Planung LPD-BS und Video-Tools zur Vortragserstellung

Termine und vorgehen

Für unseren LPD-BS planen wir Online mittels Jitsi ein Vortrags und Diskussions Angebot an den Terminen zu fahren:

- 2021-05-15 / 3. Samstag im Mai
- 2021-11-20 / 3. Samstag im November

Zeitlich wieder ab 13 Uhr, Endzeit und Ablauf sind noch festzulegen.

TODOs

- Endzeit /Zeitraum
- Ablauf
  - Jitsi Räume (Vortrag, Diskuss., Open, Close,...)
- Vorträge als Video vorproduzieren
- WERBUNG / Ab wann und wie?
- Datensilo für LPD-Videos? / https://bs-lug.de/videos/start
- Arbeiten mit OBS?
- ..

DONT's

- Darauf achten das nix klappert oder schabt (Handy Headset, Freisprech, etc. welche an der Kleidung reibt)
- Aussprache (deutlich sprechen)
- Aussteuerung (Kein klirren, Anschlagen oder zu leise)

Video Vortrag aufnehmen

- simplescreenrecorder Quelle zum aufnehmen von Fenstern und einsprechen des Ersttones
  - ErkLehrvideo: https://l-p-d.org/fahrplan_212#1211
- Evtl. weiteres Einsprechen
- Bei Fehlern klatschen (lauter Pegel) und nochmals wiederholen, wird am Ende rausgeschnitten.
- Preproduction: Erstschnitt und vorstellen in der BS-LUG
- Postproduction: Zusammenschnitt mit Intro/Offtro, evtl. Audio-Pegel Anpassung, Kleinrechenen und ins Archiv stellen.

Video encodieren (kleinrechnen)

Für ffmpeg muss die GPU richtig gesetzt werden. Im Script ggf. ändern wenn keine Nvidia-Grafikkarte vorhanden.

Es verschiedene Ansätze für Hardwarebeschleunigung je nach Hersteller der Grafikkarte.

- „NVENC“ steht immer für Nvidia Grafikkarten. Wenn man keine hat, kann man den Treiber nicht benutzen.
- „CUDA“ oder „CUVID“ sind auch Hinweise auf Nvidia.

(Script unten)

WICHTIG:

```
ffmpeg -hwaccels
```

Damit bekommt man raus, welche Optionen für Eurer FFMPEG bei der HWUnterstützung verfügbar sind

```
ffmpeg -decoders
```

Hier welche Dekodierhilfen vorhanden sind, dazu der Tip: „ffmpeg -decoders | grep 264“ schränkt das Ergebnis aus das gesuchte ein. Welche davon für Euch die beste ist, hängt von der Grafikkarte ab. Unten gibts Beispiele.

```
ffmpeg -encoders
```

Das gleiche Spiel für Codierer.

Die allgemeine Syntax:

```
ffmpeg { HWDECODING } { EINGABE }  { Streammapping } { ENCODEOPTIONS } { AUSGABE }
```

Beispiel:

```
ffmpeg -hwaccel cuvid -threads 8 -c:v h264_cuvid -i "$1" -map 0:1 -map 0:0 -c:v:0
h264_nvenc -b:v:0 750k -c:a:0 copy $2 -f mp4 "$NAME-klein.mp4"
```

HWDECODING

```
-hwaccel cuvid -threads 8 -c:v h264_cuvid
```

Der Block macht gleich zwei Sachen :

1. Er legt fest, daß wenn die CPU genutzt werden soll, alle 8 Kerne der CPU benutzt werden sollen (setzt voraus, daß man 8 hat, ist also an Eure CPU anzupassen) . Das ist sinnvoll, wenn man neben dem Codieren noch was machen will und z.b. nur 50% der Kerne benutzen möchte, weil man Rechenleistung für die anderen Sachen übrig haben will, da wäre dann bei 8 Kernen die Zahl 4 anzugeben.
2. Die Hardware Decodierung aktivieren:
    - Nvidia: -hwaccel cuvid -c:v h264_cuvid
    - Intel: -hwaccel qsv -c:v h264_qsv
    - Andere: -hwaccel vaapi -c:v h264_vaapi
    - Andere: -hwaccel vdpau -c:v h264_vdpau
    - Oder: -hwaccel auto

Wenn man keine HWDekodierung hat, dann braucht man die Optionen natürlich nicht angeben.

ACHTUNG: Die Codecs heißen unterschiedlich beim Decodieren und Encodieren, auch wenn es die gleiche

Grafikkarte ist. Im Zweifel einfach alle ausprobieren, bis einer paßt oder $suchmaschine fragen.

EINGABE:

```
-i Filmfile-REIN.mp4
```

Streammapping:

```
-map 0:1 -map 0:0
```

Geht davon aus, daß es einen Videostream und einen Audiostream gibt. Die Zahlen sind gleich wichtig, wenn wir angeben, wie was kodiert werden soll, also genau hinschauen!

ENCODEOPTIONS:

```
-c:v:0 h264_nvenc -b:v:0 750k -c:a:0 copy
```

-Codec:Video:STREAMNUMMER CODEC also hier Videostream 0 soll als Encoder den Nvidia h264_nvenc mit HW unterstützung benutzen

-Bitrate:Video:0 hier also Videostream 0 soll 750k = 2 Mb/s haben.

-Codec:Audio:STREAMNUMMER CODEC hier Audiostream 0 soll „copy“ ⇒ Nichts verändern. D.b. er nimmt das was im Eingabefile ist und benutzt es unverändert in der Zieldatei. Spart Zeit und Ziel des Scripts ist es nur die Videorate runter zu bekommen.

Wenn man jetzt kein Nvidia haben will, dann sucht mach sich seinen HWCodec aus der Liste raus:

CPU:

```
V..... libx264              libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10
(codec h264)
V..... libx264rgb           libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 RGB
(codec h264)
V..... h264_v4l2m2m         V4L2 mem2mem H.264 encoder wrapper (codec h264)
```

Intel:

```
V..... h264_qsv             H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 (Intel Quick
Sync Video acceleration) (codec h264)
```

Rest:

```
V..... h264_vaapi           H.264/AVC (VAAPI) (codec h264)
```

Nvidia:

```
V..... nvenc                NVIDIA NVENC H.264 encoder (codec h264)
V..... nvenc_h264           NVIDIA NVENC H.264 encoder (codec h264)
V..... h264_nvenc           NVIDIA NVENC H.264 encoder (codec h264)
```

AUSGABE:

```
-f mp4 "$NAME-neu.mp4"
```

-f FORMAT hier also MP4 und dann schon der neue Dateiname.

Tools:

video_encode_2_low.sh

```
#!/bin/bash

NAME=$(basename "$1" .mp4)

if [ -z ${NAME} ] ; then
    echo -e "\nusage:"
    echo -e "\tDateinamen.MP4 des kleinzurechneden Videos angeben."
    exit
fi

ffmpeg -hwaccel cuvid -threads 8 -c:v h264_cuvid -i "$1" -map 0:1 -map 0:0 -
c:v:0 h264_nvenc -b:v:0 750k -c:a:0 copy $2 -f mp4 "$NAME-klein.mp4"
```

```
#-- some tools n tricks

#- get movie streams / welche Streams gibt es in Video A und B?
ffmpeg -i a.mkv -i b.mkv  2>&1| grep -i stream

#- compose the new with audioshift (-itsoffset -0.50)
ffmpeg -i a.mkv    -itsoffset -0.00    -i b.mkv         -map '0:0' -map '1:0' -map
'0:1' -map '1:2' -map '1:4' -c copy ccc.mkv

#- change audio default / Umsetzen des Default-Flags (zB. von Sprache a zu b)
mkvpropedit --edit track:a1 --set flag-default=1 --edit track:a2 --set flag-
default=0 vid.mkv
```

From:
https://bs-lug.de/ - **BS-LUG**

Permanent link:
https://bs-lug.de/activitys/2021/20210123_lpd_video_tools/start

Last update: **2025-01-30 03:57**