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

Advanced level's goals

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

"Data maintenance"

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

Contents

Introduction

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

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

Built-in debugging

Run in debugging mode (1)

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

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


Run in debugging mode (2)

```
bash -x script

bash -xv script
bash -xvn script
bash -vn script
```


Debug a section

```
# start debugging
set -x
code_to_debug
# stop debugging
set +x
```


Debugging options (1)

```
set -x
set -o xtrace
```

- print command traces before executing command

Debugging options (2)

```
set -v
set -o verbose
```

- print shell input lines as they are read

Debugging options (3)

```
set -n
set -o noexec
```

- check for syntax errors without actually running the script

Debugging options (4)

```
set -f
set -o noglob
```

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

Debugging options (5)

```
set -u
set -o nounset
```

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

Change the tracing prompt

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

set -x  # start debugging
code_to_debug
set +x  # stop debugging
```


Debug output to a separate file

```
exec 5> debug_output.txt
BASH_XTRACEFD='5'
```


All together

```
PS4='[${BASH_SOURCE}:${LINENO}]+ '
exec 5> debug_output.txt
BASH_XTRACEFD='5'

bash -x script
```


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

```
trap 'read -p
"[${BASH_SOURCE}:${LINENO}]
${BASH_COMMAND}"' DEBUG
```

Debugging tools

Bash debugger script

bashdb [*options*] *script* [**--**] [*script_options*]

bashdb [*options*] **-c** *command_string*

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

Text editor

- syntax highlighting
- syntax verification

"A hackable text editor for the 21st Century"

➔ https://atom.io

Good practices

Programming style

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

Value initialisation

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

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

AV Preservation by
reto.ch

zone industrielle Le Trési 3
1028 Préverenges
Switzerland

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