
  SYNOPSIS        :
  DATE_RELEASED   :
  GENRE           :
  ENCODER         : Lavf54.29.104
Duration: 00:12:14.12, start: 0.000000, bitrate: 4615 kb/s
  Stream #0:0(eng): Video: h264 (Main), yuv420p, 1280x534 [SAR 1:1 DAR
640:267], 24 fps, 24 tbr, 1k tbn, 180k tbc (default)
  Stream #0:1(eng): Audio: aac, 44100 Hz, stereo, s16 (default)
  Stream #0:2(eng): Audio: ac3, 48000 Hz, 5.1(side), s16, 448 kb/s
(default)
  Stream #0:3(eng): Subtitle: ssa (default)
  Stream #0:4(fr): Subtitle: ssa (default)
```

Et pour ce fichier, on y apprend beaucoup de choses :

- L'input 0 décrit le format du fichier (Matroska ici). (Certains formats autorisent plusieurs Input, d'où la numérotation)
- Le fichier comprends 5 flux (stream) numérotés de 0 à 4:
  - un flux vidéo encodé en h264
  - un flux audio stéréo en anglais
  - un flux audio 5.1 en anglais
  - et deux sous-titres Anglais + Français.

Changer de Format

On peut facile changer de format sans toucher au flux :

```
ffmpeg -i tears_of_steel.mkv -c copy tears_of_steel.mov
```

COMMANDE	***ACTION***
-i	spécifie le fichier d'entrée
-c copy	copie à l'identique la totalité des flux
-c:v copy	copie à l'identique les pistes vidéos
-c:a copy	copie à l'identique les pistes audio
-c:s copy	copie à l'identique les pistes sous-titres

Si on n'indique pas à ffmpeg de faire la copie exacte des flux, il lancera un profil d'encodage par défaut.

Si l'on souhaite différencier les flux entre eux

COMMANDE	***ACTION***
-c:v:0	première piste vidéo
-c:a:0	première piste audio
-c:a:1	deuxième piste audio
-c:s:0	première piste sous-titre
-c:s 1	deuxième piste sous-titre

Pour des fichiers comprenant plus de 3 flux, il semble qu'il y ait un bug avec la fonction “copy”.Certains flux sont perdus lors de la copie, il faut passer par la fonction mapping pour pouvoir en copier l'intégralité.

Organiser l'ordre des flux (mapping)

ffmpeg permet de modifier l'ordre des flux pour les adapter à ses usages à l'aide du paramètre “-map” :

```
ffmpeg -i tears_of_steel.mkv -map 0:0 -map 0:2 -map 0:1 -map 0:4 -map 0:3 -c
copy tears_of_steel-v2.mkv
```

Dans cette commande j'inverse les deux flux audio entre-eux et ainsi que les deux sous-titres.

- -map 0:1: piste 1 de l'input 0 (tears_of_steel.mkv)
- -map 0:2: piste 2 de l'input 0 (tears_of_steel.mkv)

Lors de l’exécution ffmpeg indique quel croisement (mapping) il applique :

retour de la commande

```
Stream mapping:
Stream #0:0 -> #0:0 (copy)
Stream #0:2 -> #0:1 (copy)
Stream #0:1 -> #0:2 (copy)
Stream #0:4 -> #0:3 (copy)
Stream #0:3 -> #0:4 (copy)
```

Ajouter un flux

Le paramètre `-map` permet aussi d'ajouter un flux.Dans l'exemple qui suit un sous-titre en espagnol (TOS-es.srt) en position 4 :

```
ffmpeg -i tears_of_steel.mkv  -i TOS-es.srt -map 0:0 -map 0:1 -map 0:2 -map
1:0 -map 0:3 -map 0:4 -c:v copy -c:a copy  -metadata:s:s:0 language=esp
tears_of_steel-v2.mkv
```

- -map 0:0 faisant référence au premier flux du premier “input” : soit la vidéo du fichier “tears_of_steel.mkv”
- -map 1:0 faisant référence au premier flux du deuxième “input” : soit le sous-titre du fichier “TOS-es.srt”

Cela permet de l’insérer où l'on souhaite :

- Stream #0:0(eng): Video: h264 (Main), yuv420p, 1280×534 [SAR 1:1 DAR 640:267], 24 fps, 24 tbr, 1k tbn, 180k tbc (default)
- Stream #0:1(eng): Audio: aac, 44100 Hz, stereo, s16 (default)
- Stream #0:2(eng): Audio: ac3, 48000 Hz, 5.1(side), s16, 448 kb/s (default)
- Stream #0:3(esp): Subtitle: ssa (default)
- Stream #0:4(eng): Subtitle: ssa (default)
- Stream #0:5(fr): Subtitle: ssa (default)

L'option metadata permet de renseigner la langue du fichier :

- -metadata:s:s:0 language=esp

Extraire un flux

En utilisant les possibilités du mapping il est très facile d'extraire un seul flux d'un fichier en comportant plusieurs :

```
ffmpeg -i tears_of_steel.mkv -map 0:2 -acodec copy tears_of_steel_BO.mkv
```

Dans cet exemple, je copie qu'une seule des pistes audio.

Extraire une piste audio en MP3

Et l'on peut, très bien, faire une conversion dans la foulée :

```
ffmpeg -i tears_of_steel.mkv -map 0:2 -acodec libmp3lame -ar 44100 -ac 2 -ab
192k tears_of_steel_BO.mp3
```

Méthode-2/plus simple :

```
ffmpeg -i video_origine.avi -vn -ar 44100 -ac 2 -ab 192 -f mp3 son_final.mp3
```

Source : http://www.jcartier.net/FFMpeg-par-l-exemple.html

Extraire un sous-titre

```
ffmpeg -i tears_of_steel.mkv -map 0:4 -c:s srt tears_of_steel_FR.srt
```

Extraire un segment

FFmpeg permet d'extraire, des morceaux d'un média, en précisant un point d'entrée avec **-ss** (-ss 00:06:46) et précisant un durée avec **-t** (-t 00:01:00). Cette fonction est très utile pour faire des tests d'encodage et pour valider sa commande. A noter que **-ss** et **-t** doivent être placés devant le premier input (-i).

```
ffmpeg -ss 00:06:46 -t 00:01:00 -i tears_of_steel.mkv -c copy
tears_of_steel_extrait.mkv
```

Parfois la durée **-t** n'est pas prise en compte. On peut utiliser l'option **-to** en la plaçant au niveau du fichier de sortie

```
ffmpeg -ss 00:06:46  -i tears_of_steel.mkv -c copy -to 00:01:00
tears_of_steel_extrait.mkv
```

Pour couper un vob

```
ffmpeg -fflags +genpts -i "ton_vob.VOB" -ss 00:05:00 -t 00:01:00 -map 0:v -
map 0:a -c:v copy -c:a copy -y out.VOB
```

à partir de:

```
-ss 00:05:00
```

pour une durée de une minute :

```
-t 00:01:00
```

Voir sur le forum : https://debian-facile.org/viewtopic.php?pid=366931#p366931

Merci aux participants pour cette solution bien pratique. :)

Augmenter le nombre de threads

Selon les codecs, il est possible de lancer une conversion sur plusieurs threads processeurs grâce à l'option **threads** :

```
ffmpeg -threads 4 -i tears_of_steel.mkv -c copy tears_of_steel_extrait.mkv
```

Ici je force le travail sur 4 threads ce qui peux me faire gagner un temps précieux.

-threads 0 correspond au mode automatique et utilisera le maximum de threads disponibles mais il ne fonctionne pas avec tous les codecs (notamment libvpx)

Encoder la vidéo

La liste des options ffmpeg pour le traitement Video est disponible ici:
http://ffmpeg.org/ffmpeg.html#Video-Options

En voici les principales:

option	explication de l'option
-r	définit le nombre d'images par seconde
-s	configuration de la taille du cadre d'affichage
-aspect	configuration du format d'affichage (4:3, 16:9 ou 1.3333, 1.7777)
-vcodec ou -c:v	décision du choix du codec
-pass	nombre de passage à l'encodage, une passe (-pass 1) ou deux passes (-pass 2)
-crf	permet de définir un niveau de qualité entre 0 et 63 (petit nombre = meilleure qualité mais plus de temps de calcul) (défaut 23)

CRF (Constant Rate Factor)

Il faut utiliser le CRF pour définir une qualité d'image constante. L'objectif est d'obtenir une qualité d'image stable entre plusieurs vidéos. À durée égale, la taille du fichier peut varier suivant si la vidéo est facilement compressible ou pas. Un dessin animé avec des aplats prendra moins de place qu'un match de foot avec des panoramiques sur 80000 spectateurs. L'échelle crf est logarithmique entre 0 et 63, une différence de 6 points double ou divise par 2 environ la taille du fichier final. Un petit nombre égal une meilleure qualité mais plus de temps de calcul, la valeur est souvent par défaut 23.

Le choix du CRF dépends du type d'image à encoder, de la résolution de l'image, de qualité souhaité ou encore de la taille du fichier désiré. Vous pouvez faire des tests sur des segments avant de procéder à l'encodage total.

Encoder la vidéo en VP8

Le VP8 est souvent déprécié au profit du VP9 plus souple et plus performant

VP8 est un codec vidéo libre promu par Google, un bon équivalent au h264/mp4 :

```
ffmpeg -i tears_of_steel_720p.mkv -c:v:0 libvpx -crf 10 -vb 4M -c:a copy
```

```
tears_of_steel_vp8.mkv
```

Ici on utilise la librairie **libvpx** avec deux options :

- -crf permet de définir un niveau de qualité entre 0 et 63 (petit nombre = meilleure qualité mais plus de temps de calcul)
- -vb 4M permet de donner un objectif de débit à 4 Megabit/s

Une liste des options possibles pour encoder en vp8 est disponible ici:
http://wiki.webmproject.org/ffmpeg

Encoder la vidéo en VP9

VP9 est un codec vidéo libre promu par Google pour concurrencer le h265/HEVC :

```
ffmpeg -i tears_of_steel_720p.mkv -c:v:0 libvpx-vp9 -threads 8 -crf 10 -vb
4M -c:a copy   tears_of_steel_vp9.mkv
```

Ici on utilise la librairie libvpx-vp9 avec trois options :

1. crf permet de définir un niveau de qualité entre 0 et 63 (petit nombre = meilleure qualité mais plus de temps de calcul)
2. vb 4M permet de donner un objectif de débit à 4 Megabit/s
3. threads reste nécessaire pour forcer le nombre de core, libvpx-vp9 ne les detecte pas encore automatiquement.

Une liste des options possibles pour encoder en vp9 est disponible ici:
http://wiki.webmproject.org/ffmpeg/vp9-encoding-guide

Encoder la vidéo en H264/Mpeg4

Le h264 ou mpeg4 est un codec propriétaire couramment utilisé pour son bon rapport qualité/débit :

```
ffmpeg -i tears_of_steel_720p.mkv -c:v:0 libx264 -preset slow -crf 22 -c:a
copy tears_of_steel_h264.mp4
```

- **-preset** permet de définir une vitesse d'encodage, plus il sera lent plus l'image sera de qualité. Sont disponible: ultrafast,superfast, veryfast, faster, fast, medium, slow, slower et veryslow. Medium étant le réglage par défaut.
- **-crf** permet de définir un niveau de qualité entre 0 et 51 (petit nombre = meilleure qualité mais plus de temps de calcul)

Une documentation plus détaillé en anglais est disponible ici:
https://trac.ffmpeg.org/wiki/Encode/H.264

Encoder la vidéo en H.265/HEVC

Le H.265/HEVC est un nouveau codec propriétaire, successeur du H264/Mpeg4. La syntaxe suit celle

du H264

```
ffmpeg -i tears_of_steel_720p.mkv -c:v:0 libx265 -preset slow -crf 22 -c:a
copy tears_of_steel_h265.mkv
```

Une documentation plus détaillée en anglais est disponible ici : https://trac.ffmpeg.org/wiki/Encode/H.265 et ici http://x265.readthedocs.org/en/default/

Encoder VOB en MKV

- https://debian-facile.org/viewtopic.php?id=24453

Aspect 16:9

Encoder une vidéo captée en 4:3 pour la remettre en 16:9 :

```
ffmpeg -i film.mkv -aspect 16:9 -c copy film2.mkv
```

Encoder le son

La liste des options ffmpeg pour le traitement Audio est disponible ici: http://ffmpeg.org/ffmpeg.html#Audio-Options

En voici les principales:

option	explication de l'option
-acodec ou -c:a	détermine le choix du codec
-ar	configuration de la fréquence d'échantillonnage (44100 Hz)
-ab	configuration du débit binaire par défaut 64 kbps
-ac	configure le nombre de canaux (mono-stéréo)

Encoder le son en Vorbis

Vorbis est un codec audio libre, d'encodage avec perte, équivalent au mp3/h263 :

```
 ffmpeg -i tears_of_steel_720p.mkv -c:v copy -c:a:0 libvorbis -qscale:a 5 -
ar 48000   tears_of_steel_vorbis.mkv
```

- -qscale:a permet de gérer la qualité d'encodage sur une échelle de 0–10, où 10 est la meilleure qualité. par défaut qscale: a est à 3.
- -ar 48000 permet de ré-échantillonner l'audio en 48Khz.

Fondu d'un son wav

Créer un fondu entrant et sortant sur un fichier wav :

```
ffmpeg -i LE_FICHIER.wav -af "afade=t=in:ss=0:d=15,afade=t=out:st=200:d=22"
output.wav
```

Merci golgot200 🙂

- https://debian-facile.org/viewtopic.php?pid=197379#p197379

Encoder un fichier audio en mp3

Plusieurs codec permettent d'encoder le son en mp3, libmp3lame est l'un des plus utilisé.
Voici un exemple en forçant le bitrate à 256kbits/s, en ré-échantillonnant en 44100hz et en forçant la fabrication d'un stéréo :

```
ffmpeg -i 01.ogg -acodec libmp3lame -ar 44100 -ac 2 -ab 256k 01.mp3
```

Attention les métadonnées peuvent être perdues en utilisant cette méthode

Images Fixes

Transformer une série d’images en vidéo

Dans un répertoire nommé par exemple *images* on rassemble les fichiers JPG que l'on veut assembler en vidéo.
Pour faciliter le processus, ces fichiers doivent être numérotées, par exemple :
image1.jpg image2.jpg image3.jpg... On lance ensuite la commande :

```
ffmpeg -f images -i image%d.jpg video.mpg
```

Ce qui transformera les images contenues dans le répertoire images : image1.jpg, image2.jpg, image3.jpg ... en un fichier vidéo nommé video.mpg.
Notons que %d sera automatiquement transformé en 1, 2, 3, 4, 5...

Si l’on a des images nommées image001.jpg, image002.jpg, image003.jpg, ... vous utiliserez la commande :

```
ffmpeg -f images -i image%03d.jpg video.mpg
```

Mais on peut aussi utiliser d’autres types de format d’images : PGM, PPM, PAM, PGMYUV, JPEG, GIF, PNG, TGA, TIFF, SGI, PTX

On peut aussi paramétrer plus finement l’export vidéo :

```
ffmpeg -r 24 -b 1800 -i image%d.bmp video.mpg
```

Ici on spécifie 24 images par seconde et un bitrate de 1800kb/s.

huffyuv

Transformer une image fixe jpg en video mp4 :

```
ffmpeg -i votre_image.jpg -c:v huffyuv -vcodec libx264 -r 0.07
votre_image.mp4
```

L'option :

```
-r 0.-r 0.07
```

module à environ 1/15e la vitesse de défilement pour la video MP4. Si on augmente, ça accélère, si on diminue, ça ralentit.

Merci au **captnfab** qui de son vaisseau flambloie les tutos à tour de bras ! 😀

Transformer une vidéo en une série images

```
ffmpeg -i video.mpg image%d.jpg
```

ce qui générera les fichiers image1.jpg, image2.jpg, ...
Mais on peut aussi générer des images au format : PGM, PPM, PAM, PGMYUV, JPEG, GIF, PNG, TIFF, SGI. Par exemple :

```
ffmpeg -i video.mpg image%d.tif
```

Extraire une seule image (Vignette)

```
ffmpeg -i fichier_video -f mjpeg -ss 10 -vframes 1 -s 320x240
fichier_vignette.jpg
```

- -f mjpeg pour obtenir un fichier en format JPG
- -ss 10 pour choisir la vignette de la 10ème seconde après le début de la vidéo
- -vframes 1 la première frame
- -s 320×240 les dimensions de la vignette image JPG.

L'ancien ffmpeg demandait l'option : -f **image2** à la place de -f **mjpeg** maintenant.

Enregistrer son bureau

La capture d'écran vidéo dépend beaucoup des capacités de votre machine, notamment de la vitesse d'écriture des disques et de la rapidité du processeur pour l'encodage temps réel. Il est d'ailleurs conseillé de faire la capture de manière brute et de faire un encodage plus fin par la suite.
Si votre machine ne permet d'enregistrer qu'un nombre plus petit d'images que prévu, la vidéo

semblera accélérée.

FFmpeg permet de capturer la sortie du serveurX avec le module x11grab :

```
ffmpeg -f x11grab -r 25 -s 1280x1024 -i :0.0  -vcodec libx264 -crf 0 -preset ultrafast output.mkv
```

- -i 0:0 représente le premier écran
- -r donne le nombre d'image par seconde
- -s la résolution de la capture

Pour capturer seulement un morceau de l'écran, on peut préciser les coordonnées du point de départ de la capture (x=200 et y= 100) :

```
ffmpeg -f x11grab -r 25 -s 512x512 -i :0.0+200,100  -vcodec libx264 -crf 0 -preset ultrafast output.mkv
```

FFmpeg ne semble pas respecter le nombre d'image par seconde. Si les capacités de la machine le permettent, il pourra aller au delà, la vidéo sera donc ralentie.

Plus d'informations disponibles sur :

- trac.ffmpeg.org

Enregistrer sa webcam

FFmpeg permet de capturer l'image en provenance d'une Webcam grâce au module video4linux2 :

```
ffmpeg -f video4linux2 -r 25 -s 640x480 -i /dev/video0 mawebcam.avi
```

- /dev/video0 pointe vers votre webcam.
- -r 25 force la capture à 25 images par seconde (fps).
- -s 640×480 force la résolution de la capture.

Plus d'information sur la capture avec video4linux2 :

- trac.ffmpeg.org

FFmpeg ne semble pas respecter le nombre d'image par seconde. Si les capacités de la machine le permettent, il pourra aller au delà, la vidéo sera donc ralentie.

cat - Regrouper plusieurs vidéos VOB en mp4

- cat-convertion-en-mp4

Regrouper plusieurs vidéos mp4 en 1 seule

Il y a une fonction **concat** qui concatène des fichiers en entrée. C'est ce que l'on va utiliser… sauf que cela ne fonctionne pas avec des fichiers MP4 (container MOV, h264 en codec vidéo, et aac en codec audio).

Préparation

On va d'abord changer de container pour nos flux source, puis les assembler, car on ne va pas refaire de transcodage, juste de la copie de flux.
La fonction **concat** accepte bien le MPEG Transport Stream, allons y :

```
ffmpeg -y -i "source1.mp4" -c copy -bsf:v h264_mp4toannexb -f mpegts
"source1.ts"
```

Le -bsf:v h264_mp4toannexb restructure le flux h264 pour qu'il puisse rentrer dans le TS. Il le demande, et ça marche.

Assemblage

Une fois que tous les fichiers à assembler sont passés en TS, on les regroupe et les converti en mp4 ainsi :

```
ffmpeg -y -i concat:"source1.ts|source2.ts" -c copy -f mov "destination.mp4"
```

Si cela ne fonctionne pas, vérifiez que vos fichiers ont bien le même format, codec, résolution…
À la moindre différence, l'assemblage est impossible.

Source :

- http://hd3g.tv/b/2012/08/coller-des-fichiers-mp4-entre-eux-avec-ffmpeg/

Utiliser une boucle for

Si les fichiers à assembler sont nombreux, on peut utiliser une boucle.

Rassembler les vidéos afin qu'elle portent un numéro progressif pour la vidéo finale.
Créer le répertoire accueillant les MPG, par exemple :

```
mkdir ~./video-mpg
```

Y déposer les videos MPG à assembler :

```
mv /répertoire_acquis/*.mpg ~./video-mpg
```

Les numéroter dans l'ordre où nous souhaitons les assembler.
Perso, j'utilise GPRENAME qui me rend la vie facile... 😉

Lister le contenu obtenu :

```
ls ~./video/
```

retour de la commande ls

```
0001.mpg 0002.mpg 0003.mpg 0004.mpg 0005.mpg
```

Nous allons d'abord les convertir chacune en mp4 en utilisant une première fois la boucle for :

```
cd /repertoire/video-mpg
```

```
for i in {0001..0005}; do $i -c:v:0 libx264 -preset slow -crf 15 -s 1024x768
-threads 0 -c:a $i.mp4; done
```

Explication :
`for i in {0001..0005};`

Se traduit : la variable `i` prendra respectivement les chiffres `0001 0002 0003 0004 0005` comme valeur.
Le point virgule `;` indique la fin de cette indication.

Ensuite vient : `do $i -c:v:0 libx264 -preset slow -crf 15 -s 1024×768 -threads 0 -c:a $i.mp4;done`

Explication :

Au début : `do $i` indique qu'il faut considérer chacun des numéros attribués à la variable `i` précédemment définie.
Le : `;done` indique la fin de la commande qui est à répéter pour chaque variable `$i` trouvée.

Se placer dans le dossier rassemblant les vidéos MPG numérotées :

```
cd /repertoire/video-mpg
```

On exécute la commande de conversion en TS :

```
for i in {0001..0005}; do ffmpeg -y -i "$i.mp4" -c copy -bsf:v
h264_mp4toannexb -f mpegts "$i.ts"; done
```

Et pour assembler le tout directement en MP4 :

```
ffmpeg -y -i concat:"0001.ts|0002.ts|0003.ts|0004.ts|0005.ts" -c copy -f mov
"1.mp4"
```

Où la vidéo 1.mp4 est la vidéo finale assemblée.

Alternative :

En utilisant la boucle de nouveau :

```
toto=""; for i in {0001..0005}; do toto="$toto$i.ts|"; done; ffmpeg -y -i
concat:"$toto" -c copy -f mov "1.mp4"
```

Merci **kyodev** pour cette astuce ! 😀

Source de la boucle :

- copies-incrementees

Concatener facile


À tester...

Pour concaténer des mkv avec ffmpeg il faut faire comme suit : Créer un fichier avec le contenu :

```
file 'partie1.mkv'
file 'partie2.mkv'
```

Appelons ce fichier « list-mkv.txt ». Alors la commande :

```
ffmpeg -f concat -i list-mkv.txt -c copy output.mkv
```

Source sur le forum :

- https://debian-facile.org/viewtopic.php?pid=345771#p345771

Pense-bête à ska

Commentaire	Option	
On peut séparer plusieurs filtres avec une virgule, l'ordre est important	-vf filtre1,filtre2,etc	
Rogner l'image, origine = en haut à gauche	-vf crop=largeurX:hauteurY:decalageX:decalageY	
Mettre à l'échelle	-1 conserve le ratio et flag peut être 'bicubic' : -vf scale=-1:480:flags='lanczos'	
Débruitage de vidéo	défaut = 4. Utiliser 2 pour les vidéos faiblement bruitées : -vf hqdn3d=2	

Commentaire	Option	
Désentralaçage (mcdeint et kerndeint sont moins efficaces)	-vf yadif	
Ajustement des couleurs	([mini/maxi|défaut]) : -vf mp=eq2=gamma:contraste:luminosite:saturation	[0.1/10|1:-2/2|1:-1/1|0:0/3|1]
Modifie le ratio, le lecteur affichera la vidéo en 16/9 peut importe la définition	-aspect 16:9	
Change le départ, la fin ou la durée	-ss départ en secondes -t durée en secondes -to fin en secondes	
Désactive les flux audios ou vidéos	-an sans audio -vn sans vidéo	
Audio Opus	bitrate -c:a libopus -b:a 64k	
Audio Vorbis, qualité	-c:a libvorbis	-q:a 0 (~64 kbps) -q:a 2 (~96 kbps -q:a 3 (~112 kbps standard) -q:a 4 (~128 kbps) -q:a 5 (~160 kbps)
Conversion audio multi-canaux vers stéréo	-ac 2	

Crop - découpe et recadrage vidéo

Tuto en cours - Commandes en tests à suivre... ;)

Les options

- out_w est la largeur du rectangle de sortie
- out_h est la hauteur du rectangle de sortie
- x et y indiquent le coin supérieur gauche du rectangle de sortie

Exemples

```
ffmpeg -i in.mp4 -vf "crop=out_w:out_h:x:y" out.mp4
```

Recadrer en 80x60

Original en 320×240

Faire un crop de 80×60, depuis la position (200, 100) :

```
ffmpeg -i in.mp4 -vf "crop=80:60:200:100" -c:a copy out.mp4
```

Recadrer le quart inférieur droit

```
ffmpeg -i in.mp4 -vf "crop=in_w/2:in_h/2:in_w/2:in_h/2" -c:a copy out.mp4
```

Ou bien :

```
ffmpeg -i in.mp4 -vf "crop=240:120:240:120" -c:a copy out.mp4
```

Découpez 20 pixels par le haut et 20 par le bas

```
ffmpeg -i in.mp4 -vf "crop=in_w:in_h-40" -c:a copy out.mp4
```

Le filtre centrera automatiquement le rognage si x et y sont omis.

Prévisualisation

Vous pouvez réaliser un recadrage (*heh heh heh*) et le prévisualiser en direct avec ffplay :

```
ffplay -i input -vf "crop=in_w:in_h-40"
```

De cette façon, vous pouvez expérimenter et ajuster votre recadrage sans avoir besoin d'encoder, de visualiser ni de recommencer.

Réparer un fichier mp4

Il peut arriver qu'un pompage sur le net ne soit pas correctement recodé (par ex. on aura un fichier vidéo et un fichier audio non fusionnés). youtube-dl annonce dans ce cas que la «conversion» n'a pas abouti; à mon avis il s'agit d'une fusion et non d'une conversion.

ffmpeg est capable de faire cette fusion:

```
ffmpeg -i <fichier vidéo> -i <fichier audio> <fichier final>.mp4
```

Liens

- Forum df - Améliorer la qualité des vidéos : https://debian-facile.org/viewtopic.php?pid=244397#p244397
- Forum df - Encoder en haute qualité : https://debian-facile.org//viewtopic.php?pid=329699#p329699

- Forum df - Conversion video au format DVD PAL : https://debian-facile.org/viewtopic.php?id=32071
- Site officiel: (en)http://ffmpeg.org/
- Documentation ffmpeg: (en)http://trac.ffmpeg.org/wiki
- Le blog de Jacques Cartier : http://www.jcartier.net/spip.php?rubrique9

[1)]

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !