Overview of Unix file system & directories:

- Finder/GUI file directory
    - What is a path?
        - Visualize file system structure in tree format (ppt slides 2&3)
            - Root → users → your home → your documents → fie of interest
            - Folder = directory

Navigating file systems in the terminal:

- Open Terminal in Mac/Linux (or Windows Subsystem for Linux if someone has a PC)
    - Prompt includes user & hostname, and current directory
    - Basic commands to be familiar with for navigating filesystem:
        - whoami - get username
        - Current directory is ~. What is that? Special path for home
            - pwd - where are you?
            - echo $HOME
        - What is in a directory?
            - ls - list directory contents
            - ls -lh
                - File permissions, user, group
            - ls -a
                - Hidden files
                - . and ..
        - Moving directories
            - cd
                - cd ../, cd /, cd ~, cd

Connecting to cluster:

- ssh netID@dcc-slogin.oit.duke.edu (make sure everyone can connect)
- Cluster file system (ppt slide 5)
    - Currently on a login node. Not a lot of memory, don't run programs on this node.
    - Get oriented: whoami, pwd
    - Home Directory = 10Gb storage space, so don't store large files here
    - cd /hpc/group/goldberg (demonstrate tab completion & using tab to search possible path options if unsure)
    - Group directory = 1TB storage (seems like a lot, but it's not. Shared amongst all of us in lab)
        - ls -lh to review group (goldberg) vs user (netID) again
    - /work directory (just show them it exists) has 450TB (temporarily 650TB) of unpartitioned space for ALL users of cluster. Good for work in progress & large temporary files -- FILES ARE DELETED AFTER 75 DAYS from the last time they were modified--do NOT store precious data here

Manipulating files and directories in the shell:

- mkdir dir1
- cd dir1, cd ../
- mkdir dir2 dir3

- cd dir1
- nano file1.txt
    - Write something, write out (save)
- cat file1.txt
- less file1.txt
- nano file2.txt
    - Write something new
- cat file1.txt file2.txt
- cp file2.txt file2_copy.txt
- rm file2_copy.txt (be careful with rm! There is no trash bin)
- mv file2.txt ../dir2 (can also use mv to rename a file)
- cd ../dir3
- cat ../dir1/file1.txt ../dir2/file2.txt > file3.txt
- echo “some text” >> file3.txt will append to end of file
- echo “some text” > file3.txt will write over file completely
- rm -rf dir* (* is a wildcard, go over --help to figure out what -rf does)

NEXT TIME:

Short review:

- Any q’s from last time or playing around on your own?
- Ssh to cluster
- How to make file? Nano. how to write over file? > how to append to file? >>
- New pipe: |
- history
- !{index} to run previous command
- history | grep cat (| pipes output from previous command as input for next command. grep searches for patterns or words of interest)

Running custom bash script:

- nano myscript.sh
- #!/bin/bash
  echo some text
- myscript.sh (command not found - not in $PATH)
- echo $PATH (will look for programs in these paths. Can modify $PATH or have to specify path)
- ./myscript.sh (permission denied!)
- ls -l myscript.sh (no x in user = not executable by me)
- chmod u+x myscript.sh
- ./myscript.sh (it works!)

Submitting jobs:

- Don’t want to do heavy computation on login nodes. Want to connect to one of the compute nodes.

- Submitting job = telling SLURM (job scheduler) to run your program on a compute node, based on specifications you give it. Specify memory, time, number of nodes, number of CPUs, partition, etc
- Interactive jobs
    - srun --pty bash -i
    - notice new host name, but same directories
    - Interactive jobs useful for debugging
    - exit (end job allocation, return to login node)
- Batch jobs (not interactive, submit job and it runs on node without you)
    - Specify partition with sbatch:
        - scavenger (allocated quicker, but low priority) vs common (takes longer in queue, but won't be cancelled by higher priority job)
        - sbatch -p scavenger ./myscript.sh
    - Look in slurm output file (errors AND output printed here)
        - sbatch -p scavenger -o "myscript.out" ./myscript.out
    - #SBATCH options (e.g. -p, --mem, -o)
        - nano myscript.sh
        - sbatch myscript.sh
        - squeue -u netID (see queued jobs)
        - squeue -j jobID
        - seff jobID (useful for determining how much memory or CPU you really need)
    - DCC is a shared resource, so try not to request job allocations with more memory or time than you really need
- Job arrays
    - nano myscript.sh
    - #!/bin/bash
      #SBATCH -p scavenger
      #SBATCH --mem=1
      #SBATCH --job-name="myscript"
      #SBATCH -a 1-3

      c=$SLURM_ARRAY_TASK_ID

      echo this is array number $c

      sleep 20s
    - sbatch ./myscript.sh
    - ls (one output for each array)
    - cat slurm*
    - sbatch -a 10-11 ./myscript.sh (options at the command line supercede #SBATCH options in script)

Copying files from/to cluster (scp, Cyberduck):

- exit to return to local computer
- To PULL from cluster:
    - scp netID@dcc-slogin.oit.duke.edu:~/myscript.sh .
    - nano test.txt
- To PUSH to cluster
    - scp test.txt netID@dcc-slogin.oit.duke.edu:~/newfile.txt (giving copy a new name)
    - ssh back to cluster, see newfile there
- Cyberduck (https://cyberduck.io/)
    - scp is usually faster, but cyberduck can be convenient for navigating the cluster filesystem, viewing files remotely, and editing scripts on your local computer in your text editor of choice without having to download and upload yourself (e.g. vscode, sublime, atom)

TO SHARE:
https://practicalcomputing.org/files/PCfB_Appendices.pdf (appendix 3 is very useful for descriptions of commonly used shell commands)

Practical Computing for Biologists is a very good resource for introduction to scientific programming, if you can get it from the library or want to invest in buying it.

https://rc.duke.edu/dcc/cluster-storage/ description of DCC filesystem

https://rc.duke.edu/dcc/dcc-user-guide/ DCC user guide

https://cyberduck.io/ for Cyberduck