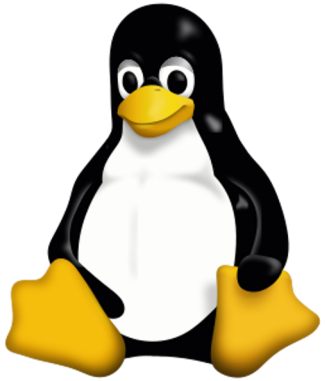
SeaWulf basics: High Performance Computing and Linux for researchers

Dave Carlson & Rebecca Drucker

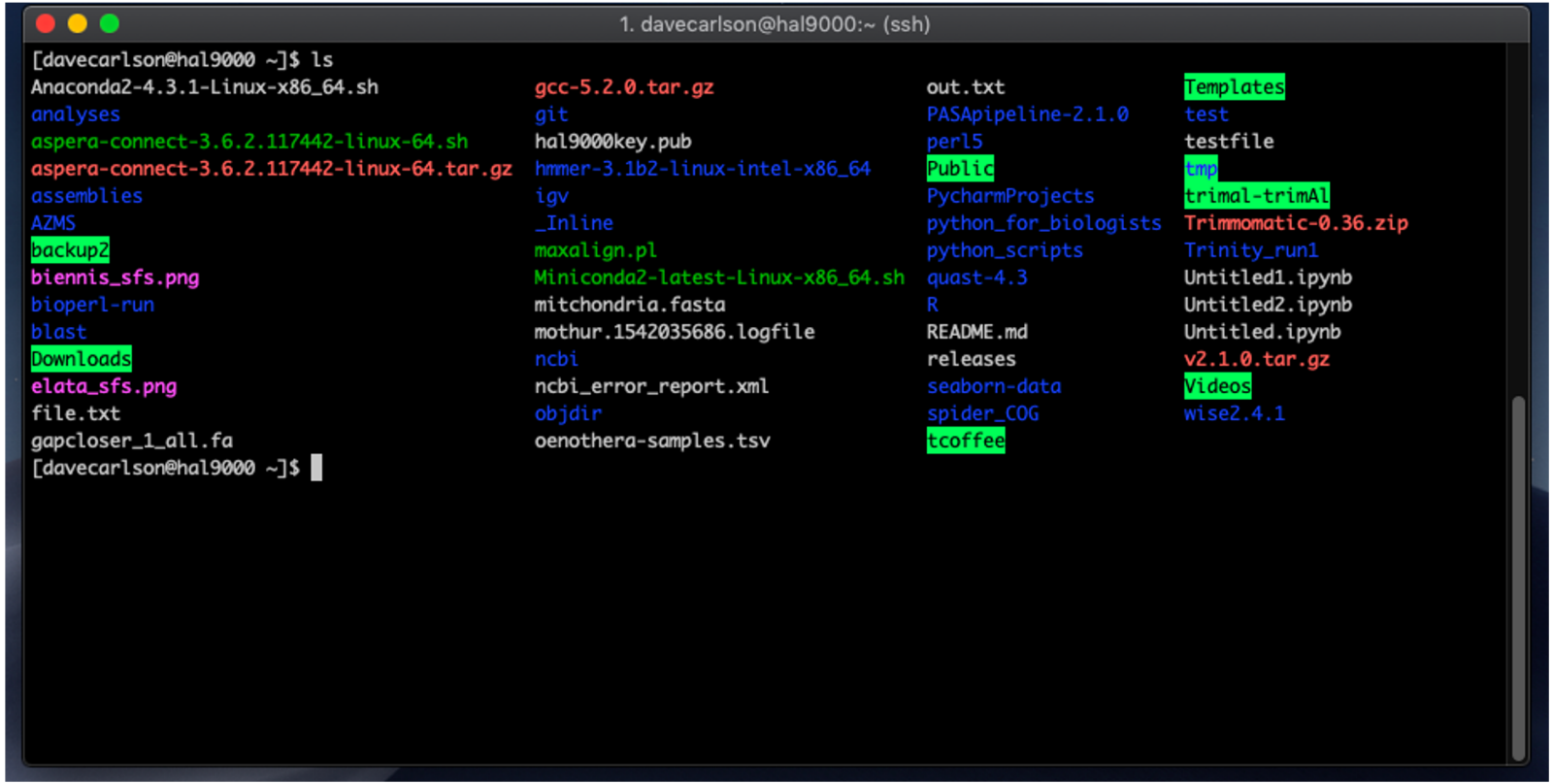
March 2nd & 3rd, 2020

What is Linux?

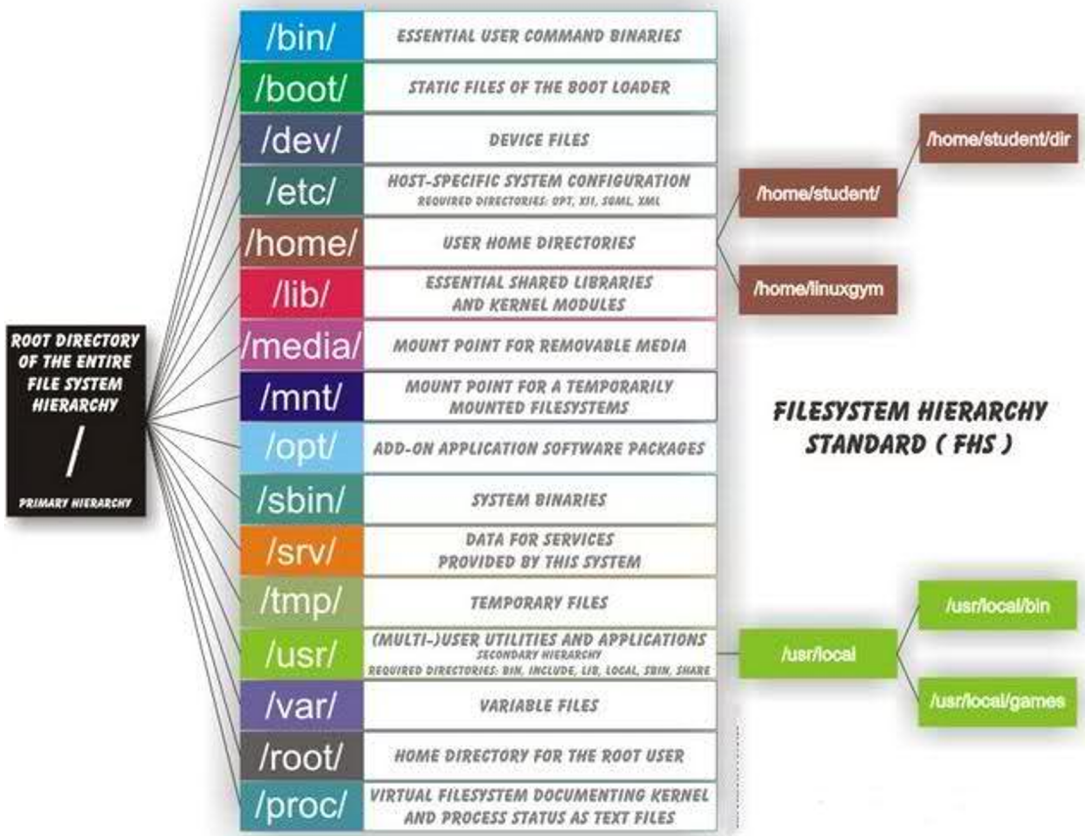


- A free, open-source operating system developed by Linus Torvalds and the GNU Project
- An alternative to the UNIX OS developed by Bell Labs in the 19070s
- The most popular OS in the world (thanks to Android)
- A very flexible OS with many different varieties
  - **Ubuntu, CentOS, Debian, Fedora**, etc.

The terminal interface: a way to interact with the “shell”



The hierarchical file system



https://www.tecmint.com/linux-directory-structure-and-important-files-paths-explained/

Paths

A **Path** = a map that tells the OS where to find a file or directory

Paths

An **Absolute Path** = a map that gives directions *starting from the root directory* to the desired location or file

E.g.,
/home/dave/hpc_workshop/welcome.sh

Paths

A **Relative Path** = a map that gives directions to the desired location or file *relative to the current directory*

E.g., ./dave/hpc_workshop/welcome.sh

./ = “start from current sub-directory”

../ = “start from one sub-directory above the current one”

../../ = “start from two sub-directories above the current one”

etc.

Paths

The **PATH** = an *environment* variable that tells the OS where to look for executable files

- If an executable file is located in a directory found in the PATH, no path information needed to execute it
- See what's in your PATH:
  - "echo $PATH"

Simple commands

- **cd** = “change directory”
- **pwd** = “print name/path of current directory”
- **ls** = “list directory contents”
  - Many useful variants!
- **mkdir** = “make directory”
- **mv** = “move” (or “rename”!)
- **cp** = “copy”
- **rm** = “remove” (be careful!)
- **touch** = “create or update a file”
- **wc -l** = “count # of line in file”

https://files.fosswire.com/2007/08/fwunixref.pdf

Unix/Linux Command Reference

FOSSwire.com

File Commands

`ls` – directory listing
`ls -al` – formatted listing with hidden files
`cd dir` - change directory to *dir*
`cd` – change to home
`pwd` – show current directory
`mkdir dir` – create a directory *dir*
`rm file` – delete *file*
`rm -r dir` – delete directory *dir*
`rm -f file` – force remove *file*
`rm -rf dir` – force remove directory *dir* *
`cp file1 file2` – copy *file1* to *file2*
`cp -r dir1 dir2` – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
`mv file1 file2` – rename or move *file1* to *file2* if *file2* is an existing directory, moves *file1* into directory *file2*
`ln -s file link` – create symbolic link *link* to *file*
`touch file` – create or update *file*
`cat > file` – places standard input into *file*
`more file` – output the contents of *file*
`head file` – output the first 10 lines of *file*
`tail file` – output the last 10 lines of *file*
`tail -f file` – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

`ps` – display your currently active processes
`top` – display all running processes
`kill pid` – kill process id *pid*
`killall proc` – kill all processes named *proc* *
`bg` – lists stopped or background jobs; resume a stopped job in the background
`fg` – brings the most recent job to foreground
`fg n` – brings job *n* to the foreground

File Permissions

`chmod octal file` – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
`chmod 777` – read, write, execute for all
`chmod 755` – rwx for owner, rx for group and world
For more options, see `man chmod`.

SSH

`ssh user@host` – connect to *host* as *user*
`ssh -p port user@host` – connect to *host* on port *port* as *user*
`ssh-copy-id user@host` – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

`grep pattern files` – search for *pattern* in *files*
`grep -r pattern dir` – search recursively for *pattern* in *dir*
`command | grep pattern` – search for *pattern* in the output of *command*
`locate file` – find all instances of *file*

System Info

`date` – show the current date and time
`cal` – show this month's calendar
`uptime` – show current uptime
`w` – display who is online
`whoami` – who you are logged in as
`finger user` – display information about *user*
`uname -a` – show kernel information
`cat /proc/cpuinfo` – cpu information
`cat /proc/meminfo` – memory information
`man command` – show the manual for *command*
`df` – show disk usage
`du` – show directory space usage
`free` – show memory and swap usage
`whereis app` – show possible locations of *app*
`which app` – show which *app* will be run by default

Compression

`tar cf file.tar files` – create a tar named *file.tar* containing *files*
`tar xf file.tar` – extract the files from *file.tar*
`tar czf file.tar.gz files` – create a tar with Gzip compression
`tar xzf file.tar.gz` – extract a tar using Gzip
`tar cjf file.tar.bz2` – create a tar with Bzip2 compression
`tar xjf file.tar.bz2` – extract a tar using Bzip2
`gzip file` – compresses *file* and renames it to *file.gz*
`gzip -d file.gz` – decompresses *file.gz* back to *file*

Network

`ping host` – ping *host* and output results
`whois domain` – get whois information for *domain*
`dig domain` – get DNS information for *domain*
`dig -x host` – reverse lookup *host*
`wget file` – download *file*
`wget -c file` – continue a stopped download

Installation

Install from source:
`./configure`
`make`
`make install`
`dpkg -i pkg.deb` – install a package (Debian)
`rpm -Uvh pkg.rpm` – install a package (RPM)

Shortcuts

`Ctrl+C` – halts the current command
`Ctrl+Z` – stops the current command, resume with `fg` in the foreground or `bg` in the background
`Ctrl+D` – log out of current session, similar to `exit`
`Ctrl+W` – erases one word in the current line
`Ctrl+U` – erases the whole line
`Ctrl+R` – type to bring up a recent command
`!!` - repeats the last command
`exit` – log out of current session

* use with extreme caution.

Exercise 1

1. In your current working directory, create a new directory called “HPC”.
2. Change to this directory and create a file called “test1.txt”
3. Rename this file to “test.txt”
4. Copy the file to your home directory

Exercise 2

Access workshop files

If you already have a SeaWulf account:

1. Log into seawulf
2. In your home directory, do “mkdir hpc_workshop”
3. Do “cp /gpfs/projects/samples/hpc_workshop_030220/* ~/hpc_workshop”

If you don’t yet have a SeaWulf account:

1. Download the files at **https://bit.ly/38albFJ**
2. Open your terminal and navigate to the directory containing the files (e.g., ~/Downloads/)

Five ways to view a text file

more	less	head	tail	cat
view a text file one screen full at a time	view a file one screen full at a time	view the top 10 lines of a file	view the last 10 lines of a file	spit the whole file at once
space-bar: scroll q: quit	-S no text wrapping; can read gzipped files	-n num controls the number of lines	-n num controls the number of lines	

(Julien Catchen)

An interlude about Pipes

A pipe (|) takes output from one command and uses it as input for another command:

Syntax :

```
command_1 | command_2 | command_3 | .... | command_N
```

```
[decarlson@login ~]$ ls -l | head
total 162304
-rw-------   1 decarlson decarlson        26 Sep 10 10:32 1_fs.e
drwx------+  2 decarlson decarlson      4096 Jun 25 10:26 28S_dictyoceratida
-rw-------   1 decarlson decarlson    235036 Jun 25 10:25 28S_dictyoceratida.tar.gz
-rw-------   1 decarlson decarlson       551 Nov  6  2017 Anaconda2
drwx------+  2 decarlson decarlson      4096 Oct  4  2017 awk_tutorial
drwx------+  3 decarlson decarlson      4096 Jan 18  2018 biopython
-rwx--x--x   1 decarlson decarlson       337 May  3 11:54 convertFASTA.sh
-rw-------   1 decarlson decarlson       557 May 31 10:32 cuda.c
drwx------+  2 decarlson decarlson      4096 Sep  6  2017 Desktop
```

Very useful for stringing together a series of commands to create a pipeline

Can save the output of a series of commands as a new file: **ls -l | head > files.txt**

More Useful Commands

grep

- Prints li
- Can se
  - ^ =
  - $ = end of line
- Example: **Find “battery” crimes in chicago_crime_stats.csv:** grep BATTERY chicago_crime_stats.csv
- Example: **Find every crime that occurred on Lake Shore Drive:** grep “LAKE SHORE DR” chicago_crime_stats.csv

****Bioinformatics bonus!****

Get all headers in fasta file:

```
cat genome.fasta | grep "^>"
```

More Useful Commands

cut

- Remove a section from each line of a file
- Great for working with tab-separated data!
- Can choose what delimiter to use with the “- d” argument
- Examples
  - **Return the first column from chicago_crime_stats.csv :** cut -d "," -f 1 chicago_crime_stats.csv
  - **Return the first and third column from chicago_crime_stats.csv :** cut -d “,” -f 1,3 chicago_crime_stats.csv
  - **Find the dates of all thefts from chicago_crime_stats.csv:** grep THEFT chicago_crime_stats.csv | cut -d "," -f 3

More Useful Commands

sed: search and replace text information (and more!):

Basic syntax: s/pattern/replace/

Examples:

Replace the file extensions in a list of files: cat file_list.txt | sed 's/csv/tsv/g’

Replace every instance of“HANDGUN” with “SQUIRTGUN” cat chicago_crime_stats.csv | sed ‘s/HANDGUN/SQUIRTGUN/g’

Starting with the 2nd line, print every other line to screen: cat chicago_crime_stats.csv | sed -n '2~2p'

More Useful Commands

sort & uniq

Sort: ar

Uniq: re

- sh

Example

Bioinformatics bonus!

Get histogram of variant allele counts from vcf:

```
cat sample.vcf | cut -f 8 | cut -d ";" -f 1 |
sed 's/AC=//g' | sort | uniq -c
```

Arrange file in numerical order based on specific column: cat ip-sectors.txt | sort -n -k 2,2

Remove duplicate entries in a list: cat food.txt | sort | uniq

More Useful Commands

awk: "a programming language for processing regularly formatted text"

(https://www.li

Ba

Examples:

Get the first column from chicago_crime_stats.csv: awk -F, '{print $1}' chicago_crime_stats.csv

Count all instances where an arrest has been made: awk -F, '{if ($9=="true") count++$9} END {print count}' chicago_crime_stats.csv

****Bioinformatics bonus!****

Get list of gene IDs in GFF file:

```
cat sample.gff | awk '$3=="gene" {print $9}' |
cut -d ";" -f 1 | sed 's/ID=//g'
```

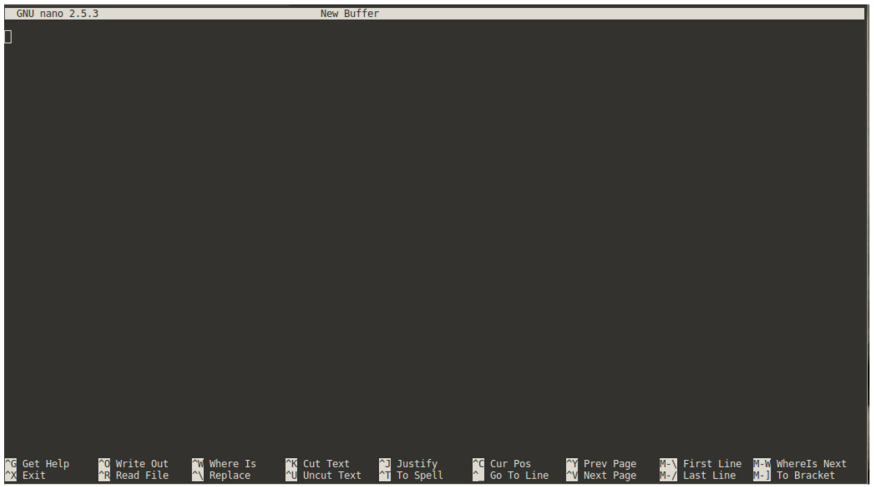
Exercise 3

1. View the “ip-sector.txt” file using the command of your choice
2. Count how many years of data are present
3. Return just the data from years in the 20th century
4. Turn ”ip-sector.txt” into a comma-separated file (hint: the symbol for a tab is “\t”)
5. What is the total volume of textiles production across all years (hint: trying using the “sum” function in awk)?

Editing text files and creating scripts

Many useful text editors for linux

- nano, vim, etc.

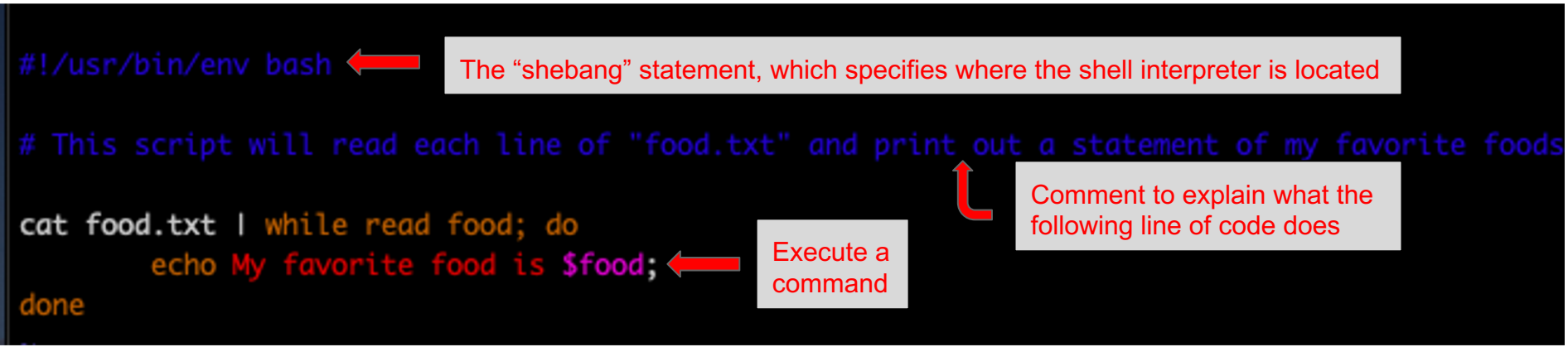


Editing text files and creating scripts

Why should you put your analyses in a shell script?

- Better reproducibility
- Easier to remember what you did
- Can use comments and multiple lines to increase legibility
- Complex tasks require a script!

Anatomy of a shell script



Anatomy of a shell script

```
#!/usr/bin/env bash

# Create variable that we will use later
WatchMe=$(echo whip nae nae)

# This is where I whip
echo Now watch me "$(echo $WatchMe | cut -d " " -f 1)"

# This is where I nae nae
echo Now watch me "$(echo $WatchMe | cut -d " " -f 2,3)"
```

The “shebang” statement, which specifies where the shell interpreter is located

Comment to explain what the following line of code does

It’s useful to save the output of a command as a variable so we can work with it later

Execute a command

Executing a shell script

1. Edit the text file (nano, vim, etc.)
2. Make it executable
3. Run it!!

```
dave@dave:~/hpc_workshop$
dave@dave:~/hpc_workshop$ chmod +x example_script.sh
dave@dave:~/hpc_workshop$ ./example_script.sh
Now watch me whip
Now watch me nae nae
dave@dave:~/hpc_workshop$
```

LOOPS

For loop

Example: for number in {1..100}; do expr 1 + $number; done

Example: for file in $(ls); do extension=$(echo $file | cut -d "." -f 2); echo This is a $extension file.; done

While loop

Example: cat food.txt | while read food; do echo My favorite food is $food; done

LOOPS

While loop

Example:

#ID	Input1	Input2
sample1	sample1_a	sample1_b
sample2	sample2_a	sample2_b

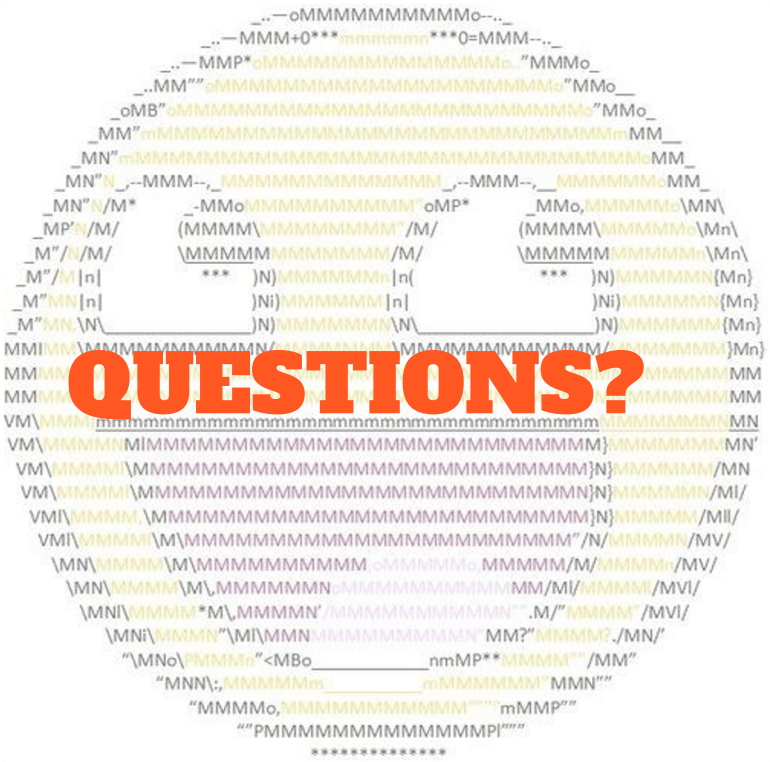
```
tail -n +2 sample.tsv | while read line; do

ID=$(echo “$line” | cut -f 1)
input1=$(echo “$line” | cut -f 2)
input2=$(echo $”line” | cut -f 3)

./my_exec --name $ID --input1 $input1 --input2 $input2 > my_results.txt
```

Exercise 3

You've been tasked with putting together an invitation list for a swanky Hollywood party. But the file you received with the names is all messed up! Using the text editor of your choice, create and execute a script that uses "hollywood.txt" to edit and rearrange the movie star names so that each of them is printed to the screen in the proper order (e.g., "John Doe").

How do we save the output to a new file?

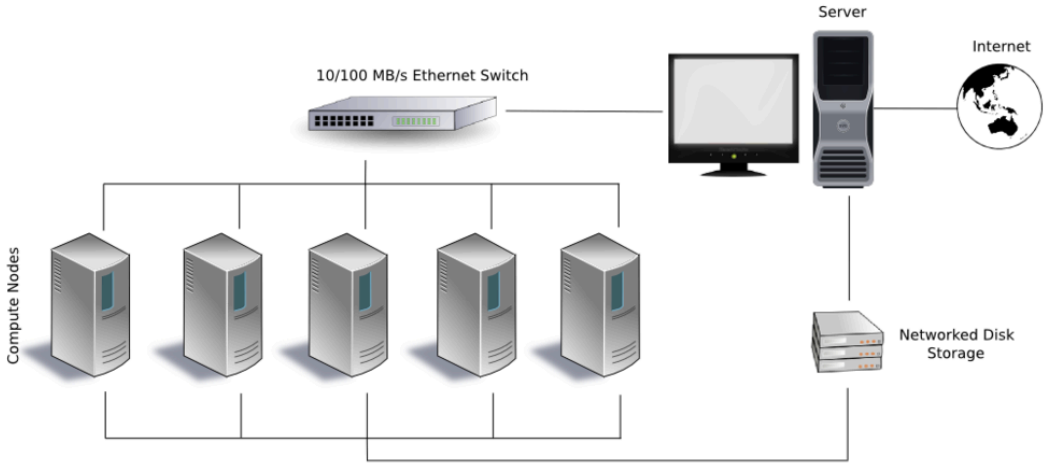
END OF DAY ONE!



What's an HPC cluster anyway?!?

A **High Performance Computing** (HPC) cluster contains multiple physically distinct computers ("nodes") that are connected over a network

A shared, parallel system allows efficient access to the same data across all nodes



How does a cluster work?

- Users do not physically interface with a cluster like they do with their laptops, etc
  - They remotely access the cluster through their own computer
- Many users can be logged in at the same time, doing work
  - The cluster must allocate resources to each user accordingly
- A cluster is composed of many smaller machines
  - Programs running across multiple machines must communicate via a network
- Clusters are always up and running, never shutting down
  - Users can tell the cluster to run a job while they sleep

Introducing SeaWulf!

SeaWulf is...

- An HPC cluster dedicated to research applications for Stony Brook faculty, staff, and students

  a portmanteau of Seawolf and BeoWulf, the name of the first HPC cluster of its kind

- much more powerful than your desktop PC!
  - 320 compute nodes, with 24-40 CPUS and 128 - 192 GB RAM each
  - 8 GPU nodes each with 4 Nvidia Tesla K80 GPUs
  - One large memory node with 3 TB of RAM

How do I get an account on Seawulf?

Two-step process (minimal pain!):

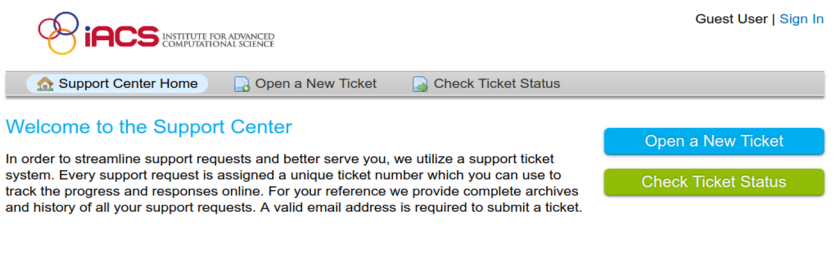
1. Have your PI submit a project request to the [IACS ticketing system]():

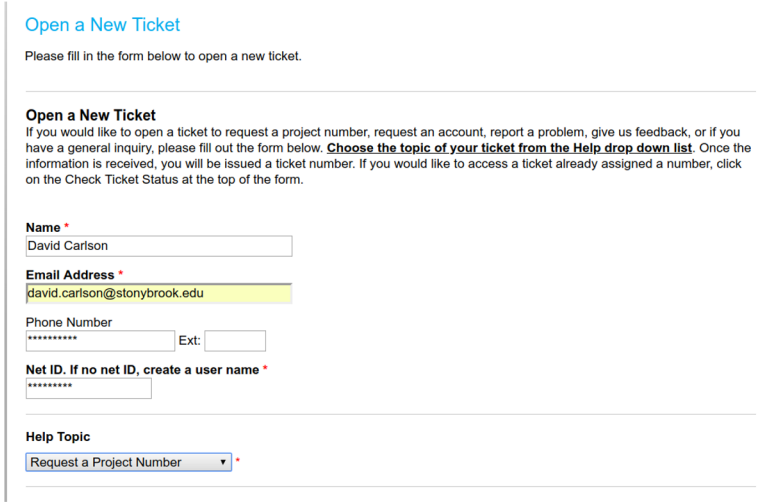
    - Brief description of project

    - Approx. CPU time, storage, & software needed

    - Names of people who will need accounts

    - Do you need a shared project space?





How do I get an account on Seawulf?

Two-step process (minimal pain!):

2. Submit an account request to the IACS ticketing system

    - Indicate name of PI and the project #

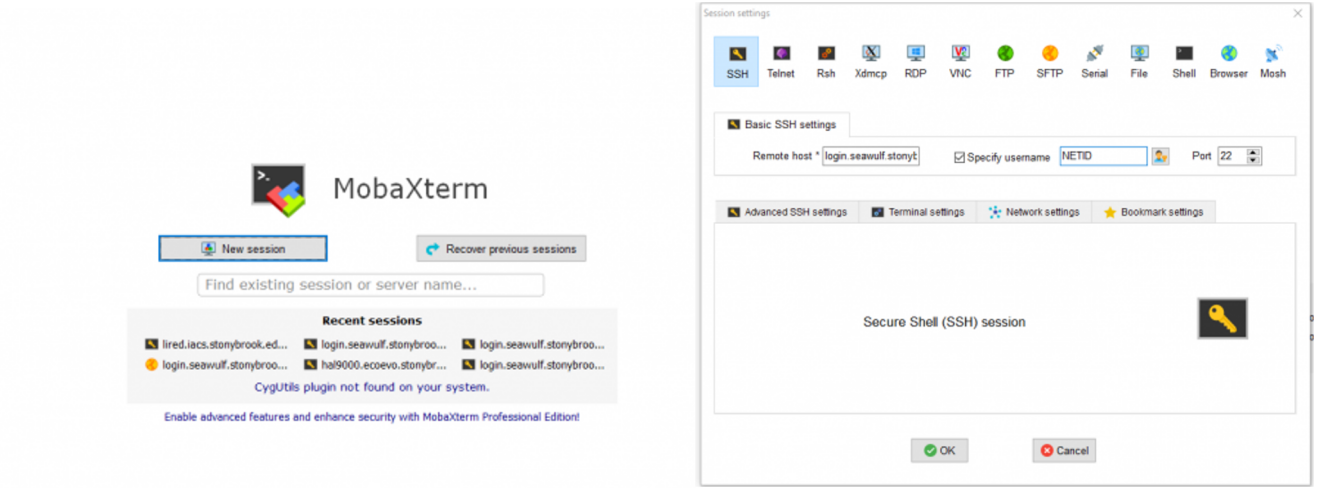
Process usually complete in 1-2 business days

How do I connect to SeaWulf?

Mac & Linux users:

ssh -X netid@login.seawulf.stonybrook.edu

Windows users:



Important paths to remember

/gpfs/home/netid = **your home directory**

/gpfs/scratch/netid = **your scratch directory (for housing temporary and intermediate files)**

/gpfs/software/ = **where most of the research-relevant software is located**

How do I transfer files onto SeaWulf?

Mac & Linux users should use scp (secure copy) to move files to and from SeaWulf

To transfer files from your computer to SeaWulf

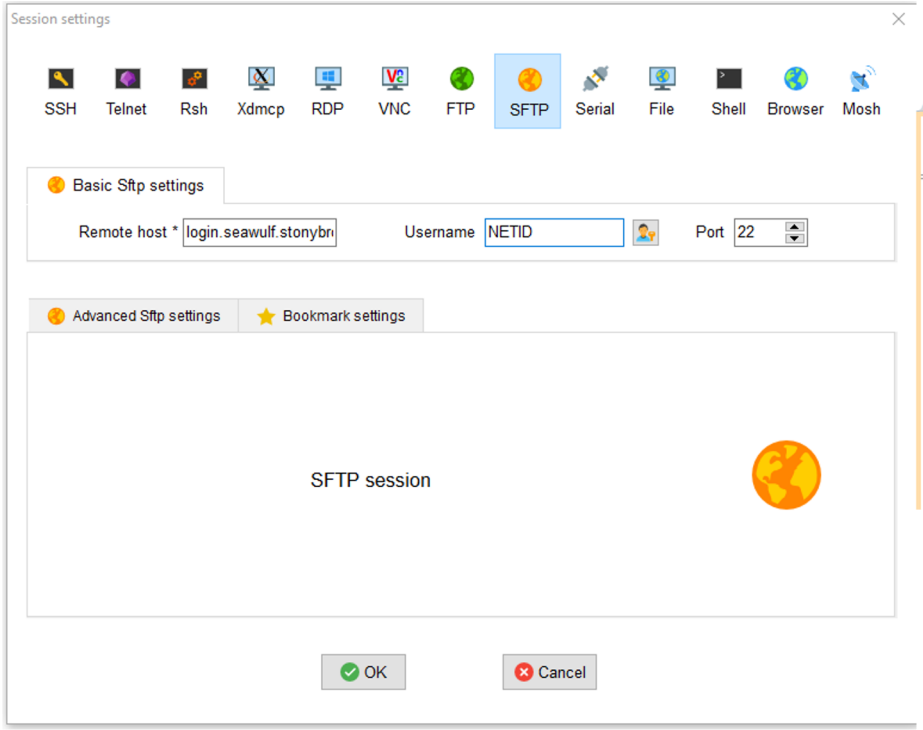
1. Open terminal
2. scp /path/to/my/file netid@login.seawulf.stonybrook.edu:/path/to/destination/

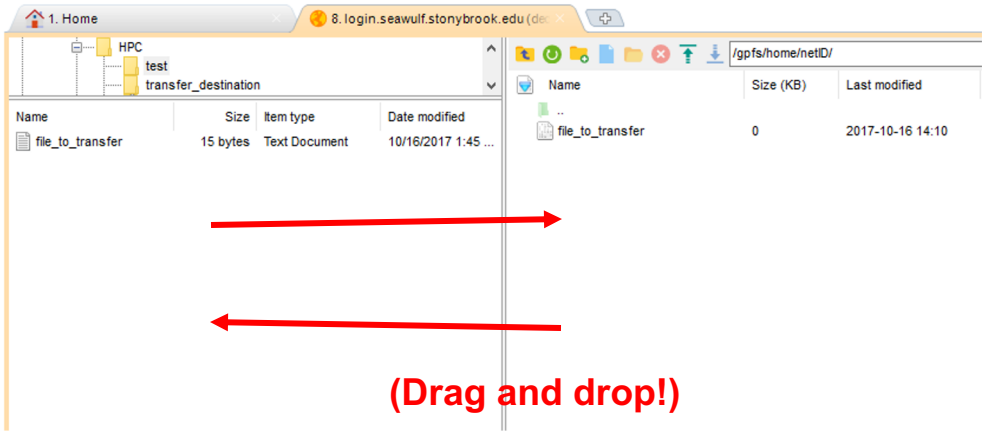
To transfer files from SeaWulf to your computer

1. Open terminal
2. scp netid@login.seawulf.stonybrook.edu:/path/to/my/file /path/to/destination

How do I transfer files onto SeaWulf?

Windows users:





Exercise 4

If you have a SeaWulf account, create a file on your laptop and transfer it to your home directory on SeaWulf.

How do I access software installed on SeaWulf?

Software on the cluster is managed through the module system

- On a multi-user system, there are many software packages, some of which interfere with each other.
- Unix systems (and many others) use environment variables to get around this problem
- Having to tweak the environment variables yourself is tedious
- Modules are designed to get around this problem by setting several environment variables with a single command

How do I access software installed on SeaWulf?

Important module commands:

module avail

module load

module list

module unload

module purge

module initadd

```
[decarlson@login software]$ module avail

------------------------------------------------------------------------------ /cm/local/modulefiles ------------------------------------------------------------------------------
cluster-tools/7.3           dot                         gcc/6.1.0                   module-git                  null                        openmpi/mlnx/gcc/64/1.10.5a1
cmd                         freeipmi/1.5.2              ipmitool/1.8.17             module-info                 openldap                    shared

------------------------------------------------------------------------------ /cm/shared/modulefiles ------------------------------------------------------------------------------
cmgui/7.3                   mvapich2/gcc/64/2.2rc1      petsc/mvapich2/gcc/3.7.5    scalapack/gcc/mvapich2/2.0.2    vesta/3
default-environment         netcdf/gcc/64/4.4.0         petsc/openmpi/gcc/3.7.2     scalapack/openmpi/gcc/64/2.0.2  visit/2.13.0
maui/3.3.1                  netperf/2.7.0               protobuf/2.6.1rc1           scotch/scotch-6.0.3             xcrysden/1.5.60
mpich/ge/gcc/64/3.2rc2      openblas/dynamic/0.2.18     qt/qt-4.8.6                 siesta/3.2-pl-5
mpiexec/0.84_432            openmpi/gcc/64/1.10.1       revbayes/1.0.6              slurm/16.05.2
MUMPS/MUMPS-5.1.1           ovito/2.8.2                 scalapack/dynamic/scalapack-2.0.2 torque/6.0.2

------------------------------------------------------------------------------ /gpfs/shared/modulefiles ------------------------------------------------------------------------------
acml/gcc/64/5.3.1                   darshan/3.1.4                       intel/mkl/64/2017/0.098             NeuroElf/1.1
acml/gcc/fma4/5.3.1                 doxygen/1.8.15                      intel/mkl/64/2018/18.0.0            nibabel/1.2.0
acml/gcc/mp/64/5.3.1                Elmer/Elmer                         intel/mkl/64/2018/18.0.1            numactl/2.0.11
acml/gcc/mp/fma4/5.3.1              envi/5.4.1                          intel/mkl/64/2018/18.0.2            nwchem/6.8
acml/gcc-int64/64/5.3.1             expat/2.2.5                         intel/mkl/64/2018/18.0.3            octopus/intel/8.2
acml/gcc-int64/fma4/5.3.1           FastTreeMP/2.1.10                   intel/mkl/64/2019/19.0.0            octopus/intel/8.2-serial
acml/gcc-int64/mp/64/5.3.1          fftw2/openmpi/gcc/64/double/2.1.5   intel/mpi/32/2017/0.098             openblas/0.2.20_serial
acml/gcc-int64/mp/fma4/5.3.1        fftw2/openmpi/gcc/64/float/2.1.5    intel/mpi/64/2017/0.098             opencv/2.4.9
afni/17.2.05                        fftw3/openmpi/gcc/64/3.3.4          intel/mpi/64/2018/18.0.0            openjpeg/1.5.0
amber/16                            fftw3/openmpi/gcc/64/3.3.7          intel/mpi/64/2018/18.0.1            openjpeg/2.3
anaconda/2                          freesurfer/5.3.0-HCP                intel/mpi/64/2018/18.0.2            openmpi/1.10.1
anaconda/3                          freesurfer/6.0.0                    intel/mpi/64/2018/18.0.3            openmpi/3.0.0
apr/1.6.3                           freetype/2.9                        intel/mpi/64/2019/19.0.0            openmpi/openmpi-opal
apr-util/1.6.1                      fsl/5.0.10                          intel/tbb/64/2018/18.0.2            openslide/3.4.0
ase/3.13.0                          fsl/5.0.6                           intel/tbb/64/2018/18.0.3            oprofile/1.3.0
atlas/3.10.3                        GATK/4.0.0                          intel/tbb/64/2019/19.0.0            ovito/2.8.2
binutils/2.27                       GATK/4.0.2                          intel-stack                         ovito/2.9.0
binutils/2.30                       gc/7.2                              intel-tbb-oss/ia32/2017_20160722oss parmetis/parmetis-4.0.3
binutils/devel/2.27-27              gcc/7.1.0                           intel-tbb-oss/intel64/2017_20160722oss pdtoolkit/3.25
blacs/openmpi/gcc/64/1.1patch03     gcc/7.2.0                           iozone/3_434                        perf/3.10.0
blas/gcc/64/3.6.0                   gcc/8.1.0                           IQ-TREE/1.6.5                       pgi/17.3
blas/gcc/64/3.7.0                   gcc-stack                           ITK/4.9.1                           pixman/0.29.3
blast+/2.7.1                        gdal/2.2.3                          JAGS/4.2.0                          proj/4.9.3
```

How do I access software installed on SeaWulf?

An example: accessing Python

```
[decarlson@login1 ~]$ module load anaconda/3
[decarlson@login1 ~]$ python
Python 3.7.3 | packaged by conda-forge | (default, Mar 27 2019, 23:01:00)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>>
```

The Conda package manager: making software installation (almost!) painless

Manages installation, deletion, and dependency conflicts

Two modules: **anaconda/2 & anconda/3**

Houses much of the software on SeaWulf

Can create separate environments

The Conda package manager

Useful commands

conda list - l

conda info --

source activ

Source deac

****Bioinformatics bonus!****

Most bioinformatics software installed in anaconda/2!

Look in both the default environment and custom environments!

conda create --prefix /path/to/my/env - creates new environment at the specified path

Exercise 5

Load the anaconda/2 module, and check which programs are installed in the “RNA-seq” environment.

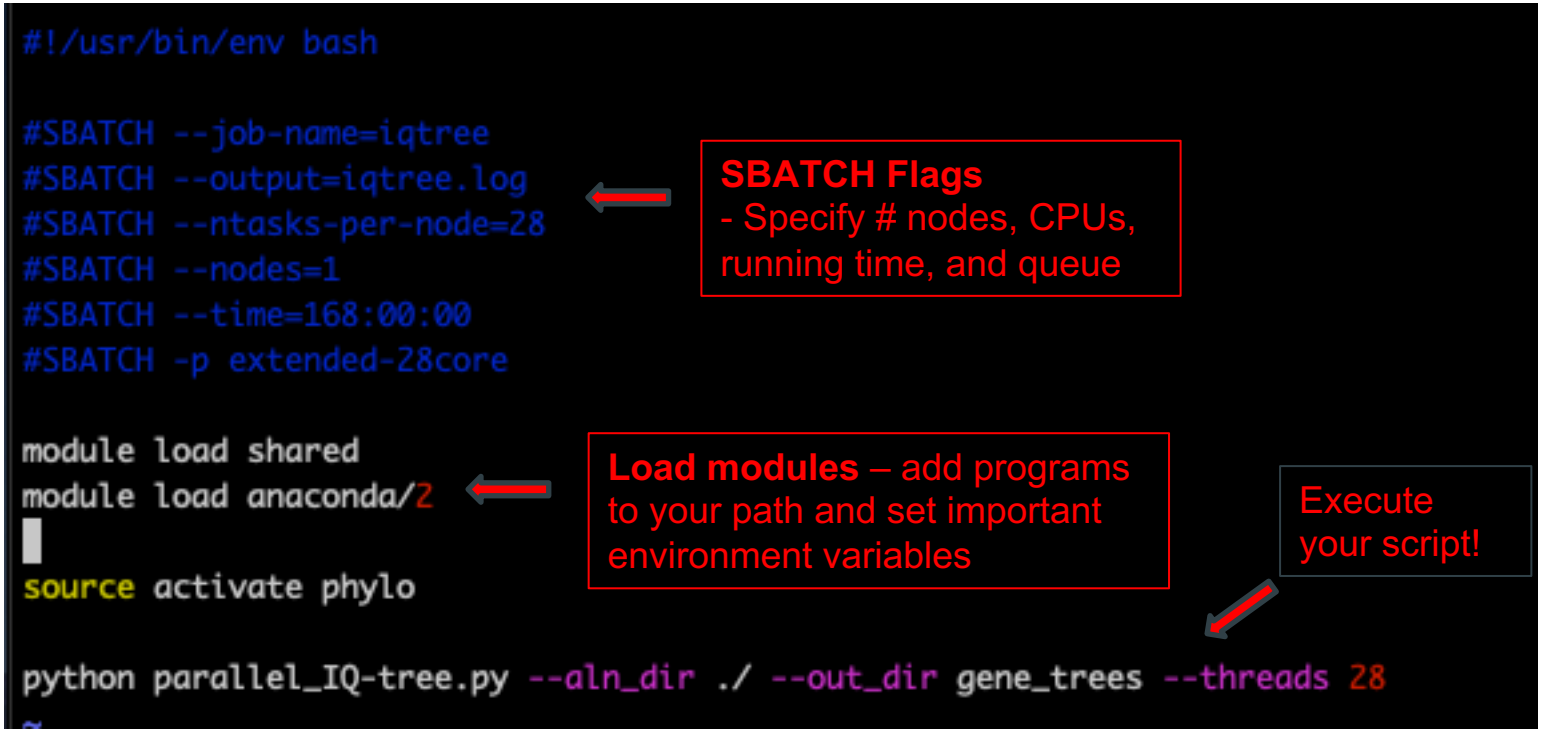
How do I do work on SeaWulf?

Computationally intensive jobs should *not* be done on the login node!

- To run a job, you must submit a batch script to the Slurm Workload Manager
- A batch script is a collection of bash commands issued to the scheduler, which which distributes your job across one or more compute nodes
- The user requests specific resources (nodes, cpus, job time, etc.), while the scheduler places the job into the queue until the resources are available

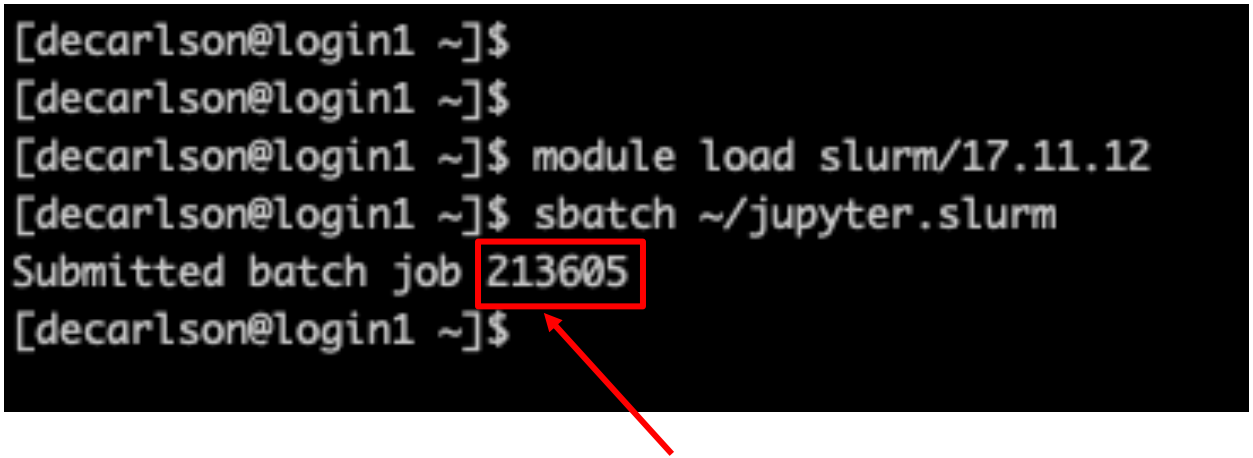
Example Slurm Job Script

All jobs submitted through a job scheduling system using scripts



How do I execute my Slurm script?

Jobs are submitted via the Slurm Workload Manager using the “sbatch” command



Useful Slurm commands

sbatch <script> = submit a job

scancel <job id> = cancel a job

squeue = get job status

scontrol show job <job id> = get detailed job info

sinfo = get info on node/queue status and utilization

What queue should I submit to?

How many nodes do you need?

How many cores per node?

How much time do you need?

There is often a tradeoff between resource usage and wait time!!

Queue	Default run time	Max run time	Max # of nodes
debug-28core	1 hour	1 hour	8
extended-24core	8 hours	7 days	2
extended-28core	8 hours	7 days	2
extended-40core	8 hours	7 days	2
gpu	1 hour	8 hours	2
gpu-long	8 hours	48 hours	1
gpu-large	1 hour	8 hours	4
large-24core	4 hours	8 hours	60
large-28core	4 hours	8 hours	80
large-40core	4 hours	8 hours	50
long-24core	8 hours	48 hours	8
long-28core	8 hours	48 hours	8
long-40core	8 hours	48 hours	6
medium-24core	4 hours	12 hours	24
medium-28core	4 hours	12 hours	24
medium-40core	4 hours	12 hours	16
p100	1 hour	24 hours	1
short-24core	1 hour	4 hours	12
short-28core	1 hour	4 hours	12
short-40core	1 hour	4 hours	8

Need to troubleshoot? Use SeaWulf interactively!

Example:

```
[decarlson@cn-mem ~]$ srun -J test_interactive -N 1 -p short-24core --ntasks-per-node=24 --time=05:00 --pty bash
srun: job 213628 queued and waiting for resources
srun: job 213628 has been allocated resources
[decarlson@cn017 ~]$
```

”srun” requests a compute node for interactive use

“--pty bash” configures the terminal on the compute node

Once a node is available, you can issue commands on the command line

Good for troubleshooting,

Inefficient once your code is working

Exercise 6

Open “test.slurm” and determine what the walltime, job name, and queue are. Modify the “cd” statement to point to your home directory. Submit the job. When it finishes, check the output file.

How can I compile code on SeaWulf?

Multiple compilers available

- **GNU compiler collection (GCC)**
  - 4.8.5 (system default)
  - 6.x, 7.x , 8.x, and 9.x (modules)
- **Intel**
  - Intel Parallel Studio 2017, 2018, 2019

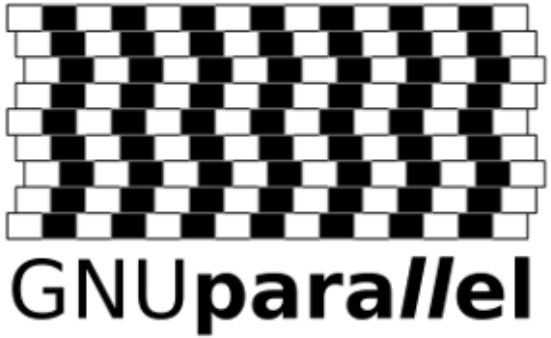
Example – compile a simple “hello world” C program

```
cd /gpfs/projects/samples/hpc_workshop_021419
module load gcc-stack
gcc hello.c -o hello
```

Parallel processing on the cluster

- **Parallelization *within a single compute node***
  - Lots of ways of doing this
  - Some software innately able to run on multiple cores
  - Some tasks easily parallelized with scripting (e.g, “Embarrassingly Parallel” tasks)
- **Parallelization across *multiple nodes***
  - Requires the use of MPI
- **Parallelization with GPUs** - only available on specific GPU nodes

Parallel processing on a single node with GNU Parallel



- Perfect for “embarrassingly parallel” situations
- Available as a module: gnu-parallel/6.0
- Can easily take in a series of inputs (e.g., files) and run a command on each input simultaneously
- Lots of tutorials and resources available on the web!

Exercise 7

Take a look at the “gnu_parallel_example” directory. There are 10 files in this directory that each contain an objectively false and scurrilous statement! We will use the “parallel_example.sh” script to correct these terrible lies. But first, let’s examine the script to get a feel for how gnu parallel works.

Helpful GNU Parallel resources:
https://www.gnu.org/software/parallel/parallel_tutorial.html (thorough!!)
https://www.biostars.org/p/63816/ (bioinformatics specific)
https://www.msi.umn.edu/support/faq/how-can-i-use-gnu-parallel-run-lot-commands-parallel (many practical examples)

Can my job use multiple nodes?

MVAPICH

Yes!

(...well...maybe)

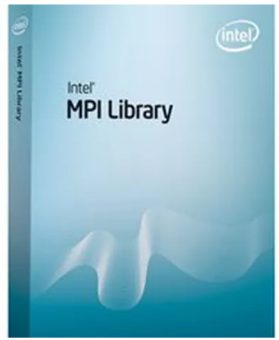
Message Passing Interface (MPI) allows communication across nodes

Not all software is compatible with MPI

Multiple “flavors” of MPI are available on SeaWulf

- E.g., **mvapich2**, **Intel MPI**, **OpenMPI, mpich2**

Open MPI



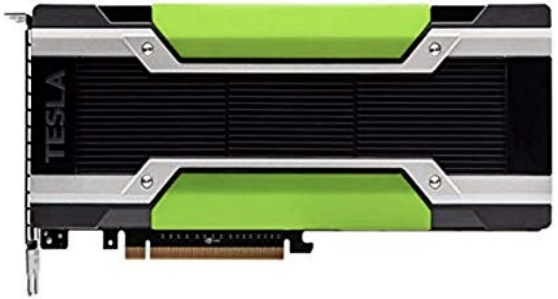
Example MPI Job Submission Script

```
#!/usr/bin/bin/env bash

#SBATCH --job-name=test
#SBATCH --output=test.txt
#SBATCH --ntasks-per-node=40
#SBATCH --nodes=2
#SBATCH --time=05:00
#SBATCH -p short-40core

module load gcc-stack

mpirun -np 56 ./gcc_mpi_hello
```

This will load both gcc and mvapich2, an implementation of MPI that is compatible with gcc

The “-np” flag specifies how many processors *across all nodes* will be used.

What about GPUs?



8 nodes containing 4 Tesla K80 GPUs each

1 node containing Tesla P100

A variety of GPU-related software available as modules

- Multiple versions of CUDA
- GPU-enabled Machine learning (Tensorflow, Pytorch, etc.)

Exercise 8

Take a look at “hello_world.py” and “hello_world_py.slurm”. How many nodes will this job use? How many total processors? Change the “cd” statement to point to your home directory. Run the job. If your job finishes, interpret what the output means.

Troubleshooting common issues

Problem:

“`ssh_exchange_identification: read: Connection reset by peer`”

Translation: Your IP address has been blocked due to too many unsuccessful login attempts.

Resolution: submit a ticket, and we’ll unblock your IP address.

Troubleshooting common issues

Problem: You want to know what software is currently available

Resolution:

Check the modules (“module avail”)

Look in the anaconda modules (“conda list”, “conda info --envs”)

Check our FAQ page: https://it.stonybrook.edu/help/kb/installed-software-on-seawulf

Troubleshooting common issues

Problem: SeaWulf is missing software that you need

Resolution:

Compile it yourself or use a package manager to install locally:

E.g., conda create --prefix /gpfs/home/decarlson/my_env

Submit a ticket, and we'll install it globally

Troubleshooting common issues

Problem: I need to use a GUI-based program!

Resolution: Use X11 forwarding to display the GUI on your local machine

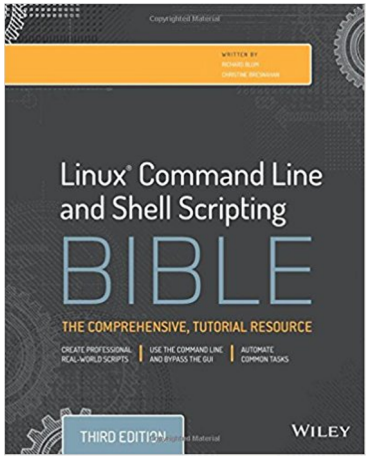
1. Make sure you have an X server installed on your computer
    a. Linux: installed automatically
    b. MAC: install XQuartz
    c. Windows: install MobXterm (recommended!) or Xming

1. ssh **-X** netid@login.seawulf.stonybrook.edu

Need more help or information?

Check out our FAQ: https://it.stonybrook.edu/services/high-performance-computing

Useful book!



Submit a ticket: https://iacs.supportsystem.com


QUESTIONS?
CRAY
CS400