HPC For NonComputer Scientists

Linux

Here we will go over the important commands of Linux we will use in this course:

pwd

- 'Print working directory'
- tells which directory we are currently in

ls

- lists contents of a directory

man

- helps give us all the information of using a command
  - Any flags that we can use with the command we're interested in
- ex) 'man ls' will give everything we need for the ls command

cd

- helps move from one directory to another
- ex) 'cd projects/mooc/'
- 'cd ..' would take us back up a folder
- 'cd ~' or 'cd' will take us back to our home directory

mkdir

- creates a new directory
- 'mkdir test' creates a directory named test

touch

- creates a new file (same syntax as makedir)

mv

- moves a file to another directory
- 'mv test.txt test' movest the text file to the test directory
- can also use to rename a file
- 'mv test.txt test1.txt' changest test.txt to file name test1.txt

cp

- copies a file
- 'cp text1.txt text2.txt'
- to cop a directory we need to use -R flage

rm

- remove a file
- use -R flag to remove a directory
- Will be gone forevery once it's removed

Accessing Remote Systems

- A remote system is one that you are accessing from another computer
- Unless you have built a cluster at home, or work in an HPC center, most HPC systems will require remote access
- Two ways one interacts with a remote system
  - Logging in
  - File transfer

Logging in

- Generally, one uses an ssh protocol to login to a remote system.
- Provides a secure channel over which one can remotely connect
- Authenticate connection through keys, public and private
- Example: ssh username@remote.hostname
- Might have some flags after the ssh
  - -X is important
    - Enables X11 forwarding
    - If we have an image from a remote system that we want to use on the computer in front of us

File Transfer

- Recommend several ways
  - Depends on your needs and size of data
  - scp, sftp, wget, rsync, Globus file transfer
  - scp and sftp are good because they are secure
- Ex) scp /home/udername/file.txt username@remote.hostname:/home/username
- scp username@remote.hostname:/home/username/file.txt .
  - transferrs from the home directory to the last listed directory.

File Systems

Typical Types of Files

- Three types of storage spaces users are typically allocates on HPC infrastructure
  - Home
  - Projects or Work
  - Scratch
- Each space is important for different reasons, and understanding the difference between each is imperative

Home

- /Home is intended for the use of the owner of this space only
- It's found at /home/$USER or ~
- Usually this space is backed up
- Also, generally allocated a small amount of space -on the order of 5 GB, varies
- Usually where you land when you login
- Test: login, type pwd
- Types of files kept in
  - Scripts
  - Code
  - Very Small Files
  - Inappropriate for sharing files with others
  - Inappropriate for job output

Projects or Work

- Generally a space for mid-level sized data
- Might have approximately 250-500 GB of space available
- Sometimes backed up
- For us: /projects/$user
- type: cd/projects/$USER
- Types of files kept in
  - Codes, files, libraries relevant for any software you are installing (if wanting to share)
  - Mid-level size input files
  - Appropriate for sharing files with others
  - Inappropriate for job output

Scratch

- Scratch space is available on most HPC systems
- Usually, a much larger space
- Temporary space
- Usually not backed up
- type cd/scracth/$USER
- Types of files kept in
  - Output from running jobs
  - Large files
  - Appropriate for sharing files with others
  - THIS IS NOT APPROPRIATE FOR LONG TERM STORAGE

Bash Scripting

What is a shell?

- A shell is the enviroment in which commans are interpreted in Linux
- GNU/Linux provides various shells; bash most popular
  - sh
  - csh
  - tcsh
  - ksh
- Shell scripts are files containing collections of commands for Linux systems that can be executed as programs
  - contains commans so we don't have to use them over and over again

Bash Script

- to create a bash script file, the first line must be
  - #!/bin/bash
- Program loader recognizes the #!, and the /bin/bash part tells the interpreter which shell should be run
- Command Examples
  - "echo "Hello!" > file.out' prints the statement
  - 'echo "Hello!" >> file.out' pends the statement at the bottom of the file

Variables

- Shell variables are local
- Enviroment variables are global
  - Contain data that are used by one or more applications
- Several pre-defined enviroment variables in your container
  - Type "env" in the terminal window
- Examples
  - name=(CU Boulder). This creates the avariable name and places it in an arrray
    - echo ${name[0]}. This prints out the first variable of the 'name' array.
  - $Home
    - echo $HOME
      - This prints out the HOME variable
  - var=$(pwd)
    - This sets a command to a variable
    - echo$var
    - The dollar sign prints out the output of that variable

Loops

While Loop

x=0

- while [\ $x -lt 10]; do
  - echo $x
  - x = $(($x+1))
- done

For loop

- list=(a b c)
- for v in ${list[@]}; do
  - echo $v
- done

Permissions

- Before you can run a script, you need to make sure the script has the appropriate permissions
- At the command line, type, "ls -l"
- Column 1: Permissions
  - d, r, w, x
    - Tells if directory and if that file has read, write or executable permissions for a certain group of people (position of the type of people listed below)
  - Owner, group, global
- Chmod changes permissions
  - chmod +x filename.sh
  - Makes the file exectuable for everyone
- Run it using ./filename.sh
- Could also have done bash filename.sh and avoided permissions