Session 5: `bash` Scripting and Getting Help

P. S. Langeslag

15 November 2018

Getting Help: Manuals and Tutorials

- For Linux tools:
  - `man` for "comprehensive brevity"
  - `info` for fuller information (uses hyperlinks)
  - `apropos` to search `man` program names and descriptions

Getting Help: Manuals and Tutorials

- For Linux tools:
  - `man` for "comprehensive brevity"
  - `info` for fuller information (uses hyperlinks)
  - `apropos` to search `man` program names and descriptions
- For Vim: `:help`, or various online tutorials
- For LaTeX: https://en.wikibooks.org/wiki/LaTeX
- For Markdown: http://pandoc.org/MANUAL.html
- For regular expressions: e.g. https://regular-expressions.info

Getting Help With Specific Questions

1. Web search is your friend
2. Specialized resources
   - https://stackexchange.com
     - https://tex.stackexchange.com
     - https://unix.stackexchange.com
     - https://stackoverflow.com
   - https://askubuntu.com
   - https://ubuntuforums.org

Strategy (1) will typically lead you to the best resources in (2).

Why Forums Don't Work

- Chatty and informal
- Responses sorted chronologically rather than by relevance
- No distinction between answers and comments
- No penalty for useless responses
- Paginated content; no way of highlighting the most useful response

Stack Exchange: Some Examples

- In the shell, what does `2>&1` mean?
- How to check if a string contains a substring in `bash`
- Getting the source directory of a `bash` script from within
- When should I use `\input` vs. `\include`?
- Set margin size when converting from Markdown to PDF with `pandoc`

The Stack Exchange Model

Your Question Must Not

- Contain formalities
- Contain irrelevant information
- Contain privacy-sensitive data (passwords)
- Duplicate existing questions
- Ask for the solution to a homework assignment

The Stack Exchange Model

Your Question Must Not

- Contain formalities
- Contain irrelevant information
- Contain privacy-sensitive data (passwords)
- Duplicate existing questions
- Ask for the solution to a homework assignment

Your Question Must Contain

- A concise but informative title (mention the framework)
- A full and precise description of the problem
- A **minimal working example (MWE)**
  - Insert using Markdown convention, i.e. 4 spaces
- Mention of the exact tools and versions used
- Relevant tags, e.g. `latex`, `biblatex`, `pandoc`, `vim`, `regex`, `tmux`

Your question is intended for everyone's future reference!

Stack Exchange: Comments

- To communicate about
  - The conditions of the problem
  - The post author's objectives
- No chitchat, "thanks," or "I agree"

Stack Exchange: Accepting Answers

- Pick the answer that worked for you as the official answer.
- You can change official answers if a better one comes along.
- If you find a satisfactory solution in a comment, ask the commenter to write it up as an answer so you can accept it.
- You can answer your own questions.

Stack Overflow: Privileges (Selective)

Privilege	Reputation
Ask a question	1
Answer a question	1
Edit your own questions and answers	1
Upvote	15
Comment	50
Downvote	125
Edit without peer review	2,000

Stack Overflow: Gaining and Losing Reputation

Your question upvoted	+5
Your answer upvoted	+10
Your answer accepted	+15
Edit accepted	+2
Bounty won	`+$bounty`
Maximum gain per day	+200

Your question downvoted	-2
Your answer downvoted	-2
You downvoted an answer	-1
Bounty placed	`-$bounty`
6 spam/offensive flags on one post	-100

Stack Exchange: Some Examples

- In the shell, what does `2>&1` mean?
- How to check if a string contains a substring in `bash`
- Getting the source directory of a `bash` script from within
- When should I use `\input` vs. `\include`?
- Set margin size when converting from Markdown to PDF with `pandoc`

Larger Resources Are Better than Smaller Resources

Doing a web search is often better than searching within a single resource.

- https://unix.stackexchange.com/questions/19451/difference-between-help-info-and-man-command
- https://askubuntu.com/questions/9325/what-is-the-difference-between-man-and-info-documentation

A Simple Shell-Agnostic Script

```
date

cal

uname -a

ping gwdg.de -c 3
```

- Can be just a sequence of commands that will run in all shells;
- In that case you need not identify the shell interpreter.
- Make your file executable: `chmod +x filename`
- If its location is not in `$PATH`, run with leading path: `./filename`

A Typical bash Script

```
#!/bin/bash

dock=$(ip link | grep dock | awk '{print $9}')
dongle=$(ip link | grep eth0 | awk '{print $9}')
if [ "$dock" = "UP" ] || [ "$dongle" = "UP" ] ; then
    echo " ð "
else
    echo " "
fi
```

- bash scripting relies on tests, variables, and logical operators;
- These require you to define your shell interpreter.

Text Variables

```
#!/bin/bash

string="Hello World!"
echo $string              # A hash comments out the rest
                          # of the line.
```

(The functional equivalent of just `echo "Hello world!"`)

- Set a text variable as `name="text"`, no spaces and no `$`
- Call it as `$name`, e.g. `echo $name`

Command Substitution

- Allows the output of a command to replace the command itself, so it can be processed more straightforwardly.
- Enclose the command in `$( … )` and assign it to a variable.

Command Substitution

- Allows the output of a command to replace the command itself, so it can be processed more straightforwardly.
- Enclose the command in `$( … )` and assign it to a variable.

```
#!/bin/bash

state=$(cat /sys/class/power_supply/AC/online)
echo $state
```

(The functional equivalent of just

```
echo $(cat /sys/class/power_supply/AC/online).)
```

Integer Arithmetic: `let`

- No spaces unless you use quotes

```
#!/bin/bash

let addition=2+2
let subtraction=2-2
let multiplication="2 * 2"
let modulus="5 % 2"
x=10
y=2
let division=$x/$y
echo $addition          # etc.
```

Integer Arithmetic: Arithmetic Expansion

- Spaces permitted

```
#!/bin/bash

addition=$((2+2))
subtraction=$(( 2 - 2 ))
echo $(( 2 * 2 ))
modulus=$(( 5 % 2 ))
x=10
y=2
division=$(( $x / $y ))
echo $addition          # etc.
```

Again, you can also just use `echo $((4*25))`.

A Simple Integer Arithmetic Script

```
#!/bin/bash

read -p "Enter two numbers separated by a space: " x y
echo "$x + $y = $(( x + y ))"
```

Floating Point Arithmetic

`bash` itself can only handle integers. Floating-point solutions include:

`bc`	basic calculator
`calc`	has interactive mode
`awk`	a scripting language
`python`	a programming language

A Simple Floating-Point Script

```
#!/bin/bash

read -p "Enter any number: " x
root=$(echo "sqrt($x)" | bc -l )
echo "The square root of $x is $root."
```

Conditionals

```
#!/bin/bash

second=$(date +%-S)
if [ $(( $second % 2 )) -eq 0 ]
then
    echo "$second is an even number of seconds!"
else
    echo "$second is an odd number of seconds!"
fi
```

- Simple conditionals are signalled with `if` and closed with `fi`
- The condition is contained in square brackets
- Square brackets **must** be spaced: they are a program (`test`)!
- Double brackets are an improved implementation (not POSIX)
- Action to be undertaken is introduced by `then` or `else`

Conditionals

Long Form

```
#!/bin/bash

if [ "$(whoami)" != 'root' ]; then
        echo "Permission denied."
        exit 1;
fi
```

- The semicolon accommodates multiple commands on one line.

Short Form

```
#!/bin/bash

[ "$(whoami)" != 'root' ] && ( echo "Denied"; exit 1 )
```

- `&&` for “if successful”; `||` “if unsuccessful.”
- `exit 0` reports success; `exit 1` error; `exit 2` fundamental issues.