

# Seventh Session

Joshua Ng • Archives New Zealand  
Reto Kromer • AV Preservation by reto.ch

**Bash Scripting for Audio-Visual Preservation**  
webinar series hosted by AMIA  
15 February 2022

1

# Advanced level's goals

- to program and debug complex Bash scripts
- to understand and write basic regular expressions ("regex")

2

# "Data maintenance"

- debugging techniques
- flow control: looping with **for**
- regular expressions
- command substitution
- flow control: branching with **select**
- asynchronous execution

3

# Contents

- built-in debugging tool
- external debugging tools
- good practices

4

# Introduction

5

## Symptoms of buggy scripts

### **syntax error**

- script doesn't run

### **logic error**

- script runs, but doesn't work as expected

### **logic bomb**

- script runs and works as expected, but has nasty side effects

6

## Trace variables and flow

- use **echo** or **printf** commands at critical points to trace the content of the variables
- use **tee** command at critical points to trace the flow

7

# Built-in debugging

8

## Run in debugging mode (1)

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

```
#!/usr/bin/env bash -x -v
```

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

9

## Run in debugging mode (2)

```
bash -x script
```

```
bash -xv script
```

```
bash -xvn script
```

```
bash -vn script
```

10

## Debug a section

```
# start debugging
```

```
set -x
```

```
code_to_debug
```

```
# stop debugging
```

```
set +x
```

11

## Debugging options (1)

```
set -x
```

```
set -o xtrace
```

- print command traces before executing command

12

## Debugging options (2)

**set -v**

**set -o verbose**

- print shell input lines as they are read

13

## Debugging options (3)

**set -n**

**set -o noexec**

- check for syntax errors without actually running the script

14

## Debugging options (4)

**set -f**

**set -o noglob**

- disable file name generation using meta-characters ("globbing")

15

## Debugging options (5)

**set -u**

**set -o nounset**

- the script runs, but gives an error message and aborts if there are unbound variables

16

## Change the tracing prompt

```
PS4=' [${BASH_SOURCE}:${LINENO}]+ '
```

```
set -x # start debugging
```

```
code_to_debug
```

```
set +x # stop debugging
```

17

## Debug output to a separate file

```
exec 5> debug_output.txt
```

```
BASH_XTRACEFD='5'
```

18

## All together

```
PS4=' [${BASH_SOURCE}:${LINENO}]+ '
```

```
exec 5> debug_output.txt
```

```
BASH_XTRACEFD='5'
```

```
bash -x script
```

19

## Use **trap** with **DEBUG**

```
trap 'read -p
```

```
"[${BASH_SOURCE}:${LINENO}]
```

```
${BASH_COMMAND}"' DEBUG
```

20

# Debugging tools

21

## Bash debugger script

```
bashdb [options] script [--]  
[script_options]
```

```
bashdb [options] -c command_string
```

```
bash --debugger [bash_options]  
script [--] script_options]
```

22

## Text editor

- syntax highlighting
- syntax verification

“A hackable text editor for the 21<sup>st</sup> Century”

→ <https://atom.io>

23

# Good practices

24

## Programming style

- comment the code
- use meaningful names
- structure the code in logical units
- use indentation
- be consistent

25

## Value initialisation

- value passed:
  - at program call
  - interactively when running
- value read from a configuration file
- default value coded in the program

26

## User interface

- always provide a built-in help (-h, --help)
- if you distribute complex scripts, provide a documentation, such as a manual page ("man")
- create temporary log files when useful

27

AV Preservation by  
**reto.ch**

zone industrielle Le Trési 3  
1028 Préverenges  
Switzerland

Web: [reto.ch](http://reto.ch)  
Twitter: [@retoch](https://twitter.com/retoch)  
Email: [info@reto.ch](mailto:info@reto.ch)



28