FFmpeg

Table of Contents

FFmpeg est une application de lecture et encodage de vidéo. Très puissante comme son alter ego **Mencoder**, il assure en ligne de commande la possibilité de convertir les fichiers vidéo d'un format à un autre, dont le mts et m2ts.

Installation

Ubuntu nécessite que les dépôts universe soient activés, ainsi que les dépôts Medibuntu

Formats et extensions

FFmpeg est très pointilleux quant à l'extension des fichiers. Contrairement à la plupart des players, il n'examine pas le contenu pour découvrir le format réel.

Containers

Container	Extension	Argument
Matroska	.mkv, .mka	
MPEG-1	.mpg	
MPEG-2 TS	.ts	
MPEG-2 PS	.mpg	
MPEG-4	.mp4, .m4a, .m4v	
OGG	.ogg, ogm	
RIFF	.avi, .wav	

Connaître les conteneurs installés: Il est important de savoir ce que **FFmpeg** est capable de lire et d'écrire pour cela, utiliser la commande :

```
ffmpeg -formats
```

Après avoir appuyé sur la touche “Entrée”, on doit avoir une longue liste de formats (conteneurs) que reconnaît **ffmpeg**. Cette liste est organisée de manière simple, la première colonne indique si l'on peut lire (D) ou écrire (E) le format en question.

`DE matroska Matroska file format`, veut simplement dire que l'on peut à la fois lire et écrire les fichiers **.mkv** avec **ffmpeg**.

Codecs vidéo

Codec	Argument	CBR	ABR	VBR
pas de vidéo	-vn			
copie	-c:v copy			
Dirac	-c:v libridac			
DV	-c:v dvvideo	-	-	-
H264	-c:v libx264	-		-crf 23 [1)]
MPEG-1	-c:v mpeg1video			
MPEG-2	-c:v mpeg2video			
MPEG-4	-c:v mpeg4			
Theora	-c:v libtheora			
Xvid	-c:v libxvid			

Connaître les codecs installés: Comme pour les conteneurs, on peut avoir la liste des codecs disponibles. Pour ceci, dans un terminal faire :

```
ffmpeg -codecs
```

Retourne une liste avec là aussi les **D** et **E**. Mais aussi des **V**, **A** ou **S** qui permettent de savoir si un codec concerne la vidéo, le son ou les sous-titres.

Les codecs externes à **ffmpeg** sont désignées par **lib**quelquechose. Par exemple **vorbis** et **libvorbis** désignent le même codec, il s'agit juste de deux encodeurs différents. Si la qualité est importante, préférer les **lib**quelquechose aux quelquechose eux-mêmes.

Codecs audio

Codec	Argument	CBR	ABR	VBR	Commentaires
pas d'audio	-an				
copie	-c:a copy				
AAC (libaac)	-c:a libfaac	-	-ab 128k	-aq 10~500 (100)	La conversion à plus de 2 canaux est possible, mais avec les canaux dans le mauvais ordre Niveaux de qualité (**10**: pire, **100**: défaut, **500**: meilleur)
AAC (libfdk_aac)					Considéré comme bien meilleur que libaac N'est pas inclus dans les paquets officiels

Codec	Argument	CBR	ABR	VBR	Commentaires
AC3	-c:a ac3	-ab 192k	-	-	Ffmpeg n'inclus pas le support **ABR** ou **VBR** (en principe pas utilisé sur un DVD) Le bitrate doit normalement défini ainsi (en décomptant le LFE): **mono**: 96k, **stéréo**: 192k, **3ch**: 356k, **4ch**: 384k, **5ch**: 448k Le résultat mixe l'arrière gauche et droite comme un centre arrière (ce qui n'est pas normal)
FLAC					
MP3 (libmp3lame) (MPEG1/2 Layer 3)	-c:a libmp3lame	-ab 128k	-	-aq 0~9	Libmp3lame ne supporte pas plus de 2 canaux Ffmpeg n'inclus pas le support de l'**ABR** (average bitrate) Niveaux de qualité (**0**: meilleur, **2**: recommandé, **9**: pire)
PCM	-c:a pcm_s16le	-	-	-	
Vorbis (libvorbis)	-c:a libvorbis	-	-ab 128k	-aq -1~10 (5)	Qualité excellente La conversion à plus de 2 canaux ne fonctionne pas (son brouillé) Niveaux de qualité (**-1**: pire, **3**: défaut, **5**: recommandé, **10**: meilleur)

Tailles d'image principales

Nom	Taille	Aspect
hd1080	1920 x 1080	16:9
sxga	1280 x 1024	4:3
hd720	1280 x 720	16:9
xga	1024 x 768	4:3
hd480	852 x 480	16:9
4cif	704 x 576	4:3
vga	640 x 480	4:3
cif	352 x 288	4:3

Pour une liste complète voir cette section

Utilisation

Instruction générale

Utilisation Multithreads

Si **FFmpeg** est compilé avec la possibilité de faire du multithreading (la version par défaut d'Ubuntu le permet par exemple), il permet sans aucune difficulté d'utiliser les processeurs actuels à leur plein potentiel pour encoder un fichier, il suffit de passer l'instruction threads à ffmpeg à chaque encodage :

```
-threads 0
```

On peut spécifier le nombre de threads à utiliser comme ceci :

```
-threads N
```

N étant le nombre de threads à utiliser. On peut donc en utiliser seulement 3 sur un processeur Quad Core pour garder une grande réactivité sur une autre tâche.

On peut ensuite compléter par n'importe quelles instructions, et si le codec supporte le multithreading il utilisera automatiquement au mieux votre installation.

Fichiers d'entrée et de sortie

Le fichier d'entrée est spécifié par l'option **-i**, suivi du fichier à encoder. Pour aller plus vite on peut taper **-i**, espace, et glisser le fichier à encoder dans le terminal.

Ce qui donne :

```
-i "/home/vous/fichier vidéo.avi"
```

Pour donner le nom du fichier de sortie, il faudra juste le taper à la fin de la commande ffmpeg.

ffmpeg -bla bla -bla bla “/home/vous/fichier vidéo de sortie.mkv”

À moins de savoir exactement ce qu'on fait, ne rien mettre après le nom du fichier final.

Choisir le conteneur

Le conteneur est choisi automatiquement via l'extension du fichier. C'est-à-dire que si le fichier final se termine par **.mkv** le format sera automatiquement mis en matroska sans demander confirmation.

Mais on peut préciser le format via l'option **-f**. On peut faire suivre “-f” de tous les formats dans la liste que donne `ffmpeg -formats`. Par exemple :

```
-f matroska
-f avi
```

Extractions

Extraction de la bande Audio

Taper la commande :

```
ffmpeg -i <nom du fichier video> -vn -acodec libmp3lame bande_son
```

Cette commande n'extraira pas la piste audio originale, mais la transcodera en mp3, ce qui a des effets néfastes sur la qualité (mais peut servir un intérêt de compatibilité).

On lui préférera de manière générale cette méthode pour extraire la piste audio originale (sans transcodage) de la vidéo :

- D'abord se renseigner sur le type de flux audio, via Mediainfo ou la commande :

```
ffmpeg -i <votre vidéo>
```

On peut glisser-déposer le fichier dans le terminal plutôt que de recopier entièrement son chemin d'accès. Cela limite le risque d'erreurs avec les espaces etc.

Et regarder l'avant-dernière ligne de ce qui ressort, par exemple :

```
...
Stream #0.1: Audio: aac, 44100 Hz, stereo, s16, 69 kb/s
At least one output file must be specified
```

- Puis appuyer sur Flèche haut (pour ré-entrer automatiquement la commande précédente) ou la retaper puis compléter :

```
ffmpeg -i <votre fichier video> -vn -acodec copy '(destination et) nom du
fichier audio créé'.EXTENSION
```

Remplacer EXTENSION par :

- **.m4a** (préférable) ou **.mp4** pour un flux audio en **aac** (comme dans l'exemple), la majorité des

vidéos Flash (FLV) récupérées d'Internet ainsi que de nombreux MKV utilisant ce format. Ne pas mettre **.aac** comme extension, sous peine d'avoir un fichier en raw **aac** qui ne sera pas (ou problématiquement) lisible. Le **AAC** doit être placé dans un conteneur pour être lu, d'où le **.m4a** qui est la version audio du **MP4**.
- **.mp3** pour un flux audio en mp3,
- **.ogg** pour un flux audio en vorbis...

Extraction de la bande Video

Taper la commande :

```
ffmpeg -i '(chemin et)nom du fichier video' -an -vcodec copy '(chemin et)
nom de la bande video extraite'.EXTENSION
```

Cette commande ne fonctionnera pas tant qu'on ne ne précise pas l'extension du fichier à créer, à savoir son conteneur. Ici cependant, on peut se contenter d'utiliser la même extension que le fichier source (.mkv, .ogm, .avi, etc), ou encore changer de conteneur (passer de .flv à .mkv par exemple) sans pour autant transcoder le flux vidéo, ce qui est bien pratique.

Extraction des soustitres

Ffmpeg ne peut pas convertir des sous-titres au format bitmap vers un format texte cela nécessite d'utiliser un logiciel OCR (reconnaissance optique de caractères) tel que **vobsub2srt** mais pour cela il faut procéder en 3 étapes:

- extraire la bande des sous titres vers un fichier sup

```
for i in *.mkv; do ffmpeg -i $i -map 0:s -c:s copy "$i.sup"; done
```

- convertir le fichier sup en idx/sub

```
for i in *.sup; do java -jar BDSup2Sub.jar $i "$i.sub"; done
```

- procéder à la reconnaissance optique (fichier srt)

```
for i in *.sub; do fbnam=$(basename "$i" .sub); vobsub2srt --tesseract-lang
eng $fbname; done
```

Encodage

Instructions d'encodage Audio

Il faut paramétrer le son. Et c'est très simple !

On peut choisir le codec.

```
-acodec "nom du codec"
```

On peut choisir le bitrate à utiliser pour le son.

```
-ab 128000
```

ou

```
-ab 128k
```

On peut choisir une qualité plutôt qu'un bitrate, utilisez :

```
-aq 4
```

Les paramètres suivants sont optionnels (si l'on ne précise rien, les caractéristiques seront identiques au fichier d'origine).

On peut choisir la fréquence d'échantillonnage comme ceci :

```
-ar 44100
```

Le nombre de canaux (stereo/mono/surround) :

```
-ac 2
```

2 étant le nombre de voies (2 : stéréo, 6 : 5.1, etc ...)

Instructions d'encodage vidéo

Le codec vidéo se choisit avec l'instruction :

```
-vcodec
```

Les codecs sont nombreux, et il faudra ajouter pas mal d'instructions pour que le codec ne fasse pas n'importe quoi.

Codec libx264

Pour l'utiliser, il est fortement conseillé d'utiliser les preset fournis par **ffmpeg**. Ce sont des “packs” d'options pré-configurées.

`-vpre` suivi d'un preset qui donnera les paramètres par défaut de l'encodeur à **ffmpeg** suivant plusieurs profils.

complexité et features activés dans le fichier vidéo de sortie		**temps nécessaire à l'encodage**		**temps nécessaire à un encodage sans perte de qualité d'image**	
preset [2)]	**description**	**preset**	**description**	**preset**	**description**
baseline	à utiliser lorsque le matériel n'arrive pas à décoder les fichiers encodés avec les autres presets.	**ultrafast**	le plus rapide, mais donne la moins bonne qualité d'image.	**lossless_max**	
main	inférieur au profile **high**	**superfast**		**lossless_ultrafast**	
ipod320		**veryfast**		**lossless_fast**	
ipod640		**faster**		**lossless_medium**	
		fast		**lossless_slow**	
		medium		**lossless_slower**	
		slow			
		slower			
		veryslow	La meilleure qualité au prix de la vitesse		
		placebo	le plus lent, mais pas la meilleure qualité (optimisé pour les benchmarks)		

Pour un encodage **libx264**, la ligne suivante permet de spécifier que le fichier de sortie doit pouvoir être lu sur à peu près n'importe quel appareil (cela donnera une moins bonne qualité d'image ou un fichier de sortie plus gros) et un encodage très lent (pour essayer de maximiser la qualité d'image) :

```
-vcodec libx264 -vpre baseline -vpre veryslow
```

Il ne reste plus qu'à dire au codec la qualité ou le débit que l'on attend pour la vidéo finale.

Codec libvpx

Le codec **libvpx** (c'est-à-dire **VP8**) propose des options par défaut de très mauvaise qualité... Il va falloir préciser quelques options non obligatoires, mais grandement conseillées:

- **-rc_lookahead 16** permet au codec de regarder les prochaines images pour adapter ces paramètres en conséquence.
- **-g 360** demande au codec de créer une image de référence toutes les 360 images.
- **-level 116** permet de passer en mode “lent”. Cela active des améliorations graphiques mais augmente le temps d'encodage. On peut la modifier en **216** pour gagner un peu de temps ou **214**, mais on perdra en qualité.

Pour un encodage **libvpx** la ligne suivante donnera une meilleure qualité:

```
-rc_lookahead 16 -keyint_min 0 -g 360 -skip_threshold 0 -level 116
```

Encodage à bitrate constant - CBR

Faire une vidéo en bitrate (débit) constant revient à rendre les scènes en mouvement moins belles que les scènes statiques, ce type d'encodage n'est pas conseillé !

Pour choisir un bitrate, il suffit de le préciser après l'instruction :

```
-b
```

Le bitrate peut être précisé en bits/s ou en kbits/s :

```
-b 2700000
```

```
-b 2700k
```

Encodage à qualité constante

Faire une qualité constante est très simple, il suffit de choisir un quantiser identique pour la qualité maximale et minimale de la vidéo avec les instructions suivantes :

```
-qmin 20 -qmax 20
```

Voilà, c'est tout ! On peut changer le nombre par un plus élevé (moins bonne qualité) ou plus bas (meilleure qualité) On peut mixer ces paramètres avec un bitrate pour créer un bitrate constant avec des limites de qualité minimum et maximum.

Pour un encodage en **h264**, il est conseillé d'utiliser l'option **-crf** (non décrite dans le manuel) plutôt que “-qmin” et “-qmax”.

```
-crf 20
```

Encodage 2-pass

L'encodage double passe, multi-passes, 2pass ou autres, est une technique d'encodage qui mêle les deux techniques d'encodage précédentes.

Cette technique permet d'encoder une première fois à bitrate constant pour faire un index des moments qui vont être encodés avec une qualité exécrable. Ensuite lors du deuxième encodage, la qualité sera équilibrée en baissant la qualité globale des meilleures scènes pour augmenter la qualité des plus mauvaises, ce qui revient quasiment au même que d'avoir une qualité constante. Au final on aura donc une vidéo plus uniforme niveau qualité, tout en ayant le contrôle via un bitrate.

Pour lancer un encodage 2 passes il faut préciser beaucoup de paramètres et lancer l'encodage deux fois.

- Pour la première pass

```
-b 2700000 -qmin 1 -qmax 31 -minrate 0 -maxrate 9000000 -pass 1 -passlogfile pass1.fpf
```

Les paramètres sont simples : on choisit le bitrate moyen voulu, le bitrate minimum (-minrate) que l'on peut atteindre (pourquoi pas 0), et le bitrate maximum (-maxrate) (là aussi, si la vidéo n'a rien à voir avec le streaming, mettre le plus haut possible). Il faut aussi préciser un quantiser minimum et maximum ! Pourquoi ? Parce que certains codecs comme le VP8 ont des bugs si on ne le fait pas...

Certains codecs ont leur maxi à 31 d'autres plus haut. Comme le VP8 et le x264 qui vont jusqu'à 51.

-pass 1 signifie qu'on effectue la première passe. et **passlogfile** est le fichier “pass” que l'on écrira temporairement.

- Pour la seconde passe, utiliser les mêmes paramètres à une exception ! (utiliser la flèche de haut pour retourner dans l'historique du terminal)

```
La seule chose à modifier est le paramètre -pass 1 à changer en -pass 2.
```

Lorsqu'on utilise un encodage en **h264** et que l'on veut une certaine qualité d'image et non une taille de fichier, utiliser l'option `-crf` sans spécifier de bitrate. En revanche, si on veut une taille de fichier final bien précise, utiliser un encodage en 2 passes.

Instructions d'encodage sous-titres

Pour forcer le codec des sous-titres ('copier' pour copier le flux):

```
-scodec [codec de sous-titres]
```

La liste de codecs ci dessous est à titre indicatif, cela dépendent en partie du système d'exploitation utilisé, sous Linux, on peut exécuter la commande suivante pour obtenir une liste de tous les codecs de sous-titres pris en charge par la version de ffmpeg:

```
ffmpeg -codecs | grep "^...S"
```

Pour limiter les codecs de sous-titres que ffmpeg est capable d'encoder:

```
ffmpeg -codecs | grep "^..ES"
```

Pour limiter la liste aux sous-titres que ffmpeg peut décoder:

```
ffmpeg -codecs | grep "^.D.S"
```

Image-based sub-title codecs		**Text-based subtitle codecs**	
codec ID	**Description**	**codec ID**	**Description**
dvb_subtitle	DVB subtitles	mov_text	MOV subtitles with metadata headers.
dvd_subtitle	DVD subtitles	ass	ASS (Advanced SSA) subtitle
hdmv_pgs_subtitle	HDMV Presentation Graphic Stream subtitles pgssub	ssa	SSA (SubStation Alpha) subtitle
		webvtt	WebVTT subtitle
		hdmv_text_subtitle	HDMV Text subtitle
		jacosub	JACOsub subtitle
		microdvd	MicroDVD subtitle
		mpl2	MPL2 subtitle
		pjs	PJS (Phoenix Japanimation Society) subtitle
		realtext	RealText subtitle
		sami	SAMI subtitle
		stl	Spruce subtitle format
		subrip	SubRip subtitle
		srt	SubRip subtitle with embedded timing
		subviewer	SubViewer subtitle
		subviewer1	SubViewer v1 subtitle
		vplayer	VPlayer subtitle

Tous les codecs de sous-titres ne peuvent pas être utilisés pour tous les formats de conteneur vidéo:
- pour les conteneurs **MKV** : copy, ass, srt, ssa
- pour les conteneurs **MP4**: copy, mov_text
- pour les conteneurs **MOV**: copy, mov_text

Transformer la vidéo

On peut transformer la vidéo avec **ffmpeg**, modifier la résolution, changer l'aspect ratio (4/3, 16/9), couper autour de l'image, etc.

Changer la résolution

On peut changer la résolution avec `-s` et en spécifiant la résolution. On peut spécifier manuellement (1280×720) ou automatiquement (hd720)

Voici la liste des abréviations :

cif		**vga**		**xga**		**hd**	
sqcif	128×96	qqvga	160×120	xga	1024×768	hd480	852×480
qcif	176×144	qvga	320×240	uxga	1600×1200	hd720	1280×720
cif	352×288	vga	640×480	sxga	1280×1024	hd1080	1920×1080
4cif	704×576	svga	800×600	wxga	1366×768		
16cif	1408×1152	wvga	852×480	wsxga	1600×1024		
		cga		wuxga	1920×1200		
		cga	320×200	qxga	2048×1536		
		ega		woxga	2560×1600		
		ega	640×350	qsxga	2560×2048		
				wqsxga	3200×2048		
				wquxga	3840×2400		
				hsxga	5120×4096		
				whsxga	6400×4096		
				whuxga	7680×4800		

Changer le nombre d'images par seconde

Le framerate, ou FPS se change comme ceci :

```
-r FPS
```

Par exemple :

```
-r 25
```

Changer l'aspect ratio

```
-aspect aspect
```

aspect étant **4:3**, **16:9** ou **1.3333**, **1.7777**. On peut également indiquer des aspects spéciaux.

Conversion

Convertir une vidéo en WebM (VP8+Vorbis) en deux passes

Ces deux commandes permettent d'encoder en deux passes un fichier **WebM**. Pour que cela fonctionne il faut avoir au minimum FFMpeg 0.6 !

```
avconv -i 'fichier source' -threads 0 -vcodec libvpx -b 1500k -minrate 0 -
maxrate 9000k -rc_lookahead 16 -keyint_min 0 -g 360 -skip_threshold 0 -level
116 -qmin 1 -qmax 51 -an  -pass 1 -passlogfile pass1.fpf pass1.webm
avconv -i 'fichier source' -threads 0 -vcodec libvpx -b 1500k -minrate 0 -
maxrate 9000k -rc_lookahead 16 -keyint_min 0 -g 360 -skip_threshold 0 -level
116 -qmin 1 -qmax 51 -acodec libvorbis -ab 192k -pass 2 -passlogfile
pass1.fpf "fichier final.webm"
```

Convertir un fichier webm en .mp4

Le format **webm** est largement utilisé sur internet, en particulier pour youtube. Vous pouvez télécharger ces vidéos de différentes façons, par exemple via youtube-dl. Pour voir ces vidéos sur une télévision, il faudra sans doute modifier le format (il n'est généralement pas reconnu). Pour passer la vidéo **webm** en **.mp4** :

```
ffmpeg -i ma-video.webm ma-video.mp4
```

La qualité de sortie est alors en qualité moyenne.

On peut spécifier une autre qualité de sortie avec l'option `-crf. "0"` donne la meilleur qualité (et le plus gros fichier), “51” donne la moins bonne qualité (et le plus petit fichier). Pour une qualité maximum (mais le fichier peut-être 10 fois plus gros que le webm initial !) :

```
ffmpeg -i ma-video.webm -crf 0 ma-video.mp4
```

Convertir une vidéo en x264 en qualité constante

Cette commande permet d'encoder un fichier vidéo en H.264 en qualité constante :

```
ffmpeg -threads 0 -i "fichier d'origine" -f matroska -vcodec libx264 -vpre
normal -crf 20 -acodec libvorbis -ab 192k -ar 44100 -ac 2 "fichier
final.mkv"
```

Convertir le format FLV en AVI

Cette conversion est utile lorsqu'on télécharge des vidéos en streaming (YouTube, Google Video, framatube.org !-), etc.).

```
ffmpeg -i nom_du_fichier.flv -f avi nom_du_fichier.avi
```

Convertir le format AVI en MPEG

Un exemple de commande, et d'options, pour faire un DVD (donc un format MPEG) depuis un fichier avi :

```
ffmpeg -i ma_video.avi -target pal-dvd -aspect 16:9 -qscale 0
mon_dvd_video.mpg
```

Où :

- **-i ma_video.avi** est le fichier départ ;
- **-target pal-dvd** le format de sortie ;
- **-fs 2000000000** la taille maximale du fichier de sortie, en bits (ici 2 Gbit) ;
- **-aspect 16:9** le ratio widescreen (avec les franges en haut et en bas).

Convertir le format AVI en DV

Cette conversion est utile pour faire du montage vidéo, dans Kino par exemple.

```
ffmpeg -i video.avi -target pal-dv video.dv
```

Ici le chemin n'est pas précisé. Le fichier doit se trouver dans le dossier personnel pour que cela fonctionne.

Convertir un grand nombre de fichiers Vidéo

```
#!/bin/bash
for i in *.avi
do
compte=$(($compte+1))
echo "# Traitement du fichier $i"
echo "# Compression vers `basename \"$i\" .avi`.mp4"
ffmpeg -i "$i" -c:v libx264 -crf 23 -maxrate 1M -bufsize 2M "`basename
\"$i\" .avi`.mp4";
done
```

Convertir un grand nombre de fichiers Vidéo vers le format MP3

Ce script permet d'automatiser la conversion de vidéos de plusieurs formats (Mpeg4, Avi, Flv etc...) vers le format MP3. Plusieurs formats de vidéos peuvent se trouver dans le même dossier au moment de la conversion.

```
#!/bin/bash
#Fichier "Conversion Script "

echo "Création de la liste de fichiers..."
cd CHEMIN_VERS_LES_VIDEOS ( par exemple ~/Downloads ou ~/Videos )
ls *.* > to_convert
echo "Préparation de ffmpeg..."
while read line
do
        echo "Traitement de $line programmé"
done < to_convert
#On s'assure que chaque vidéo a été détectée
while read line
do
    sed -i -e "s/(//g" to_convert
    sed -i -e "s/)//g" to_convert
    sed -i -e "s/\[//g" to_convert
    sed -i -e "s/\]//g" to_convert
    sed -i -e "s/\ /\./g" to_convert
done < to_convert
#Ceci permet de contourner le problème posé par d'éventuels caractères spéciaux dans le titre de la vidéo traitée ( tels que "[","]","(",")" ou " ")
echo "_____________________________________________________"
while read line
do
    rename=`echo ${line%%.*}`
    original=`ls CHEMIN_VERS_LES_VIDEOS| grep $rename`
    mv CHEMIN_VERS_LES_VIDEOS/"$original" CHEMIN_VERS_LES_VIDEOS/"$line"
    echo "$line loaded"
done < to_convert
echo "_____________________________________________________"
echo "La conversion a démarré : "
while read line
do
    sdot=`echo ${line%.*}`
    echo "CONVERSION EN COURS :[ $line ]"
    ffmpeg -ab 192k -i `echo "$line"` `echo "$sdot.mp3"` 1>.ffmpeg_last_log 2>/dev/null
        #Le fichier log de la conversion seront stockés dans le dossier courant sous le nom ".ffmpeg_last_log"
    mv $line /RÉPERTOIRE OU SERONT STOCKÉES LES VIDÉOS SI ON SOUHAITE LES CONSERVER ( COMMENTER POUR ENLEVER )
```

```
    mv "$sdot.mp3" /RÉPERTOIRE OU SERA DÉPLACÉ LE FICHIER *.MP3 GÉNÉRÉ ( COMMENTER POUR ENLEVER )
    echo "→ OK !"
done < to_convert
rm to_convert
```

Convertir tous les fichiers Windows Media Audio (WMA) en mp3

Dans un dossier contenant des fichiers au format wma. La boucle for lance la conversion des fichiers. Il suffira de supprimer les fichier wma une fois la conversion terminée.

```
for i in *.wma; do ffmpeg -i "$i" -ab 256k "${i%wma}.mp3"; done
```

Convertir le format AVI en MP4 (PSP)

Cette conversion est utile pour lire vos vidéos sur une console portable PSP. En ligne de commande, taper :

```
ffmpeg -i video.avi -f psp -r 29.97 -b 768k -ar 24000 -ab 64k -s 480x160 m4v00001.mp4
```

Si on génére plusieurs fichiers vidéos, ceux-ci doivent avoir un nommage particulier pour pouvoir être lus sur la console (m4v00001.mp4, m4v00002.mp4, m4v00003.mp4, et ainsi de suite).

Pour encoder une vidéo qui est au format **4/3**, il faut choisir une taille de **320×240** pour conserver les proportions.

Pour avoir une bonne qualité avec une résolution **480×272**, avec un débit de **768 kbps**, firmware conseillé **3.71** ou supérieur :

```
ffmpeg -i video_en_entrée.avi -r 29.97 -vcodec h264 -s 640x480 -aspect 16:9 -flags +loop -cmp +chrom
```

Convertir un mkv en gif animé

```
ffmpeg -i input.mkv -pix_fmt rgb24 -r 10 -s 320x240 output.gif
```

- **-pix_fmt rgb24** pour 'encoder' le gif
- **-r 10** pour avoir 10 images par seconde
- **-s 320×240** pour avoir une résolution de sortie de 320×240.

On peut également choisir de découper une portion de vidéo avec les options **-ss** et **-t**. Par exemple :

```
```

```
ffmpeg -i input.mkv -ss 00:01:00.00 -pix_fmt rgb24 -r 10 -s 320x240 -t
00:00:10.00 output.gif
```

Permet de convertir 10 secondes de vidéo à partir de 1 minute.

Convertir WMV en MP4

Le format **WMV** est le format vidéo **Windows Media**. Le vidéos dans ce format sont destinées à être vues sur le Lecteur **Windows Media**. Le format **MP4** est un format conteneur audio et vidéo qui est conçu pour être vu sur plusieurs lecteurs vidéo, y compris les téléphones cellulaires et les iPod .

Les codecs binaires **Win32** sont des filtres propriétaires de compression/décompression de flux audio et vidéo 32 bits utilisés par les outils multimédia sous Microsoft® Windows®. Pour convertir des fichiers **WMV** en fichiers **MP4** il faut installé **w32codecs** sur le système d'exploitation.

Installation par paquet deb*

Il existe un paquet debian uniquement pour les architectures 32 bits.

```
sudo apt-get install w32codecs
```

Installation manuelle

Télécharger l'archive essential-<arch>-20071007.tar.bz2 (où 20071007 représente l'archive la plus récente, en date d'avril 2009) correspondant à l'architecture :

- pour les systèmes 32 bits : essential-20071007.tar.bz2
- pour les systèmes 64 bits : essential-amd64-20071007.tar.bz2
- pour les systèmes PowerPC : essential-ppc-20071007.tar.bz2

Exécuter les commandes suivantes :

```
tar jxf ~/essential-20071007.tar.bz2
sudo mv ~/essential-20071007 /usr/lib/win32
```

Pour convertir les vidéos

Utiliser la commande suivante :

```
ffmpeg- i original.wmv -b 600 - s 320x240 -ab 128k - vcodec mpeg4 -ab 128
acodec aac output.mp4
```

Puis taper la commande `xine output.mp4` pour tester la conversion.

Réaliser une rotation d'une vidéo

Pour réaliser la rotation d'une vidéo pourtant prise en “paysage” qui se visualise en mode “portrait”, on utilise l'option **filtergraph** (-vf) avec transpose.

Le chiffre passé avec transpose correspond à:

chiffre	filtrage appliqué	désignation en anglais
0	-90° (sens anti-horaire) puis symétrie verticale (par défaut)	90CounterClockwise and Vertical Flip (default)
1	90° (sens horaire)	90Clockwise
2	-90° (sens anti-horaire)	90CounterCLockwise
3	90° (sens horaire) puis symétrie verticale	90Clockwise and Vertical Flip (default)

Voici la commande pour -90°:

```
ffmpeg -i vidéo_originale.mp4 -vf "transpose=2"
vidéo_correctement_orienté.mp4
```

du fait du codec audio utilisé, **aac**, une erreur peut survenir en précisant d'ajouter **-strict -2**:

```
ffmpeg -i vidéo_originale.mp4 -strict -2 -vf "transpose=2"
vidéo_correctement_orienté.mp4
```

Effectuer une capture vidéo (screencast) de l'écran

Après avoir exécuté cette commande, appuyer sur **q** pour arrêter la capture.

```
ffmpeg -f x11grab -s 1920x1080 -r 25 -i :0.0 screencast.mp4
```

- **-f** force la capture de l'écran (x11grab)
- **-s** définit la taille de la capture, doit être inférieur ou égale à la taille réelle d'affichage!
- **-r** définit le nombre d'images pas seconde : 12.5, 25, 30 sont les plus courants, pour un screencast de tutoriel, on peut descendre à 5, mais du fait de la compression, l'économie sur la taille mémoire n'est pas proportionnel.
- **-i:0:0** correspond au nombre du serveur X11, peut permettre de définir un offset auquel cas, il faut réduire la taille d'autant avec l'option **-s**

Si une autre session est active (ou l'a été) il se peut que ça ne fonctionne pas avec **-i:0:0**, en effet, le premier nombre correspond au numéro de session qui s'affiche entre parenthèse dans ce que retourne la commande who et qui correspond à l'ordre d'ouverture de session (:0 pour le premier connecté, :1 pour le deuxième …). Si c'est le cas, utiliser la commande ci-dessous qui ajuste le numéro de session:

```
ffmpeg -f x11grab -s 1920x1080 -r 25 -i $(who | awk '{print
substr($5,2,2)}').0 screencast.mp4
```

Pour une capture d'écran Hi-Fi, son et vidéo sans perte, avec Pulse Audio comme serveur de son :

```
ffmpeg -f alsa -ac 2 -i pulse -f x11grab -r 25 -s 1024x768 -i :0.0 -acodec
flac -vcodec libx264 -preset ultrafast -qn 0 output.mkv
```

Pour créer une vidéo à partir de photos

À partir des images img001.png, img002.png, img003.png:

```
ffmpeg -framerate 24 -i img%03d.png output.mp4
```

Exemples de Script de conversion

Convertir plusieurs fichiers

Dans l'exemple suivant le flux

1. video 0 (1er flux) est compressé en h264,
2. audio 0 (1er flux) et 1 (2eme flux) sont compressés en aac
3. subtitle 1 (2eme flux) n'est pas compressé (copy)
4. avec un preset hd720

```
for i in *; do ffmpeg -n -loglevel error -i "$i" -s hd720 -map 0:v:0 -map
0:a:0 -map 0:a:1 -map 0:s:1 -c:v libx264 -crf 23 -preset faster -c:a aac -
c:s copy "${i}.mkv"; done
```

Combiner plusieurs sources

ffmpeg permet de combiner plusieurs sources (fichiers), par exemple si on a un media en VF uniquement et que l'on veut y inclure les sous-titres d'un media VOST on peut utiliser cette commande:

```
ffmpeg -n -loglevel error -i film1-vf.mkv -i film1-vost.mkv -s hd480 -map
0:v:0 -map 0:a:0 -map 1:s:0 -c:v libx264 -crf 23 -preset faster -c:a aac -
c:s copy "film1.mkv"
```

- **-map 0:v:0** indique de traiter la piste **video** de la première source (dans l'ordre des **-i**)
- **-map 0:a:0** indique de traiter la piste **audio** de la première source (dans l'ordre des **-i**)
- **-map 1:s:0** indique de traiter la piste **subtitle** de la seconde source (dans l'ordre des **-i**)

Lorsque les sous-titres sont incrustés dans les images du flux source (par exemple une video mp4) il faut insérer l'audio du flux contenant la version souhaitée. Par exemple si film1-vost.mkv contient le format audio souhaité et film1-vost.mp4 contient les sous titres incrustés (pas de canal subtitle lorsqu'on utilise ffprobe), il faut faire:

```
ffmpeg -n -loglevel error -i film1-vf.mkv -i film-vost.mp4 -s hd480
-map 1:v:0 -map 0:a:0 -c:v libx264 -crf 23 -preset faster -c:a aac
"film1.mkv"
```

* **-map 1:v:0** indique de prendre la piste **video** de la seconde source (dans l'ordre des **-i**)
* **-map 0:a:0** indique de prendre la piste **audio** de la première source (dans l'ordre des **-i**)
* il n'y a pas de piste **subtitle** distinct, ceux-ci sont incrustés dans le flux video

Choix des protocoles

Dans l'exemple suivant:

1. la première piste video 0 est compressée en h264,
2. la première piste audio 0 et la deuxième piste audio 1 sont compressées en aac
3. la deuxième piste subtitle 1 est compressée en dvd_subtitle (image-based subtitles)
4. avec un preset hd720

```
for i in *; do ffmpeg -n -loglevel error -i "$i" -s hd720 -map 0:v:0 -map
0:a:0 -map 0:a:1 -map 0:s:0 -c:v libx264 -crf 23 -preset faster -c:a aac -
c:s dvd_subtitle "${i}.mkv"; done
```

Détection des flux

ffprobe permet de récupérer les informations sur un flux multimédia, ces information sont typées:

1. `Stream #(...).fre.: Audio:` identifie un le flux audio
2. `Stream #(...).fre.: Subtitle:` identifie le flux de sous-titres

Il est donc possible d'utiliser ces informations, pour ne trairter que les flux souhaités, par exemple le script suivant, permet d'extraire les flux audios en anglais (en) et en françsis (fre) et les flux de sous-titres en français (fre):

```
for i in *; do ffmpeg -n -loglevel error -i "$i" -s hd720 -map 0:v:0
`ffprobe "$i" 2>&1 |sed -rn 's/Stream #(...).fre.: Audio: .*$/-map \1/p'`
`ffprobe "$i" 2>&1 |sed -rn 's/Stream #(...).eng.: Audio: .*$/-map \1/p'`
`ffprobe "$i" 2>&1 |sed -rn 's/Stream #(...).fre.: Subtitle: .*$/-map \1/p'`
-c:v libx264 -crf 23 -preset faster -c:a aac `ffprobe "$i" 2>&1 |sed -rn
's/Stream #.:..fre.: Subtitle: .*$/-c:s copy/p'` "${i}.mkv"; done
```

- `for i in *; do ... ; done` : boucle de traitement des objets dans le dossier
- `ffmpeg -n -loglevel error -i "$i" -s hd720 -map 0:v:0`: profil d'encodage video (stream video 0)
- `ffprobe "$i" 2>&1 |sed -rn 's/Stream #(...).fre.: Audio: .*$/-map \1/p'` : bloc pour détecter les pistes audio fre
- `ffprobe "$i" 2>&1 |sed -rn 's/Stream #(...).eng.: Audio: .*$/-map \1/p'` : bloc pour détecter les pistes audio eng
- `ffprobe "$i" 2>&1 |sed -rn 's/Stream #(...).fre.: Subtitle: .*$/-map \1/p'` : bloc pour détecter les sous-titres fre
- `-c:v libx264 -crf 23 -preset faster -c:a aac` : codec d'encodage video
- `ffprobe "$i" 2>&1 |sed -rn 's/Stream #.:..fre.: Subtitle: .*$/-c:s copy/p'` : bloc pour les codecs d'encodage des sous-titres

Limitation de la mémoire tampon

Lors du transcodage de flux audio et/ou vidéo, ffmpeg ne commencera pas à écrire dans la sortie tant qu'il n'aura pas un paquet pour chacun de ces flux. En attendant que cela se produise, les paquets pour les autres flux sont mis en mémoire tampon. Cette option définit la taille de ce tampon, en paquets, pour le flux de sortie correspondant.

La valeur par défaut de cette option doit être suffisamment élevée pour la plupart des utilisations, cependant il peut arriver que ffmpeg échoue à transcoder certains fichiers, avec le message suivant:

```
Too many packets buffered for output stream 0:0.
(...)
Conversion failed!
```

Dans ce cas il faut ajouter l'option `-max_muxing_queue_size 1024` juste après le flux concerné par l'erreur, par exemple dans l'exemple suivant on utilise l'option afin de limiter la taille du buffer des flux audios:

```
for i in *; do ffmpeg -n -loglevel error -i "$i" -s hd720 -map 0:v:0
`ffprobe "$i" 2>&1 |sed -rn 's/Stream #(...).fre.: Audio: .*$/-map \1 -
max_muxing_queue_size 1024/p'` `ffprobe "$i" 2>&1 |sed -rn 's/Stream
#(...).eng.: Audio: .*$/-map \1 -max_muxing_queue_size 1024/p'` `ffprobe
"$i" 2>&1 |sed -rn 's/Stream #(...).fre.: Subtitle: .*$/-map \1/p'` -c:v
libx264 -crf 23 -preset faster -c:a aac `ffprobe "$i" 2>&1 |sed -rn
's/Stream #.:..fre.: Subtitle: .*$/-c:s copy/p'` "${i}.mkv"; done
```

1)

22: excellent, **24**: bon, **26** ... - CRF 23 est le meilleur compromis, un peu trop d'artefacts de

compression en CRF 24
[2)]

Le profile **high** n'est pas disponible en option car il est le preset par défaut.