

Texas Advanced Computing Center
Introduction to High Performance Computing for Life Scientists
Hands on Session
August 11, 2011

Goals: This hands on session is designed to walk a user step-by-step through basic, common activities on TACC systems: customizing shell parameters, copying data, editing files, and running jobs.

If you do not already have a login for TACC systems, go to the TACC User Portal and create an account: <https://portal.tacc.utexas.edu> Click the [New User?] link and follow the directions to get your TACC user name and setup your password.

Logging In

<u>Windows</u>	<u>Mac/Linux</u>
Open a secure shell client PuTTY is a great client for Windows: http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe “ Secure Shell Client ” is another great client for Windows. (Also good for doing GUI-style file transfers from your Windows system): http://sci.tamucc.edu/pub/ssh/SSHSecureShellClient-3.2.9.exe	Open Terminal (included in the OS)

Use your TACC user name when you ssh to Lonestar. Replace **<username>** with your user name:

```
% ssh <username>@lonestar.tacc.utexas.edu
```

Enter your password.

Changing Directories and Copying Files

Lonestar has three filesystems: /home, /work, and /scratch.

```
$ pwd
```

You always start in your home directory when you log in. It is saved in the variable \$HOME

```
$ echo $HOME
```

```
$ cdw
```

```
$ pwd
```

This is your work directory, also in the variable \$WORK

```
$ echo $WORK
```

```
$ cds
```

```
$ pwd
```

This is your scratch directory, also in the variable \$SCRATCH

```
$ echo $SCRATCH
```

```
$ cdh
```

Now copy the training files into your home directory:

```
$ cp ~jfonner/training811.tgz ./
```

```
$ ls
```

After entering the ls command, you should see a compressed “tarball” of all the training files in your home directory. Now extract the files using the tar command and look at what is there:

```
$ tar -xzvf training811.tgz
```

```
$ ls
```

The tarball contained a directory, training811, which you should now see in your directory

```
$ cd training811
```

```
$ ls
```

Notice that files and directories beginning with a “.” are not shown. Dot files are Linux’s version of hidden files or folders.

```
$ ls -a
```

Now we can see all files. We’ll look next at the files named .modules, .profile, and .cshrc

Customizing Your Shell

Try these shell commands at the prompt.

```
$ echo $SHELL
```

Displays the value of the login `$SHELL` environment variable. If you change shells during your session, the command “`echo $0`” (that’s a zero) will tell you your current shell.

```
$ chsh -l
```

Lists available shells. Try changing your shell, then change it back.

```
$ chsh -s <shell> <username> [use full path of <shell>]
```

example:

```
$ chsh -s /bin/bash jondoe
```

If you don’t know which shell you want to use, we recommend bash or tcsh.

Depending on your shell choice, follow one of the columns below:

Bash	TCSH
Copy the <code>.profile</code> and <code>.modules</code> files from the training directory to your home directory. <pre>\$ cp .profile ~/</pre> <pre>\$ cp .modules ~/</pre>	Copy the <code>.cshrc</code> and <code>.modules</code> files from the training directory to your home directory. <pre>\$ cp .cshrc ~/</pre> <pre>\$ cp .modules ~/</pre>
Now look at these files and customize them as desired:	Now look at these files and customize them as desired:
<pre>\$ vim ~/.profile</pre> <pre>\$ vim ~/.modules</pre>	<pre>\$ vim ~/.cshrc</pre> <pre>\$ vim ~/.modules</pre>
You can “comment out” any lines you do not want to run by putting a ‘#’ character at the beginning of the line.	You can “comment out” any lines you do not want to run by putting a ‘#’ character at the beginning of the line.

Submitting Jobs

We'll do an example job that only requires a few seconds of computing time. We will take a protein crystal structure from PDB, remove water molecules, add hydrogens, and minimize the energy. In the training811 folder, you should see the following files that we need:

1KAC.pdb - this is a protein structure taken from pdb.org
em.mdp - this is a molecular dynamics parameter file used by gromacs

Now we need to load the "gromacs" module. To find out how to load gromacs, type:

```
$ module spider gromacs
```

The first time you run this command, it will take some time to build a cache. After that, it will be fast. On Ranger, there are 2 versions of gromacs. If we want to load gromacs/4.5.3, we can get more information about it by typing:

```
$ module spider gromacs/4.5.3
```

This should display some information about gromacs, as well as a line that says: "This module can be loaded through the following modules:" In general, we need to load a compiler and a parallel message passing (MPI) environment. On Ranger, gromacs has been built under different combinations of compilers and MPI environments. We will use intel compiler and mvapich2 for this example. We need to make sure we have all of those modules loaded.

Now we need to see what modules we have already loaded:

```
$ module list
```

This shows you what is currently loaded

```
$ module avail
```

This shows you what you can load. You have to load all the "dependencies" before you can see a specific module. The "module spider" command helps you find specific modules if you know the name (or partial name) of the module you are looking for.

Lonestar should have intel and mvapich2 loaded by default, so all we need to do there is type:

```
$ module load gromacs
```

On Ranger, the pgi compiler and mvapich are loaded by default, so we need to type:

```
$ module swap pgi intel  
$ module swap mvapich mvapich2  
$ module load gromacs
```

Once you have the gromacs module loaded, we can prepare and submit our job. A couple of commands are very short and not processor intensive. We can run these on the head node directly. Make sure you are in the training811 folder and type:

```
$ grep -v HOH 1KAC.pdb > 1KACnoH2O.pdb
```

This command removes all the lines with "HOH" in them and prints the remaining lines into 1KAC-noH2O.pdb. In essence, we have remove all the water atoms from our file.

```
$ pdb2gmx -f 1KACnoH2O.pdb -o 1KACnoH2O.gro -water none -ff amber03
```

This command converts our pdb file into Gromacs format and will make several new files.
To see what is new in the directory, type:

```
$ ls -lrt
```

If you're not sure what the "-lrt" option does, you can get more information by typing "man ls" at the command prompt.

Now we can run another quick utility on the head node that puts our protein and parameters into a binary file that Gromacs will use to run molecular dynamics (or energy minimization in our case)

```
$ grompp -f em.mdp -c 1KACnoH2O.gro -o 1KACem.tpr | tee grompp.log
```

The "tee" command lets us see output and also put it in a file. We should now have 1KACem.tpr in our directory.

The last step is to make a job submission script and submit the job to TACC. The mdrun command is processor intensive and must not be run on a head node. A sample job submission script is in the training811 directory under the name "qrun." Let's edit the qrun file in vim or your favorite text editor.

```
$ vim qrun
```

Lines we need to pay attention to are the following:

```
#$ -pe 12way 12 - On lonestar, the second number needs to be a multiple of 12.  
On Ranger, it must be a multiple of 16 (e.g. 16way 16).
```

```
#$ -A 20110811BIO - This should be the name of an Allocation that you have. Your  
allocations are displayed right after you log in to Ranger or  
Lonestar. If you are not using the training allocation, change  
this to your allocation. If you only have 1 active allocation, you  
can omit this line.
```

Once you think your file is ready, submit the job by typing:

```
$ qsub qrun
```

"qsub" is the command that submits a job to our queue. The file that contains your script can have any name. "qrun" is just a personal choice.

If the job was submitted successfully, you can see its status by typing:

```
$ showq -u
```

Processing Files: grep and awk

This example was taken from the Linux PowerPoint presentation. This portion can only be performed on Lonestar, since that is where this 7GB file is located. (Note: some of these command wrap over to two lines in this document, but they should be entered as one command.)

```
$ grep "Score" /scratch/00944/jlockman/PTFA-assembly.fa_blastx.results | awk '$2>300 {print;}' | wc -l
```

The result should be: 5667564

```
$ grep "Medicago_truncatula" /scratch/00944/jlockman/PTFA-assembly.fa_blastx.results | grep -v ">" | wc -l
```

The result should be: 302151

```
$ grep "Medicago_truncatula" /scratch/00944/jlockman/PTFA-assembly.fa_blastx.results | grep -v ">" | awk '$2>300 {print;}' | wc -l
```

The result should be: 15233

Additional Exercise: redirecting output, more, less, cat

\$ cdh

With no arguments, it will take you back to your home directory.

\$ ls -l /etc > mylist

Use the **list** command and the **>** to redirect your output to a file named **mylist**

There are 3 methods for looking at this file from the command prompt: **cat, more, less**

\$ cat mylist

This will show the contents of the entire file in the terminal, and scroll automatically.

\$ more mylist

more will show the contents of the file, pausing when it fills the screen. Use the spacebar to advance 1 page at a time. [**q** to quit]

\$ less mylist

less will show the contents of the file, pausing when it fills the screen. Use the spacebar to advance 1 page at a time, or you can use the arrow keys to scroll one line at a time. [**q** to quit]

Try these commands here to become more familiar with redirection techniques.

\$ cat > lincoln.txt

Four score and seven years ago

^d [Control-D]

The three lines above set up a redirection of standard output (from the concatenation command) to enter (from standard input) the famous phrase from Lincoln's Gettysburg address into a file called "lincoln.txt". The input to the **cat** command is ended with a Control-D (**^d**). If you already have a file by this name, please choose another name.

```
$ cat >> lincoln.txt
our fathers brought forth on this continent a new nation
^d
```

These three lines append new text to the previously created file.

The next three commands demonstrate how to redirect input to a command (in this case, a script file that you create), that only reads from standard in.

```
$ cat > tryme.sh
#!/bin/sh
cat
^d [Control-D]
```

These four lines create a script file called "tryme.sh" that contains the **cat** command. The way the **cat** command is used here, (with no arguments), forces it to read from standard input.

```
$ chmod u+x tryme.sh
```

Don't forget this step when you create script files.

```
$ tryme.sh < lincoln.txt
```

This line redirects standard input to the script file, resulting in the text of the file "lincoln.txt" being sent to standard out. If you omit the redirection character (<), the script will try to read from standard in this is the only way this script will display the contents of a file.