The Unix Shell

Stacey Borrego

1/11/2022

Contents

Lesson 1: Introducing the Shell

- Can everyone find CLI?
- Download to Desktop, "shell-lesson-data"
- Inventory of Windows users
- The Unix shell
  - command line interface and scripting language (BASH)
  - learn new words/commands
  - Script help automate tasks
    - commands -> pipelines -> scripts
  - Interact with remote machines and supercomputers
- Prompt
  - Prompt will be a symbol: `$`
  - Usually will contain additional information: User, current directory, machine name
  - Prompt is followed by text cursor

Key Points

- Advantages are automation of repetitive tasks and the ability to access networked machines like supercomputers and high performance computing systems.
- Disadvantage is how cryptic its commands and operations can be.

Lesson 2: Navigating Files and Directories

- File system
  - manages files and directories (aka folders)

```
# Open shell window
# Print Working Directory
pwd
# The output will look different depending on your system. Keep this in
# mind when you need to find files in future workshop sessions, esp in R

# Want to be in home/root directory "/Users/name"
# Change Directory
cd
# This is a command that can take an argument. If no argument is
# provided, it will take you to the root directory.
# The root directory holds all other directories

# Check new current directory after using `cd`
pwd

# Move into directory we will be working in today `/Desktop/shell-lesson-data"
# absolute path
cd /Users/stacey/Desktop/shell-lesson-data
# relative path
# cd Desktop/shell-lesson-data
# take advantage of tab complete

# Check new current directory after using `cd`
pwd
```

```
## /Users/stacey/Data/GitHub/The-Unix-Shell
## /Users/stacey
## /Users/stacey/Desktop/shell-lesson-data
```

```
# Error messages
pws
# bash: pws: command not found
```

```
cd /Users/stacey/Desktop/shell-lesson-data

# See the contents of a directory
# Listing
ls
# lists the names of files and directories in the current directory

# Use options/flags to modify the output
# -F (adds `/` to indicate directory)
# **capitalization matters**
# **spaces between command and flag matter**
ls -F

# -l (makes output a long list with additional information)
ls -F -l

# combine flags
ls -Fl
```

```
## creatures
## data
```

```
## molecules
## north-pacific-gyre
## notes.txt
## numbers.txt
## pizza.cfg
## project
## projects
## solar.pdf
## thesis_backup
## two
## writing
## creatures/
## data/
## molecules/
## north-pacific-gyre/
## notes.txt
## numbers.txt
## pizza.cfg
## project/
## projects/
## solar.pdf
## thesis_backup/
## two/
## writing/
## total 72
## drwxrwxr-x@  5 stacey  staff    160 Jul 30  2021 creatures/
## drwxrwxr-x@ 12 stacey  staff    384 Jan 13 19:32 data/
## drwxrwxr-x@ 10 stacey  staff    320 Jan 12 08:32 molecules/
## drwxrwxr-x@  3 stacey  staff     96 Jul 30  2021 north-pacific-gyre/
## -rw-rw-r--@  1 stacey  staff     86 Jul 30  2021 notes.txt
## -rw-rw-r--@  1 stacey  staff     13 Aug  4  2021 numbers.txt
## -rw-rw-r--@  1 stacey  staff     32 Jul 30  2021 pizza.cfg
## drwxr-xr-x   3 stacey  staff     96 Jan 12 07:21 project/
## drwxr-xr-x   2 stacey  staff     64 Jan 12 07:26 projects/
## -rw-rw-r--@  1 stacey  staff  21583 Jul 30  2021 solar.pdf
## drwxr-xr-x   3 stacey  staff     96 Jan 12 08:12 thesis_backup/
## drwxr-xr-x   2 stacey  staff     64 Jan 12 07:26 two/
## drwxrwxr-x@  6 stacey  staff    192 Jul 30  2021 writing/
## total 72
## drwxrwxr-x@  5 stacey  staff    160 Jul 30  2021 creatures/
## drwxrwxr-x@ 12 stacey  staff    384 Jan 13 19:32 data/
## drwxrwxr-x@ 10 stacey  staff    320 Jan 12 08:32 molecules/
## drwxrwxr-x@  3 stacey  staff     96 Jul 30  2021 north-pacific-gyre/
## -rw-rw-r--@  1 stacey  staff     86 Jul 30  2021 notes.txt
## -rw-rw-r--@  1 stacey  staff     13 Aug  4  2021 numbers.txt
## -rw-rw-r--@  1 stacey  staff     32 Jul 30  2021 pizza.cfg
## drwxr-xr-x   3 stacey  staff     96 Jan 12 07:21 project/
## drwxr-xr-x   2 stacey  staff     64 Jan 12 07:26 projects/
## -rw-rw-r--@  1 stacey  staff  21583 Jul 30  2021 solar.pdf
## drwxr-xr-x   3 stacey  staff     96 Jan 12 08:12 thesis_backup/
## drwxr-xr-x   2 stacey  staff     64 Jan 12 07:26 two/
## drwxrwxr-x@  6 stacey  staff    192 Jul 30  2021 writing/
```

Helpful Tricks

- Tab complete
- Arrows to show last commands
- `clear` to move content of screen up

Getting Help

- Manual pages: man "command name" (Windows esp)
  - exit by typing "q" as in "quit"
- Internet: Man "command name"
- "command name" –help

General Syntax

- `command option/flag argument`
- options change the behavior of a command
- arguments tell the command what to operate on
- a command can have more than one option and argument
- each part is separated by spaces
- capitalization is important

```
# -s displays the size of files and directories
ls -s /Users/stacey/Desktop/shell-lesson-data

# -S sorts the files and directories by size
ls -S /Users/stacey/Desktop/shell-lesson-data
```

```
## total 72
##  0 creatures
##  0 data
##  0 molecules
##  0 north-pacific-gyre
##  8 notes.txt
##  8 numbers.txt
##  8 pizza.cfg
##  0 project
##  0 projects
## 48 solar.pdf
##  0 thesis_backup
##  0 two
##  0 writing
## solar.pdf
## data
## molecules
## writing
## creatures
## north-pacific-gyre
## project
## thesis_backup
## notes.txt
## projects
## two
## pizza.cfg
## numbers.txt
```

```
# Errors with unsupported options
ls -j
```

```
# ls: illegal option -- j
# usage: ls [-ABCFGHLOPRSTUWabcdefghiklmnopqrstuwx1] [file ...]
```

Exploring other directories

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data"
cd /Users/stacey/Desktop/shell-lesson-data

cd Desktop
pwd

cd shell-lesson-data
pwd

cd data
pwd

ls -F
ls -Fl
```

```
## bash: line 3: cd: Desktop: No such file or directory
## /Users/stacey/Desktop/shell-lesson-data
## bash: line 6: cd: shell-lesson-data: No such file or directory
## /Users/stacey/Desktop/shell-lesson-data
## /Users/stacey/Desktop/shell-lesson-data/data
## amino-acids.txt
## animal-counts/
## animals.txt
## elements/
## morse.txt
## pdb/
## planets.txt
## salmon.txt
## session-info.R.save
## sunspot.txt
## total 216
## -rw-rw-r--@   1 stacey  staff    283 Jul 30  2021 amino-acids.txt
## drwxrwxr-x@   3 stacey  staff     96 Jul 30  2021 animal-counts/
## -rw-rw-r--@   1 stacey  staff    136 Jul 30  2021 animals.txt
## drwxrwxr-x@ 105 stacey  staff   3360 Jul 30  2021 elements/
## -rw-rw-r--@   1 stacey  staff    554 Jul 30  2021 morse.txt
## drwxrwxr-x@  50 stacey  staff   1600 Jul 30  2021 pdb/
## -rw-rw-r--@   1 stacey  staff   8898 Jul 30  2021 planets.txt
## -rw-rw-r--@   1 stacey  staff     45 Jul 30  2021 salmon.txt
## -rw-------    1 stacey  staff     15 Jan 13 19:32 session-info.R.save
## -rw-rw-r--@   1 stacey  staff  73861 Jul 30  2021 sunspot.txt
```

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data"
cd /Users/stacey/Desktop/shell-lesson-data

# List all files, including hidden ones
ls -a

# Shortcut to navigate btx directories
# "." current directory
```

```
# ".." directory containing the current directory
cd ..
pwd
```

```
## .
## ..
## .DS_Store
## .bash_profile
## creatures
## data
## molecules
## north-pacific-gyre
## notes.txt
## numbers.txt
## pizza.cfg
## project
## projects
## solar.pdf
## thesis_backup
## two
## writing
## /Users/stacey/Desktop
```

Navigating the Filesystem

- pwd
- ls
- cd
  - cd (alone) takes you to root directory
  - cd ..
    - cd ../..
  - cd with relative path (from your current position)
    - cd data
  - cd with absolute path
    - cd /Users/stacey/Desktop/shell-lesson-data/data
- Tab complete is helpful with path names

Key Points

- **cd** [path] change working directory
- **ls** [path] lists specific directory
- **pwd** prints working directory
- / alone indicates working directory
- **relative path** specifies a location starting from current location
- **absolute path** specifies a location from the root of the file system
- paths written / on Unix, **** on Windows
- **..** directory above current one
- **.** current directory

Lesson 3: Working with Files and Directories

We know how to explore files and directories but how do we create them in the first place?

Creating Directories

Go to the shell-lesson-data directory on the desktop and see its contents

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data"
cd /Users/stacey/Desktop/shell-lesson-data

# check current directory
pwd

# check contents of directory
ls -F

# create a new directory call "thesis"
# Make directory
mkdir thesis
# this uses a relative path and makes the new directory in current directory

# check contents of directory, again
ls -F

# check contents of new directory
ls -F thesis

# using `mkdir -p` allows you to create a directory with any number of
# subdirectories
mkdir -p project/data/results

# look at all nested subdirectories within a directory use `ls -R`
# `-R` recursively lists contents of directory
ls -F
ls -F project
ls -FR project
```

Good names for files and directories

- No spaces
  - spaces separate arguments
  - use `-` or `_` instead
- No dashes at beginning
  - commands treat names like that as options
- Stick with LETTTERS, NUMBERS, PERIOD, DASH, UNDERSCORES
- Refer to files/dir with special characters with quation marks

Create a text file

Here we use Nano but you can use other, more powerful text editors like Emacs, Vim, Notepad++.

- Nano is a simple text editor
- Can only use cursor and arrow keys to move around
  - Shortcuts help: `command + a` for beginning of line, `command + e` for end of line

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data"
cd /Users/stacey/Desktop/shell-lesson-data

# change into thesis directory
cd thesis

# open nano with document name
# should include extension
```

```
# this doesn't guarentee anything. You can't add .pdf and expect the
# file to be a PDF
nano draft.txt
```

In Nano

- Type: Hello World
- All options are listed on the bottom of na
- `control + o`
  - write out (aka save)
  - writes it to disk
- `control + x`
  - exit

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data/thesis"
cd /Users/stacey/Desktop/shell-lesson-data/thesis

# list all files, should see new file "draft.txt"
ls

# inspect size
ls -l

# see top of filel
head draft.txt

# print all of file
# CAT = concatenate or join together, prints contents one after another
cat draft.txt

# edit file
nano draft.txt
```

Moving Files and Directories

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data/thesis"
cd /Users/stacey/Desktop/shell-lesson-data/thesis

# return to lesson directory
cd Desktop/shell-lesson-data

# check contents of thesis
ls -l thesis

# rename the file using `mv` command
# mv [old] [new]
# MOVE
mv thesis/draft.txt thesis/hello.txt

# check contents of thesis, new file name
ls -l thesis
```

MV be careful. Can silently overwrite an existing file with the same name. You can use `mv -i`, interactive mode, to see these warnings.

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data"
cd /Users/stacey/Desktop/shell-lesson-data
```

```
# move hello.txt to current directory using shortcut `.`
mv thesis/hello.txt .

# check contents of thesis and current directory
ls thesis
ls .
```

Copying files

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data"
cd /Users/stacey/Desktop/shell-lesson-data

# check current directory
pwd

# use `cp` to make a new copy of a file
# cp [from] [to]
cp hello.txt thesis/hello2.txt

# look at contents
ls -l
ls -l hello.txt
ls -l thesis
```

Copying directories

```
# providing error to copy directory
cp thesis thesis_backup

# use `-r` to copy contents recursively within directory first
cp -r thesis thesis_backup

# look at contents
ls -l
```

Wildcard

```
# use the wildcard to search within your current directory and below for
# part of a word
ls thesis*
ls *.txt
ls h*.txt
ls *lo*
```

Removing files and directories

- Deleting is FOREVER
- No trash bin
- Use interactive option to be cautious [-i]
- Recommend .bashrc profile to add `rm -i` in list of aliases

```
# Remove
rm hello.txt

ls hello.txt
```

```
# rm error message
rm thesis
rm -r thesis
rm -ri thesis
```

Lesson 4: Pipes and Filters

now that we kno a few basic commands we can combine them to be more efficient. We will use the `molecules` directory which contains six files of simple organic molecules. Each document contains the type and position of each atom in the molecule.

```
# current directory should be "/Users/stacey/Desktop/shell-lesson-data"
cd /Users/stacey/Desktop/shell-lesson-data

# check out the molecules directory
ls molecules

# change into that directory
cd molecules

# inspect a file using `wc`
# Word count but can do much more
# OUTPUT: lines, words, characters, file name
wc cubane.pdb

# list 'wc' output for all `.pdb` files in current directory
wc *pdb

# use options with `wc` to restrict output
# -l lines
# -m characters
# -w words
wc -l *pdb

# ERROR if argument is not provided
# exit by using `control+c`
wc -l
```

Combine processes

If we had a large number of files, we may want to automate the way we answer questions.

Question: which file contains the fewest lines?

```
# redirect the output that would print to screen into a new file
# called "lengths.txt"
wc -l *pdb > lengths.txt

# check for new file
ls lengths.txt

# inspect contents of file
cat lengths.txt

# sort the list by number of lines using the command `sort` and the option `-n`
```

```
sort -n lengths.txt

# save the sorted file in a new file
sort -n lengths.txt > sorted-lenths.txt

# inspect file
cat sorted-lengths.txt

# print the first line to get your answer
head -n 1 sorted-lengths.txt
```

Quick note about redirecting content to a file

- The angular brackets are used to redirect output to a file
- single angular bracket
  - will generate a new file
  - if file exists, will silently overwrite the contents of file
- double angular brackets
  - will append redirected output to end of file

```
cat sorted-lengths.txt

# echo repeats the argument provided. Is useful for variables
echo end of file

echo end of file >> sorted-lengths.txt

cat sorted-lengths.txt

tail sorted-lengths.txt
```

Combining it all

Instead of running each command separately, we combine them by having the output of one command go to anther command. The vertical bar, pipe character is used to do this.

```
sort -n lengths.txt | head -n 1

# remove any intemediate file
wc -l *pdb | sort -n | head -n 1
```

```
## sort: No such file or directory
## wc: *pdb: open: No such file or directory
```

Key Points

- `cat` displays the contents of file
- `head` displays first 10 lines of its input
- `tail` displays the last 10 lines
- `sort` sorts its input
- `wc` count lines, words, and characters
- command > [file] redirects a command's output to a file (overwriting existing content)
- command » [file] appends a command's output to a file
- [first] | [second] is a pipeline: the output of the first command is used as the input to the second

Lesson 5: Loops

Variables

```
echo hello

x=hello
echo x
echo $x

stacey=hello
echo $stacey
echo ${stacey}

echo ${stacey}_goodbye
echo $stacey_goodbye
```

```
## hello
## x
## hello
## hello
## hello
## hello_goodbye
```

Loops

Allows to repeat a command or set of commands for each item in a list

- Application: modifying photo files

We are going to use the `creatures` directory. It contains genome data files that contains the common name, classification, updated date, and DNA sequence.

```
# current directory
cd /Users/stacey/Desktop/shell-lesson-data/creatures

# change directory
cd creatures

# inspect files
head -n 5 basilisk.dat minotaur.dat unicorn.dat

# retrieve any line
# classification
head -n 2 basilisk.dat
head -n 2 basilisk.dat | tail -n 1

# Example of loops
nano loops.txt

for thing in list_of_things
do
  operation_using $thing (#variable representing the item in the list)
done
# repeat command fro each item
# 1 item from list is assigned to variable
# call variable inside loop
```

```
# each iteration assigns file to variable


# Loop for command line - use TAB complete
# Written on the
for filename in basilisk.dat minotaur.dat unicorn.dat
do
head -n 2 $filename | tail -n 1
done
# variable name is not important
# change variable name to `x`
```

- Unnecessary to write out list of things. Can use tools to identify files such as wildcard.
- Important to give good filenames. Makes referring to lots of files easier. Avoid spaces, quotes, and other symbols.

```
# Good example
for filename in *.dat
do
echo $filename
done

# Bad example - nothing is being done to the file
for filename in *.dat
do
$filename
done
```

Save file in loop

```
# Loop that overwrites output file
for filename in *.dat
do
echo $filename > file.txt
done

cat file.txt

# Loop that appends output to file
for file in *.dat
do
echo $filename >> file.txt
done

cat file.txt
```

Adding on to the loop

- print file name
- print lines from file

```
for filename in *.dat
do
echo $filename
head -n 2 $filename | tail -n 1
done
```

Making copies

```
# Bad example
# COPY: make a copy of files. Requires two arguments [old] [new]
cp *.dat original-*.dat

ls *.dat

# needs to be written one file at a time
cp basilisk.dat original-basilisk.dat

# use a loop
# check loop first with echo. It is harmless when it fails and succeeds
for filename in *.dat
do
echo $filename
cp $filename original-${filename}
done

# check for the new files
ls
```

Access history

```
history
history | tail -n 5

# Reverse search
# `control+R`
```

Key Points

- For loop repeats commands for everything in list
- Every for loop needs a variable to refer to
- Give good filenames
  - Avoid spaces and special characters
  - Simple names make it easier to process files

Lesson 6: Shell Scripts

Scripts take commands that we use and saves them in a file. This is a small program.

Important to write scripts in a "plain text editor" and save file as "plain text". Text editors like Word, store extra info that the command line will try to interpret.

```
# current directory
cd /Users/stacey/Desktop/shell-lesson-data

cd molecules

# creating new file to save script
nano middle.sh

# script to select lines 11-15 of the file octane.pdb
head -n 15 octane.pdb | tail -n 5
# control+o, write out/save file
```

```
# control+x, exit

# check for middle.sh
ls

# run script
bash middle.sh

# edit script to be more versatile
# special variables indicate the argument provided on the CL when the
# script is run
# $1 first argument on the command line
nano middle.sh
head -n 15 "$1" | tail -n 5

bash middle.sh
bash middle.sh octane.pdb
bash middle.sh pentane.pdb
```