Cool Shell Scripts

Introduction

About me

- Sander van Vugt
- Living in the Netherlands
- Author and presenter of many titles on this platform – Linux, Kubernetes and Ansible
- Founder of the Living Open Source Foundation
  - The mission of the Living Open Source Foundation is to stimulate the growth of local economies by enabling people to develop themselves as experts in the area of Open Source
  - Current focus is on education in Africa
  - See livingopensource.net for more information

About this Course

- This course is developed to allow you to get more experience in shell scripting
- To do so, we'll work through some scenarios, and next discuss possible solutions
- This course is supposed to be interactive! You'll be presented with a challenge, and after working on it for a few minutes we'll discuss the solution
- This course is NOT an introduction to Bash shell scripting – take my "Bash scripting in 4 Hours" if you need an introduction level course

Course Resources

- Sample scripts used in this course are on https://github.com/sandervanvugt/cool-bash

Poll Question 1

How would you rate your own Bash scripting experience?

- None
- Poor
- Average
- Strong

Poll Question 2

Have you attended my Shell scripting in 4 hours class?

- yes
- no

Poll Question 3

On which OS platform are you planning to use Bash shell scripts

- Linux
- MacOS
- Windows Subsystem for Linux
- UNIX
- Other

Note 1

- Shell scripting sometimes is an art, where many solutions are possible. I don't pretend the solutions presented here are the best solutions, but they will provide a good learning experience
- Suggestions for improvement can be made on all of the sample solutions. Feel free to do so, this is a useful part of this class
- Do you have cool scripts that you want me to present? Send them to me and I will consider them. (Don't forget to leave your name in the script comments so that you get the credentials if I use it)
- Sample solutions are available at https://github.com/sandervanvugt/cool-bash

Note 2

- Bash scripts do have their limits. Human intelligence does not. In the assignments in this course, use human intelligence where necessary to go beyond the limits of Bash shell scripting

Note 3

- This is a new course. Your feedback is much appreciated. Provide your feedback by taking the end-of-course survey, or send it to me directly at mail@sandervanvugt.nl

Cool Shell Scripts

1. Monitoring Process Activity

Assignment: Monitor Process Activity

- Write a script that alerts on high process activity. If a process is generating more than 80% CPU load over a period longer than 7 seconds, the script should send an alert to the root user

Cool Shell Scripts

2. A flexible vi

Assignment: a Flexible vi

- I often confuse **vi** and **cd**, with the result that I'm opening a directory with **vi**, and try to use **cd** to edit a file. This is easy to fix with a script. Write a script that works with one argument and meets the following requirements:
  - If no argument is provided, it should exit with an error message
  - If the argument is a directory, the script should **cd** to it
  - If the argument is a file, the script should open it in **vi** for editing
- Write this script as compact as possible: shorted is better!

Cool Shell Scripts

3. Writing a Menu

Assignment: Writing a Menu

- Bash scripts can be used to write a menu, using select. Write a simple menu that allows the user to select between 3 items. (For instance directory names)

Cool Shell Scripts

4. Rebooting and Continuing

Assignment: Reboot and Continue

- Sometimes, you want a script to reboot and continue after rebooting. Write a script that will do so, and contains at least the following elements:
  - The script should ask the user if it's OK to reboot
  - After rebooting, the script should create a file with the name /tmp/after-reboot
  - If this file already exists before reboot, the script should show an error
  - If this file exists after reboot, the script should congratulate the user for his successful work

Cool Shell Scripts

5. Advanced Pattern Matching

5. Advanced Pattern Matching

- Pattern matching can be used to clean up a string. It works well to remove parts from the beginning or end of a string, but it doesn't do so well in leaving just the middle of a string. Write a script that will do it anyway. It is mandatory this scripts works on a variable that has been set: **DATE=$(date +%d-%m-%Y)**. The script should use pattern matching on this variable and print three lines as its result:
  - Today is …
  - The current month is …
  - The current year is …

Cool Shell Scripts

6. Create a Stresstest

6. Create a stresstest

- Write a script that performs a stress test. It should push your system to its ultimate limits

Cool Shell Scripts

7. Using trap

7. Using trap

- Use trap to run commands on specific behavior caused by signals

Cool Shell Scripts

8. Working with Options

8. Working with Options

- Write a script that is used as a wrapper script around the **useradd** command. It should work with options to create a user account. If no options are provided, it should try to detect relevant options, or become interactive and ask the user to provide information

Cool Shell Scripts

9. Monitoring Critical Processes

9. Monitoring Critical Processes

- Write a script that monitors a critical process. If the process goes down, the script should try to start it again, and at the same time it should send an email message alerting bob@example.com that the process has gone down

Cool Shell Scripts

10. Multiplier Tables

10. Multiplier Tables

- Write a script that allows children to practice their multiplier tables. The script should run until manually interrupted with the Ctrl-C key sequence and allow kids to practice multiplier tables up to 10. If a question was not answered corrently, the same question should be repeated until answered correctly While running the script, it should write a log file, indicating for each answer if it was answered correctly or not