

Published January 2022



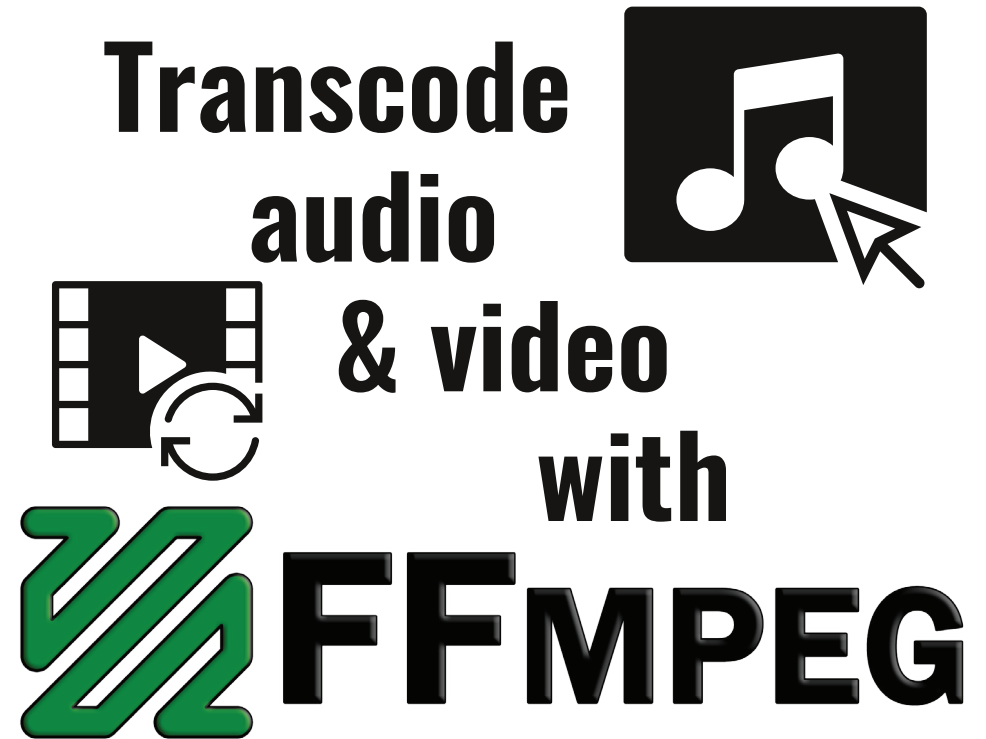
Philadelphia, Pennsylvania

*Version 0.8*



Download this zine as a PDF:  
<https://iffybooks.net/ffmpeg>

Anti-copyright,  
no rights reserved.



*a how-to guide  
from*





319 N. 11th St. #3D  
Philadelphia, PA 19107

Join our email list  
at [iffybooks.net](http://iffybooks.net)

Follow @iffybooks  
on social media

Send corrections to  
[iffybooks@protonmail.com](mailto:iffybooks@protonmail.com)

<b>VP9</b>	An open video compression standard from Google.	.mkv, .webm
<b>Theora</b>	A free video compression format from Xiph.Org.	.ogv
<b>MJPEG</b>	Motion JPEG format, in which each frame is encoded as a separate JPEG image.	.mov, .mkv, etc.

## → Read the docs and keep learning

You can find the complete FFmpeg documentation at the following URL:

<https://ffmpeg.org/ffmpeg.html>

If you can't figure out how to do something with FFmpeg, try running a web search. Someone has usually posted the exact command you need on Stack Overflow, GitHub, or a blog.

Here are some example web searches:

- extract lossless video clip with ffmpeg
- convert wmv to 1080p mp4 with ffmpeg
- create WebM video with VP9 and Opus with ffmpeg
- embed subtitles in video file with ffmpeg
- combine video clips with ffmpeg

Have fun!

<b>Vorbis</b>	A free lossy audio compression format (similar to MP3 or AAC) developed by the Xiph.Org Foundation.	.ogg
<b>Opus</b>	Opus is a newer lossy compression format from the Xiph.Org Foundation, currently preferred to Vorbis.	.opus

## → Some popular video coding formats

Container format	Description	File extension(s)
<b>MPEG-2 Part 2 (H.262)</b>	Video compression format introduced by the Moving Picture Experts Group in 1996. Still used for TV broadcasts.	.mp2
<b>MPEG-4 Part 2</b>	A popular video coding format in the 2000s.	.mp4
<b>AVC (H.264) a.k.a. MPEG-4 Part 10</b>	Currently the most widely used video compression format, with files around half the size of comparable MPEG-2s.	.mp4, .mkv, etc.
<b>HEVC (H.265) a.k.a. MPEG-H Part 2</b>	The successor to AVC (H.264). File sizes are smaller, but more computing power is required to decode them.	.mp4, .mkv, etc.

# FFmpeg is a super versatile command-line tool for converting between audio and video formats.

With FFmpeg, you can convert audio and video files to and from just about any format you've heard of. FFmpeg interfaces with hundreds of smaller programs (such as codecs, muxers, and demuxers), so you won't have to worry about the details.

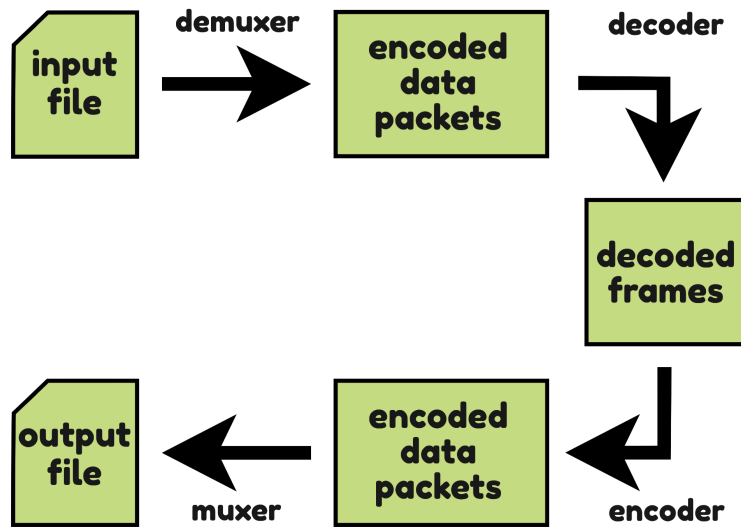
Here are some reasons you might want to use FFmpeg:

- You're making a podcast and you want to create MP3 and/or AAC files for distribution.
- You have FLAC audio files that you want to play on your phone or portable audio player.
- Your video editing program can't decode an old/obscure file format.
- You need to share a media file for school/work, and you want it to be in an accessible format.
- You're trying to embed a video in an HTML5 web page, and you need to provide multiple formats for different browsers.
- You're converting a collection of media files to a different format, and doing it one at a time is taking too long.

Digital audio and digital video are complex topics, and you won't learn everything in a day. This zine is designed to be a starting point and handy reference, but you'll still have questions when you finish it. That's good, keep reading and learning!

## → What is transcoding?

Transcoding a multimedia file means converting it from one coding format to another. Transcoding is often **lossy**, which means information is lost or noise is introduced. Converting a CD-quality WAV audio file to a 320Kbps MP3 is an example of lossy **compression**.



Transcoding can also be **lossless**, which means the original signal is preserved perfectly. Lossless transcoding can be reversed without losing any information, while lossy transcoding can't.

## → What is remuxing?

**Remuxing** means changing a file's **container format** while preserving the original audio/video **streams**. Remuxing is a lossless process, without any transcoding.



## → Some popular audio coding formats

Container format	Description	File extension(s)
<b>Pulse-code modulation (PCM)</b>	A common method for storing uncompressed audio, used for WAV files and CDs. The audio signal is encoded as a series of numbers, with each number representing the sound pressure level at a specific moment.	.wav, .aiff
<b>MP3</b>	A lossy audio compression format developed by the Fraunhofer Society in Germany and first released in 1991. MP3 compression is based on the modified discrete cosine transform (MDCT), the fast transform (FFT), and perceptual coding algorithms.	.mp3
<b>AAC</b>	Introduced in 1997 as a replacement for MP3, but less popular for many years. Part of the MPEG-2 and MPEG-4 standards, and used widely by Apple. AAC compression is based on the modified discrete cosine transform (MDCT) and perceptual coding algorithms.	.aac, .m4a
<b>FLAC</b>	FLAC (Free Lossless Audio Codec) is a lossless audio compression format developed by the Xiph.Org Foundation. FLAC files are often used for music, and they're usually half (or less) the size of equivalent WAV files.	.flac

<b>WAV</b>	The most common file format for uncompressed audio, developed by Microsoft and IBM. Audio is typically encoded using pulse-code modulation (PCM).	.wav
<b>AVI</b>	A proprietary audio/video container format introduced by Microsoft in 1992.	.avi
<b>Ogg</b>	A free, open container format maintained by the Xiph.Org Foundation.	.ogg for audio, .ogv for video
<b>3GP</b>	Third Generation Partnership format, used widely on Android phones. Implemented using the ISO/IEC Base Media File format.	.3gp
<b>M4V</b>	A video container format used by Apple (e.g., for videos purchased from iTunes). Similar to MP4, but with the possibility of DRM.	.m4v
<b>QuickTime / MOV</b>	A container format from Apple that supports a large number of audio and video coding formats.	.mov
<b>MKV</b>	Matroska Multimedia Container, a free and open container format that supports a large number of audio and video coding formats.	.mkv for video, .mka for audio
<b>WebM</b>	A royalty-free format for HTML5 video, based on MKV and sponsored by Google.	.webm

## → Glossary of digital A/V terms

<b>media file</b>	A broad term for any kind of digital document. This zine focuses on media files that contain audio, video, or a combination of the two.
<b>container format</b>	A container is a file that stores one or more data streams, such as audio and/or video. Many container formats can store media encoded using a range of different coding formats.
<b>stream</b>	<p>A stream is a single piece of audio, video, or related metadata stored in a media container file.</p> <p>An audio file normally contains one audio stream, which can be stereo or mono. A typical video file contains one stream for video and one for audio.</p> <p>Some media files have additional streams containing alternate audio tracks, subtitles, preview images, and other metadata.</p>
<b>demuxer</b>	A demuxer is a piece of software that can parse a media container file, extract an audio or video stream, and split the stream into a series of encoded data packets.
<b>muxer</b>	A muxer is a piece of software that takes a series of encoded data packets and writes them to a media container file.
<b>coding format (also known as compression format)</b>	<p>A media file's coding format describes how each data packet is encoded.</p> <p>Some coding formats are uncompressed, such as WAV audio. Uncompressed video files are so large that they aren't very common.</p> <p>Some coding formats offer lossless compression, such as FLAC audio.</p>

<b>coding format (cont'd)</b>	Nearly all of the audio and video you encounter has been compressed using a lossy compression format. The most common audio compression formats are AAC and MP3. For video, AVC (H.264) and HEVC (H.265) are currently popular.
<b>lossless &amp; lossy compression</b>	<p>The term "compression" refers to a range of techniques for reducing the size of digital files.</p> <p>Lossless compression algorithms use mathematical tricks to store data in a more compact form. Because lossless compression preserves the original signal perfectly, lossless compression can be reversed. The FLAC audio format is an example of lossless compression.</p> <p>Lossy compression algorithms can make media files dramatically smaller by exploiting the quirks of human perception. In the MP3 and AAC formats, for example, parts of the audio that humans can't hear are discarded.</p>
<b>codec (encoder/decoder)</b>	A codec is a piece of software used to encode and decode audio or video. For example, libmp3lame is a popular codec used to create MP3 audio files.
<b>transcode</b>	<p>Transcoding is the process of converting a multimedia file from one coding format to another.</p> <p>A lossy transcode is one that involves compressing (or re-compressing) an audio or video stream. Lossy transcoding isn't reversible; you'll always lose some information in the process.</p> <p>Transcoding can also be lossless, such as encoding or decoding a file using a lossless compression algorithm.</p>

## → Create a video from a series of images

The following command will take a series of images called **Frame1.jpg**, **Frame2.jpg**, **Frame3.jpg**, and so on, and create a video file called **Video\_output.mkv**.

```
ffmpeg -framerate 30 -i Frame%d.jpg \
-codec copy Video_output.mkv
```

The option **-codec copy** tells FFmpeg to combine the JPEG files losslessly, so it uses MJPEG format. The MKV (Matroska Video) format supports a wide range of coding formats, so it's a good fit.

## → Some popular container formats

Container format	Description	Common file extension(s)
<b>MP4</b>	<p>The MP4 container format is part of the MPEG-4 family of standards, originally developed for low-bitrate video encoding in the late 1990s and early 2000s. Starting in 2004, the MP4 format was generalized to create the ISO/IEC Base Media File format.</p> <p>These days, the MP4 container format is most often used with the AVC (H.264) or HEVC (H.265) video coding formats. Audio streams can use AAC or MP3 compression; AAC is currently more popular.</p>	.mp4 for video, .m4a for audio

## → Remux an M4V video to MP4

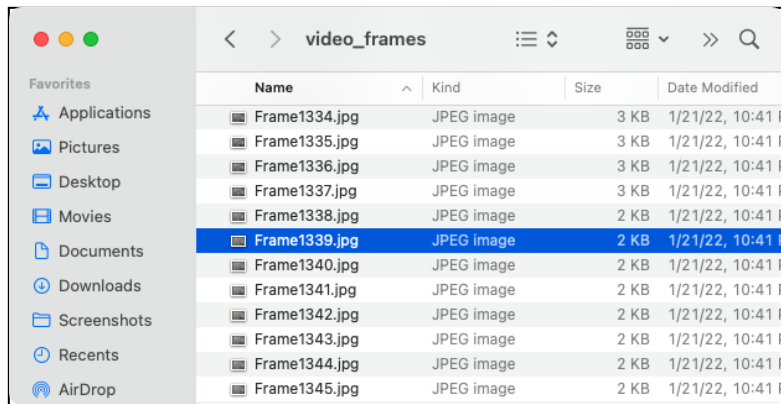
In order to play an M4V video on a Windows or Android device, you may need to convert it to an MP4 file first. Since M4V and MP4 containers support many of the same audio and video coding formats, try using the `-c:a copy` and `-c:v copy` options to copy the audio and video streams losslessly.

```
ffmpeg -i Video_input.m4v -c:a copy -c:v copy \
Video_output.mp4
```

## → Split a video into a series of images

The following command will convert each frame in the video `Video_input.mp4` to a separate JPEG file, creating a series of files called `Frame1.jpg`, `Frame2.jpg`, `Frame3.jpg`, and so on.

```
ffmpeg -i Video_input.mp4 Frame%d.jpg
```



You should move your video file to an empty directory before running this command, because 30 frames per second adds up to a lot of JPEGs. Your operating system is likely to slow down if you have more than 10,000 files in a single directory.

The following command will generate PNG image files instead:

```
ffmpeg -i Video_input.mp4 Frame%d.png
```

<b>remux</b>	Remuxing a multimedia file refers to changing its container format while leaving the underlying audio/video stream(s) untouched. Remuxing is a lossless process.
<b>generation loss</b>	<p>Every time you re-compress an audio or video signal that was already compressed, you'll lose some quality. This process is called generation loss. With digital audio and video, you may end up with glitchy artifacts after a media file has been re-compressed two or three times.</p> <p>Remux audio/video streams when possible to avoid generation loss, and keep your original media files if they aren't too big. That way you can re-encode from the original source if you need a different format.</p>

## → Install FFmpeg (macOS)

If you haven't already installed the **Homebrew** package manager, go to <https://brew.sh> and copy the provided line of code. Open **Terminal**, paste the line of code into the window, and press enter. You'll need to enter your password to continue.

You may get a popup window asking you to install **XCode Command Line Tools**, which requires ~3 GB of hard drive space. When installation is done, follow the prompts in the terminal to finish installing Homebrew.

Run the following command to install FFmpeg:

```
brew install ffmpeg
```

When the installation is complete, type the command `ffmpeg --help` and press enter. If FFmpeg is installed correctly, you'll see a list of options you can include in FFmpeg commands.



## → Install FFmpeg (Windows)

Open the Windows search box and type "PowerShell." Right-click "Windows PowerShell" and select "Run as administrator." Enter your password at the prompt.

If you haven't already installed the **Chocolatey** package manager, go to <https://chocolatey.org/install/> and follow the installation instructions. You'll need to copy a line of code and paste it into your PowerShell window.

When Chocolatey is finished installing, type the following command in your PowerShell window and press enter.

```
choco install -y ffmpeg
```

When the installation is complete, type the command **ffmpeg --help** and press enter. If FFmpeg is installed correctly, you'll see a list of options you can include in FFmpeg commands.

## → Install FFmpeg (Debian-based Linux)

Open a terminal window and update your package manager:

```
sudo apt-get update
```

Now run the following command to install FFmpeg:

```
sudo apt install ffmpeg
```

When the installation is complete, type the command **ffmpeg --help** and press enter. If FFmpeg is installed correctly, you'll see a list of options you can include in FFmpeg commands.



## → Losslessly extract audio from a video file

In many cases it's possible to extract the audio from a video file losslessly, without re-compressing the audio. First, you can use **ffprobe** to check how the audio is encoded.

```
ffprobe Video_input.mp4
```

The example video file uses AAC audio compression, so the output filename in the command below uses the extension ".aac". The **-vn** option tells FFmpeg to ignore the video stream.

```
ffmpeg -i Video_input.mp4 -vn \
-acodec copy Audio_output.aac
```

## → Convert an iPhone video to MP4

When you export a video from your iPhone, it's likely to have a MOV container, HEVC (H.265) video compression, and AAC audio. Here's a quick FFmpeg command to convert it to an MP4 container, AVC (H.264) video compression, and re-encoded AAC audio.

```
ffmpeg -i IMG_4620.MOV IMG_4620_re-encode.mp4
```

Instead of re-encoding the audio, a better approach is to copy the audio stream losslessly by adding the **-c:a copy** option.

```
ffmpeg -i IMG_4620.MOV -c:a copy \
IMG_4620_lossless_audio.mp4
```

To copy the video stream as well, you can add the **-c:v copy** option. The command below will losslessly remux the MOV to an MP4. However, HEVC (H.265) video compression isn't as widely supported as AVC (H.264).

```
ffmpeg -i IMG_4620.MOV -c:v copy -c:a copy \
IMG_4620_remux.mp4
```



## → Batch convert FLAC files to MP3

Until a few years ago, it was necessary to install a separate codec to read and write MP3 files with FFmpeg. But the patents associated with the MP3 have finally expired, and the company that controlled them (Fraunhofer Institute for Integrated Circuits) stopped collecting license payments in 2017. FFmpeg now includes the LAME codec (libmp3lame) by default.

If you want to convert every FLAC file in the current directory to an MP3, try using this short shell script (Linux and macOS only).

```
for file in *.flac;
do
ffmpeg -i "$file" "${file%.flac} ".mp3;
done
```

To create an AAC audio file instead, replace ".mp3" in the output filename with ".aac":

```
for file in *.flac;
do
ffmpeg -i "$file" "${file%.flac} ".aac;
done
```

## → Split a stereo audio file into separate mono files

A stereo audio file contains two separate channels: one for the left speaker, and one for the right speaker. The command below takes an audio file called **Audio\_input.wav** and extracts each channel to a separate WAV file: **Left\_output.wav** and **Right\_output.wav**.

```
ffmpeg -i Audio_input.wav \
-map_channel 0.0.0 Left_output.wav \
-map_channel 0.0.1 Right_output.wav
```

## → Use ffprobe to view a file's metadata

FFmpeg comes with a command-line tool called **ffprobe**, which you can use to examine media files' metadata. If you aren't sure how a media file was encoded, ffprobe can help.

Enter the following URL in your browser, and save the video **Example\_video.mp4** to your Downloads folder.

**[https://iffybooks.net/Example\\_video.mp4](https://iffybooks.net/Example_video.mp4)**

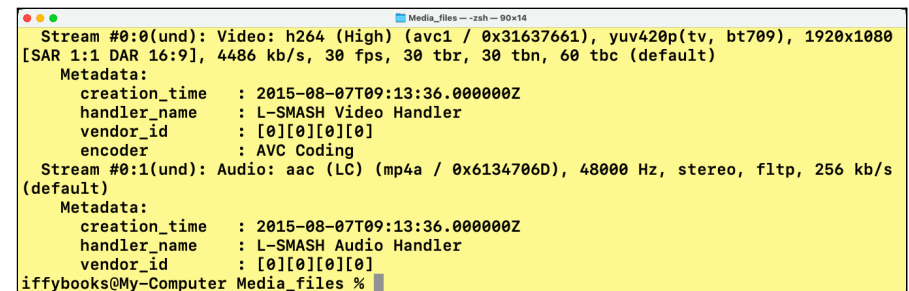
Open a Terminal/PowerShell window. To switch your current directory to the Downloads folder, type the following command and press enter.

```
cd ~/Downloads
```

Now type **ffprobe**, followed by a space, then the name of the video file. Press enter.

```
ffprobe Example_video.mp4
```

You'll see a waterfall of metadata in the terminal window. At the end of the output, look for the sections beginning "Stream #0:0" and "Stream #0:1". These are the video and audio streams, respectively. This file uses AVC (H.264) video compression and AAC audio compression.



```
Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv, bt709), 1920x1080 [SAR 1:1 DAR 16:9], 4486 kb/s, 30 fps, 30 tbr, 30 tbn, 60 tbc (default)
Metadata:
  creation_time   : 2015-08-07T09:13:36.000000Z
  handler_name    : L-SMASH Video Handler
  vendor_id      : [0][0][0][0]
  encoder        : AVC Coding
Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 256 kb/s (default)
Metadata:
  creation_time   : 2015-08-07T09:13:36.000000Z
  handler_name    : L-SMASH Audio Handler
  vendor_id      : [0][0][0][0]
iffybooks@My-Computer Media_files %
```

If you're using a media file with spaces in the filename, you'll need to put quotes around the filename.

```
ffprobe "Example video.mp4"
```

## → Convert a WAV to FLAC and back

Go to the following URL and download the file **Audio\_input.wav** to your Downloads folder.

**`https://iffybooks.net/Audio_input.wav`**

Open a Terminal/PowerShell window. To switch your current directory to the Downloads folder, type the following command and press enter.

**`cd ~/Downloads`**

Type the following command to convert your WAV file to a FLAC. The **-i** option indicates that the input file is called **Audio\_input.wav**. Next is the output filename, **Audio\_output.flac**. (If a filename you're using contains spaces, you'll need to put quotation marks around it.)

**`ffmpeg -i Audio_input.wav Audio_output.flac`**

When you press enter, FFmpeg will create a new file called **Audio\_output.flac**. Try opening the FLAC file with a media player program, such as VLC.

Go to your Finder/File Explorer and check the size of your FLAC file compared to the original WAV. It should be considerably smaller.

Next you'll convert your FLAC file back to an uncompressed WAV. First, run the following command to change the name of your FLAC file to **Audio\_input.flac**. (This isn't required; it just makes the next command easier to read.)

**`mv Audio_output.flac Audio_input.flac`**

Now you can use the following command to turn your FLAC file back into a WAV.

**`ffmpeg -i Audio_input.flac Audio_output.wav`**

Because FLAC is a lossless compression format, you haven't damaged the audio in your original WAV file by converting it to FLAC and back. You can repeat the process as many times as you want, and it will always sound the same.

## → Convert a FLAC file to MP3

The following FFmpeg command will load uncompressed audio from the file **Audio\_input.flac**, apply MP3 compression, and create a new file called **Audio\_output.mp3**.

**`ffmpeg -i Audio_input.flac Audio_output.mp3`**

Your command will look similar if you're converting a WAV file to an MP3, as in the example below.

**`ffmpeg -i Audio_input.wav Audio_output.mp3`**

By default, FFmpeg will create an MP3 with a bitrate of 128 kilobits per second (kbps). To use a bitrate of 320 kbps, you can add the option **-ab 320k** to your command.

**`ffmpeg -i Audio_input.flac -ab 320k \`  
**`Audio_output.mp3`****

**Note:** You can type the command above as one line, without the backslash. Or copy and paste it into your terminal as-is, and it should work. The backslash tells your terminal program that a command continues on the next line.

To create a variable-bitrate (VBR) MP3 instead, use the option **-q:a 0**. This will give you an MP3 with a bitrate range of 220-260 kbps. You can replace the number **0** with an integer up to **9**, where **0** is the highest quality setting and **9** is the lowest quality.

**`ffmpeg -i Audio_input.flac -q:a 0 Audio_output.mp3`**

To create an AAC audio file instead, replace ".mp3" in the output filename with ".aac":

**`ffmpeg -i Audio_input.flac Audio_output.aac`**