FFmpeg Basics

Bei FFmpeg handelt es sich um eine mächtige Sammlung an freien Computerprogrammen und Programmbibliotheken, die digitales Video- und Audiomaterial aufnehmen, konvertieren, senden, filtern und in verschiedene Containerformate verpacken können. FFmpeg wird zu meist über die Kommandozeile bedient, es bestehen aber auch unterschiedliche grafische Oberflächen für die unterschiedlichsten Einsatzzwecken. Zum Beispiel Videoschnitt Programme, welche auf FFmpeg aufbauen oder Programme um Videos von zum Beispiel DVDs herunter zu bekommen. Auf dieser Seite beschäftigen wir uns mit dem streamen eines Desktops mithilfe FFmpeg auf der Kommandozeile.

Download und Installation

FFmpeg steht für verschiedene Betriebssystemen zur Verfügung und kann von der offiziellen Seite, sowie aus den Paketquellen des jeweiligen Paketmanagers heruntergeladen werden.

Windows

Installer herunterladen und ausführen:

FFmpeg: https://ffmpeg.org/download.html#build-windows

GitHub Source: https://github.com/obsproject/obs-studio/releases

Aus dem Quellcode kompilieren

Dokumentation: https://trac.ffmpeg.org/wiki/CompilationGuide

macOS

Installer herunterladen und ausführen:

FFmpeg: https://ffmpeg.org/download.html#build-mac

GitHub Source: https://github.com/obsproject/obs-studio/releases

Aus dem Quellcode kompilieren

Dokumentation: https://trac.ffmpeg.org/wiki/CompilationGuide

Mittels `brew` installieren:

```
brew install ffmpeg
```

macOS

Installer herunterladen und ausführen:

FFmpeg: https://ffmpeg.org/download.html#build-mac

GitHub Source: https://github.com/obsproject/obs-studio/releases

Aus dem Quellcode kompilieren

Dokumentation: https://trac.ffmpeg.org/wiki/CompilationGuide

Mittels `brew` installieren:

```
brew install ffmpeg
```

GNU/Linux

details

Die Installation für Linux kann auf verschiedenen Wegen erfolgen.

Ubuntu:

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

Arch Linux und Manjaro:

details

```
sudo pacman -S ffmpeg
```

Im AUR befinden sich verschiedene PKBUILDs um FFmpeg in unterschiedlichen Ausführungen zu bauen:

- ffmpeg-full
  - Die aktuelle stabile Version von FFmpeg wird komplett aus dem Quellcode gebaut.
- ffmpeg-full-git
  - Der aktuelle Release Canidate von FFmpeg wird komplett aus dem Quellcode gebaut.
- ffmpeg-amd-full
  - FFmpeg wird in der aktuellsten stabilen Version für AMD Hardware gebaut.
- ffmpeg-amd-full-git
  - Der aktuelle Release Canidate von FFmpeg wird für AMD Hardware gebaut.
- ffmpeg-cuda
  - FFmpeg wird in der aktuellen stabilen Version mit CUDA (Nvidia) Unterstützung gebaut.

Debian:

details

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

Fedora:

details

```
sudo dnf -y install ffmpeg
```

Gentoo:

details

```
emerge --ask media-video/ffmpeg
```

NixOS:

details

https://github.com/NixOS/nixpkgs/blob/master/pkgs/development/libraries/ffmpeg-full/default.nix

OpenMandriva Lx4

details

```
dnf install ffmpeg
```

Solus

details

```
eopkg install ffmpeg
```

Void

details

```
sudo xbps-install ffmpeg
```

Einführung

Es bestehen unterschiedliche Möglichkeiten mittels FFmpeg zu streamen. Der Einsatzzweck kann hierbei sehr unterschiedlich sein. Zum Beispiel soll ein bestehender RTMP Stream abgegriffen und umgeleitet werden, oder es befinden sich Videos auf einem Server, welche gestreamt werden sollen, damit der eigene PC nicht dauerhaft laufen muss.

Für diesen Wikiartikel wurde `FFmpeg` in Version **n5.0.1** unter Arch Linux verwendet. Die Syntax, Optionen oder Argumente können je nach Version und Betriebssystem abweichen.

Als Beispiel Zielserver wird eine eigene Owncast Instanz gewählt.

Lokales Video streamen

Um mit FFmpeg ein lokales Video an ein Zielserver zu streamen muss an FFmpeg die Quelle mittels -i angegeben werden, sowie das Ziel zum Schluss angehangen werden. Dazwischen können Optionen gesetzt werden. Zum Beispiel, ob das Video vor dem streamen konvertiert werden soll.

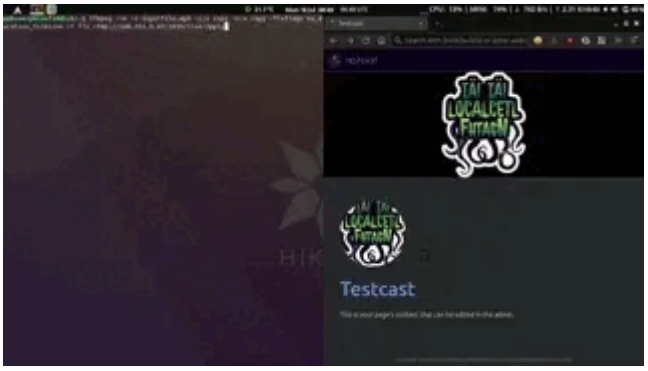
```
ffmpeg -re -i inputfile.mp4 -c:v copy -c:a copy -f flv rtmp://127.0.0.1:1935/live/<streamkey>
```

Mit dem Befehl von oben liest ffmpeg die datei inputfile.mp4 ein und sendet das Video als flv zum Zielserver, welcher via RTMP auf Port 1935 erreichbar ist. Folgende Optionen wurden an FFmpeg mit angeheftet:

`-re` Der Input wird mit nativer Bildwiederholungsrate eingelesen.
`-i` Mittels dieser Option wird der Pfad zur Datei gesetzt.

> Wichtig. Der Dateiname darf keine Leerzeichen haben. FFmpeg hat hiermit Probleme, auch wenn man diese escaped.

`-c:v copy` Gibt an, dass der Video Codec beibehalten werden soll und FFmpeg nichts en/decodieren muss.
`-a:v copy` Wie beim Video Codec wird auch der Audio Codec beibehalten.
`-f flv` Mit dieser Option wird das Format FLV erzwungen, welches zwingend beim RTMP Streams ist.



> Die Optionen `-c:v copy` und `-a:v copy` sind wichtig zu setzen, sollte der jeweilige Computer langsamer en/decodieren, als mit 1x Geschwindigkeit. Im Fall einer langsamen berechnung des Videos, wird FFmpeg den Stream abbrechen, da das Video langsamer berechnet wird als FFmpeg senden kann und somit Frames erwartet, welche FFmpeg noch nicht kennt. Es kommt zu einem `Broken Pipe` und `connection reset by peer`.

Weitere Optionen

Sollte eine Grafikkarte vorhanden sein, mit welcher ein reencoding schnell genug durchgeführt werden kann, kann das Video während des streamens entsprechend auf Größe und Format gebracht werden. Dies ist besonders interessant, falls man das Video nicht im Vorfeld skaliert hat und die Bandbreite im Upload begrenzt ist.

`-video_size 1280x720` Gibt die Ausgabeauflösung an
`-c:v libx264` mit der Option wird der H264 Codec benutzt
`-preset veryfast` Das vorgefertigte Profil für eine schnelle Berechnung, anstelle von Qualität wird benutzt
`-b:v 3000k` Die Bitrate des Videos wird auf 3000 KB/s eingestellt
`-maxrate 3000k` Die maximale Bitrate des Videos entspricht 3000 KB/s
`-bufsize 6000k` Die Buffer größe muss das doppelte der maximalen Bitrate entsprechen.
`-g 60` Die g Option stellt die Group Of Picture länge an. Hohe werte verursachen eine größere Kompression.
`-c:a aac` Als Audio Ausgabe Codec soll aac benutzt werden
`-b:a 128k` Dies stellt ein mit welcher Bitrate das Audio im Video gesendet werden soll. 192KB/s entsprich CD Qualität
`-ar 44100` Dies stellt die Samplerate Hz ein.

Streamen von Webcam und Mikrofon (Linux)

Falls eine USB Webcam gestreamt werden muss die Audioquelle, sowie die Video-Hardware angegeben werden. Um die Audio Hardware zu bestimmen, nutzen wir `$ aplay --list-devices` und listen jegliche Hardware auf.

```
$ aplay -l
**** Liste der Hardware-Geräte (PLAYBACK) ****
Karte 0: PCH [HDA Intel PCH], Gerät 0: ALC892 Analog [ALC892 Analog]
  Sub-Geräte: 0/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 3: HDMI 0 [HDMI 0]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 7: HDMI 1 [HDMI 1]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 8: HDMI 2 [HDMI 2]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 9: HDMI 3 [HDMI 3]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 10: HDMI 4 [HDMI 4]
```

```
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 11: HDMI 5 [HDMI 5]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 12: HDMI 6 [HDMI 6]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 2: Creative [HDA Creative], Gerät 0: ALC898 Analog [ALC898 Analog]
  Sub-Geräte: 0/1
  Sub-Gerät #0: subdevice #0
Karte 3: Device [USB Advanced Audio Device], Gerät 0: USB Audio [USB Audio]
  Sub-Geräte: 0/1
  Sub-Gerät #0: subdevice #0
```

In der Ausgabe können wir sehen, welche Audio-Hardware vom Betriebssystem erkannt wurde und zur Verfügung steht. Nun müssen wir die passende `Karte` und `Subdevice` wählen. In diesem Beispiel will ich die Karte 3, das USB Mikrofon benutzen. Sprich, meine Hardware ist die `3,0`.

Alternative Möglichkeiten die Hardware zu definieren kann hier nachgelesen werden:

https://en.wikibooks.org/wiki/Configuring_Sound_on_Linux/HW_Address

Nun muss ich die Video-Hardware Bestimmen, welche genutzt werden soll. Diese befindet sich unter `/dev/video*`. Diese wird als Input mit `-i` definiert. In meinen Fall ist es die Webcam unter dem Pfad `/dev/video2`. Nun muss zusätzlich das Input Format, sowie das ausgehende Videoformat gewählt werden. Der Input wird mit `-input_format yuyv422` eingestellt. Dieses Format kommt von der Webcam und sollte für den ausgehenden Stream in `yuv420p` mit `-vf "format=yuv420p"` umgewandelt werden. Mit der Kombination aus den vorherigen Befehlen und zusätzlichen Optionen sieht der gesamte Befehl wie folgt aus:

```
ffmpeg -f alsa -ac 2 -i hw:3,0 \
  -f v4l2 -framerate 60 -video_size 1280x720 -input_format yuyv422 -i /dev/video2 \
  -c:v libx264 -preset veryfast -b:v 1984k -maxrate 1984k -bufsize 3968k \
  -vf "format=yuv420p" -g 60 -c:a aac -b:a 128k -ar 44100 \
  -f flv rtmp://127.0.0.1:1935/live/<streamkey>
```

Revision #14
Created 18 July 2022 05:20:43 by Kascha
Updated 19 July 2022 06:39:19 by Kascha