Session 4: Shell scripts (Part II) (Andrew Weisman)

Notes:

- The notes below are based on the "bottom" part of this webpage
- The page you are reading can be found here: https://github.com/CBIIT/p2p-datasci/blob/master/workshop_materials/2021-09-21-introduction_to_linux/instructors_notes/shell_scripts_part_2-andrew.md
- Copy that link into the chat, including the link for the course website: https://cbiit.github.io/p2p-datasci/2021-09-09-introduction_to_linux

Review of Part I

A shell script is a single file that contains a bunch of shell (e.g., Bash) commands.

For example, create a file `~/Desktop/shell-lesson-data/molecules/middle.sh` that contains:

```
head -n 15 octane.pdb | tail -n 5
```

Run the script using:

```
bash middle.sh
```

Modify the script to use arguments:

```
head -n "$2" "$1" | tail -n "$3"
```

Run the script using:

```
bash middle.sh octane.pdb 15 5
```

Increase clarity using comments:

```
# Select lines from the middle of a file.
# Usage: bash middle.sh <FILENAME> <END-LINE> <NUM-LINES>

head -n "$2" "$1" | tail -n "$3"
```

Increase clarity in the code itself:

```
# Select lines from the middle of a file.
# Usage: bash middle.sh <FILENAME> <END-LINE> <NUM-LINES>

filename="$1"
end_line="$2"
num_lines="$3"

head -n "$end_line" "$filename" | tail -n "$num_lines"
```

New material

Arbitrary numbers of arguments

What if you want to write a script to process an arbitrary number of arguments? For example, in order to sort by length all the `.pdb` files in the `~/Desktop/shell-lesson-data/molecules` directory, you would run from the command line:

```
# From the command line:

wc -l *.pdb | sort -n
```

You could put this into a script called `sorted.sh` and run it using

```
# From the command line:

bash sorted.sh
```

and it would indeed sort all the `.pdb` files by length.

What if you wanted to sort an arbitrary set of files given on the command line? E.g.,

```
# From the command line:

bash sorted.sh *.pdb ../creatures/*.dat
```

The answer is to use the special variable `$@` to refer to "all command-line arguments to the shell script":

```
# Contents of sorted.sh:

wc -l "$@" | sort -n
```

Note: `"$@"` is special syntax and is equivalent to `"$1" "$2" "$3" ...`.

Homework: List unique species

Saving commands from `history` into a script

If you have done something useful in the shell that you would like to be able to repeat in the future, you can save these commands from history, e.g.:

```
# From the command line:

history | tail -n 5 > redo_steps.sh
```

Then clean up the file and you will be able to simply run the script in the future to repeat your work, such as creating a plot for a presentation.

Note that the shell always adds the last call to the history stack prior to calling the last line in case the last line crashes the shell; this would allow you to see which command caused the crash.

Nelle's pipeline: creating a script

Nelle's supervisor wants to create a script of her analysis so that it is repeatable.

```
# From the command line:

cd ../north-pacific-gyre/2012-07-03/
nano do-stats.sh
```

```
# Contents of do-stats.sh:

for datafile in "$@"; do
    echo $datafile
    bash goostats.sh $datafile stats-$datafile
done
```

Note that `goostats.sh` runs some analysis on the first argument (a file) and writes the results to the second argument (another file).

Nelle can then run her analysis on her datafiles using:

```
# From the command line:

bash do-stats.sh NENE*A.txt NENE*B.txt          # she could do this...
bash do-stats.sh NENE*A.txt NENE*B.txt | wc -l  # ...or this, to output just the
number of files that were processed
```

Exercise: Variables in shell scripts

Exercise: Find the longest file with a given extension

Exercise: Script reading comprehension

Exercise: Debugging scripts

Key Points