HOW TO BUILD A FREE ENCODER/PACKAGER WITH WATCH FOLDER OPERATION USING OPEN SOURCE TOOLS

Jan Ozer

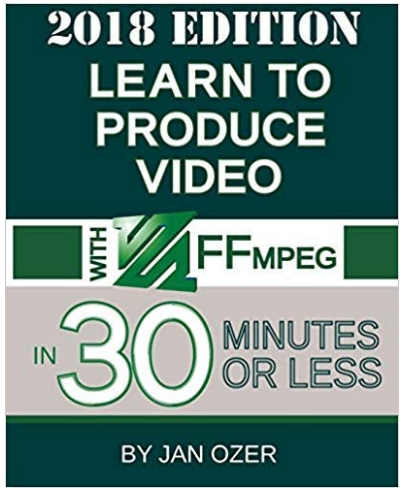
www.streaminglearningcenter.com

Agenda

- Shameless plug
- Introduction
- FFmpeg
- Bento4
- Automating it all with Ubuntu

Book/Course Which Some Materials Derived

- Book ($34.95 on Amazon):
  - Includes H.264/H.265
  - Creation of variant playlists with FFmpeg
  - Variant/master playlists with Apple tools
  - All scripts downloadable after purchase
  - Show special:
    - - Buy book on Amazon/wherever
    - - Email receipt to janozer@gmail.com
    - - get free copy of PDF ($24.95 value)
    - - Valid till 6/31
- Course: $29.95 - http://bit.ly/learn_ffmpeg
  - 32 short lessons, over 3 hours of video
  - Same content as above



Introduction

- There are always multiple ways; seldom is there a single correct “one”
- We’re showing minimum necessary commands; there are lots more configuration options
- Location of configuration option in string typically doesn’t matter
- If you don’t choose a configuration option, FFmpeg uses the default
- Configurations in command line override defaults

Script 1: Choosing Codec

```
ffmpeg -i   TOS_1080p.MOV   -c:v libx264      TOS_s1.mp4
```

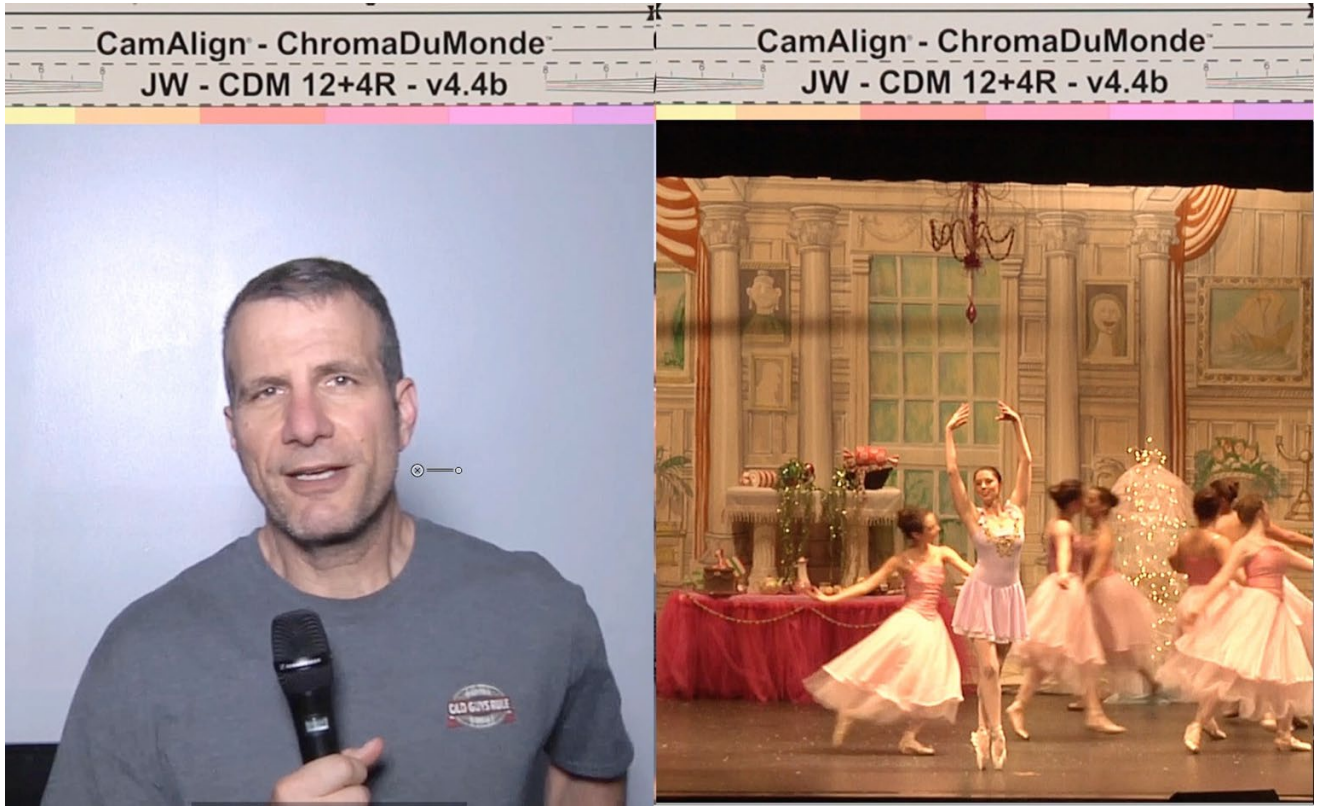
Program input file video codec Output file

- Input file: 1080p file in MOV format
  - YUV video
  - PCM audio
- Simple script means that you accept all FFmpeg defaults
- Generally acceptable for home movies; not acceptable for streaming, particularly adaptive streaming

Encoding Output - Default

- Codec: x264
  - Data rate: 15 Mbps
  - Bitrate control: average bitrate
  - Key frame: 250
  - Scene change: Yes
  - Resolution: same (1080p)
  - Frame rate: same (24)
  - Profile: High
  - CABAC: Yes
  - x264 preset: Medium
  - B-frames: preset (3)
  - B-adapt: preset (1)
  - Reference frames preset (3)
- Audio codec: AAC
  - Audio channels: 2
  - Audio samples: 48 khz
  - Audio bitrate: 2277 b/s
- Other Topics
  - Encoding multiple files
  - Converting to HLS

Bitrate Control

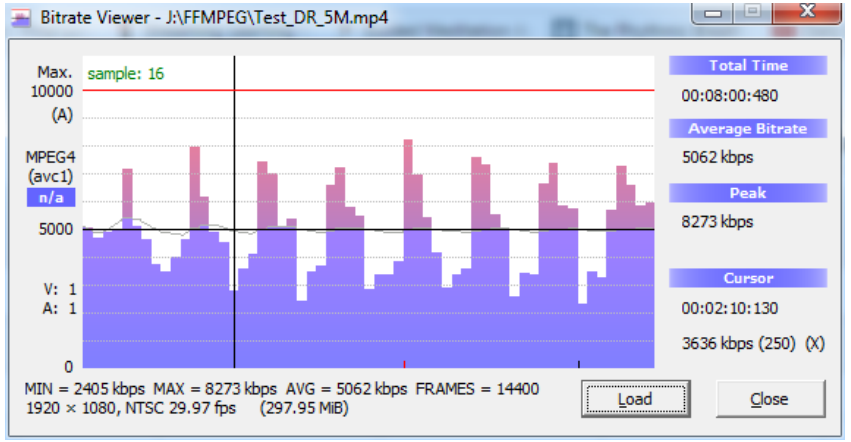


30 seconds talking head/30 seconds ballet

Setting Data Rate-Video

```
-b:v 5000k
```

bitrate video



- Sets video bitrate to 5 mbps

- No real bitrate control
- Spikes may make file hard to play

Images from Bitrate Viewer

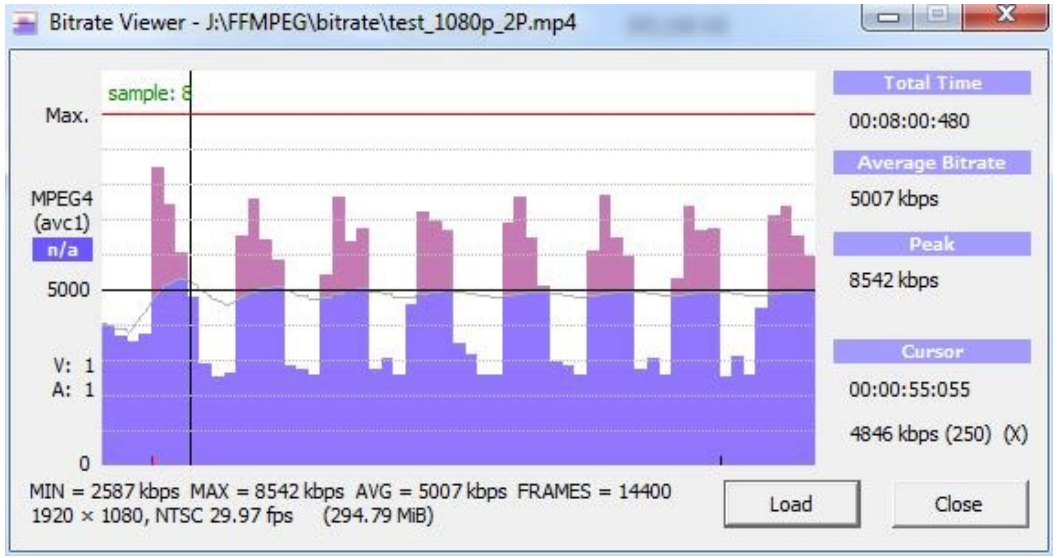
Setting Data Rate-Two-Pass

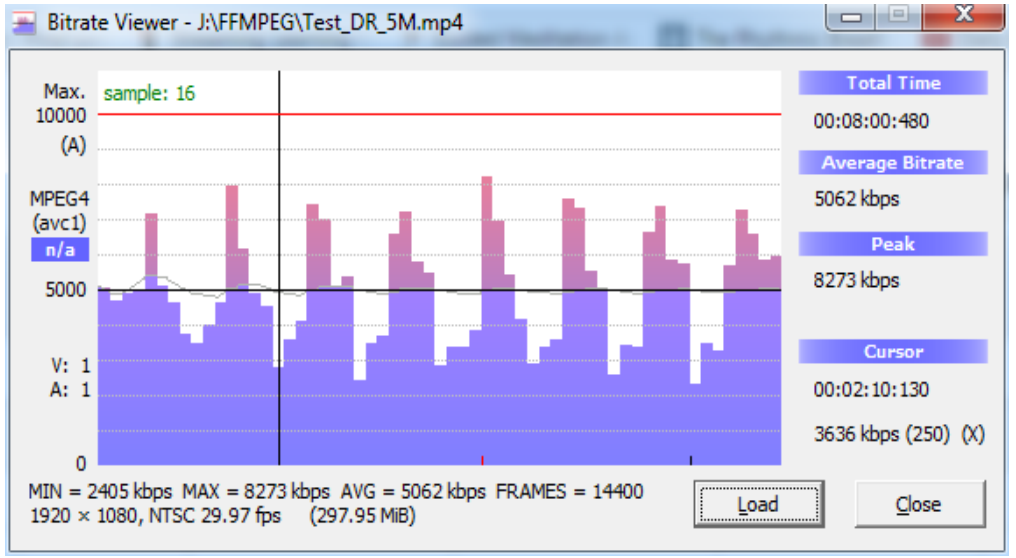
```
ffmpeg -y -i Test_1080p.MOV -c:v libx264 -b:v 5000k -pass 1  -f
mp4 NUL && \

ffmpeg -i  Test_1080p.MOV -c:v libx264 -b:v 5000k -pass 2
Test_1080p_2P.mp4
```

- Line 1:
- -y - overwrite existing log file
- - pass 1 - first pass, no output file
- -f mp4 - output format second pass
- NUL - creates log file cataloguing encoding complexity (can name log file if desired)
- && \ - run second pass if first successful

- Line 2:
- -pass 2 - find and use log file for encode
- Test_1080p_2P.mp4 - output file name
- Note - all commands in first pass must be in second file; can add additional commands in second line (more later)

Setting Data Rate-Two-Pass





- Two-Pass Encode
  - Improved bitrate control (5007 kbps)
  - Higher peak!

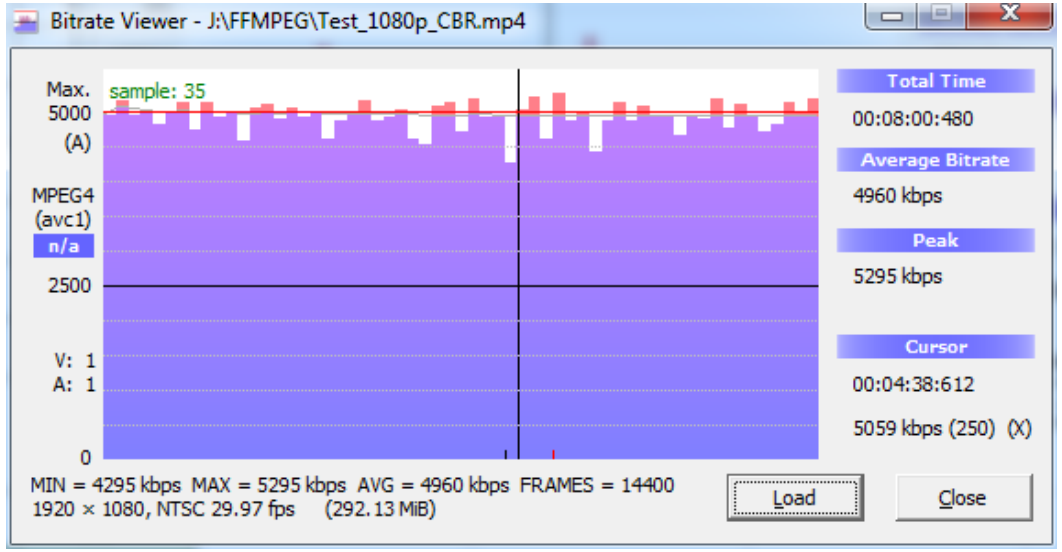
- Single-Pass Encode
  - Poor data rate control (5062 kbps)

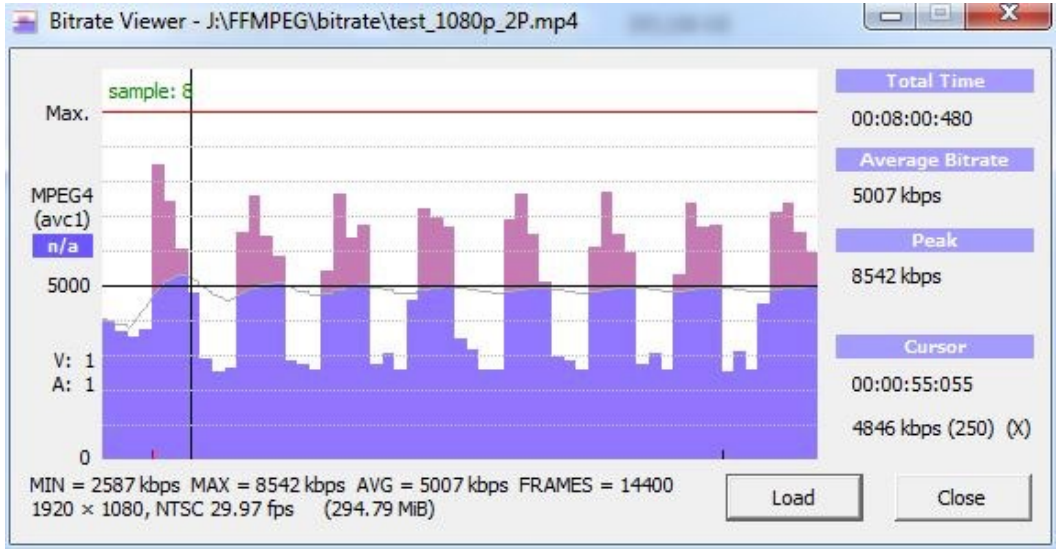
Setting Data Rate-CBR

```
ffmpeg -y -i  test_1080p.MOV -c:v libx264  -b:v 5000k -pass 1
-f mp4 NUL && \ (same)

ffmpeg -i  test_1080p.MOV -c:v libx264  -b:v 5000k -maxrate
5000k -bufsize 5000k -pass 2  test_1080p_CBR.mp4
```

- Line 2:
- - maxrate 5000k - maximum rate same as target
- - bufsize 5000k - VBV (Video Buffering Verifying) buffer set to one second of video (limits stream variability)

Setting Data Rate-Two-Pass





- CBR - not flat line
  - Peak is 5295
  - Much less variability
  - Lower overall quality (not much)
  - Can show transient quality issues

- Two-pass ABR
  - Poor data rate control
  - Better overall quality

CBR Can Show Transient Quality Issues

- http://bit.ly/vbr_not_cbr

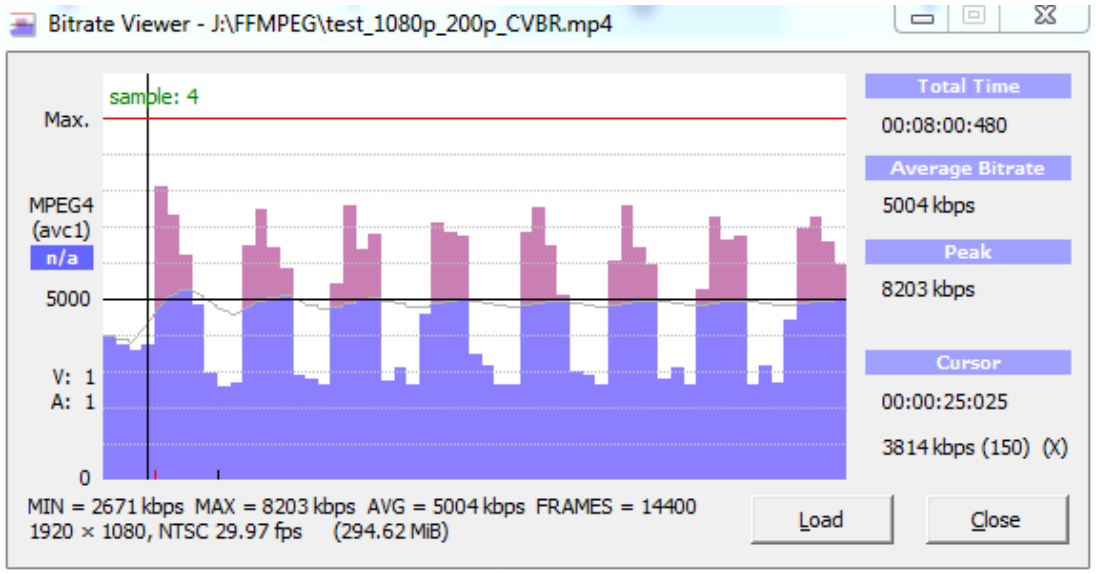
Setting Data Rate-Constrained VBR

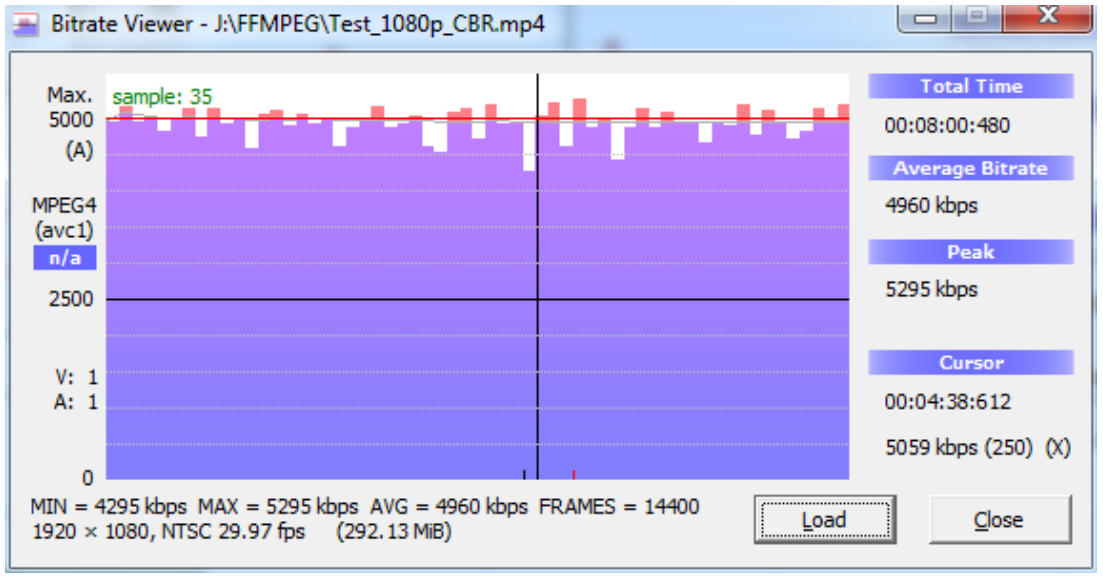
```
ffmpeg -y -i  Test_1080p.MOV -c:v libx264  -b:v 5000k -pass 1  -f mp4 NUL
&& \ (same)

ffmpeg -i  Test_1080p.MOV -c:v libx264  -b:v 5000k -maxrate 10000k -
bufsize 10000k -pass 2  Test_1080p_200p_CVBR.mp4

ffmpeg -i  Test_1080p.MOV -c:v libx264  -b:v 5000k -maxrate 5500k -
bufsize 5000k -pass 2  Test_1080p_110p_CVBR.mp4
```

- Line 2: 200% Constrained VBR
- - maxrate 10000k - 200% of target
- - bufsize 10000k - VBV buffer set to two seconds of video (more variability)

- Line 2: 110% Constrained VBR
- - maxrate 5500k - 110% of target
- - bufsize 5000k - VBV buffer set to one second of video (less variability)

Setting Data Rate-Constrained VBR vs. CBR

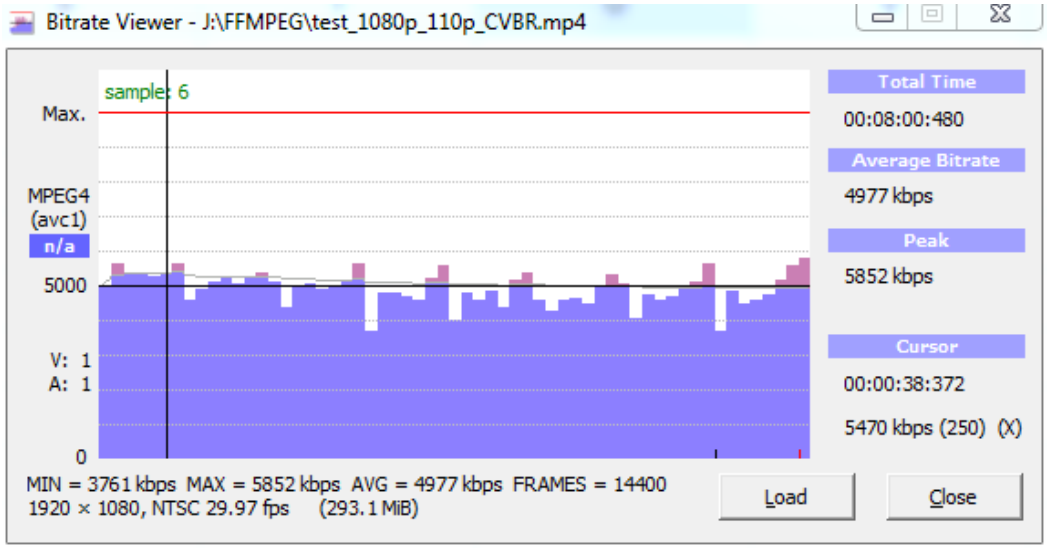


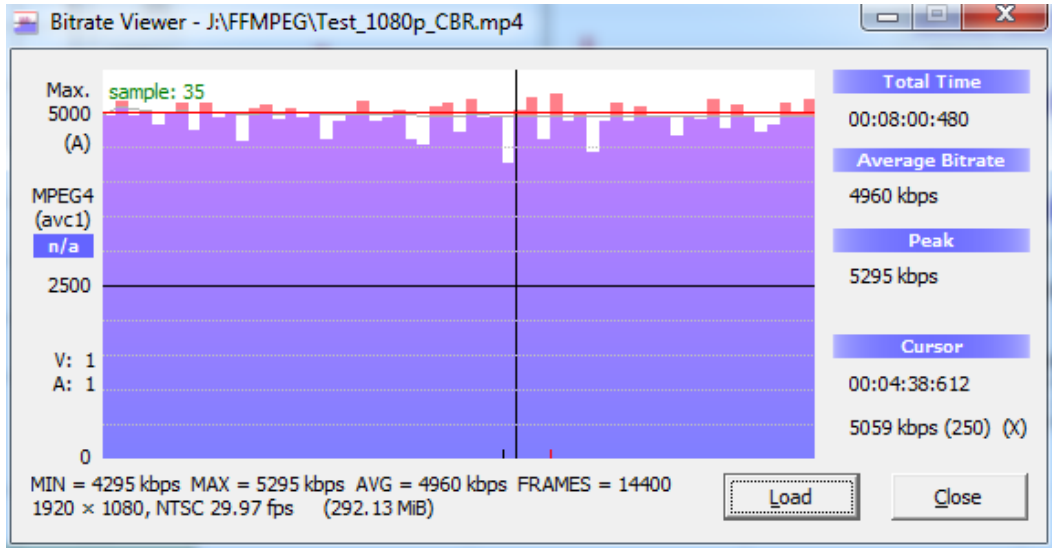


- 200% Constrained VBR
  - Slightly higher quality
  - Avoids transient quality issues
- Too much variability

- CBR
  - Peak is 5295
  - Much less variability
  - Lower overall quality (not much)
  - Can show transient quality issues

Setting Data Rate-Constrained VBR

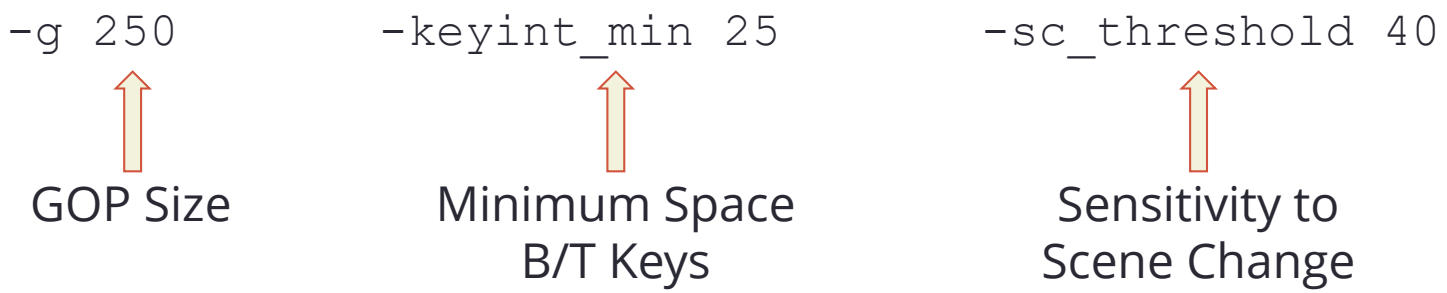




- 110 Constrained VBR
  - Slightly higher quality than CBR
  - Slightly higher peak (5852)
  - Avoids transient frame issues
  - More easily deliverable than 200% constrained

- CBR
  - Peak is 5295
  - Much less variability
  - Lower overall quality (not much)
  - Can show transient quality issues

Keyframe/Scene Change - Single File



- Default is:
  - Interval of 250
  - Scene change enabled
  - Minimum interval between 25
  - Sensitivity of 40

- Don’t have to do add anything; FFmpeg will deliver these defaults with or without entries

Key Frame/Scene Change - Single File

`-g 250`

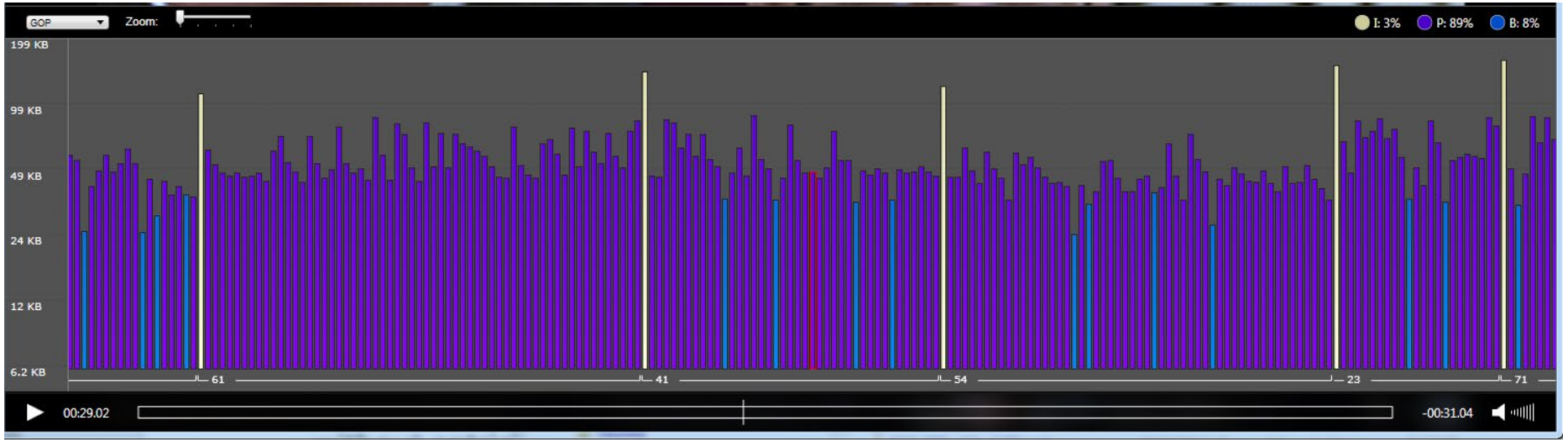
GOP Size

`-keyint_min 25`

Minimum Space B/T Keys

`-sc_threshold 40`

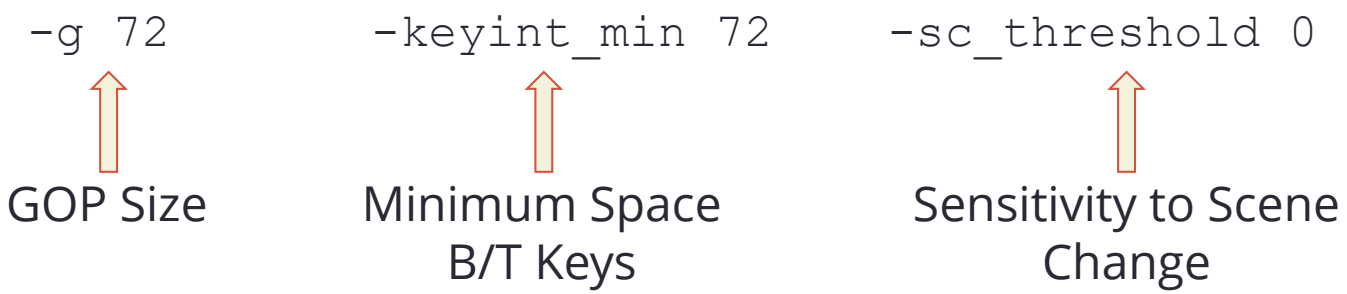
Sensitivity to Scene Change



Irregular Keyframes

Images from Telestream Switch

Key Frame/Scene Change - ABR - Alt 1



- ABR
  - Need smaller GOP so can switch to different streams much faster
  - Need consistent keyframe interval
    - Have to be at the start of all segments

- GOP 72 (3 seconds)
  - 72 is about the longest; many use 2-seconds
  - Adjust for frame rate
- Minimum 72 e.g. no scene changes
- -sc_threshold 0 - no scene changes
- Need in Pass 1 and Pass 2

Key Frame/Scene Change - ABR - Alt 1

`-g 72`

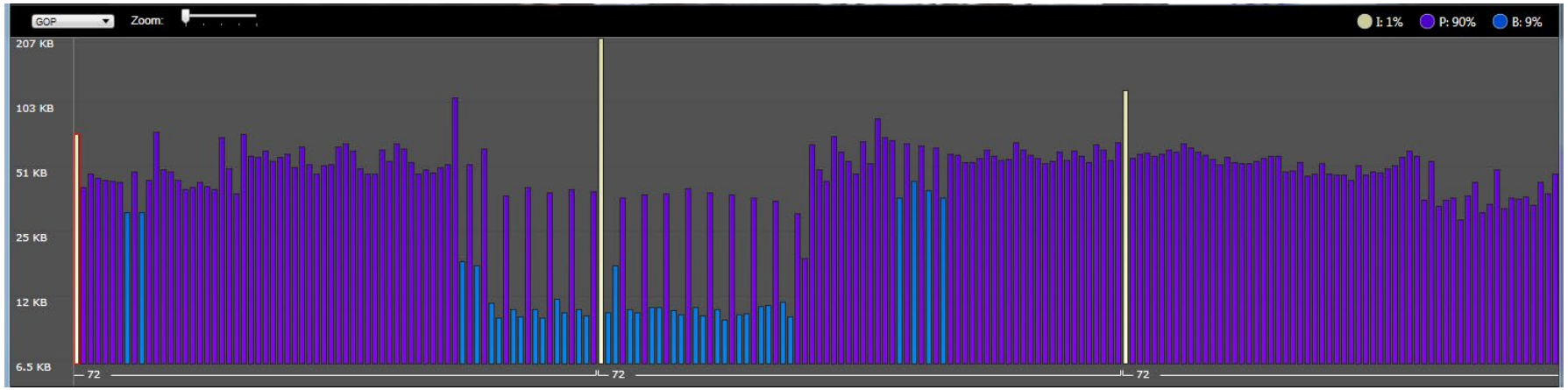
GOP Size

`-keyint_min 72`

Minimum Space B/T Keys

`-sc_threshold 0`

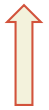
Sensitivity to Scene Change



Regular Keyframes but none at scene changes

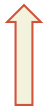
Key Frame/Scene Change - ABR - Alt 2

`-force_key_frames expr:gte(t,n_forced*3)`



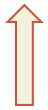
Force Keyframe every 3 seconds

`-keyint_min 25`



Default Minimum

`-sc_threshold 40`



Default Sensitivity

- Should deliver
  - Keyframe every 72 frames

- Green are defaults
  - Don’t really need to be there

Key Frame/Scene Change - ABR - Alt 2

`-force_key_frames expr:gte(t,n_forced*3)`

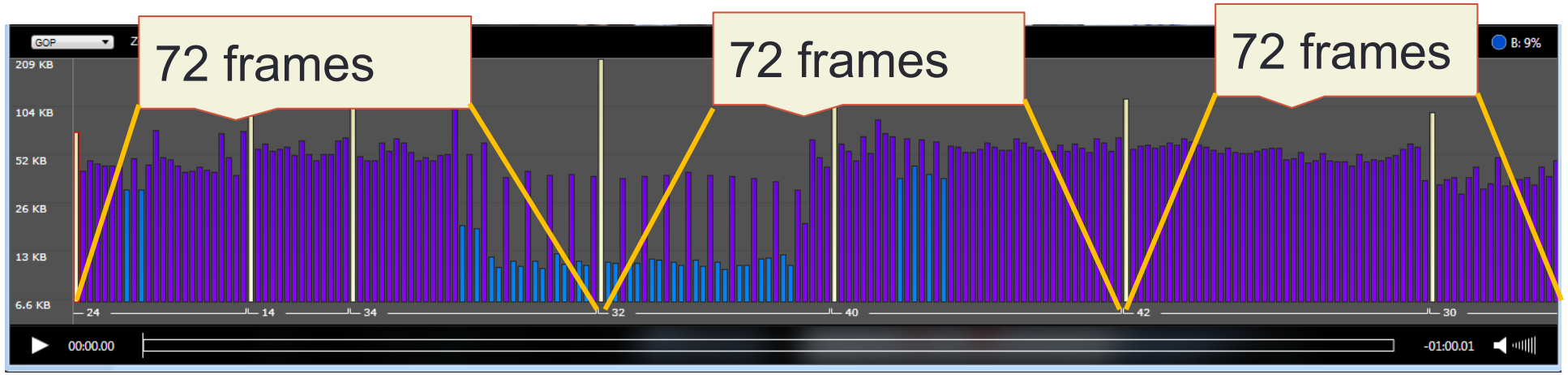
Force Keyframe every 3 seconds

`-keyint_min 25`

Default Minimum

`-sc_threshold 40`

Default Sensitivity



Regular Keyframes, and keyframes at scene changes

Which Alternative is Better?

- Static (no scene change)
- PSNR - 41.22207

- Scene Change Detection
- PSNR - 41.25565
- .08% better

Resolution

```
-s 1280x720
```

Resolution

```
-vf  scale=1280:trunc(ow/a/2)*2
```

Video Filtergraph

Set width

Compute height
Same aspect ratio
Multiple of 2

Simple

- Default resolution is same as original; if not changing resolution can leave out
- Set size directly
- Simple and easy
- May letterbox or crop if aspect ratio changes

More Complex

- More flexible approach
- Preserves aspect ratio
- Makes sure height is multiple of 2 (mod 2)
  - If odd value can cause encoding problems

Frame Rate

```
-r 12
```

- Don’t need to include
  - Default is use source frame rate
  - Typically used to cut frame rate on lower quality streams
    - 480x270@12 fps

x264 Preset/Tuning

```
-preset preset name (slow)
      - preset slow
```

- x264 has collections of encoding parameters called presets
  - Ultrafast to placebo
  - Trade encoding speed against quality (see next page)
- Default is medium - if no entry, medium parameters are applied

```
-tune tune name (animation)
      - tune animation
```

- Tune encoding parameters for different footage types
  - Animation, film, still images, PSNR, SSIM, grain
- My experience - animation works pretty well, the rest not so much
- Default is no tuning

Audio

- Default:
  - AAC for MP4
  - Channels: source
  - Sample rate: source
  - Data rate: inconsistent

- HE, HE2 are different codecs
- Channels
  - 1 = mono
  - 2 - stereo

Multipass Encoding ABR Streams

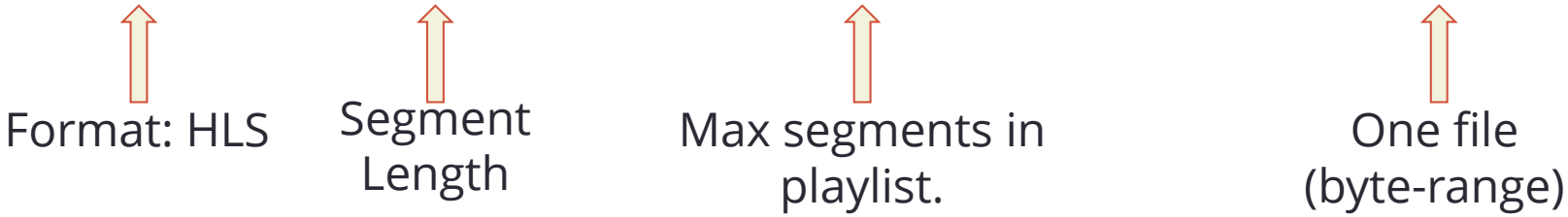
```
Pass 1: ffmpeg -y -i TOS_1080p.mov -c:v libx264 -s 1280x720 -preset medium -g 48 -keyint_min
48 -sc_threshold 0 -bf 3 -b_strategy 2 -refs 5 -b:v 3100k -c:a aac -b:a 128k -ac 2 -ar 48000 -
pass 1 -f mp4 NUL && \

Pass 2: ffmpeg -i TOS_1080p.mov -c:v libx264 -preset medium -g 48 -keyint_min 48 -sc_threshold
0 -bf 3 -b_strategy 2 -refs 5 -b:v 5200k -maxrate 5720k -bufsize 5200k -c:a aac -b:a 128k -ac
2 -ar 48000 -pass 2 TOS_1080p.mp4

Pass 2: ffmpeg -i TOS_1080p.mov. -c:v libx264 -s 1280x720 -preset medium -g 48 -keyint_min 48
-sc_threshold 0 -bf 3 -b_strategy 2 -refs 5 -b:v 3100k -maxrate 3410k -bufsize 3100k -c:a aac
-b:a 128k -ac 2 -ar 48000 -pass 2 TOS_720p_h.mp4

Pass 2: ffmpeg -i TOS_1080p.mov -c:v libx264 -s 1280x720 -preset medium -g 48 -keyint_min 48 -
sc_threshold 0 -bf 3 -b_strategy 2 -refs 5 -b:v 2400k -maxrate 2640k -bufsize 2400k -c:a aac -
b:a 128k -ac 2 -ar 48000 -pass 2 TOS_720p_l.mp4
```

- Can run first pass once, and apply to multiple encodes;
- Can reuse for different rez and bitrates
- Can’t reuse if change:
  - frame rate
  - Keyframe interval
  - Profile

- Which config options must be in first pass?
  - Frame settings (B-frame/Key frame)
  - Target data rate
  - Some say audio settings
    - My tests haven’t shown this is true

HLS Packaging

```
-f hls  -hls_time 6  -hls_list_size 0  -hls_flags single_file
```



- Format: Must be in first and second pass
- Segment length
  - Keyframe interval must divide evenly into segment size
  - Shorter improves responsiveness
- -HLS_list_size
  - Typically set to 0 which means all
- HLS_Flags
  - When single_file, one TS file with byte-range requests
  - When left out, individual .ts segments
- Creates individual .m3u8 files; you have to create master

HLS Command Line for First Three Files

Pass 1: ffmpeg -y -i Test_1080p.mov -c:v libx264 -s 1280x720 -g 48 -keyint_min 48 -sc_threshold 0 -b:v 3000k -c:a aac -b:a 128k -ac 2 -ar 48000 -pass 1 -f HLS -hls_time 6 -hls_list_size 0 -hls_flags single_file NUL && \

Pass 2: ffmpeg -i Test_1080p.mov -c:v libx264 -g 48 -keyint_min 48 -sc_threshold 0 -b:v 7800k -maxrate 8600k -bufsize 7800k -c:a aac -b:a 128k -ac 2 -ar 48000 -pass 2 -f hls -hls_time 6 -hls_list_size 0 -hls_flags single_file Test_1080p.m3u8

Pass 2: ffmpeg -i Test_1080p.mov -c:v libx264 -s 1280x720 -g 48 -keyint_min 48 -sc_threshold 0 -b:v 6000k -maxrate 6500k -bufsize 6000k -c:a aac -b:a 128k -ac 2 -ar 48000 -pass 2 -f hls -hls_time 6 -hls_list_size 0 -hls_flags single_file Test_720p_H.m3u8

Pass 2: ffmpeg -i Test_1080p.mov -c:v libx264 -s 1280x720 -g 48 -keyint_min 48 -sc_threshold 0 2   -b:v 4500k -maxrate 5000k -bufsize 4500k -c:a aac -b:a 128k -ac 2 -ar 48000 -pass 2 -f hls -hls_time 6 -hls_list_size 0 -hls_flags single_file Test_720p_M.m3u8

HEVC/x.265

```
ffmpeg -y -i TOS_1080p.mov -c:v libx265 -preset slow -x265-params profile=main:keyint=48:min-
keyint=48:scenecut=0:ref=5:bframes=3:b-adapt=2:bitrate=4000:vbv-maxrate=4400:vbv-bufsize=4000:pass=1 -an
-f mp4 NUL && \

ffmpeg -i TOS_1080p.mov -c:v libx265 -preset slow  -x265-params profile=main:keyint=48:min-
keyint=48:scenecut=0:ref=5:bframes=3:b-adapt=2:bitrate=4000:vbv-maxrate=4400:vbv-bufsize=4000:pass=2 -an
TOS_1080p_h.mp4

ffmpeg -i TOS_1080p.mov -c:v libx265 -s 1280x720 -preset slow  -x265-params profile=main:keyint=48:min-
keyint=48:scenecut=0:ref=5:bframes=3:b-adapt=2:bitrate=1000:vbv-maxrate=1100:vbv-bufsize=1000:pass=1 -an
TOS_720p_1.mp4
```

- Integrate x265 commands into FFmpeg

-x265-params – start of x265 commands, in x265 syntax

- http://x265.readthedocs.io/en/default/
- One string of commands, separated by colon, no spaces until finished

- Preset, an (audio no), format, and Null outside of this structure
- Scaling commands outside of –x265-params structure

Download free chapter - http://bit.ly/jo_freechap

Intro to Bento4

- Set of encoding/packaging utilities
- We will use three of these

What can I do with Bento4? https://www.bento4.com/

- HLS generation, including master manifests, stream level manifests, mpeg-2 ts files, and fMP4 (fragmented MP4)
- MP4 to fMP4 conversion
- DASH generation
- Parsing and multiplexing of H.264 and AAC streams
- Support for DRM (Marlin, PlayReady, Widevine and FairPlay).
- Support for H.264, H.265, AAC, AC3, eAC3, DTS, ALAC, and other codec types.
- Dual generation of HLS and DASH from fragmented MP4
- Atom/box editing, and stream/codec information
- A lot more… https://www.bento4.com/

HLS options

- Master playlists
- Single file output with byte range requests
- I-Frame only playlists
- AES encryption
- DRM
- Audio stream sidecar
- Subtitle sidecar
- fMP4

Package to .TS HLS

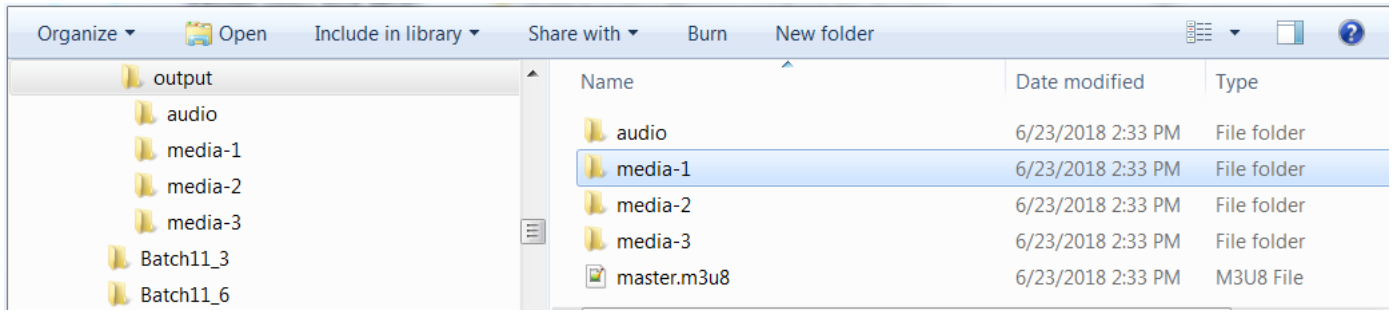
```
mp4hls  --hls-version 4     input_7000kb.mp4 input_5000kb.mp4 input_3500kb.mp4
```

Call Bento4 Program

Specify HLS Version

List files in ABR group

- Outputs:
  - Master.m3u8
  - Stream.m3u8 for each bitrate
  - Iframe.m3u8 for each bitrate
  - ts fragments for each bitrate



WebVTT Subtitles

```
mp4hls video.mp4 [+format=webvtt,+language=en]english.vtt
```

- Outputs
  - Master.m3u8
  - Stream.m3u8
  - Webvtt manifest and .vtt file

Encryption

```
mp4hls --hls-version 4 --encryption-mode AES-128 --encryption-key
abaa09cd8c75abba54ac12dbcc65acd7 --encryption-url http://getmyKey?token=token
video.mp4
```

- Outputs
  - All HLS assets
  - Assets are encrypted with AES-128, and encryption URL is added to the stream manifests
  - Segment duration will be set to 6 seconds, but will only segment at the closest i-frame

Encryption Help File

--encryption-mode=<mode> - Encryption mode (only used when --encryption-key is specified). AES-128 or SAMPLE-AES (default: AES-128)

--encryption-key=<key> - Encryption key in hexadecimal (default: no encryption)

--encryption-iv-mode=<mode> - Encryption IV mode: 'sequence', 'random' or 'fps' (Fairplay Streaming) (default: sequence). When the mode is 'fps', the encryption key must be 32 bytes: 16 bytes for the key followed by 16 bytes for the IV.

--encryption-key-uri=<uri> - Encryption key URI (may be a realtive or absolute URI). (default: key.bin)

--encryption-key-format=<format> - Encryption key format. (default: 'identity')

--encryption-key-format-versions=<versions> - Encryption key format versions.

--output-encryption-key - Output the encryption key to a file (default: don't output the key). This option is only valid when the encryption key format is 'identity'

https://www.bento4.com/documentation/mp4hls/

Dual HLS and DASH from fMP4

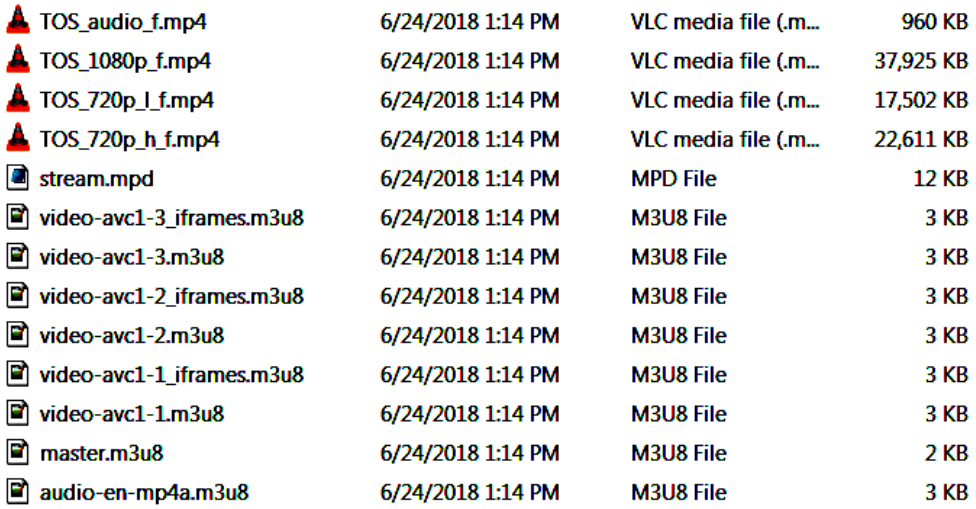
```
mp4fragment input.mp4 output.mp4 (converts mp4 to fmp4)

mp4dash --hls --no-split --use-segment-list output.mp4
```

- Run each separately
  - Mp4fragment – to convert each input file into fragmented MP4 file
  - Mp4dash
    - `--hls` – forces HLS manifest
    - `--no-split` – output single file
    - `--use-segment-list` needed for `--no-split`
    - List all files
- Outputs
  - Master.m3u8
  - Audio.m3u8
  - Video.m3u8
  - Stream.mpd (DASH manifest)

Dual HLS and DASH from fMP4

```
mp4fragment   TOS_720p_l.mp4  TOS_720p_l_f.mp4
mp4fragment   TOS_720p_h.mp4  TOS_720p_h_f.mp4
mp4fragment   TOS_1080p.mp4   TOS_1080p_f.mp4
mp4fragment   TOS_audio.mp4   TOS_audio_f.mp4
```

```
mp4dash --hls --no-split --use-segment-list
TOS_720p_l_f.mp4 TOS_720p_h_f.mp4 TOS_1080p_f.mp4
TOS_audio_f.mp4
```

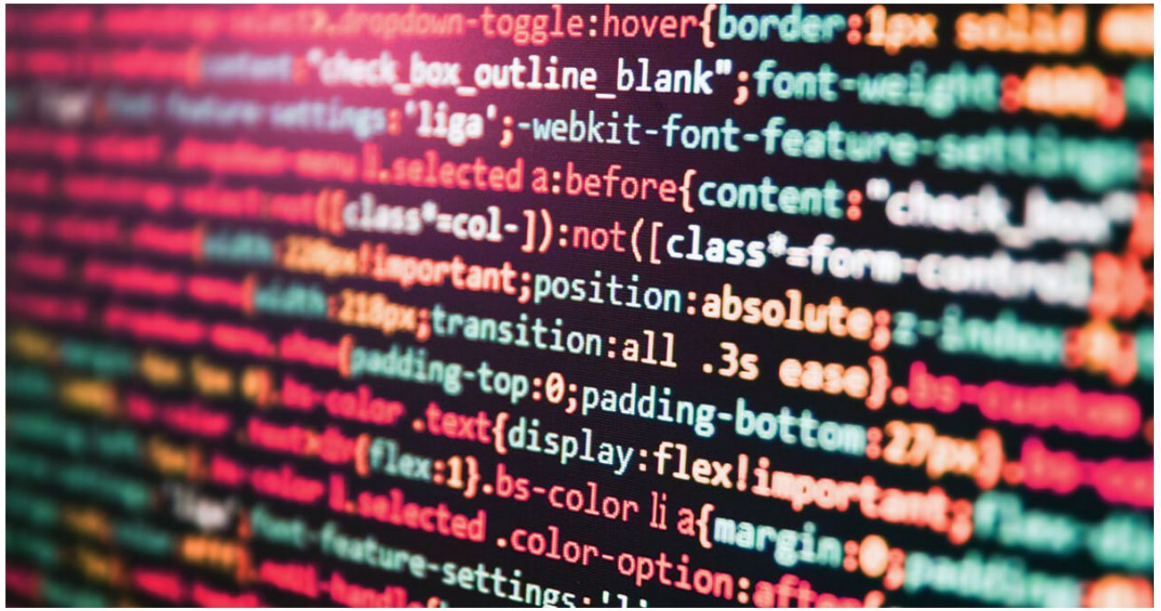


Automating on Ubuntu

- Why Ubuntu?
  - It's free
  - It has simpler automation controls than Windows
  - FFmpeg runs on Ubuntu
- Where to learn more
  - All operations defined in article on the right

February 20, 2019
By Jan Ozer Contributing Editor
Featured Articles

How to Automate FFmpeg and Bento4 With Bash Scripts



bit.ly/Bash_ozer

From Batch to Bash

```
ffmpeg -i TOS_1080p.MOV -c:v libx264 TOS_s1.mp4
```

- Save as batch.bat
- Run by double clicking or typing on command line

```
#!/bin/bash

ffmpeg -i TOS_1080p.MOV -c:v libx264 TOS_s1.mp4
```

- Add Shebang
- Save as batch.**sh**
- Get permission to run by typing `chmod +x batch.sh`
- Run by typing from terminal `./batch.sh`

Encoding Entire Directory with “For” Loop

```
1  #!/bin/bash
2  for file in *.mp4
3  do
4  ffmpeg -i "$file" -c:v libx264 -crf 23 "${file%.*}_crf".mp4
5  done
```

1. Shebang
2. `for` - define files to encode (*.mp4)
3. `Do` (tell Ubuntu what to do)
4. FFmpeg command string
5. done

- Save as `crf.sh` in folder to encode
- Open terminal in that folder
- Get permission (`chmod +x batch.sh`)
- Run batch file (`./batch.sh`)

Download all Bash scripts here: http://bit.ly/bash_scripts

Encoding/Packaging

ffmpeg_d.sh

```
1  #! /bin/bash
2  for file in *.mp4;
3  do
4  mkdir "${file%.*}"
5  ffmpeg -y -i "$file" -c:v libx264 -b:v 5200k -g 60 -pass 1 -f mp4 /dev/null && \
6  ffmpeg -i "$file" -c:v libx264 -b:v 4500k -g 60 -pass 2 "${file%.*}/${file%.*}_1080p".mp4
7  ffmpeg -i "$file" -s 1280x720 -c:v libx264 -b:v 3000k -g 60 -pass 2 "${file%.*}/${file%.*}_720p".mp4
8  ffmpeg -i "$file" -s 640x360 -c:v libx264 -b:v 1400k -g 60 -pass 2 "${file%.*}/${file%.*}_360p".mp4
9  ffmpeg -i "$file" -vn -c:a aac -b:a 128k -ac 2 -r 48k -pass 2 "${file%.*}/${file%.*}_audio".mp4
10 mp4fragment "${file%.*}/${file%.*}_1080p".mp4 "${file%.*}/${file%.*}_1080p_frag".mp4
11 mp4fragment "${file%.*}/${file%.*}_720p".mp4 "${file%.*}/${file%.*}_720p_frag".mp4
12 mp4fragment "${file%.*}/${file%.*}_360p".mp4 "${file%.*}/${file%.*}_360p_frag".mp4
13 mp4fragment "${file%.*}/${file%.*}_audio".mp4 "${file%.*}/${file%.*}_audio_frag".mp4
14 mp4-dash --hls --output-dir="${file%.*}/${file%.*}_output" --use-segment-list  "${file%.*}/${file%.*}_720p_frag".mp4
   "${file%.*}/${file%.*}_360p_frag".mp4 "${file%.*}/${file%.*}_1080p_frag".mp4 "${file%.*}/${file%.*}_audio_frag".mp4
15 mv "$file" "${file%.*}"
16 done
```

- What `ffmpeg_d.sh` is doing:
  - Line 5 - Creating a folder to store output in (otherwise, script will recursively encode new files)
    - Will name the folder the file name without the extension
  - Lines 6-10 - Encoding single input file to four outputs using two-pass encoding (lines 6-10)
    - Copying file into the newly created folder
    - Naming it the name of the file without the extension and with the _1080p/720p/360p designator (so, input_1080p.mp4).
  - Lines 11-14 - Converting each input to fMP4
  - Line 15 - Creating DASH/HLS output with MP4dash ubuntu version – storing the output into the newly created folder (`--output-dir`)
  - Line 16 - Moves source file into folder we created in line 5
  - To execute, store in folder with content, open Terminal
    - `chmod +x ffmpeg_d.sh`
    - `.\ffmpeg_d.sh`

Setting Up the Watch Folder

watch.sh

```
1  #! /bin/bash
2  while inotifywait /home/jan/Watch
3  do
4  /home/jan/Watch/ffmpeg_d.sh
5  done
```

- What we're doing:
  - Line 2 – using `inotifywait` function to watch folder
  - Line 4 - running the script if changes are detected

- execute, store in folder with content, open Terminal
  - `chmod +x watch.sh`
  - `.\watch.sh`
- Stop watching with Ctrl-Break

Questions?