DAVE LIDDAMENT

INTRODUCTION TO BASH

@daveliddament

FORMAT

- Short lectures
- Practical exercises (help each other)
- Write scripts

LEARNING OBJECTIVES

- What is Bash
- When should you use Bash
- Basic concepts of Linux shells
- Running several commands together
- Writing scripts
- Home work: Useful commands to learn

WHAT IS BASH?

WHEN SHOULD YOU USE BASH?

HOW EXPERIENCED ARE YOU?

- Not at all, that’s why I’m here! [1]
- A bit, I’ve been using Bash and I know the basics. [2]
- Very, I should be running the workshop! [3]

SECTION 1 - BASICS

- Structure of a command
- Getting help

ANATOMY OF A COMMAND

command [option(s)] <arguments> [<optional arguments>]

ANATOMY OF A COMMAND

command [option(s)] <arguments> [<optional arguments>]

ANATOMY OF A COMMAND

command [option(s)] <arguments> [<optional arguments>]

EXAMPLE

```
mkdir app/src
```

EXAMPLE

```
mkdir app/src app/test target docs
```

EXAMPLE

```
mkdir -p -m 0755 app/src app/test
```

OPTIONS THAT ARE FLAGS

```
mkdir -p -m 0755 app/src app/test
```

OPTIONS THAT TAKE PARAMETERS

```
mkdir -p -m 0755 app/src app/test
```

SHORT AND LONG OPTIONS

-v	--verbose
-a	--archive
-D	
	--append
-l	--links
-L	--copy-links

GETTING HELP

man <command>	man rsync
<command> -h	rsync -h
<command> –help	rsync --help

HOW EXPERIENCED ARE YOU?

- Not at all, that’s why I’m here! [1]

Please help others:

- A bit, I’ve been using Bash and I know the basics. [2]
- Very, I should be running the workshop! [3]

PRACTICAL

- List files in a directory. ls
- List files in a directory showing file size, largest first. ls
- Show the date. date
- Show the date in format RFC 2822. date
- Count the number of **lines** in a file. wc

REVIEW 1 – BASICS

- Structure of a command
- Getting help

SECTION 2 - PERMISSIONS

- Why have them
- How to understand them
- The root user

WHY HAVE PERMISSIONS?

FILE PERMISSIONS

USER, GROUP, OTHER

```
ls -l

-rw-r--r--   1 dave  staff   155 17 Jun  2015 readme.md
-rwxr-xr--   1 dave  staff   155 17 Jun  2015 build
drwxr--r--   1 dave  staff   578 17 Jun  2015 src
```

ROOT USER

HOW EXPERIENCED ARE YOU?

- Not at all, that’s why I’m here! [1]

Please help others:

- A bit, I’ve been using Bash and I know the basics. [2]
- Very, I should be running the workshop! [3]

PRACTICAL

- What groups are you a member of?
  - whoami
  - id
- List files in your current directory. Who can view and edit them?
- List files in /etc/ssh. Who can view and edit the files in here?
  - Find a file that anyone can view but only root can edit.
  - Fine a file that only root can view. What happens when you try and look at it. Use: cat <filename>

REVIEW 2 – PERMISSIONS

- Why have them
- How to understand them
- The root user

SECTION 3 – VARIABLES

- How to set them
- How to read them
- Using variables in commands

SETTING VARIABLES

NAME=dave

MESSAGE=“hello world”

READING VARIABLES

```
echo $MESSAGE

echo “Here is a message from $NAME to you: $MESSAGE”
```

READING VARIABLES 2

```
# Set up a variable
DIRECTORY=/tmp/
# Following line will print nothing. No variable DIRECTORYfoo
echo “$DIRECTORYfoo”
# Following line will print /tmp/foo
echo “${DIRECTORY}foo”
```

VARIABLES IN COMMANDS

```
dir=/tmp

ls  $dir
```

VARIABLES IN COMMANDS

```
# Returns current user

whoami


# Assign user to variable me

me=`whoami`


# Print out message

echo “Your username is $me”
```

VARIABLES IN COMMANDS

```
echo “The current directory is `pwd`”
```

PRACTICAL

- Create variables to hold your first name and surname.
- Create a variable to hold the current time (use the date function)
- Print to screen "Hello <first name> <last name>, the time is <time>"

REVIEW 3 – VARIABLES

- How to set them
- How to read them
- Using variables in commands

SECTION 4 – CHAINING COMMANDS

- Introduction to piping
- Writing to files

PIPES

```
# List all files in a directory

ls


# Count how many files in a directory

ls | wc -l


# Give messages

echo “There are `ls | wc -l` files in the directory `pwd`”
```

REDIRECTING TO FILES

```
# Write Hello to the file

echo “Hello” > message.txt


# Append Goodbye to the file greetings.txt

echo “Goodbye” >> message.txt
```

PRACTICAL

- Look at the following commands. If there are 4 files in the directory what will the output be?
  - ls > files.txt
  - echo “Number of files `cat files.txt | wc -l`”
  - ls >> files.txt
  - echo “Number of files `cat files.txt | wc -l`”
  - ls > files.txt
  - echo “Number of files `cat files.txt | wc -l`”

REVIEW 4 – CHAINING COMMANDS

- Introduction to piping
- Writing to files

SECTION 5 – CHANGING FLOW

- For loops
- If statements

FOR LOOPS

```
# Assume we have a file months.txt of the year on each line:

jan

feb

march

# Run a for loop like this:

for month in `cat months.txt`

do

echo $month

done
```

IF STATEMENTS

```
# Set up some variables:
name1=dave


# If statements like this:
if [ $name == “dave”]
then
  echo “Hello Dave”
fi
```

IF STATEMENTS

```
# Set up some variables:
age=21


# If statements like this:
if [ $age -lt 37 ]
then
  echo “You look much older”
else
   echo “I believe that”
fi
```

IF STATEMENTS

- [-a FILE] True if file exists
- [A -eq B] True if A == B
- [A -ne B] True if A != B

- Lots more: http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html

PRACTICAL

- Experiment with for command
  - Create file with days of week on each line
  - Loop through each line and echo it out
- Play with if command
  - Create simple if statement using string comparison
  - Create simple if statement using integer comparison
  - Create simple if statement to check if file exists

REVIEW 5 – CHANGING FLOW

- For loops
- If statements

SECTION 6 – WRITING A SCRIPT

- Hello World example
- Capturing arguments
- Write your own deployment script

FIRST SCRIPT

```
#!/bin/bash

echo “Hello world”


# Run the script

chmod a+x hello

./hello
```

PASSING ARGUMENTS TO A SCRIPT

```
#!/bin/bash

echo “You passed $# arguments to this script”

echo “Argument 1: $1”

echo “Argument 2: $2”
```

```
# Run the script

./hello

./hello foo

./hello foo bar
```

PRACTICAL 1

- Write a script that takes 1 argument (which is name) and echoes that back to the user
- Checks 1 argument has been passed to it. If it hasn't then print an error message and exit (use exit)
- If name is "Apple" then echo a message saying "Thanks for hosting us"
- Run scripts with different names and missing / too many arguments.

PRACTICAL 2 - DEPLOY SCRIPT

- Create a new directory. Within this directory create the following:
  - directory called log (use mkdir)
  - directory called deploy (use mkdir)
  - directory called code (contains a clone of of https://github.com/DaveLiddament/PHPTraining-PHPUnit-RomanNumerals)
    - git clone https://github.com/DaveLiddament/PHPTraining-PHPUnit-RomanNumerals code

PRACTICAL 2 - DEPLOY SCRIPT

- Write a script that takes 1 argument which is the name of the tag that needs deploying.
- Checks 1 argument has been passed to it. If it hasn't then print an error message and exit (use exit)
- In the code directory checkout tag
- Copy code from code to deploy
- Append to log/deploy.log file an entry that includes time, user who ran the script and the tag that was deployed.
- Add a check that makes sure that the git tag exists (use grep). If it doesn't then report an error.

REVIEW 6 – WRITING A SCRIPT

- Hello World example
- Capturing arguments
- Write your own deployment script

HOMEWORK 1 – USEFUL COMMANDS

- tar
- grep
- sed
- find
- rsync

HOMEWORK 2 - SCRIPTS

- Write a script that takes a dump or your database. Include in the name the time the database was dumped in the format dbname-YYYYMMDD-HHMMSS.dump
- Write a script that generates a release note. It takes 2 git commits SHAs and generates a doc that contains only the commits between the 2 SHAs with messages that start “Add”. Generate various release notes for the RomanNumerals project.