

Work Smarter, Not Harder

Automating Repetitive Tasks in the Audio Pipeline

Introduction

Introduction: Who am I?



Introduction: Who am I?

Adam Kay



Introduction: Who am I?

Adam Kay

Audio Designer - Paragon Studios



Introduction: Who am I?

Adam Kay

Audio Designer - Paragon Studios



Introduction: What is this all about?



Introduction: What is this all about?

Let your computer work for you



Introduction: What is this all about?

Let your computer work for you

Spend less time clicking, more time being creative



Introduction: What is this all about?

Let your computer work for you

Spend less time clicking, more time being creative

Translation:



Introduction: What is this all about?

Let your computer work for you

Spend less time clicking, more time being creative

Translation:



Introduction: What is this all about?

Let your computer work for you

Spend less time clicking, more time being creative

Translation:

Introduction: What is this all about?

Let your computer work for you

Spend less time clicking, more time being creative

Translation:



...or creating flashy slideshows

Introduction: Why is this important?



Introduction: Why is this important?

Save time



A problem has been detected and windows has been shut down to prevent damage to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

*** gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory

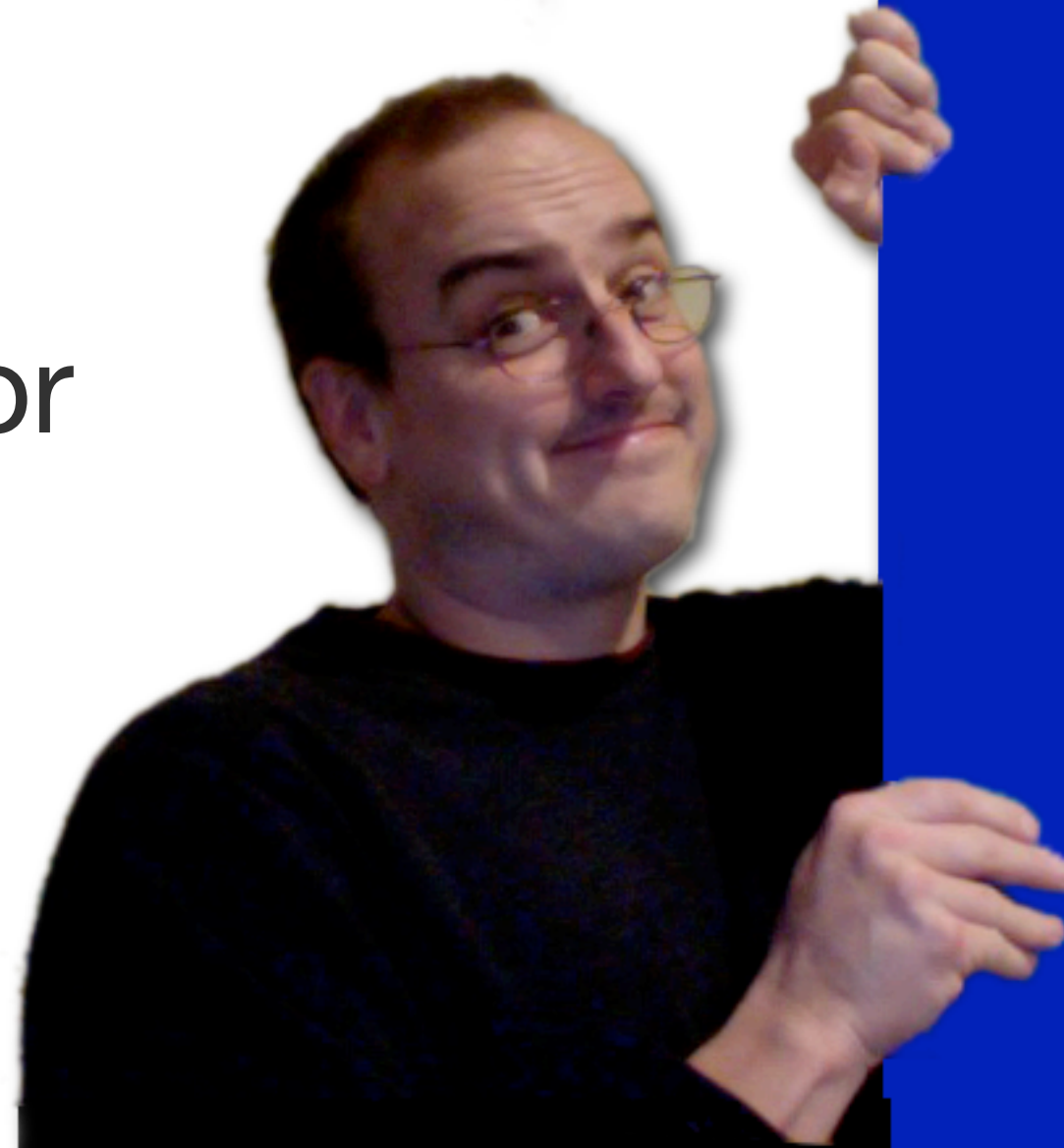
Physical memory dump complete.

Contact your system administrator or technical support group for further assistance.

Introduction: Why is this important?

Save time

Reduce error



A problem has been detected and windows
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen
restart your computer, If this screen
these steps:

Check to make sure any new hardware o
If this is a new installation, ask yo
for any windows updates you might nee

If problems continue, disable or remo
or software. Disable BIOS memory opti
If you need to use Safe Mode to remov
your computer, press F8 to select Adv
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00

*** gv3.sys - Address F86B5A89

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or
assistance.

Introduction: Why is this important?

Save time

Reduce error

Ensure consistency



Introduction: What are we going to do?



Introduction: What are we going to do?

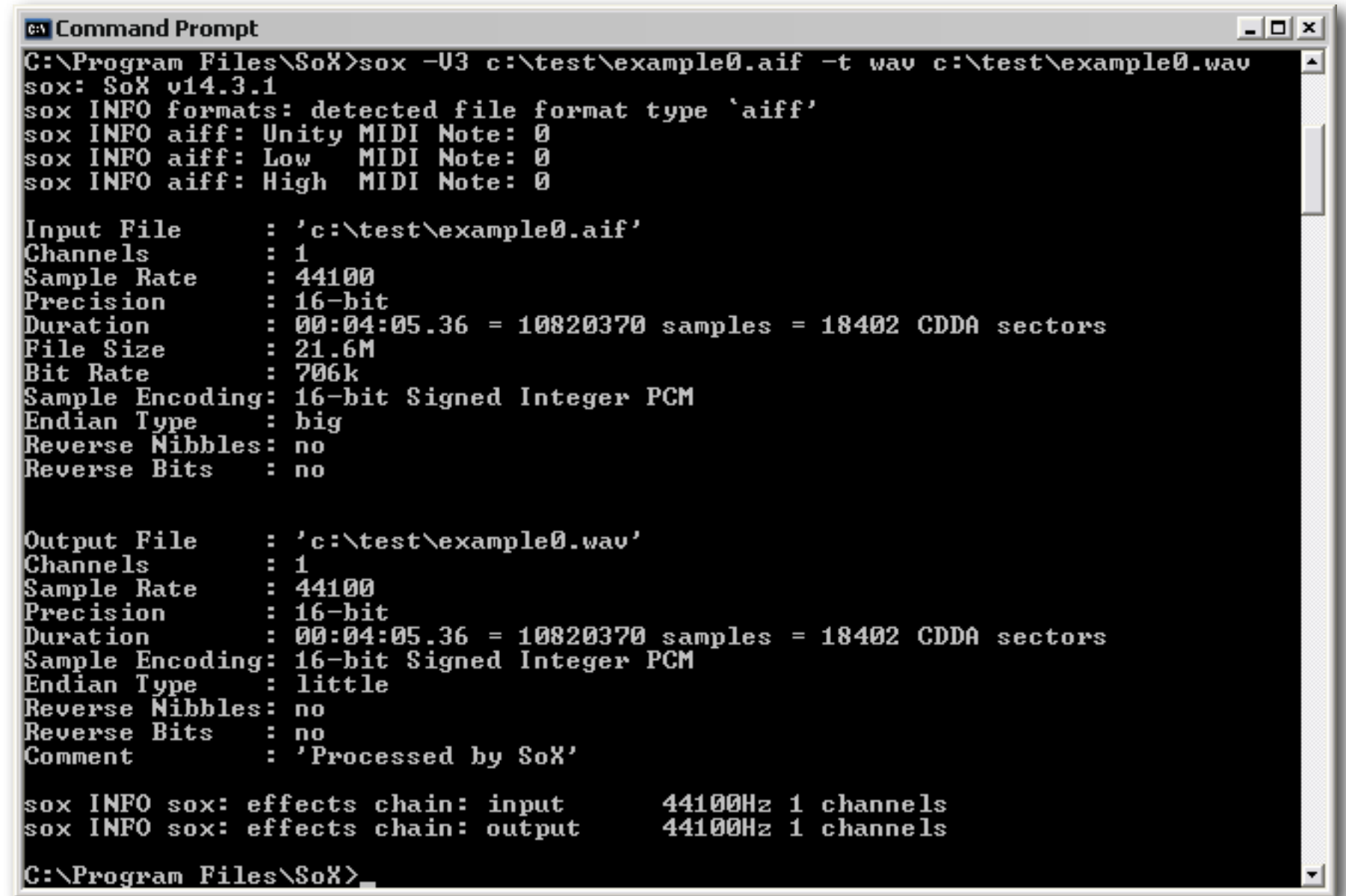


Batch Audio Processor

Introduction: What are we going to do?

Batch Audio Processor

SoX (Sound eXchange)



```
C:\Program Files\SoX>sox -U3 c:\test\example0.aif -t wav c:\test\example0.wav
sox: SoX v14.3.1
sox INFO formats: detected file format type 'aiff'
sox INFO aiff: Unity MIDI Note: 0
sox INFO aiff: Low MIDI Note: 0
sox INFO aiff: High MIDI Note: 0

Input File      : 'c:\test\example0.aif'
Channels       : 1
Sample Rate    : 44100
Precision      : 16-bit
Duration       : 00:04:05.36 = 10820370 samples = 18402 CDDA sectors
File Size      : 21.6M
Bit Rate       : 706k
Sample Encoding: 16-bit Signed Integer PCM
Endian Type    : big
Reverse Nibbles: no
Reverse Bits   : no

Output File     : 'c:\test\example0.wav'
Channels        : 1
Sample Rate     : 44100
Precision       : 16-bit
Duration        : 00:04:05.36 = 10820370 samples = 18402 CDDA sectors
Sample Encoding: 16-bit Signed Integer PCM
Endian Type     : little
Reverse Nibbles: no
Reverse Bits    : no
Comment        : 'Processed by SoX'

sox INFO sox: effects chain: input      44100Hz 1 channels
sox INFO sox: effects chain: output    44100Hz 1 channels

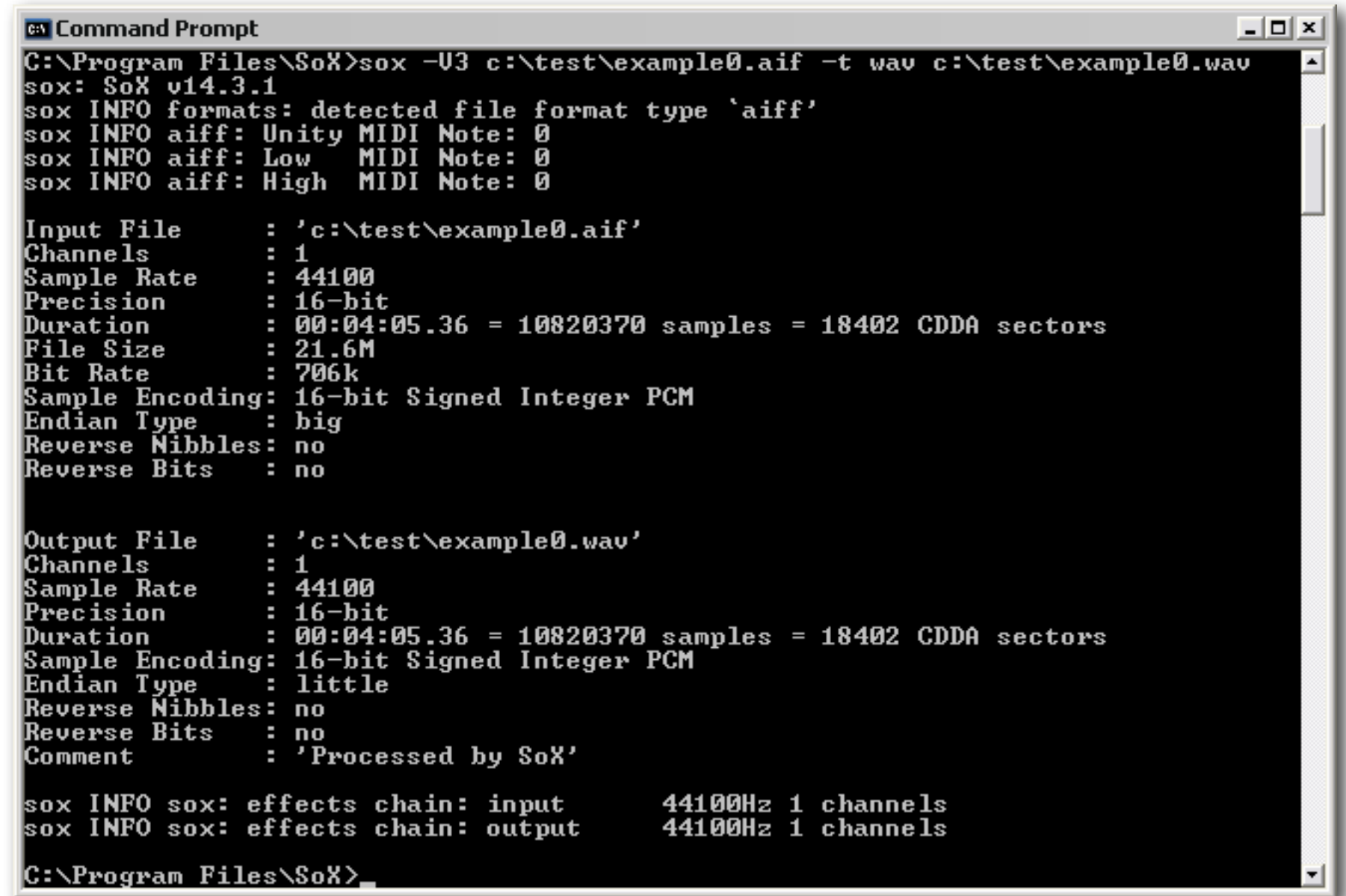
C:\Program Files\SoX>
```

Introduction: What are we going to do?

Batch Audio Processor

SoX (Sound eXchange)

Batch Video Converter



```
C:\Program Files\SoX>sox -U3 c:\test\example0.aif -t wav c:\test\example0.wav
sox: SoX v14.3.1
sox INFO formats: detected file format type 'aiff'
sox INFO aiff: Unity MIDI Note: 0
sox INFO aiff: Low MIDI Note: 0
sox INFO aiff: High MIDI Note: 0

Input File      : 'c:\test\example0.aif'
Channels        : 1
Sample Rate     : 44100
Precision       : 16-bit
Duration        : 00:04:05.36 = 10820370 samples = 18402 CDDA sectors
File Size       : 21.6M
Bit Rate        : 706k
Sample Encoding : 16-bit Signed Integer PCM
Endian Type     : big
Reverse Nibbles : no
Reverse Bits    : no

Output File     : 'c:\test\example0.wav'
Channels        : 1
Sample Rate     : 44100
Precision       : 16-bit
Duration        : 00:04:05.36 = 10820370 samples = 18402 CDDA sectors
Sample Encoding : 16-bit Signed Integer PCM
Endian Type     : little
Reverse Nibbles : no
Reverse Bits    : no
Comment         : 'Processed by SoX'

sox INFO sox: effects chain: input      44100Hz 1 channels
sox INFO sox: effects chain: output    44100Hz 1 channels

C:\Program Files\SoX>
```

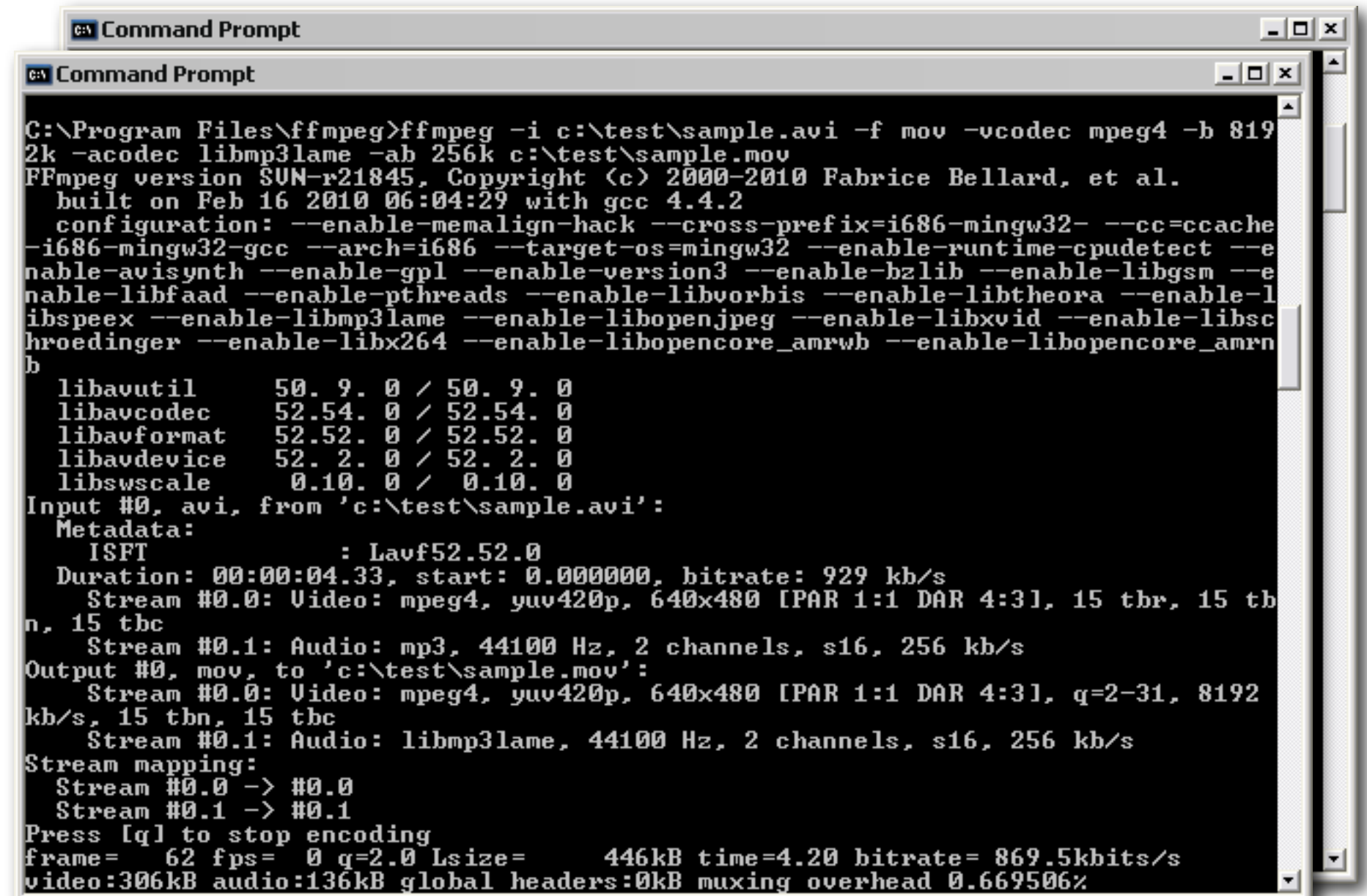
Introduction: What are we going to do?

Batch Audio Processor

SoX (Sound eXchange)

Batch Video Converter

FFmpeg



```
C:\Program Files\ffmpeg>ffmpeg -i c:\test\sample.avi -f mov -vcodec mpeg4 -b 8192k -acodec libmp3lame -ab 256k c:\test\sample.mov
FFmpeg version SUN-r21845, Copyright (c) 2000-2010 Fabrice Bellard, et al.
  built on Feb 16 2010 06:04:29 with gcc 4.4.2
  configuration: --enable-memalign-hack --cross-prefix=i686-mingw32- --cc=ccache
-i686-mingw32-gcc --arch=i686 --target-os=mingw32 --enable-runtime-cpudetect --e
nable-avisynth --enable-gpl --enable-version3 --enable-bzlib --enable-libgsm --e
nable-libfaad --enable-pthreads --enable-libvorbis --enable-libtheora --enable-l
ibspeex --enable-libmp3lame --enable-libopenjpeg --enable-libxvid --enable-libsc
hroedinger --enable-libx264 --enable-libopencore_amrwb --enable-libopencore_amrn
b
  libavutil      50. 9. 0 / 50. 9. 0
  libavcodec     52.54. 0 / 52.54. 0
  libavformat    52.52. 0 / 52.52. 0
  libavdevice    52. 2. 0 / 52. 2. 0
  libswscale     0.10. 0 / 0.10. 0
Input #0, avi, from 'c:\test\sample.avi':
Metadata:
  ISFT           : Lavf52.52.0
  Duration: 00:00:04.33, start: 0.000000, bitrate: 929 kb/s
  Stream #0.0: Video: mpeg4, yuv420p, 640x480 [PAR 1:1 DAR 4:3], 15 tbr, 15 tbn, 15 tbc
  Stream #0.1: Audio: mp3, 44100 Hz, 2 channels, s16, 256 kb/s
Output #0, mov, to 'c:\test\sample.mov':
  Stream #0.0: Video: mpeg4, yuv420p, 640x480 [PAR 1:1 DAR 4:3], q=2-31, 8192 kb/s, 15 tbn, 15 tbc
  Stream #0.1: Audio: libmp3lame, 44100 Hz, 2 channels, s16, 256 kb/s
Stream mapping:
  Stream #0.0 -> #0.0
  Stream #0.1 -> #0.1
Press [q] to stop encoding
frame= 62 fps= 0 q=2.0 Lsize=      446kB time=4.20 bitrate= 869.5kbits/s
video:306kB audio:136kB global headers:0kB muxing overhead 0.669506%
```

Introduction: What are we going to do?

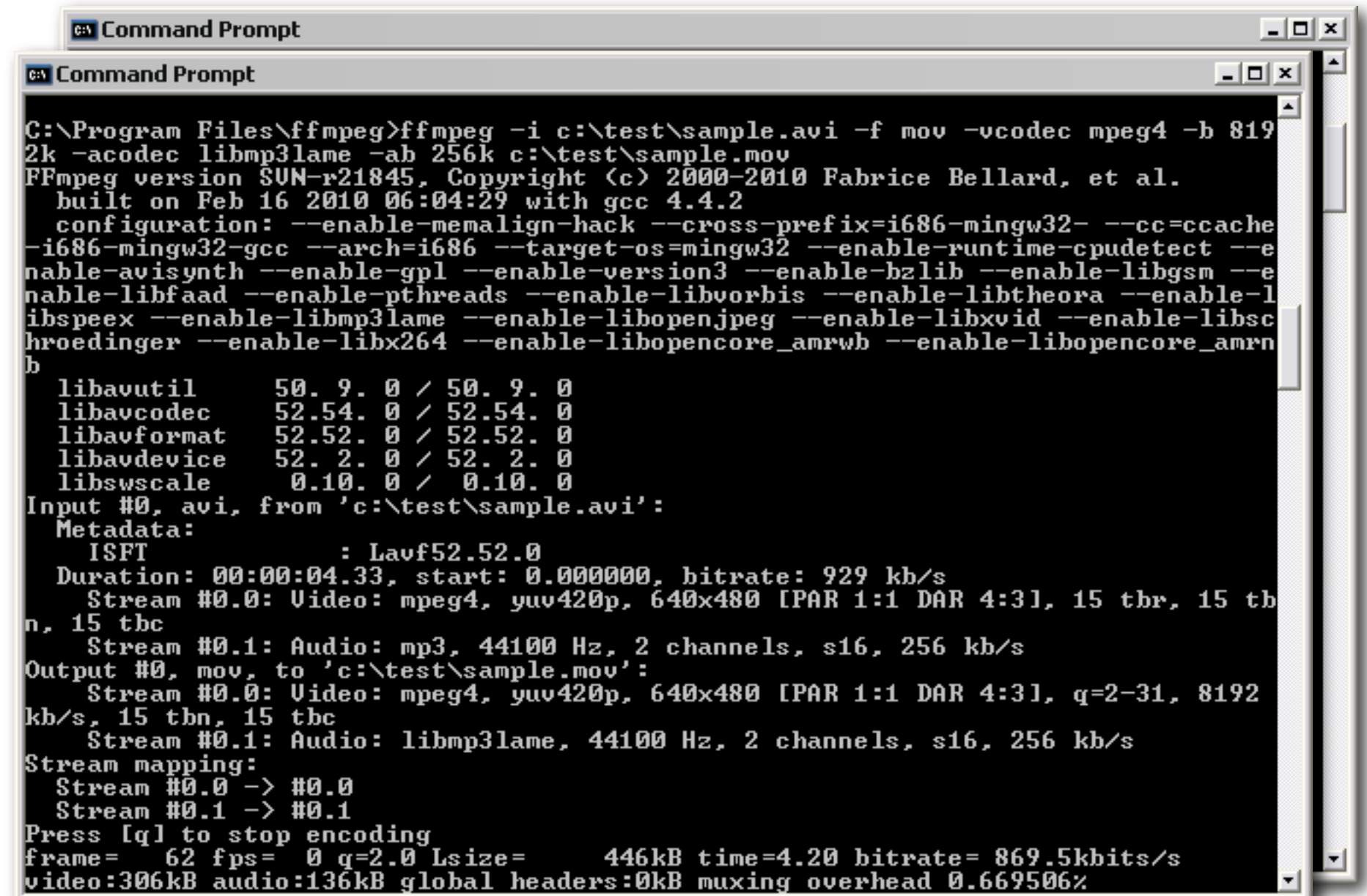
Batch Audio Processor

SoX (Sound eXchange)

Batch Video Converter

FFmpeg

Batch text manipulation



```
C:\Program Files\ffmpeg>ffmpeg -i c:\test\sample.avi -f mov -vcodec mpeg4 -b 8192k -acodec libmp3lame -ab 256k c:\test\sample.mov
FFmpeg version SUN-r21845, Copyright (c) 2000-2010 Fabrice Bellard, et al.
  built on Feb 16 2010 06:04:29 with gcc 4.4.2
  configuration: --enable-memalign-hack --cross-prefix=i686-mingw32- --cc=ccache
-i686-mingw32-gcc --arch=i686 --target-os=mingw32 --enable-runtime-cpudetect --e
nable-avisynth --enable-gpl --enable-version3 --enable-bzlib --enable-libgsm --e
nable-libfaad --enable-pthreads --enable-libvorbis --enable-libtheora --enable-l
ibspeex --enable-libmp3lame --enable-libopenjpeg --enable-libxvid --enable-libsc
hroedinger --enable-libx264 --enable-libopencore_amrwb --enable-libopencore_amrn
b
  libavutil      50. 9. 0 / 50. 9. 0
  libavcodec     52.54. 0 / 52.54. 0
  libavformat    52.52. 0 / 52.52. 0
  libavdevice    52. 2. 0 / 52. 2. 0
  libswscale     0.10. 0 / 0.10. 0
Input #0, avi, from 'c:\test\sample.avi':
  Metadata:
    ISFT           : Lavf52.52.0
  Duration: 00:00:04.33, start: 0.000000, bitrate: 929 kb/s
    Stream #0.0: Video: mpeg4, yuv420p, 640x480 [PAR 1:1 DAR 4:3], 15 tbr, 15 tbn, 15 tbc
    Stream #0.1: Audio: mp3, 44100 Hz, 2 channels, s16, 256 kb/s
Output #0, mov, to 'c:\test\sample.mov':
  Stream #0.0: Video: mpeg4, yuv420p, 640x480 [PAR 1:1 DAR 4:3], q=2-31, 8192 kb/s, 15 tbn, 15 tbc
    Stream #0.1: Audio: libmp3lame, 44100 Hz, 2 channels, s16, 256 kb/s
Stream mapping:
  Stream #0.0 -> #0.0
  Stream #0.1 -> #0.1
Press [q] to stop encoding
frame= 62 fps= 0 q=2.0 Lsize=      446kB time=4.20 bitrate= 869.5kbits/s
video:306kB audio:136kB global headers:0kB muxing overhead 0.669506%
```


Introduction: What are we going to do?

Batch Audio Processor

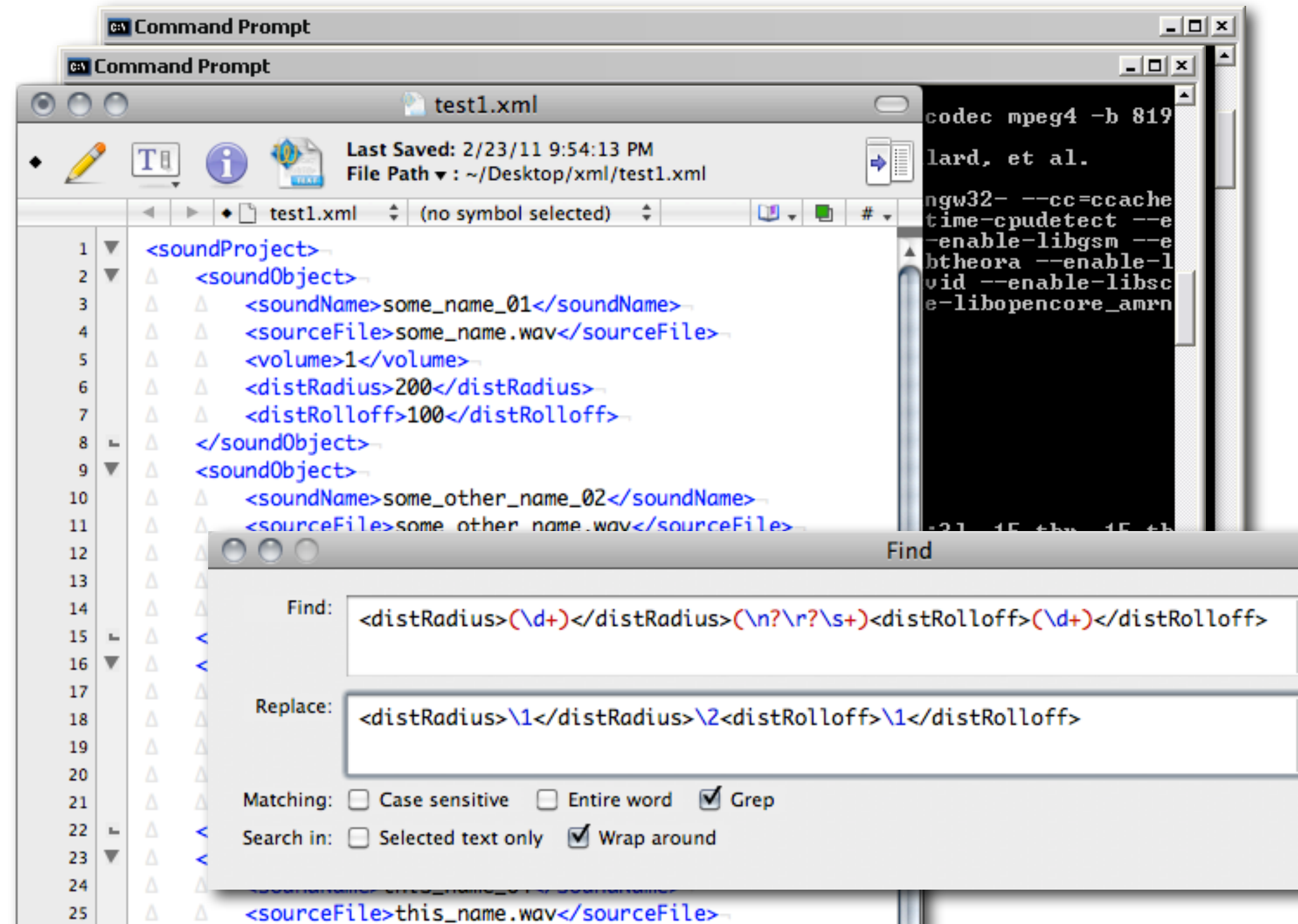
SoX (Sound eXchange)

Batch Video Converter

FFmpeg

Batch text manipulation

regular expressions

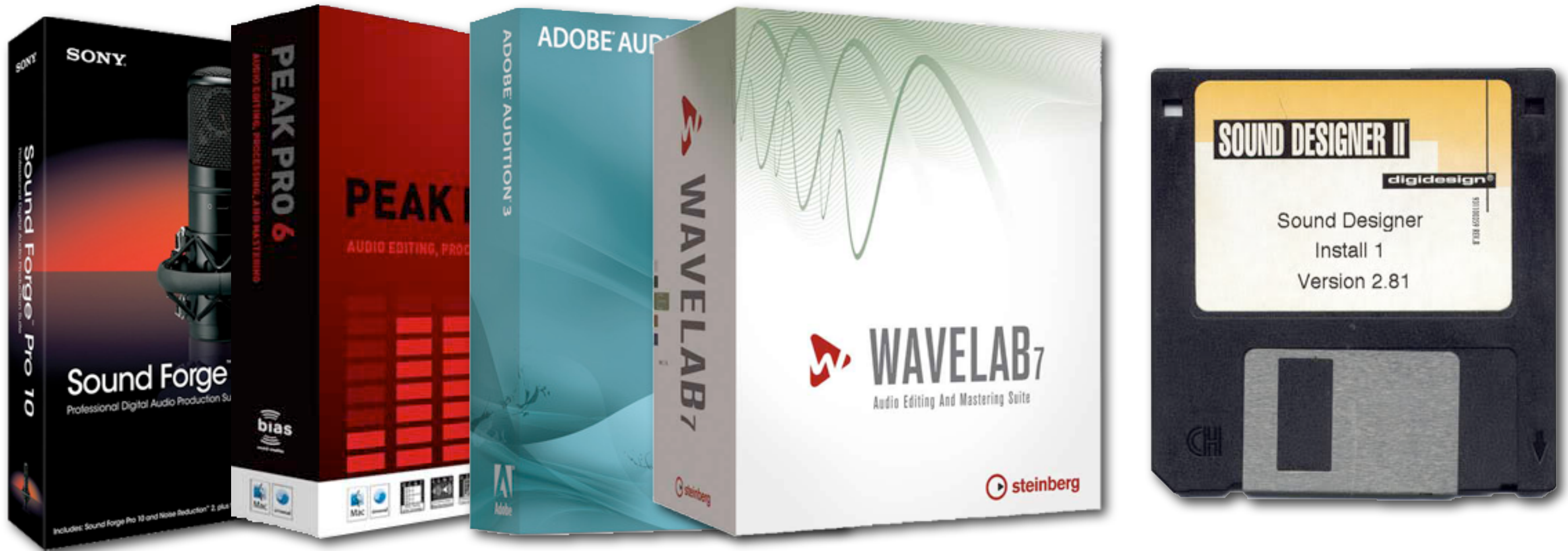


Introduction: Really, Why?



Introduction: Really, Why?

...but what about Sound Forge, Peak, Audition, WaveLab, Sound Designer II?



Introduction: Really, Why?

...but what about Sound Forge, Peak, Audition, WaveLab, Sound Designer II?



Introduction: More about me...



Introduction: More about me...

I am NOT a programmer

[illegible]

Introduction: More about me...

I am NOT a programmer

I'm just an audio guy who doesn't like repetition



Introduction: More about me...

I am NOT a programmer

I'm just an audio guy who doesn't like repetition

I'm just an audio guy who doesn't like repetition



Introduction: More about me...

I am NOT a programmer

I'm just an audio guy who doesn't like repetition

I'm just an audio guy who doesn't like repetition

What's my point?



Introduction: More about me...

I am NOT a programmer

I'm just an audio guy who doesn't like repetition

I'm just an audio guy who doesn't like repetition

What's my point?

You don't have to remember commands

?



Introduction: More about me...

I am NOT a programmer

I'm just an audio guy who doesn't like repetition

I'm just an audio guy who doesn't like repetition

What's my point?

You don't have to remember commands

Let the scripts remember for you

?



Introduction: More about me...

I am NOT a programmer

I'm just an audio guy who doesn't like repetition

I'm just an audio guy who doesn't like repetition

What's my point?

You don't have to remember commands

Let the scripts remember for you

Google is your friend

?



Introduction: Before we proceed...



Introduction: Before we proceed...

I'm going to move quickly

Introduction: Before we proceed...

I'm going to move quickly

Feel free to ask questions



Introduction: Before we proceed...

I'm going to move quickly

Feel free to ask questions

Extensive documentation and examples will be available



Introduction: Before we proceed...



I'm going to move quickly

Feel free to ask questions

Extensive documentation and examples will be available

NOT the talk about judicious use of VFX in PowerPoint



Examples

Examples: Scripting



Examples: Scripting



Batch Audio Converter

Batch Video Converter

Live Example

Examples: Regular Expressions



Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

2. How many of this type of character are you looking for?

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

- . = any character

2. How many of this type of character are you looking for?

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

- . = any character

- \d = any numeric character

2. How many of this type of character are you looking for?

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

- `.` = any character

- `\d` = any numeric character

- `[a-z]` = any alpha character

2. How many of this type of character are you looking for?

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

- . = any character

- \d = any numeric character

- [a-z] = any alpha character

2. How many of this type of character are you looking for?

- ? = 0 or 1

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1

+ = 1 or more

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1

+ = 1 or more

* = 0 or more

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 pin\d?hot

+ = 1 or more

* = 0 or more

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` *matches* `pin2hot`

+ = 1 or more

* = 0 or more

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` *matches* `pin2hot` *or* `pinhot`

+ = 1 or more

* = 0 or more

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` *matches* `pin2hot` *or* `pinhot` *not* `pin12hot`

+ = 1 or more

* = 0 or more

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` *matches* `pin2hot` *or* `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot`

* = 0 or more

Examples: Regular Expressions

Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` matches `pin2hot` or `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot` matches `pin3hot`

* = 0 or more

Examples: Regular Expressions

Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` matches `pin2hot` or `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot` matches `pin3hot` or `pin32hot`

* = 0 or more

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` matches `pin2hot` or `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot` matches `pin3hot` or `pin32hot` *not* `pinhot`

* = 0 or more

Examples: Regular Expressions

Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` matches `pin2hot` or `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot` matches `pin3hot` or `pin32hot` *not* `pinhot`

* = 0 or more `pin\d*hot`

Examples: Regular Expressions



Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` matches `pin2hot` or `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot` matches `pin3hot` or `pin32hot` *not* `pinhot`

* = 0 or more `pin\d*hot` matches `pinhot`

Examples: Regular Expressions

Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` matches `pin2hot` or `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot` matches `pin3hot` or `pin32hot` *not* `pinhot`

* = 0 or more `pin\d*hot` matches `pinhot` or `pin1hot`

Examples: Regular Expressions

Regex uses succinct (and inherently cryptic) keys to identify patterns within text

On a basic level, Regex expects two types of arguments

1. What kind of character you are looking for?

. = any character

\d = any numeric character

[a-z] = any alpha character

2. How many of this type of character are you looking for?

? = 0 or 1 `pin\d?hot` matches `pin2hot` or `pinhot` *not* `pin12hot`

+ = 1 or more `pin\d+hot` matches `pin3hot` or `pin32hot` *not* `pinhot`

* = 0 or more `pin\d*hot` matches `pinhot` or `pin1hot` or `pin9999hot`

Live Example

Conclusion

Conclusion: What do I do now?



Conclusion: What do I do now?

Build your own player/converter

Conclusion: What do I do now?



Build your own player/converter

Look for other ways to use batch scripts

Conclusion: What do I do now?



Build your own player/converter

Look for other ways to use batch scripts

Next time, use regular expressions

Links:



Links:

yakmatter.com/worksmarter

Links: yakmatter.com/worksmarter



SoX

<http://sox.sourceforge.net/>

FFmpeg

<http://ffmpeg.org/>

Regular Expressions

<http://perldoc.perl.org/perlre.html>

Links:



SoX

<http://sox.sourceforge.net/>

FFmpeg

<http://ffmpeg.org/>

Regular Expressions

<http://perldoc.perl.org/perlre.html>

Please fill out the evaluation form

yakmatter.com/worksmarter