Shell

Introduction

Learning Goals

1. Explain how the shell relates to the keyboard, the screen, the operating system, and users' programs.
2. Explain when and why command-line interfaces should be used instead of graphical interfaces.

Files and Directories

1. Explain the similarities and differences between a file and a directory.
2. Translate an absolute path into a relative path and vice versa.
3. Construct absolute and relative paths that identify specific files and directories.
4. Explain the steps in the shell's read-run-print cycle.
5. Identify actual command, flags, and filenames in command-line call.
6. Demonstrate the use of tab completion, and explain its advantages.

- whoami
- pwd
- /
- ls
- ls -F
- ls -F data
- ls -F /data
- cd data
- cd ..
- ls -F -a
- ls north-pacific-gyre/2012-07-03
- ls no tab

Creating Things

1. Create a directory hierarchy that matches a given diagram.
2. Create files in that hierarchy using an editor or by copying and renaming existing files.
3. Display the contents of a directory using the command line.
4. Delete specified files and/or directories.

- mkdir thesis
- cd thesis
- nano draft.txt
- rm draft.txt
- rm thesis
- rmdir thesis

- rm -r thesis
- mv thesis/draft.txt thesis/quotes.txt
- mv thesis/quotes.txt .
- cp quotes.txt thesis/quotations.txt

Create a workspace on your desktop so that it's easy to find, and easy to explore with your GUI filesystem tool (Explorer, Finder, Nautilus, ...)

```
$ cd
$ cd Desktop
$ mkdir swc
$ cd swc
```

Creating Things

1. Create a directory hierarchy that matches a given diagram.
2. Create files in that hierarchy using an editor or by copying and renaming existing files.
3. Display the contents of a directory using the command line.
4. Delete specified files and/or directories.

- mkdir thesis
- cd thesis
- nano draft.txt
- rm draft.txt
- rm thesis
- rmdir thesis

- rm -r thesis
- mv thesis/draft.txt thesis/quotes.txt
- mv thesis/quotes.txt .
- cp quotes.txt thesis/quotations.txt

Exercise

What command(s) could you run so that the commands below will produce the output shown? (and do it)

```
$ ls
analyzed   raw
$ ls analyzed
fructose.dat    glucose.dat    sucrose.dat
```

Pipes and Filters

1. Redirect a command's output to a file.
2. Process a file instead of keyboard input using redirection.
3. Construct command pipelines with two or more stages.
4. Explain what usually happens if a program or pipeline isn't given any input to process.
5. Explain Unix's "small pieces, loosely joined" philosophy.

- cd molecules
- wc *.pdb
- *, ?
- wc -l
- wc –help
- wc -l *.pdb > lengths
- cat lengths
- sort lengths
- sort lengths > sorted-lengths
- head -1 sorted-lengths
- sort lengths | head -1
- cd north-pacific-gyre/2012-07-03
- wc -l *.txt
- wc -l *.txt | sort | head -5
- ls *Z.txt

We're going to start working with Nelle Nemo's Great Pacific Garbage Patch files, so everybody needs a copy of her directories and files so that you can pretend that you are Nelle.
Use Mercurial to grab the files from Bitbucket and put them in a `nnemo` directory in your SWC workspace:

```
$ cd
$ cd Desktop/swc
$ hg clone https://bitbucket.org/douglatornell/swc-nelle-files nnemo
```

You can copy and paste the `hg clone` command from the Etherpad. We'll learn what it means in the Version Control with Mercurial section later today.

Pipes and Filters

1. Redirect a command's output to a file.
2. Process a file instead of keyboard input using redirection.
3. Construct command pipelines with two or more stages.
4. Explain what usually happens if a program or pipeline isn't given any input to process.
5. Explain Unix's "small pieces, loosely joined" philosophy.

- cd molecules
- wc *.pdb
- *, ?
- wc -l
- wc –help
- wc -l *.pdb > lengths
- cat lengths
- sort lengths
- sort lengths > sorted-lengths
- head -1 sorted-lengths
- sort lengths | head -1
- cd north-pacific-gyre/2012-07-03
- wc -l *.txt
- wc -l *.txt | sort | head -5
- ls *Z.txt

Loops - Part 1

1. Write a loop that applies one or more commands separately to each file in a set of files.
2. Trace the values taken on by a loop variable during execution of the loop.
3. Explain the difference between a variable's name and its value.
4. Explain why spaces and some punctuation characters shouldn't be used in files' names.

- for ... do ... done
- varname, $varname
- echo
- "$varname"

Loops - Part 2

1. Demonstrate how to see what commands have recently been executed.
2. Re-run recently executed commands without retyping them.

- ls *[AB].txt
- Up-Arrow
- history
- Ctrl-A, Ctrl-E
- Ctrl-R
- Ctrl-C

Exercise

In your `analyzed` directory, what is the effect of this loop?

```
for sugar in *.dat
do
    echo $sugar
    cat $sugar > xylose.dat
done
```

1. Prints `fructose.dat`, `glucose`, and `sucrose`, and copies `sucrose` to create `xylose`.
2. Prints `fructose`, `glucose`, and `sucrose`, and concatenates all three files to create `xylose`.
3. Prints `fructose`, `glucose`, `sucrose`, and `xylose`, and copies `sucrose` to create `xylose`.
4. None of the above.

Shell Scripts

1. Write a shell script that runs a command or series of commands for a fixed set of files.
2. Run a shell script from the command line.
3. Write a shell script that operates on a set of files defined by the user on the command line.
4. Create pipelines that include user-written shell scripts.

- bash myscript.sh
- $1, $2, ... $n, $*
- # comment
- history | tail -4 > script.sh

Exercise

Write a shell script called `longest.sh` that takes the name of a directory and a filename extension as its parameters, and prints out the number of lines, directory, and name of the file with the most lines in that directory with that extension. For example:

```
$ bash longest.sh more-molecules pdb
```

would print the number of lines, directory, and name of the `.pdb` file in `more-molecules` that has the most lines.

Finding Things

1. Use `grep` to select lines from text files that match simple patterns.
2. Use `find` to find files whose names match simple patterns.
3. Use the output of one command as the command-line parameters to another command.
4. Explain what is meant by "text" and "binary" files, and why many common tools don't handle the latter well.