FFMPEG An Intermediate Guide/image sequence

FFMPEG has a powerful set of features relating to creating video from images, or generating an image sequence from a video.

Overview

To create a video from a set of images:

```
ffmpeg -i image-%03d.png video.webm
```

To create a set of images from a video:

```
ffmpeg -i video.webm image-%03d.png
```

When no further arguments are given a set of defaults will be used. For example the framerate will be 25 and the encoding will be inferred from the filenames.

The `image-%03d.png` part is a filename pattern. This particular pattern corresponds to `image-000.png`, `image-001.png`, `image-002.png` up to `image-999.png`. The "`%03d`" represents a sequence of three, zero-padded, decimal digits. There is more to know about filename patterns which will explained in a later section.

The canonical form to work with image sequences is to use the `-f image2` argument like this:

```
ffmpeg -f image2 -i image-%03d.png video.webm
```

and

```
ffmpeg -i video.webm -f image2 image-%03d.png
```

But ffmpeg is very good with inferring that information so this chapter will omit that argument in all examples unless absolutely necessary.

Making a video from an Image Sequence

```
ffmpeg -i image-%03d.png video.webm
```

This will create a video with the filename video.webm from the image files named image-000.png, image-001.png, image-002.png, up to the last sequentially numbered three digit image file.

The encoding of the images and of the video is inferred from the extensions. The framerate is 25 fps by default. The video width and height is taken from the images. The images have to be all of the same dimension.

Filename numbering

Ffmpeg expects the numbering to start at 0 (or 000 in this example). If your files do not start at 000 then use the argument `-start_number` to tell ffmpeg the first number of your files:

```
ffmpeg -start_number 100 -i image-%03d.png video.webm
```

The same argument can be used to skip over a number of files and start at a certain number.

The numbering has to be consecutive. Ffmpeg will stop at the last consecutive numbered filename.

For example if you have following files:

```
image-000.png
image-001.png
image-002.png
image-005.png
```

Then ffmpeg will only process the first three and ignore the last file. If you have missing numbers you can use a glob pattern or rename all the remaining files to close the gap.

Framerate

The default framerate is 25 fps. That means each image is shown 1/25 of a second. This is a speed which creates the illusion of a smooth animation for most humans eyes. If you still want more frames per second then use the `-framerate` argument:

```
ffmpeg -framerate 60 -i image-%03d.png video.webm
```

If you want to have a slideshow then you have to lower the input framerate. The following command will show each image 5 seconds long:

```
ffmpeg -framerate 1/5 -i image-%03d.png video.webm
```

It may be necessary to force the video framerate up:

```
ffmpeg -framerate 1/5 -i image-%03d.png -r 30 video.webm
```

Slideshow with different durations

You can make a slideshow with diferent duration for each picture, by using the zoompan filter. In the following example each picture is shown 1 second (25 frames), except the 3rd picture is shown 3 seconds (25+50 frames) and the 5th picture is shown 5 seconds (25+100 frames):

```
ffmpeg -i image-%3d.jpg -vf "zoompan=d=25+'50*eq(in,3)'+'100*eq(in,5)'" test.mp4
```

Slideshow with crossfading between the pictures

You can make a slideshow with crossfading between the pictures, by using a combination of the zoompan and framerate filters. "A" is the duration in seconds how long each picture is shown (without the crossfade duration), and "B" is the crossfade duration in seconds

```
ffmpeg -i IMG_%3d.jpg -vf zoompan=d=(A+B)/B:fps=1/B,framerate=25:interp_start=0:interp_end=255:scene=100
-c:v mpeg4 -maxrate 5M -q:v 2 out.mp4
```

Making an Image Sequence from a video

```
ffmpeg -i video.webm image-%03d.png
```

This will extract 25 images per second from the file video.webm and save them as `image-000.png`, `image-001.png`, `image-002.png` up to `image-999.png`. If there are more than 1000 frames then he last image will be overwritten with the remaining frames leaving only the last frame.

The encoding of the images and of the video is inferred from the extensions. The framerate is 25 fps by default. The images width and height is taken from the video.

Extract one image per second:

```
ffmpeg -i video.webm -vf fps=1 image-%03d.png
```

Extract one image from a specific time:

```
ffmpeg -i video.webm -ss 00:00:10 -vframes 1 thumbnail.png
```

Filename patterns

Usually ffmpeg has one input file and one output file, but when we want to create a video from a set of image then we have a set of input files. Likewise, when extracting images from a video file there is a set of output files. A filename pattern is a pseudo filename to describe a set of filenames.

There are roughly three types of filename patterns:

pattern type	corresponding filenames	description

`image-%03d.png`	`image-000.png`
	`image-001.png`
	`image-002.png`
	`...`
	`image-999.png`
		Numbered with leading zeroes

`image-%d.png`	`image-1.png`
	`image-2.png`
	`...`
	`image-9.png`
	`image-10.png`
	`image-11.png`
	`...`
	`image-99.png`
	`image-100.png`
	`...`
		Numbered without leading zeroes

`image-*.png`
(needs an argument to enable)
	`image-a.png`
	`image-b.png`
	`image-c.png`
	`...`
	`image-foo.png`
	`image-bar.png`
	`...`
	`image-.png`
	`...`
		Not numbered or no regular pattern

Technically there is a fourth form: the single image form. But that is a trivial special case and will not be discussed further in this section.

Numbered with leading zeroes

This pattern:

```
image-%03d.png
```

Corresponds to the following filenames:

```
image-000.png
image-001.png
image-002.png
...
image-999.png
```

In words: the string `image-` followed by three digits followed by the string `.png`.

The heart of the filename expansion mechanism is the substring starting at the character `%` and ending at the character `d`. Everything before the `%` and after the `d` is taken literally. Everything from `%` to `d` is replaced by one or more digits.

The general form is `%0Nd` where `N` is a number larger than zero which determines how many digits there are.

Note: the `%0Nd` notation is commonly used in many programming languages. The notation is so complex because it is designed to accommodate many different use cases. The ffmpeg developers borrowed the notation and stripped it down to just one use case: how many digits. That is why it may seem unnecessarily complex.

Percent in filename

There is a special case if the filenames itself contains the `%` character. Because the `%` character has a special meaning in the pattern care has to be taken. A single `%` character in the pattern will be interpreted as the beginning of the substring to be replaced by digits. To specify a literal `%` the character has to be doubled in the pattern. The technical jargon for this is to "escape" the character.

For example this pattern:

```
image-%02d%%.png
```

Corresponds to the following filenames:

```
image-00%.png
image-01%.png
image-02%.png
...
image-99%.png
```

Numbered without leading zeroes

If the number in the filenames are not padded with leading zeroes then the pattern is `%d`.

This pattern:

```
image-%d.png
```

Corresponds to the following filenames:

```
image-1.png
image-2.png
...
image-9.png
image-10.png
image-11.png
...
image-99.png
image-100.png
...
```

Not numbered or no regular pattern

If the filenames are not numbered using digits or are otherwise not formated regularly then one can use the glob pattern. Glob patterns must be enabled using the argument `-pattern_type glob`.

Example ffmpeg command with glob pattern enabled:

```
ffmpeg -pattern_type glob -i "image-*.png" video.webm
```

This pattern:

```
 "image-*.png"
```

Corresponds to the following filenames:

```
image-a.png
image-b.png
image-c.png
```

but also to the following filenames:

```
image-foo.png
image-bar.png
```

and also to:

```
image-.png
```

and to any filename which follows the pattern string `image-` followed by anything followed by the string `.png`.

Generally the pattern `*` corresponds to anything of any length and that means even nothing.

There are more glob patterns. For example `?` which corresponds to any single character. But we will not go in to more details here. The interessted reader can read more in depth information in the wikipedia page about glob (programming).

Quote the glob pattern

It is necessary to place the glob pattern in quotes otherwise the shell will expand the glob before executing the command.

For example the glob pattern in this command:

```
ffmpeg -pattern_type glob -i image-*.png video.webm
```

will be expanded by the shell to:

```
ffmpeg -pattern_type glob -i image-foo.png image-bar.png video.webm
```

before it is executed. Note that this command has a different meaning because now `image-bar.png` is the second filename argument while `video.webm` is a third filename argument.

When placed in quotes the shell will not expand the glob pattern:

```
ffmpeg -pattern_type glob -i "image-*.png" video.webm
```

Order of files from glob pattern

Numbered files have a clearly defined ordering. For the most part humans and computers alike agree how to sort the numbers. This is especially true for numbers with leading zeroes. `015` clearly comes before `110`. Without leading zeroes things get a bit more complicated. Most computers will sort `15` after `110`. But this problem is solved by the `%d` notation.

When the files are no longer numbered or not regularly then things can get very strange. For example should `ae` sort before or after `aaa`? Computers are deterministic and always have an idea of how to sort things, but that idea of order does not necessarily makes sense for a human.

The order of the files from a glob pattern is determined by the glob system call. On most systems the idea of order of the glob system call can be manipulated using an environment variable named `LC_COLLATE`. But this book will not discuss that.

For small number of files the order returned by a glob can be inspected by letting a shell expand the glob. This works because most shells uses the same glob system call as ffmpeg would.

This command typed in a shell:

```
echo image-*.png
```

will print a list of filenames. That same list in the same order will be the list of files processed by ffmpeg if given the glob pattern.

If the order is not satisfactory or the number is too large to inspect by humans then the best course of action is to rename the files to be regularly numbered and then use the `%0Nd` pattern.

References

The fine ffmpeg manual and wiki located at:

- http://www.ffmpeg.org/documentation.html
- https://trac.ffmpeg.org/wiki

More specific the following pages:

- http://www.ffmpeg.org/ffmpeg.html
- http://www.ffmpeg.org/faq.html#How-do-I-encode-single-pictures-into-movies_003f
- http://www.ffmpeg.org/faq.html#How-do-I-encode-movie-to-single-pictures_003f
- http://www.ffmpeg.org/ffmpeg-formats.html#image2-1
- http://www.ffmpeg.org/ffmpeg-formats.html#image2-2
- https://trac.ffmpeg.org/wiki/Create%20a%20video%20slideshow%20from%20images
- https://trac.ffmpeg.org/wiki/Create%20a%20thumbnail%20image%20every%20X%20seconds%20of%20the%20vide