Video Encoding Systems P2: Python and Video

Carlos Hortensius Fernández
NIA: 205437

December 2020

Link to the repository: Pratice 2 Carlos Hortensius

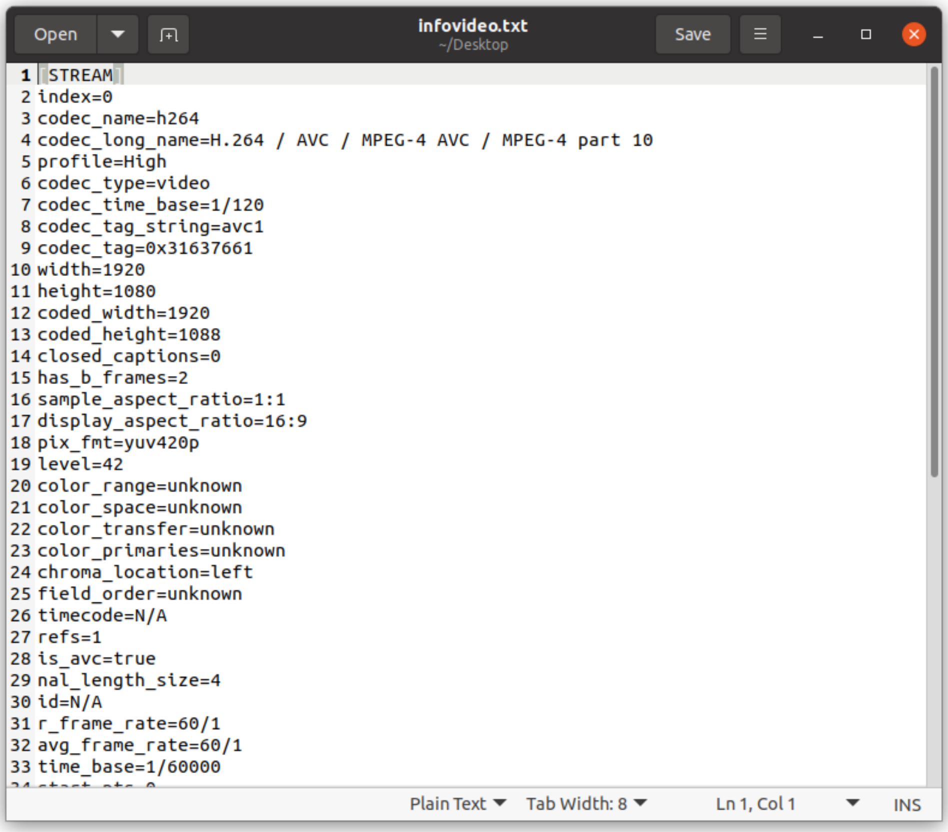
1 Relevant data from the container

In this first exercise we are going to Create a python script able to parse the 'ffmpeg –i BBB.mp4' file, which can mark at least 3 relevant data from the container. To do this, I have used the following ffmpeg commands:

```
ffprobe -show_streams -select_streams v:0 input_video >infovideo.txt'
```

```
ffprobe -show_streams -select_streams a:0 input_video >infoaudio.txt'
```

This two commands save all the information of the streams in a text file, the first one for the video data, and the second one for the audio data. After that, I went into the text files and looked for the different data I wanted to extract. In this case I have read from the text file the lines in which the audio and video codecs were shown and the size of the video, and I have shown them on screen with a print. The text files look like this:



```
[STREAM]
index=0
codec_name=h264
codec_long_name=H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10
profile=High
codec_type=video
codec_time_base=1/120
codec_tag_string=avc1
codec_tag=0x31637661
width=1920
height=1080
coded_width=1920
coded_height=1088
closed_captions=0
has_b_frames=2
sample_aspect_ratio=1:1
display_aspect_ratio=16:9
pix_fmt=yuv420p
level=42
color_range=unknown
color_space=unknown
color_transfer=unknown
color_primaries=unknown
chroma_location=left
field_order=unknown
timecode=N/A
refs=1
is_avc=true
nal_length_size=4
id=N/A
r_frame_rate=60/1
avg_frame_rate=60/1
time_base=1/60000
```

The code used for that is the following:

```
with open("infovideo.txt") as f:
    data = f.readlines()[2]
    f.close()

with open("infovideo.txt") as f:
    w = f.readlines()[9]
    f.close()

with open("infovideo.txt") as f:
    h = f.readlines()[10]
    f.close()

print('Video Codec: ', data)

with open("infoaudio.txt") as f:
    data = f.readlines()[2]
print('Audio Codec: ',data)

print('Dimensions of the video')
print(w)
print(h)
```

2 Rename files

In this case I don't think I've really understood what the statement asked for, what I've finally done is to create a python script that changes the name of the file you enter by console with the following command:

```
    subprocess.call(['mv', file, newname])
```

3 Resize

Given an input file, change its dimensions. Once again the user is asked to enter the directory where the video is located and also the name of the video. Then he is also asked to enter the new dimensions he wants for his input file. Finally, two options are added in other to know if the input file is a photo or a video:

```
if option == 1:
    cmd = 'ffmpeg -i %s -vf scale=%d:%d resized.mp4' % (file, width, height)
if option == 2:
    cmd = 'ffmpeg -i %s -vf scale=%d:%d resized.png' % (file, width, height)

subprocess.call(cmd, shell=True)
```

4 Transcode files

Now I am creating a Python script able to transcode the input into an output with another codec that we've seen in the Theory class. The codecs that we have seen are the following: MPEG4 (mpeg4), h.264 (libx264), h.265 (libx265) and AV1 (libaom-av1). So finally this commands are executed:

```
if option == 1:
    subprocess.call(['ffmpeg', '-i', file, '-c:v', 'mpeg4', 'Newcodecfile.mp4'])
if option == 2:
    subprocess.call(['ffmpeg', '-i', file, '-c:v', 'libx264','Newcodecfil.mp4'])
if option == 3:
    subprocess.call(['ffmpeg', '-i', file, '-c:v', 'libx265', 'Newcodecfile.mp4'])
if option == 4:
    subprocess.call(['ffmpeg', '-i', file, '-c:v', 'libaom-av1',  'Newcodecfile.mp4'])
```

5 Python Script

In this final exercise, I have integrated some or all the previous exercises into one script, which allows you to choose which variables of the input video you'd like to change.