



+

¿Te gusta el diseño web? ¡Echa un vistazo a la documentación de [Laravel](#)

Recibe las actualizaciones de **Emezeta.com** en tu correo:

¡O sígueme en

[Twitter!](#)

25 13min

## Cómo usar ffmpeg para editar video

FFmpeg es una potente herramienta con la que podemos convertir entre formatos de video, rotar, reducir tamaño, calidad o resolución, y muchas otras operaciones, todo ello automatizado desde una terminal.

Publicidad

Una de mis herramientas preferidas para conversión multimedia es [ffmpeg](#). Se trata de una herramienta de línea de comandos que permite realizar multitud de tareas relacionadas con video, audio o incluso imágenes.

Aunque su uso no es excesivamente complejo, la gran cantidad de parámetros, combinaciones y el inmenso abanico de formatos y sus características, hacen que el uso de ffmpeg sea poco intuitivo cuando empezamos a utilizarlo (o incluso más tarde, si no lo tenemos bien claro).

En este artículo vamos a explicar como utilizar esta herramienta de forma sencilla, dando un repaso a las tareas de edición de video más comunes y prácticas, viendo ejemplos prácticos para aplicar.

# Instalación de ffmpeg

La herramienta ffmpeg es multiplataforma, por lo que puede instalarse en cualquier sistema, ya sea [GNU/Linux](#), [Windows](#) o [Mac](#). En el caso de Windows, la página tiene varias versiones (static, shared y dev). Aconsejo instalar la versión static de 64 bits (si nuestro sistema es de 64 bits), que contiene una carpeta bin con el archivo ffmpeg.exe que nos interesa. Dicho archivo puede colocarse en una carpeta que esté en el PATH del sistema, o simplemente colocarlo en una carpeta del sistema como C:\windows, por ejemplo.

Si acostumbras a trabajar en una terminal de texto bajo windows, mira primero el artículo [Cómo mejorar la terminal de Windows](#) donde encontrarás algunos consejos para instalar ConEmu y Chocolatey y tener una potente terminal de texto.

En el caso de GNU/Linux puedes utilizar el comando apt-get install ffmpeg. En Windows, si tienes ya instalado Chocolatey, puedes usar choco install ffmpeg.

## Obtener información de un video

Una vez instalado, podemos ver información sobre nuestros videos de la siguiente forma:

```
ffmpeg -i video.mp4
```

El parámetro -i indica que se va a indicar un archivo de entrada (en nuestro caso, el video) y a continuación, dicho archivo. Obtenremos que ffmpeg nos devuelve gran cantidad de información. La parte clave es la que resalto a continuación:

- Naranja: Se trata del comando que escribimos en la terminal de texto para lanzar ffmpeg y realizar alguna acción. En este caso, obtener información del video.
- Verde/Amarillo: Características del archivo en general (duración y [bitrate W](#) del video+audio, por ejemplo).
- Azul: Información sobre el canal de video: codec, resolución, proporción de aspecto, calidad de video, [fps W](#), etc...
- Rojo: Información sobre el canal de audio: codec, [frecuencia de muestreo W](#), calidad de audio, etc...

Con esto puedes obtener información sobre un archivo de video y sus características.

## Conversión entre formatos de video

Uno de los puntos fuertes de ffmpeg es que permite realizar prácticamente cualquier tipo de conversión entre los diferentes formatos de video y audio (que no son pocos). El único requisito es tener una idea general de los diferentes codecs que existen, por lo que recomiendo altamente leer [Formatos de video: todo lo que deberías saber ↗](#) y [Formatos de audio: todo lo que deberías saber ↗](#) antes de comenzar a utilizar ffmpeg. De esa forma, teniendo clara la diferencia de un contenedor y un codec, de los diferentes formatos que existen y otras cuestiones, es muy sencillo utilizar ffmpeg.

Podemos realizar conversiones aprovechándonos de las extensiones comunes de ciertos formatos de video, ya que ffmpeg los detecta automáticamente. Por ejemplo:

```
ffmpeg -i video_original.avi video_destino.mp4
```

En este ejemplo, le indicamos a ffmpeg que el formato de entrada es video\_original.avi (mediante el parámetro -i) y que lo convierta a un archivo de destino al cuál le hemos indicado la extensión .mp4. De esta forma, ffmpeg busca los codecs de video y de audio apropiados para este formato (automáticamente selecciona h264 para video y aac para audio).

Veamos otro ejemplo haciendo una conversión de MP4 a MKV (matroska):

No obstante, podemos ser más específicos y personalizar los formatos que queremos utilizar. Por ejemplo, especificando el codec de video y el codec de audio que queremos en el archivo de destino:

```
ffmpeg -i video_original.mp4 -vcodec libx264 video_264.mkv  
ffmpeg -i video_original.mp4 -vcodec libx265 video_265.mkv
```

En ambos ejemplos anteriores estamos creando un archivo de video Matroska (MKV), con canal de audio vorbis en ambos casos. Sin embargo, en el primer ejemplo, estamos utilizando el codec de video h264 (mediante la librería libx264), mientras que en el segundo ejemplo

utilizamos el codec de video h265 (mediante la librería libx265), aún en desarrollo y futuro sucesor de H264.

Los resultados son bastante interesantes, puesto que el primer archivo resultante ocupa 120MB, mientras que el segundo 69MB. Ambos a 1080p (HD) y con la misma aparente calidad (eso sí, el segundo tarda más en comprimir y crearse).

También podemos hacer lo mismo con los canales de audio:

```
ffmpeg -i video.mp4 -vcodec copy -acodec mp3 h264_mp3.mkv  
ffmpeg -i video.mp4 -vcodec copy -acodec aac h264_aac.mkv  
ffmpeg -i video.mp4 -vcodec copy -acodec libvorbis h264_vorbis.mkv
```

Observa que en estos casos, hemos especificado copy en el codec de video, lo que obliga a ffmpeg a no hacer conversión de formatos de video, sino utilizar el que ya tiene (ahorrando mucho tiempo porque no hay que recomprimir el video), mientras que le especificamos el codec de audio mediante acodec.

Tanto en vcodec como en acodec tenemos que especificar la librería de codec a utilizar. En el caso de especificar el formato (como por ejemplo, «mp3»), el ffmpeg se encarga de seleccionar la librería más apropiada (libmp3lame, en este caso).

Publicidad

Podemos ver las librerías de codecs y formatos que soporta ffmpeg escribiendo en una terminal ffmpeg -formats o ffmpeg -codecs.

Si todo esto se te queda muy grande (o necesitas preparar algo para usuarios sin conocimientos de terminal), recuerda que tienes [video-converter-scripts](#), una colección de scripts ya preparados para que sólo tengas que arrastrar y soltar como puedes ver en este [GIF](#).

## Reducir calidad de video/audio

Por defecto, ffmpeg se encarga de detectar la calidad (bitrate) del video y audio del archivo original y le asigna uno equivalente al archivo de destino. Sin embargo, esto depende mucho del archivo en cuestión y puede que ffmpeg reduzca demasiado la calidad (y se vea con mala calidad el archivo resultante) o mantenga una calidad demasiado alta (y ocupe demasiado el archivo final).

Generalmente, ffmpeg hace un gran trabajo en este aspecto. Pero si queremos personalizar la calidad, utilizaremos el parámetro `-b:v` para el bitrate de video y `-b:a` para el bitrate de audio. Por ejemplo:

```
ffmpeg -i video.avi -b:v 2500k -b:a 192k video_final.mp4
```

Esto conseguiría que el archivo MP4 final tenga un video con un bitrate de 2500kb/s y un audio con un bitrate de 192kb/s.

## Extraer audio de un video

Otra operación muy común es querer extraer el audio de un video, para pasarlo a un archivo MP3, por ejemplo. Esto es muy sencillo de hacer con ffmpeg:

```
ffmpeg -i video.mp4 -vn audio.mp3
```

En algunos casos, como el caso anterior, ffmpeg detectará que el archivo de destino es un archivo de audio y hará la conversión automáticamente sin necesidad de indicar parámetros como -vn. Sin embargo, será necesario si queremos hacerlo en un formato final mkv sin canal de video.

## Silenciar (eliminar) el audio de un video

De la misma forma análoga al ejemplo anterior (donde eliminábamos el canal de video en un archivo de video), podemos eliminar el canal de audio de un archivo de video:

```
ffmpeg -i video.mp4 -an video_mute.mp4
```

## Subir el volumen de un video (o audio)

Muchas veces tenemos un video que tiene un audio muy bajo. Podemos subirle el volumen del canal de audio con el parámetro -vol, indicándole un valor numérico donde 256 es el volumen original, pudiendo subirlo o bajarlo:

```
ffmpeg -i video.mp4 -vol 512 video_final.mp4  
ffmpeg -i video.mp4 -af volume=2
```

En el segundo ejemplo vemos una forma alternativa utilizando filtros de audio, en el que subimos el volumen al doble de su volumen original.

## Rotar o girar un video

También es muy común tener la necesidad de rotar un video, ya sea porque nos interesa

tenerlo así o porque alguien tuvo la osadía de hacer un [maldito vídeo vertical](#). Para solucionarlo, podemos rotar el video haciendo lo siguiente:

```
ffmpeg -i video.mp4 -vf transpose=clock video_rotado_90.mp4  
ffmpeg -i video.mp4 -vf transpose=clock,transpose=clock video_rotado_180.mp4  
ffmpeg -i video.mp4 -vf hflip video_invertido_horizontalmente.mp4  
ffmpeg -i video.mp4 -vf vflip video_invertido_verticalmente.mp4
```

En el primer ejemplo, utilizamos el filtro de video transpose para girar 90 grados en el sentido de las agujas del reloj. En el caso de indicar el valor cclock en lugar de clock, se gira en el sentido contrario de las agujas del reloj. En el segundo ejemplo, aplicamos el filtro dos veces, por lo que conseguimos como resultado un giro de 180 grados. Los dos últimos ejemplos son para invertir un video horizontal o verticalmente.

Existe un parámetro rotate en ffmpeg que es mucho más flexible, pero cuidado, los valores deben expresarse en [radianes W](#).

## Redimensionar o cambiar tamaño de un video

Otra operación bastante frecuente es la de redimensionar el tamaño de un video. También es una operación muy sencilla de realizar con ffmpeg, aunque hay que tener en cuenta la proporción de aspecto, de modo que al redimensionar no se deforme el mismo. Veamos algunos ejemplos:

```
ffmpeg -i video.mp4 -vf scale=320:240 video_320x240.mp4  
ffmpeg -i video.mp4 -vf scale=320:-1 video_320x180.mp4  
ffmpeg -i video.mp4 -vf scale=iw/2:ih/2 video_x.mp4
```

En el primero de los casos, obligamos a ffmpeg a redimensionar el video a la resolución 320x240, independientemente del tamaño del video original. En la segunda opción, sin embargo, al indicar -1 le decimos a ffmpeg que utilice el alto apropiado para que el video no se deforme, reemplazándolo por su alto equivalente:

En el tercer ejemplo, utilizamos las palabras clave `iw` y `ih` que significan `input width` e `input_height`. Al dividirlas entre dos, lo que indicamos es que el video resultante tenga la mitad de tamaño tanto de ancho como de alto.

Nota: Hay que tener cuidado con algunos codecs, puesto que no permiten redimensiones a tamaños que no sean múltiplos de 4 o restricciones similares.

## Recortar fragmentos de un video

Otra operación interesante que nos puede surgir es la de recortar un fragmento de tiempo de un video más largo. Por ejemplo, obtener el fragmento de video desde los 35seg hasta los 65seg (30seg).

Existe un parámetro con el que podemos realizar estas operaciones:

```
ffmpeg -i video.mp4 -ss 35 -t 30 fragmento.mp4  
ffmpeg -i video.mp4 -ss 00:35 -to 01:05 fragmento.mp4
```

En el primer ejemplo, estamos seleccionando el fragmento de video desde los 35seg (como marca de inicio), y a partir de ahí, 30seg hacia adelante. En el segundo ejemplo, estamos seleccionando el fragmento de video desde la marca de tiempo de 35seg hasta 1min 5seg, ambos del video original.

## Añadir o eliminar pistas de audio a un video

La mayoría de formatos de video de la actualidad funcionan como contenedores que son capaces de incluir varios canales de audio (e incluso de otros tipos). Es muy común, por ejemplo, para añadir audio en diferentes idiomas. Con ffmpeg podemos gestionar esas pistas múltiples presentes en un archivo de video.

Un ejemplo de uso podrían ser los siguientes comandos:

```
ffmpeg -i video.mp4 -i audio_es.mp3 -map 0:v -map 0:a -map 1:a -vcodec copy video_final.mp4  
ffmpeg -i video.mp4 -map 0:0 -map 0:2 solo_canal1_audio.mp4
```

En el primer ejemplo, obtenemos dos archivos de entrada: video.mp4 (un video en inglés) y audio\_es.mp3 (el audio en español). Nuestra intención es añadir este audio en el video, de modo que se pueda seleccionar el idioma. Con -map 0:v y -map 0:a indicamos que utilizaremos el video y audio del primer archivo de entrada (video.mp4) y con -map 1:a indicamos que utilizaremos el audio del segundo archivo de entrada (audio\_es.mp3) para incorporarlo al video final.

En el segundo ejemplo, partimos de un archivo video.mp4 que tiene varios canales de audio, y con los parámetros -map 0:0 y -map 0:2 indicamos que queremos generar un archivo con el video (canal 0) y la segunda pista de audio (canal 2), eliminando por tanto la primera pista (canal 1).

## Mezclar pistas de audio

Pero en algunas ocasiones no interesa añadir otra pista de audio al video, sino mezclarla con la pista existente. Para ello, podemos utilizar el filtro de audio amerge, como vemos en el siguiente ejemplo:

```
ffmpeg -i audio1.mp3 -i audio2.mp3 -filter_complex amerge audio_mezclado.mp3
```

Para tareas un poco más complejas, quizás lo mejor sería utilizar un buen [editor de audio gratuito](#), que permite hacer cosas más avanzadas de forma más cómoda.

## Recorta zonas del video

Imagina por un momento que tienes un video, pero sólo te interesa quedarte con una región concreta del mismo. Esto es lo que se llama crop (recortar), y también se puede realizar con ffmpeg, con una línea de comandos como la que vemos a continuación:

```
ffmpeg -i screencast.mp4 -ss 00:30 -to 03:50 -vf crop=640:480:500:250 video_final.mp4
```

En ella, escogemos un fragmento de video, en el que sólo nos quedamos con una región de tamaño 640x480 desde la posición (500,250) de la pantalla. Esto puede ser muy útil para recortar [screencasts](#) que hemos hecho y queremos extraer sólo una región de la pantalla.

## Aplicar un viñeteado al video

Un efecto muy elegante y utilizado es el [viñeteado W](#), muy utilizado por los fans de Instagram. Con ffmpeg podemos también aplicar un viñeteado a nuestro video, en todos sus fotogramas, dándole un aspecto más elegante y profesional.

Para aplicarlo, basta con utilizar el filtro de video vignette, junto a un valor que representa el ángulo del mismo en radianes:

```
ffmpeg -i video.mp4 -vf vignette=PI/4 video_vignette.mp4  
ffmpeg -i video.mp4 -vf vignette='PI/4+random(1)*PI/50':eval=frame
```

El valor aplicado por defecto al viñeteado es pi/5, sin embargo se pueden hacer cosas un poco más complejas, como el segundo ejemplo, donde se crea un viñeteado que vibra aleatoriamente, simulando un efecto retro de reproducción antigua.

## Crear fundidos (fade-out o fade-in)

Otro efecto elegante utilizado a menudo en videos es el de los «fundidos a negro» (fade-out) o su proceso inverso (fade-in). Estos fundidos son muy comunes al inicio o al final de un video, ya que es una forma elegante de terminarlo.

Con ffmpeg se pueden crear de la siguiente forma:

```
ffmpeg -i video.mp4 -vf fade=t=in:st=0:d=5 video-fadein.mp4  
ffmpeg -i video.mp4 -vf fade=t=in:st=0:d=5,fade=t=out:st=25:d=5 video-fadeout.mp4
```

Teniendo en cuenta que nuestro video.mp4 tiene una duración de 30 segundos, en el primer ejemplo, creamos un fundido desde negro (fade-in). Los parámetros indicados fade=t=in:st=0:d=5 son para realizar un tipo de fundido de entrada (fade-in), que comience en la marca de tiempo de 0seg y dure 5seg desde que pasa de negro a desvanecerse por completo.

En el segundo ejemplo, añadimos además un fundido a negro (fade-out), que comienza a los 25seg y dura 5seg. También podríamos añadir un parámetro c=white para realizar los fundidos al color blanco, en lugar de negro.

Sin embargo, ya que tenemos el fundido a negro visualmente hecho, también podemos utilizar el filtro de audio afade para hacer lo mismo con el sonido:

```
ffmpeg -i true.mp4 -vf fade=t=in:st=0:d=5,fade=t=out:st=25:d=5 -af afade=t=in:ss=0:d=5,afad
```

## Extraer fotogramas de un video

Con ffmpeg también podemos realizar tareas con [formatos de imágenes ↗](#). Es posible extraer los fotogramas de un video (o un fragmento de video) y pasarlos a imágenes individuales. Para hacerlo, simplemente escribimos la siguiente línea de comandos:

```
ffmpeg -i video.mp4 image%d.jpg
```

Teniendo en cuenta que los videos, por lo general, tienen entre 25-30fps (fotogramas por segundo), esto quiere decir que, salvo que se trate de un video muy corto, cada video generará gran cantidad de imágenes.

Para evitarlo, se puede reducir los fotogramas por segundo a uno, por ejemplo, utilizando el siguiente filtro:

```
ffmpeg -i video.mp4 -vf fps=1 image%d.png
```

De esta forma, sólo generará una imagen por segundo. También podemos realizar la operación contraria, de modo que teniendo una serie de imágenes, las unamos y convirtamos en un video, algo ideal para técnicas de [stop-motion W](#), realizadas con una cámara digital:

```
ffmpeg -f image2 -i image%d.jpg video.mp4
```

## Poner un logo o marca de agua a un video

Quizás, uno de los ejemplos más prácticos puede ser el siguiente. En él, lo que hacemos es insertar una imagen en una posición concreta de la imagen. Esto es ideal para insertar logotipos o marcas de agua en un video. Ten en cuenta, que si utilizas imágenes PNG puedes aprovechar el canal alfa de transparencia del mismo y colocar marcas de aguas que no sean totalmente opacas. El primer ejemplo que se ve a continuación inserta la imagen logo.png en la posición (10,10), comenzando desde la esquina superior izquierda:

```
ffmpeg -i video.mp4 -i logo.png -filter_complex overlay=10:10 final.mp4  
ffmpeg -i video.mp4 -i logo.png -filter_complex overlay=x=(main_w-overlay_w):y=(main_h-over
```

Por otro lado, el segundo ejemplo realiza la misma tarea, pero utilizando variables predefinidas, que equivalen a colocar ese logotipo en la esquina inferior derecha, algo quizás más habitual.

## Reproducir videos en una terminal (ASCII art)

Como dato adicional y extravagante, ffmpeg incorpora la biblioteca [libcaca](#), de la cual ya hablé en [20 curiosidades geeks para terminales GNU/Linux](#). Se trata de una biblioteca especializada en la conversión multimedia a ASCII art, o lo que es lo mismo, es capaz de reproducir un video en una terminal de texto, como podemos ver a continuación:

Para ello, sólo hay que indicar el parámetro `-f caca`, seguido de `-pix_fmt rgb24` para indicar el formato de colores, y por último, utilizar el guión para utilizar la salida estándar:

```
ffmpeg -i video.mp4 -pix_fmt rgb24 -f caca -
```

Los fotogramas son del videoclip [True Survivor](#) , de la genial película Kung Fury, de la cual ya hablé en este completo artículo de [referencias y curiosidades de Kung Fury](#) .

Esto sólo es una selección práctica con algunos pequeños ejemplos de lo que se puede hacer

con ffmpeg. Si te ha resultado útil o conoces algún parámetro interesante que no esté en la lista, por favor, compártelo con nosotros en los comentarios.

Escrito por  Manz, el Domingo 7 de febrero de 2016, en [software](#). Comentarios recibidos: 25.

Este artículo aún no ha sido enviado a Menéame. Puedes [ser el primero en enviarlo](#) !



 20  39,3K

 40  128,9K

## 10 trucos para Windows que deberías conocer



Trucos para Windows: 10 programas que deberían venir de serie en Windows, y que al instalarlos, te harán la vida un poco más fácil.

## Guía de trucos para organizar tus archivos



Guía de trucos y consejos para organizar archivos y ficheros de tu disco duro. Liberar espacio, eliminar duplicados, encontrar imágenes con diferente resolución, etc...

 37  126,8K

## Guía de Sublime Text: ¿El mejor editor de código?



Sublime Text es uno de los mejores editores de código que existen. ¿Por qué? En este artículo se explican detalladamente sus características y funcionalidades.

[Leer más artículos](#)

Artículo escrito por J. Román Hernández, más conocido como Manz (autor de Emezeta). Es Ingeniero-técnico informático de Gestión por la Universidad de La Laguna y reside en Santa Cruz de Tenerife (Canarias). Puedes seguirlo en las siguientes redes:

[Twitter](#)[Linkedin](#)[Google+](#)

Anuncios patrocinados



## 25 comentarios de lectores

Publicidad



### Jesus De Baldoma

Domingo, 7 de febrero de 2016, 19:45

Por lo visto, en Ubuntu, ffmpeg fue sustituido en versiones anteriores por avconv del proyecto libav.

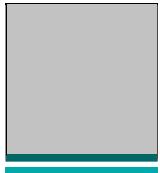
Más información en [askubuntu.com/questions/432542/is-ffmpeg-missing-from-the-official-repositories-in-14-04](http://askubuntu.com/questions/432542/is-ffmpeg-missing-from-the-official-repositories-in-14-04) ↗



## Hiber

Lunes, 8 de febrero de 2016, 02:20

Como todos los demás artículos, este está excelente.



Autor

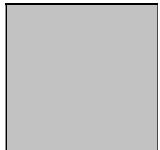


## Manz

Lunes, 8 de febrero de 2016, 13:12

[@Jesus de Baldoma](#) : Efectivamente, buena puntuación. En la wikipedia comentan un poco que fue lo que ocurrió con el tema de [Libav, el fork de FFmpeg](#).

[@Hiber](#) : ¡Muchas gracias!



Autor



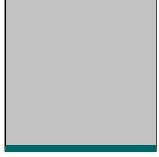
## Manz

Lunes, 8 de febrero de 2016, 13:20

Con el siguiente comando, es posible crear un efecto de morphing entre dos imágenes, es decir, que una primera imagen se convierta, poco a poco, en una segunda imagen. Con el parámetro t indicamos que dure 4 segundos y finalmente, que lo guarde en formato de GIF animado:

```
ffmpeg -loop 1 -i image1.jpg -loop 1 -i image2.jpg -filter_complex [1:v][0:v]b[
```



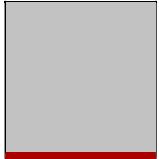
  
Autor

## Manz

Miércoles, 17 de febrero de 2016, 22:40

Con el comando que muestro a continuación se puede incrementar la velocidad del video y conseguir crear un efecto de [time-lapse](#) [W](#). Cuando más bajo sea el valor 0.02, más rápido irá el video, y cuanto más alto, más lento:

```
ffmpeg -i original.mp4 -vf setpts=0.02*PTS destino.mp4
```

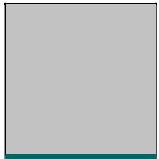


## Lustman

Martes, 1 de marzo de 2016, 14:59



Enhorabuena, ¡muy buen artículo! Nunca me defrauda lo que pones.



Autor



## Manz

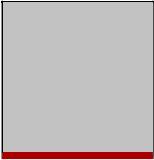
Miércoles, 3 de agosto de 2016, 11:34

Con el siguiente comando, se puede crear un video que haga "zoom in" (zoom acercándose) lentamente sobre la parte central del video desde una imagen estática (en loop), añadiendo un audio:

```
ffmpeg -loop 1 -i imagen.png -i audio.mp3 -vf "zoompan=z='min(zoom+0.0010,1.5)
```

No olvidar colocar el **-t 15** (*antes del filtro vf*) si se quiere limitar la duración del video resultante a 15 segundos, por ejemplo, ni establecer el tamaño de la imagen en el parámetro de **s=**.





## ElpidioMC

Sábado, 22 de octubre de 2016, 03:45

Muy buen articulo, te felicito. Me a servido de ayuda, pero tengo una pregunta, mi ordenador tiene una tarjeta de video nvidia, de la cual tengo entendido que cuenta con núcleos Cuda que permiten agilizar las labores de codificación de videos, y para ello se debe compilar FFmpeg incluyendo o habilitando NVENC. Una vez trate de hacerlo pero se me hizo un lio, asi que me podrias indicar una guia o pasos para compilar ffmpeg incluyendo esta capacidad?



Autor



## Manz

Jueves, 1 de diciembre de 2016, 18:54

Con el siguiente comando se pueden eliminar de un video las escenas con frames duplicados:

```
ffmpeg -i input.mp4 -vf mpdecimate, setpts=N/FRAME_RATE/TB output.mp4
```



Autor



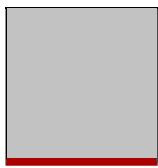
# Manz

Viernes, 6 de enero de 2017, 17:10

Con el siguiente comando se puede hacer un "trim" del audio (eliminar el silencio antes y después):

```
ffmpeg -i original.mp3 -af silenceremove=1:0:-50dB:1:0:-75dB destino.mp3
```

Los parámetros son: - **1:0:-50dB** Hace el trim al principio del audio (1), sin especificar tiempo (0) hasta que detecte un sonido en la barrera de -50dB. - **1:0:-75dB** Hace el trim al final del audio (1), sin especificar tiempo (0) hasta que detecte un sonido en la barrera de -75dB.

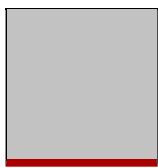


## Trum

Miércoles, 18 de enero de 2017, 23:19

Para rotar 270 grados funciona con:

```
ffmpeg -i video.mp4 -vf transpose=clock,transpose=clock,transpose=clock video_
```



## Cloud

Viernes, 10 de marzo de 2017, 15:42

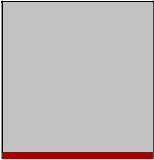
Enves de tanta CHORRADA hacer un puto INTERFAZ grafico para el FFMPEG, en lugar de poneros a enseñar comandos a estas alturas.

LOS COMANDOS SE USABAN en los ordenadores de hace 50 años.

Ahora existe una cosa, que se llama INTERFAZ GRAFICO, coño hacer uno DECENTE y entendible (y funcional) y dejarlos de OSTIAS.



-2



## Erknrio

Sábado, 11 de marzo de 2017, 12:54

Y aquí me trajo Google, otra vez :P.

¿Saben alguna forma de aplicar un doble overlay en el mismo vídeo? Tengo 3 vídeos, uno será el principal a pantalla completa y los otros dos serán miniaturas en las esquinas inferiores del vídeo principal.



Autor



## Manz

Sábado, 11 de marzo de 2017, 16:52

@Cloud : Ese comentario desborda ignorancia.

Realizar un interfaz gráfico para un programa como ffmpeg, llevaría mucho tiempo y trabajo (de hecho, ya existen algunos, pero son muy limitados debido a lo complejo que es crear programas versátiles de este tipo, que tienen infinitas posibilidades).

Los comandos NO se utilizaban en ordenadores de hace 50 años, sino que en ese entonces ocurría algo bien diferente: no existía interfaz gráfico. Hoy tenemos alternativas a las interfaces gráficas, pero las terminales de texto se utilizan más de lo que crees (servidores, administración de sistemas...), lo que probablemente ocurre es que en tu entorno no hay nadie con estas características y tienes la falsa percepción de que es algo que no se utiliza ya.

Si te interesa, puedes probar algo como Adobe Premiere (377,7/mes), que quizás está más cerca de lo que buscas. Pero te aviso, no es un asistente mágico :-)

Saludos,



-1

Autor



## Manz

Sábado, 11 de marzo de 2017, 16:53

[@erknrio](#) : Probablemente, buscas algo como lo que se comenta por aquí: [Video in a video with ffmpeg](#).

Saludos,



## Erknrio

Lunes, 13 de marzo de 2017, 09:02

[@Manz](#) : Ahora sí, ya encontré el fallo. Si lo traducimos a la declaración de una variable vendría siendo algo así:

Yo pretendía esto:

```
video = "video"  
overlay1 = video + "foo"  
overlay2 = video + "bar"  
output = overlay1 + overlay2
```

Cuando debía hacer algo como esto:

```
video = "video"  
overlay1 = video + "foo";  
overlay2 = overlay1 + "bar"  
output = overlay2;
```

Muchas gracias por la info :-).



# Erknrio

Lunes, 13 de marzo de 2017, 09:13

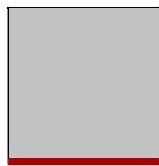
@Manz ➔: Pongo los comandos que usé al final. En el ejemplo usa -vf, pero se puede hacer con filter\_complex. Los vídeos pequeños los coloco arriba a la derecha y abajo a la derecha:

Ejemplo -vf:

```
ffmpeg -i video_main.mp4 -vf "movie=smaller_video1.mp4[thumb1];  
movie=smaller_video2.mp4[thumb2]; [in][thumb1] overlay=W-w[step1]; [step1][thumb2]  
overlay=W-w:H-h" output-overlay.mp4
```

Ejemplos filter\_complex:

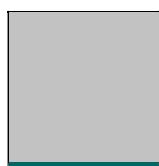
```
ffmpeg -y -i video_main.mp4 -filter_complex "movie=smaller_video1.mp4[thumb1];  
movie=smaller_video2.mp4[thumb2]; [thumb1] overlay=W-w[step1]; [step1][thumb2]  
overlay=W-w:H-h" output-overlay.mp4
```



## Antonio Baez

Lunes, 13 de marzo de 2017, 23:15

Soy un completo novato canario y me he hecho un lío con el ffmpeg. Básicamente lo quiero utilizar para convertir video .mp4 (H.264) a video .mp4 (H.265). Por favor, podrías mandarme en un mail la línea de comando para realizar esta operación. Te lo agradecería muchísimo.



Autor



# Manz

Martes, 14 de marzo de 2017, 11:36

@erknrio : Muchísimas gracias por compartir los comandos en la página. ¡Te lo agradezco mucho! Siempre que realices algo con ffmpeg, puedes comentarlo por aquí y así tendremos varios ejemplos creados a modo de instrucciones disponibles para todos.

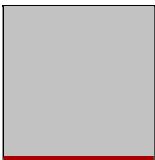
@Antonio Baez : Realmente es bastante sencillo. Simplemente escribe:

```
ffmpeg -i videooriginal.mp4 videotfinal.hevc
```

O de forma alternativa:

```
ffmpeg -i videooriginal.mp4 -vcodec hevc videotfinal.mp4
```

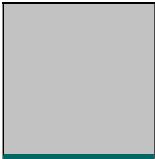
Saludos,



## Antonio Baez

Martes, 14 de marzo de 2017, 22:27

Perdona que te moleste de nuevo, pero el ffmpeg me tiene loco. Tengo el static y poniendo cualquiera de los dos comandos que amablemente pusistes, siempre me dice "gimnasio.mp4: No such file or directory."



Autor



**Manz**

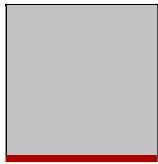
Martes, 14 de marzo de 2017, 23:02

@Antonio Baez ↗: El error nosuch file or directory aparece porque no está encontrando el archivo que le has dicho (gimnasio.mp4). Asegúrate que ese archivo está en la carpeta en la que estás trabajando con ffmpeg y que le estás poniendo el -i antes al video original y no al video final definitivo.

¡Saludos!



Like Dislike +1



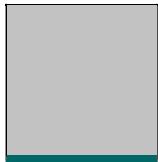
**Antonio Baez**

Miércoles, 15 de marzo de 2017, 01:23

El archivo está en la carpeta ffmpeg, pero ho hay manera. Por favor, podrías dejarme un e-mail donde pueda mandarte el reporter del cmd. Tal vez puedes ver algo que aclare el asunto. Gracias por atenderme.



Like Dislike



Autor

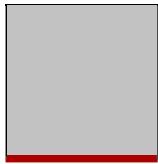


Jueves, 16 de marzo de 2017, 13:43

@Antonio Baez ➔ : El error que te aparece debe ser porque no está en la carpeta o tiene un nombre diferente (¿quizás alguna letra que no va?). En la parte superior de este blog, tienes un icono de una carta donde puedes enviarme el correo con la duda concreta que deseas.

Aún así, si es posible, preferiría que lo resolviésemos por aquí (salvo que haya algún dato que no quieras mostrar públicamente) para que así cualquier usuario que venga detrás tenga información para solucionar el problema si le ocurre lo mismo.

¡Saludos!

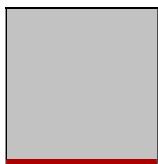


## Preguntoncojonero

Martes, 25 de abril de 2017, 16:17

@Cloud ➔ : Sin duda tu ignorancia que has expuesto es legendaria. Los comandos son necesarios para automatizar tareas programáticamente, lo que beneficia la productividad en ciertas tareas. Además de otras ventajas, de la línea de comandos.

Tu falta de respeto sobra aquí. Demuestras ser una ameba.



# Preguntoncojonero

Martes, 25 de abril de 2017, 16:34

@Manz ➔ :

Falta alguna página de scripts completos al estilo imagemagick

<http://www.fmwconcepts.com/imagemagick/> ➔

Ojalá se convierta el artículo con el tiempo en la GU? A DEFINITIVA DE FFmpeg.

@erknrio ➔ :

En gist pueden añadirse los fragmentos de código útiles y reales. Así disponibles para TODOS. Compartir es vivir.

Uno interesante: Convert MOV to GIF using FFmpeg and ImageMagick

<https://gist.github.com/tskaggs/6394639> ➔

<https://gist.github.com/ndarville/10010916> ➔

<https://gist.github.com/padde/2623647> ➔

<https://gist.github.com/Brainiarc7/95c9338a737aa36d9bb2931bed379219> ➔

<https://gist.github.com/Brainiarc7/2afac8aea75f4e01d7670bc2ff1afad1> ➔

Notas: Alguno hace uso también de ImageMagick y Libva. Lo desconozco el último.

En mi caso, no he probado ffmpeg. Lo instalaré. Instalé Imagemagick y cygwin en Windows 7. Manejo SO Windows.



## Publica tu opinión

Si lo deseas, puedes utilizar el siguiente formulario para publicar tu opinión o responder a alguna de las existentes:

Tu nombre

tu@email.com

http:// (o perfil de Twitter)

Escribe aquí tu comentario... ¡Separa en párrafos los textos muy abundantes y revisa la previsualización del comentario antes de enviarlo! Tu comentario puede tardar algunos segundos en aparecer después de enviarlo.

- Acepto las [condiciones y políticas de privacidad](#) de este sitio web.  
 Suscribirme a través de [FeedBurner](#) a los nuevos artículos del blog por email.

**Publicar comentario**

## Previsualización

Aquí se previsualizará su comentario. Revise que sea correcto antes de publicarlo.

## Artículos populares

### Las 100 mejores canciones de los 80



Es imposible reunir las mejores canciones de los 80, pero aquí va -a mi criterio- la lista de los mayores éxitos de la mejor década en el mundo de la música: los 80.

### 15 aplicaciones gratis para recuperar archivos borrados



Selección de 15 programas gratis para Windows que permiten recuperar información eliminada o borrada de nuestros discos o memorias.

### 13 paradojas que quizás no conocías



Una paradoja es una situación que desafía el sentido común y da como resultado una situación imposible. Aquí tienes 13 paradojas.

### 18 programas gratis para capturar pantalla en vídeo



Más de 18 programas gratuitos para crear screencasts: capturar o grabar en vídeo lo que hacemos en la pantalla de nuestro escritorio.

## ¿Qué significa G, E, 3G, H/3G+, H+, 4G?



¿Qué son esos iconos y letras G, E, 3G, H, 3G+, H+, 4G que aparecen en nuestro smartphone? ¿A qué velocidad puedo descargar con cada uno?

## Internet más rápido (o cómo mejorar tu conexión)



10 consejos y trucos sobre cómo conseguir que nuestra conexión a Internet funcione mejor y más rápida.



[Artículos](#) [Feed RSS](#) [Aviso legal](#) [Privacidad & Cookies](#) [Publicidad](#) [Contacto](#) [HTML5](#) [CSS3](#)



Recibe artículos recién publicados

8 monos escribieron 297,64 páginas con sus máquinas de escribir en 0.02 segundos  
CMS programado y diseñado por José Román Hernández Martín. Alojado en [Dedicated Server](#).



Esta web utiliza cookies. Si sigue navegando, se entiende que acepta las condiciones de uso. [¡Cuéntame más!](#) [Cerrar](#)