Software Carpentry Workshop

Berkeley Institute for Data Science

June 11, 2018

Instructors: Diya Das, Rebecca Barter, Richard Barnes

Helpers: Jenna Baughman, Caroline Cypranowska

Unix Shell: Automating tasks with the Unix Shell

LINK TO INSTRUCTOR GUIDE: http://swcarpentry.github.io/shell-novice/

Monday AM: Shell

Download the data here and put it on your Desktop: http://swcarpentry.github.io/shell-novice/data/data-shell.zip

To open shell:
- mac: find "terminal" in your computer
- windows: download git bash

*for git bash on Windows, you may need to change the settings to allow copy and paste within the terminal. Click the git icon in the top left corner, go to Properties, Options, and under Edit Options check the box for "QuickEdit Mode" You should now be able to click and highlight text in terminal, then use control + c and control + v to copy and paste.

You could take notes here!

Automating tasks with the Unix shell --notes

ls: list files in current "directory" (folder)

man shows the manual for the command
e.g. **man ls**
q gets us out of the help box
pwd present working directory - tells us where we are
cd takes us to the home directory
cd can also move you around to a different directory; e.g. **cd Downloads**
cd .. takes you back to your previous directory
ls -F : similar to ls but shows trailing / indicating which items in list are directories
ls -a: hidden files

- hidden files are prefixed with .

upzip xxx.zip to unzip
Hit 'Tab' key when typing in a directory name for "tab completion"

The computer writes dates as YYYY-MM-DD, so it can sort them 'alphabetically' i.e. chronologically. Put the date at the start of the file name version. e.g. **2018-06-11_draft.txt 2018-06-12_draft.txt**

mkdir: make directory
rmdir: remove directory

a space is interpreted as an additional argument to the command
e.g. **mkdir my super thesis** : will make three directories, "my", "super" and "thesis"

use a \ to escape the space i.e. make it interpreted as a space
e.g. **mkdir my\ thesis** : will make a single directory "my thesis"
or quotations e.g. mkdir "my thesis"

it's recommended to use lower case file names with no spaces

Command + /- : make font size bigger or smaller (mac) Control +/- in windows

pico/nano command takes you into the (nano?) text editor
^O, 'WriteOut' or save
^=control

If you accidentially end up in the text editor vim, google how to get out of vim. It is powerful but difficult to use.

Up arrow in the terminal brings up the previous command

history: shows everything you've done so far

!### runs that line number from the history

alias and unalias
To redefine commands that you type into the terminal.
go to <**nano .bashrc**>
type alias function e.g. <**alias ls='ls -F'**>
ls will always default to function as if you typed **ls -F**

mv moves files from source to target
-i "interactive" option warns us before overwriting an existing file
-v "verbose" option indicates outcome of the command

cd takes you back to home directory

cd - get you back to where you just were

source command accesses the commands stored in a file(?), at least for .bashrc and .bash_profile

pico .bashrc
source .bashrc

rm : removes a file
rm -rf : force remove a directory (?)

cat command, allows you to see what's inside a file in the terminal window, though it stands for concatenate (likely has other functionality)

less command, allows you to see what's in a file, but in the 'text editor' view in terminal

***** is the wild card. Represents any number of any type of character.

wc: shows word count in each file (lines, words, characters)

note the difference betwen the command :wc -l *pdb and wc *pdb

> is to send the output of a command into something else
e.g. **wc *pdb > lengths.txt**

sort organises the display of the file (doesn't modify the file itself)
sort -n numeric sort
sort -k 2 sort on the second column
sort -r sort in reverse

| is pipe. To pass the output of one command on as the input to the next command.
e.g. **wc *pdb | sort**
note the command " sort -n lengths.txt" does the same job as the command : wc *pdb | sort

control xc to exit at any time

> this creates new file and may overwrite the exisiting file
>> append stuff to the existing file

head look at the top of the file (top 10 lines by default)
tail look at the end of the file (bottom 10 lines by default)
can use -n option followed by a number to define the number of lines shown.

Q: find the file with fewer than 300 lines
wc *txt | sort | head

- - -

Loops
e.g. we want to see the first three lines in both files in the creatures directory
Syntax: for ... in ...
> do

$filename. Define as filename as variable.
Choose this name wisely, so it helps inform you what you're working with when you read back on your code.

for filename in basilisk.dat unicorn.dat; do head -n 3 $filename; done

for datafile in *.pdb; do ls $datafile; done

here 'filename' and 'datafile' are the variable names
do lists the command
done concludes the command

echo command. Like 'print'. Find what is stored in a variable.

for filename in *.dat; do echo $filename; head -n 100 $filename | tail -n 20; done | less
-- prints the name of the file and the 81st to 100th lines in the file, for all .dat files in this directory

for filename in *.dat; do cp $filename original-$filename; done
-- makes a copy of each .dat file, renaming it prefixed with "original-"

With loops, very important to make sure you're iterating over the right variables.

Regular expression: [ ] include options that you want to include. e.g. for datafile in *[AB].txt
Takes filenames ending in either A.txt, or B.txt

To specify that a file is in the folder you are in type: ./filename.

to run a script in Mac, use bash scriptname

goostats is a shell script that's saved in a file called 'goostats'. It takes two arguments (the original filename and the output filename).

if you type **cat goostats** it shows you what's in the file

for datafile in NENE*[AB].txt; do ./goostats $datafile stats-$datafile; done
-- this runs the script goostats on each $datafile, creating a new file prefixed with "stats-"

!! repeats last command you ran.

for alkanes in *.pdb; do cat $alkanes >> alkanes.pdb; done
-- concatonates all the contents of the *.pdb files into a new file "alkanes.pdb"
(... is that right?) yes! (>> means append)

Shell scripts

nano middle.sh
-- makes a shell script file called "middle.sh"
in that file, type **head -n 15 octane.pdb | tail -n 5**

In shell script, "$1", indicates input of the first variable, "$2", input of the second variable, etc.

e.g. **head -n 15 "$1" | tail -n 5**
to run this, **bash middle.sh pentane.pdb**

"$1" use quote just incase some filename has space in it

e.g. **head -n "$2" "$1" | tail -n "$3"**

-- now we're specifying the number of head and tail lines to print as variables
bash middle.sh pentane.pdb 15 5

comments out text .. use for describing what we're doing in the file. Can be used at the end of a line or in a line, too

"$@" takes unlimited number of arguments

we made a shell script (nano do-stats.sh)

```
# calculate stats for data files
for datafile in "$@"
do
echo $datafile
bash goostats $datafile stats-$datafile
done
```

then we can run **bash do-stats.sh NENE*[AB].txt**
so **NENE*[AB].txt** is the argument we're passing to "$@"

grep: to search something in the file