Introduction to Linux Scripting (Part 1)

Anita Orendt, David Heidorn and Wim Cardoen
CHPC User Services

Overview

- Scripting in Linux
  - What is a script?
  - Why scripting?
  - Scripting languages + syntax
  - Bash/tcsh scripting exercises

What is a script?

- A script is a collection of linux commands that:
  - are stored in a file
  - the file **MUST** be executable
  - commands are separated by:
    - either being a carriage return (new line)
    - or separated by the semi colon (“;”)
  - executed sequentially until
    - the end of the file has been reached
    - or an error is met

Why scripting?

Scripting is a timesaver

The real question: When should you script?

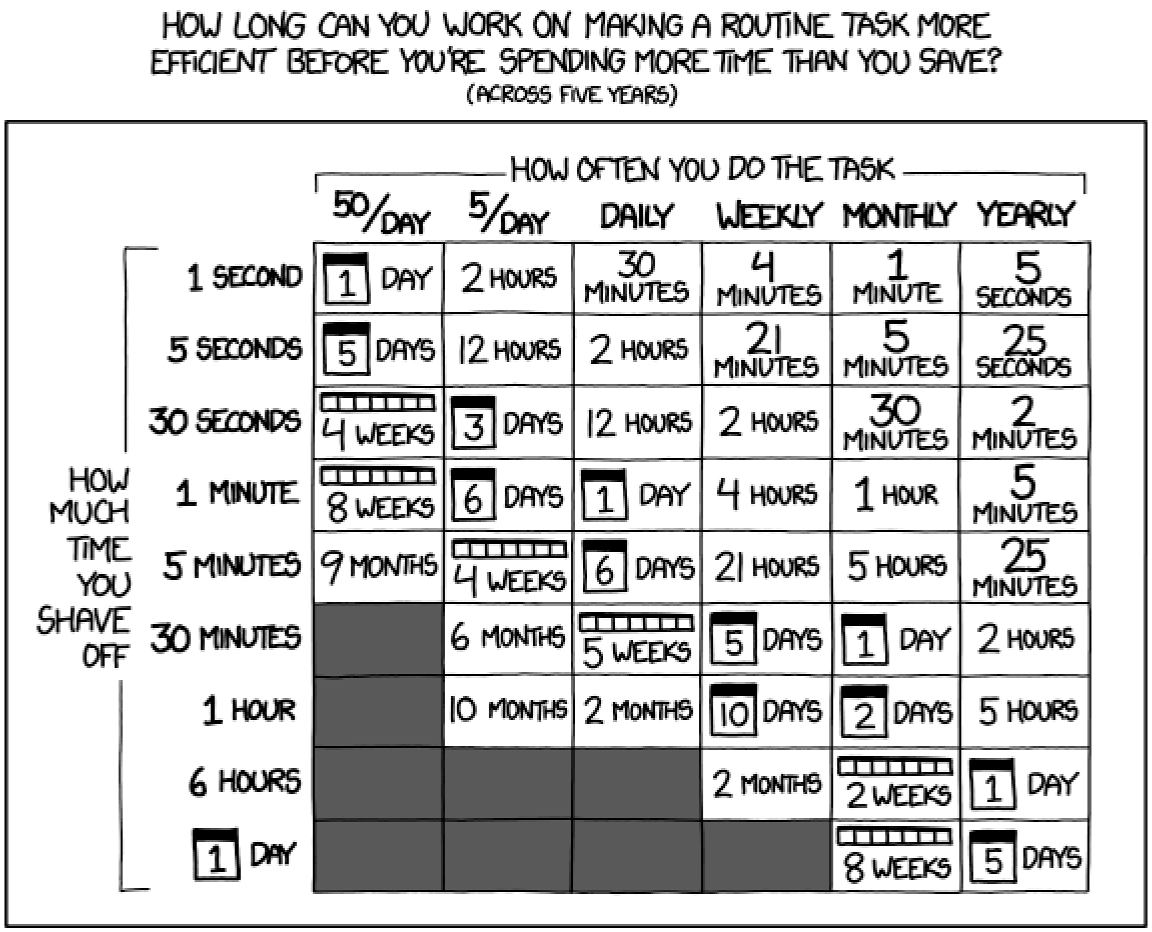
Scenarios for scripting

- Using the batch system at CHPC (discussed in the talk on Slurm Basics)

- Automating pre- and post- processing of datasets

- Performing lots of menial, soul draining tasks efficiently and quickly (like building input files)

How long should you script?



HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE? (ACROSS FIVE YEARS)

HOW MUCH TIME YOU SHAVE OFF	HOW OFTEN YOU DO THE TASK: 50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	■	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	■	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	■	■	■	2 MONTHS	2 WEEKS	1 DAY
1 DAY	■	■	■	■	8 WEEKS	5 DAYS

http://xkcd.com/1205/

Task time saver calculator: http://c.albert-thompson.com/xkcd/

What to script in?

- Basic scripting needs can be done in the Bash shell or the Tcsh/Csh shell.
- If you have more complicated tasks to perform, then you should consider something more advanced (like python* or matlab).
- If your workload is computationally heavy, you should be consider to write your application in a compiled language (e.g. C/C++, Fortran, …).

*CHPC also holds a three part workshop focusing on Python

bash vs tcsh/csh

- A Shell is:
  - a. user interface to the OS's services
  - b. a layer (=> shell) around the kernel
  - c. programming env.
- CHPC currently supports 2 types of "shell-languages"/shells:
  - a. B(ourne) Again Shell (bash)
  - b. Csh/Tcsh shelll
- Syntactic differences are significant (and quirky) => **NO MIXING ALLOWED**
- Some programs do not support different shells (rather rare)
- Very easy to switch between shells
- What shell do I currently use? ***echo $SHELL***

WHILE LEARNING TO SCRIPT,
PICK ONE AND STICK WITH IT.

Can I change my shell? Yes, you can

- To change your default shell: go to chpc.utah.edu and login with your U of U credentials. You will be presented with your profile, which will have a link “Edit Profile”. A new dialogue will show, and you will see an option to change shell. Change it to whatever you want, and save it. Changes will go through in about 15 minutes.
- (Also can be used to change your email on record, please do this if you change email addresses.)

Getting the exercise files

- For today's exercises, open a session to one of the cluster interactives and run the following commands:

```
cp ~u0253283/Talks/LinuxScripting1.tar.gz .
tar  -zxvf  LinuxScripting1.tar.gz
cd LinuxScripting1/
```

Write your first script (ex1)

- Open a file named ex1.sh (Bash) or ex1.csh (Tcsh) using Vi(m)
- ‘#’ character: start of a comment
- Top line always contains the ‘she-bang’ followed by the lang. interpretor:

  ‘#!/bin/bash’ (if you use Bash) or

  ‘#!/bin/tcsh’ (if you use Tcsh)
- Put the following content in a file:

```
echo " My first script:"
echo " My userid is:"
whoami
echo " I am in the directory:"
pwd
echo "Today's date:"
date
echo " End of my first script"
```

- Make the script executable + execute:

```
chmod u+x ./ex1.sh       or  chmod u+x ./ex1.csh
./ex1.sh                 or  ./ex1.csh
```

Setting and Using Variables

```bash
#!/bin/bash
#set a variable (no spaces!)
VAR="hello bash!"
#print the variable
echo $VAR

#make it permanent
export VAR2="string"
echo $VAR2

#remove VAR2
unset VAR2
```

```tcsh
#!/bin/tcsh
#set a variable
set VAR = "hello tcsh!"
#print the variable
echo $VAR

#make it permanent (no =)
setenv VAR2 "string"
echo $VAR2

#remove VAR2
unset VAR2
```

Be careful what you export! Don’t overwrite something important!

Script Arguments

```
#!/bin/bash
ARG1=$1
ARG2=$2
#ARG3=$3, and so on
echo $ARG1
echo $ARG2
```

```
#!/bin/tcsh
set ARG1 = $1
set ARG2 = $2
#set ARG3 = $3, so on
echo $ARG1
echo $ARG2
```

If the script is named “myscript.sh” (or “myscript.csh”), the script is executed with “myscript.sh myarg1 myarg2   ...   myargN”

$0 : returns the name of the script

$#: returns the # arguments

Using grep and wc

- grep searches files for test strings and outputs lines that contain the string
  - VERY fast, very easy way to parse output
  - can use regex and file patterns
  - use backslash (\) to search for special characters (e.g. to search for "!" use "\!")

```
grep "string" filename
```

- wc can count the number of lines in a file

```
wc -l filename
```

Command line redirection (refresher)

- You can output to a file using the “>” operator.

```
cat filename > outputfile
```

- You can append to the end of a file using “>>”

```
cat filename >> outputfile
```

- You can redirect to another program with “|”

```
cat filename | wc –l
```

Exercise 2

Write a script that takes a file as an argument, searches the file for exclamation points with grep, puts all the lines with exclamation points into a new file, and then counts the number of lines in the file. Use “histan-qe.out” as your test file.

Don’t forget **`#!/bin/bash`** or **`#!/bin/tcsh`**

Variables - Bash style: **`VAR="string"`** (no spaces!)
Tcsh style: **`set VAR = “string”`**

Arguments - **`$1 $2 $3 ...`**

Grep - **`grep 'string' filename`**

Counting Lines - **`wc –l filename`**

Solution to Exercise 2

```
#!/bin/bash
INPUT=$1
grep '\!' $INPUT > outfile
wc -l outfile
```

```
#!/bin/tcsh
set INPUT = $1
grep '\!' $INPUT > outfile
wc -l outfile
```

The output from your script should have been "34".

Questions?

Email issues@chpc.utah.edu