# Lecture 17 - Advanced Bash

## **Learning Objectives:**

- 2. Become familiar with the use of Bash, shell programming, and console editors
  - 2.2 Understand the use of basic functions in Bash shell.
  - 2.3 Become proficient in the use of a console editor.
  - 2.4 Understand the syntax Bash commands.
  - 2.5 Understand the use of shell wildcards and regular expressions.
  - 2.6 Learn the use of advanced Bash commands (grep, awk).
- 4. Produce code that is reproducible and produces results that are replicable.
  - 4.5 Use scripts to make command-line functions reproducible.

### **Your Bash Shell**

### Two types of Shells: login and non-login

- Login shell: for interactive instances (mostly)
  - logs in with /bin/login and /etc/profile.d/
  - will read in ~/.bash\_profile instead of ~/.bashrc by default
  - most instances of Terminal, other prompt-type interfaces
  - When tested with echo \$0 should return -bash
- Non-login shell: started by a program without a login, by just passing the name of the shell
  - will call ~/.bashrc but not ~/.bash\_profile
  - ~/.bashrc (if it exists) will call /etc/profile.d/
  - mostly called by executed scripts

### **Your Bash Shell**

### Setting your .bash\_profile

- Purpose: configure your personal shell environment
- Location: in home directory, hidden file
- put this in .bash\_profile to ensure that you have the same working environment in your login and nonlogin shells:

```
if [ -f ~/.bashrc ]; then . ~/.bashrc; fi
```

- setting aliases: this is really helpful for creating shortcuts to common locations and programs

```
alias matlab="/Applications/MATLAB_R2019a.app/bin/matlab
-nodisplay -nosplash -nodesktop"

alias gobox="cd '/Users/waldrop/Dropbox (Chapman)/'"
```

after making changes, don't forget to source it to load those new commands:
 source ~/.bash\_profile

### **Bash command structure**

Basic structure: command flags arguments

rm -r directory/

- Command: command you wish to call.

rm -r directory/

- Flags: options beyond command defaults.

rm -r directory/

- Arguments: items you wish to act upon.

rm -r directory/

### **Important features:**

- spaces: spaces separate commands, flags, and arguments! They are really important!!
- capitalization: bash commands and options are case sensitive. The flags -a and -A may be completely different options!
- working directory/path: need to be specified, pay attention to where you are! Any argument that accepts a file will accept a path.

## Other useful Bash commands

Command	Description	Example
touch	create a new, empty file	touch example.txt
head	print out first 10 lines of a file	head allpara.txt
tail	print out last 10 lines of a file	tail allpara.txt
less	read long files	less allpara.txt
WC	count number of lines, words, characters in a file	wc allpara.txt
basename	extracts base file/directory name from path	basename \$HOME
diff	shows differences between two files	

diff allpara.txt allpara2.txt

# Other useful Bash commands

Command	Description	Example
cut	cuts columns of text file	cut -f 1 allpara.txt
ps	examine process information	ps -ef
kill	kill a process with processid	kill processid
nohup	run a process in the background	nohup command &
chmod	change file permissions	<pre>chmod +x setparameters.s ./setparameters.sh</pre>

### **Shell Wildcards**

? match any 1 alphanumeric character

\* match 0 to any number of alphanumeric characters

[Bb] match character (ignore case)

[0-9] match number sequence

[A-Z] match letter sequence

[A-Z,a-z] match letter sequence (ignore case)

[^0-9] negate a match sequence

\\* escapes special character to interpret literally

#### There are more!

# **Regular Expressions**

#### NOTE: These are a little different than shell wildcards!

- match any 1 alphanumeric character
- \* match 0 to any number of the pervious alphanumeric character
- \* match 0 to any number of alphanumeric characters
- + match 1 or more of the pervious alphanumeric character
- [] square brackets work the same way as in shell
- search at beginning of line
- \$ search at end of line
- \\* escapes special character to interpret literally

There are more!

## **Advanced Bash Commands: grep**

Find a specified pattern. Patterns can be literal or regular expressions. (Note: shell wildcards WILL NOT work in grep.)

Find lines with word "kale": grep "kale" kale.txt

Include line numbers: grep -n "kale" kale.txt

Ignore case within pattern: grep -i "kale" kale.txt

Return only number of times: grep -ni "kale" kale.txt | wc -l

Specify an anchor using ^: grep -ni "^grow" kale.txt

Return only line number: grep -ni "^grow" kale.txt | cut -d : -f 1

Capture *n* lines after pattern: grep -ni -A 10 "^grow" kale.txt

How would you create a file with only growing instructions?

## **Advanced Bash Commands: grep**

**Group work:** In the lecture folder, there is a csv file NW1.csv which is a 28 MB file with 132k lines! It's really large and contains two separate components of a recording stacked on top of each other. The column headers for each data set start with:

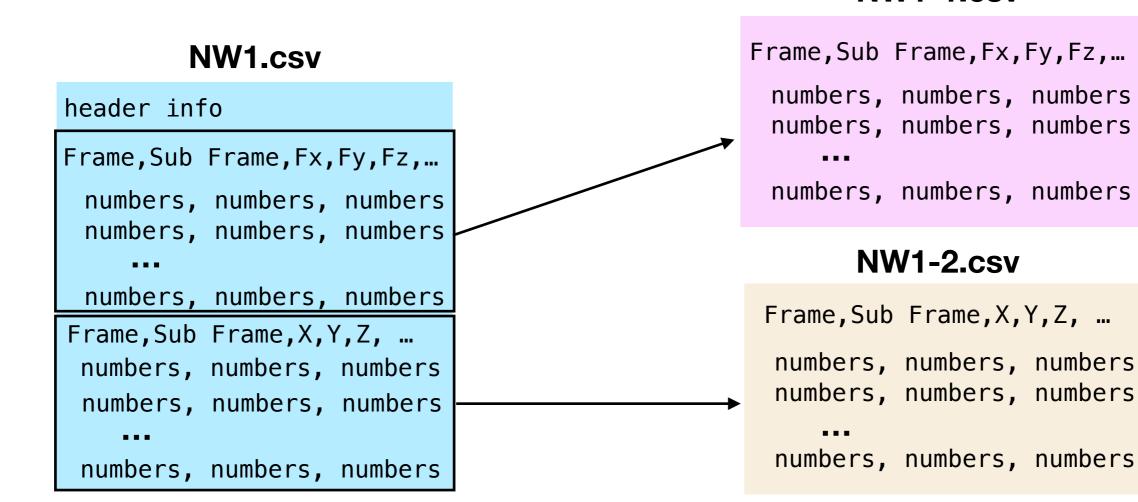
```
Frame, Sub Frame, Fx, Fy, Fz, Mx, My, Mz, Cx, Cy, Cz, Fx, Fy, Fz, Mx, My, Mz, Cx, Cy, Cz, ...

Frame, Sub Frame, X, Y, Z, X, Y, Z
```

Break NW1.csv into two files: NW1-1.csv that contains the first data set and NW1-2.csv that contains the second data set.

NW1-1.csv

NW1-1.csv



# **Your First Bash Script**

#### **Create a script file:**

\$ touch helloworld.sh

#### In console editor:

```
#!/bin/bash

#Prints 'Hello world'
echo Hello world
```

#### Run the script:

\$ sh helloworld.sh

### Make the script executable, then run:

- \$ chmod +x helloworld.sh
- \$ ./helloworld.sh

### Why use scripts?

- You can look at it without running it
- Can be run by you or anyone else
- Makes your setup/analysis/ whatever reproducible!!

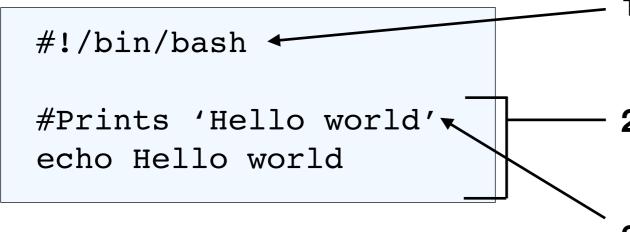
# **Your First Bash Script**

#### **Create a script file:**

\$ touch helloworld.sh

#### Parts of a shell script:

#### In console editor:



- Hashbang tells shell how to interpret commands
- 2. Contents Content of script (from Bash shell)
- 3. Comments begin with pound.

### **Run the script:**

\$ sh helloworld.sh

### Make the script executable, then run:

- \$ chmod +x helloworld.sh
- \$ ./helloworld.sh

# **Your Second Bash Script**

**Group work:** Make a script of the NW1 exercise. Add a few lines that will extract the first

### **More Information**

More on Regular Expressions: <a href="https://www.cyberciti.biz/faq/grep-regular-expressions/">https://www.cyberciti.biz/faq/grep-regular-expressions/</a>

More on grep:

https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/

Difference between login and nonlogin shells: <a href="http://howtolamp.com/articles/difference-between-login-and-non-login-shell/">http://howtolamp.com/articles/difference-between-login-and-non-login-shell/</a>

Bash Programming Tutorial: <a href="https://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html">https://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html</a>

More on parameter expansion: <a href="http://wiki.bash-hackers.org/syntax/pe">http://wiki.bash-hackers.org/syntax/pe</a>