Orientation
Joint High Performance Computing Exchange (JHPCE)
CMS Subcluster (C-SUB)

JOHNS HOPKINS
BLOOMBERG SCHOOL
of PUBLIC HEALTH

http://www.jhpce.jhu.edu/

Version: 20240506

Schedule

- **Introductions – who are we, who are you?**
- Terminology
- Logging in and account setup
- Basics of running programs on the cluster
- Details – limits and resources
- Examples
- Moving data into & out of C-SUB
- Addendum: Additional UNIX topics

Why The C-SUB?

The Centers for Medicare and Medicaid Services (CMS) is part of the U.S. Department of Health and Human Services. It provides important data for researchers of patients, their conditions, and the American health care system.

Acquiring and managing sensitive information from the Federal government is time-consuming, and requires on-going administrative and information technology support by groups with specific expertise.

To facilitate research, an effort has been made to create an infrastructure that can provide those resources, as well as a computational facility to store and analyze the data.

Researchers can leverage this existing infrastructure to more quickly and efficiently start and conduct their work.

The Health Analytics Research Platform (HARP) was created to implement this vision. It is a collaboration of existing personnel across multiple organizations. Its leaders provide funding and guidance to an IT group which created a computing facility named the C-SUB.

The CMS subcluster (C-SUB) makes use of some of the resources of an existing High Performance Computing cluster called JHPCE.

Health Policy & Management (HPM) Component of C-SUB

The Health Analytics Research Platform (HARP) provides data services to HBHI-affiliated and HEADS Center-affiliated investigators to facilitate research collaborations advancing HBHI's strategic pillars.

- HPM
  - HEADS/HARP Director: Dan Polsky
  - HEADS/HARP Deputy Director: Matt Eisenberg
  - CMS Data Expert: Frank Xu

Please send C-SUB data-specific requests to
support@harp-csub.freshdesk.com

- exporting files out of the C-SUB
- data inventory – current and desired additions or updates

Health Analytics Research Platform (HARP)
https://hbhi.jhu.edu/affiliate-resource/health-analytics-research-platform-harp

Hopkins Business of Health Initiative (HBHI)
https://hbhi.jhu.edu/about-us

Hopkins Economics of Alzheimer Disease & Services (HEADS)
https://publichealth.jhu.edu/hopkins-economics-of-alzheimers-disease-and-services-center

JHPCE C-SUB – Joint High Performance Computing Exchange

- Co-Directors: Brian Caffo, Mark Miller
- Systems Engineers: Jiong Yang, Jeffrey Tunison
- Application Developer: Adi Gherman

Beyond this class, when you have questions:

- http://jhpce.jhu.edu
  - lots of good info – use the search field!!!!
  - these slides
- bitsupport@lists.johnshopkins.edu
  - System issues (accessing software, general questions)
  - Monitored by the 5 people above
- bithelp@lists.johnshopkins.edu
  - Complex application issues (R/SAS/python...)
  - Monitored by volunteer power users
- support@harp-csub.freshdesk.com
  - Data layout questions, export & update requests
- https://jhpce-app02.jhsph.edu (only visible from Hopkins networks)
  - password resets, one-time password tokens
- Others in your lab
- Web Search – Google your error message

CMS on JHPCE Is A Specialized Thing

The CMS subcluster (C-SUB) makes use of some of the resources of the original JHPCE cluster. For example, the scientific software such as STATA and SAS.

However, in many other ways it operates differently than the rest of the cluster.

Please keep that in mind when reading JHPCE documentation or asking for help via the bitsupport & bithelp mailing lists. Mention that you are a C-SUB user.

Tell us about yourself

- Name
- Department
- How do you plan on using the cluster? What data or applications will you be using?
- Will you be accessing the cluster from a Mac or a Windows system?
- What is your experience with Unix?
- Any experience using other clusters?

Schedule

- Introductions – who are we, who are you?
- **Terminology**
- Logging in and account setup
- Basics of running programs on the cluster
- Details – limits and resources
- Examples
- Moving data into & out of C-SUB
- Addendum: Additional UNIX topics

Clusters – what and why?

What is a cluster?

- A collection of many computers (**nodes**) that can be shared by multiple users.
- To access the cluster, you log into a **“head”** or **“login”** node.
- The head node is also used to transfer data in or out.

Why would you use a cluster?

- Need resources not available on your local laptop
- Need to run a program (**job**) that will run for a long time
- Need to run a job that can make use of multiple computers simultaneously (**parallel computing**)
- Want to queue multiple jobs so they run ASAP without needing your attention to launch them (using a **job scheduler**)
- Meet data security requirements (e.g. physical isolation)

Node (Computer) Components

- Each computer is called a "**node**"
- Each node, like a desktop/laptop, has:
  - CPUs (central processing units)
  - RAM (dynamic memory)
  - Disk space (permanent storage)
- Unlike desktop/laptop systems, nodes do not make use of a connected display/keyboard/mouse – they are used over a network, often from a **command line interface** (CLI) known as a "**shell**".
- **Graphical user interface** (GUI) programs can be run, displaying on your desktop/laptop.

JHPCE CMS Subcluster (C-SUB)

- Joint High Performance Computing Exchange (JHPCE)
- Located in a locked cabinet at Bayview Colocation Facility

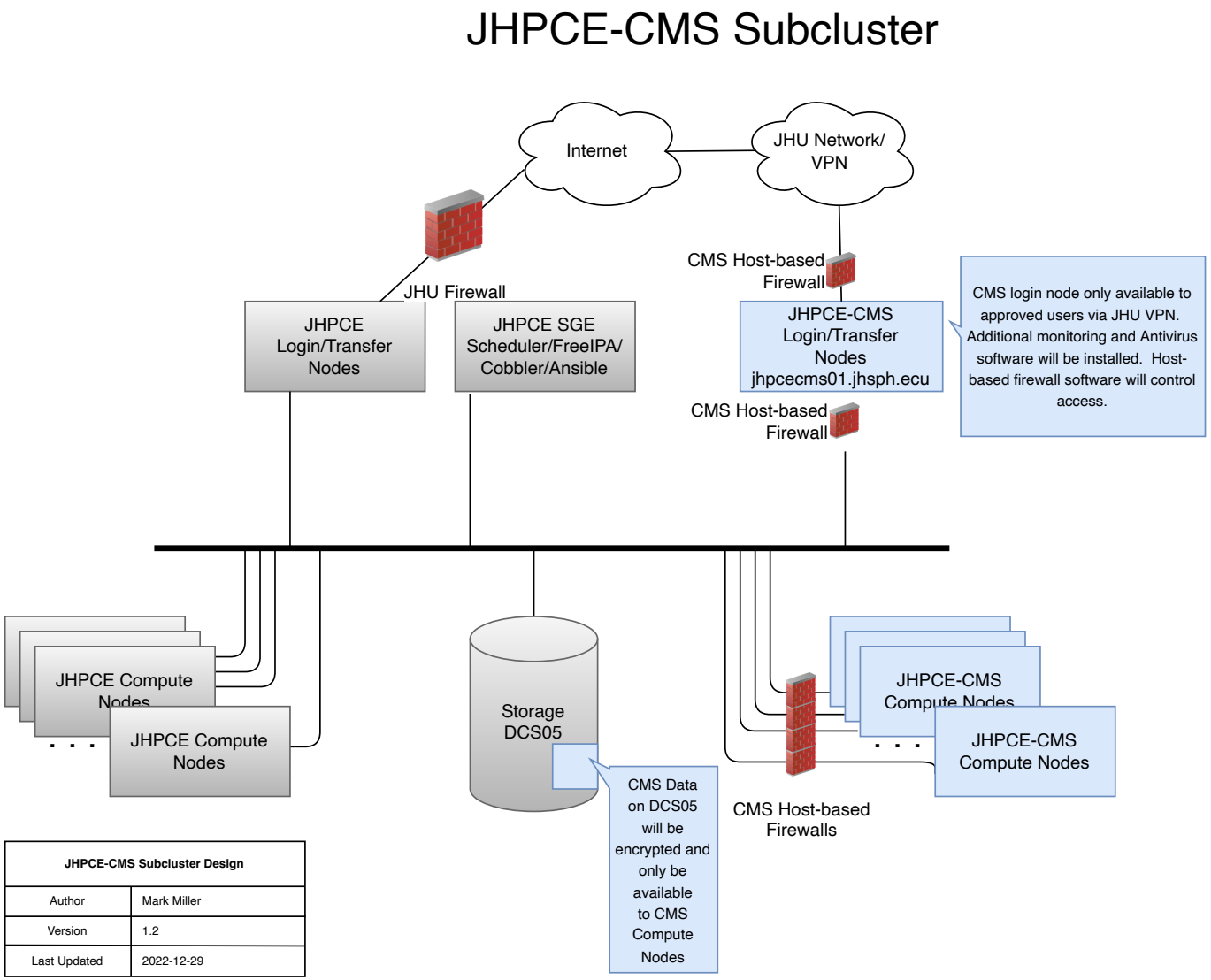
Hardware:

- 3 Nodes – 1 login/transfer node, 2 compute
  - Login node: 2x16-core CPUs, 8GB RAM
  - Each compute node has 2x64-core CPUs
    (128 physical cores; hyperthreading creates 256 virtual cores)
  - Each node has 1024GB or 1TB of RAM
- 100 TB Disk Allocation dedicated to all CMS work.
  - Storage is network-attached and available only to the CMS nodes.

Software:

- Based on Rocky 9 Linux (Rocky is a Red Hat Enterprise Linux clone)
- Jobs are managed by SLURM (Simple Linux Utility for Resource Management)
- Main JHPCE cluster used for a wide range of Biostatistics – gene sequence analysis, population simulations, medical treatment.
- Common applications: R, SAS, Stata, python, Jupyter ...

JHPCE CMS Subcluster Schematic



Schedule

- Introductions – who are we, who are you?
- Terminology
- **Logging in and account setup**
- Basics of running programs on the cluster
- Details – limits and resources
- Examples
- Moving data into & out of C-SUB
- Addendum: Additional UNIX topics

How do you use the cluster?

- The JHPCE CMS cluster is accessed using SSH (Secure SHell), so you will need an ssh client.
- Use **`ssh`** to login to "**`jhpcecms01.jhsph.edu`**"
- The login node is only available from with the JHU network

- For Mac and Linux users, you can use **`ssh`** from a Terminal application window.

- For MS Windows users, you need to install an ssh client – such as MobaXterm (strongly recommended) or Cygwin, Putty, Winscp, or WSL2 :

  http://mobaxterm.mobatek.net/

  http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

  http://www.cygwin.com

  http://winscp.net

Quick note about graphical programs

To run graphical (GUI) programs on the JHPCE cluster, you will need to have an **X11 server** running on your laptop.

- For Microsoft Windows, MobaXterm has an X server built into it.
- For Windows, if you are using Putty, you will need to install an X server such as Cygwin.

- For Macs:

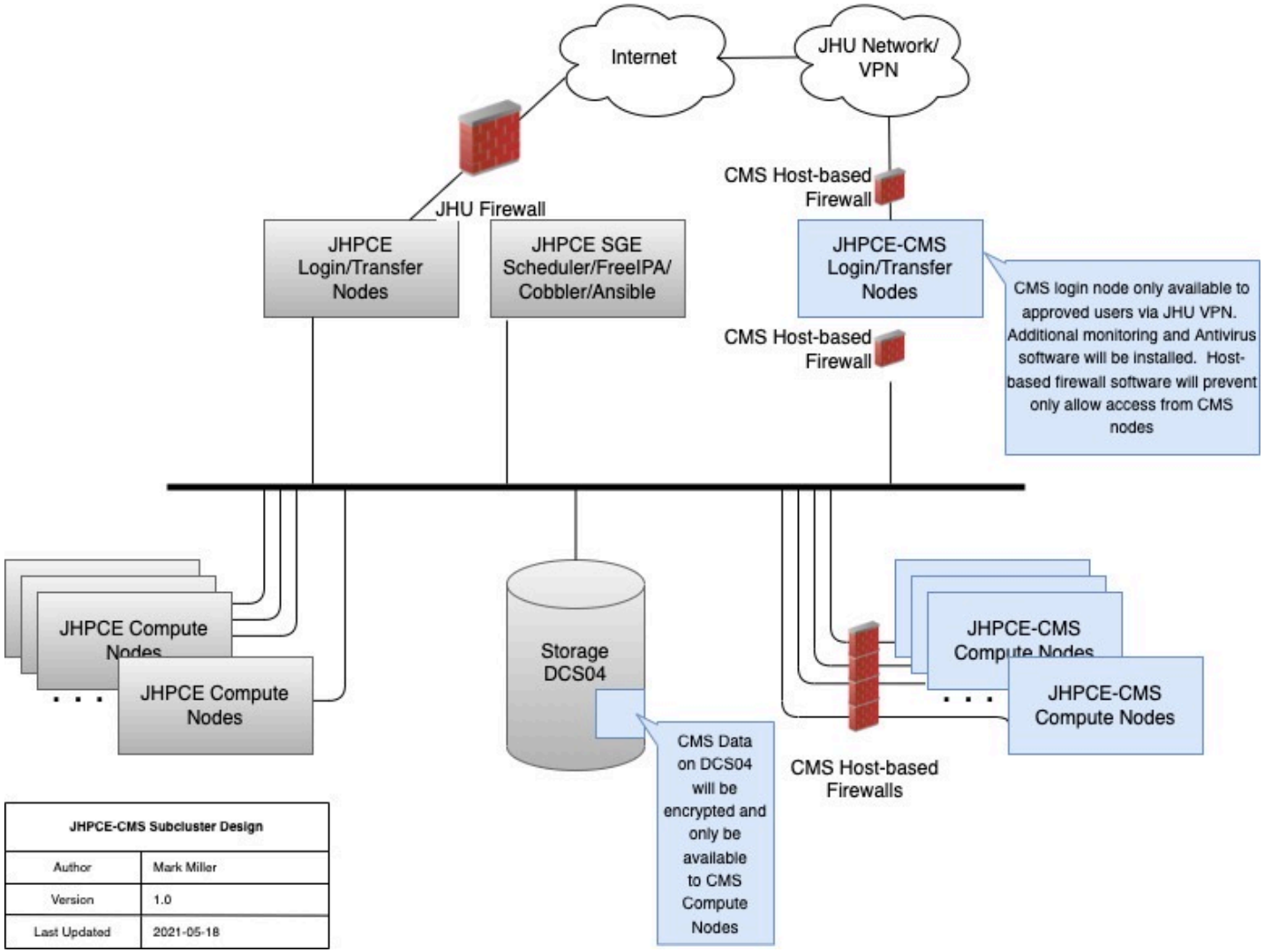
1) You need to have the Xquartz program installed on your laptop. This software is a free download from Apple, and does require you to reboot your laptop http://xquartz.macosforge.org/landing/
2) You need to add the "-X" option to your ssh command:

```
$ ssh -X my-username@jhpcecms01.jhsph.edu
```

- For Linux laptops, you should already have an X11 server installed. You will though need to add the –X option to ssh:

```
$ ssh -X my-username@jhpcecms01.jhsph.edu
```

JHPCE CMS Subcluster Schematic



Example 1 – Logging in

- Bring up Terminal
- Run: **`ssh -X my-username@jhpcecms01.jhsph.edu`**
- Password & 2 Factor authentication
  - When you type your password, the cursor will not move. This is a security mechanism so that someone looking over your shoulder won't be able to see your password.
  - The first time you login, you will use the initial Verification Code and initial Password sent to you.
  - After you login the first time, you will configure the Google Authenticator app to provide more verification codes
  - Verification codes are always required to login, and can only be used once!!!
- Shell prompt

Lab 1 - Logging In

- For Mac/Linux laptop Users:

  - Bring up a Terminal
  - Run: **`ssh –X USERID@jhpcecms01.jhsph.edu`**

  - Login with the initial Verification Code and Password that were sent to you

- For PC Users:

  - Launch MobaXterm
  - Do NOT try to run ssh from inside one of MobaXterm's built-in terminals.
  - Click on the "Sessions" icon in the upper left corner 
  - On the "Session settings" screen, click on "SSH" 

  - Enter "jhpcecms01.jhsph.edu" as the "Remote host". Click on the "Specify username" checkbox, and enter your C-SUB username in the next field. Then click the "OK" button.
  - Login with the initial Verification Code and Password that were sent to you.
  - If dialog windows pop up, click "No" or "Do not ask this again" when prompted to save your password (you are about to change it to something else).

Lab 1 - Logging In - continued

- You have a limited amount of time to enter both your password and the verification code.

- You will then be asked to change your password.
  - FIRST, enter your current password. The temporary one we gave you
  - SECOND, enter your new password. It needs to have 3 of these 4 types of characters: lowercase, uppercase, numerical digit, special (e.g. & + >).
  - THIRD, re-enter your new password

- In the future you can change your password with the "**`kpasswd`**" command. (Again, you will be prompted for your current password, and then twice prompted for a new password.)

Lab 1 - Logging In - cont

- You need to enter a “verification code” every time you log into the C-SUB. Therefore we will now configure your smartphone to be able to generate them, and test the result.

- Configure 2 factor authentication
  - http://jhpce.jhu.edu/knowledge-base/how-to/2-factor-authentication/
  - 1) On your smartphone, bring up the "Google Authenticator" app
  - 2) On the JHPCE cluster, run "auth_util"
  - 3) In "auth_util", use option "5" to display the QR code (you may need to resize your ssh window (in MobaXterm: from the “view” menu select “terminal unzoom”)
  - 4) Scan the QR code with the Google Authenticator app
  - 5) Next, in “auth_util” use option 2 to display your scratch codes – record these
  - 6) In " auth_util", use option "6" to exit from "auth_util”

- In a new terminal window or MobaXterm session, test that you know your new password and have valid verification codes by logging in a second time
- If successful, log out of the cluster in your first terminal by typing "exit".

Lab 1 - Logging In - cont

- Keep some "Emergency Scratch Codes" on hand
- 500 GB limit on storage, (including home directory, SFTP incoming and SFTP outgoing data directories).
- Home directories are backed up, but other storage areas are probably not.
- If you fail to log in five times within a few minutes, your IP address will be blocked for 15 minutes. If you get blocked, take note of the time and wait the full 15 minutes.
- Your login will fail if you enter a Google Authenticator verification code TOO CLOSE to the end of its one-minute validity period. If you see on your phone that you are approaching the last few seconds, just wait until a new code is generated.

Schedule

- Introductions – who are we, who are you?
- Terminology
- Logging in and account setup
- **Basics of running programs on the cluster**
- Details – limits and resources
- Examples
- Moving data into & out of C-SUB

General Linux/Unix Commands 

Navigating Unix:

- ls
- ls -l
- ls -al
- pwd
- cd
- . and ..

Commands in example script:

- date
- echo
- hostname
- sleep
- control-C

Looking at files:

- cat/less

Changing files with editors:

- nano, geany
- vim/emacs

Spelling and spaces are important!!!

Good resources for learning Linux (and at end of presentation):

http://korflab.ucdavis.edu/Unix_and_Perl/unix_and_perl_v3.1.1.html

https://www.digitalocean.com/community/tutorials/a-linux-command-line-primer

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.

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

How do programs get run on the CMS compute nodes?

- For the CMS nodes, we use a **job scheduler** called “SLURM” that schedules programs (**jobs**).
- Jobs specify resources that they will need, such as the number of CPU **cores** & amount of **RAM** (in megabytes)
- Jobs are submitted to **queues** or **partitions** (collections of resources governed by specific rules (such as priorities))
- Jobs are assigned to compute nodes as the CPU and RAM resources required become available in sufficient quantities
- Jobs can be **interactive**, (run in real time), or **batch**, which are scheduled for future unattended execution

Primary Commands – SLURM (Simple Linux Utility for Resource Management)

- **`sbatch`** – submit a batch job to the cluster
- **`srun --pty --x11 bash`** – establish an interactive session*

- **`scancel`** – cancel or pause a job
- **`squeue`** – see the status of running & pending jobs
- **`sacct`** – see the status of past (& running) jobs**
- **`sinfo`** – see status of the compute nodes
- **`sstat`** – see statistics from running jobs

* You can omit the "--x11" if you are not going to run X11 programs; bash must always come last
** sacct only works on jhpcecms01

(These commands are documented in manual pages on our computers. Also you can find them online (in a more readable format) at https://slurm.schedmd.com/archive/slurm-22.05.9 (Our SLURM is version 22.05.09, so you should not read the latest docs.)

slurm®
workload manager

Job Submission

salloc - Obtain a job allocation.
sbatch - Submit a batch script for later execution.
srun - Obtain a job allocation (as needed) and execute an application.

–array=<indexes> (e.g. "--array=1-10")	Job array specification. (sbatch command only)
–account=<name>	Account to be charged for resources used.
–begin=<time> (e.g. "--begin=18:00:00")	Initiate job after specified time.
–clusters=<name>	Cluster(s) to run the job. (sbatch command only)
–constraint=<features>	Required node features.
--cpu-per-task=<count>	Number of CPUs required per task.
–dependency=<state:jobid>	Defer job until specified jobs reach specified state.
–error=<filename>	File in which to store job error messages.
–exclude=<names>	Specific host names to exclude from job allocation.
–exclusive[=user]	Allocated nodes can not be shared with other jobs/users.
–export=<name[=value]>	Export identified environment variables.
–gres=<name[:count]>	Generic resources required per node.
–input=<name>	File from which to read job input data.
–job-name=<name>	Job name.
–label	Prepend task ID to output. (srun command only)
–licenses=<name[:count]>	License resources required for entire job.
--mem=<MB>	Memory required per node.
--mem-per-cpu=<MB>	Memory required per allocated CPU.
-N<minnodes[-maxnodes]>	Node count required for the job.
-n<count>	Number of tasks to be launched.
--nodelist=<names>	Specific host names to include in job allocation.
--output=<name>	File in which to store job output.
--partition=<names>	Partition/queue in which to run the job.
--qos=<name>	Quality Of Service.
--signal=[B:]<num>[@time]	Signal job when approaching time limit.
--time=<time>	Wall clock time limit.
--wrap=<command_string>	Wrap specified command in a simple "sh" shell. (sbatch command only)

Accounting

sacct - Display accounting data.

--allusers	Displays all users jobs.
--accounts=<name>	Displays jobs with specified accounts.
--endtime=<time>	End of reporting period.
--format=<spec>	Format output.
--name=<jobname>	Display jobs that have any of these name(s).
--partition=<names>	Comma separated list of partitions to select jobs and job steps from.
--state=<state_list>	Display jobs with specified states.
--starttime=<time>	Start of reporting period.

SchedMD
Slurm Support and Development

sacctmgr - View and modify account information.
Options:

--immediate	Commit changes immediately.
--parseable	Output delimited by '|'

Commands:

add <ENTITY> <SPECS> create <ENTITY> <SPECS>	Add an entity. Identical to the **create** command.
delete <*ENTITY*> where <*SPECS*>	Delete the specified entities.
list <ENTITY> [<SPECS>]	Display information about the specific entity.
modify <*ENTITY*> where <*SPECS*> set <*SPECS*>	Modify an entity.

Entities:

account	Account associated with job.
cluster	*ClusterName* parameter in the *slurm.conf*.
qos	Quality of Service.
user	User name in system.

Job Management

sbcast - Transfer file to a job's compute nodes.

sbcast [options] SOURCE DESTINATION

--force	Replace previously existing file.
--preserve	Preserve modification times, access times, and access permissions.

scancel - Signal jobs, job arrays, and/or job steps.

--account=<name>	Operate only on jobs charging the specified account.
--name=<name>	Operate only on jobs with specified name.
--partition=<names>	Operate only on jobs in the specified partition/queue.
--qos=<name>	Operate only on jobs using the specified quality of service.

--reservation=<name>	Operate only on jobs using the specified reservation.
--state=<names>	Operate only on jobs in the specified state.
--user=<name>	Operate only on jobs from the specified user.
--nodelist=<names>	Operate only on jobs using the specified compute nodes.

squeue - View information about jobs.

--account=<name>	View only jobs with specified accounts.
--clusters=<name>	View jobs on specified clusters.
--format=<spec> (e.g. "--format=%i %j")	Output format to display. Specify fields, size, order, etc.
--jobs<job_id_list>	Comma separated list of job IDs to display.
--name=<name>	View only jobs with specified names.
--partition=<names>	View only jobs in specified partitions.
--priority	Sort jobs by priority.
--qos=<name>	View only jobs with specified Qualities Of Service.
--start	Report the expected start time and resources to be allocated for pending jobs in order of increasing start time.
--state=<names>	View only jobs with specified states.
--users=<names>	View only jobs for specified users.

sinfo - View information about nodes and partitions.

--all	Display information about all partitions.
--dead	If set, only report state information for non-responding (dead) nodes.
--format=<spec>	Output format to display.
--iterate=<seconds>	Print the state at specified interval.
--long	Print more detailed information.
--Node	Print information in a node-oriented format.
--partition=<names>	View only specified partitions.
--reservation	Display information about advanced reservations.
-R	Display reasons nodes are in the down, drained, fail or failing state.
--state=<names>	View only nodes specified states.

scontrol - Used view and modify configuration and state. Also see the **sview** graphical user interface version.

--details	Make show command print more details.
--oneliner	Print information on one line.

Commands:

create *SPECIFICATION*	Create a new partition or .
delete *SPECIFICATION*	Delete the entry with the specified SPECIFICATION
reconfigure	All Slurm daemons will re-read the configuration file.
requeue JOB_LIST	Requeue a running, suspended or completed batch job.
show ENTITY ID	Display the state of the specified entity with the specified identification
update *SPECIFICATION*	Update job, step, node, partition, or reservation configuration per the supplied specification.

Environment Variables

SLURM_ARRAY_JOB_ID	Set to the job ID if part of a job array.
SLURM_ARRAY_TASK_ID	Set to the task ID if part of a job array.
SLURM_CLUSTER_NAME	Name of the cluster executing the job.
SLURM_CPUS_PER_TASK	Number of CPUs requested per task.
SLURM_JOB_ACCOUNT	Account name.
SLURM_JOB_ID	Job ID.
SLURM_JOB_NAME	Job Name.
SLURM_JOB_NODELIST	Names of nodes allocated to job.
SLURM_JOB_NUM_NODES	Number of nodes allocated to job.
SLURM_JOB_PARTITION	Partition/queue running the job.
SLURM_JOB_UID	User ID of the job's owner.
SLURM_JOB_USER	User name of the job's owner.
SLURM_RESTART_COUNT	Number of times job has restarted.
SLURM_PROCID	Task ID (MPI rank).
SLURM_STEP_ID	Job step ID.
SLURM_STEP_NUM_TASKS	Task count (number of MPI ranks).

Daemons

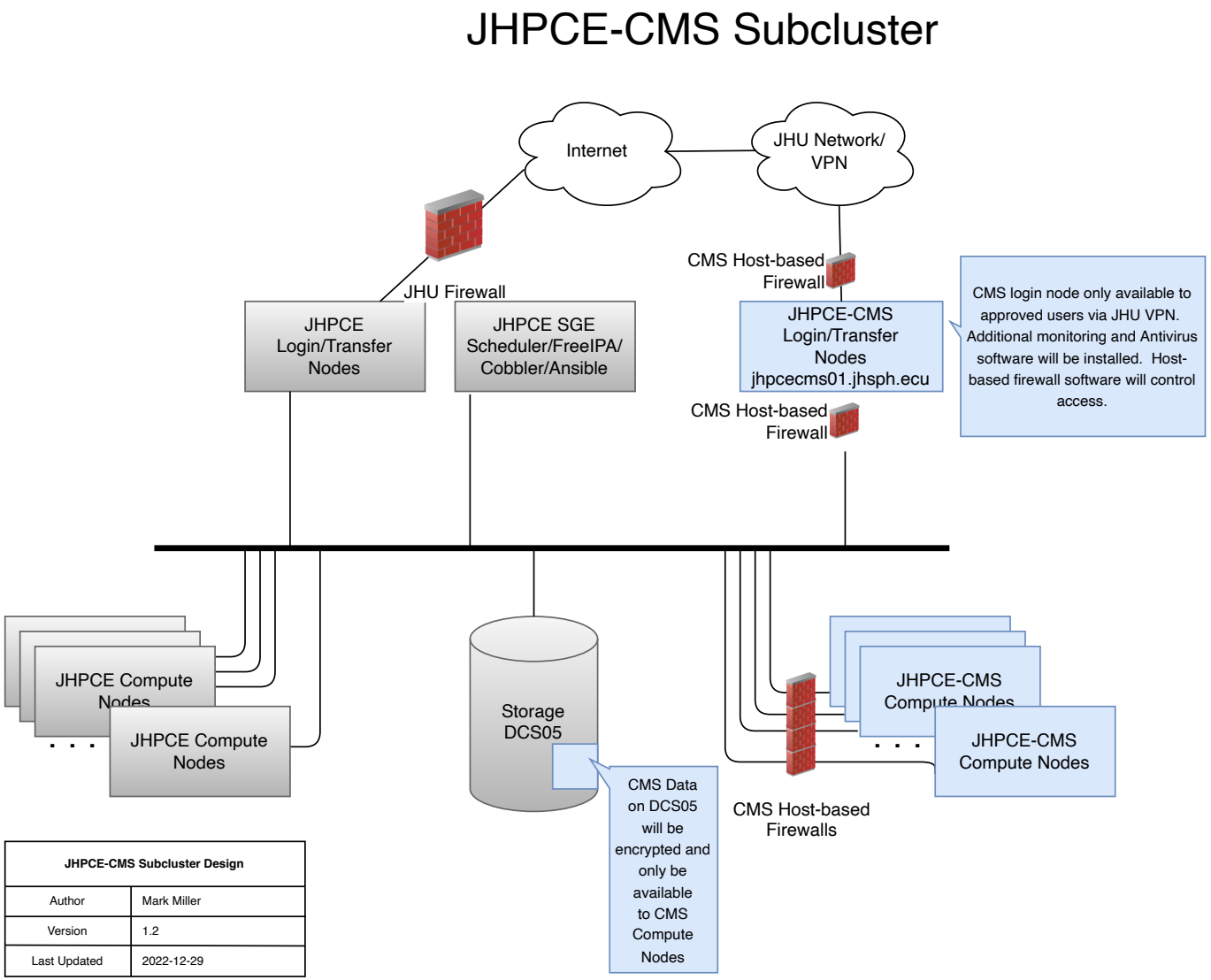
slurmctld	Executes on cluster's "head" node to manage workload.
slurmd	Executes on each compute node to locally manage resources.
slurmdbd	Manages database of resources limits, licenses, and archives accounting records.

SchedMD Slurm Support and Development

slurm workload manager

http://www.schedmd.com

JHPCE CMS Subcluster Schematic



Lab 2 - Using the cluster

Example 2a – using an **interactive** session

```
cd class-scripts
srun --pty --x11 bash
./script1                    # you could run any program this way
exit                  # log out of compute node, return  to login node
```

Example 2b – submitting a **batch** job

```
cd class-scripts
sbatch script2    # note script2 doesn't need to be executable
squeue --me
sacct –j jobid
```

examine results files with the **`cat`** or **`less`** commands

```
slurm-JOBID.out slurm-JOBID.err
```

Note: your script and interactive shell will run in the same directory in which you ran sbatch or srun, unless the `--chdir` argument is used.

Never run a job on the login node!

The login node has many fewer resources than the compute nodes

- Always use "**`sbatch`**" or "**`srun`**" to make use of the compute nodes
- Jobs that are found running on the login node may be killed at will
- If you are going to be compiling programs, do so on a compute node via **`srun`**.
- Even something as simple as copying large files should be done via **`srun.`**

- **NOTE: THE CMS DATA IS NOT AVAILABLE ON THE LOGIN NODE. IT IS ONLY VISIBLE ON THE COMPUTE NODES (in** `/cms01/data/`**)**

Useful Slurm command arguments

`sinfo` – basic info about the cluster

`slurmpic` – better view of cluster status (our custom script, uses sinfo output)

`squeue` – shows information about running & pending jobs

```
squeue  # defaults to all jobs for all users
squeue --me -t r,pd  # just my running & pending jobs
```

`sacct` – shows information about completed jobs

```
sacct -aj JOBID  # -a short for --allusers
sacct -a -S=2022-12-1514:30 # started after 2:30pm
sacct -a --units=M –j 130.batch -o
JobID,MaxVMSizeNode,MaxVMSize,AveVMSize,MaxRSS,AveRSS,MaxDiskRead,MaxDiskWrite,AveCPUFreq,TRESUsageInMax%-20  # that is all on one line; capitalization does not matter
```

Always use –a because of some quirks with sacct. (By default sacct only shows you your own jobs. But it also behaves differently on the login node versus the compute nodes.)

`scancel` – deletes your job

```
scancel JOBID
scancel –u <username>  # cancels all of that user's jobs
```

Schedule

- Introductions – who are we, who are you?
- Terminology
- Logging in and account setup
- Basics of running programs on the cluster
- **Details – limits and resources**
- Examples
- Moving data into & out of C-SUB
- Addendum: Additional UNIX topics

Requesting additional RAM & cores

- By default, when you submit a job with sbatch, or run srun, you are allotted 5GB of RAM and 1 core for your job.

- You can request more RAM by setting the "--mem" or “--mem-per-cpu” options
  - --mem: memory per node (for all cores used)
  - --mem-per-cpu: memory per core (harder to accurately estimate)

- Examples:

```
sbatch --mem=10G job1.sh
```

or

```
srun --mem-per-cpu=5G --cpus-per-task=2 --pty --x11 bash
```

Estimating RAM usage

- No easy formula. Running an example job usu best.
- A good place to start is the size of the files you will be reading in. As a starting point, then add a bit extra
- You can add sstat commands to your sbatch scripts (more than once, if desired) to gather info from your running job:

```
sstat –a –o
JobID,MaxVMSizeNode,MaxVMSize,AveVMSize,MaxRSS,AveRSS,MaxDiskRead,MaxDiskWrite,AveCPUFreq,TRESUsageInMax –j ${SLURM_JOB_ID} # this is all entered on a single line
```

- Display the fields available for use with –o

```
sstat –e
sacct –e
```

(sstat only works for currently running jobs. Use sacct to see completed jobs.)

Setting a time limit for your job

- By default, when you submit a job with sbatch, or run srun, you are will have a 1 day time limit for your job.
- You can request more time with the --time option to sbatch and srun with the time in the format of DAYS-HH:MM:SS
- Your job will die at the end of the time limit!!

Examples:

- To set a 4 day limit for your job:

```
sbatch --time=4-00:00:00 job1.sh  # 4-00 also works
```

- To set an 8 hour time limit for your interactive session:

```
srun --time=08:00:00 --pty --x11 bash # 8:00 means 8 minutes
```

- Shorter jobs are given higher priority via the “backfill scheduler”
- More information at: https://jhpce.jhu.edu/slurm/time-limits/

Supplying options to your sbatch job

- You can supply SLURM directives to sbatch in 4 ways:
- Order of precendence:
  - On the command line

    ```
    $ sbatch --mail-type=FAIL,END --mail-user=john@jhu.edu script2
    ```

    Email notification is a great option for a **handful** of long running jobs. This is a **horrible** option for 1000s of jobs, and has caused users to have their email accounts suspended.
  - Environment variables (Advanced topic. See Addendum page)
  - Embedding them in your batch job script

    Lines which start with "#SBATCH" are interpreted as options to **`sbatch.`**

    Such lines must:
    - start at the very beginning of a line
    - come after the interpreter line #!/bin/bash
    - before any commands

    ```
    $ less script3.annotated # this file contains many examples!!
    ```

Supplying options to your sbatch job (cont'd)

4. In your ~/.slurm/defaults file

```
Simple syntax is: <directive> = <value>
Full syntax is: [<command>:<cluster>:] <directive> = <value>
Where [ ] indicates an optional argument
Command can be one of (at least): srun, sbatch, salloc or *
(But perhaps other commands also refer to the file???)
You need to specify an asterisk in between colons (or a SLURM
cluster name (ours are "jhpce3" & "cms")
Blank and #-commented lines are allowed.
```

Example contents:

```
mem=2GB
mail-user=franksmith@jh.edu
:*:time=5-00
srun:*:partition=interactive
sbatch:*:mail-type=FAIL,END
```

Schedule

- Introductions – who are we, who are you?
- Terminology
- Logging in and account setup
- Basics of running programs on the cluster
- Details – limits and resources
- **Examples**
- Moving data into & out of C-SUB
- Addendum: Additional UNIX topics

Modules

Modules are sets of configuration information which change your environment to suite a particular software package.

We have modules for multiple versions of R, SAS, Mathematica, python . . .

- module list
- module avail
- module avail stata
- module load
- module unload
- module describe

Lab 3

Running R on the cluster:
- In `$HOME/class-scripts/R-demo`, note 2 files – Script file and R file

- Submit Script file
  - `sbatch plot1.sh`
- Run R commands interactively
  - `srun --pty --x11 bash`
  - `module load conda_R`
  - `R`
  - `Open and run plot1.r`

```
[compute-132 /users/myjhedid/class-scripts/R-demo]$ cat plot1.sh
#!/bin/bash
# Run the "R" program to read in the "plot1.r" script.

echo "`date`: Loading R Module"
module load conda_R

echo "`date`: Running R Job"
R CMD BATCH plot1.r

echo "`date`: R job complete"
exit 0
[compute-132 /users/myjhedid/class-scripts/R-demo]$
```

plot1.r creates plot1-R-results.pdf which you can view with xpdf or a web browser (firefox or chromium-browser)

Lab 4

Running RStudio

- X Windows Setup
  - For Windows, MobaXterm has an X server built into it
  - For Mac, you need to have the Xquartz program installed (which requires a reboot), and you need to add the "-X" option to ssh:

```
$ ssh -X c-yourjhedid-dua#@jhpcecms01.jhsph.edu
```

- Start up Rstudio

```
$ srun --pty --x11 --mem=10G bash
$ module load R
$ module load rstudio
$ rstudio
$ exit                          # log out of your srun session
```

Lab 5 – Running Stata

Batch:

```
$ cd $HOME/class-scripts/stata-demo
$ ls
$ less stata-demo1.sh   # see contents of the batch script
$ cat stata-demo1.do    # see contents of the stata program
$ sbatch stata-demo1.sh
$ cat stata-demo1.log   # see the output
```

Interactive:

```
$ srun --pty --x11 --cpus-per-task=4 bash
$ module load stata
$ stata-mp  # starts the multiprocessor version
```

or

```
$ xstata-mp # starts the GUI interface
```

Notes:

- The program and script do not need to be named the same, but it is good practice to keep them the same when possible.
- File extensions are sometimes meaningful in Linux. SAS doesn't care, but Stata programs need to have ".do" as the extension. It is good practice for human readability.
- By default "stata" runs a single thread. For faster results when running on real data, request 2 or more cores and use the command "stata-mp" instead of "stata"
- By default stata stores temporary files in /tmp. You may need to define an environmental variable to avoid job failure due to lack of space. `export STATATMP=$HOME`

Lab 6 – Running SAS

- SAS example:

Batch:

```
$ cd $HOME/class-scripts/SAS-demo
$ ls
$ cat sas-demo1.sh
$ cat class-info.sas
$ sbatch sas-demo1.sh
```

Interactive:

```
$ srun --pty --x11 bash      # or use handy bash routine named: jsrun
$ module load sas
$ csas –WORK /tmp            # note the single hyphen used here
$ exit                       # log out of your srun session
```

To display a plot, sas needs to send it to a web browser. We have created bash routines which start sas configured to launch Firefox (fsas) or Chrome (csas).

To see bash routines, run `declare –f routine_name`
Example routine definitions:

```
jsrun ()
{
    /usr/bin/srun --pty --x11 "$@" bash
}

csas ()
{
    sas –helpbrowser SAS –xrm "SAS.webBrowser:'/usr/bin/chromium-browser'" –xrm
"SAS.helpBrowser:'/usr/bin/chromium-browser'" "$@" > /dev/null 2>&1
}
```

Some GUI Utility Programs You Might Consider

Last updated: 20240425 C-SUB ROCKY-9.2

CATEGORY	CLI? GUI?	NAME (cmd)	DESCRIPTION	NOTABLE FEATURES	HOMEPAGE & TUTORIALS	COMMENTS
TEXT EDITORS						
	CLI-ish	nano	simple text editor	simple, undo/redo, regular expressions-capable search and replace	https://en.wikipedia.org/wiki/GNU_nano	Fast. More features than you would think. Read the built-in help.
	CLI	emacs	powerful text editor	autosave, spell-checking, regular expressions-capable search and replace	https://en.wikipedia.org/wiki/GNU_Emacs https://www.gnu.org/software/emacs/	Opens an X11 window if DISPLAY is set. If you want the CLI version, use the "--no-window-system) (shortcut: "-nw") argument. To exit from CLI version, type control-x then control-c. First released 20 March 1985
	GUI	gedit	extensible text editor	file browser in side panel, multiple files in tabs, undo/redo, regular expressions-capable search and replace	https://wiki.gnome.org/Apps/Gedit https://en.wikipedia.org/wiki/Gedit	
	GUI	geany	extensible text editor	Column / block / vertical select (via Shift + Ctrl + arrow keys)	https://www.geany.org/manual/current/index.html https://en.wikipedia.org/wiki/Geany	Very powerful, esp if you enable plugins (under Tools menu).
	GUI	oowriter	WYSIWG text editor	compatibility with Microsoft Office	https://www.libreoffice.org/discover/writer/	
FILE MANAGERS						
	GUI	thunar	GUI file manager	bulk renaming, custom actions, fast	https://en.wikipedia.org/wiki/Thunar https://docs.xfce.org/xfce/thunar/start	View menu—>Show Hidden Files
	GUI	nautilus	GUI file manager	tabs	https://apps.gnome.org/Nautilus/	As with alll of these apps, useful features enabled by visiting preferences. Some features won't work b/c we haven't installed the whole GNOME constellation of pkgs.

Some GUI Utility Programs You Might Consider

GRAPHIC EDITORS						
	GUI	inkscape	Vector image editor	many features	https://inkscape.org/about/features/	Fast
	GUI CLI	ImageMagick	Image editors, viewer	very scriptable	https://github.com/ImageMagick/ImageMagick6	Many programs available, "display" is viewer
	GUI	oodraw	Vector image editor	Full featured. Flow charts.	https://www.libreoffice.org/discover/draw/	
	GUI	oomath	Equation & formula editor	Fractions, exponents, formulas, …	https://www.libreoffice.org/discover/math/	
GRAPHIC VIEWERS						
	GUI	gv	Postscript & PDF viewer	Many options		
	GUI	gs	Postscript & PDF viewer	This is Ghostscript Many options	https://www.geeksforgeeks.org/gs-command-in-linux-with-examples/#	Use "gs -h" to see possible output "devices" you can specify, such as "jpeg", "pdfwrite", "textwrite" or "x11"
	GUI	xpdf	Postscript & PDF viewer			
TERMINAL EMULATORS		MobaXterm is probably the best for Windows users.				
	GUI	terminator	enhanced terminal emulator (can use instead of xterm)	can split horizontally or vertically, find text in session, horizontal & vertical scrolling, safe quit	https://en.wikipedia.org/wiki/Terminator_(terminal_emulator)	
SPREADSHEETS						
	GUI	oocalc	WYSIWG spreadsheet	Full featued.	https://www.libreoffice.org/discover/calc/	
WEB BROWSERS						
	GUI	chromium-browser		You can browse local files, in order to for example look at a JPG, PDF or TXT file		Enter ~ in the location bar to open your home directory.
	GUI	firefox		same as above		same as above

LibreOffice offers MS Office Compatibility

Microsoft Office (MSO) is not available for Linux. The LibreOffice (LO) suite is installed on the compute nodes and offers the same functionality. It can read and write in MSO file formats, although esp complicated documents are more likely to experience formatting differences.

LO by default saves files in Open Document Format (*.odf). If you want to export you will need to use the File->Save As… or File->Export

- libreoffice – app launcher for whole suite
- oocalc – spreadsheets
- oowriter – text documents
- oodraw – vector drawing
- oomath – create formula for inclusion as figures

Lab 7 – Some useful GUI tools

```
jhpcecms01$ srun --pty --x11 bash
compute-132$ xterm &
compute-132$ terminator &
compute-132$ thunar &
compute-132$ geany &
compute-132$ gedit &
```

Notes:

- Ampersand runs program in background
- xterm – alternative to opening a new login via Apple Terminal or Windows MobaXterm session
- terminator – accepts mouse right-clicks, can split window into two (each running its own bash shell)
- thunar – can view & change file permissions (File->Properties)
- thunar – (Edit -> Preferences)
- geany & gedit – vital to enable plugins, explore them

C-SUB File Systems

Files of different types are placed in separate locations to help implement security controls.

Type of Files	Is Stored Under	Available on Which Computers?
User Directories	/users/	All
Data Files	/cms01/data/	Only compute nodes
Uploaded Files	/cms01/incoming/	All (read-only on compute nodes)
Files Which Can Be Downloaded	/cms01/outgoing/	Only jhpcecms01

C-SUB File Systems, cont'd

jhpcecms01
/users/
/cms01/incoming/
/cms01/outgoing/

compute-132 compute-133
/users/
/cms01/data/
/cms01/incoming/ (read-only)

The login node, jhpcecms01, is configured such that:

- users cannot write into the outgoing file system. Only the Data Custodian can.
- users cannot upload files via SFTP into the outgoing file system.

The compute nodes are configured such that:

- they have access to the restricted data files
- users can copy their uploaded files out of incoming to their home directories
- the Data Custodian can copy uploaded data files into the data file system.

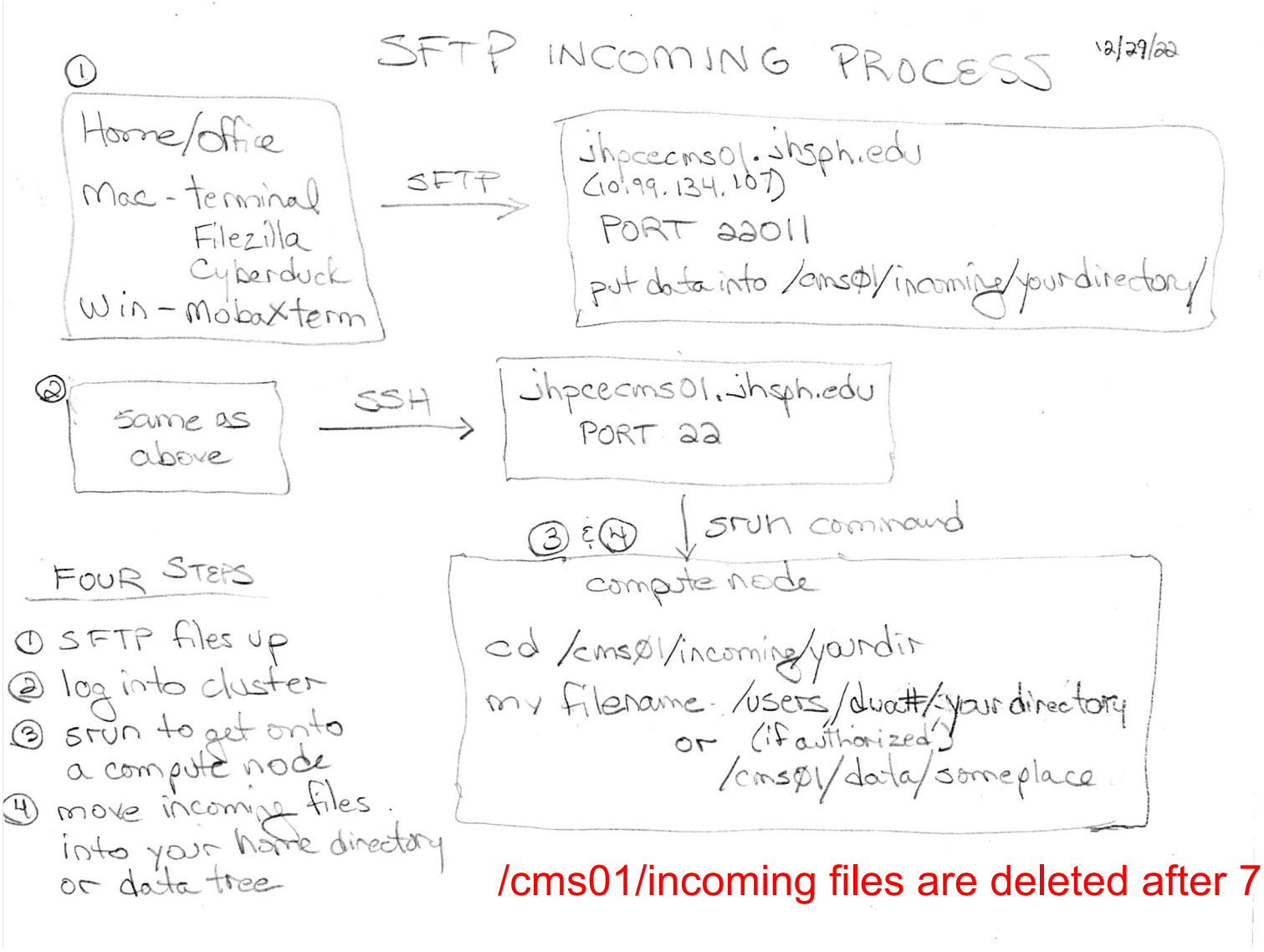
Moving data into & out of C-SUB

C-SUB users have strict obligations to keep confidential data protected. Cluster computers have been modified to assist you in keeping restricted data from leaking.

Data movement in and out of C-SUB occurs in specific ways.

- Only SFTP is allowed in or out of C-SUB. No scp or rsync.
- Incoming data is SFTP'd to port 22011 of jhpcecms01.jhsph.edu
- Incoming data can be saved only in `/cms01/incoming/yourusername`
- You can copy those files into your home directory by using srun to log into a compute node
- Outgoing data is reviewed by the appropriate Data Custodian on request (send email about it to support@harp-csub.freshdesk.com)
- You place your proposed outgoing data in a specific directory: ~/proposed/
- That directory has special permissions which allow the Data Custodian to view
- Once approved, you can retrieve the files using SFTP from the directory `/cms01/outgoing/yourusername`
- Outgoing data is SFTP'd from port 22027 of jhpcecms01.jhsph.edu
- Ports 22011 & 22027 are not standard – you will need to configure your SFTP client program to use the right one for the desired data direction.
- Files are deleted automatically after 7 days from both /cms01/incoming and /cms01/outgoing

Moving data into C-SUB



/cms01/incoming files are deleted after 7 days.

Moving data out of C-SUB

No diagram yet. Outgoing is the reverse of what is shown on the incoming slide except for the placement of files into `~/proposed/` and the email back & forth between you and the Data Custodian.

1. User copies material into `~/proposed/`
2. User notifies Data Custodian (DC) via email support@harp-csub.freshdesk.com , describing the material, how it was generated and from what data it was derived
3. The DC cd's into `~/proposed/`
4. The DC reviews the files
5. If unsatisfied, they email the user.
6. If satisfied, they copy files from `~/proposed/` into `/cms01/outgoing/user/` and email the user.
7. User retrieves files via SFTP to port 22027 of jhpcecms01.jhsph.edu
8. Files older than 7 days are deleted from `/cms01/outgoing/user/`

SFTP – A Secure File Transfer Protocol

SFTP allows you to “put” files to, or “get” files from, a remote computer
SFTP programs are available in both Command Line and Graphic User interfaces

Example of uploading a file from a macOS Terminal window:

```
[bash]$ sftp -P 22011 c-jtuniso1-10101@jhpcecms01.jhsph.edu
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Verification code:
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Password:
Connected to jhpcecms01.jhsph.edu.
sftp> ls
README.txt        c-cnelso65-98765   c-jtuniso1-10101   c-jxu123-55548     c-mmill116-10101
c-mtrieb2-10201   c-tbrow261-55548   dev
sftp> cd c-jtuniso1-10101
sftp> put 2021.pdf
Uploading 2021.pdf to /c-jtuniso1-10101/2021.pdf
2021.pdf                                      100%  169KB   1.0MB/s   00:00
sftp> quit
```

Here is a slick one-liner which uploads a directory named “2021”, recursively:

```
[bash]$ sftp -P 22011 c-jtuniso1-10101@jhpcecms01.jhsph.edu:c-jtuniso1-10101 <<< $'put –r 2021'
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Verification code:
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Password:
Connected to jhpcecms01.jhsph.edu.
Changing to: /c-jtuniso1-10101
sftp> put 2021.pdf
Uploading 2021.pdf to /c-jtuniso1-10101/2021.pdf
2021.pdf                                      100%  169KB   1.0MB/s   00:00
sftp> quit
```

SFTP – A Secure File Transfer Protocol

SFTP has a –r (recursive) option, too, by the way, for put and get

Example of downloading a file from a macOS Terminal window:

```
[bash]$ sftp -P 22027 c-jtuniso1-10101@jhpcecms01.jhsph.edu
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Verification code:
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Password:
Connected to jhpcecms01.jhsph.edu.
sftp> ls
README.txt        c-cnelso65-98765   c-jtuniso1-10101   c-jxu123-55548     c-mmill116-10101
c-mtrieb2-10201   c-tbrow261-55548   dev
sftp> cd c-jtuniso1-10101
sftp> get bob.tar
Fetching /c-jtuniso1-10101/bob.tar to bob.tar
/c-jtuniso1-10101/bob.tar                                   100% 3539    92.3KB/s   00:00
sftp> quit
```

Here is a slick one-liner which accomplishes the same task:

```
[bash]$ sftp -P 22027 c-jtuniso1-10101@jhpcecms01.jhsph.edu:c-jtuniso1-10101/bob.tar bob.tar
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Verification code:
(c-jtuniso1-10101@jhpcecms01.jhsph.edu) Password:
Connected to jhpcecms01.jhsph.edu.
Fetching /c-jtuniso1-10101/bob.tar to bob.tar
/c-jtuniso1-10101/bob.tar                                   100% 3539    92.3KB/s   00:00
sftp> quit
```

SFTP – A more secure file transfer protocol

A good SFTP Graphic User interface program on Windows is MobaXterm

When configuring an SFTP session in MobaXterm, you need to check a box indicating that multi-factor authentication will be occurring!!!

SSH programs (ssh, sftp) can use configuration files where you can store defaults. For example, on a Mac, a file named ~/.ssh/config can store entries like these, which allow me to sftp in and out with fewer keystrokes:

```
#       incoming
 Host cms-sftpin
     User c-jtuniso1-10101
     Hostname jhpcecms01.jhpce.jhu.edu
     Port 22011

#       outgoing
 Host cms-sftpout
     User c-jtuniso1-10101
     Hostname jhpcecms01.jhpce.jhu.edu
     Port 22027
```

So I can use those entries with a simple “sftp cms-sftpin” etc.

Summary

- Review
  - Familiarize yourself with Linux & X11
  - Use ssh to connect to JHPCE cluster
  - Use sbatch and srun to submit jobs
  - Never run jobs on the login node
  - Helpful resources
    - http://www.jhpce.jhu.edu/
    - bitsupport@lists.johnshopkins.edu - System issues
    - bithelp@lists.johnshopkins.edu – Complex application issues
    - support@harp-csub.freshdesk.com – data export & import requests
- What to do next
  - Make note of your Google Authenticator scratch codes (option 2 in "auth_util")
  - Play nice with others – this is a shared community-supported system.

Thanks for attending! Questions?

Addendum: Additional UNIX/SLURM topics

Some additional resources for you to consult if you want to improve your UNIX & SLURM skills

- Some GUI Utility Programs You Might Consider
- Manual Pages
- Text Processing Tools on UNIX
- Helpful UNIX Concepts To Know
- Shell Aliases
- LibreOffice Information
- SLURM Environment Variables
- Traversing deep directory structures

Manual Pages

Most commands have manual pages which describe their function & arguments. Because one must be very specific when using the command line interface, it is important to understand which arguments are optional and which are required. Here examples of the primary kinds of notation you will see, and comments about the syntax. Press “q” to exit from a manual page session.

Man command – To learn more about it run: man man

Example Usage statements:

man -l [**-C** file] [**-d**] [**-D**] [**--warnings**[=warnings]] [**-T**[device]]file **...**

man -K [**-w**|**-W**] [**-S** list] [**-i**|**-I**] [--regex] [section] term **...**

- Matters: capitalization, spaces or lack of, one dash or two
- Commands can have multiple usage statements, as shown
- Required – these are listed without any annotation
- Optional – indicated by [ ] brackets
- Exclusive – separated by a “pipe” symbol – you can only use one of them
- Repeatable – Items shown before an … (“ellipsis”) symbol may be repeated
- Values to arguments might need to single or double-quoted, for example if they contain spaces

CLI Text Processing Tools on UNIX

Excellent introduction to many of the commands below, with good detail on awk and sed
https://developer.ibm.com/articles/au-unixtext/

cat: Read lines from stdin (and more files), and concatenate them to stdout.
comm: Outputs lines common to two files or unique to them, provided the files are sorted.
csplit: Splits input into output files. The split can be driven by the number of lines and by a regex match.
cut: Cut specified byte, character, or field from each line of stdin and print to stdout.
diff: Identify differences between files
expand: Replaces tabs with spaces
fmt: Formats text, including reflowing paragraphs to a specific maximum number of characters per line.
fold: Limits maximum length of line, opposite of fmt
grep: Find lines in stdin that match a pattern and print them to stdout.
head: Read the first few lines from stdin (and more files) and print them to stdout.
join: Combines lines from files based of their fields, assuming the files are sorted on the fields used for joining.
less: A much better version of more
more: Read lines from stdin, and provide a paginated view to stdout.
paste: Read lines from stdin (and more files), and paste them together line-by-line to stdout.
pr: Formats input for printing, including pagination with header and footer.
sort: Sort the lines in stdin, and print the result to stdout.
tail: Read the last few lines from stdin (and more files) and print them to stdout.
tr: Translate or delete characters read from stdin and print to stdout.
unexpand: Converts spaces to tabs, defaulting to 8 spaces per tab
uniq: Read from stdin and print unique (that are different from the adjacent line) to stdout.
wc: Read from stdin, and print the number of newlines, words, and bytes to stdout.

I've installed these additional tools on jhpcecms01
---
dos2unix: convert plain text files from DOS or Mac format to Unix
unix2dos: convert plain text files from UNIX to DOS or Mac format

---
AWK and SED

AWK
https://awk.js.org/help.html
  these pages mentioned at bottom of above page:
        HANDY ONE-LINE SCRIPTS FOR AWK
        https://www.pement.org/awk/awk1line.txt

Important UNIX Concepts To Master

Processes

- Parents/children, Inheritance of environment
- Identifying: ps -ef
- Signals, Killing: kill –KILL *pid*

Input/Output Redirection

- Stdin, Stdout, Stderr
- Pipes

Job Control

- Foreground, Background
- Cancel (^C), Stop (pause) (^Z) – (Not actually capitalized)
- https://v4.software-carpentry.org/shell/job.pdf

Important UNIX Concepts To Master (cont'd)

X11 or "the X Windowing System"

- DISPLAY environment variable
- xauth program
- The –X argument to ssh (which enables X support for that connection)
- Test functionality of your X session by running a simple program such as "xeyes" or "xclock"

Shell aliases – Helpful? You Decide!

A note about removing files.
An alias for rm (as well as cp and mv) is defined in a file that is read in by your .bashrc configuration file (/etc/bashrc.c-sub).

Those aliases are:

```
alias rm='rm -i'
alias mv='mv -i'
alias cp='cp -i'
```

The -i or interrogative flag says to ask if you're sure. In particular, they prevent one from unintentionally overwriting existing files with mv and cp

But being asked, file by file, if you want to remove a bunch of files is obviously a sad experience.

The ways to get around the aliases are:
1) unalias rm, then proceed to delete files in that shell without the alias in place
2) Issue a remove command with a backslash: \rm
3) Issue a remove command with a full path: /bin/rm
4) Insert at the bottom of your .bashrc "unalias rm" so that future shells are not equipped with that alias

Also, of course, the two flags -r and -f to remove are helpful.
-r means recursively remove everything under the path you specify
-f means force, or "don't complain to me about that!"

One can see all aliases with the "alias" command.

You can define your own by adding them to your .bashrc file.
You can, as seen above, undo ones set by the defaults, by unaliasing or redefining them in your .bashrc file (AFTER they are defined elsewhere).

LibreOffice Information

In general, you are advised to use productivity apps like Microsoft Office on other computers over LibreOffice for significant editing.

- Built-in help packages are not installed. It is better to use your local machine's web browsers.
- There are command-line arguments you can use, including ones which allow for the conversion of documents, which might be useful. For example: file conversion - PDF, HTML, DOC, DOCX, EPUB, plain text, and many more with commands like:
  `libreoffice --headless --convert-to epub example.odt`

- Note which version you are using and use the correct help section.

  https://help.libreoffice.org/latest/he/text/shared/guide/start_parameters.html

SLURM Environment Variables

Environment variables can be used to:

- pass parameters to your shell scripts
  - See --export option to sbatch (read the manual page with: `man sbatch`)
  - Scripts need to be written to look for them
- define SLURM directives
  - Remember their order of precedence (after command line arguments and before #SBATCH lines)
  - Some examples (which point out that there are more variables than just SLURM_ ones, e.g. SALLOC_ ones) can be found here: https://uwaterloo.ca/math-faculty-computing-facility/services/service-catalogue-teaching-linux/job-submit-commands-examples#slurm-options

SAFE Desktop

https://ictr.johnshopkins.edu/service/informatics/safe-desktop/

We encourage you to request an account for this service.

It is a secure virtual Windows environment which provides Johns Hopkins Medicine investigators with a secure environment to analyze and share sensitive data (e.g. PHI, PII) with colleagues. It is equipped with Stata, SAS, MobaXterm, R Studio, MS Office and other useful programs.

You can use it to access C-SUB via MobaXterm, which is useful if you need to leave a session running while you, say, go home for the weekend while something interactive is in-progress. Or if you need to access C-SUB while not at your normal computer. It is a good place to improve your skills with analytical software.

--However, you cannot copy CMS data there to work on or share it. CMS data MUST remain in the C-SUB.--

Traversing deep directory structures

If you have to switch between two directories with long paths, these two techniques can make life better.

(A Nest of) Symbolic Links

A symbolic link is a special kind of file which points at another file. Also known as symlinks. The symbolic link takes up almost no space. It is not a copy of the original file.

You can refer to the symbolic link and in most cases* the results will be the same as if you specified the original file. (*=some commands treat symbolic links in ways you might not expect. cp, rm, rsync, tar Their man pages will discuss how they treat symlinks.)

You create a symlink like this:

```
ln –s realfile newname
```

The resulting files when listed with `ls –l`:

```
lrwxr–xr–x  1 tunison  wheel  8 Jan 11 10:12 newname@ –> realfile
–rw–r––r––  1 tunison  wheel  0 Jan 11 10:11 realfile
```

So a way to make use of symlinks is to create a directory of them, and refer to those when doing things like changing directories. I use the following scheme myself. You can name your directory whatever you want, such as “redirect” instead of the shorter ”r”.

```
mkdir ~/r
cd ~/r
ln –s /cms01/data/puf–free/VERICRED/2019 v2019
ln –s /cms01/incoming/c–myjhed–98765 in
ln –s ~/mycode/thatlanguage/src/yes–i–wrote–it pride–n–joy
```

Now you can use the symlinks named “v2019”, “in” “pride-n-joy” instead of the longer real directory names.

Traversing deep directory structures (cont'd)

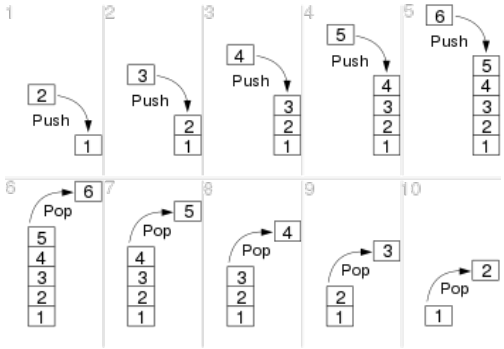
Symbolic Links (cont'd)

You can change directory with `cd ~/r/in` and you will wind up in `/cms01/incoming/c-myjhed-98765`

Or, You can copy downloaded source code into your home directory with
`cp ~/r/in/pkg-3.2.tar ~/mycode/thatlanguage/src/`

Pushd, Popd and Dirs Commands

If you're repeatedly working in several directories and don't need to open multiple windows to look at them simultaneously, these commands allow you to switch rapidly between them. They use a "stack" data structure. Think of a stack as a pile of plates in a cafeteria. When you want one, you usually take the one from the top. Then a fresh one is exposed. You've "popped" a plate from the stack. If you "push" three plates onto the stack, then the stack is deeper. You can access the top plate easily, but in this case you can also get at the third plate down.



`pushd directoryname`
changes your directory from the current one to directoryname, and creates a stack of two directories.

`dirs -v`
will list those directories in the stack.

`pushd`
by itself will switch you between the current directory and the top one in the stack. This is often the main way I use it.

`pushd +2`
will switch you between the current directory and the third down into the stack. (third because the index into the stack starts with 0 not 1)

`popd`
will cd back to the top directory in the stack while removing your current directory from the stack.

Traversing deep directory structures (cont'd)

Pushd, Popd and Dirs Commands (cont'd)

You can create a stack ahead of time using the –n option to pushd. That option adds the directory to the stack but does not change to it.

So if you used the nano text editor to add this to your .bashrc file, it would create a set of directories you can pushd between every time you log in!!!

```
# set up directory stack. Note that they appear in stack in reverse order than
listed here

# dirs –v will show their order (and index number)

for i in /cms01/data /cms01/outgoing /users/55548 /cms01/incoming; do pushd –n $i
1>/dev/null;done
```

Here we use that technique after adding it to our .bashrc file:

```
[~]$ source .bashrc

[~]$ dirs
~ /cms01/incoming /users/55548 /cms01/outgoing /cms01/data

[~]$ pushd +2
/users/55548 /cms01/outgoing /cms01/data ~ /cms01/incoming

[55548]$ pwd
/users/55548

[55548]$ dirs –v
 0  /users/55548
 1  /cms01/outgoing
 2  /cms01/data
 3  ~
 4  /cms01/incoming
```

Copying deep directory structures

The rsync command is much better than cp or mv

Rsync is a program which copies files from a source to a destination location. It has many available arguments but they aren't needed in most cases. Rsync's key utility is that it will compare the source and destination locations and only copy changed or missing files to the destination. If

A symbolic link is a special kind of file which points at another file. Also known as symlinks. The symbolic link takes up almost no space. It is not a copy of the original file.

You can refer to the symbolic link and in most cases* the results will be the same as if you specified the original file. (*=some commands treat symbolic links in ways you might not expect. cp, rm, rsync, tar Their man pages will discuss how they treat symlinks.)

```
mkdir ~/r
cd ~/r
ln –s /cms01/data/puf–free/VERICRED/2019 v2019
ln –s /cms01/incoming/c–myjhed–98765 in
ln –s ~/mycode/thatlanguage/src/yes–i–wrote–it pride–n–joy
```

Now you can use the symlinks named "v2019", "in" "pride-n-joy" instead of the longer real directory names.

SLURM Documentation

(Our SLURM is version 20.11.09. So one should look at the docs here https://slurm.schedmd.com/archive/slurm-20.11.9/)