Webcam Emulation

tropf

ABSTRACT

short description of how to emulate a webcam on linux using v4l2loopback and ffmpeg

1. Idea

To create a virtual webcam feed, the process is split into two seperate steps. (Well, actually three.)

The first step is creation of the virtual webcam, while the second step is feeding data into that virtual webcam.

1.1. Tools

Using v4l2loopback a virtual webcam is created. This guide will employ ffmpeg to write to that device from various sources. In this guide usage of a file as well as the X server (your screen) is shown.

The usage of ffmpeg allows to use a wide variety of input sources and all filters supported by ffmpeg can be applied on the fly.

1.2. Order

Order matters. The steps should be performed in this order:

1 create virtual webcam

2 launch ffmpeg to write to that webcam

3 launch the program reading from that webcam (e.g. your browser)

It is critical that you do not launch the application requesting webcam access (e.g. your browser) before launching ffmpeg: The virtual webcam will only be announced as "having data" after ffmpeg will writes data to it.

2. Creating the Virtual Webcam

Make sure to install the kernel module `v4l2loopback`. Then load with the option`exclusive_caps=1` like so:

```
sudo modprobe v4l2loopback exclusive_caps=1
```

You may apply additional options, listed at the v4l2loopback github repo ⟨`https://github.com/umlaeute/v4l2loopback`⟩.

For the rest of this document it is assumed that your virtual webcam is `/dev/video0`. Replace accordingly if needed.

3. Writing to the virtual Webcam

Basically it works like writing to a file, but we use these options for output with ffmpeg:

```
-vcodec rawvideo -threads 0 -pix_fmt yuv420p -f v4l2 /dev/video0
```

3.1. Sending from Screen

Invoke ffmpeg like so.

```
ffmpeg -f x11grab -r 15 -s 1280x720 -i $DISPLAY.0+0,0 -vcodec rawvideo -threads 0 -
```

› Make sure that you are actually running X. E.g. by executing `loginctl show-session $(awk '/tty/ {print $1}' <(loginctl)) -p Type | awk -F= '{print $2}'` See this stackexchange question ⟨`https://unix.stackexchange.com/questions/202891/how-to-know-whether-wayland-or-x11-is-being-used`⟩.

The meaning of the individual options is:

option	meaning
`-f x11grab`	input format is x11grab: use an x screen
`-r 15`	framerate: 15 FPS
`-s 1280x720`	resolution: capture 1280 (horizontal) by 720 (v ertical) pixels
`-i $DISPLAY.0+0,0`	Capture from Display $display (generally :0), of fset of 0,0 pixels
`-vcodec rawvideo`	output format is ra wvideo
`-pix_fmt yuv420p`	output pix el format for individual pixels (depth, byte order...) is yuv420p
`-f v4l2`	enforce output format
`-threads 0`	use optimal number of threads
`/dev/video0`	file to write to

3.2. Sending from File

Just specify regular input options. Typically, this is `-i FILE`.

```
ffmpeg -re -stream_loop -1 -i video.mp4 -vcodec rawvideo -threads 0 -pix_fmt yuv420
```

option	meaning
`-re`	replay at real speed of video (instead of replaying video as fast as possible)
`-stream_loop -1`	loop video infinite times
`-i video.mp4`	read from file video.mp4

For other options see above.

4. Notes

4.1. VLC

When writing to Video file from your X-Device to check if everything is working avoid using the H.264 codec (`.mp4`) and checking it with VLC. At the time of writing, VLC has a nasty bug that will only display a black video. This problem disappears when using either different encoding, e.g. webm, or by using a different player, e.g. ffplay.

4.2. Mirrored Image

When using videoconferencing software check if your image is flipped. Some videoconferencing systems do that, like nextcloud talk for example, to make show your webcam like you would look in a mirror. This is obviously mirrored and the other participants will see you the correct way around. So check with your specifically used software.

To mirror the image of your virtual webcam use ffmpegs hflip filter. If are not applying filters yet, insert this option:

```
-vf hflip
```

Alternatively you must append the filter hflip to your existing ffmpeg filter graph.

4.3. Testing Your Output

Use ffplay to show the current state of the webcam:

```
ffplay /dev/video0
```

5. See also

- **ffmpeg**(1): write to virtual webcam
- **v4l2loopback**: kernel module for creating virtual webcams
- **vlc**(1): do not use to test x11grab when using H.264 (`.mp4`); use ffplay instead
- **ffplay**(1): show current webcam output