FFmpeg for beginners

Compressing, converting, extracting and experimenting with its tools

Vitor Zanetti

1 Introduction to FFmpeg

FFmpeg is a powerful multimedia processing tool, widely used for video and audio conversion, compression, extraction, and manipulation. It offers a command-line interface, which allows for advanced users to have the necessary flexibility to automate tasks and perform batch operations. FFmpeg is often considered a Swiss Army knife for multimedia files, as it supports a broad range of file formats and codecs.

The core components of FFmpeg are its libraries and tools:

- **Libraries:**
  - `libavcodec` - A library containing a wide variety of video/audio codecs.
  - `libavformat` - A library for demuxing and muxing multimedia files.
  - `libavfilter` - A library for filtering audio and video streams.
- **Tools:**
  - `ffmpeg` - The main command-line tool for converting multimedia files.
  - `ffplay` - A tool for playback of multimedia files.
  - `ffprobe` - A tool used for inspecting multimedia file details and metadata.

Common use cases for FFmpeg include:

- Compression of audio and video files to reduce file sizes.
- Converting multimedia files from one format to another.
- Extracting audio from video files.
- Manipulating video attributes, such as resizing or trimming.

2 Installation and Setup

Before using FFmpeg, it's necessary to have it installed on your system. The installation steps differ slightly depending on the operating system.

2.1 Installing FFmpeg on Windows

1. Download the latest FFmpeg Windows build from https://ffmpeg.org/download.html.
2. Extract the contents of the downloaded ZIP file.
3. Move the extracted folder to a location of your choice (e.g., `C:\ffmpeg`).
4. Add FFmpeg to your system's PATH (explained in the next section).

2.2 Installing FFmpeg on macOS

1. **Make sure Homebrew is installed**. If it isn't, open Spotlight (F4 for default, or FN + F4) and type in "Terminal". Then, run the following command:

```
/bin/bash -c "$(curl␣-fsSL␣\
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2. Install FFmpeg with this command:

```
brew install ffmpeg
```

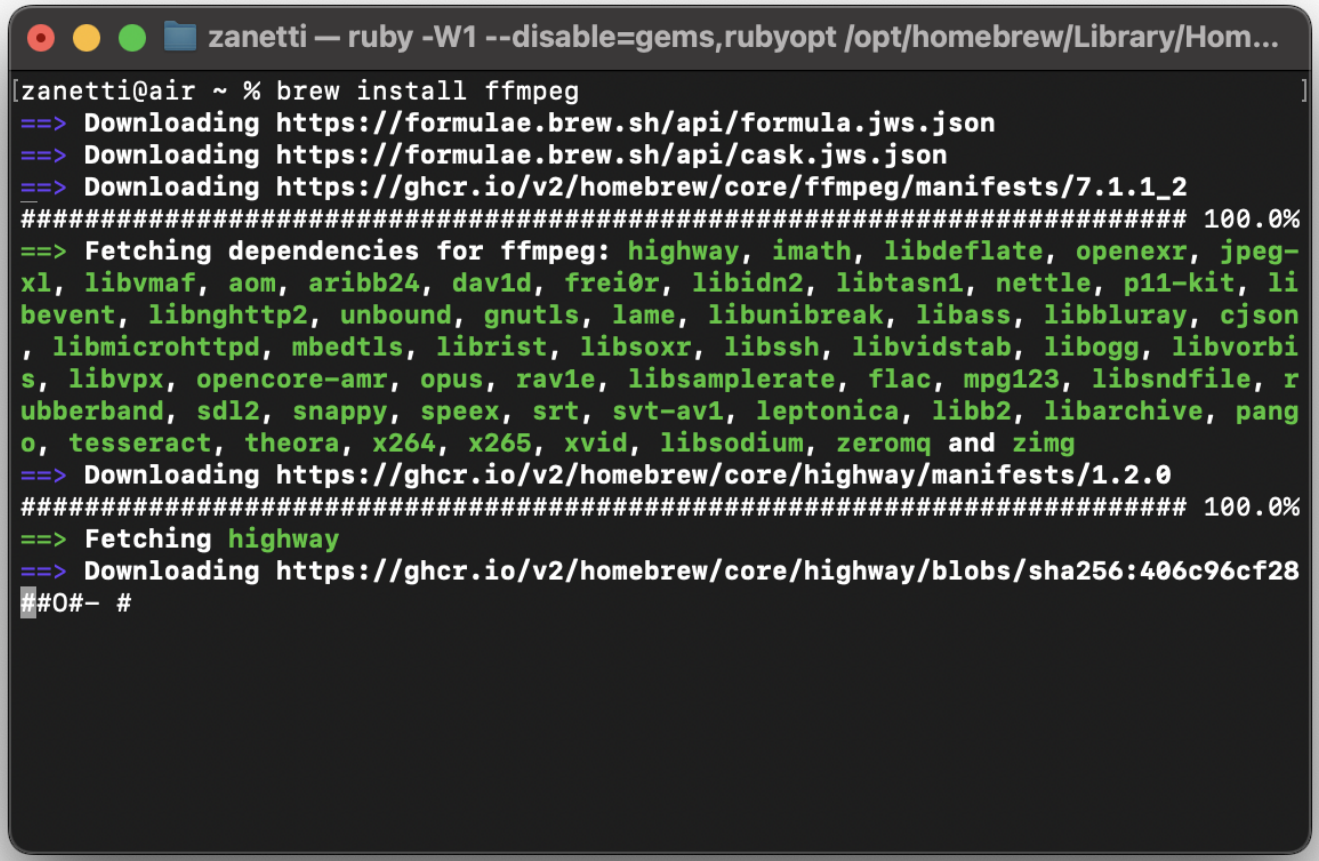


Figure 1: Download and install process via Brew on macOS Sonoma

2.3 Installing FFmpeg on Linux

On Linux, FFmpeg can typically be installed through the package manager. For Ubuntu/Debian-based distributions:

```
sudo apt update
sudo apt install ffmpeg
```

2.4 Verifying Installation

After installation, verify that FFmpeg is working properly by typing the following in your terminal/command prompt:

```
ffmpeg -version
```

If FFmpeg is correctly installed, version information will be displayed.

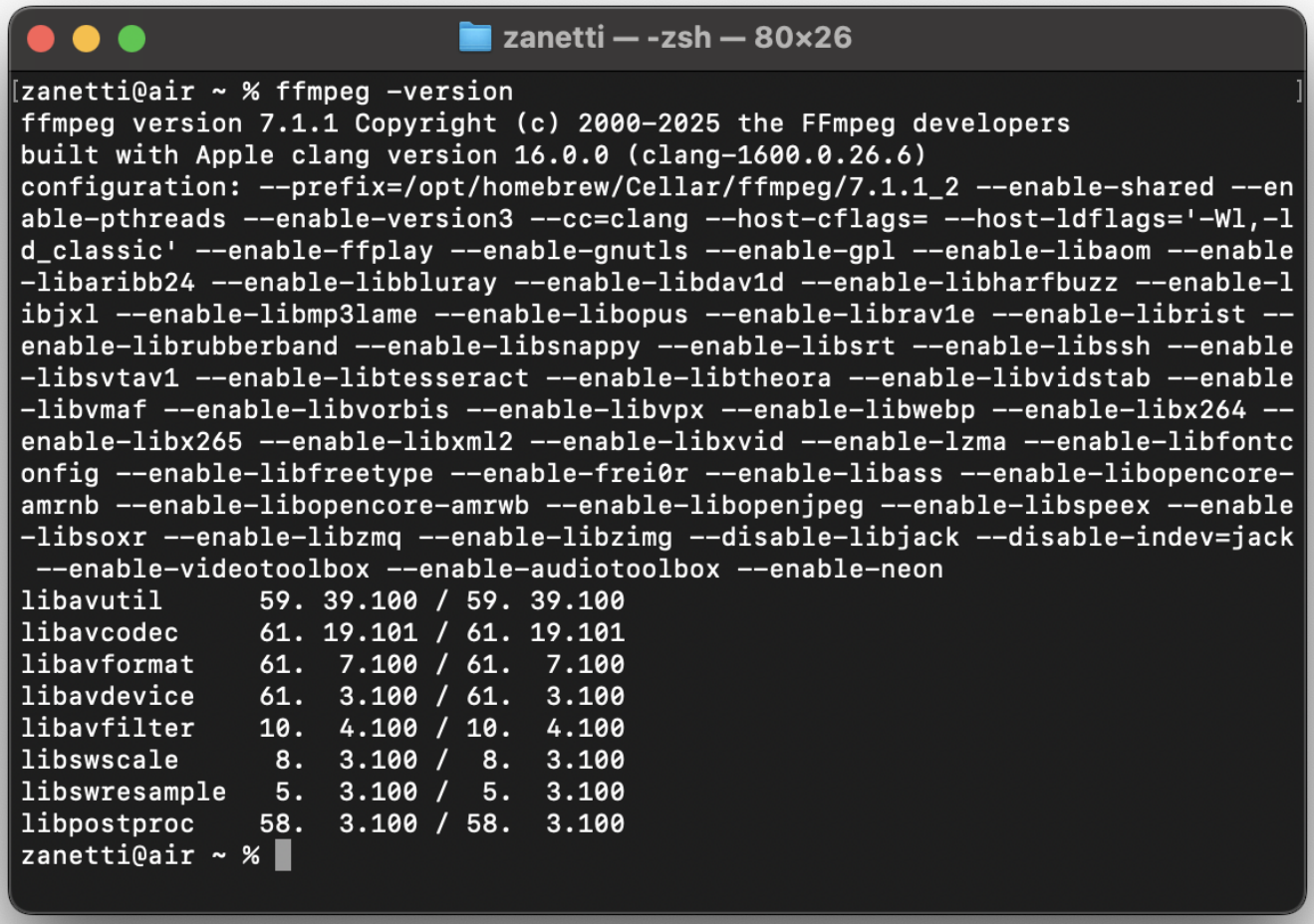


Figure 2: Sucessful installation on macOS exhibits the information above

3 Adding FFmpeg to PATH (Windows Only)

In order to run FFmpeg from any location in the command prompt, it's necessary for the user to add it to the system's PATH. This allows the usage of FFmpeg commands without specifying the full path to the executable.

To add FFmpeg to the PATH:

- Right-click on `This PC` or `My Computer` and choose `Properties`.
- Click on `Advanced system settings`.
- In the `System Properties` window, click on the `Environment Variables` button.

- In the `System Variables` section, find and select the `Path` variable, then click `Edit`.
- Add the path to the `bin` folder of your FFmpeg installation. For example: `C:\ffmpeg\bin`.
- Click `OK` to save the changes.

Now, FFmpeg commands can be run from anywhere in the command prompt.

4 Basic Commands for Beginners

4.1 Converting Media Formats

To convert an `input.mp4` file to `output.avi`, use the following command:

```
ffmpeg -i input.mp4 output.avi
```

Here, `-i` specifies the input file, and `output.avi` is the desired output format (in this case, AVI).

What is AVI? AVI (Audio Video Interleave) is a multimedia container format introduced by Microsoft, allowing video and audio to be stored together in a single file. While AVI files can store high-quality video, they tend to be large compared to more modern formats like MP4.

4.2 Compressing Video/Audio

File size of a video can be reduced by adjusting its bitrate. The bitrate determines the amount of data used to represent audio or video, which in turn affects its quality and size. Lowering the bitrate will reduce the file size, but may also degrade the quality.

To compress a video by adjusting the bitrate, use the following command:

```
ffmpeg -i input.mp4 -b:v 1M -b:a 128k output.mp4
```

This command sets the video bitrate (`-b:v`) to 1 megabit per second (1M) and the audio bitrate (`-b:a`) to 128 kilobits per second (128k).

Why Compress by Bitrate? Compressing by bitrate can be useful when there is, for instance, a necessity to make a video smaller for storage or upload purposes. By reducing the bitrate, file sizes can be reduced. However, quality will be affected, so there is always a trade-off between size and quality. It is important to test the video to find the right balance.

4.3 Changing Resolution to 720p

To scale the video resolution to 720p (HD), use the following command:

```
ffmpeg -i input.mp4 -vf "scale=1280:720" output.mp4
```

Here, `-vf` specifies the video filter, and `scale=1280:720` sets the resolution to 1280x720 pixels (720p).

Should the video have higher or lower resolution before a conversion? If one wants to scale a video down to 720p, the original resolution must be higher than 720p. If the video is already smaller than 720p, scaling it up may result in a loss of quality.

4.4 Extracting audio from video (MP3)

To extract the audio from a video file and save it as an MP3, use the following command:

```
ffmpeg -i input.mp4 -q:a 0 -map a output.mp3
```

Here, `-q:a 0` specifies the audio quality (0 is the highest), and `-map a` selects the audio stream.

What does it mean to "extract audio"? Extracting audio refers to the process of separating the audio track from a video file. This can be useful if it is needed only for the sound, such as in a podcast or audio editing. The extracted audio file is typically in MP3, AAC, or WAV format.

4.5 Extracting audio without re-encoding

Audio from a video can be extracted without re-encoding the file, which is faster and avoids potential loss of quality. To do this, use:

```
ffmpeg -i input.mp4 -vn -c:a copy output.aac
```

Here, `-vn` tells FFmpeg to ignore the video, and `-c:a copy` copies the audio stream directly without re-encoding.

What is re-encoding? Re-encoding is the process of converting a media file from one format to another, which often involves compressing or changing the file's characteristics. When an audio or video is re-encoded, there may be some loss of quality depending on the settings used. If there is no need to modify the audio, extracting it without re-encoding will preserve the original quality.

4.6 Extracting frames from video

To extract individual frames from a video and save them as images, use the following command:

```
ffmpeg -i input.mp4 -vf "fps=1" frame_%03d.png
```

This command extracts one frame every second and saves them as PNG images (e.g., $\texttt{frame}_0 01.png, \texttt{frame}_0 02.png, etc.$).

Trimming refers to cutting a video by removing parts from the beginning or end of the file. To trim a video, the start and end times can be specified for the portion to be kept. Use the follow command to trim a video from 00:01:00 (1 minute) to 00:03:00 (3 minutes), for example:

```
ffmpeg -i input.mp4 -ss 00:01:00 -to 00:03:00 -c copy output.mp4
```

Here, -ss specifies the start time, and -to specifies the end time. The -c copy option ensures that the video is copied without re-encoding.

4.8 What is Metadata?

Metadata is information about a media file, such as its format, codec, resolution, duration, and more. FFmpeg can extract and display metadata about a video or audio file using the ffprobe tool which is covered in the next section. Metadata is useful for understanding the properties of a file and troubleshooting issues.

5 Using FFprobe

5.1 What is FFprobe?

FFprobe is a companion tool to FFmpeg that provides detailed information about media files. It is often used to inspect the metadata and streams of multimedia files.

To use FFprobe to display metadata for a file, use the following command:

```
ffprobe -v quiet -print_format json -show_format -show_streams input.mp4
```

This command outputs detailed metadata in JSON format. In order, the flags work for suppressing unnecessary messages and displaying file-level and stream-level metadata, respectively.

5.2 Common FFprobe output

The output from FFprobe may include information such as:

- Format Information: Contains the container format (e.g., MP4, AVI) and file size.
- Stream Information: Provides details on each stream (video, audio, subtitle, etc.), includin codec, resolution, bitrate, and duration.
- Metadata: Information like title, artist, and other tags embedded in the media file.

6 Advanced commands

6.1 Resizing video and maintaining aspect ratio

When resizing a video, it is important to maintain the aspect ratio to avoid distortion. The scal filter, with a formula, can be used to automatically preserve the aspect ratio.

For example, to scale the video to a width of 640 pixels while maintaining the aspect ratio, use:

```
ffmpeg -i input.mp4 -vf "scale=640:-1" output.mp4
```

Here, -1 tells FFmpeg to automatically calculate the height based on the width.

6.2 Changing audio sample rate

If it is necessary to change the audio sample rate (e.g., from 44100 to 48000 Hz), use the followi command:

```
ffmpeg -i input.mp4 -ar 48000 output.mp4
```

This command changes the sample rate to 48000 Hz, which is commonly used in professional audio and video production.

6.3 Overlaying text on video

FFmpeg allows for overlaying text on a video. For example, to add text that says "Hello World!" at the top of the screen, use:

```
ffmpeg -i input.mp4 -vf "drawtext=text='Hello World!':x=10:y=10:fontsize=24:fontcolor=white" output.
    mp4
```

This command overlays the text in the top left corner, with a font size of 24 and white color.

6.4 Adding subtitles to video

To add subtitles to a video, use the following command:

```
ffmpeg -i input.mp4 -i subtitles.srt -c:v copy -c:a copy -c:s mov_text output.mp4
```

Here, -i subtitles.srt specifies the subtitle file, and the latter ensures that the subtitles are added in a format compatible with MP4.

7 Conclusion

FFmpeg is a versatile tool for anyone working with multimedia files, providing the tools needed to handle most media tasks directly from the command line.

For more detailed documentation on FFmpeg commands and features, visit the official documentation https://ffmpeg.org/documentation.html