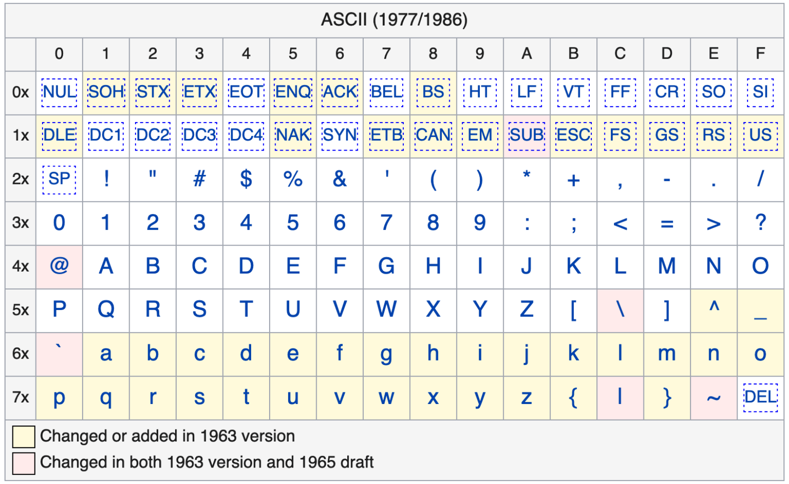
# FFmpeg

Reto Kromer • AV Preservation by reto.ch

**Using FFmpeg in a film archive**
FIAF, on-line, 30 Mars 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

source: wikipedia.org

# 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

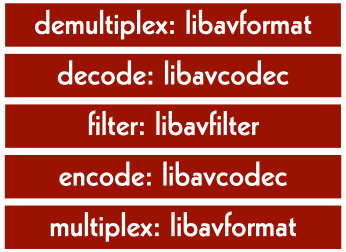
**Tools**

- ffmpeg
- ffprobe
- ffplay

**Libraries**

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

# Data Transformations



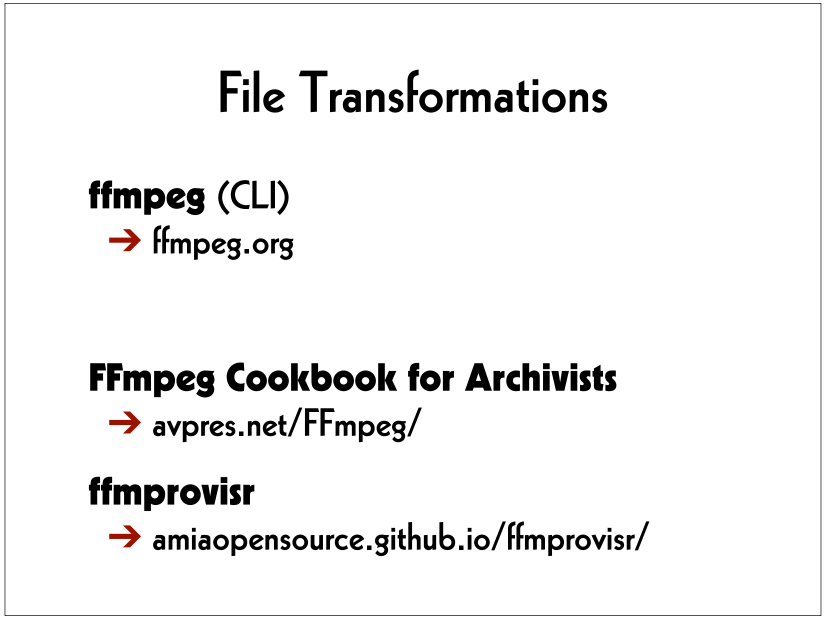
demultiplex: libavformat

decode: libavcodec

filter: libavfilter

encode: libavcodec

multiplex: libavformat

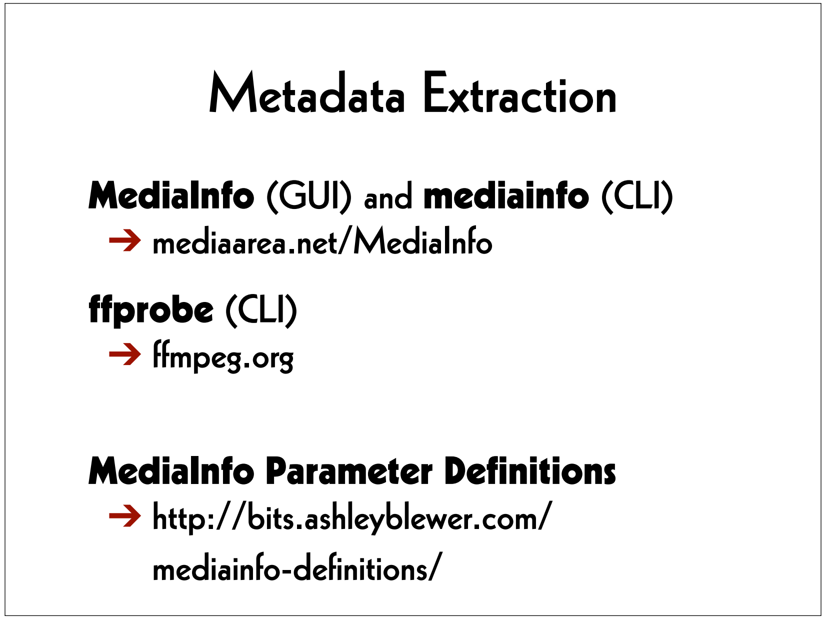


## File Transformations

**ffmpeg** (CLI)
→ ffmpeg.org

**FFmpeg Cookbook for Archivists**
→ avpres.net/FFmpeg/

**ffmprovisr**
→ amiaopensource.github.io/ffmprovisr/

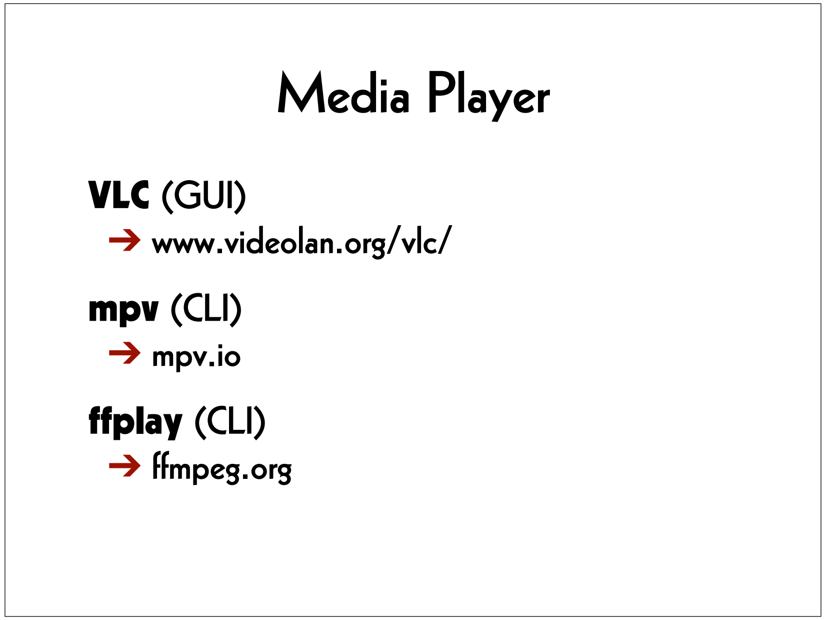


## Metadata Extraction

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

**ffprobe** (CLI)
→ ffmpeg.org

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

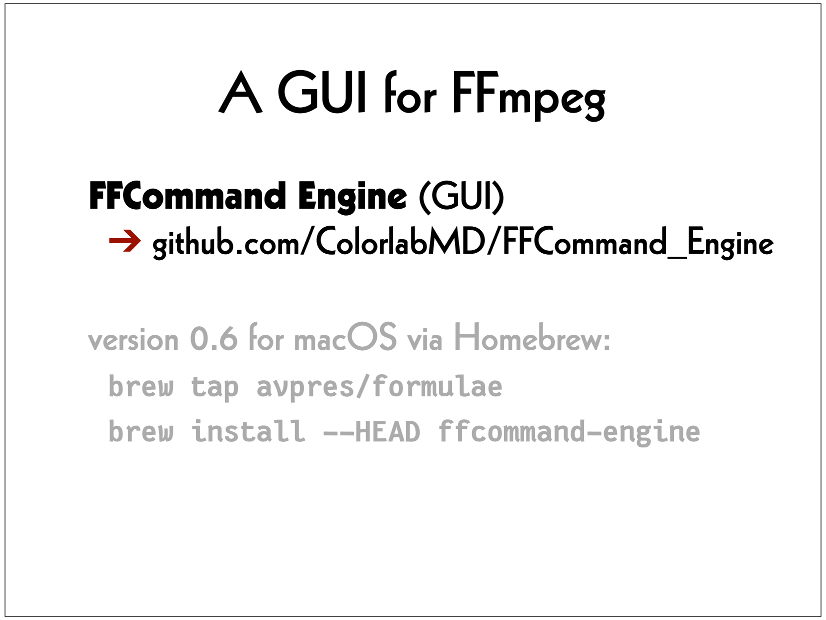


## Media Player

**VLC** (GUI)
→ www.videolan.org/vlc/

**mpv** (CLI)
→ mpv.io

**ffplay** (CLI)
→ ffmpeg.org

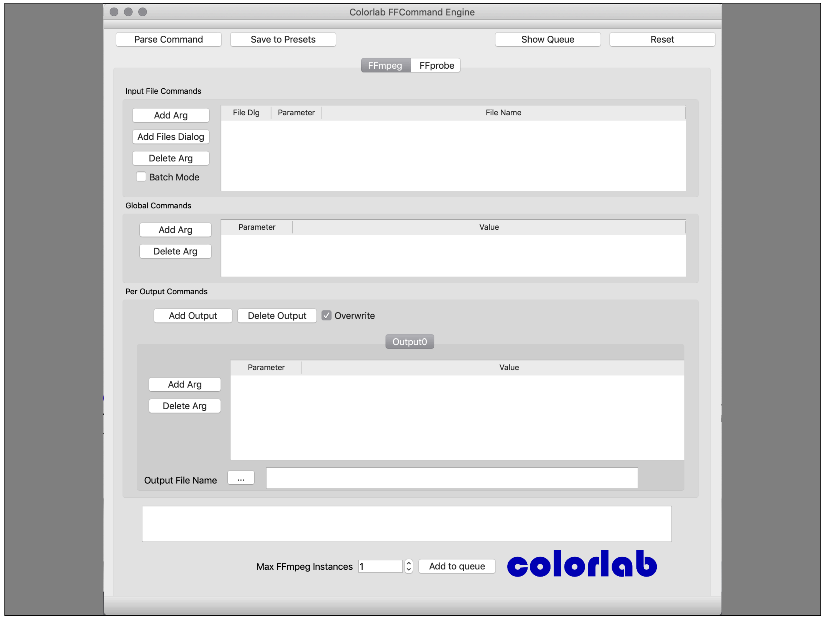


## A GUI for FFmpeg

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

version 0.6 for macOS via Homebrew:

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



# Exercises

## Set the working directory

Linux/Mac/Windows with Terminal or WSL:

```
cd ~/Desktop
```

Windows locally:

```
cd Desktop
```

Windows on OneDrive Cloud:

```
cd OneDrive
```

## Find Help

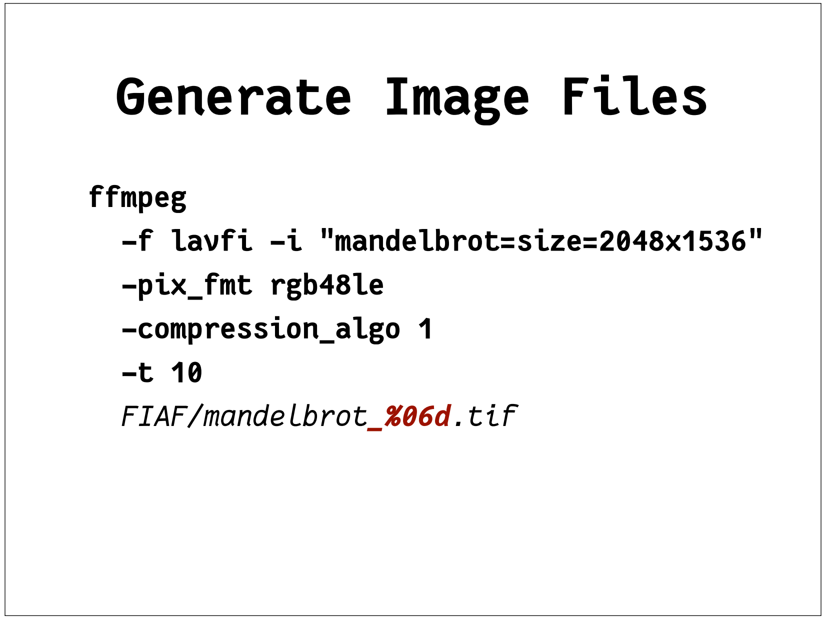
```
ffmpeg -h encoder=tiff

ffplay -f lavfi -i testsrc
ffplay -f lavfi -i testsrc2
ffplay -f lavfi -i mandelbrot
```

# Create Exercice Folder

```
mkdir
  FIAF
```

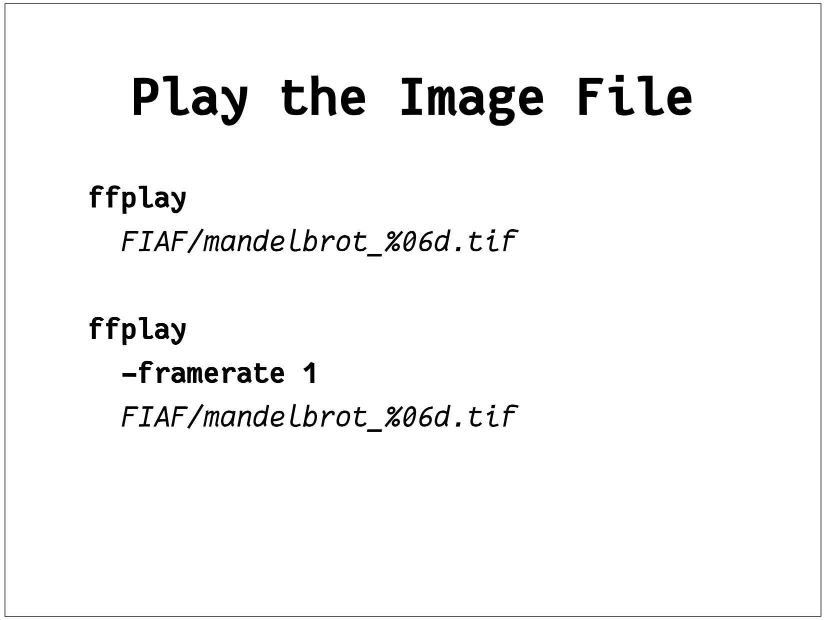
17



# Generate Image Files

```
ffmpeg
  -f lavfi -i "mandelbrot=size=2048x1536"
  -pix_fmt rgb48le
  -compression_algo 1
  -t 10
  FIAF/mandelbrot_%06d.tif
```

18

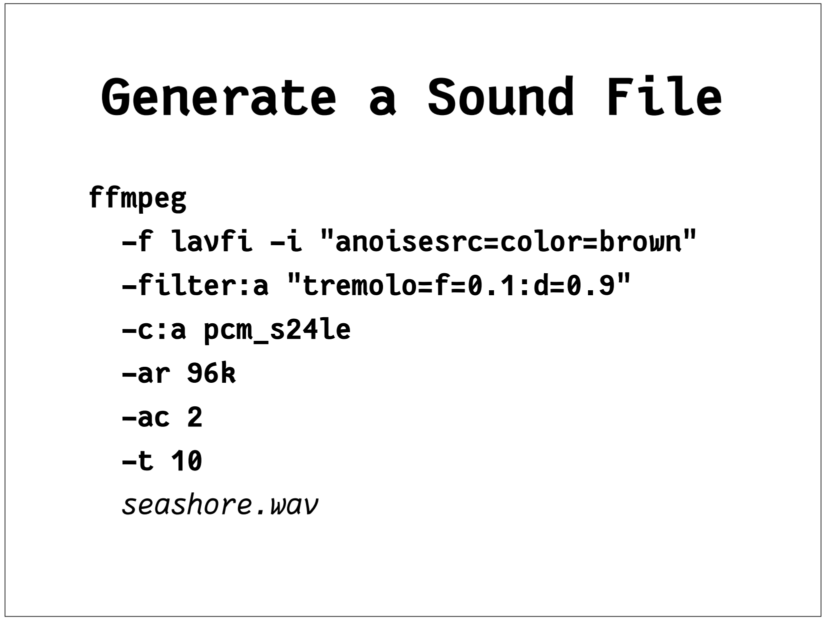


# Play the Image File

```
ffplay
  FIAF/mandelbrot_%06d.tif

ffplay
  -framerate 1
  FIAF/mandelbrot_%06d.tif
```

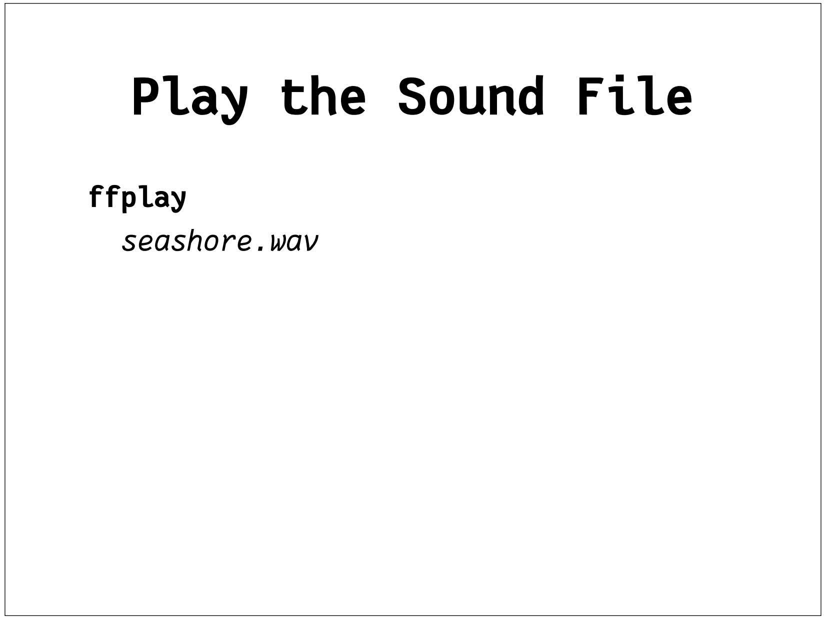
19



# Generate a Sound File

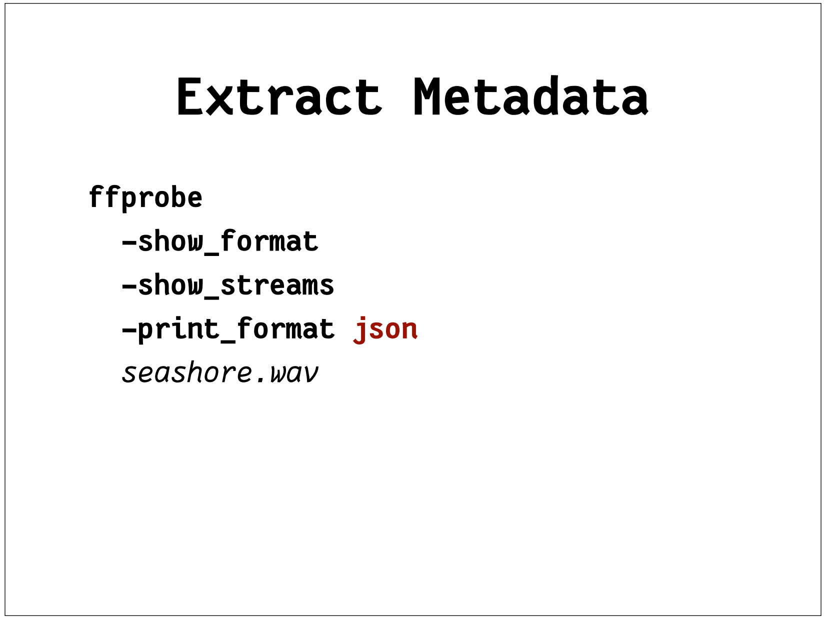
```
ffmpeg
  -f lavfi -i "anoisesrc=color=brown"
  -filter:a "tremolo=f=0.1:d=0.9"
  -c:a pcm_s24le
  -ar 96k
  -ac 2
  -t 10
  seashore.wav
```

20



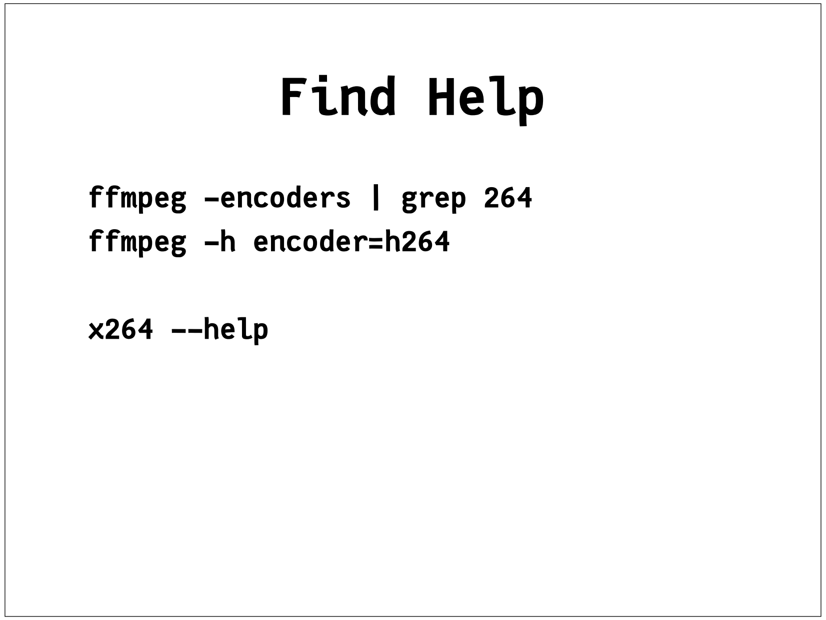
## Play the Sound File

```
ffplay
  seashore.wav
```



## Extract Metadata

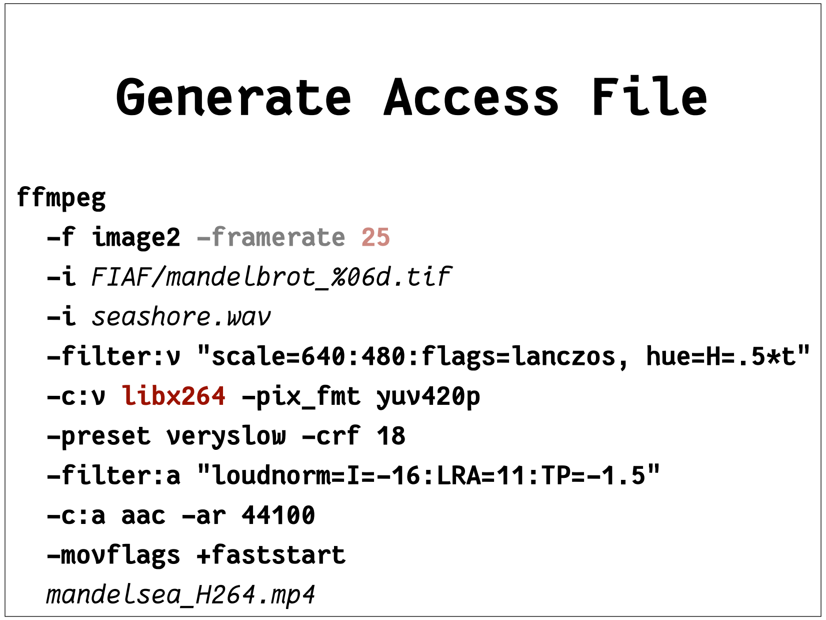
```
ffprobe
  -show_format
  -show_streams
  -print_format json
  seashore.wav
```



## Find Help

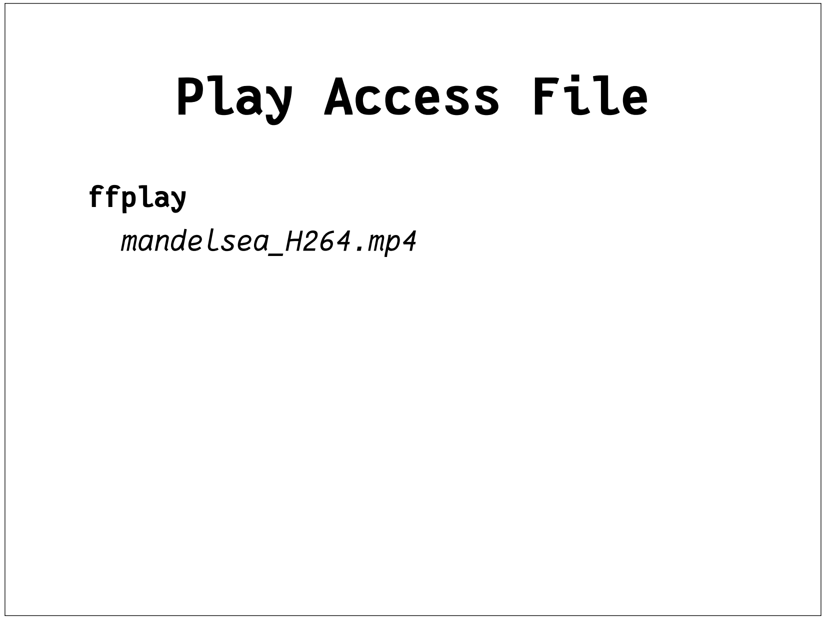
```
ffmpeg -encoders | grep 264
ffmpeg -h encoder=h264

x264 --help
```



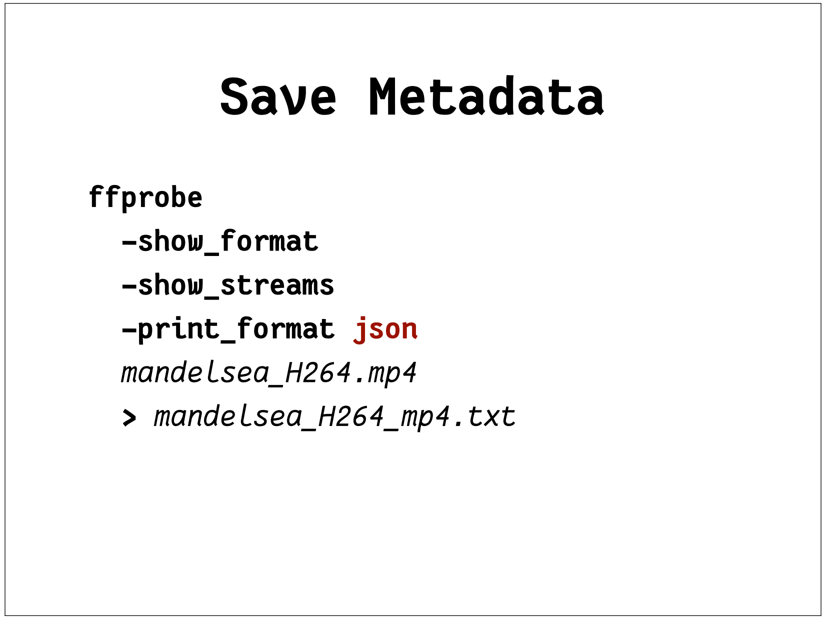
## Generate Access File

```
ffmpeg
  -f image2 -framerate 25
  -i FIAF/mandelbrot_%06d.tif
  -i seashore.wav
  -filter:v "scale=640:480:flags=lanczos, hue=H=.5*t"
  -c:v libx264 -pix_fmt yuv420p
  -preset veryslow -crf 18
  -filter:a "loudnorm=I=-16:LRA=11:TP=-1.5"
  -c:a aac -ar 44100
  -movflags +faststart
  mandelsea_H264.mp4
```



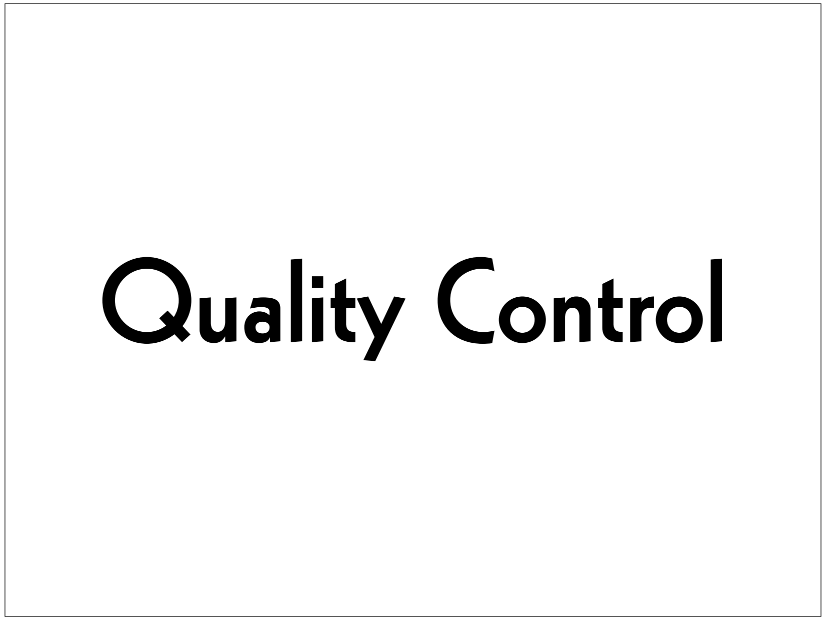
# Play Access File

```
ffplay
  mandelsea_H264.mp4
```

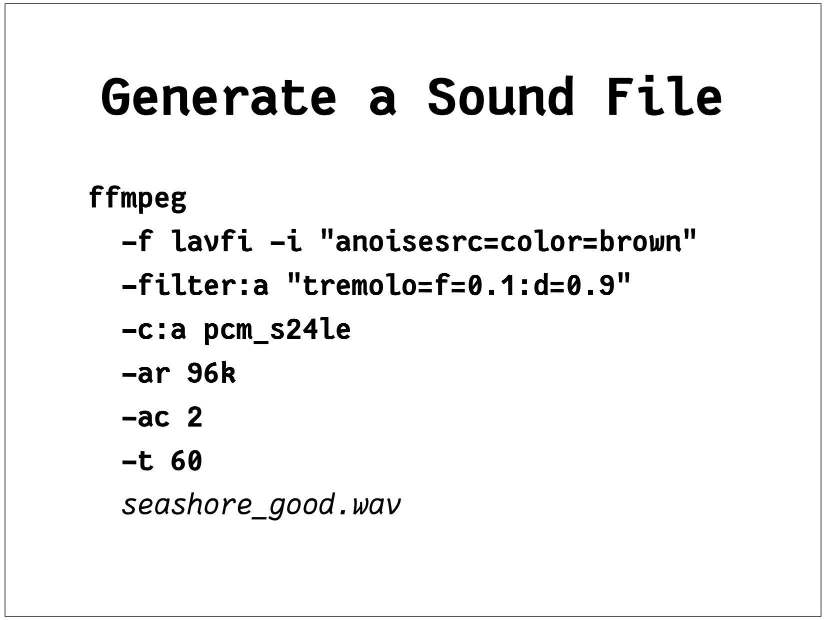


# Save Metadata

```
ffprobe
  -show_format
  -show_streams
  -print_format json
  mandelsea_H264.mp4
  > mandelsea_H264_mp4.txt
```

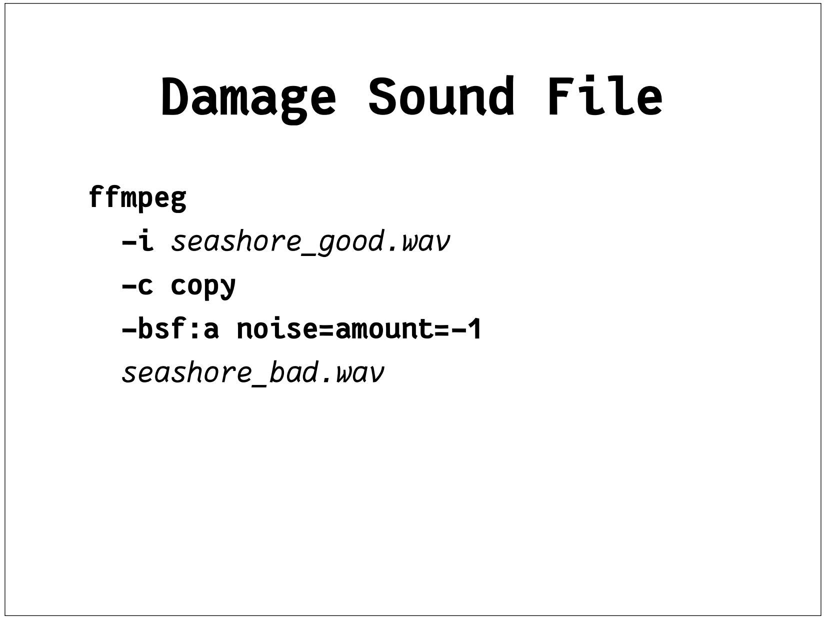


# Quality Control



# Generate a Sound File

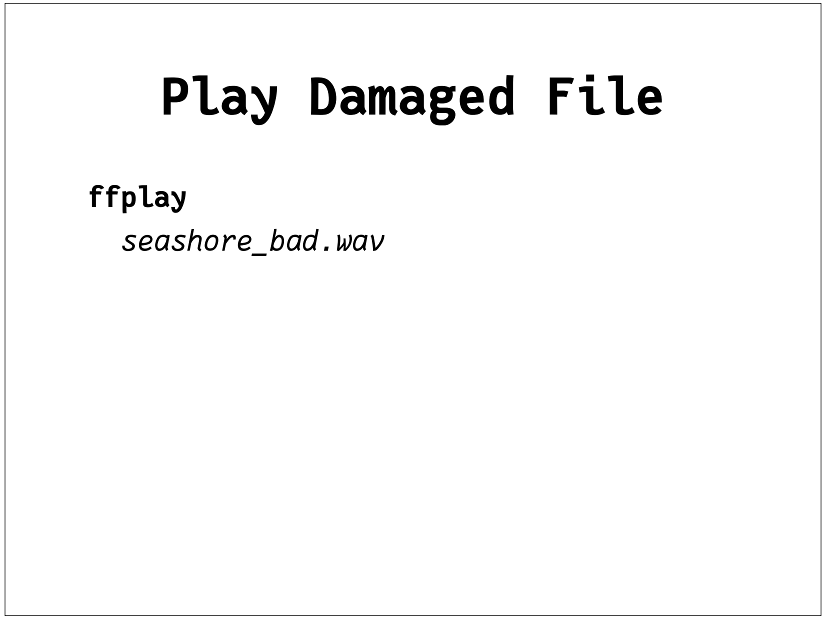
```
ffmpeg
  -f lavfi -i "anoisesrc=color=brown"
  -filter:a "tremolo=f=0.1:d=0.9"
  -c:a pcm_s24le
  -ar 96k
  -ac 2
  -t 60
  seashore_good.wav
```



## Damage Sound File

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

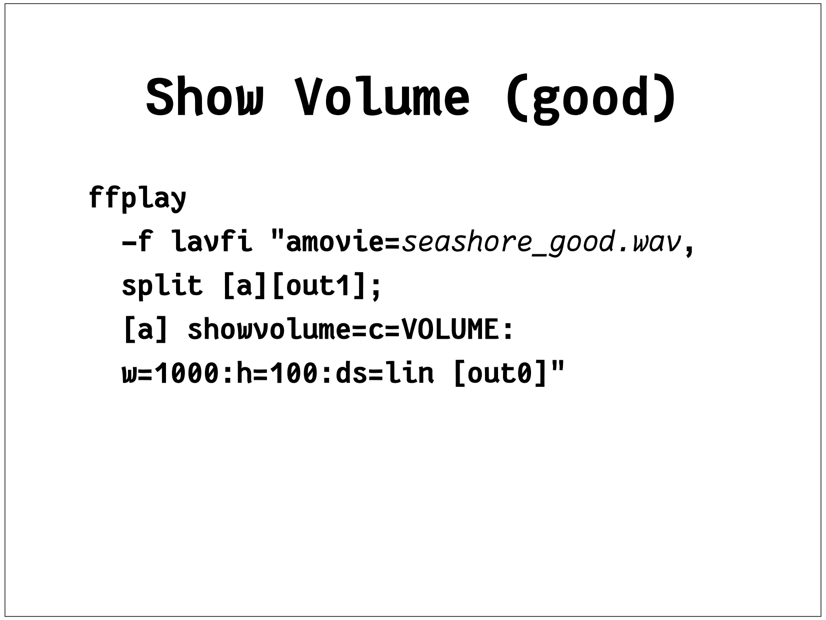
29



## Play Damaged File

```
ffplay
  seashore_bad.wav
```

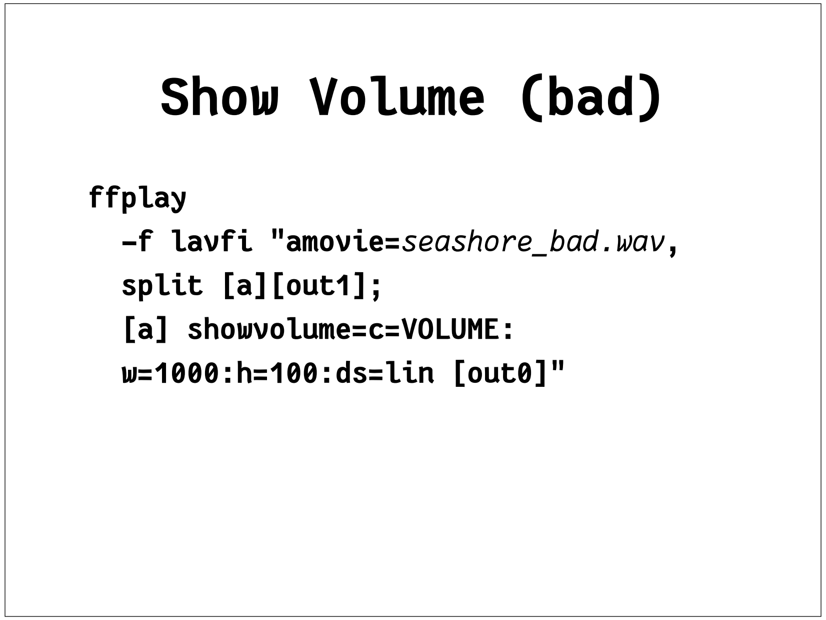
30



## Show Volume (good)

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

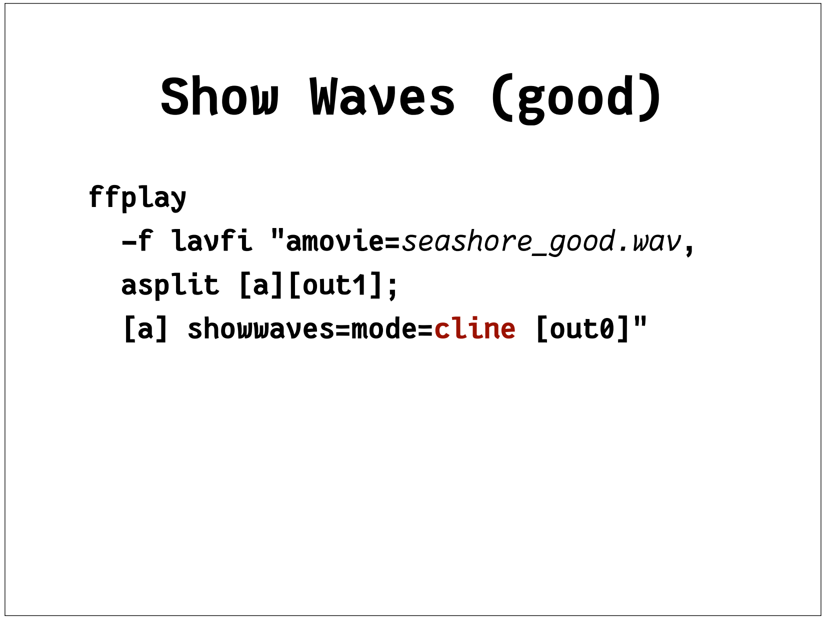
31



## Show Volume (bad)

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

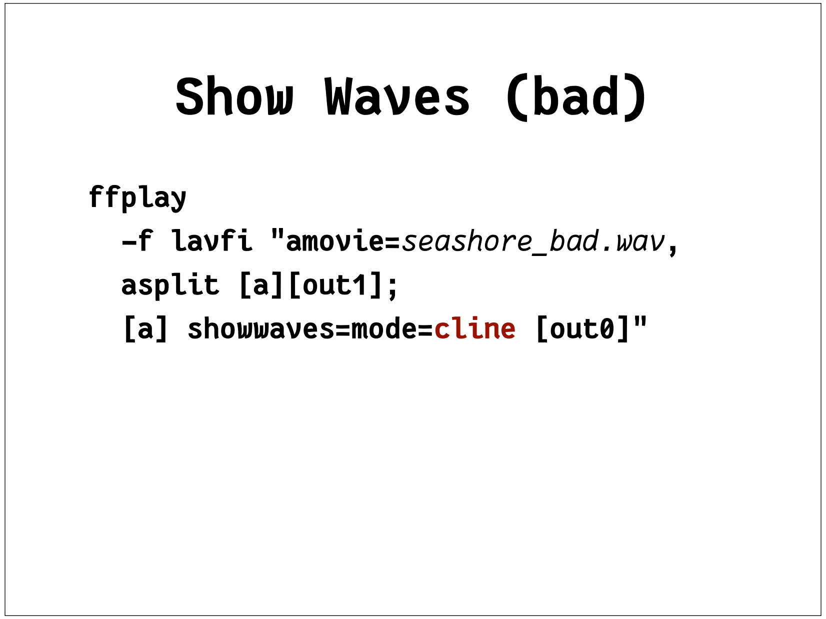
32



# Show Waves (good)

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

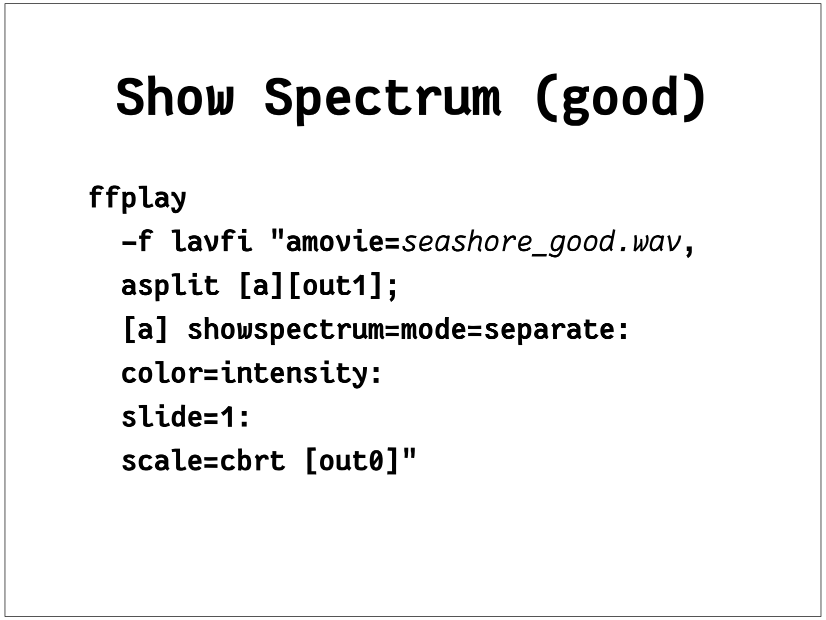
33



# Show Waves (bad)

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

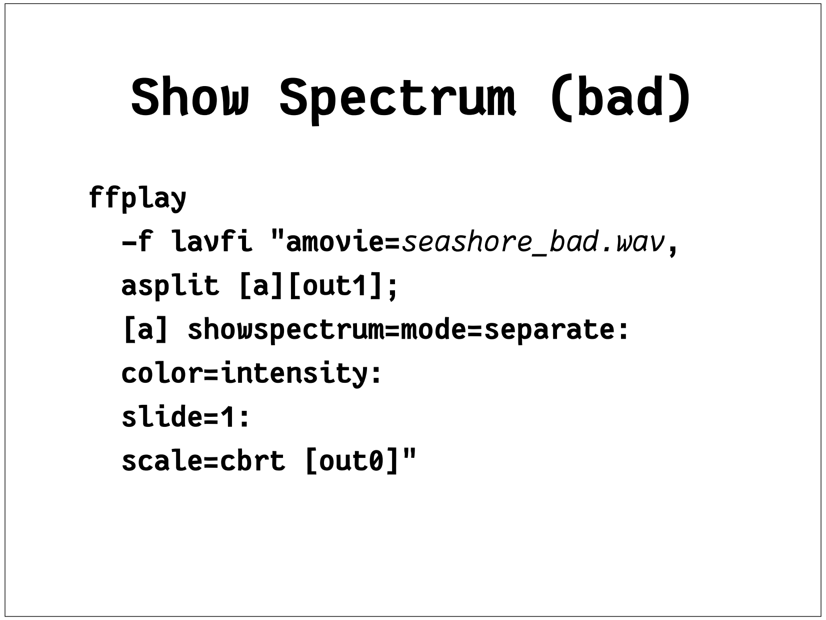
34



# Show Spectrum (good)

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

35

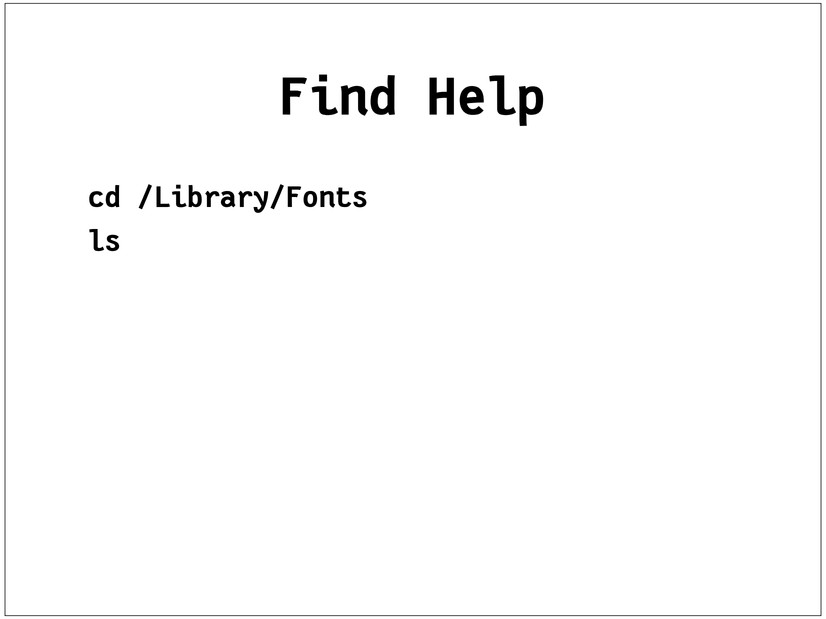


# Show Spectrum (bad)

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

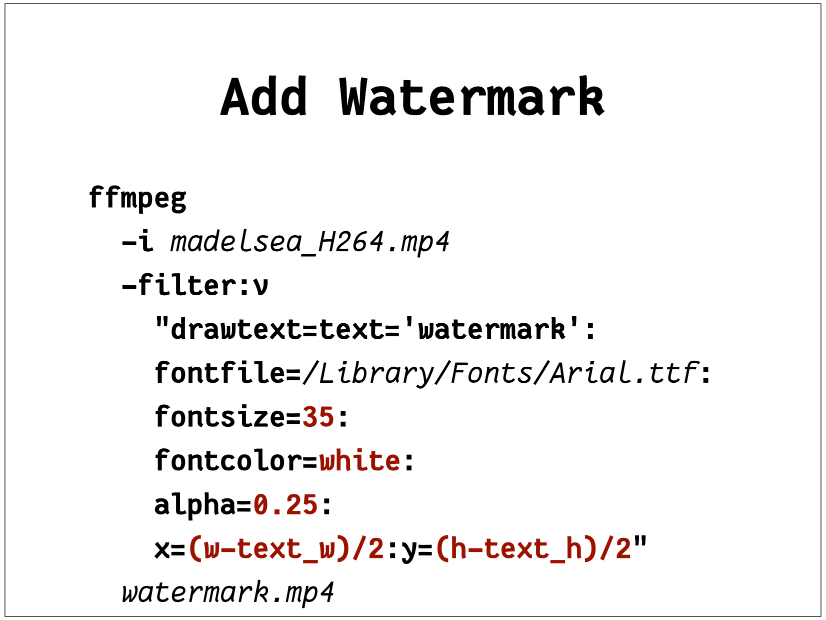
36

# Personalise



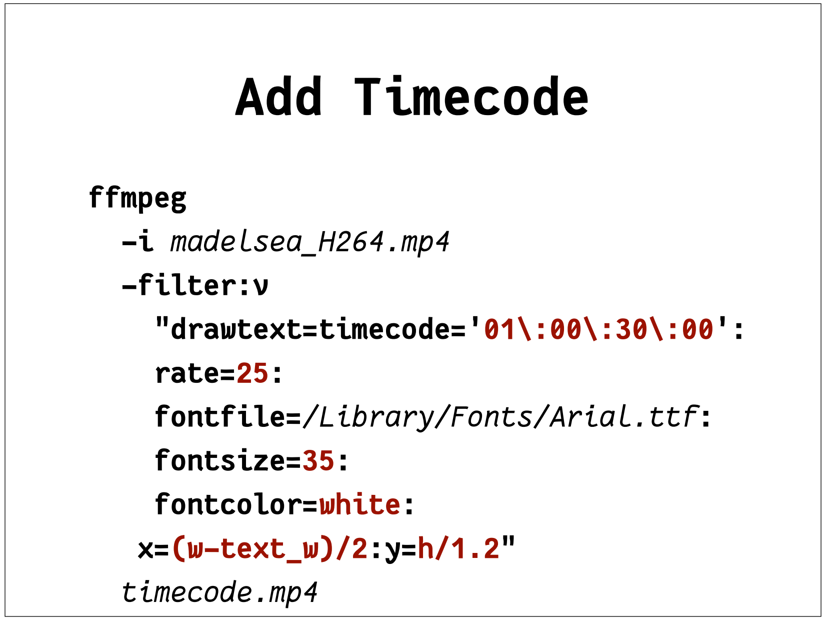
## Find Help

```
cd /Library/Fonts
ls
```



## Add Watermark

```
ffmpeg
  -i madelsea_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 madelsea_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 madelsea_H264.mp4
  -i logo.png
  -filter_complex
    "overlay=10:main_h-overlay_h-10"
  with_logo.mp4
```

# Not Used

# Workflow

define accepted file formats

perform quality control

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

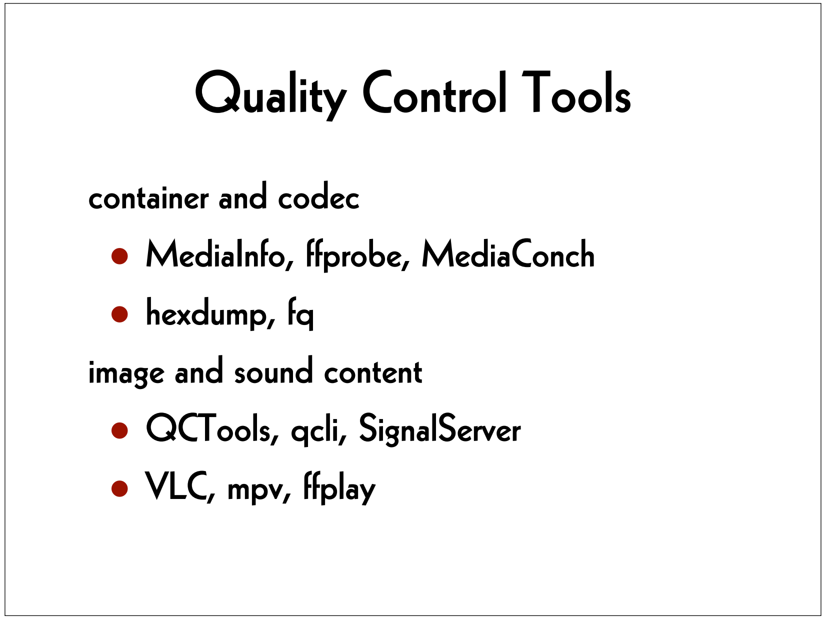
prepare archive package

store packages (e.g. onto LTO tapes)

# Quality control

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

- difference file
- split screen



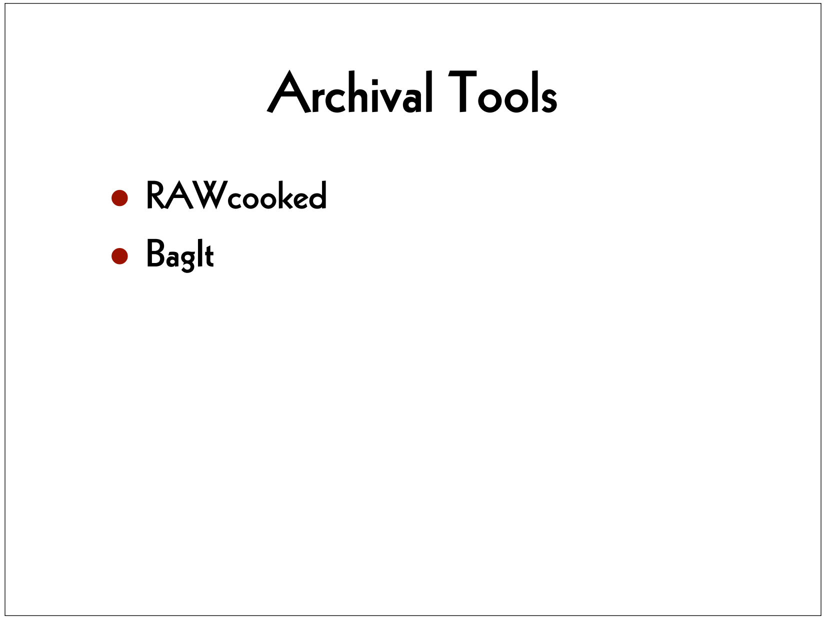
# Quality Control Tools

container and codec

- MediaInfo, ffprobe, MediaConch
- hexdump, fq

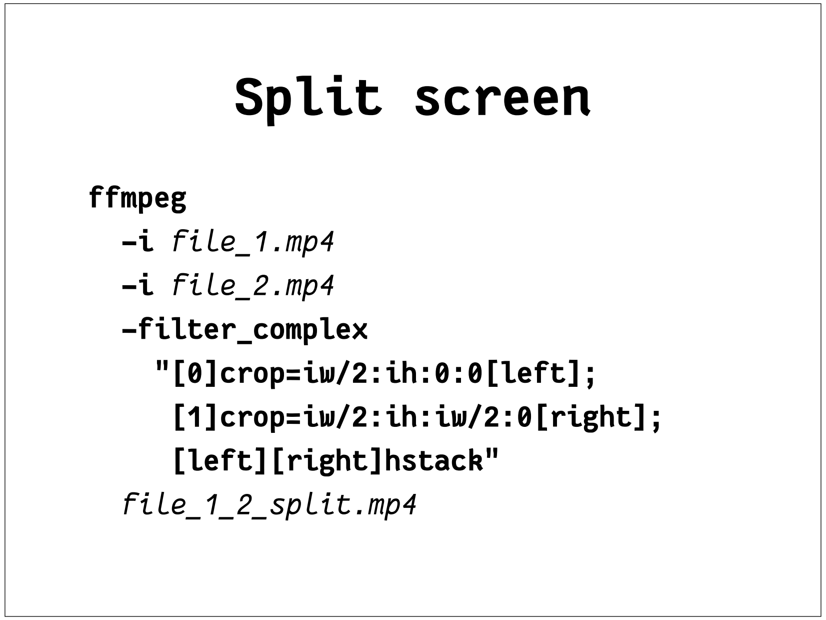
image and sound content

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



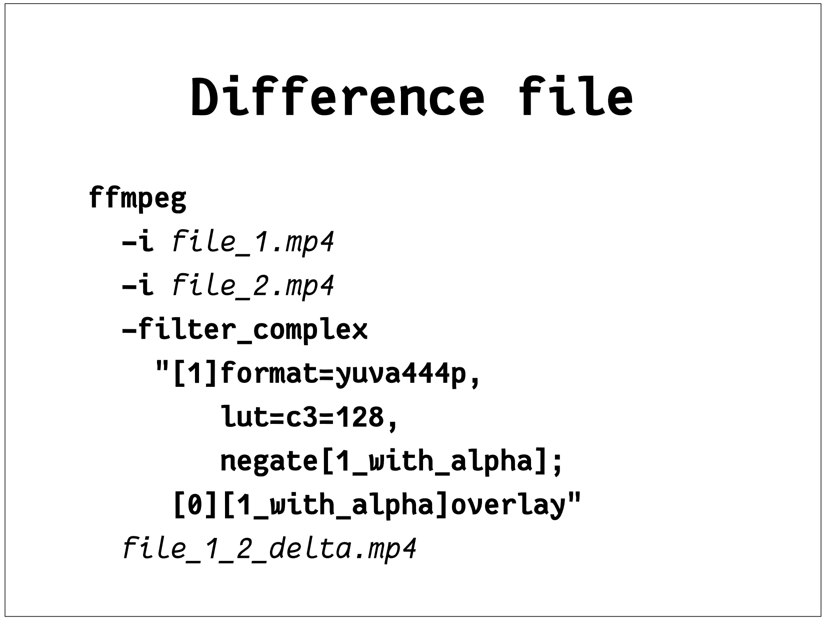
# Archival Tools

- RAWcooked
- BagIt



# Split screen

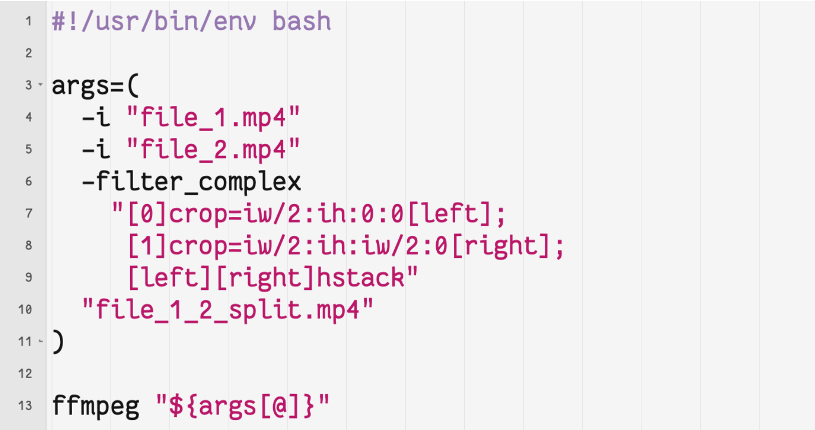
```
ffmpeg
  -i file_1.mp4
  -i file_2.mp4
  -filter_complex
    "[0]crop=iw/2:ih:0:0[left];
     [1]crop=iw/2:ih:iw/2:0[right];
     [left][right]hstack"
  file_1_2_split.mp4
```



# Difference file

```
ffmpeg
  -i file_1.mp4
  -i file_2.mp4
  -filter_complex
    "[1]format=yuva444p,
        lut=c3=128,
        negate[1_with_alpha];
     [0][1_with_alpha]overlay"
  file_1_2_delta.mp4
```

## Integrate FFmpeg commands in your own scripts (e.g. in Bash)



```
#!/usr/bin/env bash

args=(
  -i "file_1.mp4"
  -i "file_2.mp4"
  -filter_complex
    "[0]crop=iw/2:ih:0:0[left];
     [1]crop=iw/2:ih:iw/2:0[right];
     [left][right]hstack"
  "file_1_2_split.mp4"
)

ffmpeg "${args[@]}"
```

## Default Values

three layers increase universality

- passed by the user at execution
- defined by the user in a configuration file
- default values coded in the script

## 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"

**AV Preservation by**
**reto.ch**

Sandrainstrasse 3
3007 Bern
Switzerland

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

cc BY