Inhoud van een container

- Maak een zeer beknopte tabel met voor elk multimediabestand het containerformaat en de compressiestandaard die gebruikt werd voor audio en video?

Bestand	Containerformaat	Compressiestandaard
ElFuente2b.avi	Audio Video Interleave (AVI)	MPEG-4, MP3
ElFuente2b.webm	WebM	VP8, Vorbis
ElFuente2b.mp4	ISO Base Media File Format	H264, AAC
ElFuente2b.mkv	Matroska (MKV)	HEVC, Opus

- Vat de gegevens van de verschillende programma's samen in een tabel. Vermeld voor elk programma, de verschillende stromen en voor elke stroom zeker audio en video standaard, bitsnelheid, video resolutie en audio taal indien dit af te leiden is uit de probe informatie.

Programma	Stream	Video	Audio	Bitsnelheid	Resolutie	Taal
1	#0.12	HEVC			3840x2160	
	#0.5		AAC	223 Kb/s		Und
2	#0.0	MPEG-2		15 Mb/s	720x570	
	#0.1		MP2	128 Kb/s		Eng
	#0.2		MP2	128 Kb/s		Eng
	#0.3		MP2	128 Kb/s		Eng
	#0.4		MP2	128 Kb/s		Eng
16982	#0.9	H264				
	#0.10		MP3			Ita
	#0.11		AC3			Oth
16983	#0.6	H264				
	#0.7		MP3			Ita
	#0.8		AC3			Oth

- Probeer elk programma in de transportstroom te bekijken en elk audiokanaal te beluisteren aan de hand van de meegeleverde mediaplayers. Wat is volgens jou de functie van de verschillende geluidskanalen?

Een ander geluid afspelen op de video. Bijvoorbeeld naar een andere taal switchen.

- Kan je elk videoprogramma bekijken? Geef een mogelijke verklaring.

Neen, want programma 16982 en 16983 zijn gecodeerd. Deze kunnen we dus niet afspelen.

- Bedenk een situatie waarin een programma meerdere videokanalen zou bevatten en slechts 1 audiokanaal.
Een race waarbij er 1 commentator te horen is. Er zijn meerdere video's waartussen men kan switchen en meerdere camera's verdeeld zijn over verschillende delen van het circuit.

Een container samenstellen

- Rapporteer de commandolijnen zodat wij zelf de volgende output kunnen genereren:

ElFuenteb.avi

ffmpeg -i ElFuente.mkv -c:v mpeg4
-c:a libmp3lame
-b:v 4000k -b:a 128k ElFuenteb.avi

ElFuenteb.webm

ffmpeg -i ElFuente.mkv -c:v libvpx
-c:a libvorbis
-b:v 4000k -b:a 128k ElFuenteb.webm

ElFuenteb.mp4

ffmpeg -i ElFuente.mkv -c:v libx264
-c:a aac
-b:v 4000k -b:a 128k ElFuenteb.mp4

ElFuenteb.mkv

ffmpeg -i ElFuente.mkv -c:v libx265
-c:a libopus
-b:v 4000k -b:a 128k ElFuenteb.mkv

- De volgende twee commando's resulteren beiden in een container met daarin een videostroom volgens de MPEG-4 Part-2 standaard. Wat doen de volgende twee commando's dan verschillend?

Het formaat is voor beide commando’s hetzelfde, maar de manier waarop de bytestream aangemaakt wordt is verschillend. Ze gebruiken een verschillend compressiealgoritme.

Hercoderen tegenover stream copy

- Geef dus nu de 4 commando's zoals in voorgaande opgave, maar nu aan de hand van stream copy van ofwel het video gedeelte of het audiogedeelte indien mogelijk. In de andere gevallen kopieer je gewoon het resultaat uit vorige opgave.

ElFuenteb.avi

ffmpeg -i ElFuente.mkv -c:v mpeg4
-c:a libmp3lame
-b:v 4000k -b:a 128k ElFuenteb.avi

ElFuenteb.webm

ffmpeg -i ElFuente.mkv -c:v libvpx
-c:a libvorbis
-b:v 4000k -b:a 128k ElFuenteb.webm

ElFuenteb.mp4

ffmpeg -i ElFuente.mkv -c:v copy
-c:a aac
-b:v 4000k -b:a 128k ElFuenteb.mp4

ElFuenteb.mkv

ffmpeg -i ElFuente.mkv -c:v libx265
-c:a libopus
-b:v 4000k -b:a 128k ElFuenteb.mkv

- Wat zijn de voordelen van stream copy tegenover hercoderen?

1) Met betrekking tot computationele complexiteit. Waarom?

Het decoderen en encoderen is nu overbodig. Simpelweg kopiëren is veel sneller en minder rekenkracht is vereist.

2) Met betrekking tot videokwaliteit. Waarom?

De videokwaliteit is exact hetzelfde. De video is niet gecomprimeerd.

Ondertitels

- Worden de ondertitels weergegeven? Hoe komt dit en wat is het verschil met de hierop volgende methoden?
Ja, de ondertitels worden weergegeven.
De mediaspeler zoekt vanzelf naar het ondertitelbestand.
In de onderstaande methoden worden de ondertitels toegevoegd aan de mp4 container.

- Beschrijf kort het verschil dat je opmerkt?
Bij de 1ste methode wordt de tekst letterlijk op het frame geplaatst.
Bij de 2de methode wordt enkel het ondertitelbestand aan de container toegevoegd.
Er is dus een extra stream.

- Kan je bij beide bestanden de ondertitels afzetten?
Neen, in het 1ste bestand kun je de ondertitels niet afzetten.
In het 2de bestand kan dat wel.

- In welke mediastroom zitten de ondertitels bij zowel ElFuenteTalk_sub.mp4 als ElFuenteTalk_sub2.mp4?
In ElFuenteTalk_sub.mp4 zitten de ondertitels in de videostream.
In ElFuenteTalk_sub2.mp4 zitten de ondertitels in een aparte stream.

- Welke methode is het best met betrekking tot videokwaliteitsverlies en waarom?
De 2de methode is het best. Er is niets veranderd aan de videostream en de subtitels kunnen eventueel later worden aangepast.
In de 1ste methode is informatie over pixels onder de letters verloren.

- Welke methode is het snelst en hoe komt dit?
De 2de methode is het snelst. Hier moet enkel het ondertitelbestand toegevoegd worden aan de mp4 container.
In de 1ste methode moet de tekst frame voor frame toegevoegd worden, dit neemt veel tijd in beslag.

- Wat wordt er gebruikt op DVD en Blu-Ray als er meerdere talen ondersteund moeten worden?
Uiteraard wordt er hierbij gebruikt gemaakt van de 2de methode.

Een selectie uit een videobestand maken

1. ffmpeg -i ElFuente5b.mp4 -ss 2.5 -t 5 -c:v libx264 -c:a copy -b:v 1063000 ElFuente5b_1.mp4

- Welk type beeld is het eerste beeld van de output?

Het eerste beeld van de output is van het type I. Via het commando 'ffprobe -show_frames ElFuente5b_1.mp4 > test.txt' schrijven we detailinformatie van de zopas gegenereerde video naar een tekstbestand. Via 'coded_picture_number=0' vinden we het eerste beeld.

- Het hoeveelste beeld is dit uit de originele video? Hoe heb je dit gevonden?

De beeldsnelheid van de originele video bedraagt 59.94 fps. Dit betekent dat na 2.5 seconden we aan frame 149 zitten. In de originele video is dit frame een B-type. Het eerste frame van de nieuwe video is van het type I.

2. ffmpeg -i ElFuente5b.mp4 -ss 2.5 -t 5 -c copy ElFuente5b_2.mp4

- Van welk type is het eerste beeld?

Opnieuw van het type I.

- Het hoeveelste beeld is dit uit de originele video? Hoe heb je dit gevonden?

Frame 192. We starten met kopiëren vanaf de eerstvolgende keyframe.

3. ffmpeg -i ElFuente5b.mp4 -ss 2.5 -t 5 -c copy -copyinkf ElFuente5b_3.mp4

- Zoek op wat er gebeurt wanneer je deze parameter gebruikt?

De timestamp waar wij beginnen kopiëren is tussen twee frames. Stream copy zorgt voor een freeze in het begin van de video omdat deze niet start op een keyframe. Via –copyinkf kopiëren we ook de niet-keyframes in het begin van de video.

- Van welk type is het eerste beeld van de output?

Opnieuw van het type I.

- Het hoeveelste beeld is dit uit de originele video? Hoe heb je dit gevonden?

Terug frame 149. Copyinkf start vanaf het huidige frame.

4.

- Zeg voor elke methode (re-encode, stream-copy en copyinkf) in 1 zin wat de methode doet.

Re-encode hercodeert de gespecifieerde stream met kwaliteitsverlies.
Stream-copy kopieert de gespecifieerde stream, zonder veranderingen, zonder kwaliteitsverlies.
De optie copyinkf zorgt ervoor dat het starten met kopiëren begint op de eerste frame van de input stream in plaats van de eerste keyframe.

- Wat zijn de voor- en nadelen van elk van deze methoden?

Re-encode is trager, maar zal de video optimaal comprimeren.
Stream-copy is snel (geen encodering/decodering), heeft geen kwaliteitsverlies, maar kan freezen als kopiëren niet start bij een I-frame.
Copyinkf lost probleem van stream-copy op, maar zal niet optimaal gecomprimeerd zijn.

- Wanneer zou je welke methode gebruiken? Geef een voorbeeld.
Re-encode wanneer ik een stukje video wil doorsturen naar een andere persoon over het web, zodat het bestand niet té groot is. Copyinkf om een stukje video eruit te halen om te tonen in voorstelling of zaal.

TEDxGhent:Automatische videomontage

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.

ffmpeg -i talk.mp4 -i logo_alpha.png
-filter_complex “overlay=15:15”
solution_logo.mp4

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.

ffmpeg -i header.mp4 -i talk.mp4
-filter_complex “concat =a=1[v] [a1]”
-map “[v]” -map “[a1]”
solution_merge.mp4

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.

ffmpeg -i talk.mp4 -i start_pancarte.png -i end_pancarte.png
-filter_complex "[0][1]overlay=0:0:enable='between(n,0,76)'[v1];
[v1][2]overlay=0:0:enable='between(n,12390,12465)'[v2]"
-map "[v2]" -map “[0:a]”
solution_pancarte.mp4

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.

ffmpeg -i header.mp4 -i start_pancarte.png -i talk.mp4 -i logo_alpha.png -i end_pancarte.png
-filter_complex “[2][3]overlay=15:15[v1];
[v1][1]overlay=0:0:enable=’between(n,0,76)’[v2];
[v2][4]overlay=0:0:enable=’between(n,12390,12465)’[v3];
[0][0:a][v3][2:a]concat=a=1[v][a]”
-map "[v]" -map "[a]"
solution_total.mp4

- Geef een tabel met voor elke parameter een beknopte uitleg waarom je die hebt toegevoegd en waarom op die positie. Dit zou overeen moeten komen met de uitleg die je zou geven aan iemand die verder zou moeten werken op dit commando.

[2][3]overlay=15:15[v1]	[2] slaat op talk.mp4 [3] slaat op logo_alpha.png Gebruik [3] als overlay op [2] en noem dit [v1]
[v1][1]overlay=0:0:enable='between(n,0,76)'[v2]	[1] slaat op start_pancarte.png Gebruik [1] als overlay op [v1] en noem dit [v2]
[v2][4]overlay=0:0:enable='between(n,12390,12465)'[v3]	[4] slaat op end_pancarte.png Gebruik [4] als overlay op [v2] en noem dit [v3]
[0][0:a][v3][2:a]concat=a=1[v][a]	[0] slaat op header.mp4 [2] slaat op talk.mp4 Concateneer video [0] met [v3] en audio [0:a] met [2:a]

We doen eerst de overlay van het logo, omdat de pancarte het logo dan overschrijft. Vervolgens plaatsen we de pancartes, want als we eerst concateneren met header.mp4 kennen we de juiste frames voor de pancartes niet meer.