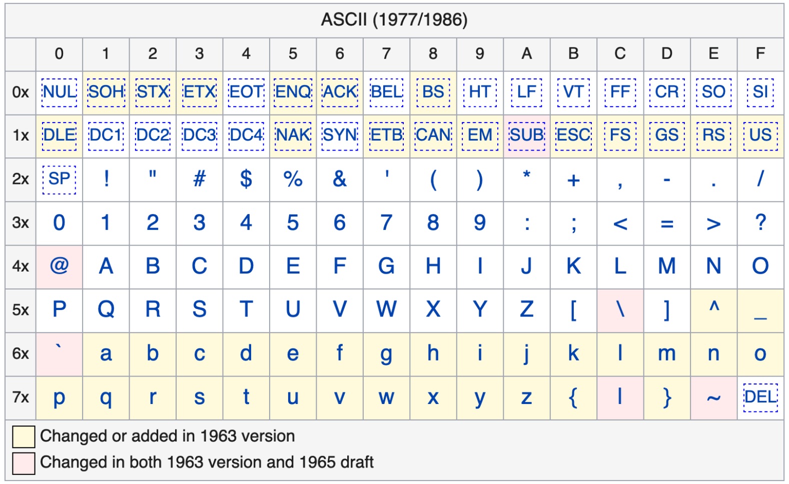
An Introduction to FFmpeg

Reto Kromer • AV Preservation by reto.ch

Open-Source Tools and Resources for Audio-Visual Archives
Elías Querejeta Zine Eskola
Donostia (San Sebastián), Spain
16–19 May 2023

Interacting with the computer

- punched cards reader and line printer
- command-line interface (CLI)
- graphical user interface (GUI)
- touchless interface



ASCII (1977/1986)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

🟨 Changed or added in 1963 version
🟥 Changed in both 1963 version and 1965 draft

Unix/Linux Command Structure

```
 $0      $1              ${n}
command argument_1 ... argument_n
```

common syntaxes of arguments include:

```
--parameter
--parameter=value
-p
-p value
```

FFmpeg Command Structure

```
 $0      $1          ${n}
command argument_1 ... argument_n
```

FFmpeg syntax of arguments:

```
-parameter
-parameter value
-p
-p value
```

FFmpeg Syntax

```
ffmpeg [global_options]
[input_options_n] -i input_file_n
[output_options_n] output_file_n

ffprobe [input_options] input_file

ffplay [input_options] input_file
```

The FFmpeg Family

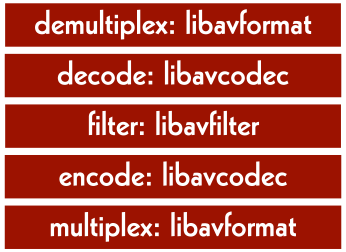
Tools

- ffmpeg
- ffprobe
- ffplay

Libraries

- libavcodec
- libavformat
- libavfilter
- libavutil
- libavdevice
- libswscale
- libswresample
- libpostproc

Data Transformations



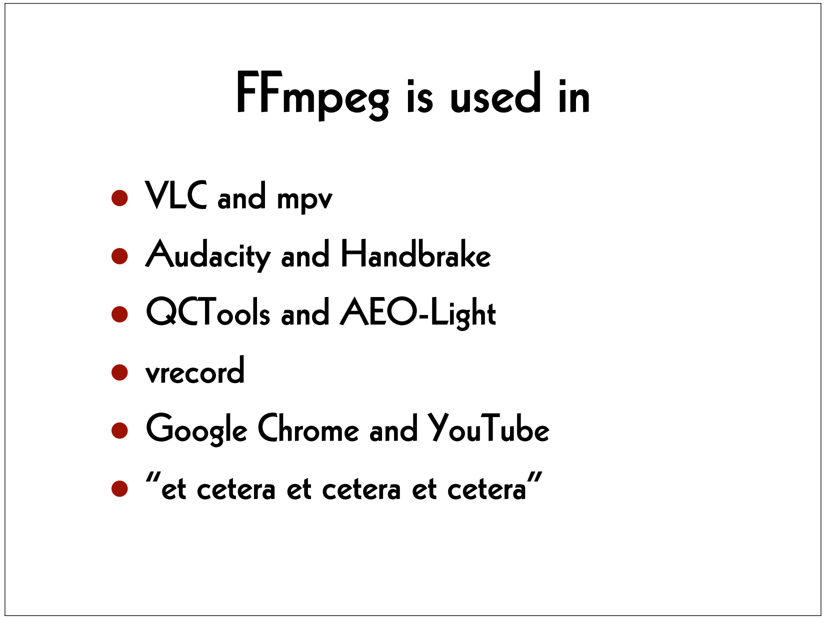
demultiplex: libavformat

decode: libavcodec

filter: libavfilter

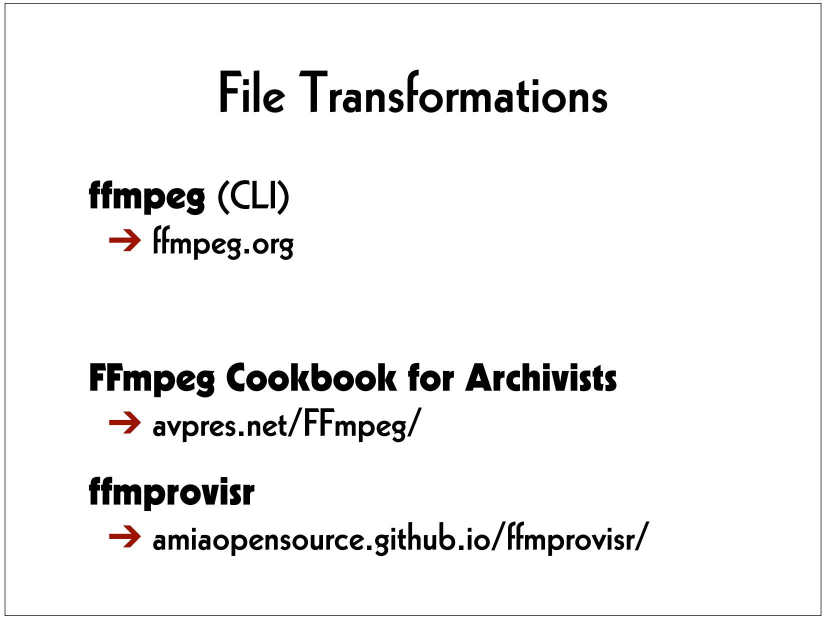
encode: libavcodec

multiplex: libavformat



FFmpeg is used in

- VLC and mpv
- Audacity and Handbrake
- QCTools and AEO-Light
- vrecord
- Google Chrome and YouTube
- “et cetera et cetera et cetera”

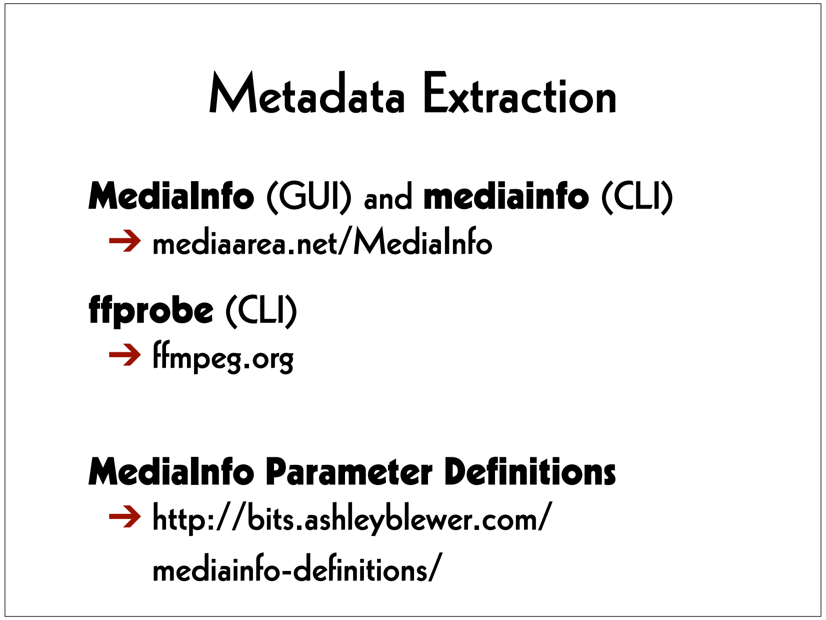


File Transformations

ffmpeg (CLI)
➔ ffmpeg.org

FFmpeg Cookbook for Archivists
➔ avpres.net/FFmpeg/

ffmprovisr
➔ amiaopensource.github.io/ffmprovisr/

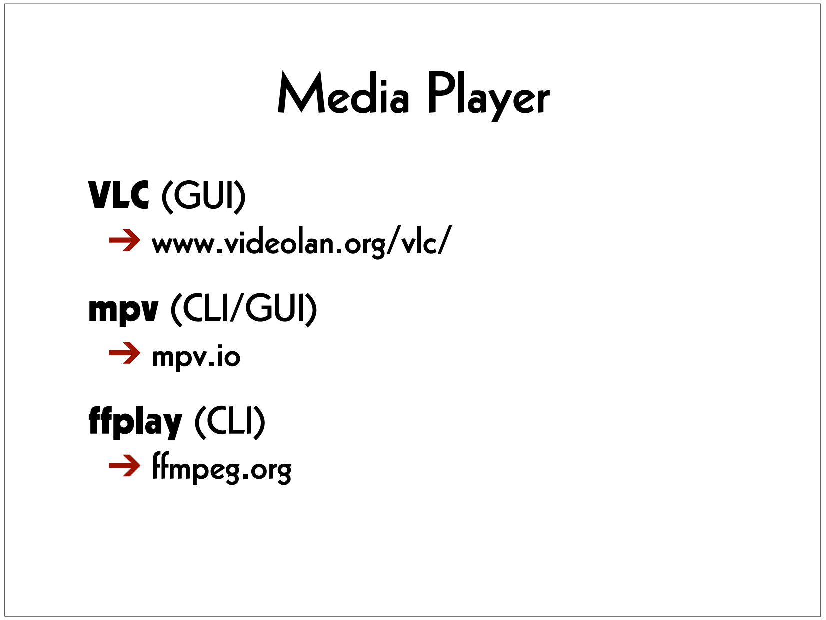


Metadata Extraction

MediaInfo (GUI) and **mediainfo** (CLI)
➔ mediaarea.net/MediaInfo

ffprobe (CLI)
➔ ffmpeg.org

MediaInfo Parameter Definitions
➔ http://bits.ashleyblewer.com/mediainfo-definitions/



Media Player

VLC (GUI)
➔ www.videolan.org/vlc/

mpv (CLI/GUI)
➔ mpv.io

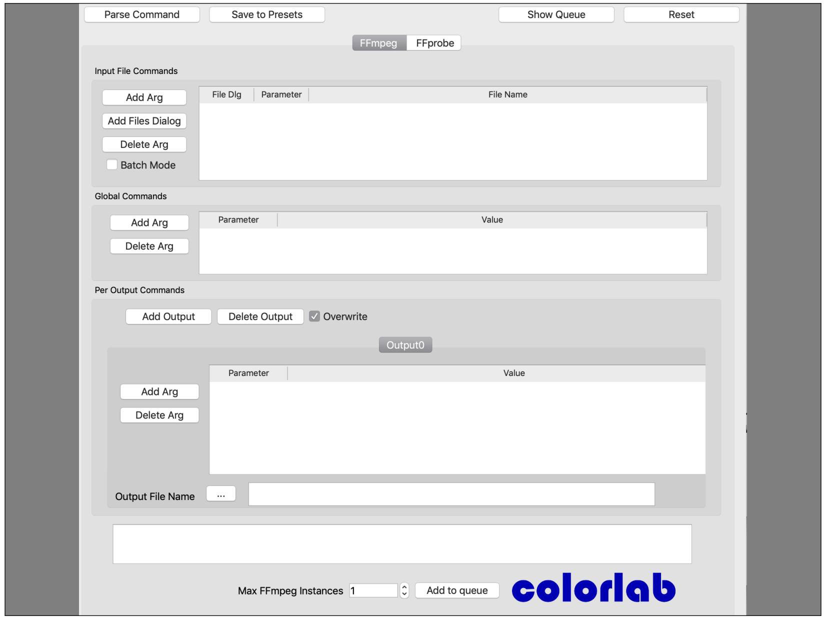
ffplay (CLI)
➔ ffmpeg.org

A GUI for FFmpeg

FFCommand Engine (GUI)
→ github.com/ColorlabMD/FFCommand_Engine

version 0.6 for macOS via Homebrew:

```
brew tap avpres/formulae
brew install --HEAD ffcommand-engine
```



☐ Batch Mode

☑ Overwrite

Max FFmpeg Instances 1

1st Part

Set the Working Space

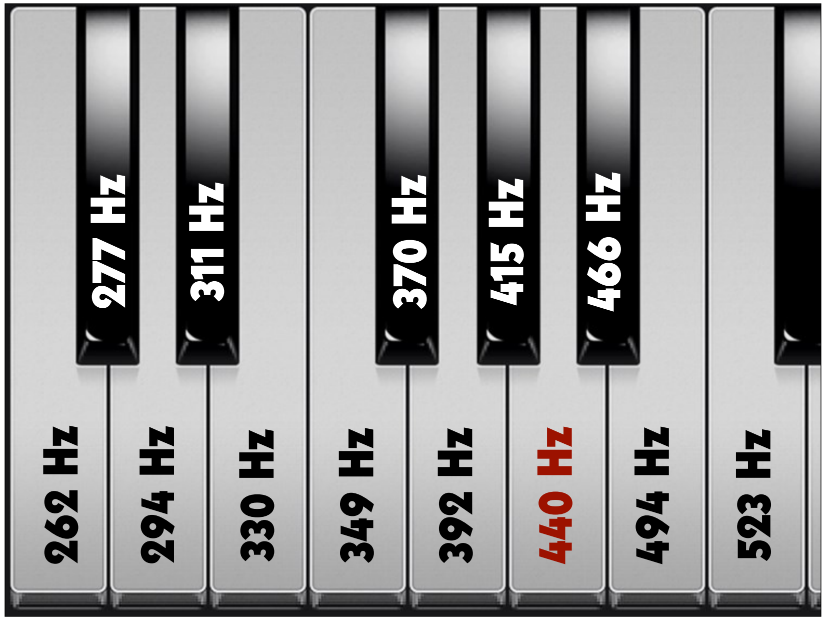
```
# Linux
# Mac
# Windows Terminal or WSL
# Windows locally:
cd Desktop

# Windows on OneDrive Cloud:
cd OneDrive
```

Generate an Image File

```
ffmpeg
  -f lavfi -i mandelbrot
  -t 10
  -c:v rawvideo
  -pix_fmt uyvy422
  mandelbrot.avi
```

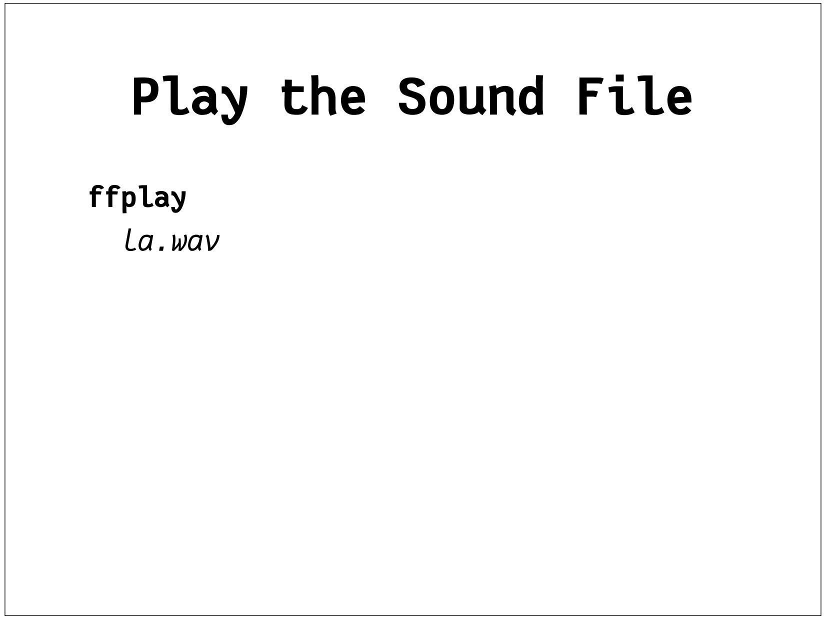
Play the Image File

```
ffplay
  mandelbrot.avi
```



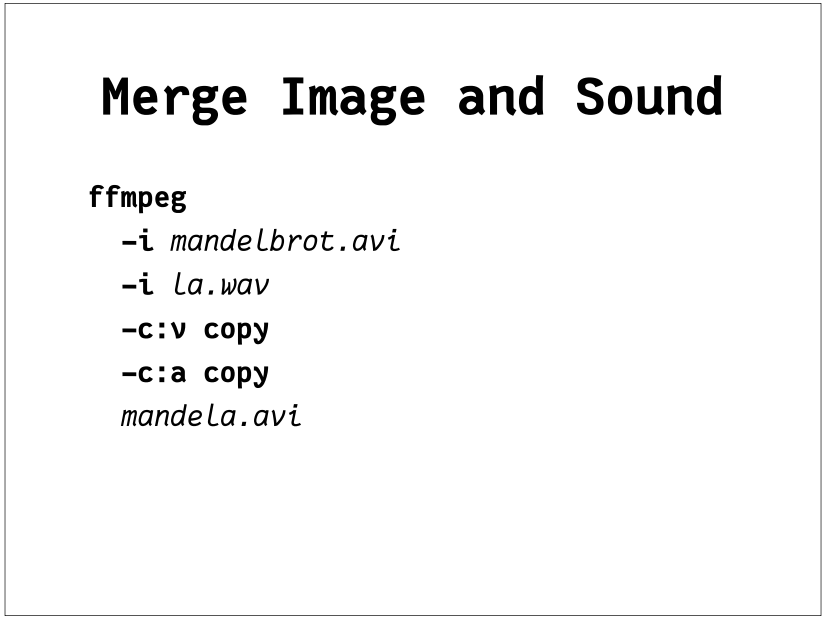
Generate a Sound File

```
ffmpeg
  -f lavfi -i "sine=frequency=440"
  -t 10
  la.wav
```



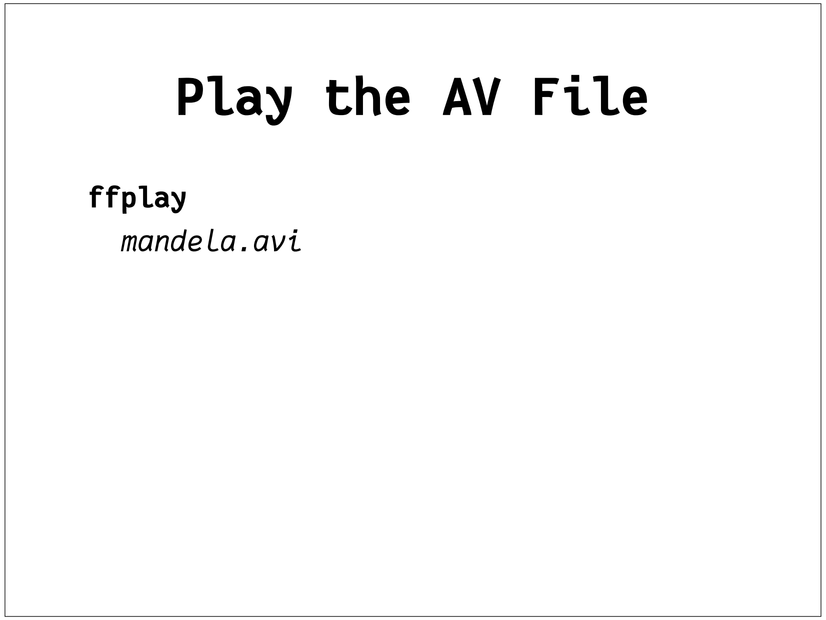
Play the Sound File

```
ffplay
  la.wav
```



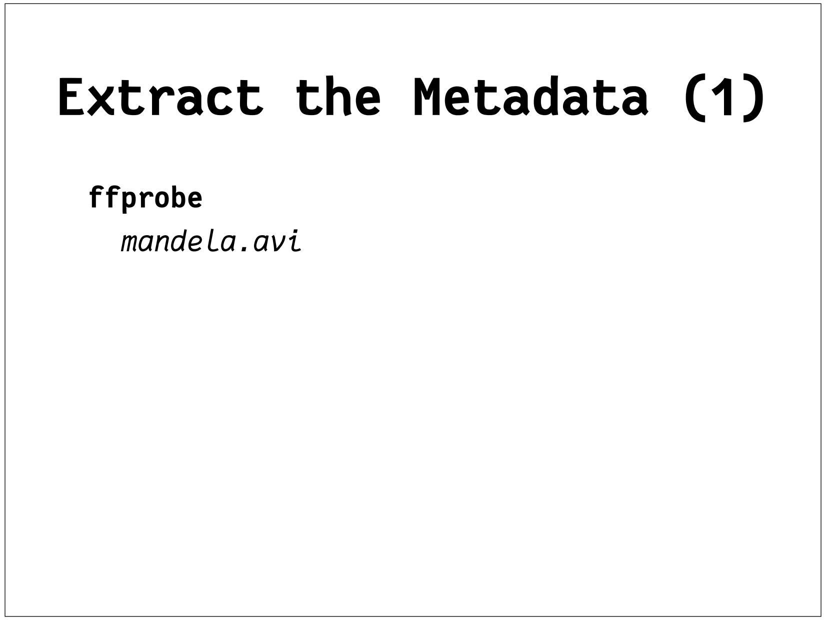
Merge Image and Sound

```
ffmpeg
  -i mandelbrot.avi
  -i la.wav
  -c:v copy
  -c:a copy
  mandela.avi
```



Play the AV File

```
ffplay
  mandela.avi
```



Extract the Metadata (1)

```
ffprobe
  mandela.avi
```

Container

```
ffprobe
  -show_format
  mandela.avi
```

Codec

```
ffprobe
  -show_streams
  mandela.avi
```

Container and Codec

```
ffprobe
  -show_format
  -show_streams
  mandela.avi
```

Format the Output

```
ffprobe
  -show_format
  -show_streams
  -print_format json
  mandela.avi
```

Save the Metadata

```
ffprobe
  -show_format
  -show_streams
  -print_format json
  mandela.avi
  > mandela.txt
```

Extract the Metadata (2)

```
mediainfo
  mandela.avi
```

Find Help

```
ffmpeg -h
ffmpeg -codecs
ffmpeg -decoders
ffmpeg -h decoder=aac
ffmpeg -encoders
ffmpeg -h encoder=libx264
ffmpeg -filters
ffmpeg -pix_fmts
```

Modify the Container

```
ffmpeg
  -i mandelbrot.avi
  -c copy
  mandelbrot.mov
```

Generate Checksums (1)

```
ffmpeg
  -i mandelbrot.avi
  -f framemd5
  mandelbrot_avi_framemd5.txt
```

Generate Checksums (2)

```
ffmpeg
  -i mandelbrot.mov
  -f framemd5
  mandelbrot_mov_framemd5.txt
```

Compare Checksum Files

```
# Linux/Mac/Windows Terminal or WSL:
diff -s
  mandelbrot_avi_framemd5.txt
  mandelbrot_mov_framemd5.txt

# Windows:
fc
  mandelbrot_avi_framemd5.txt
  mandelbrot_mov_framemd5.txt
```

2nd Part

Play Single Images

```
ffplay
  -loop 0
  DUFAY_TIFF/Dufay_%06d.tif
```

File Transformations

- from the master file to a mezzanine file and from the mezzanine file to an access file
- from the master file to an access file

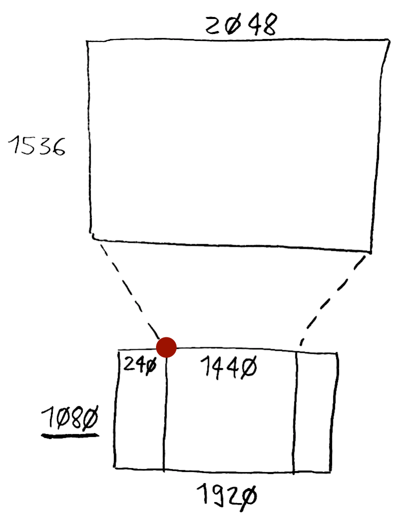
→ Compare the quality of the access files.

→ Compare the quality of the mezzanine files (Apple ProRes and AVID).

2K
4:3

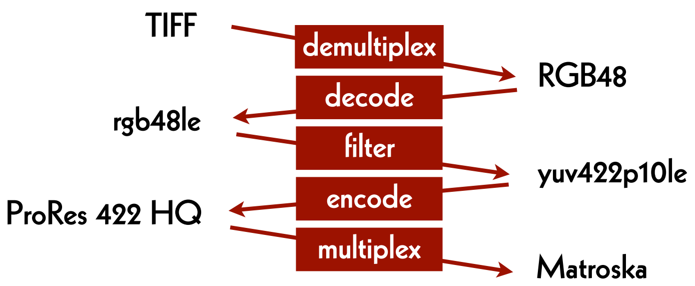
↓

HD
16:9



ProRes

From TIFF to ProRes



ProRes 422 and ProRes 4444

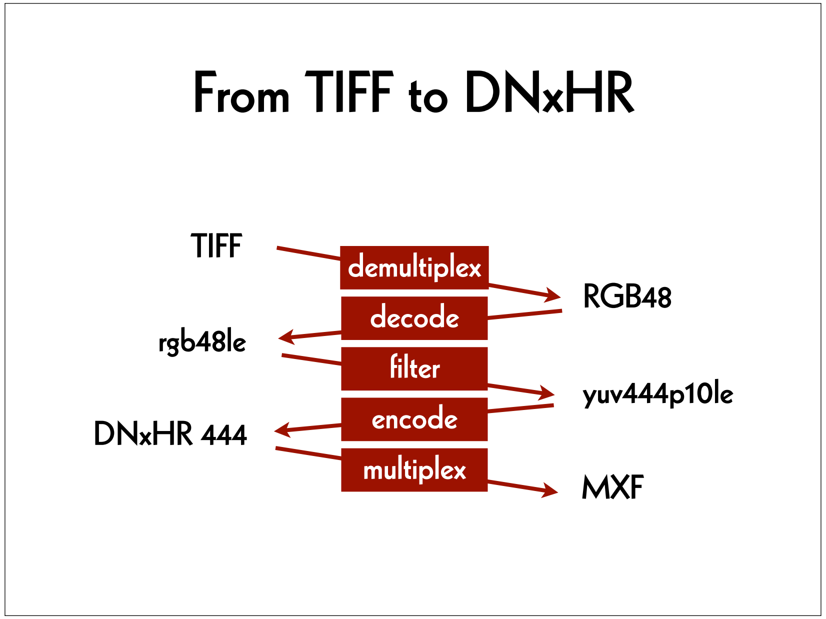
possible containers

- QuickTime (.mov)
- Matroska (.mkv)
- MXF = Material eXchange Format (.mxl)

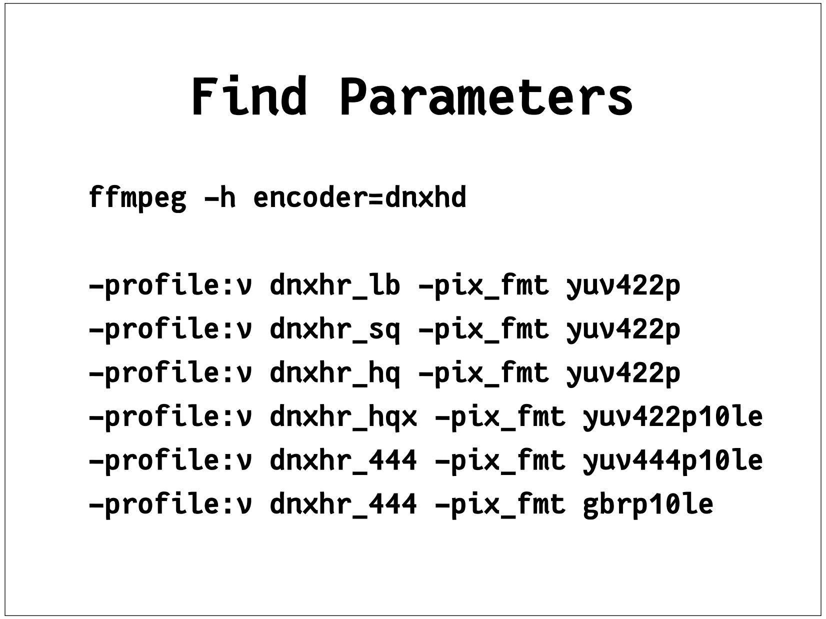
Master -> Mezzanine (1)

```
ffmpeg
  -f image2 -framerate 24
  -i DUFAY_TIFF/Dufay_%06d.tif
  -filter:v
    "scale=1440:1080:flags=lanczos,
     pad=1920:1080:240:0"
  -c:v prores_ks -profile:v 3
  Dufay_ProRes.mkv
```

AVID

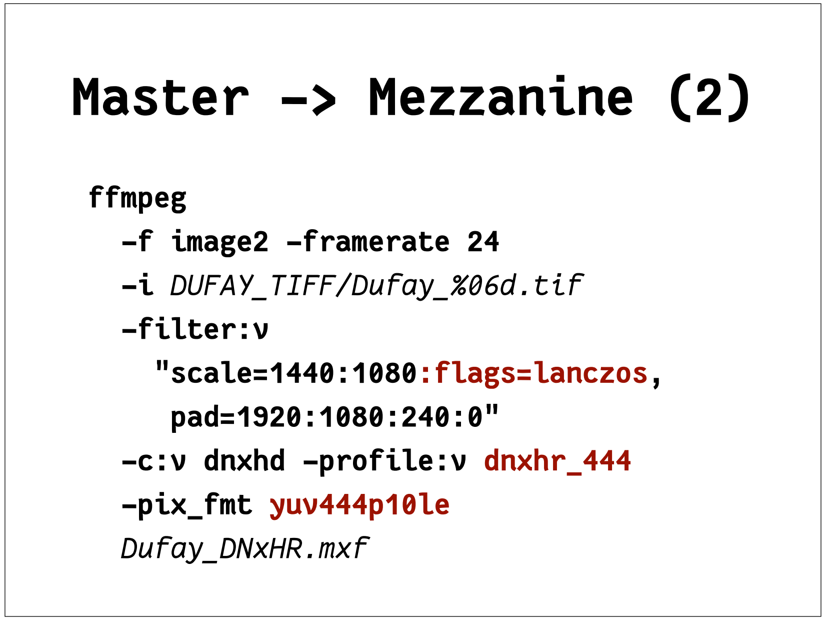


From TIFF to DNxHR



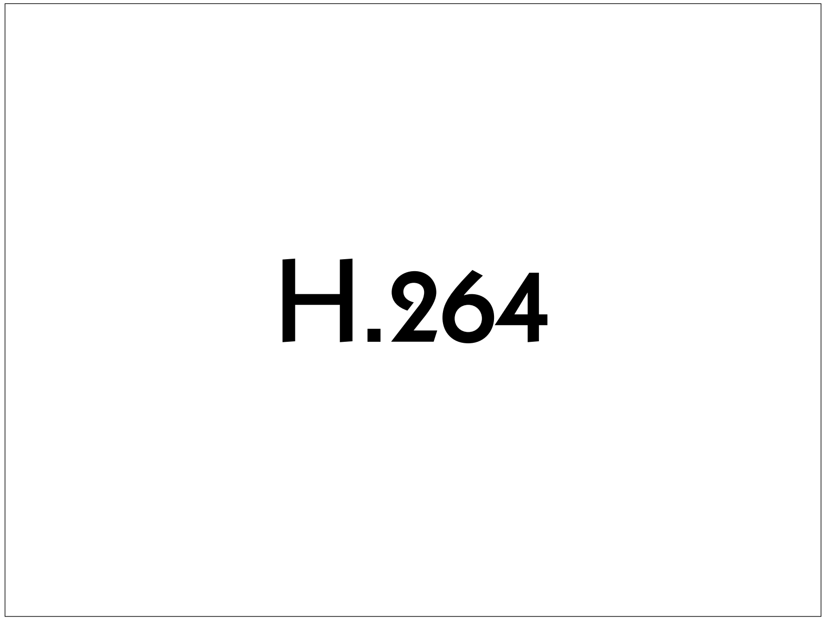
Find Parameters

```
ffmpeg -h encoder=dnxhd

-profile:v dnxhr_lb -pix_fmt yuv422p
-profile:v dnxhr_sq -pix_fmt yuv422p
-profile:v dnxhr_hq -pix_fmt yuv422p
-profile:v dnxhr_hqx -pix_fmt yuv422p10le
-profile:v dnxhr_444 -pix_fmt yuv444p10le
-profile:v dnxhr_444 -pix_fmt gbrp10le
```



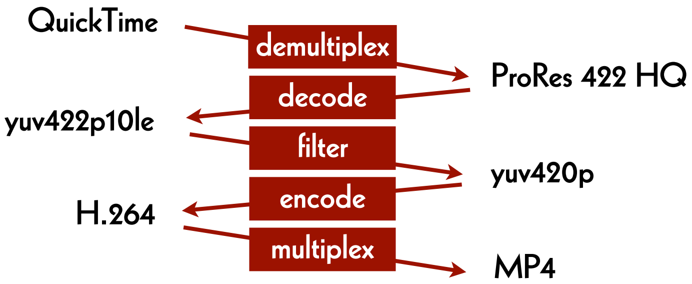
Master -> Mezzanine (2)

```
ffmpeg
  -f image2 -framerate 24
  -i DUFAY_TIFF/Dufay_%06d.tif
  -filter:v
    "scale=1440:1080:flags=lanczos,
     pad=1920:1080:240:0"
  -c:v dnxhd -profile:v dnxhr_444
  -pix_fmt yuv444p10le
  Dufay_DNxHR.mxf
```



H.264

From ProRes to H.264



QuickTime

demultiplex

ProRes 422 HQ

decode

yuv422p10le

filter

yuv420p

encode

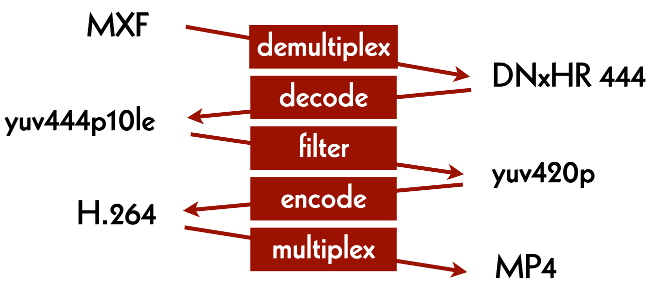
H.264

multiplex

MP4

49

From DNxHR to H.264



MXF

demultiplex

DNxHR 444

decode

yuv444p10le

filter

yuv420p

encode

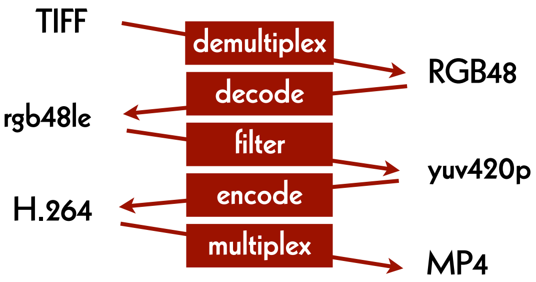
H.264

multiplex

MP4

50

From TIFF to H.264



TIFF

demultiplex

RGB48

decode

rgb48le

filter

yuv420p

encode

H.264

multiplex

MP4

51

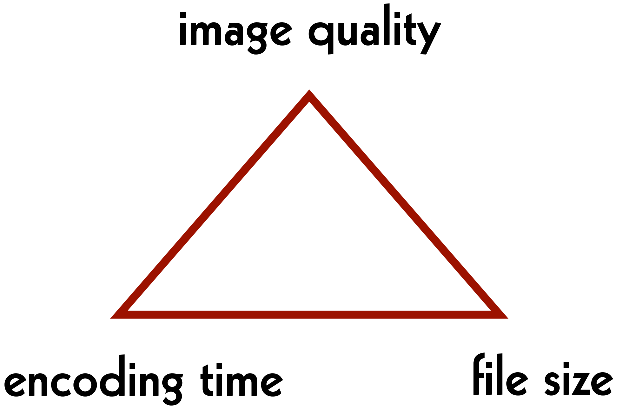
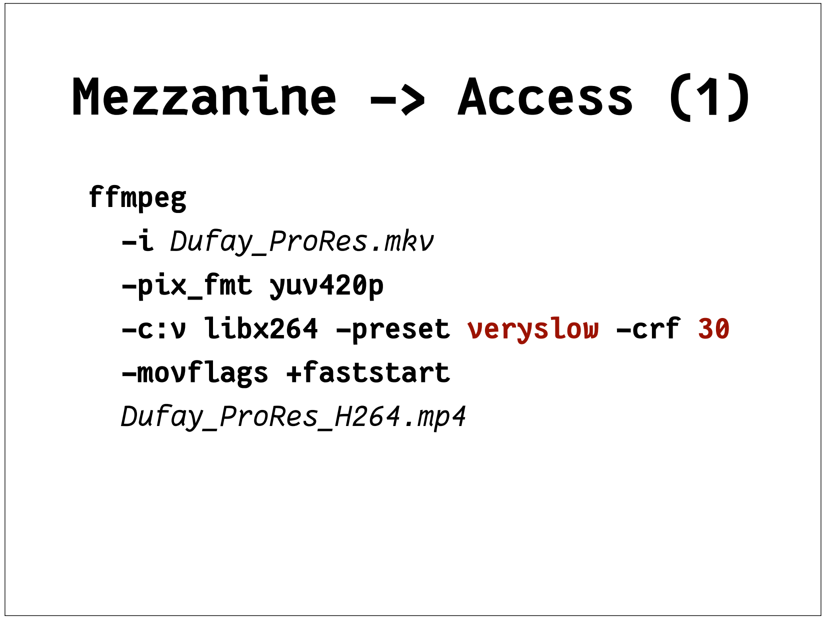


image quality

encoding time

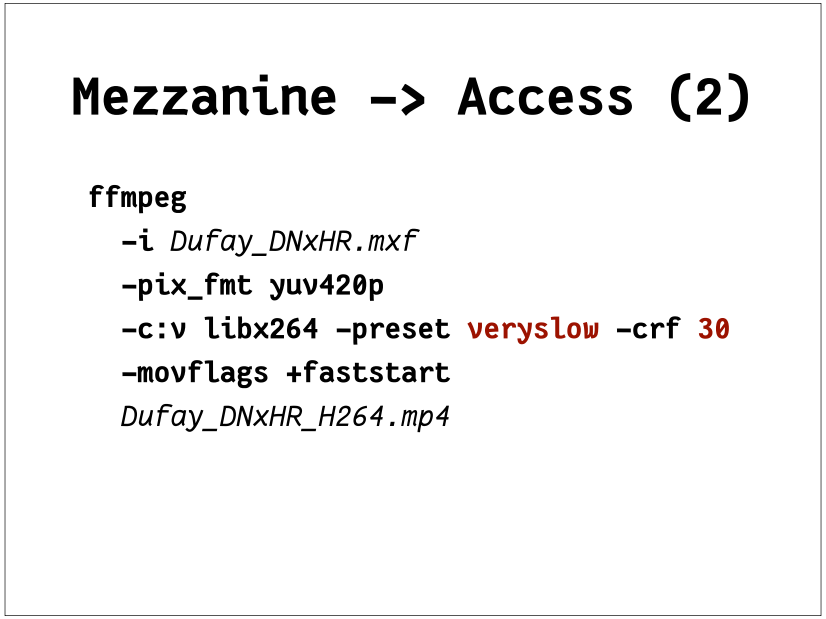
file size

52



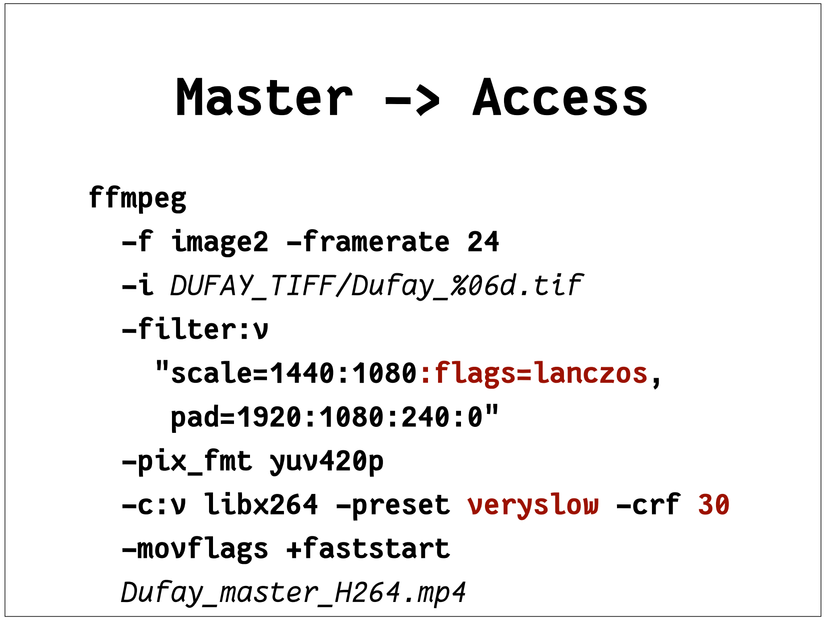
Mezzanine -> Access (1)

```
ffmpeg
  -i Dufay_ProRes.mkv
  -pix_fmt yuv420p
  -c:v libx264 -preset veryslow -crf 30
  -movflags +faststart
  Dufay_ProRes_H264.mp4
```



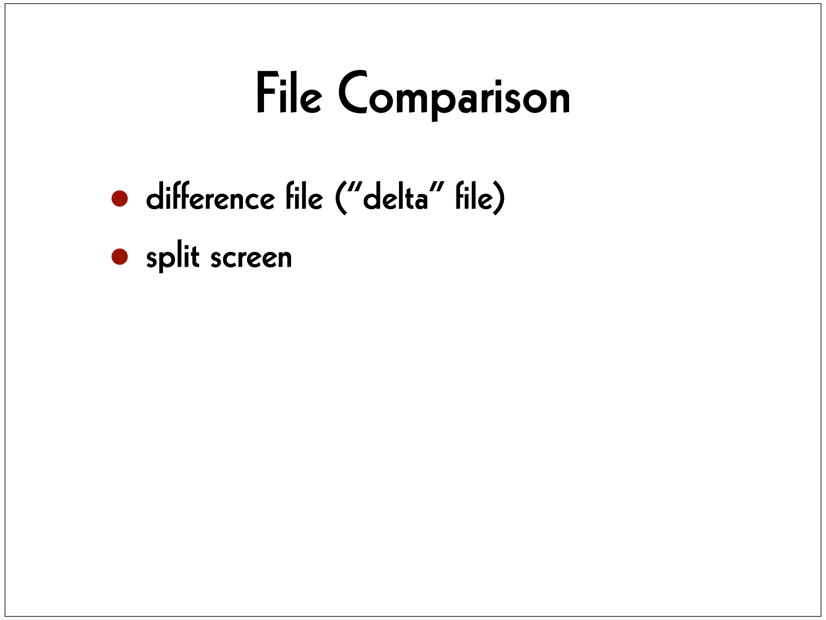
Mezzanine -> Access (2)

```
ffmpeg
  -i Dufay_DNxHR.mxf
  -pix_fmt yuv420p
  -c:v libx264 -preset veryslow -crf 30
  -movflags +faststart
  Dufay_DNxHR_H264.mp4
```



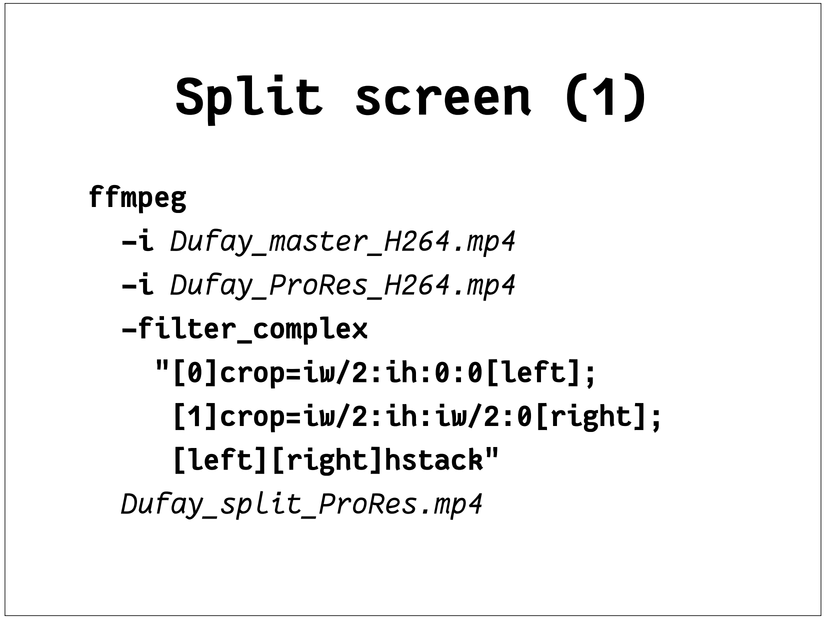
Master -> Access

```
ffmpeg
  -f image2 -framerate 24
  -i DUFAY_TIFF/Dufay_%06d.tif
  -filter:v
    "scale=1440:1080:flags=lanczos,
     pad=1920:1080:240:0"
  -pix_fmt yuv420p
  -c:v libx264 -preset veryslow -crf 30
  -movflags +faststart
  Dufay_master_H264.mp4
```



File Comparison

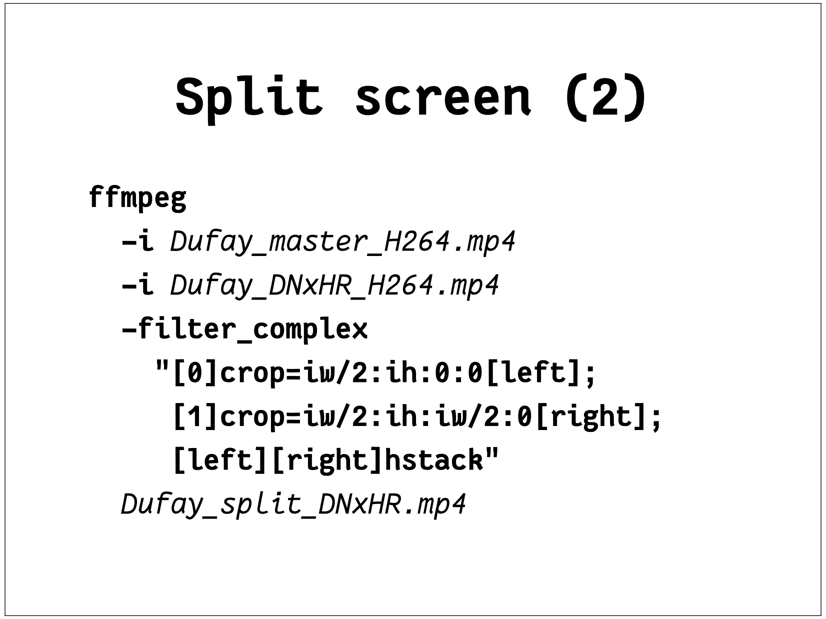
- difference file (“delta” file)
- split screen



Split screen (1)

```
ffmpeg
  -i Dufay_master_H264.mp4
  -i Dufay_ProRes_H264.mp4
  -filter_complex
    "[0]crop=iw/2:ih:0:0[left];
     [1]crop=iw/2:ih:iw/2:0[right];
     [left][right]hstack"
  Dufay_split_ProRes.mp4
```

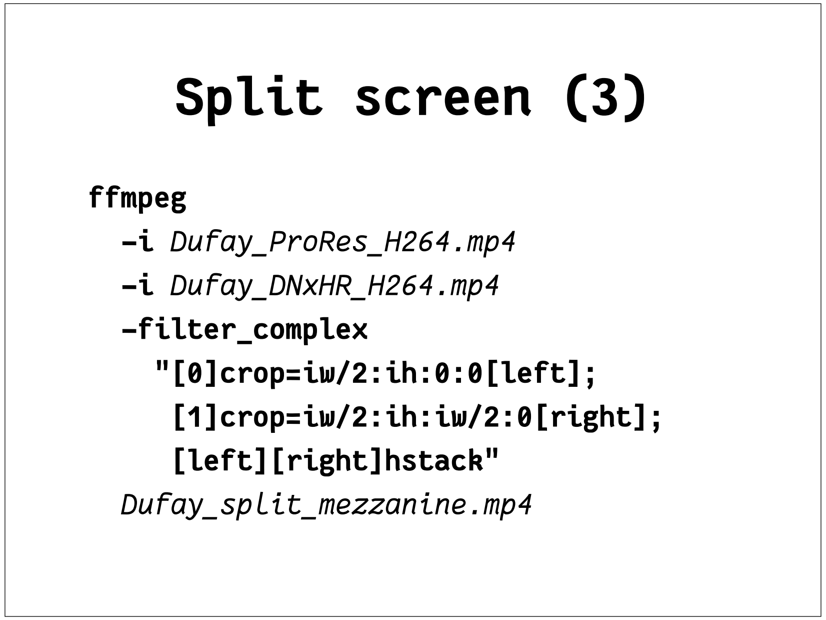
57



Split screen (2)

```
ffmpeg
  -i Dufay_master_H264.mp4
  -i Dufay_DNxHR_H264.mp4
  -filter_complex
    "[0]crop=iw/2:ih:0:0[left];
     [1]crop=iw/2:ih:iw/2:0[right];
     [left][right]hstack"
  Dufay_split_DNxHR.mp4
```

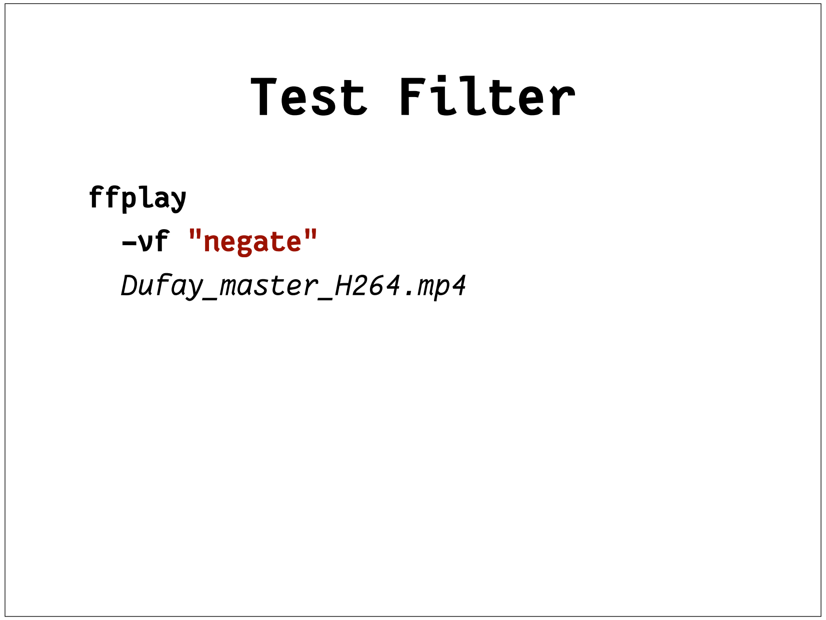
58



Split screen (3)

```
ffmpeg
  -i Dufay_ProRes_H264.mp4
  -i Dufay_DNxHR_H264.mp4
  -filter_complex
    "[0]crop=iw/2:ih:0:0[left];
     [1]crop=iw/2:ih:iw/2:0[right];
     [left][right]hstack"
  Dufay_split_mezzanine.mp4
```

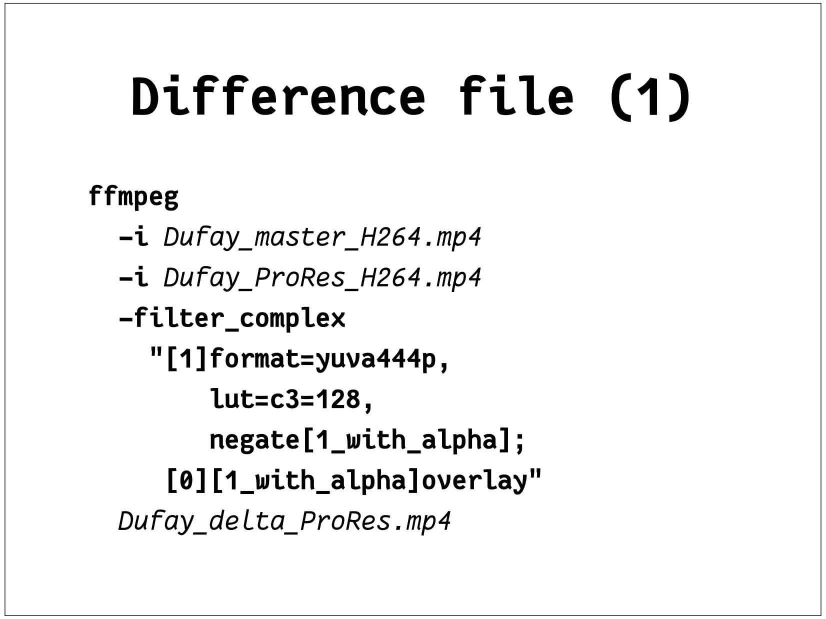
59



Test Filter

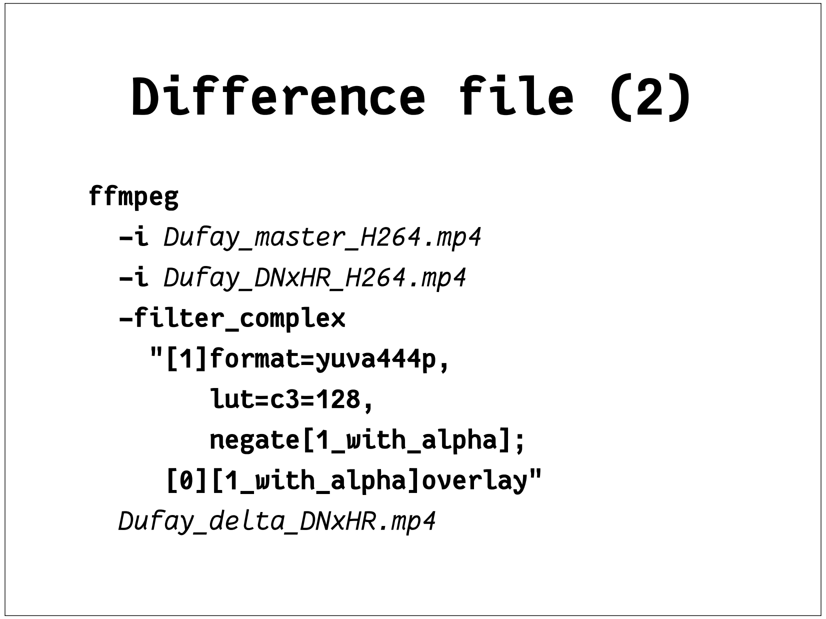
```
ffplay
  -vf "negate"
  Dufay_master_H264.mp4
```

60



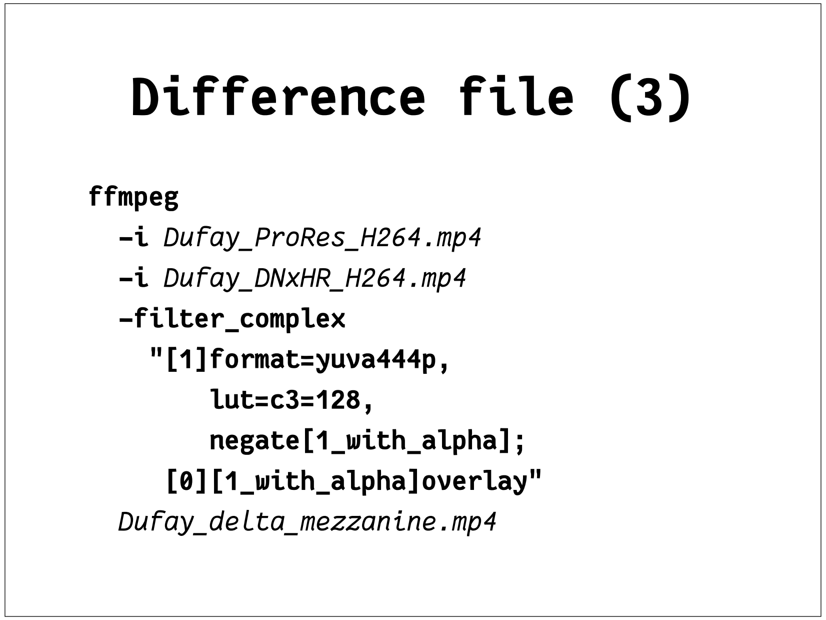
Difference file (1)

```
ffmpeg
  -i Dufay_master_H264.mp4
  -i Dufay_ProRes_H264.mp4
  -filter_complex
    "[1]format=yuva444p,
        lut=c3=128,
        negate[1_with_alpha];
     [0][1_with_alpha]overlay"
  Dufay_delta_ProRes.mp4
```



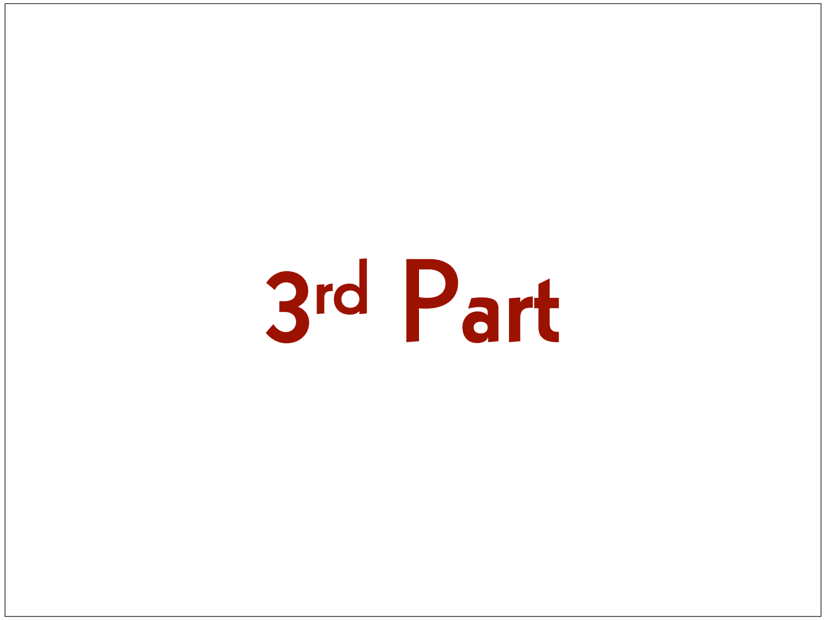
Difference file (2)

```
ffmpeg
  -i Dufay_master_H264.mp4
  -i Dufay_DNxHR_H264.mp4
  -filter_complex
    "[1]format=yuva444p,
        lut=c3=128,
        negate[1_with_alpha];
     [0][1_with_alpha]overlay"
  Dufay_delta_DNxHR.mp4
```



Difference file (3)

```
ffmpeg
  -i Dufay_ProRes_H264.mp4
  -i Dufay_DNxHR_H264.mp4
  -filter_complex
    "[1]format=yuva444p,
        lut=c3=128,
        negate[1_with_alpha];
     [0][1_with_alpha]overlay"
  Dufay_delta_mezzanine.mp4
```



3rd Part

Workflow

define accepted file formats

perform quality control

- checksum
- filename
- container, codec and data formats
- image and sound content

prepare archive package

store packages (e.g. onto LTO tapes)

Quality control

- check technical metadata
- analyse signal
- watch image and listen sound

- difference file
- split screen

Quality Control Tools

container and codec

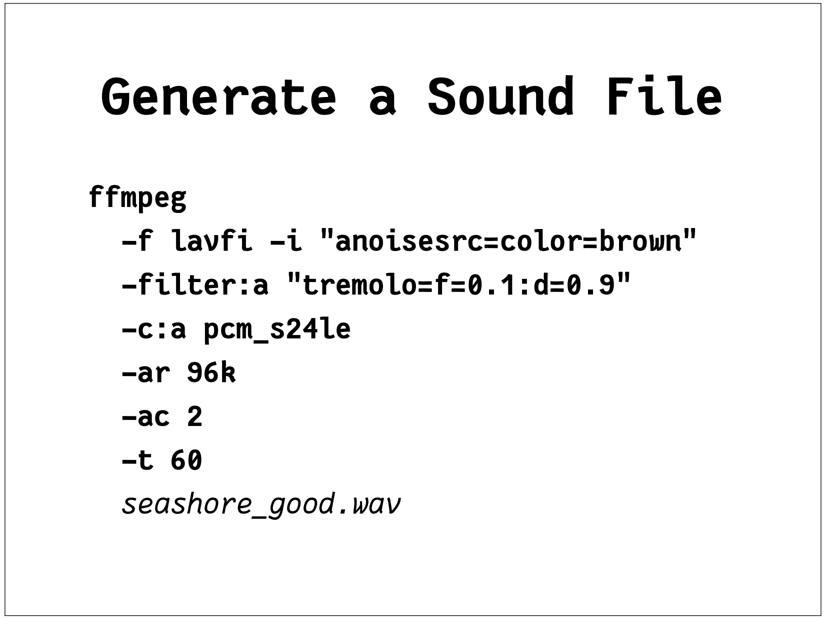
- MediaInfo, ffprobe, MediaConch
- hexdump, fq

image and sound content

- QCTools, qcli, SignalServer
- VLC, mpv, ffplay

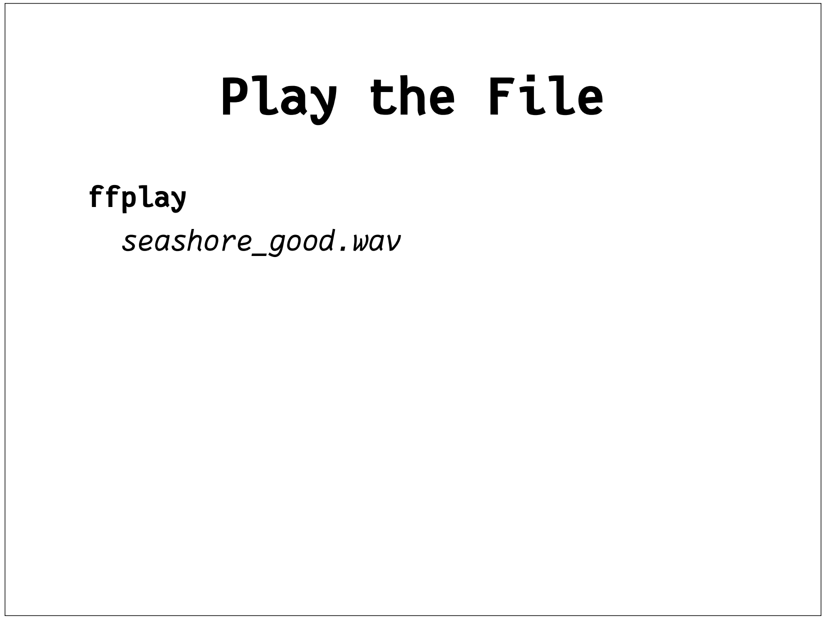
Archival Tools

- RAWcooked
- BagIt



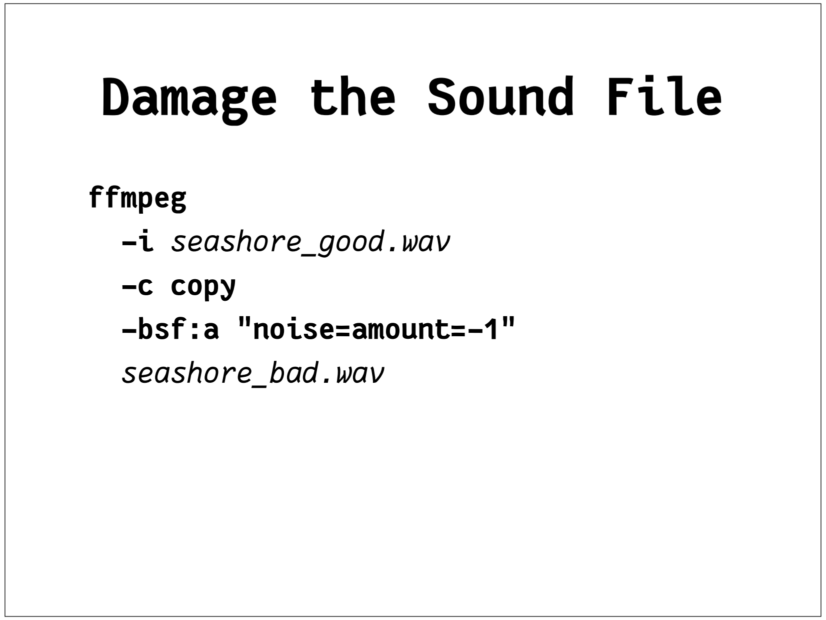
Generate a Sound File

```
ffmpeg
  -f lavfi -i "anoisesrc=color=brown"
  -filter:a "tremolo=f=0.1:d=0.9"
  -c:a pcm_s24le
  -ar 96k
  -ac 2
  -t 60
  seashore_good.wav
```



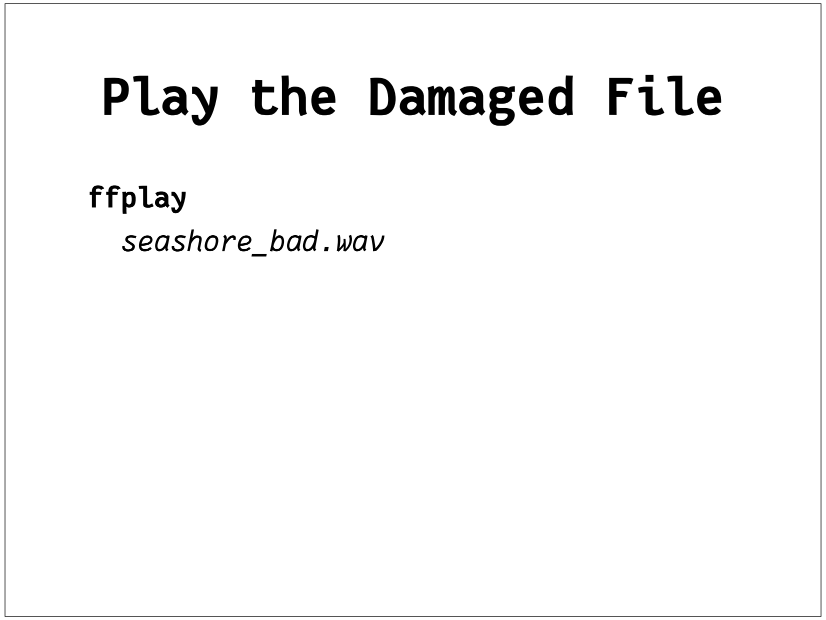
Play the File

```
ffplay
  seashore_good.wav
```



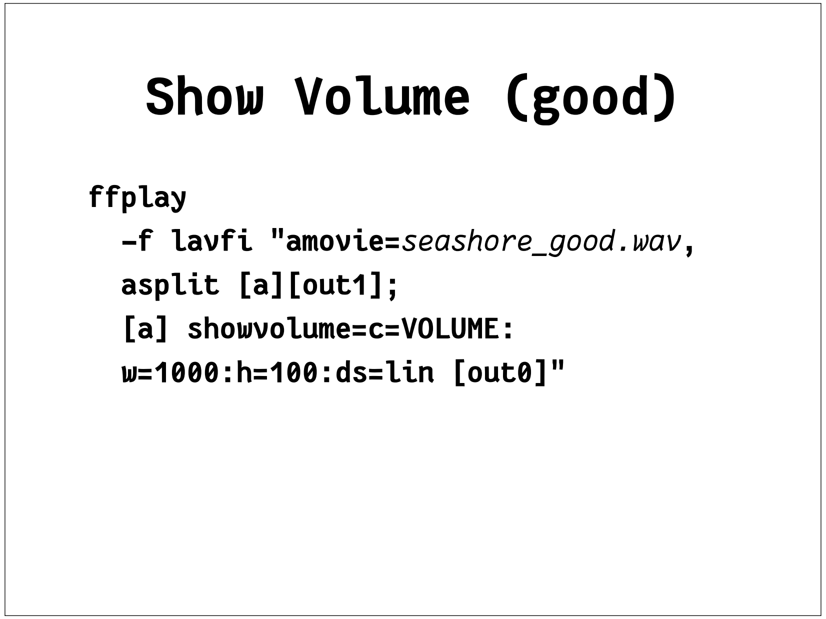
Damage the Sound File

```
ffmpeg
  -i seashore_good.wav
  -c copy
  -bsf:a "noise=amount=-1"
  seashore_bad.wav
```



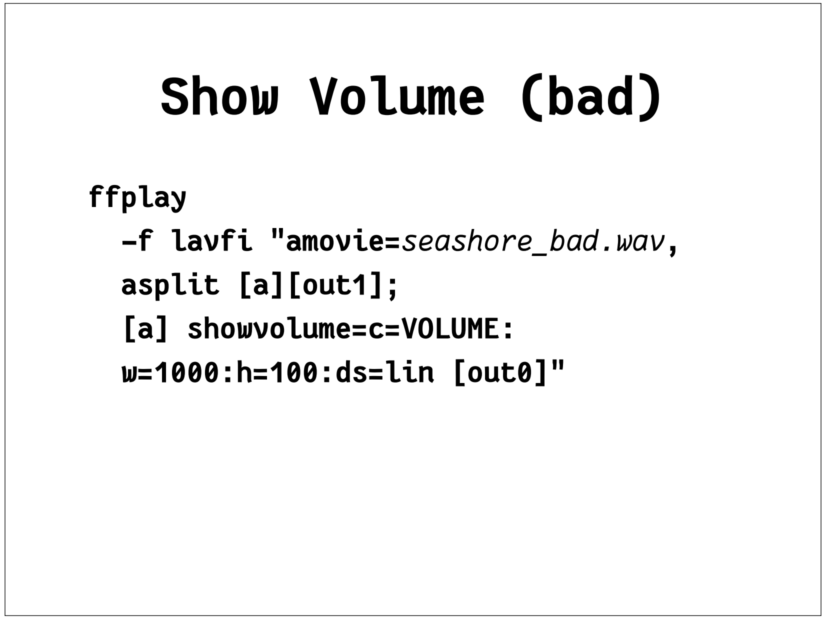
Play the Damaged File

```
ffplay
  seashore_bad.wav
```



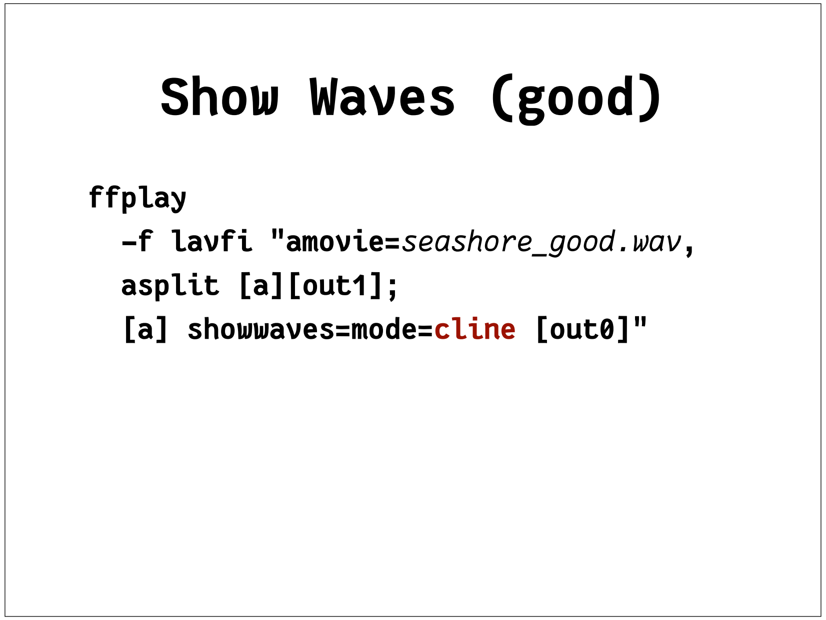
Show Volume (good)

```
ffplay
  -f lavfi "amovie=seashore_good.wav,
  asplit [a][out1];
  [a] showvolume=c=VOLUME:
  w=1000:h=100:ds=lin [out0]"
```



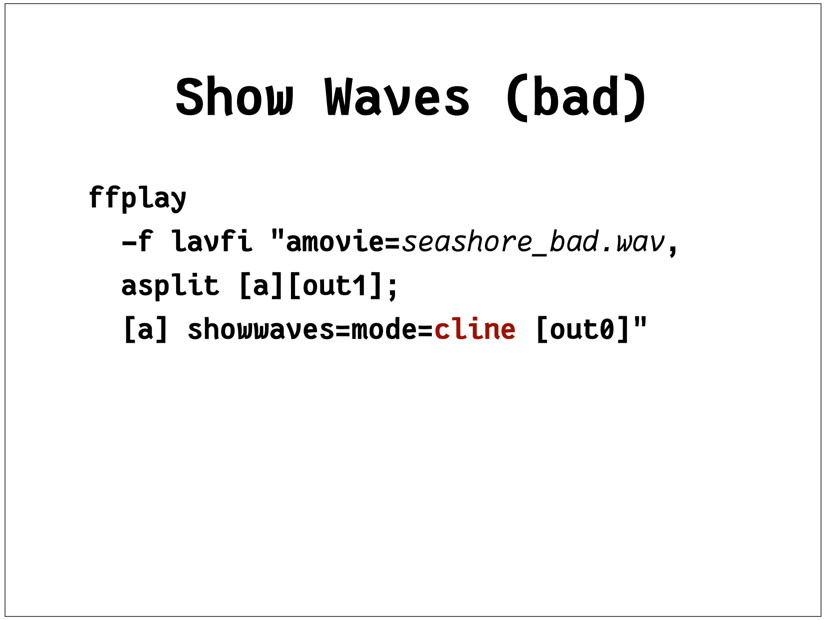
Show Volume (bad)

```
ffplay
  -f lavfi "amovie=seashore_bad.wav,
  asplit [a][out1];
  [a] showvolume=c=VOLUME:
  w=1000:h=100:ds=lin [out0]"
```



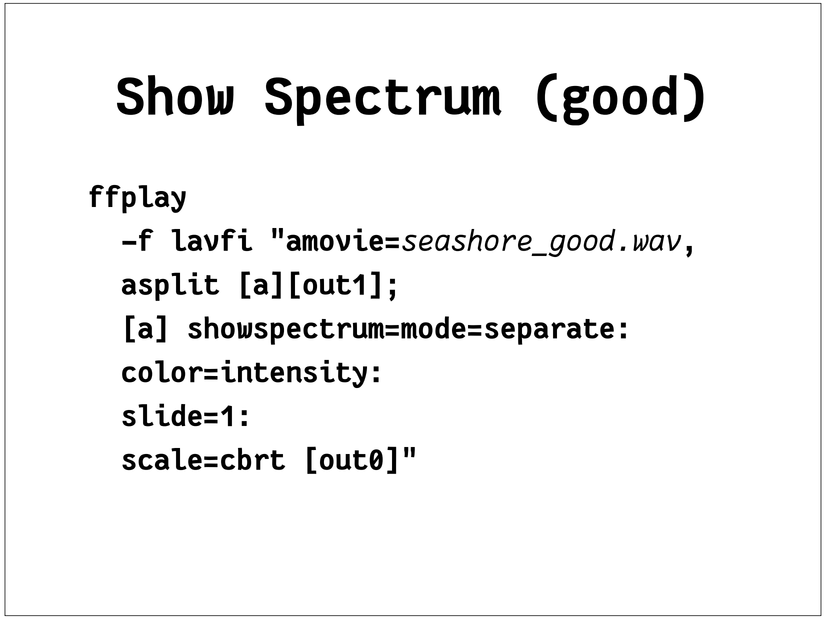
Show Waves (good)

```
ffplay
  -f lavfi "amovie=seashore_good.wav,
  asplit [a][out1];
  [a] showwaves=mode=cline [out0]"
```



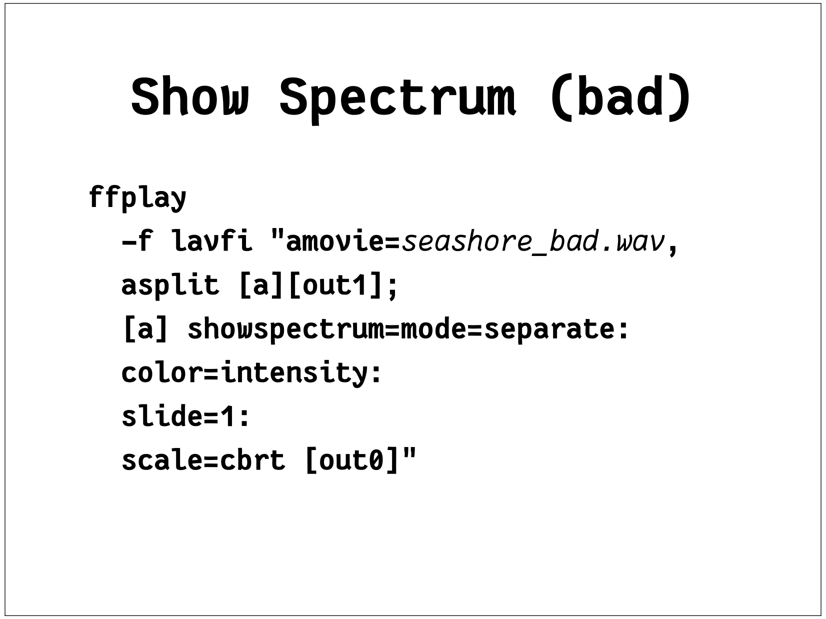
Show Waves (bad)

```
ffplay
  -f lavfi "amovie=seashore_bad.wav,
  asplit [a][out1];
  [a] showwaves=mode=cline [out0]"
```



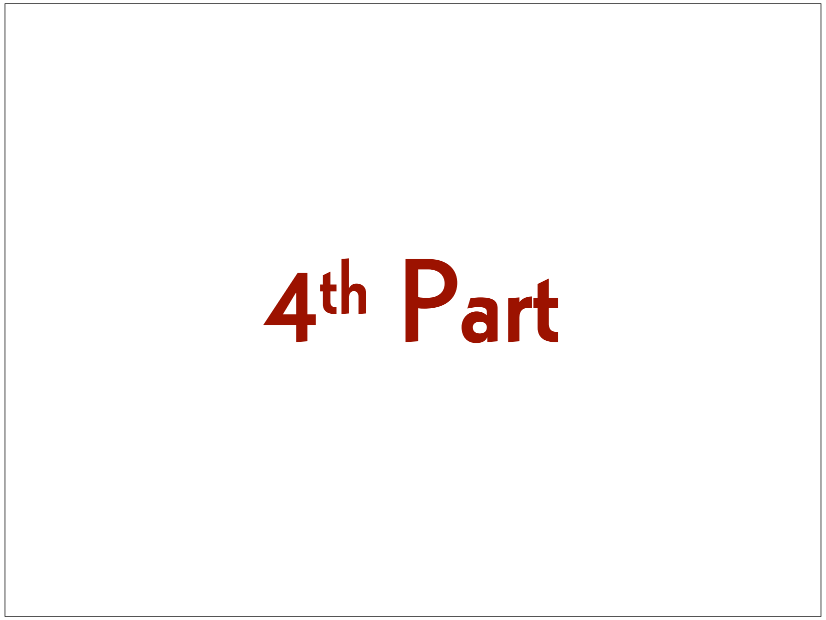
Show Spectrum (good)

```
ffplay
  -f lavfi "amovie=seashore_good.wav,
  asplit [a][out1];
  [a] showspectrum=mode=separate:
  color=intensity:
  slide=1:
  scale=cbrt [out0]"
```

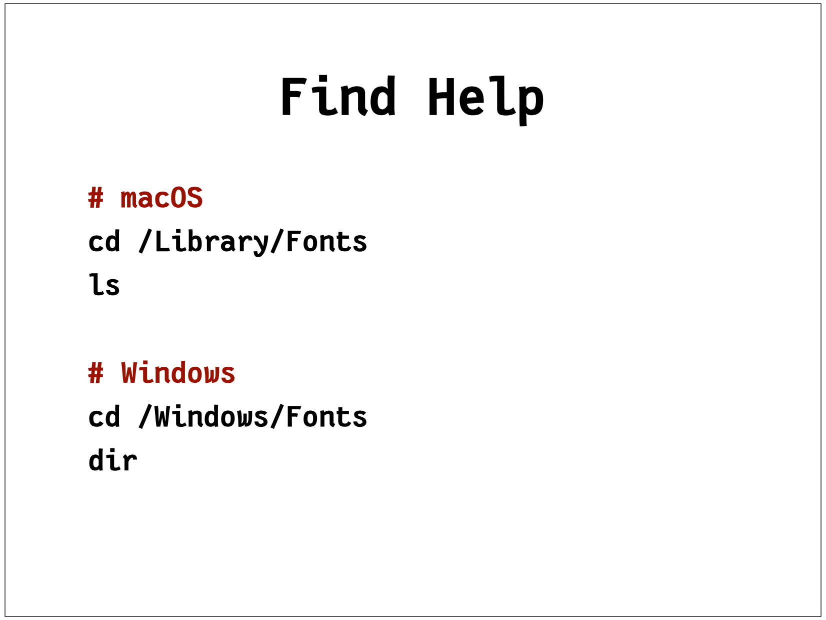


Show Spectrum (bad)

```
ffplay
  -f lavfi "amovie=seashore_bad.wav,
  asplit [a][out1];
  [a] showspectrum=mode=separate:
  color=intensity:
  slide=1:
  scale=cbrt [out0]"
```

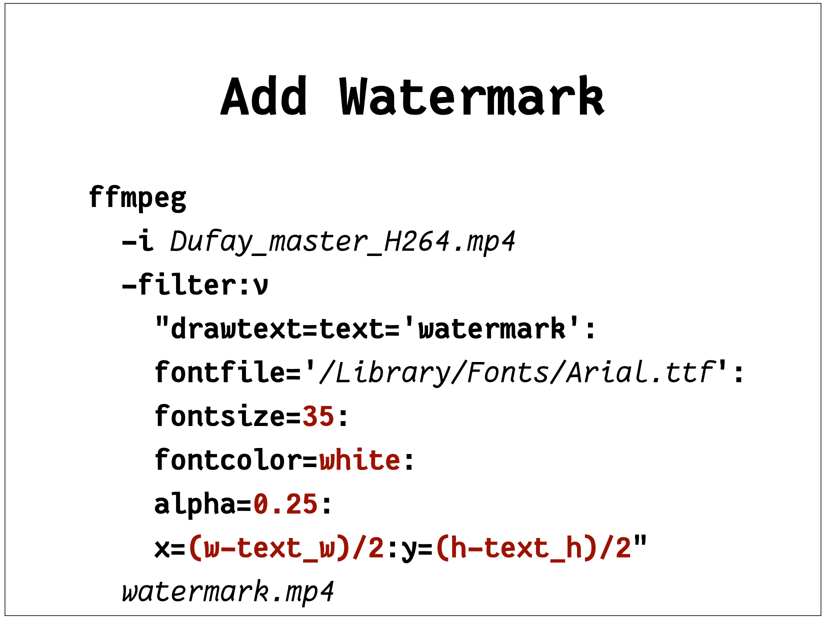


4th Part



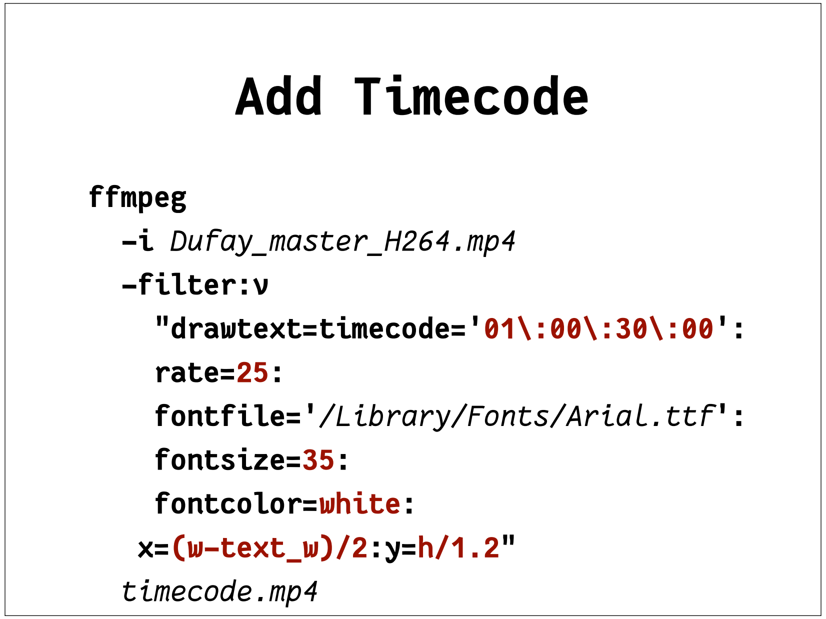
Find Help

```
# macOS
cd /Library/Fonts
ls

# Windows
cd /Windows/Fonts
dir
```



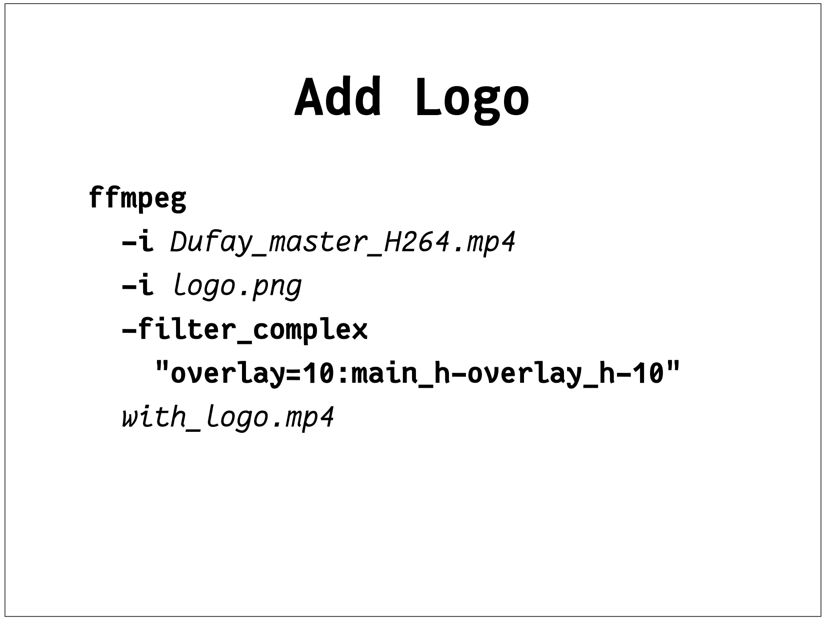
Add Watermark

```
ffmpeg
  -i Dufay_master_H264.mp4
  -filter:v
    "drawtext=text='watermark':
    fontfile='/Library/Fonts/Arial.ttf':
    fontsize=35:
    fontcolor=white:
    alpha=0.25:
    x=(w-text_w)/2:y=(h-text_h)/2"
  watermark.mp4
```



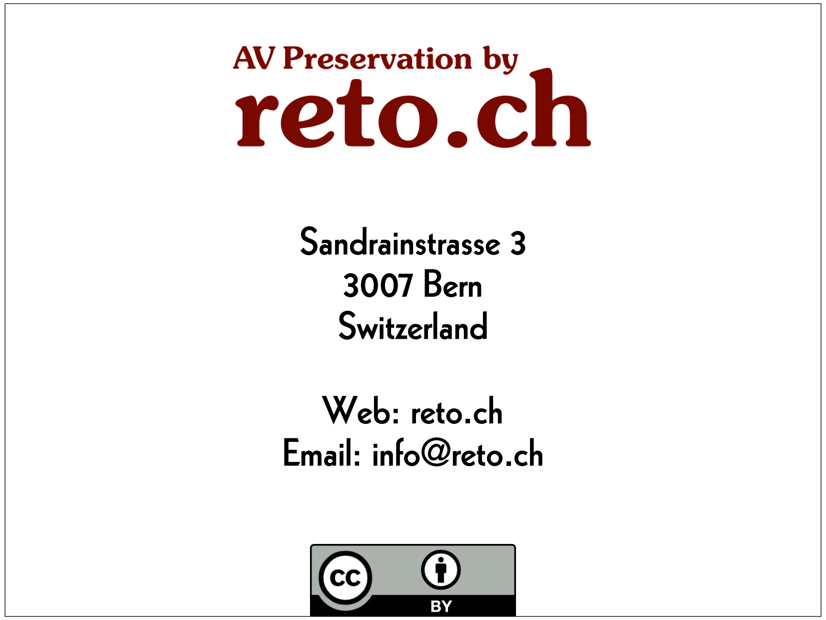
Add Timecode

```
ffmpeg
  -i Dufay_master_H264.mp4
  -filter:v
    "drawtext=timecode='01\:00\:30\:00':
    rate=25:
    fontfile='/Library/Fonts/Arial.ttf':
    fontsize=35:
    fontcolor=white:
   x=(w-text_w)/2:y=h/1.2"
  timecode.mp4
```



Add Logo

```
ffmpeg
  -i Dufay_master_H264.mp4
  -i logo.png
  -filter_complex
    "overlay=10:main_h-overlay_h-10"
  with_logo.mp4
```



AV Preservation by
reto.ch

Sandrainstrasse 3
3007 Bern
Switzerland

Web: reto.ch
Email: info@reto.ch