Introduction to High Performance Computing (HPC) on Lewis and Clark clusters

Research Computing Support Services (RCSS)
http://docs.rnet.missouri.edu

RCSS CIE team
Ashkan Mirzaee, Asif Magdoom
Brian Marxkors, Christina Roberts
Predrag Lazic
What is a cluster (supercomputer)?

- A supercomputer is a computer with a high level of performance as compared to a general-purpose computer.
- Purpose: massive parallelization because life is too short!
- The world's fastest 500 supercomputers run Linux-based operating systems.
Lewis and Clark

**Lewis**
- A large-scale cluster for requesting high amount of resources
- Great for parallel programming
- GPU resources
- No cost for MU members for general usage
- Investment option is available to receive more resource (more fairshare)

**Clark**
- Great for learning and teaching
- No need for registration and it is available to all MU members by MU username and PawPrint
- Usually less crowded - receive resources very fast
- No cost for MU members
# Lewis and Clark Partitions

<table>
<thead>
<tr>
<th>Partition Name</th>
<th>Time Limit</th>
<th>Nodes</th>
<th>Cores (per node*)</th>
<th>Cores (total)</th>
<th>Memory in GB (per nodes*)</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive</td>
<td>4:00:00</td>
<td>4</td>
<td>24+</td>
<td>144</td>
<td>251+</td>
<td>Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz+</td>
</tr>
<tr>
<td>General</td>
<td>4:00:00</td>
<td>187</td>
<td>24+</td>
<td>5636</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz+</td>
</tr>
<tr>
<td>BioCompute</td>
<td>2:00:00:00</td>
<td>37</td>
<td>56</td>
<td>2072</td>
<td>509</td>
<td>Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz</td>
</tr>
<tr>
<td>Lewis</td>
<td>2:00:00:00</td>
<td>90</td>
<td>24+</td>
<td>3564</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz+</td>
</tr>
<tr>
<td>hpc3</td>
<td>2:00:00:00</td>
<td>22</td>
<td>24</td>
<td>1296</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz</td>
</tr>
<tr>
<td>hpc4</td>
<td>2:00:00:00</td>
<td>37</td>
<td>28</td>
<td>1260</td>
<td>251+</td>
<td>Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz</td>
</tr>
<tr>
<td>hpc4rc</td>
<td>2:00:00:00</td>
<td>34</td>
<td>28</td>
<td>1008</td>
<td>251</td>
<td>Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz</td>
</tr>
<tr>
<td>hpc6</td>
<td>2:00:00:00</td>
<td>61</td>
<td>48</td>
<td>2976</td>
<td>379+</td>
<td>Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz</td>
</tr>
<tr>
<td>hpc5</td>
<td>2:00:00:00</td>
<td>35</td>
<td>40</td>
<td>1320</td>
<td>379</td>
<td>Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz</td>
</tr>
<tr>
<td>Gpu</td>
<td>2:00:00</td>
<td>16</td>
<td>16+</td>
<td>372</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz+</td>
</tr>
<tr>
<td>gpu3</td>
<td>2:00:00:00</td>
<td>13</td>
<td>16+</td>
<td>284</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
</tr>
<tr>
<td>gpu4</td>
<td>2:00:00:00</td>
<td>3</td>
<td>40+</td>
<td>124</td>
<td>379+</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
</tr>
<tr>
<td>Serial</td>
<td>2:00:00:00</td>
<td>1</td>
<td>64</td>
<td>64</td>
<td>1,025</td>
<td>AMD EPYC 7601 32-Core Processor</td>
</tr>
<tr>
<td>Dtn</td>
<td>2:00:00:00</td>
<td>2</td>
<td>16+</td>
<td>36</td>
<td>66+</td>
<td>Intel(R) Xeon(R) CPU X5550 @ 2.67GHz+</td>
</tr>
</tbody>
</table>

* plus sign (+) indicates a mixed environment. The number before the plus represents the minimum

http://docs.rnet.missouri.edu/policy/partition-policy/
Slurm is a system for cluster management and job scheduling. All RCSS clusters use Slurm (https://slurm.schedmd.com).

- Slurm is a workload scheduler and has set of tools for submitting and monitoring jobs
- Slurm is a resource management system and has many tools to find available resources in the cluster
- All Slurm commands start with letter “s”. In this course we will learn many of them
- Resource allocation depends on your fairshare i.e. priority in the queue
Login node

All users connect to Clark and Lewis clusters through the login nodes.

```
[user@lewis4-r630-login-node675 ~]$  
[user@clark-r630-login-node907 ~]$  
```

All jobs must be run using Slurm submitting tools to prevent running on the Lewis login node. Jobs that are found running on the login node will be immediately terminated followed up with a notification email to the user.
# Unix commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>man</code> manual</td>
<td>Display manual page</td>
</tr>
<tr>
<td><code>cd</code> change directory</td>
<td>Change directory to specified one</td>
</tr>
<tr>
<td><code>pwd</code> print working directory</td>
<td>Print working directory</td>
</tr>
<tr>
<td><code>ls -la</code> long list of all files</td>
<td>List all files recursively with permissions</td>
</tr>
<tr>
<td><code>mkdir</code> make directory</td>
<td>Create a directory</td>
</tr>
<tr>
<td><code>cp</code> copy</td>
<td>Copy files</td>
</tr>
<tr>
<td><code>hostname</code> host name</td>
<td>Print server name</td>
</tr>
<tr>
<td><code>nproc</code> number of processors</td>
<td>Print number of processors</td>
</tr>
<tr>
<td><code>lscpu</code> list CPU architecture</td>
<td>List CPU architecture</td>
</tr>
<tr>
<td><code>free</code> -h free memories</td>
<td>Print free memory</td>
</tr>
<tr>
<td><code>df</code> disk free</td>
<td>Print disk usage</td>
</tr>
<tr>
<td><code>echo $USER</code> echo user id</td>
<td>Print user id</td>
</tr>
<tr>
<td><code>cat</code> concatenate files and print output</td>
<td>Concatenate files and print output</td>
</tr>
<tr>
<td><code>top</code> information about processes</td>
<td>Display process information</td>
</tr>
<tr>
<td><code>Tab</code> tab completion</td>
<td>Show tab completion</td>
</tr>
<tr>
<td><code>clear</code> or <code>Ctrl + l</code> clear</td>
<td>Clear screen</td>
</tr>
<tr>
<td><code>exit</code> or <code>Ctrl + d</code> exit</td>
<td>Exit terminal or clear screen</td>
</tr>
<tr>
<td><code>Ctrl + z suspend</code></td>
<td>Suspend terminal</td>
</tr>
<tr>
<td><code>fg</code> return</td>
<td>Bring terminal to foreground</td>
</tr>
<tr>
<td><code>Ctrl + x + Ctrl + c close</code></td>
<td>Close terminal</td>
</tr>
<tr>
<td><code>Ctrl + x + Ctrl + s save</code></td>
<td>Save changes to disk</td>
</tr>
<tr>
<td><code>Ctrl + o write out</code></td>
<td>Write current buffer to file</td>
</tr>
<tr>
<td><code>Esc d d</code> delete line</td>
<td>Delete line</td>
</tr>
<tr>
<td><code>Esc yy yank (paste)</code></td>
<td>Copy previous line to clipboard</td>
</tr>
<tr>
<td><code>Esc u undo</code></td>
<td>Undo last command</td>
</tr>
<tr>
<td><code>Esc p paste</code></td>
<td>Paste from clipboard</td>
</tr>
<tr>
<td><code>Esc + :w write</code></td>
<td>Write current buffer to file</td>
</tr>
<tr>
<td><code>Esc + :q quit :q! no change</code></td>
<td>Exit terminal without saving changes</td>
</tr>
</tbody>
</table>

# Text editors

<table>
<thead>
<tr>
<th>Editor</th>
<th>Key_bindings</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>emacs -nw</code></td>
<td><code>Ctrl-k</code> kill (cut)</td>
</tr>
<tr>
<td><code>nano -z</code></td>
<td><code>i</code> insert</td>
</tr>
<tr>
<td><code>vim</code></td>
<td><code>Alt-w</code> copy</td>
</tr>
<tr>
<td></td>
<td><code>Alt-6</code> copy</td>
</tr>
<tr>
<td></td>
<td><code>Shift-arrow keys</code> select</td>
</tr>
<tr>
<td></td>
<td><code>Alt-m + alt-a</code> Select</td>
</tr>
<tr>
<td></td>
<td><code>Ctrl-z suspend fg return</code></td>
</tr>
<tr>
<td></td>
<td><code>Ctrl-z suspend fg return</code></td>
</tr>
<tr>
<td></td>
<td><code>Esc Ctrl + z suspend fg return</code></td>
</tr>
<tr>
<td></td>
<td><code>Esc + :w write</code></td>
</tr>
<tr>
<td></td>
<td><code>Esc + :q quit :q! no change</code></td>
</tr>
</tbody>
</table>

# Command line utilities

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssh</code> <a href="mailto:username@lewis.rnet.missouri.edu">username@lewis.rnet.missouri.edu</a></td>
<td>Connect to Lewis by ssh</td>
</tr>
<tr>
<td><code>ssh</code> <a href="mailto:username@clark.rnet.missouri.edu">username@clark.rnet.missouri.edu</a></td>
<td>Connect to Clark by ssh</td>
</tr>
<tr>
<td><code>cp -r /group/training/hpc-intro/ ~</code></td>
<td>Copy training files to your home directory</td>
</tr>
</tbody>
</table>
Cluster info

`sinfo -s` summary of cluster resources
`sinfo -p <partition-name> -o %n,%C,%m,%z` compute info of nodes in a partition
`sinfo -p Gpu -o %n,%C,%m,%G` GPUs information in Gpu partition
`sjstat -c` show computing resources per node
`scontrol show partition <partition-name>` partition information
`scontrol show node <node-name>` node information
`scontrol show node <node-name>` node information
`sacctmgr show qos format=name,maxwall,maxsubmit` show quality of services
`./ncpu.py` show number of available CPUs and GPUs per node

* ***************************************** */ncpu.py show number of available CPUs and GPUs per node

Scheduling pool data:

<table>
<thead>
<tr>
<th>Pool</th>
<th>Memory</th>
<th>Cpus</th>
<th>Total Usable</th>
<th>Free</th>
<th>Other Traits</th>
</tr>
</thead>
<tbody>
<tr>
<td>r630-hpc3</td>
<td>122534Mb</td>
<td>24</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>hpc3</td>
<td>122534Mb</td>
<td>24</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>General*</td>
<td>122534Mb</td>
<td>24</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

CPUS/NODES(A/I/O/T) count of CPUs/nodes in the form "available/idle/other/total"

S:C:T counts number of “sockets, cores, threads”

The Scheduling data contains information pertaining to the:

- **Pool**: a set of nodes
- **Memory**: the amount of memory on each node
- **Cpus**: the number of cpus on each node
- **Total**: the total number of nodes in the pool
- **Usable**: total usable nodes in the pool
- **Free**: total nodes that are currently free
Users info

**sshare - sacctmgr - df - lfs quota**

- **sshare**
  - `sshare -U [username]` show your fairshare and accounts
- **sacctmgr**
  - `sacctmgr show assoc user=$USER format=acc,user,share,qos,maxj` your QOS
- **Groups**
  - `sacctmgr show your groups`
- **df - h**
  - `df -h /home/$USER` home storage quota
- **lfs quota**
  - `lfs quota -hg $USER /storage/hpc` data/scratch storage quota
  - `lfs quota -hg <group-name> /storage/hpc` data/scratch storage quota
- **./userq.py**
  - `./userq.py` show user’s fairshare, accounts, groups and QOS

- Resource allocation depends on your fairshare. If your fairshare is **0.000000** you have used the cluster more than your fair share and will be de-prioritized by the queuing software.
- Users have 5GB at their home directory `/home/$USER` and 100GB at `/storage/hpc/data/$USER`
- **Do not** use home directory for running jobs, storing data or virtual environments.
- Clark users have 100G on their home storage. The above methods does not apply for Clark.
- The RCSS team reserves the right to delete anything in `/scratch` and `/local/scratch` at any time for any reason.
- There are **no backups** of any storage. The RCSS team is not responsible for data integrity and data loss. You are responsible for your own data and data backup.
- Review our storage policy at [http://docs.rnet.missouri.edu/policy/storage-policy/](http://docs.rnet.missouri.edu/policy/storage-policy/)

**Note:** $USER is a default environmental variable that returns your user id, try `echo $USER`
Job submission

All jobs must be run using `srun` or `sbatch` to prevent running on the Lewis login node.

**srun: requesting resources to run jobs**

- `srun <slurm-options> <software-name/path>`
- `srun --pty /bin/bash` requesting a pseudo terminal of bash shell to run jobs interactively
- `srun -p Interactive --pty /bin/bash` requesting a p.t. of bash shell in Interactive Node
- `srun -p <partition-name> -n 4 --mem 16G --pty /bin/bash` req. 4 tasks* and 16G memory
- `srun -p Gpu --gres gpu:1 -N 1 --ntasks-per-node 8 --pty /bin/bash` req. 1 GPU and 1 node for running 8 tasks on Gpu partition

*Slurm by default allocates 1 CPU per tasks

**sbatch: submitting jobs**

- `sbatch <batch-file>` submitting a batch file
- Batch file is a shell script (`#!/bin/bash`) including Slurm options (`#SBATCH`) and computational tasks
- After job completion we will receive outputs (`slurm-jobid.out`)
**Slurm options**

### man srun - man sbatch

- **man srun**
  - `-p` --partition `<partition-name>`
  - `--mem <memory>`
  - `-n` --ntasks `<number of tasks>`
  - `-N` --nodes `<number-of-nodes>`
  - `-c` --cpus-per-task `<number-of-cpus>`
  - `-w` --nodelist `<list-of-node-names>`

- **man sbatch**
  - `--pty <software-name/path>`
  - `--gres <general-resources>`
  - `-t` --time `<days-hours:minutes>`
  - `-A` --account `<account>`
  - `-L` --licenses `<license>`
  - `-J` --job-name `<jobname>`

### Environmental Variables

- `$SLURM_JOB_ID`
- `$SLURM_JOB_NAME`
- `$SLURM_JOB_NODELIST`
- `$SLURM_CPUS_ON_NODE`
- `$SLURM_SUBMIT_HOST`
- `$SLURM_SUBMIT_DIR`

### Example

**test.py**

```python
#!/usr/bin/python3
import os

os.system("""""""
    echo hostname: $(hostname)
    echo number of processors: $(nproc)
    echo data: $(date)
    echo job id: $SLURM_JOB_ID
    echo submit dir: $SLURM_SUBMIT_DIR
""
)

print("Hello world")
```

**jobpy.sh**

```bash
#!/bin/bash

#SBATCH --partition Interactive
#SBATCH --mem 8G
python3 test.py
```

**srun**

```bash
srun -p Interactive -n 4 --mem 8G --pty bash
```

**sbatch**

```bash
sbatch jobpy.sh
```

**Output**

- hostname: lewis4-lenovo-hpc2-node282
- number of processors: 4
- job id: 21437062
- submit dir: /home/user/training
- Hello world
## Monitor jobs

```
sacct -X show your jobs in the last 24 hours
sacct -X -S <yyyy-mm-dd> show your jobs since a date
sacct -X -S <yyyy-mm-dd> -E <yyyy-mm-dd> -S <R/PD/F/CA/CG/CD> show running/pending/failed/cancelled/completing/completed jobs in a period
sacct -j <jobid> show info about the jobid
squeue -u <username> show a user jobs (R/PD/CD) in the queue
squeue -u <username> --start show estimation time to start pending jobs
scancel <jobid> cancel jobs
../jobstat.py <day/week/month/year> show info about running, pending and completed jobs of a user within a time period (default is a week)
```

### Completed jobs for the last week:

<table>
<thead>
<tr>
<th>JobID</th>
<th>User Account</th>
<th>State Partition</th>
<th>QOS NCPUS</th>
<th>NNode</th>
<th>RegMe</th>
<th>Submit</th>
<th>Reserved</th>
<th>Start</th>
<th>Elapsed</th>
<th>End</th>
<th>NodeList</th>
<th>JobName</th>
</tr>
</thead>
<tbody>
<tr>
<td>21564719</td>
<td>antec general</td>
<td>COMPLETED Interact+ normal</td>
<td>1</td>
<td>1</td>
<td>86n</td>
<td>2020-07-11T16:28:40</td>
<td>00:00:00</td>
<td>2020-07-11T16:28:40</td>
<td>00:12:35</td>
<td>2020-07-11T16:41:15</td>
<td>lewis-lenovo-hpc2-node282</td>
<td>bash</td>
</tr>
<tr>
<td>21564707</td>
<td>antec general</td>
<td>COMPLETED Interact+ normal</td>
<td>1</td>
<td>1</td>
<td>246n</td>
<td>2020-07-11T16:42:22</td>
<td>00:00:00</td>
<td>2020-07-11T16:42:22</td>
<td>00:16:13</td>
<td>2020-07-11T16:58:36</td>
<td>lewis-lenovo-hpc2-node282</td>
<td>bash</td>
</tr>
<tr>
<td>21564292</td>
<td>antec general</td>
<td>CANCELLED+ General normal</td>
<td>1</td>
<td>1</td>
<td>10c</td>
<td>2020-07-10T18:09:53</td>
<td>00:00:06</td>
<td>2020-07-10T18:09:59</td>
<td>00:00:00</td>
<td>2020-07-10T18:09:59</td>
<td>None assigned</td>
<td>bash</td>
</tr>
<tr>
<td>21564293</td>
<td>antec general</td>
<td>COMPLETED Lewis normal</td>
<td>1</td>
<td>1</td>
<td>10c</td>
<td>2020-07-10T18:18:19</td>
<td>00:00:00</td>
<td>2020-07-10T18:18:19</td>
<td>00:15:02</td>
<td>2020-07-10T18:33:21</td>
<td>lewis-r630-hpc4-node674</td>
<td>bash</td>
</tr>
<tr>
<td>21564559</td>
<td>antec general</td>
<td>COMPLETED Interact+ normal</td>
<td>1</td>
<td>1</td>
<td>86n</td>
<td>2020-07-11T16:00:07</td>
<td>00:00:00</td>
<td>2020-07-11T16:00:07</td>
<td>00:06:26</td>
<td>2020-07-11T16:12:32</td>
<td>lewis-lenovo-hpc2-node282</td>
<td>bash</td>
</tr>
<tr>
<td>21563692</td>
<td>antec general</td>
<td>CANCELLED+ Lewis normal</td>
<td>16</td>
<td>1</td>
<td>64n</td>
<td>2020-07-11T15:19:18</td>
<td>00:00:12</td>
<td>2020-07-11T15:19:22</td>
<td>00:00:00</td>
<td>2020-07-11T15:19:22</td>
<td>None assigned</td>
<td>bash</td>
</tr>
</tbody>
</table>
Monitor CPU and Memory

```
sacct -o User,Acc,AllocCPU,Elaps,CPUPer,TotalCPU,AveDiskRead,AveDiskWrite,ReqMem,MaxRSS -j <jobid> info about CPU and virtual memory for completed jobs
seff <jobid> show jobs CPU and memory efficiency
```

### Completed jobs

```
User  Account  AllocCPU  Elaps  CPUPer  TotalCPU  AveDiskRead  AveDiskWrite  ReqMem  MaxRSS
amtwc  general  16  00:48:39  12:58:24 01:49.774  66.88M  44.75M  64Gn  216K
```

```
Core Efficiency = CPU Utilization / Core - walltime
Core-walltime = Core per node * Elapsed time
```

### Running jobs:

```
sacct -j <jobid> -o User,Acc,AllocCPU,Elaps,CPUPer,TotalCPU,AveDiskRead,AveDiskWrite,ReqMem,MaxRSS info about CPU and virtual memory for running jobs (srun only)
srun --jobid <jobid> --pty /bin/bash attach to a srun/sbatch session and run `top` command to see information about processes
```

### Test

```
emacs -nw test.py
emacs -nw jobpy.sh
Control+x+s to save and Control+x+c to exit
sbatch jobpy.sh
seff <jobid> show jobs CPU and memory efficiency
```

```
resident set size (RES) = memory KB
```

```
#SBATCH -n 4
#SBATCH --mem 8G
```

We are using 0% CPU and less than 6MB memory
# Modules

**module avail/load/unload/list/show/purge**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module avail</td>
<td>available modules</td>
</tr>
<tr>
<td>module load</td>
<td>loaded modules</td>
</tr>
<tr>
<td>module show</td>
<td>show modules info</td>
</tr>
<tr>
<td>module unload</td>
<td>unload loaded modules</td>
</tr>
<tr>
<td>module list</td>
<td>list loaded modules</td>
</tr>
<tr>
<td>module purge</td>
<td>unload all loaded modules</td>
</tr>
</tbody>
</table>

For example let’s run R and MATLAB interactively:

**R**

```bash
srun -p Interactive --mem 4G --pty /bin/bash
module load R
module list
  1) R/R-3.3.3
R
```

**MATLAB**

```bash
srun -p Interactive --mem 4G -L matlab --pty /bin/bash
module load matlab
module list
  1) matlab/matlab-R2020a
matlab -nodisplay
```

**Never load modules in the login node.** It makes login node slow for all users. **Many modules don’t work in the login node.** We can load modules in batch files, for example:

**test.R**

```bash
#!/usr/bin/R

for (i in 1:3) {
  cat("Hello world", i,"\n")
}
```

**runr.sh**

```bash
#!/bin/bash

#SBATCH -p Interactive
#SBATCH --mem 4G

module load R
Rscript test.R
```

**sbatch**

```bash
sbatch runr.sh
```

**Output**

```
Hello world 1
Hello world 2
Hello world 3
```
What is next:

**Version control**

**Job dependencies**
- Slurm dependency option (--dependency)
  [https://slurm.schedmd.com/sbatch.html](https://slurm.schedmd.com/sbatch.html)
- Snakemake [https://snakemake.readthedocs.io/](https://snakemake.readthedocs.io/)

**Virtual Environments**

**Software Installation**

**Parallel programming**
- Scientific languages C/Fortran
  - OpenMP
  - OpenACC for GPU parallelization
  - MPI for massive parallelization
- Python
  - NumPy
  - Numba
  - mpi4py

**RCSS Documentation**
[http://docs.rnet.missouri.edu](http://docs.rnet.missouri.edu)

**XSEDE**
[https://www.xsede.org/for-users/training](https://www.xsede.org/for-users/training)

**Software Carpentry**
[https://software-carpentry.org/lessons/](https://software-carpentry.org/lessons/)

**HPC Carpentry**
[https://hpc-carpentry.github.io](https://hpc-carpentry.github.io)

**Data Carpentry**
[https://datacarpentry.org/lessons/](https://datacarpentry.org/lessons/)

**Cornell Virtual Workshop**
[https://cvw.cac.cornell.edu/topics](https://cvw.cac.cornell.edu/topics)

**Pittsburgh Supercomputing Center (PSC)**
[https://www.psc.edu/resources-for-users/training/](https://www.psc.edu/resources-for-users/training/)

**TACC Learning Portal**
[https://learn.tacc.utexas.edu/course/](https://learn.tacc.utexas.edu/course/)

**Feedback and Questions**
mudoitrcss@missouri.edu