EEB 177 Lecture 3

Topics

- Shell basics
- Git recap

Preliminaries

- Start gedit: `$ gedit` and save the file "classwork-Tues-1-17.txt" to your class-assignments directory
- push this to your remote repository
- you can write answers to today's exercises in this file.

What is Unix?

- operating system written by AT&T Bell scientists in the 70s
- many variants today: OpenBSD, Sun Solaris, Apple OS X, Linux
- multi-user, network-oriented, stores data as plain text files

The Shell

why use it?

- easier to log commands
- easier to script
- writing GUIs takes a long time

What is a shell?

- program which displays command line
- interface for communicating with core OS (kernel)
- several versions. Default on OS X is bash
- check now. Open terminal and type `echo $SHELL`

Why use Linux?

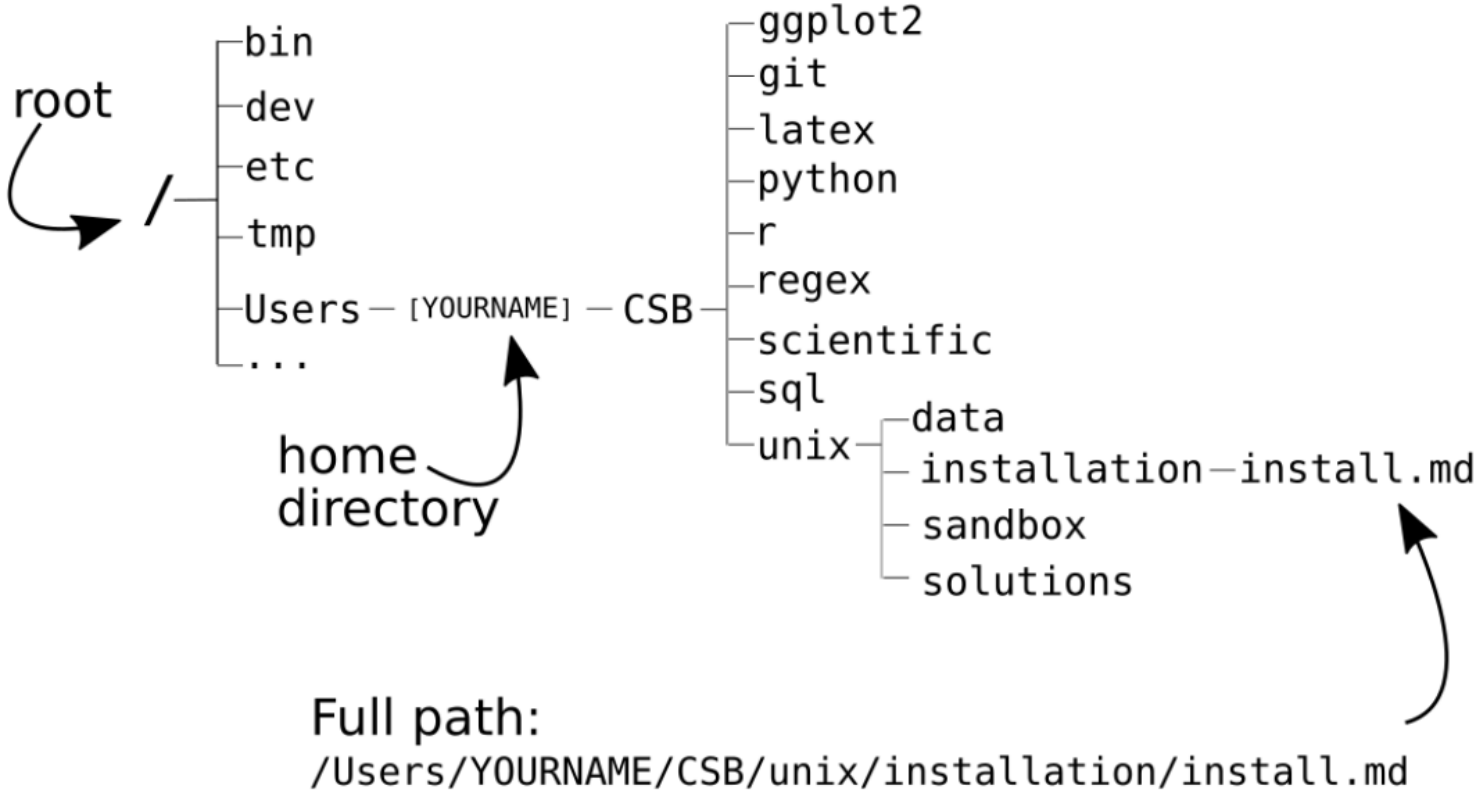
- environment written by programmers/scientists for programmers/scientists
- 100s of super efficient small programs available, easily strung together
- text-based
- stable, many tutorials
- can't avoid it!

Directory structures

know these

- root `/` the topmost directory in the hierarchy
- home `/Users/YourName` where your stuff lives
    - shortcut to home directory: `~`
    - print your home directory `echo $HOME`

Figure 1.1: An exemplary directory structure in the OS X operating system (in Ubuntu, the directory `home` is equivalent to the directory `Users` in OS X).



Using the shell

- open terminal in your virtual box using `Ctrl + Alt + t`

Ctrl + A	Go to the beginning of the line
Ctrl + E	Go to the end of the line
Ctrl + L	Clear the screen
Ctrl + U	Clear the line before the cursor position
Ctrl + K	Clear the line after the cursor
Ctrl + C	Kill the command that is currently running
Ctrl + D	Exit the current shell
Alt + F	Move cursor forward one word (in OS X, `Esc + F`)
Alt + B	Move cursor backward one word (in OS X, `Esc + B`)

Spaces

Challenge 1: Open a terminal and create a file called "badly named file.txt" using the `touch` command

- try copying this file to your home directory using with the `cp command`
    - `cp filename directory`
    - what just happened?

Dealing with spaces in file names

- file names with spaces must be indicated with escape character: `\`
- `cp badly\ named\ file.txt ~`
- (avoid spaces in file names)

Naming files

- file names can contain letters, numbers, and . , _
- names that start with a period will be hidden `.config`
- avoid using uppercase letters
- avoid naming files with same name as a unix command

Overview of navigating files and directories

we will work through the software carpentry lesson on this topic here
http://swcarpentry.github.io/shell-novice/02-filedir/

Getting help in Unix

- `man` brings up the manual
- try `man ls`
- google and stackoverflow also extremly useful

What if no manual is available?

Sometimes you will see this message when you try to access the manual.

```
$ man cd
```

```
No manual entry for cd
```

This can happen with commands that are part of the shell program itself

```
$ type cd
```

```
cd is a shell builtin
```

To see the manual for these commands type `man bash`

Getting around in a manual

Manuals can be long and intimidating. To find a specific character string within a manual, type `/stringname` within the manual. You can press `n` to see the next occurrence of the string.

Use the manual to find the arguments of the **`cd`** command.

cd

- `cd ..` move one directory up
- `cd /` move to the root directory
- `cd ~` move to your home directory
- try going to the sandbox directory in `CSB/unix`

pwd

- prints the path of your current directory
- try it!

ls

- lists files and subdirectories of current directories
- takes arguments
  - `ls -a` lists all files (including hidden)
  - `ls -l` lists files with details
  - `ls -h` gives sizes in human readable format

absolute and relative paths

- the absolute path is the entire path starting from the root, `/`
  - `/Users/michael_alfaro/tools`
- the relative path is defined relative to the current directory
  - if we were in the `/tools` directory above, we could get to the root using the relative path: `cd ../../../`
  - we could also go there using the aboslute path: `cd /`

Challenge 2

- Go to your home directory.
- Navigate to the sandbox within the CSB/unix directory.
- Use a relative path to go to the data within the python directory.
- Use an absolute path to go to the sandbox directory within ggplot2.
- Return to the data directory within the python directory.

cp

- copies files
- `cp file_to_copy path_to_destination`

```
# If you specify the full path,
# your current location does not matter
$ cp ~/CSB/unix/data/Buzzard2015_about.txt
    ~/CSB/unix/sandbox/
# assume your current location is the unix sandbox
# we can use a relative path
$ cp ../data/Buzzard2015_about.txt .
# the dot is shorthand to say "here"
# rename the file in the copying process
cp ../data/Buzzard2015_about.txt ~/buzz-2015.txt
```

mv

moves (or renames) files

- `mv file_to_move destination`

```
# move the file to the data directory
    $ mv Buzzard2015_about2.txt ../data/
# rename a file
$ mv ../data/Buzzard2015_about2.txt Buzzard-2015.txt
```

touch

- creates an empty file (or updates date of access of exiting file)
- `touch my_file_name.txt`

rm

- removes a file
- `rm -r` removes files recursively

mkdir

- creates an empty directory
- `mkdir my_new_dir`
- to make nested directories use `-p`
- `mkdir -p dir_a/dir_b/dir_c`

rmdir

- removes an empty directory
- `rmdir directory_name`
- be careful--no trashcan in unix!
- use `-r` to recursively remove files from non-empty directory
- `rmdir -r directoryname`

history

- shows the command line history from the current session (this is a BASH command)
- `history` (try it)

>

the `>` symbol redirects output. This turns out to be very handy. We can redirect the command line history, for example, like so:

```
history > unix-commands-used-in-class.txt
```

try it! Check to see if this file shows up in your current directory.

Appending output with >>

If you want to send output to an existing file, use `>>`. This will add the contents to the end of the file.

Challenge 3

send the list of the directory contents of `class-assignments` to `classwork-Tues-1-17.txt`

Challenge 4

1. cd to your classwork directory
2. Create a nested directory sequence that corresponds to Linnean taxonomy in one line.
3. Navigate to the species directory and create three text files corresponding to Disney animal sidekicks.
4. Move one file to the Class directory using the absolute path.
5. Copy a second file to the Family directory using the relative path
6. Delete the genus directory.
7. send the command history file to `classwork-Tues-1-17.txt` and push to your repo.

Introduction to Regular Expressions

Regular expressions (regex) are

- powerful and portable tool for manipulating text
- built into many languages and environments
- think of regex as super-powered search-and-replace

Examples

Regular expressions will save you a lot of time!

You will work through these in lab. Lets get a taste for these right now at this link: https://regexone.com/

Files and directories practice

We will go through the software carpentry lesson on files and directories here: http://swcarpentry.github.io/shell-novice/03-create/

Git recap

We will now review basic Git operations from lab using examples from CSB section 2.4 and the git lessons at http://swcarpentry.github.io/git-novice/

ignoring files in git

Often we want git to ignore some of the files in our workign directories. You can set up a gitignore file to do this.

- gitignore example from `http://swcarpentry.github.io/git-novice/06-ignore/`
- useful files to ignore: https://gist.github.com/octocat/9257657

Printing and modulating files

We will go through examples from CSB section 1.6.4 together.

Wildcards

We will go through examples from CSB section 1.6.5 together.

Permissions

We will go through examples from CSB section 1.6.6 together.

Advanced Commands

If we have time, we will go through CSB section 1.7 together.

Homework

- Review the creation of a remote repository (http://swcarpentry.github.io/git-novice/07-github/). Then, find a new partner and do the exercises on collaborating in git here: http://swcarpentry.github.io/git-novice/08-collab/
-
- Do questions 1-3 from exercises 1.11.1 in CSB