Working on remote machines

fjord (or whatever remote linux machine you got an account on)

- You should all have gotten accounts on fjord from David Darr.
- To login to your account from the linux prompt on your home computer use "ssh" (secure shell)
- ssh [username]@fjord.ocean.washington.edu
- and it will ask for your password. Now, even though you are in the same terminal window your commands are being carried out in a room inside OSB! All the same linux-bash commands should work basically the same as on your laptop.

fjord, cont.

- There are many disks and folders and users on fjord, but you only have permission to do things (like create or delete files) in two places:
- /home/[username] is where you keep environment stuff like the .bashrc where you set aliases and default paths (like where to find ipython). This location has very little space, so don't do ANY work there except for editing your .bashrc.
- /data1/effcom/[username] is a much bigger disk, with many TB of space. You can use this for working with files and code for this class.

Essentials: ssh, scp

- "ssh" is how you log on. Type "logout" to get off fjord and back to your machine. One of the tricky things about working across machines is remembering where you are currently. Make a note for yourself on how the prompt changes between the two machines.
- To move a file from your computer to fjord, you can use a file transfer program like Transmit or WinSCP. Often it can be comforting to have the usual graphical folders and files that you click and drag. But you can also do it from the command line using "scp".
- Working from your home terminal, find a small file to copy, and navigate to the directory where it is, then try:
- `scp [filename] [username]@fjord.ocean.washington.edu:/data1/effcom/[username]`
- And then check to see if it got there. If you had used .../[username]/[newfilename] it would have renamed the file in the process.
- You can transfer lots of files this way by using wildcards.
- During the transfer it will tell you how fast it is going. What is your typical "upload" speed? I get around 10-100 MB/sec downloads and 3 MB/sec uploads at home.

Disk Usage: df

- **ALWAYS be aware of how much disk space you are using and how much is available. If you fill up /data1 no one else will be able to use it. ☹**
- "df -h" gives you info on how much disk space is available on all disks mounted on the computer. Here is what fjord currently looks like:

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        96G   78G   14G  86% /
tmpfs            32G     0   32G   0% /dev/shm
/dev/sda1       477M  153M  300M  34% /boot
/dev/sdb1        22T   16T  6.0T  73% /data1
gaggle:/pmr3     30T   14T   16T  47% /pmr3
gaggle:/pmr4     30T   21T  8.9T  70% /pmr4
boiler:/data1    73T   41T   32T  57% /boildat1
perigee:/data1  146T  116T   30T  80% /pgdat1
gaggle:/pmr2     33T   19T   15T  57% /pmr2
```

- So /data1 has 6 TB available and is 73% full.

Disk Usage: du

- To find out how much diskspace you are using in various folders use "du -sh *". The "s" means it gives a summary of the size of each folder, not each file, and as before "h" means human-readable. It gives info specific to all (*) the files and folders at your current location, not everywhere, so it is the tool to use for your own housekeeping.

```
[parker@fjord output]$ du -sh *
5.0T cas6_v3_lo8b
4.7T cas6_v3_lo8da
1.2T sj0_v0_lo8nest
255G WCOFS_avg_Exp37
120G wcofs_avg_now
```

Job Control: &, CTRL-z, bg

- A really common problem when working on a remote machine is that you start some long calculation and then are stuck having to keep the connection open. You don't have to do this – linux is designed to keep working, even on multiple jobs, after you logoff.
- Here is a nice way to run remote jobs, working from the remote machine:
- python [my_program].py > log.txt &
- Here I run a python program, redirect (>) the screen output – like my print statements – to a file called log.txt, and then (&) "escape to shell". When you start a program this way it will return you to the command line prompt and tell you the Process ID (PID) number. Then you can safely logoff and come back later.
- If you start a big job but forget the "&" no worries. Just type CTRL-z (meaning hold down the control key and at the same time hit z) to suspend the job, and then "bg" to send the job to the background. Try it out so you feel comfortable with it. The command “fg” brings the job back to the foreground. A simple python test program could just be something like:

```
import time
print('starting')
time.sleep(10) # sleep for 10 seconds
print('done')
```

Job Control: top, kill

- "top -u [username]" will bring up a screen of all your running processes, including the PID of each one. The various columns can be hard to understand (for me) but it does allow me to see what jobs are running.
- Type “q” to get out of top.
- "kill [PID]" will terminate any of your jobs. Don't worry – you can't mess up things being run by other users.

Editing text files

- To edit a text file remotely you can't use the same graphical text editor you use to write code on your personal machine. In fact it is such a pain to edit code remotely I recommend that you do ALL your coding on your personal computer, and then use git (best) or scp to put it on the remote machine. Next week we'll talk through strategies for organizing your code to streamline this process.
- To edit files on remote machines the standard workhorse is "vi" (or "nano") which you invoke from the command line like:
- `vi [filename]`

vi commands 1

- start editing
  - vi filename
- save changes and quit (:q! to quit without saving changes)
  - :wq
- insert before cursor, on cursor, or after cursor. **You HAVE to use one of these before you can start typing.**
  - i, s, a
- escape from insert mode
  - ESC
- add new line below or above (these also allow you to start typing)
  - o, O
- delete 2 lines
  - 2dd

vi commands 2

- from line 1 to end of file ($), substitute (s), all instances on line (g)
  - :1,$s/old_pattern/new_patern/g
- from this line (.) and following 4 lines (.+4), substitute (s), all instances on line (g)
  - :.,.+4s/old_pattern/new_patern/g
- to include "/" in a pattern use \/
- yank one line into buffer a
  - "a1yy
- insert buffer a below cursor
  - "ap

vi commands 3

- see line numbers (nonu to remove)
  - :set nu
- see hidden characters (nolist to hide)
  - :set list
- get rid of highlights (caused by shift 3)
  - :noh

Exercise

- Use vi to create a short text file, with a few lines of text. Close the file (:wq)
- Edit the file using vi to add a new line of text (o) and delete another (dd).
- Edit your .bashrc on fjord to include an alias that gets you to your directory in /data1. Don't worry – we can always fix it if needed.
- Anytime you get lost just hit the Escape key and then type :q! to exit vi without saving changes.

cron

- The last linux tool I find really useful is "cron" which allows you to schedule a job to happen at a certain time, say every day at 1 AM.
- "crontab -e" will allow you to begin editing the list of jobs. **You use vi commands to do this.**
- "crontab -l" will list your current cron jobs.

cron example

- Here are the current cron jobs for my daily forecast model:

```
[parker@boiler ~]$ crontab -l
LOd="/data1/parker/LiveOcean/driver/"
10 00 * * * cd $LOd && ./driver_forcing2.sh -g cas6 -t v3 -f ocn4 -r forecast > ./dlog_638_ocn4
20 00 * * * cd $LOd && ./driver_forcing2.sh -g cas6 -t v3 -f tide2 -r forecast > ./dlog_638_tide2
30 00 * * * cd $LOd && ./driver_forcing2.sh -g cas6 -t v3 -f riv2 -r forecast > ./dlog_638_riv2
30 02 * * * cd $LOd && ./driver_forcing2.sh -g cas6 -t v3 -f atm1 -r forecast > ./dlog_638_atm1
00 05 * * * cd $LOd && ./driver_post.sh -g cas6 -t v3 -x lo8b -r forecast > ./dlog_638_driver_post
00 20 * * * cd $LOd && ./forecast_cleaner.sh
```

- So for example at 2:30 AM (oddly written as 30 02 * * *) I run the shell script to make the atmospheric forcing for the model.
- Creating your own cron jobs just consists of creating a text file (using crontab -e) with lines like these.
- Here is a very short cron cheatsheet:
  - https://devhints.io/cron