Introduction to Bash Scripting

Lucas Czech
Computational Molecular Evolution Course
Hinxton, 9th of May 2017

Overview

- Variables
- Command Line Arguments
- Loops
- Conditions
- Exercises!

Scripts

- Begin with: `#!/bin/bash`
- Make executable via `chmod 775 script.sh`
- Execute via `./script.sh`

- Task: Create a script that prints “Hello World”

Variables

- Store some information
- Assign via `foo=”Hello CoME”`
- Read via `$foo` or `${foo}`

- Task: Change the script so that it uses a variable with the text

Command Line Arguments

- Access to how the script was called, e.g. `./script.sh some arguments`
- Use `$#` for number of arguments
- Use `$@` for all argument contents
- Use `$1 (etc)` for single argument content

- Task: Print the number of arguments and the content of the first argument, and on the next line, the full content of the command line arguments

Loops

- Use to repeat a certain thing
- Syntax:

```
for variable in list
do
    …
done
```

- Task: Loop over all words in the argument content and print them

Loops

- To repeat a certain number of times:
- ```
  for variable in `seq 10`
  do
      …
  done
  ```

- Task: Print all numbers that are divisible by 10, counting from 100 to 0, i.e.,

```
100 90 .. 0
```
```

Conditions

- Only execute certain parts of the script
- Use for control flow
- Syntax:

```
if [ "foo" = "foo" ]; then
    echo "it's foo!"
fi
```

- Task: Print “`it's foo!`” if the first command line argument is “`foo`”

Conditions

- Use `else` for the other case
- Syntax:

```
if [ "@1" = "foo" ]; then
    echo "it's foo!"
else
    echo “oohh no”
fi
```

Conditions

- Use special if statements to check file stuff
- File exists: `[ -e file ]`
- Directory exists: `[ -d file ]`
- File exists and not empty: `[ -s file ]`
- `file1` newer than `file2`:
  `[ file1 -nt file2 ]`
- Negate expression: `[ ! -e file ]`


Google
Google is
google is god
google is gold
google is your friend
google is
Search

Task: Write a script

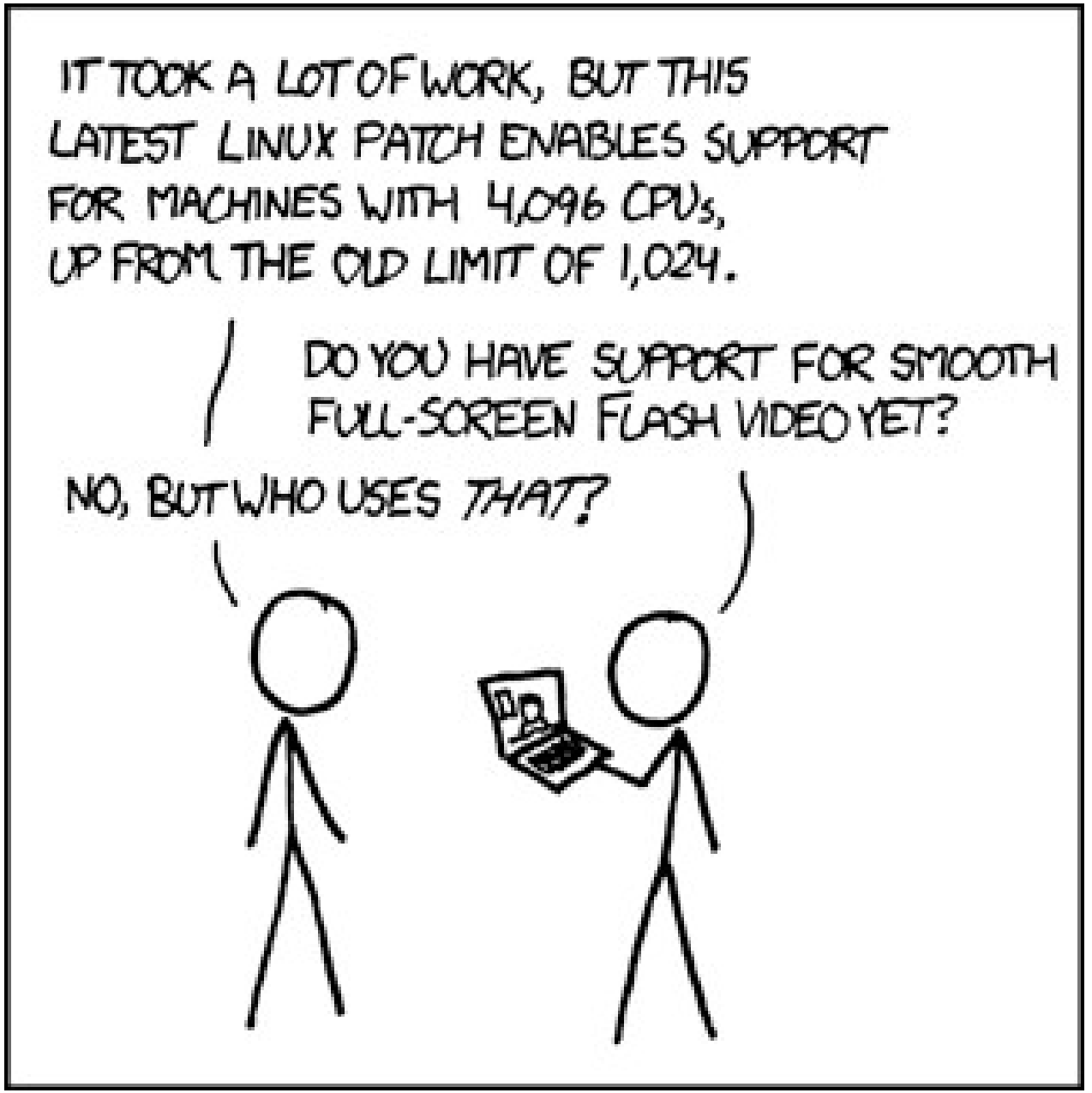
Download “day_2” from
`github.com/lczech/come2017_linux`

- Loop over all `.fasta` files: `for file in ...`
- Get the base name: `name=${file%%.fasta}`
- Move the file to a directory of that name
- Call mafft to get `file.aln`
- Call FastTree to get `tree.newick`

Task: Write a script

- Take a number as input
- Create that many directories, named like this: `dir_` + a random number

  (optional: make sure the directory does not yet exists)
- Create a script in each directory that outputs the random number
- Call these scripts from within your main script



https://xkcd.com/619/