deepgram.com

Getting Started with ffmpeg for Audio - Deepgram Blog ⚡ | Deepgram

Kevin Lewis

6–7 minutes

FFmpeg is an open source toolkit for converting and manipulating both audio and video files from the terminal. It's the go-to for this kind of work - if you've used any conversion or simple editing tools there is a good chance that it relies on FFmpeg in some way. When I started using FFmpeg I would have loved more audio-focused beginner tutorials, so this is to help those people who are in the same boat I was.

Before we begin you will need to download a few things:

-
- A copy of FFmpeg - download it here.
-
- Two audio files to work with - you can download one here and the other here.

Create a new directory on your computer. Pop FFmpeg and both audio files into it. Then, open the directory in your terminal.

Your First Command

FFMpeg takes one or more files as an input and finishes a command by exporting a file. Between these two points we can further manipulate the file and export the result of the manipulations, but to start we will take in a file and spit out a new file in a different file format.

```
./ffmpeg -i nasa-spacewalk-interview.wav output.mp3
```

You should see a new file appear in your directory. The -i tells FFmpeg that the next string is an infile that operations should happen against. You can provide multiple infiles like so:

```
./ffmpeg -i file1.wav -i file2.wav [...] output.wav
```

Overwriting Files Automatically

If you run the above command more than once, you will be asked if you should overwrite the existing output.mp3 file. To automatically overwrite add the -y flag before any infiles:

```
./ffmpeg -y -i nasa-spacewalk-interview.wav
output.mp3
```

To never overwrite you can replace -y with -n.

Minimizing Terminal Information

There is a lot of information shown in the terminal when using FFmpeg. When FFmpeg first runs it will tell you about the configuration of your FFmpeg instance followed by the information related to your command. You can hide the configuration by adding -hide_banner anywhere in your command:

```
./ffmpeg -hide_banner -i nasa-spacewalk-interview.wav
output.mp3
```

Trimming Audio

We have already seen several options which start with -LETTER and are followed by a value if required (like the -i needing a file name). To trim an audio file we need two options - a starting sample time and either a duration or an ending point.

Run this command:

```
./ffmpeg -i nasa-spacewalk-interview.wav -ss 10 -t 15
output.mp3
```

This will start at 10 seconds and create a clip, from that point, that lasts 15 seconds. The following command will create a sample from 10 seconds to 15 seconds (effectively lasting 5 seconds):

```
./ffmpeg -i nasa-spacewalk-interview.wav -ss 10 -to
15 output.mp3
```

We won't cover it in great depth, but you can also provide timestamps in the format HOURS:MINS:SECS.MILLISECONDS instead of just a number of seconds:

```
./ffmpeg -i nasa-spacewalk-interview.wav -ss
00:00:10.5 -to 00:00:15.75 output.mp3
```

That's 10.5 seconds to 15.75 seconds.

Simple Filters

Simple audio filters (-af) have a single input, do a single thing, and provide a single output. Let's see how a few work:

Changing Volume Of Whole File

```
./ffmpeg -i nasa-spacewalk-interview.wav -af
"volume=0.25" output.mp3
```

This will set the volume to 25% for the whole sample.

Fading In Volume

```
./ffmpeg -y -i demo.wav -af "afade=t=in:ss=0:d=15"
output.mp3
```

This will fade in audio from the start of the file where it will be silent, to 15 seconds where it will be full volume.

Reduce Background Noise

```
./ffmpeg -i nasa-spacewalk-interview.wav -af
"highpass=f=200, lowpass=f=3000" output.mp3
```

This uses two audio filters at the same time and allows frequencies higher than 200hz to pass, and under 3000hz to pass. You may need to play with the exact values. Thanks to Stack Overflow user av8r for this one!

Complex Filters

Complex filters are both complex in their functionality and their syntax. Unlike simple filters which do a single operation to a single input, complex filters can be chained together. We pass in audio by a variable name, do something to it, and export a new variable which we can then further chain in a single filter.

They follow this syntax:

```
[INPUT]operations[OUTPUT1];
[OUTPUT1]operations[OUTPUT2]
```

As you can see, OUTPUT1 is created as a result of the first filter, and fed into the next as an input. Inputs are number-based and zero indexed based on the order they are provided as infiles - that means the first infile is [0], the second [1], and so on.

Let's see how this works in practice:

Overlaying Two Audio Files

```
./ffmpeg -y -i nasa-spacewalk-interview.wav -i
Bueller-Life-moves-pretty-fast.wav -filter_complex
"[0][1]amix=inputs=2" output.mp3
```

amix takes in both infiles and creates an output by directly overlaying them.

Trimming In Complex Filters

```
./ffmpeg -y -i Bueller-Life-moves-pretty-fast.wav
-filter_complex "[0]atrim=start=0:end=5" output.mp3
```

This creates a new audio file with just the first 5 seconds of the infile.

Fading in Complex Filters

```
./ffmpeg -y -i nasa-spacewalk-interview.wav
-filter_complex "[0]afade=t=in:ss=0:d=10" output.mp3
```

Combining Complex Filters

You can trim and fade audio files with simple filters, but what makes compelx filters so exciting is that you can combine them. Try this and we'll talk about it after:

```
./ffmpeg -y -i nasa-spacewalk-interview.wav -i
Bueller-Life-moves-pretty-fast.wav -filter_complex
"[0]afade=t=in:ss=0:d=10[fadeIn];
[1]atrim=start=0:end=5[trimmed];[fadeIn]
[trimmed]amix=inputs=2" output.mp3
```

There are three parts to this filter:

1. [0]afade=t=in:ss=0:d=10[fadeIn]; takes the first infile and applies a 10 second fade in.
2. [1]atrim=start=0:end=5[trimmed]; takes the second infile and trims it to the first 5 seconds.
3. [fadeIn][trimmed]amix=inputs=2 takes the output of the above two steps and overlays them.

In Summary

FFmpeg is hugely powerful, and with that comes a learning curve. We've only scratched the surface of what it can do but hopefully, its syntax makes more sense. If you have any questions please feel free to reach out on Twitter - we're @DeepgramAI.

If you have any feedback about this post, or anything else around Deepgram, we'd love to hear from you. Please let us know in our GitHub discussions .