RAWcooked Automation Cheat Sheet (MacOS/Linux)

RAWcooked is open source software from MediaArea.net. It is designed to encode raw DPX, TIFF or EXR audio-visual sequences into a lossless video stream using FFV1 codec in a Matroska wrapper. RAWcooked uses open source FFmpeg to encode the image sequences to FFV1 while embedding binary reversibility data in the Matroska wrapper. There are significant storage reductions after encoding, and the video file can be viewed easily in VLC and other player software.

```
rawcooked --all "image_sequence_folder" (or) "file.mkv"
```

This basic command will let you transcode an image sequence to a Matroska file, or return a RAWcooked FFV1 Matroska video back to an image sequence. The Matroska contains a bit perfect copy of the original sequence with embedded hashes.

Optimizing for automation

To optimize your workstation for efficient RAWcooked encoding it is recommended you have good input and output communications speeds (IO) between the workstation, ample Central Processing Unit (CPU) cores and a separate fast solid-state drive (SSD) or networked accessed storage (NAS) containing your audio-visual sequences.

An average is estimated upon the following calculation:
Width x Height x Components x Bit Depth x Frame Rate
1920 x 1080 x 3 x 10 x 30 = ~1900 Mbps

For realtime processing for one pass you will require a CPU able to handle 1900 Mbps. This estimation is based on 50 Mbps per 1 GHz for each pass:
1900 Mbps ÷ 50 Mbps ÷ 3 GHz = 12 cores
IO able to handle 1900 Mbps = 250 MB/s

Random Access Memory (RAM) helps bridge any gaps that may appear with bottlenecks between IO interrupts and CPU processing. It's helpful to have ample memory available.

Parellelization

Parellelization is the act of processing jobs in parallel using resources that complete in less time by dividing up the work. Normally a computer processes jobs serially, one job after another. When optimized for parallel processing a server will complete a series of RAWcooked jobs far quicker.

Parallelization is not the same as multi-threading. This is used by FFmpeg for encoding and decoding FFV1. The FFV1 codec has slices through each frame which allow for granular checksum verification of the asset, but it also allows FFmpeg multithread as each slice block can be separated into different processing tasks. How efficient your multithreading can make your RAWcooked workstation will vary based on this slice-based multithreading and the CPU core count. Multithreading on a single CPU will give the illusion of jobs running in parallel, but it's the scheduling algorithm that switches depending on IO input interruptions and thread priorities. Multithreading on multiple CPU cores allows for true parallelization, as the more cores you have the more parallel processes you can run giving optimum encoding efficiency.

Encoding with GNU Parallel

Open source software GNU Parallel adds an extra layer of parallelization that helps maximise encoding throughput. It separates encoding jobs across CPU cores maintaining maximum CPU and IO activity at all times on the server. It's free software to download and use but requires installation, and there are lots of guides online to help: ***gnu.org/software/parallel***

Most simply you can input a list of audio-visual sequence paths and total number of jobs, then pass your RAWcooked command and multiple jobs will be processed with the same command at the same time. The examples below show the 'cat' command which lists each item in the text file, then the vertical line or 'pipe' hands each item from the list to GNU Parallel.

```
cat sequence_list.txt | parallel --jobs 10 "rawcooked -y --all {}"
```

Parallel then takes ten jobs at a time and launches the RAWcooked command, with the curly brackets representing each individual sequence path from the list and the '-y' answers yes to all encoding questions allowing RAWcooked to proceed. This command will place each encoded Matroska files alongside the path of the audio-visual sequence supplied in the text file.

Log files for automated assessment

When automating encoding you should capture to logs the console output made by both RAWcooked and FFmpeg. The information in these logs is important and can be used to script an automated check of each new Matroska, sorting successful files from failed encoded files. To capture a log add the following text to the end of your encoding command:

```
rawcooked -y --all "image_sequence" >> "image_sequence.log"
```

This captured log will contain pre-encoding RAWcooked analysis, FFmpeg encoding details and post-encoding RAWcooked analysis. Any errors or warnings about the encoding will be captured in this log along with a completion statement, such as 'Reversibility was checked, no issue detected.' The logs also contain details about the software versions used to encode the file, paths to the file destinations, and attachments within the Matroska.

Troubleshooting

`rawcooked --help`	Loads the RAWcooked help pages with information about all the flags.
`rawcooked --version`	Display the version of RAWcooked software you are using.
`man rawcooked`	Manual page for RAWcooked listing flags available and use examples.
`rawcooked -d -all "image_sequence"`	Display the FFmpeg command created by RAWcooked, but do not run.
`rawcooked --show-licence`	See which audio-visual flavours are supported by your license.
`rawcooked --store-license "value"`	Add a new licence key to unlock additional features.
`rawcooked --info "file.mkv"`	Display data about compressed FFV1 Matroska file.
`github.com/MediaArea/RAWcooked/issues/`	Media Area issue tracker where you can discuss problems experienced.

mediaarea.net/RAWcooked