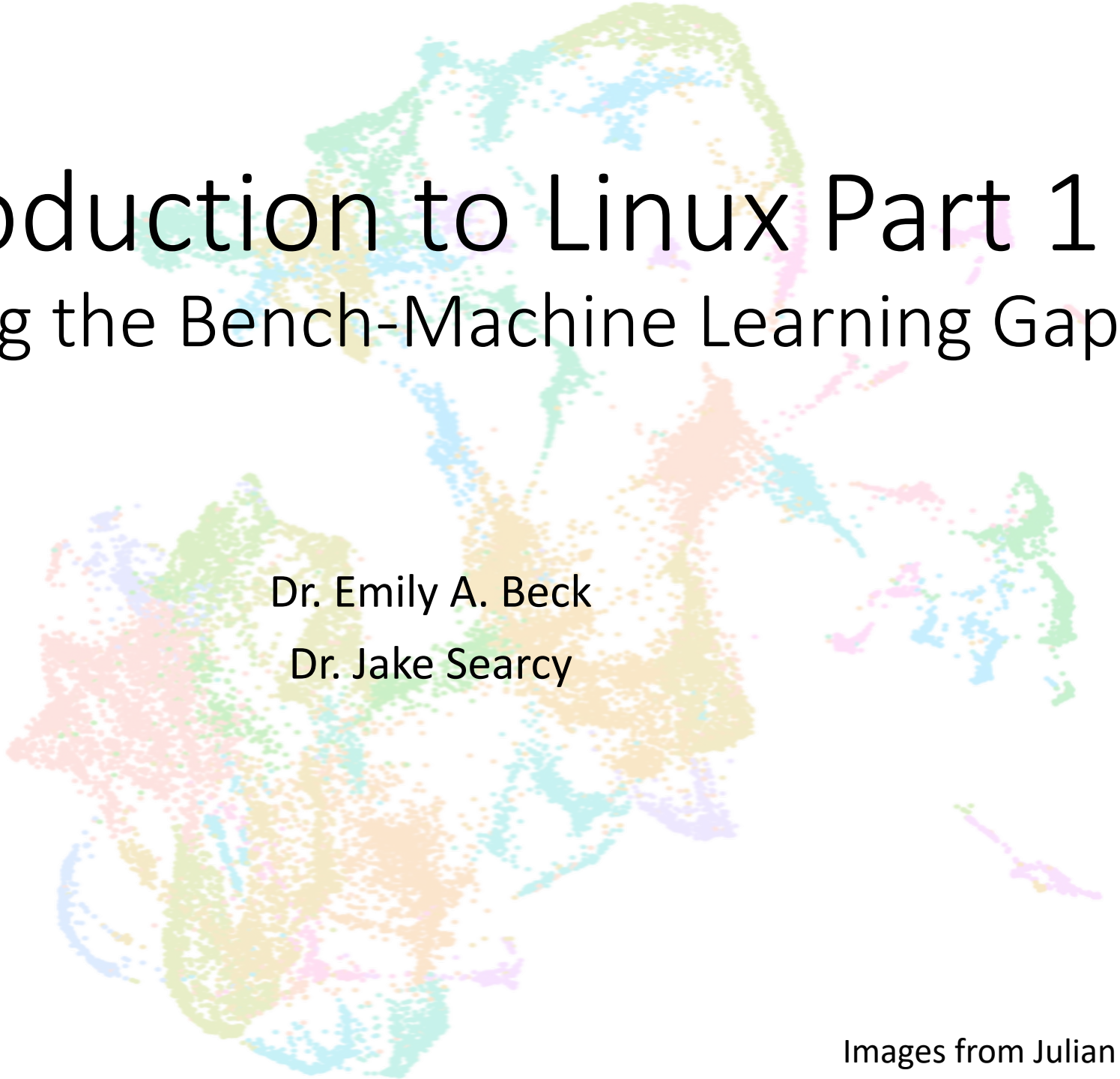
Introduction to Linux Part 1

Bridging the Bench-Machine Learning Gap

Dr. Emily A. Beck

Dr. Jake Searcy

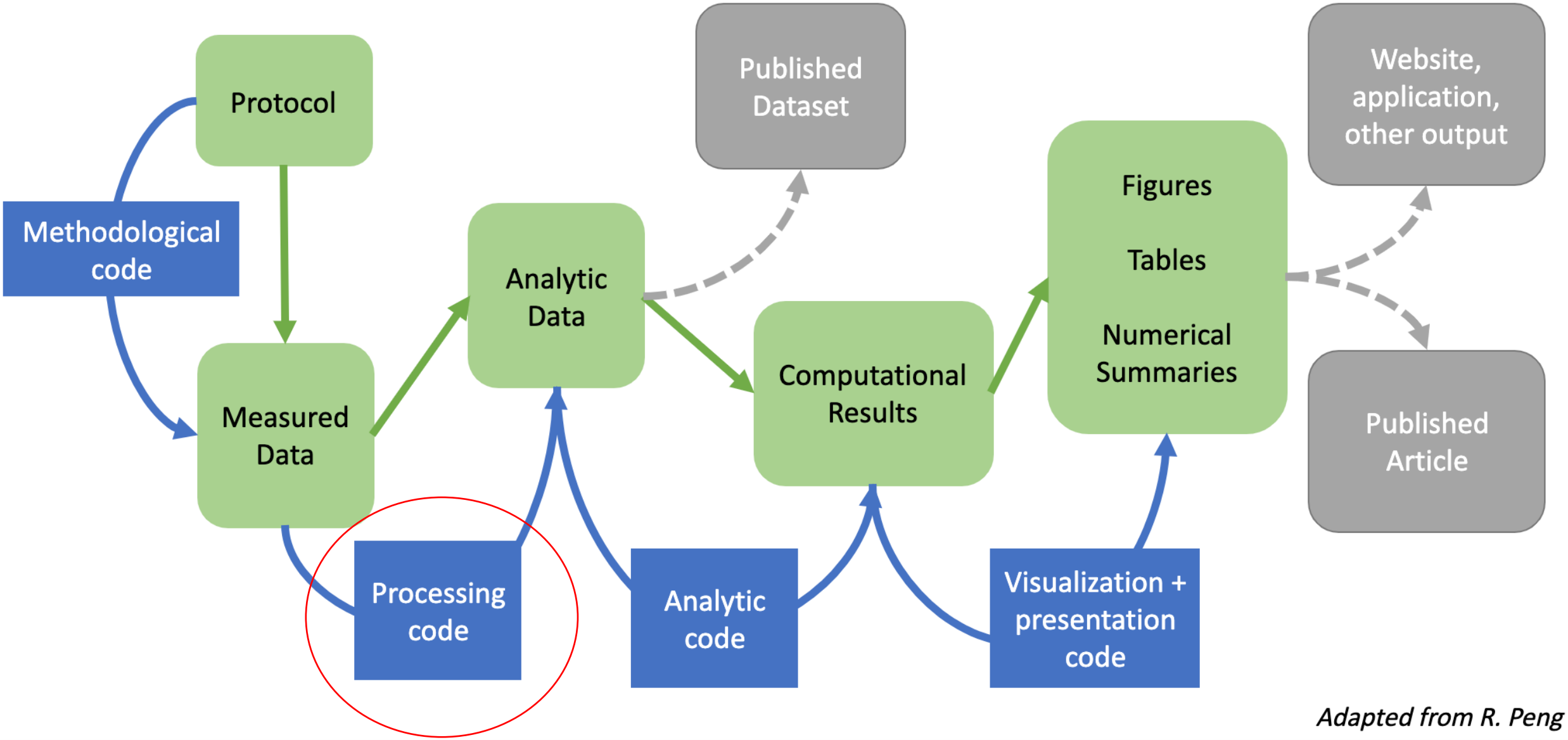


Images from Julian Catchen-Evomics

Learning Objectives

- Become familiar with the terminal
- Learn to navigate the terminal and understand file structures
- Develop strategies for good data practices and version control
- Learn various methods for viewing large files
- Learn to move large files between and within machines

Research workflow



Adapted from R. Peng

We will be using Open OnDemand but there are other ways of interfacing with the Terminal

Mac

Command + Space type “Terminal”

Finder + Search type “Terminal”

PC

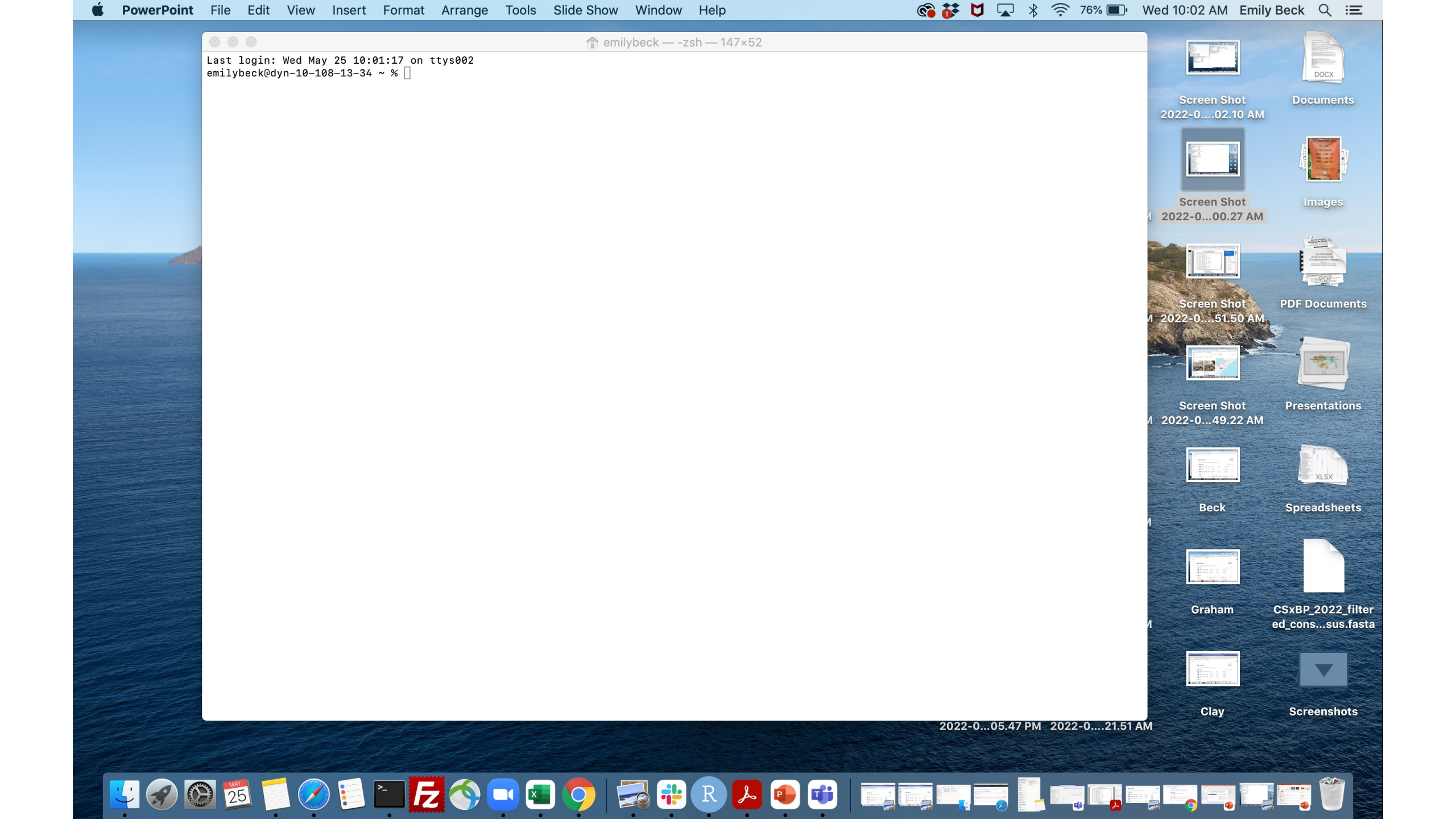
1 **Open your computer's Start menu.** Click the Windows icon on the bottom-left corner of your desktop or press the ⊞ Win key on your keyboard.

- Alternatively, you can click the search or Cortana button next to the Start menu icon.

2 **Type `cmd`, `Command Prompt` or `PowerShell`.** After opening the Start menu, type this on your keyboard to search the menu items. Command Prompt or PowerShell will show up as the top result.

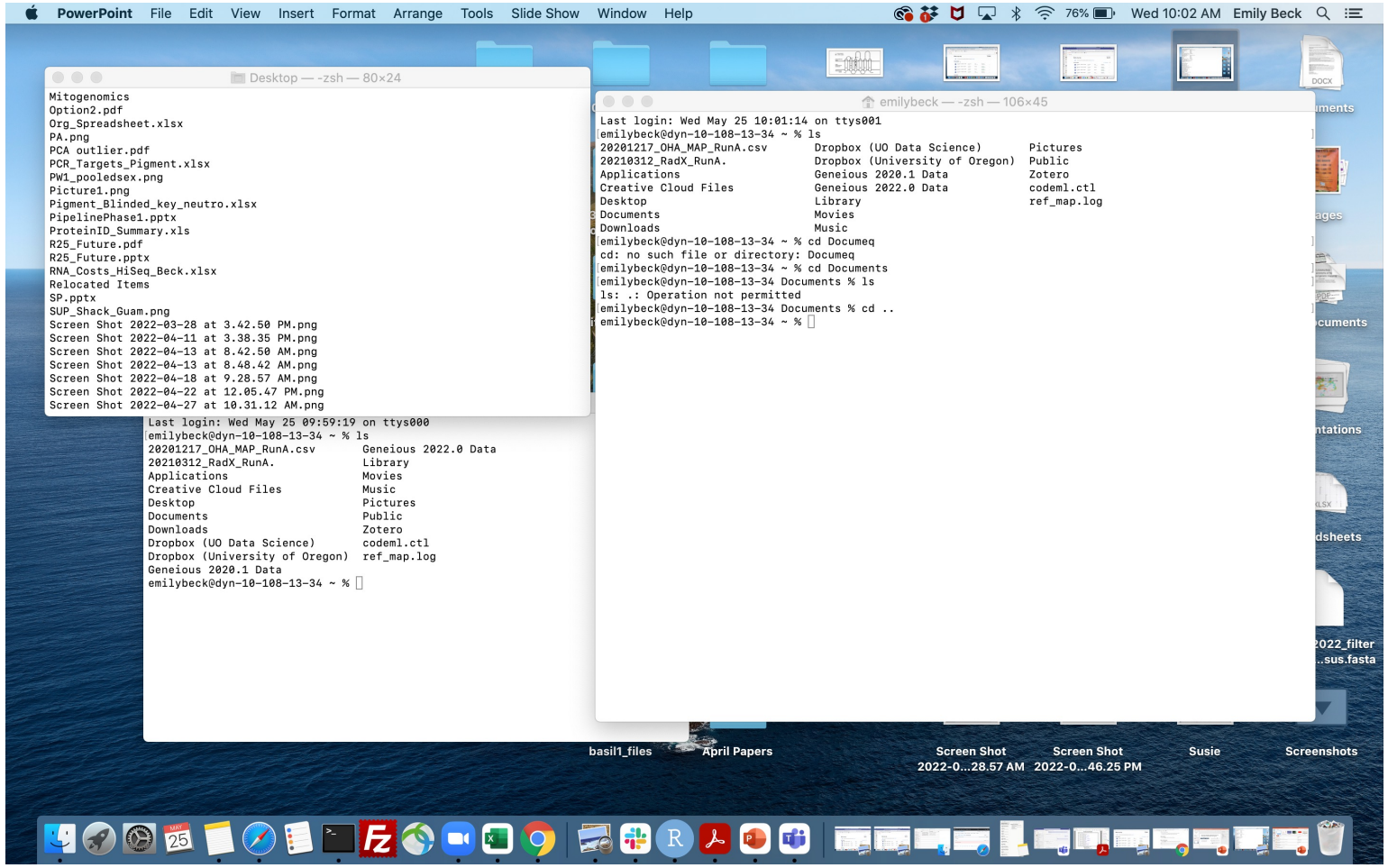
- Alternatively, you can manually find Command Prompt or PowerShell on the Start menu.
- Command Prompt or PowerShell is in the **Windows System** folder on Windows 10 & 8, and in the **Accessories** folder under All Programs on Windows 7, Vista & XP.

3 **Click the Command Prompt or PowerShell app on the menu.** This will open the Command Prompt or PowerShell terminal in a new window.


PowerPoint File Edit View Insert Format Arrange Tools Slide Show Window Help
76% Wed 10:02 AM Emily Beck
emilybeck — -zsh — 147×52
Last login: Wed May 25 10:01:17 on ttys002
emilybeck@dyn-10-108-13-34 ~ %
Screen Shot 2022-0....02.10 AM
Documents
Screen Shot 2022-0...00.27 AM
Images
Screen Shot 2022-0...51.50 AM
PDF Documents
Screen Shot 2022-0...49.22 AM
Presentations
Beck
Spreadsheets
Graham
CSxBP_2022_filtered_cons...sus.fasta
Clay
Screenshots
2022-0...05.47 PM
2022-0....21.51 AM

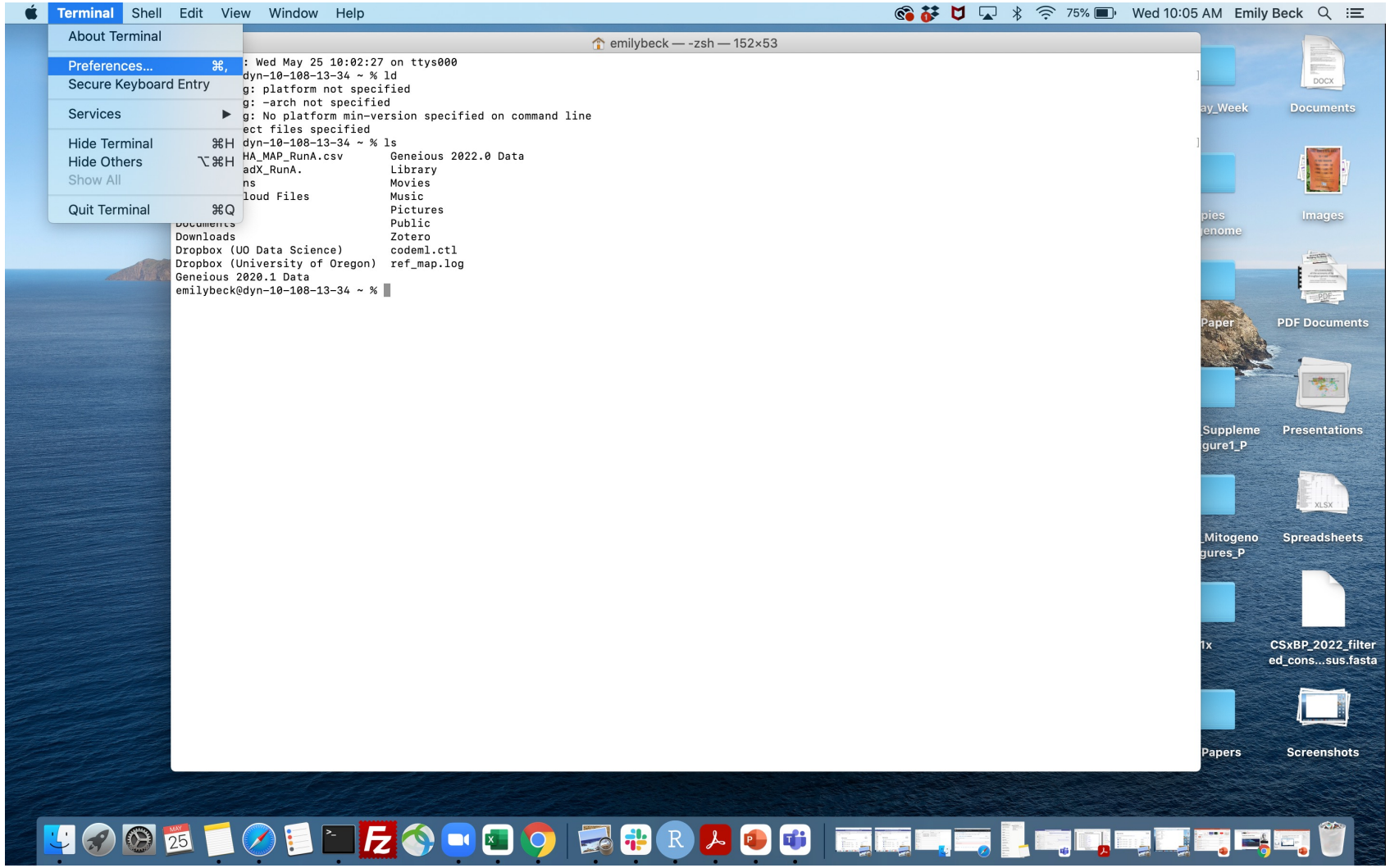
You can open multiple terminals which is useful for multi-tasking

(We will learn more ways to multi-task later during our Programming Powerup)

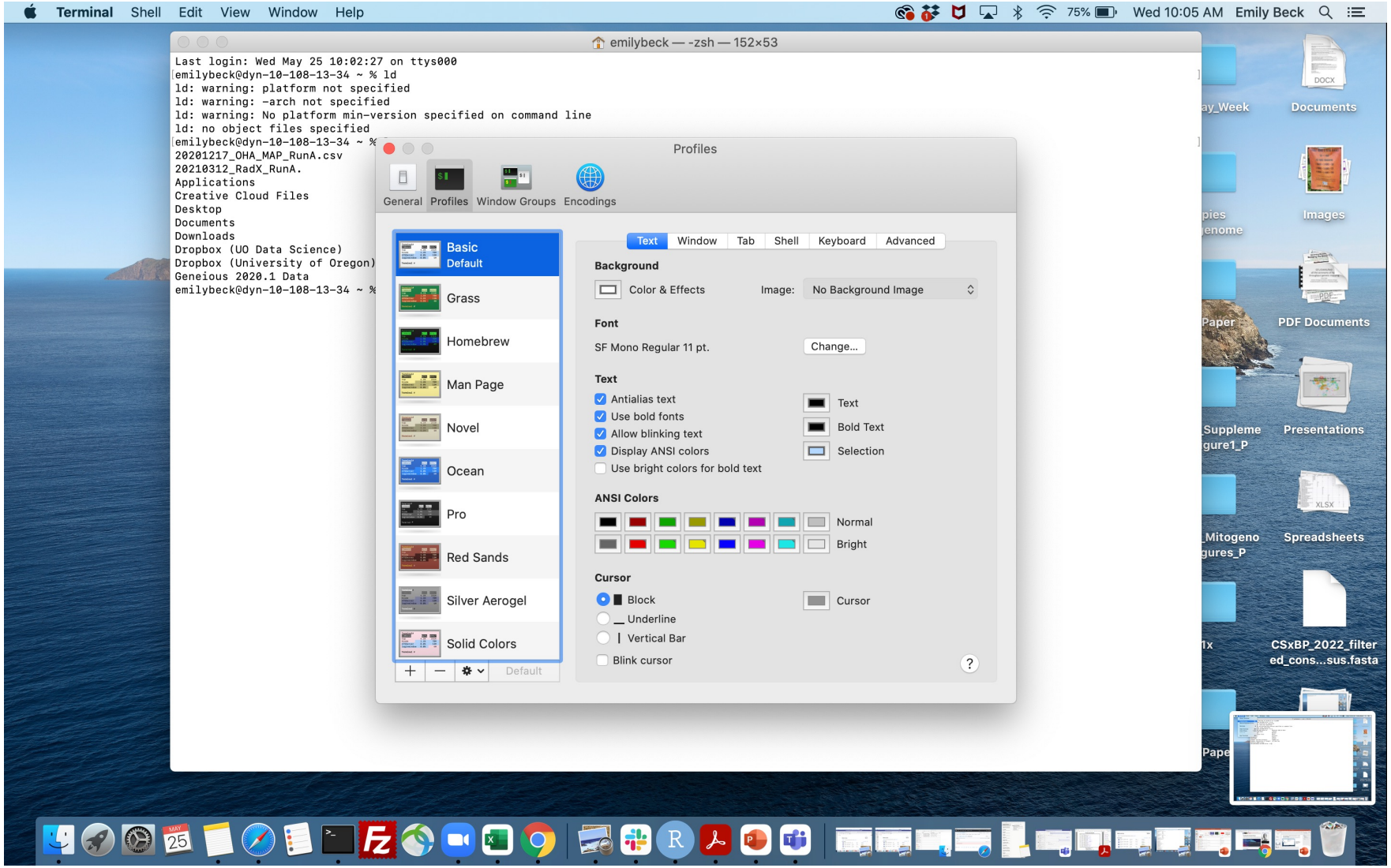


You can resize the terminal or customize it based on your color preferences!

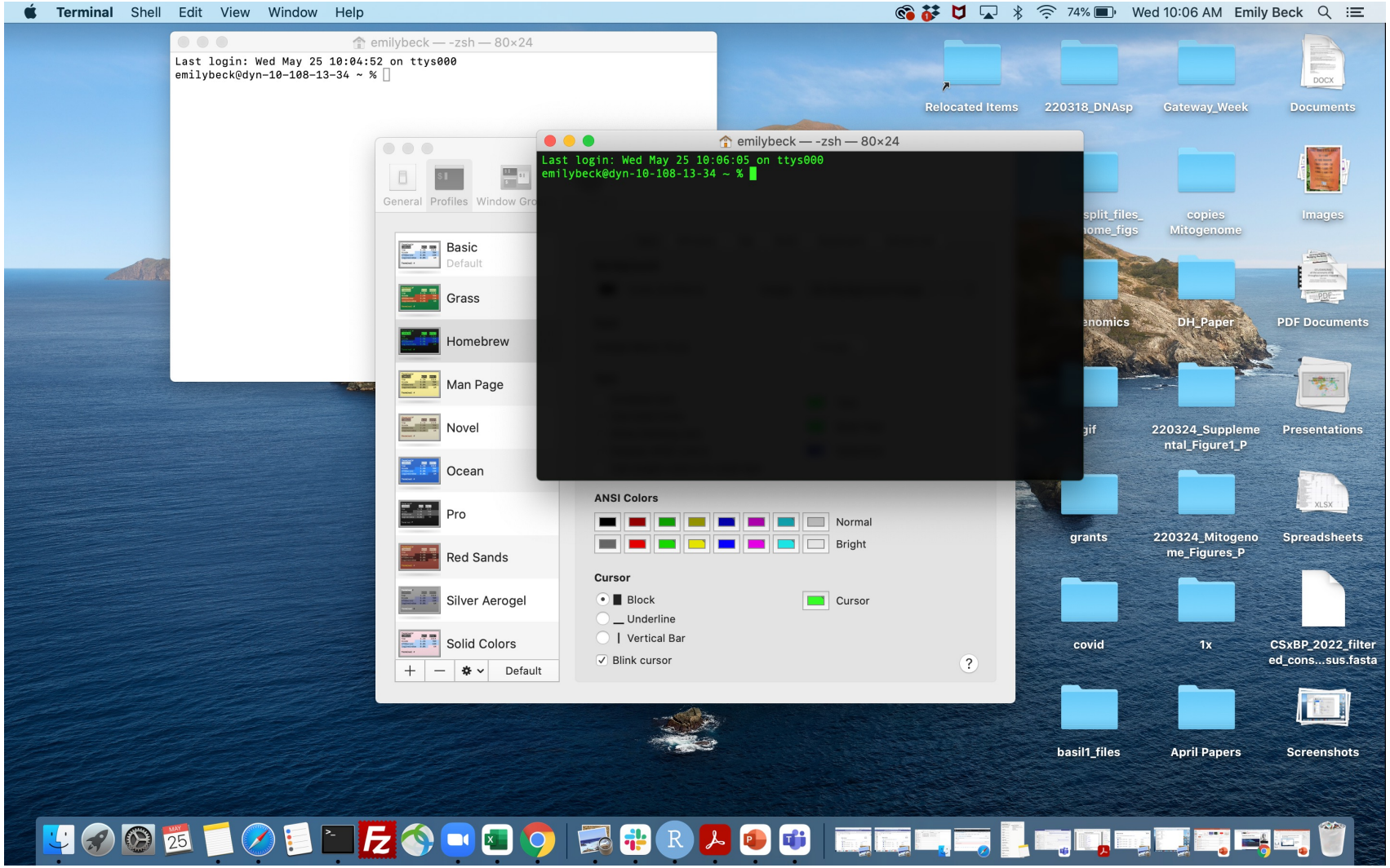
It is worth it to customize the terminal to what works for you



It is worth it to customize the terminal to what works for you

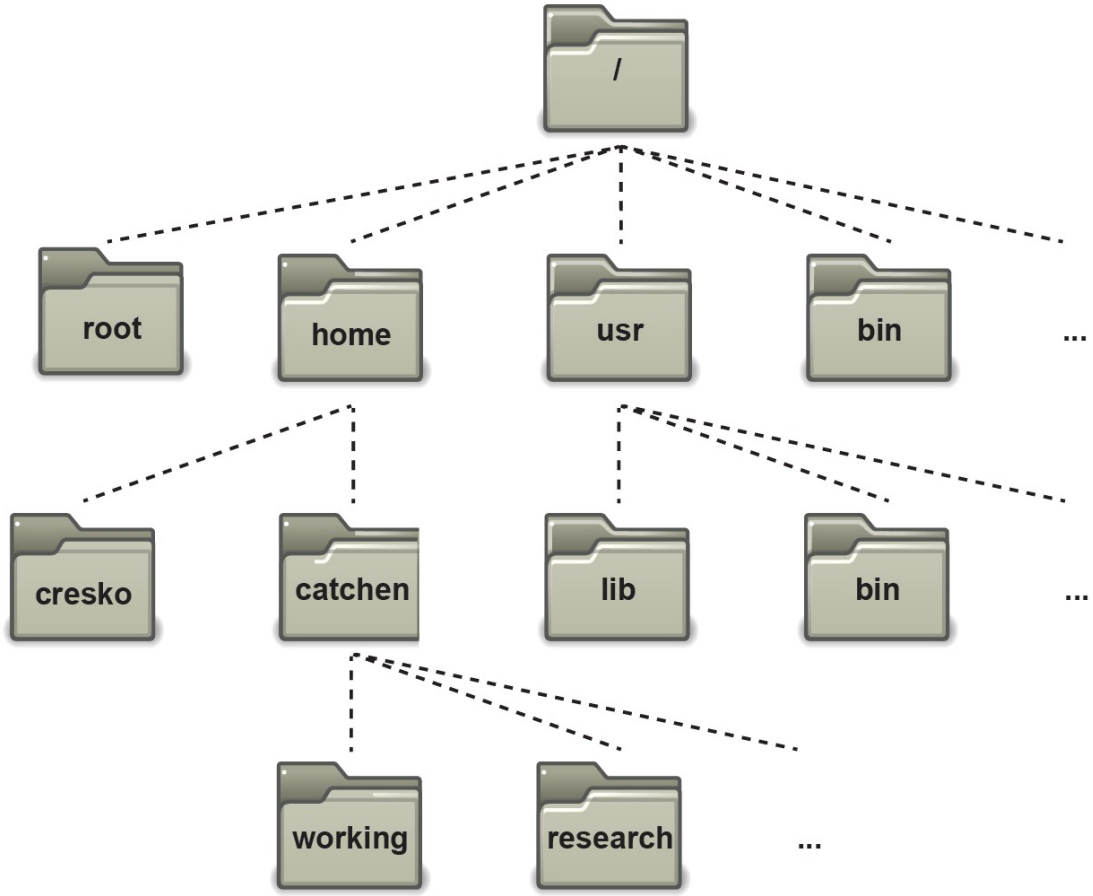


It is worth it to customize the terminal to <u>what works for you</u>

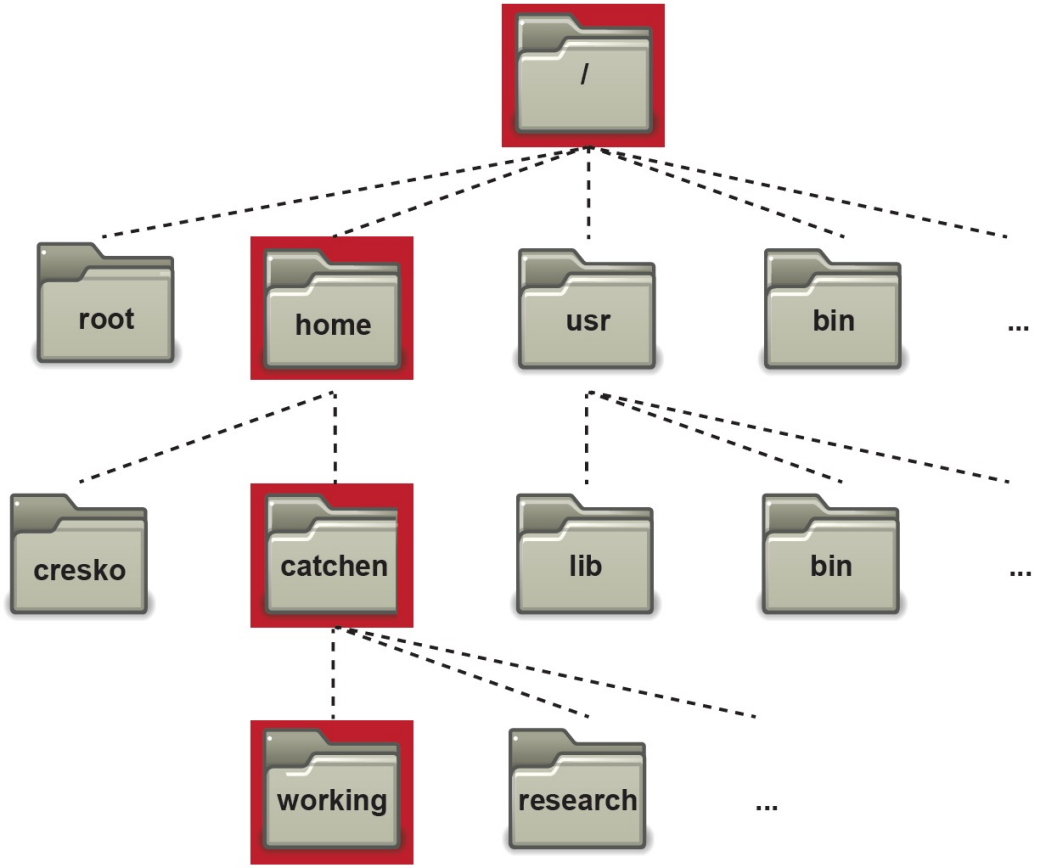


I’m in the terminal, but WHERE AM I?

In Unix everything is organized in a file hierarchy



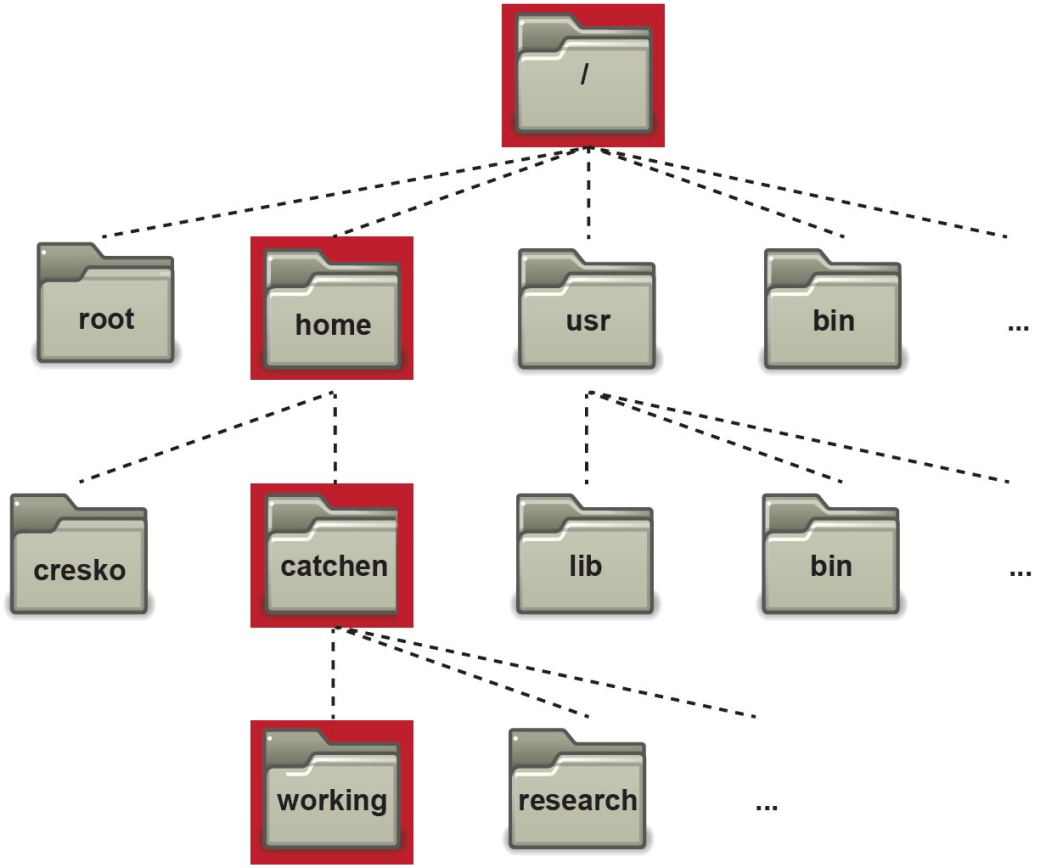
I’m in the terminal, but WHERE AM I?
In Unix everything is organized in a file hierarchy



`/home/catchen/working`

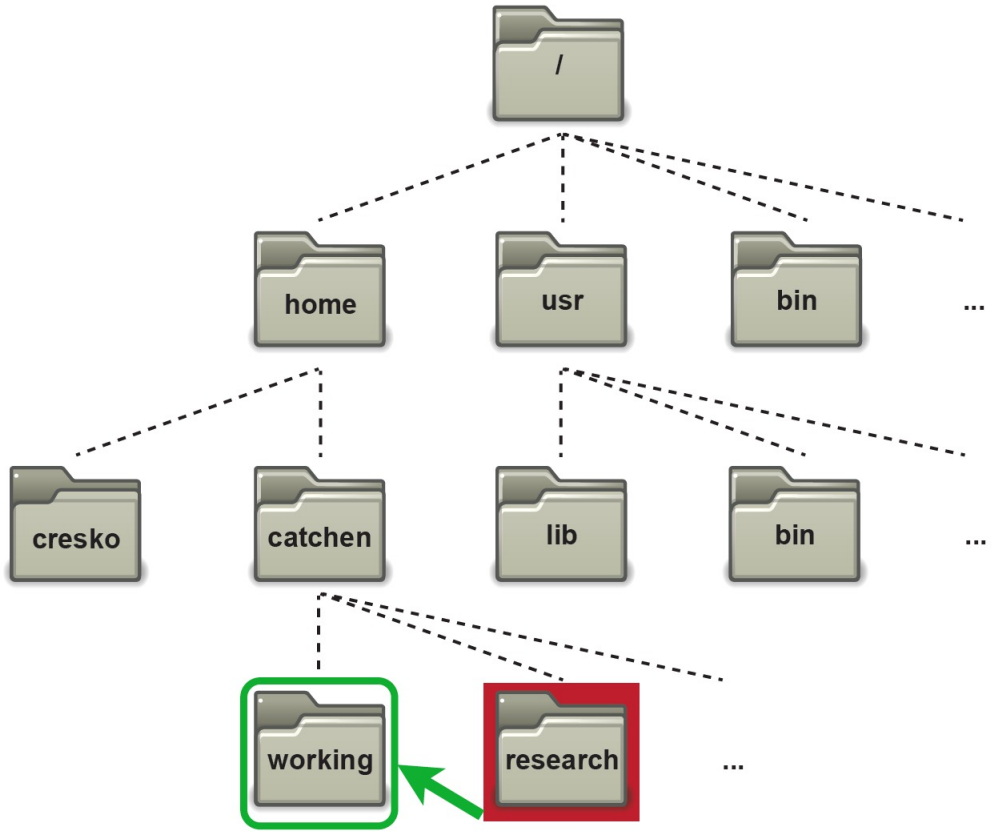
This is how we represent a “Path” which lets you know where you are

The Absolute Path is the Path from my home directory to the folder of choice



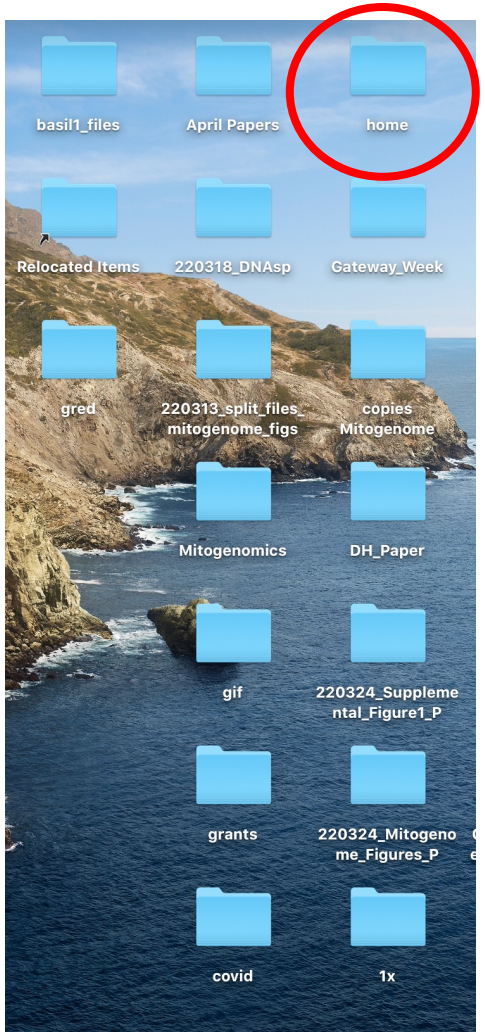
/home/catchen/working

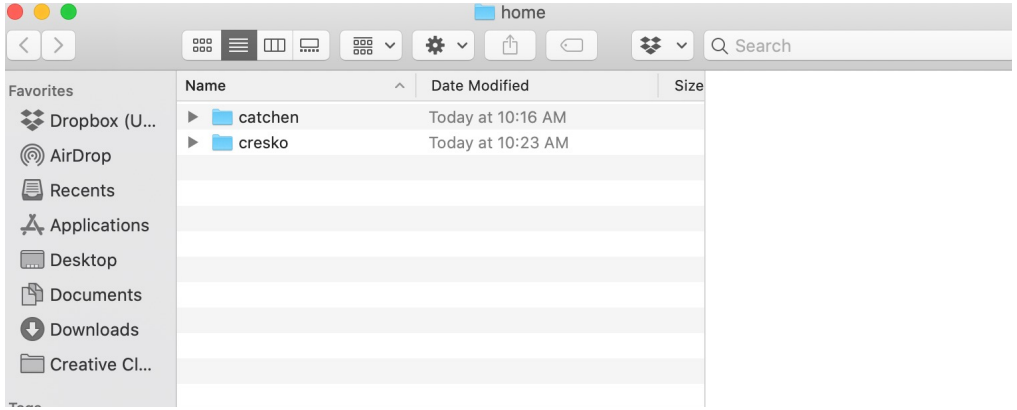
The Relative Path is the Path from wherever I am to where I want to go

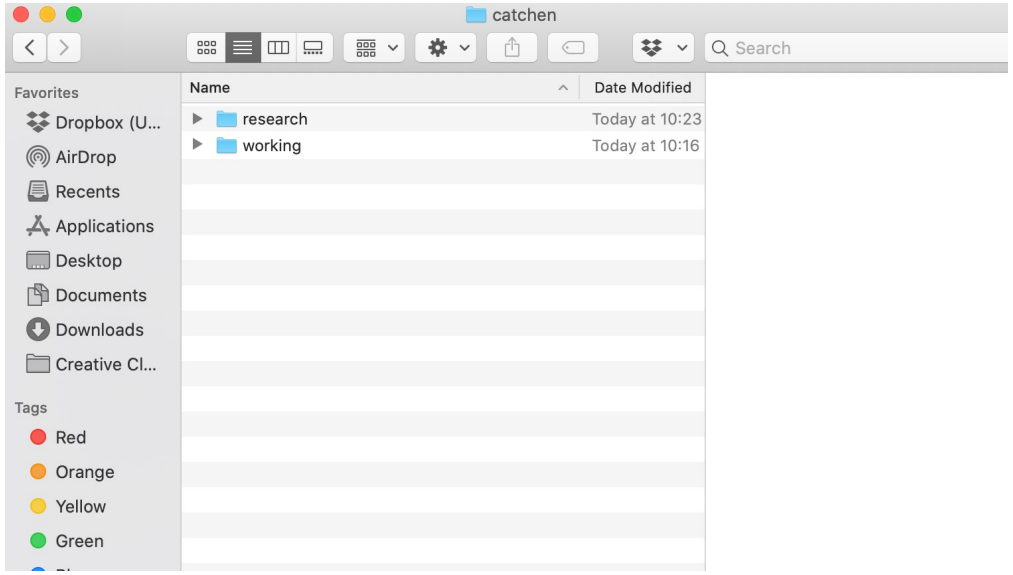


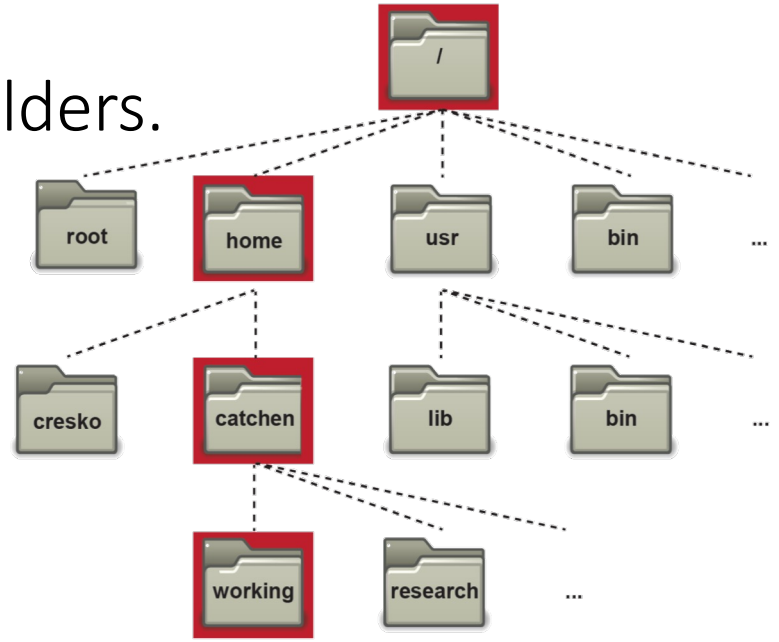
Terminal vs GUI

Don’t worry yet about how I am navigating to these folders.
Just focus on the layout!







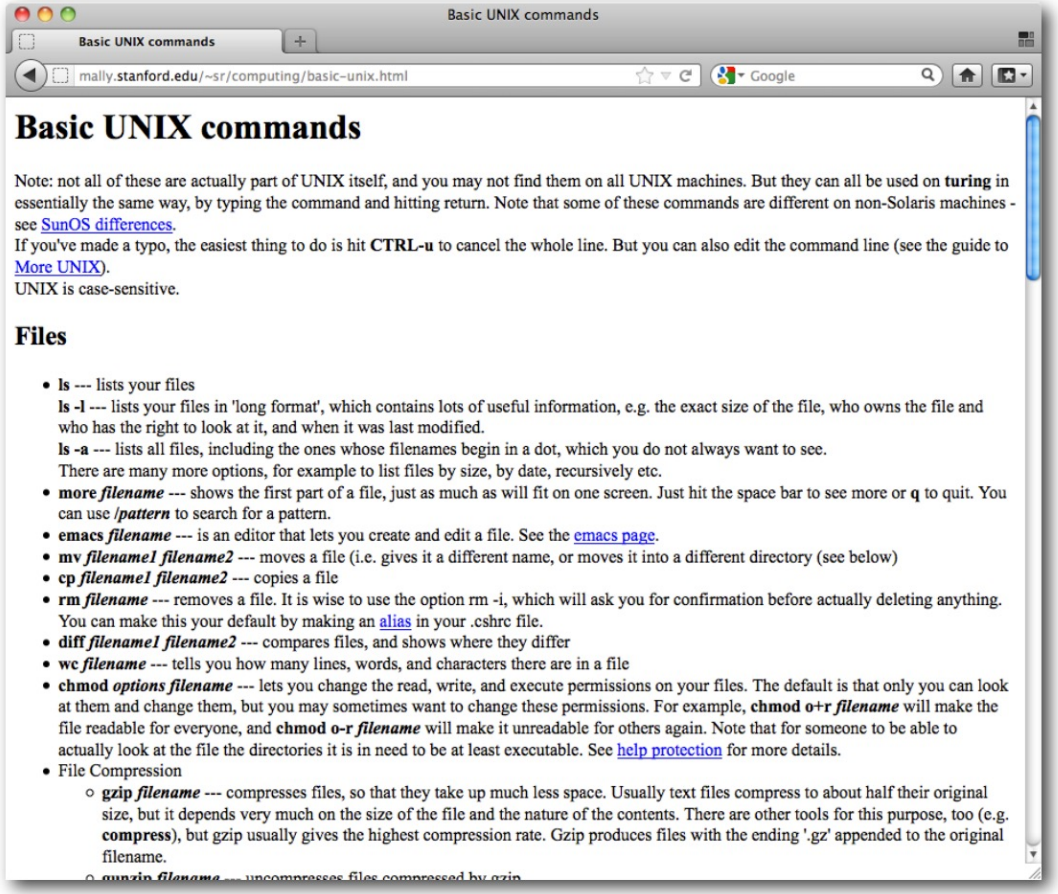


/home/catchen/working

Let's play around in the terminal

Step 1 in any new endeavor: Obtain a Cheat Sheet!

google "unix commands"



Basic UNIX commands

Note: not all of these are actually part of UNIX itself, and you may not find them on all UNIX machines. But they can all be used on **turing** in essentially the same way, by typing the command and hitting return. Note that some of these commands are different on non-Solaris machines - see SunOS differences.
If you've made a typo, the easiest thing to do is hit **CTRL-u** to cancel the whole line. But you can also edit the command line (see the guide to More UNIX).
UNIX is case-sensitive.

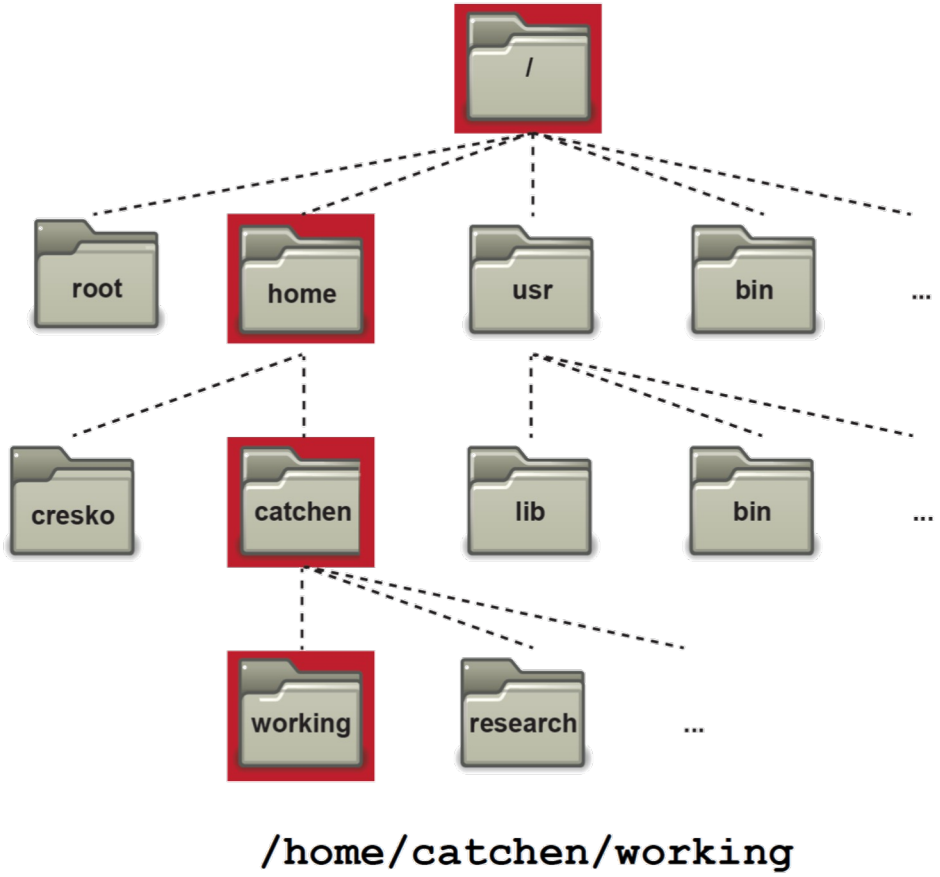
Files

- **ls** --- lists your files
  ls -l --- lists your files in 'long format', which contains lots of useful information, e.g. the exact size of the file, who owns the file and who has the right to look at it, and when it was last modified.
  ls -a --- lists all files, including the ones whose filenames begin in a dot, which you do not always want to see.
  There are many more options, for example to list files by size, by date, recursively etc.
- **more** ***filename*** --- shows the first part of a file, just as much as will fit on one screen. Just hit the space bar to see more or **q** to quit. You can use ***/pattern*** to search for a pattern.
- **emacs** ***filename*** --- is an editor that lets you create and edit a file. See the emacs page.
- **mv** ***filename1 filename2*** --- moves a file (i.e. gives it a different name, or moves it into a different directory (see below)
- **cp** ***filename1 filename2*** --- copies a file
- **rm** ***filename*** --- removes a file. It is wise to use the option rm -i, which will ask you for confirmation before actually deleting anything. You can make this your default by making an alias in your .cshrc file.
- **diff** ***filename1 filename2*** --- compares files, and shows where they differ
- **wc** ***filename*** --- tells you how many lines, words, and characters there are in a file
- **chmod** ***options filename*** --- lets you change the read, write, and execute permissions on your files. The default is that only you can look at them and change them, but you may sometimes want to change these permissions. For example, **chmod o+r** ***filename*** will make the file readable for everyone, and **chmod o-r** ***filename*** will make it unreadable for others again. Note that for someone to be able to actually look at the file the directories it is in need to be at least executable. See help protection for more details.
- File Compression
  - **gzip** ***filename*** --- compresses files, so that they take up much less space. Usually text files compress to about half their original size, but it depends very much on the size of the file and the nature of the contents. There are other tools for this purpose, too (e.g. **compress**), but gzip usually gives the highest compression rate. Gzip produces files with the ending '.gz' appended to the original filename.
  - **gunzip** ***filename*** --- uncompresses files compressed by gzip

Most common commands I use:

cd- ”change directory” primary command for moving between directories
mkdir- “make directory” create a new folder or directory
pwd- “print working directory” tells you where you are!
ls- “list” lists all the files in your current directory
cp- ”copy” makes a copy of a file
mv- “move” moves a files

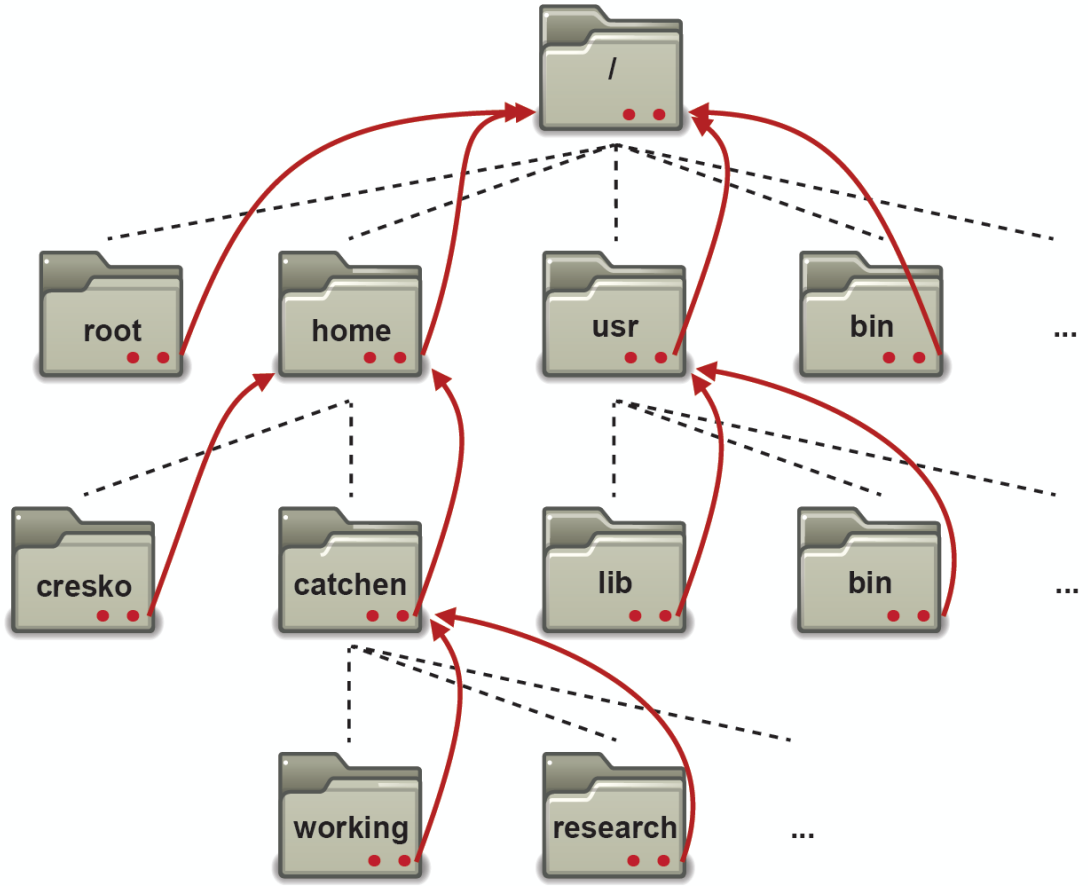
Moving forward in Easy!



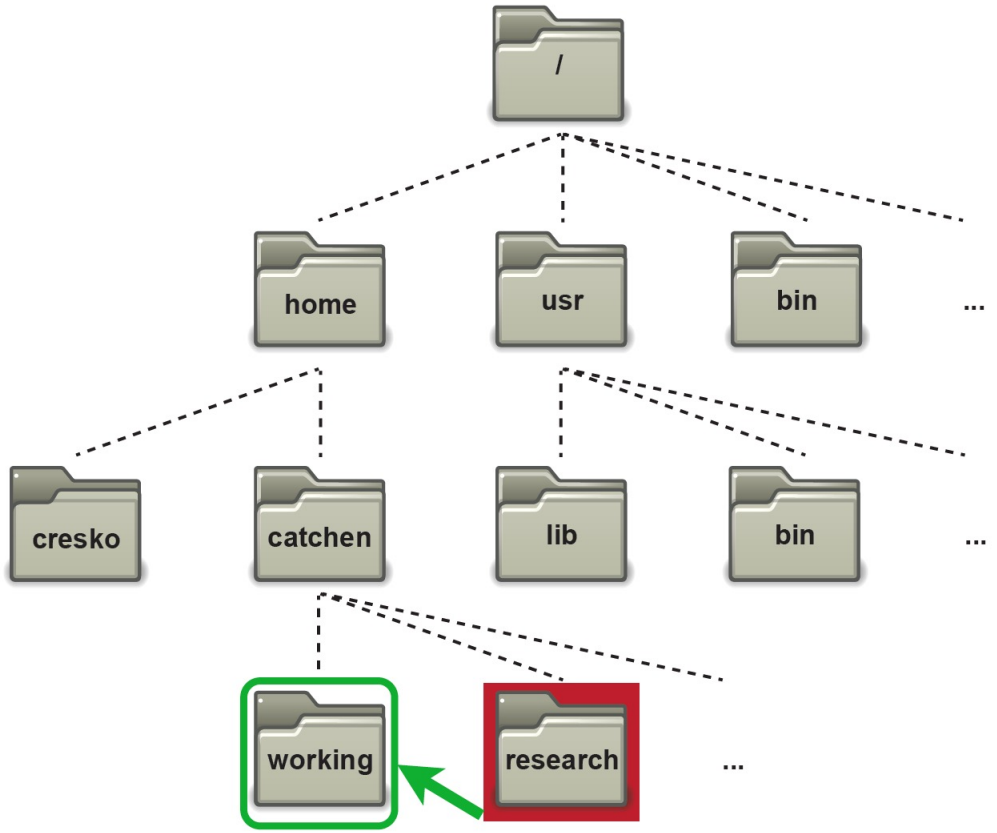
Option 1:
Pwd (where am I)
cd home
cd catchen
cd working

Option 2:
Pwd (where am I)
cd home/catchen/working

How do I move backward? Use *dot dot*



How would I move from *research* to *working*



Open OnDemand

Where am I? *Type pwd*
What else is here? *Type ls*

Interactive Exercise 1: Organizing Some Penguins

We are starting with one giant folder with a bunch of files. This can get messy quickly!

Exercise 1: Create a Directory in your home directory to store our data. Name it something useful like “Linux_Lecture_Exercises”

Exercise 2: Create two directories within this new directory “Raw_data” and “Working_directory” This will keep us safe if we make mistakes later or overwrite our files

Exercise 3: Navigate to the Directory “Mess_of_penguins” in the “Linux_Lecture” folder

Exercise 4: Copy our files from Mess_of_penguins” to our new directories “raw_data” and ”Working directory”

Interactive Exercise 1: Organizing Some Penguins

We are starting with one giant folder with a bunch of files. This can get messy quickly!

Exercise 1: Create a Directory in your home directory to store our data. Name it something useful like “Linux_Lecture_Exercises”

Exercise 2: Create two directories within this new directory “Raw_data” and “Working_directory” This will keep us safe if we make mistakes later or overwrite our files

Exercise 3: Navigate to the Directory “Mess_of_penguins” in the “Linux_Lecture” folder

Exercise 4: Copy our files from Mess_of_penguins” to our new directories “raw_data” and ”Working directory”

Tip #1: Save yourself a headache and avoid special characters
Using underscores in place of spaces makes things much easier

Raw_data vs Raw data

Are you typing? You are doing it wrong!
Tip #2: Tab Complete!

Tab-completion:

-Tab once to complete uniquely

-Tab twice to see all possible completions

Up-arrow:

-Find your most recently used commands

Want to look back at what you have already done?

Type: history

You can search for something specific in your history using ctrl + r in your history

Interactive Exercise 1: Organizing Some Penguins

We are starting with one giant folder with a bunch of files. This can get messy quickly!

Exercise 1: Create a Directory in your home directory to store our data. Name it something useful like “Linux_Lecture_Exercises”

Exercise 2: Create two directories within this new directory “Raw_data” and “Working_directory” This will keep us safe if we make mistakes later or overwrite our files

Exercise 3: Navigate to the Directory “Mess_of_penguins” in the “Linux_Lecture” folder

Exercise 4: Copy our files from Mess_of_penguins” to our new directories “raw_data” and ”Working directory”

Are you typing? You are doing it wrong! Tip #3: Moving groups of files

<u>Deciding on naming conventions is useful for several reasons</u>

If I name all my files in a structured way I can move them together using *

Example:

RADseq_PopulationOcean1_R1.fastq
RADseq_PopulationOcean1_R2.fastq
RADseq_PopulationOcean2_R1.fastq
RADseq_PopulationOcean2_R2.fastq

RADseq_PopulationFreshwater1_R1.fastq
RADseq_PopulationFreshwater1_R2.fastq
RADseq_PopulationFreshwater2_R1.fastq
RADseq_PopulationFreshwater2_R2.fastq

Are you typing? You are doing it wrong! Tip #3: Moving groups of files

<u>Deciding on naming conventions is useful for several reasons</u>

If I name all my files in a structured way I can move them together using *

Example:

RADseq_Ocean1_R1.fastq
RADseq_Ocean1_R2.fastq
RADseq_Ocean2_R1.fastq
RADseq_Ocean2_R2.fastq

RADseq_Freshwater1_R1.fastq
RADseq_Freshwater1_R2.fastq
RADseq_Freshwater2_R1.fastq
RADseq_Freshwater2_R2.fastq

If I want to move all my files I can type cp R*

Are you typing? You are doing it wrong! Tip #3: Moving groups of files

<u>Deciding on naming conventions is useful for several reasons</u>

If I name all my files in a structured way I can move them together using *

Example:

RADseq_Ocean1_R1.fastq
RADseq_Ocean1_R2.fastq
RADseq_Ocean2_R1.fastq
RADseq_Ocean2_R2.fastq

RADseq_Freshwater1_R1.fastq
RADseq_Freshwater1_R2.fastq
RADseq_Freshwater2_R1.fastq
RADseq_Freshwater2_R2.fastq

If I want to move all my ocean files I can type cp RADseq_O*

Are you typing? You are doing it wrong! Tip #3: Moving groups of files

<u>Deciding on naming conventions is useful for several reasons</u>

If I name all my files in a structured way I can move them together using *

Example:

RADseq_Ocean1_R1.fastq
RADseq_Ocean1_R2.fastq
RADseq_Ocean2_R1.fastq
RADseq_Ocean2_R2.fastq

RADseq_Freshwater1_R1.fastq
RADseq_Freshwater1_R2.fastq
RADseq_Freshwater2_R1.fastq
RADseq_Freshwater2_R2.fastq

If I want to move all my ocean files for a single population I can type cp RADseq_Ocean2*

Interactive Exercise 1: Organizing Some Penguins

We are starting with one giant folder with a bunch of files. This can get messy quickly!

Exercise 1: Create a Directory in your home directory to store our data. Name it something useful like “Linux_Lecture_Exercises”

Exercise 2: Create two directories within this new directory “Raw_data” and “Working_directory” This will keep us safe if we make mistakes later or overwrite our files

Exercise 3: Navigate to the Directory “Mess_of_penguins” in the “Linux_Lecture” folder

Exercise 4: Copy our files from Mess_of_penguins” to our new directories “raw_data” and ”Working directory”

Protecting yourself with chmod

Remember from *Good Data Practices* that we want to keep a raw copy of our data safe.

One way to add protection to to change the permissions on your raw data so they cannot be overwritten.

Chmod is a useful command for altering permissions

Chmod ### <file> is used to change permissions

First number: User permissions
Second number: Group permissions
Third number: Other user permissions

These different classes exist to give you fine control over who can view and manipulate your data

#	Sum	rwx	Permission
7	4(r) + 2(w) + 1(x)	rwx	read, write and execute
6	4(r) + 2(w)	rw-	read and write
5	4(r) + 1(x)	r-x	read and execute
4	4(r)	r--	read only
3	2(w) + 1(x)	-wx	write and execute
2	2(w)	-w-	write only
1	1(x)	--x	execute only
0	0	---	none

Chmod 777 means anyone can read, write, and execute your file.

What does Chmod 740 mean?

Interactive Exercise 1: Organizing Some Penguins

We are starting with one giant folder with a bunch of files. This can get messy quickly!

Exercise 1: Create a Directory in your home directory to store our data. Name it something useful like “Linux_Lecture_Exercises”

Exercise 2: Create two directories within this new directory “Raw_data” and “Working_directory” This will keep us safe if we make mistakes later or overwrite our files

Exercise 3: Navigate to the Directory “Mess_of_penguins” in the “Linux_Lecture” folder

Exercise 4: Copy our files from Mess_of_penguins” to our new directories “Raw_data” and ”Working directory”

Exercise 5: Change permissions on your “Raw_data” files do they cannot be overwritten

Now that I have my working directory let's learn how to view files in the terminal

`more`	`head`	`tail`	`cat`
view a text file one screen full at a time	view the top 15 lines of a file	view the last 15 lines of a file	spit the whole file at once
space-bar: scroll q: quit	-n num controls the number of lines	-n num controls the number of lines	

We will work with other regular expressions that can be used to pull specific parts of large data files in the next session

Interactive Exercise 2:

`more`	`head`	`tail`	`cat`
view a text file one screen full at a time	view the top 15 lines of a file	view the last 15 lines of a file	spit the whole file at once
space-bar: scroll q: quit	-n num controls the number of lines	-n num controls the number of lines	

How many columns are in penguins.csv?

What Island was the penguin from line 50 collected from?

How many species of penguins are represented in this file?

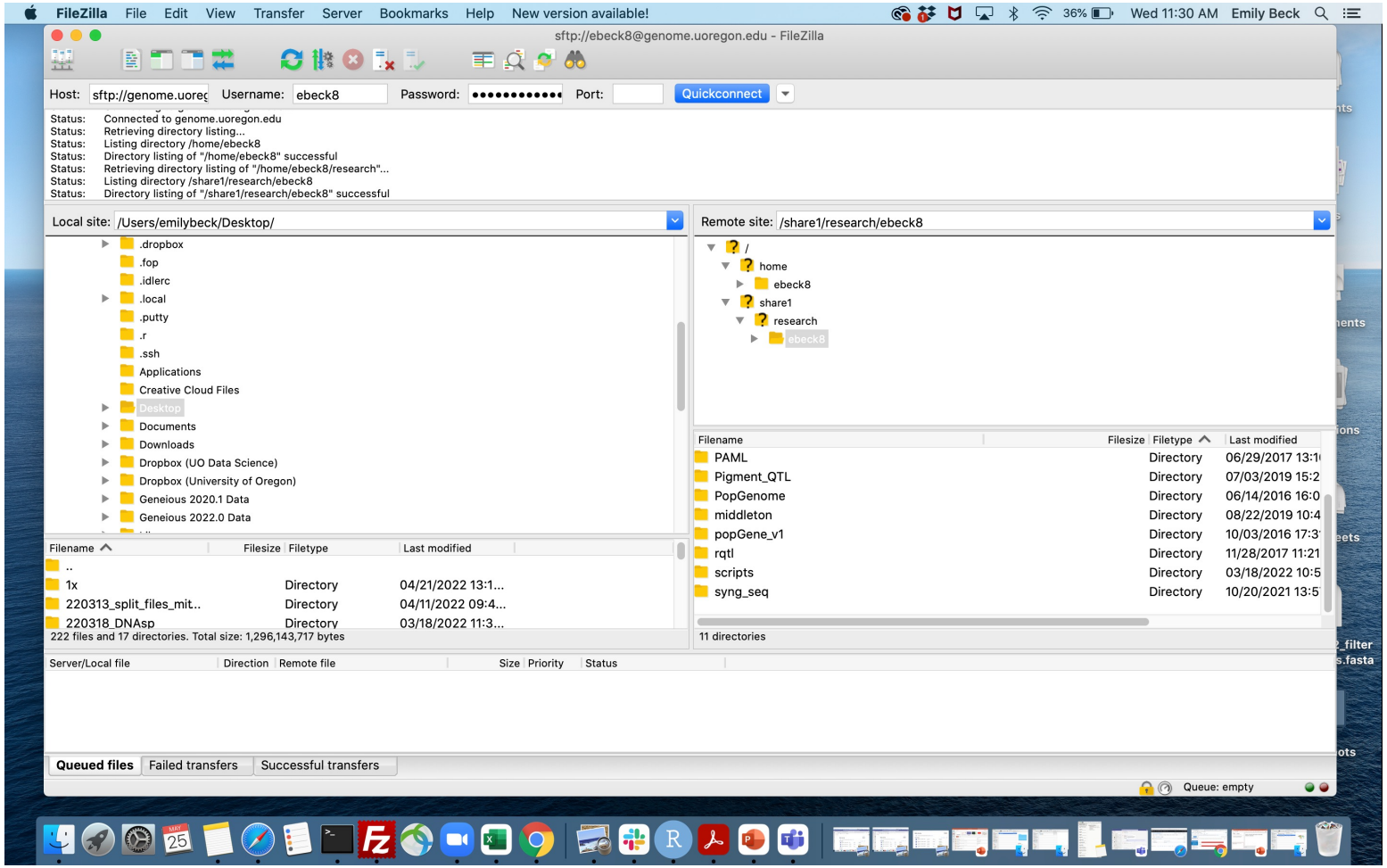
If you accidentally get “hung–up” in the terminal ctrl +c will reset you!

An alternate use for cat:

`more`	`head`	`tail`	`cat`
view a text file one screen full at a time	view the top 15 lines of a file	view the last 15 lines of a file	spit the whole file at once
space-bar: scroll q: quit	-n num controls the number of lines	-n num controls the number of lines	

Cat can be used to con**cat**enate files together cat file1 file2 > bigfile

PIVOT: Let's learn other ways to move files between computers Option 1: FTP



PIVOT: Let's learn other ways to move files between computers Option 2: scp in the terminal

Scp is a secure copy command. Use it to:

-transfer files from your local system to a remote system
-transfer files from a remote system to your local system
-transfer files between remote systems

General format:
scp -r [Source] [Destination]

Transferring a file from Jake's Desktop to Talapas:
scp ~/Desktop/my_file.csv jsearcy@talapas-ln1.uoregon.edu:~/


TIME FOR
A BREAK