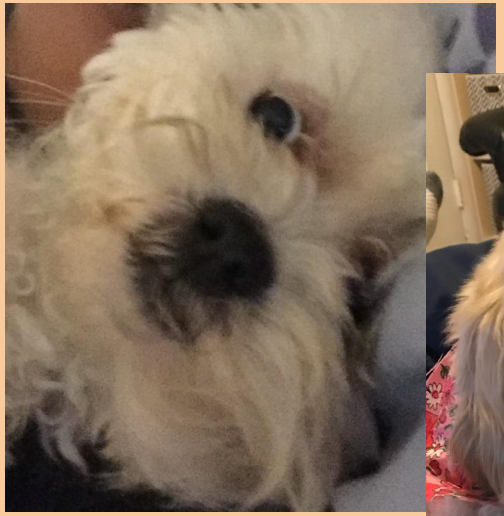
BASH SCRIPTING

Keiffer + Deepti

DOGGO TAX



01

Intro to Scripting

Review the Shell,
Learn to Write Scripts!

What is a Shell?

- Remember the second lecture on terminal!
- Execute **custom commands** that directly affect the computer (file/process management, processing, monitoring, etc.)
- We have been using **Unix** shells, specifically **bash**
  - Other flavors include: `zsh`, `fish`, etc

Command Review

Task	Command
Show directory contents	`ls`
Change directory	`cd directory_name`
Move a file	`mv file.txt location`
Rename a file	`mv file.txt renamed.txt`
Copy a file	`cp file.txt copy.txt`
Execute a binary	`./binary_file` ~~`binary_file`~~ `path/binary_file`
Print something	`echo "Hello World"`

Over, over, and over again

- Sometimes, you want to run the same set of commands multiple times.
  - Compiling and testing your code
  - Renaming a bunch of files
  - Initializing your git repos with a `.gitignore`
  - Archiving your new favorite xkcd comics
  - Setting up your dev environment on a new machine
- Shell scripts allow you to run a sequence of shell commands from a file.

Solution: Bash Scripting!

- Lets you run a sequence of commands from a file
- Can be executed like a binary file

```
#!/usr/bin/env bash

touch source_file.c
mkdir src
mv source_file.c ./src
```

Example

Shebang

```
#!/usr/bin/env bash

touch source_file.c
mkdir src
mv source_file.c ./src
```

Regular commands

bash_script.sh

CHMOD

- Files are not executable by default

```
twildenh@unix5:~/private/script$ ./script.sh
-bash: ./script.sh: Permission denied
```

- Have to add executable permission
  - `chmod +x script.sh`
- Then we can run the script

```
twildenh@unix5:~/private/script$ ./script.sh
Hello World!
```

- `chmod` **ch**anges the **mod**e to add(**+**) e**x**ecutable

Bash as a Programming Language

- Apart from interfacing with the operating system, they support other programming constructs as well
  - Variables
  - Conditionals
  - Loops
- Variable assignment: `VAR=value`
  - <u>Note</u>: NO SPACES IN VARIABLE ASSIGNMENT!!!
- Variable Access: `echo $VAR`
- Command Line Arguments: `$1, $2, $3`, and so on

Scripting Summary

- Bash scripts end in a .sh extension
- Always start with a shebang
  - `#!/usr/bin/env bash`
- Add permissions with `chmod +x script.sh`

Demo Time!

- You're designing different costumes for Halloween 🎃🎃🎃
- You want to make a new directory for each costume design, each containing text files about the design with a git repository initialized
- How can you do this easily with a script?

A more realistic example

- You want to make a new directory for each 15112 HW initialized with git to stay organized
- You give the script a HW number, and it would create the directory for you
- You could create a script like the one on the right

```bash
#!/usr/bin/env bash

HW_NAME=15112_hw$1
mkdir $HW_NAME
cd $HW_NAME
touch README.md
touch main.py
git init
git add .
git commit -m "Create homework directory"
```

02

Globs and Ranges

Write many characters at once!

WHAT DOES A SHELL DO?

01

Enter commands

02

Command string is parsed

```
$ mv source_file.c destination
```

Globs/Ranges

03

```
1 char *args[]={
2   "mv",
3   "source_file.c",
4   "destination"
5 };
6 execvp("mv", args);
```

Bash loads the arguments
and executes the command

04

Stuff happens

Globs and Ranges

```
mv file{1..3}.txt dst/
```

⬇

```
mv file1.txt file2.txt file3.txt dst/
```

Ranges - { .. }

- Surrounded with brackets
- Expands into all possible permutations
- Comma-separated for list of values
- Use “..” to indicate a range of letters or numbers

Matches	Range
1, 2, 3	`{1,2,3} OR {1..3}`
ghost, boo, spooky	`{ghost, boo, spooky}`
bug1, bug2, bug3	`bug{1..3}`
a.1, a.2, a.3, b.1, b.2, b.3, c.1, c.2, c.3	`{a..c}.{1..3}`

Ranges Demo!

- You finished creating the perfect costume design!
- You want to quickly create a checklist of all houses to go trick-or-treating
- How can you do this quickly with a script?

GLOBS

- Special arguments that pattern-matches argument names
- Can be combined with normal text
  - e.g., `rm *.txt` deletes all .txt files

Glob	Matches
?	• any ONE character, e.g. a, b, c, 1, 2, 3, #, @, etc.
*	• ANY number of ANY character

Globs Demo!

- You want to quickly find the houses that have the best candy
- How can you use globs to find this easily?

More Examples

Matches	Pattern
file1 file2 file3	file? OR file{1..3}
file1 file2 item1 item2	{file,item}{1,2}
file4.pdf readme.pdf	*.pdf
file2 file3 file4.pdf	file{2..4}*

Directory Contents

```
file1
file2
file3
file4.pdf
readme.pdf
item1
item2
```

STRINGS

- `echo * Bash scripting is fun *`
- `echo "Bash scripting is "*fun*""`
- Arguments containing spaces/special characters can be written in quotes
  - `echo "* Bash scripting is fun *" -> Bash scripting is fun`
- They can also be written in single quotes
  - `echo 'Bash scripting is "*fun*"' -> Bash scripting is "fun"`

STRING ESCAPING

- Special characters can also be escaped with backslash
  - `echo "Bash scripting is \"fun\""` -> `Bash scripting is "fun"`
- In single quotes, escape characters are ignored.
  - `echo 'Bash scripting is \"fun\"'` -> `Bash scripting is \"fun\"`

BASH SCRIPTING SUMMARY

- Bash scripts end in a .sh extension
- Always start with a shebang
  - `#!/usr/bin/env bash`
- Add permissions with `chmod +x script.sh`
- Globs for pattern matching
  - `*` matches 0 or more of any
  - `?` matches a single char
- More globs can be found here!

LAB PRO TIPS

- Forcelab is out!
- Don’t forget the shebang **#!/usr/bin/env bash**
- Don't forget to do **chmod +x script.sh**
- Do not copy message from the pdf, the apostrophes are different!
- Extratation on Unified Modeling Language (UML) and Object-Oriented Programming (OOP) w/ Amy and Jeremy

Finally, please give us feedback!! :D

http://tinyurl.com/f21-gpi-feedback