

The Unix Command Line and You

Basic Interaction with the Unix Command-Line Interface

C. Zachary Palmer* and Ally Watrous

Official: Computational Astrochemistry Group @ Ole Miss

Unofficial: The Fortenberry Patch

E-mail: cpalmer5@go.olemiss.edu

1 Introduction

So, you've found yourself staring face-to-face with the unix command-line interface (CLI). Whether you are using the "Terminal" app on MacOS or were brave enough to install a distribution of Linux, you are now at the helm of one of the most unfriendly user-interfaces for beginners. Now this is something of a debate as some believe the (CLI) to be the most user-friendly since it allows you the most control over your work station. While I agree with that, it is not as intuitive as the default GUI that comes stock on a Windows or Mac. However, the use of the command line is essential for your ability to conduct computational research and will be beneficial for you to learn if you ever have to utilize the CLI again in the future. That is why I am taking it upon myself to introduce the most basic commands of the CLI to those who are new to the environment entirely. In this document, command line basics like movement through the folder structure of the computer and basic file manipulation will be addressed to initiate CLI novices into the world of the Unix command line. Additionally, the commands necessary to interact with the supercomputers from the Mississippi Center for Supercomputing Research will be explored.

2 Command Line Basics

2.1 Listing the folders and files on your computer with “ls”

When you first pull up a terminal window, you are greeted with a prompt that will vaguely resemble that of Fig. 1.



Figure 1: Command line prompt

The amount of information that is currently displayed on your terminal screen is impalpable. However, if you were to click on “Folder” or “Finder” on Windows or MacOS, respectively, you are immediately bombarded with all of the information you could possibly need. You see the most easily accessible folders and their files to boot. But once you open that terminal window, you get exactly squat. You may even ask yourself “uh.. is this thing broken?” The answer to that is usually, “no,” but you do have to actually interact with the command line in order to view the folders that you have on your computer. In order to actually view the folders, or “directories” as they are more commonly referred to and will be referred to from here on out, you have to type the letters “ls” into your prompt and press **Enter**. The return for this argument will be a list of all of the visible files and directories you have in your “home” area on your computer as can be seen in Fig. 2.

Utilizing this command is one of the few ways to view what is currently in the directory that you are in. In Fig 2, the extension “-color=auto” is a flag that can be added that will provide colors to the different types of “files” in your directory tree. In this case, the light blue color is for directories indicating that you can move into those directories to access what

```
zach@dogwood ~> ls --color=auto
auto/ chem/ class/ Downloads/ go/ Library/ packages/ Pictures/ programs/ Reviews/ roam/ sandbox/ scripts/
zach@dogwood ~> █
```

Figure 2: The output of the “ls” command

is in them. The “ls” command has many more flags that can be utilized in order to display hidden files, or further information such as when a file was last saved. If you would like to learn more about the “ls” command you can type “man ls” into your prompt and it will bring up the manual page of the “ls” command. Most distributions of Linux, and Windows and MacOS, will come with the “manpage” package that allows you to learn more about these commands. The man page for “ls” can be traversed using the arrows keys and can be exited by pressing “q.”

2.2 Directory Manipulation with “cd” and “mkdir”

Now that you know how to list the directories and files in your area it may be beneficial to access them. As many people know, in a GUI you can easily use the cursor to click on the folders and files in order to access them. This is not so in the CLI, however, as you do not have a cursor to access your directories. So in order to enter the directories that have been listed by the “ls” command, you must type the letters “cd” followed by the name of the directory as can be seen in Fig. 3.

```
zach@dogwood ~> ls --color=auto
auto/ chem/ class/ Downloads/ go/ Library/ packages/ Pictures/ programs/ Reviews/ roam/ sandbox/ scripts/
zach@dogwood ~> cd Downloads/
zach@dogwood ~/Downloads> █
```

Figure 3: The output of the “cd” command

One of the major things to note in Fig. 3 is that the prompt has now changed to reflect the change into a new directory. As seen, the prompt before changing into the “Downloads” directory did not include it in the prompt, but it does after you are in the directory.¹ Now that you are in the “Downloads” directory, you can use “ls” to list all of the files that are in there. This is a great way of pairing the two commands to enter a directory and see what files are present. The “cd” command can be extended and doesn’t have to be used one directory at a time. If you want to access a directory that is within a directory, you can extend the path with a “/” character and entering the name of the next directory you wish to go into. Now it may seem a bit eccentric to memorize the directory structure of your entire computer, so its a good thing that you don’t need to. When you are using many commands on the command line, there is a feature present that allows you to auto-complete a command, directory name, or file name. When you are using the “cd” command, you can press the **Tab** key twice and it will list all of the directories within a directory that you are currently in as seen in Fig. 4. Utilizing this feature is a quicker way of viewing the directories in a directory for you to easily move between the directory structure of your computer.

```
zach@dogwood ~-> cd Downloads/
..ownloads/acs_template/  ..ownloads/elsarticle/  ..ownloads/Sulfur_Info/  ..ownloads/zach/
..ownloads/auto/         ..ownloads/overleaf/    ..ownloads/Useful_Docs/
```

Figure 4: Using the auto-complete feature with “cd”

Now that you know how to use “cd” to change into directories, how do you get back to the previous directory? The name of the previous directory isn’t present in the name of the directory you are currently in, so you cannot “cd” into it like you did to get there. Instead, you have to utilize “cd ..” in order to move back to the previous directory. Much like if you changed through multiple directories in one go using “cd dir1/dir2/dir3/,” you can change back out of each of these directories with the same number of “..” with the command “cd

¹The inclusion of the directory in the prompt is not the same across terminals, however, as some prompts do not include the current directory or directory path in them. The prompts can be edited to include them, but that is not within the scope of the topic at hand.

../../../" this will put you three directories back. An example of this can be seen in Fig. 5.²

```
zach@dogwood ~> ls --color=auto
auto/ chem/ class/ Downloads/ go/ Library/ packages/ Pictures/ programs/ Reviews/ roam/ sandbox/ scripts/
zach@dogwood ~> cd Downloads/
zach@dogwood ~/Downloads> cd ../
zach@dogwood ~> █
```

Figure 5: Going back a directory using “cd ../”

Just as important as changing between directories, is the ability to create new directories for easy file organization. As seen in Fig. 6, you can make a new directory with the “mkdir” command with really just about any name.³ Once you have made a directory, it will show up when you use “ls” and you can now “cd” into it. This command is one of the most essentially organization tools that you can learn as it allows you to neat organize your directory tree as you see fit. Now that you have created your first directory, it’s time to put some stuff in it.

```
zach@dogwood ~> ls --color=auto
auto/ chem/ class/ Downloads/ go/ Library/ packages/ Pictures/ programs/ Reviews/ roam/ sandbox/ scripts/
zach@dogwood ~> cd Downloads/
zach@dogwood ~/Downloads> cd ../
zach@dogwood ~> mkdir dir1
zach@dogwood ~> ls --color=auto
auto/ class/ Downloads/ Library/ Pictures/ Reviews/ sandbox/
chem/ dir1/ go/ packages/ programs/ roam/ scripts/
zach@dogwood ~> cd dir1/
zach@dogwood ~/dir1> █
```

Figure 6: Creating the “dir1” directory and “cd”’ing into it

2.3 File interaction with “touch” and the almighty “Vim” editor

Oh, what to do with this new empty directory that we have created called “dir1?” What good is a directory if you can not put anything in it? Luckily, that is not a problem in the terminal! Now in a Windows or Mac environment you’d probably open something like Microsoft Word or some other Mac equivalent in order to create a file to be stored in a

²You can easily get back to your home directory by not giving “cd” any file path.

³Try not to put spaces in the directory name. If you are going to have a “dual name” directory, try using an underscore like “dir_1.” Putting spaces in a mkdir argument will create two directories.

folder. Once again, this is not so for the CLI. Instead, there are programs that usually come with most distributions of Linux or in the MacOS Terminal app that act as “text editors” rather than “word processors” like Microsoft Word. In the CLI, there are a few common text editors, but the one that we will be working with in this tutorial is the “Vim” text editor. “Vim” is a very versatile text editor that allows the user a lot of neat features that help to streamline the text editing process, but I we are getting ahead of ourselves. First we actually need to make a document to edit. As seen in Fig. 7, the easiest way to create a file without actually opening one is using the “touch” command. The return value for this command is the creation of an empty file in the current directory you’re working in.⁴

```
zach@dogwood ~/dir1> touch file1.txt
zach@dogwood ~/dir1> ls --color=auto
file1.txt
zach@dogwood ~/dir1> █
```

Figure 7: Creating the “file1.txt” file and using “ls” to see it in the current directory

2.3.1 Using “Vim”

Creating an empty file seems a bit useless, and in most cases it is, so let’s open the file and actually store something in it. In order to open the new file that we have just created, type the command “vi” and if you hit the **f** + **Tab** keys, it will auto-complete on your prompt to “vi file1.txt.”⁵ The steps above can be seen in Fig. 8.

```
zach@dogwood ~/dir1> touch file1.txt
zach@dogwood ~/dir1> ls --color=auto
file1.txt
zach@dogwood ~/dir1> vim file1.txt █
```

Figure 8: Using “vi” to open the selected “file1.txt”

⁴If you give a file path as an argument to the “touch” command it will create the empty file at that location instead.

⁵This auto-completion will only work if you do not have any other files in your directory that start with “f,” if you do hitting **Tab** twice will show all of the files that start with “f” and you can select the one you want that way.

Once you have opened the selected file, you will be greeted with... well.. nothing! It's an empty file. It's rather unimpressive with its vast emptiness. Something that you will immediately notice is your inability to type. "Vim" is not like Word with it's ability to immediately begin typing. Instead, "Vim" has its own key bindings and commands that must be learned. Most of the key bindings in "Vim" will change the "mode" that you are in. When you first open a file in "Vim", you are in what is considered "normal mode." In this mode you cannot type any text and can only move through the file or interact in other ways. If you want to be able to type in the file, you must hit the **i** key in order to enter "insert" mode. In Fig. 9, you can see how the "Vim" window changes in this mode.



Figure 9: Insert mode in "Vim"

Once in "insert" mode, you can type all of the text you want and can put just about anything. You are now essentially in a Note pad-esque environment. So have fun and type just about anything to your hearts content. When you are finished typing the best note to yourself or another person, you probably want to save the document but don't see a floppy disk icon in the top left corner to save your work. Uh oh... Was this all for not? Are we hopeless type words that we can't keep? Of course not! When you are finished editing your document press the **Esc** key. This will place you back into normal mode and will allow you to interact with what is essentially the "Vim" menu. To enter the "Vim" menu, hit the **Shift**

and ; keys to get `:`.⁶ This is a very powerful menu. It allows you to do many cool things in your file, but most importantly it allows you to save! The key combination `:w` saves all of the changes you have made to the document since you either opened, or since the last time you saved to the file. It's generally good practice you save your file often so you don't lose any progress due to any stray extraneous causes.

Now that you've saved your file, let's learn how to traverse the document and while in normal mode. Let's say that you need something important that's all the way at the bottom of a long file. You don't want to smash the down arrow key a million times! So instead, use **Shift** + **g** to easily get from anywhere in the file to last line! Pretty cool, huh? Conversely, if you wanted to get to the top of the file from anywhere in the document just hit **gg**, and no I didn't accidentally type **g** twice. These are very easy ways of getting to the two extremities of the file. Now let's say you just want move forward through the file and not to the end of the entire file. You can hit the **Ctrl** + **f** keys to move through an entire page at a time. If you hold **Ctrl** and mash **f**, you can move through as many time as you press it. Holding both down will move you through the document very fast. Once again the converse is also true and this same scheme works to move backwards through the file one page at a time if you use **Ctrl** + **b**, instead. If you think that **Ctrl** + **f** and **Ctrl** + **b** is too extreme and a bit jarring, you can move by a half page following the same scheme as before using **Ctrl** + **u** and **Ctrl** + **d** to move backward and forward, respectively.

So now that you've learned how to traverse through a document, let's try manipulating the text in a few different ways. The first way is copying and pasting in normal mode in "Vim!" If you want to copy one line from a file, you can type **y** in order to "yank" a line into your clipboard. In order to paste that yanked line, you can type **p** for "paste." Gone are the days of **Ctrl** + **c** and **Ctrl** + **v**! The new copy and paste kings are here! Now, what if you wanted to yank more than just one line into your clipboard? Luckily, "Vim" has the useful tools to select multiple lines at once in order to do so! In normal mode, you can hit

⁶From here on out this will just be denoted as the `:` key.

“Vim” editor. After editing your file and you decide you want to add a new line to your document, instead of going into insert mode and hitting **Enter**, you can just press the **o** key while in normal mode and it will open a new line below your cursors position and will automatically put you into insert mode! If, however, you’d rather open a line above where your cursor is, press **Shift** + **o**. Finally, the ability to search through a document will be essential for your success in your research. In order to search from the top of the file down, you can press **/** while in normal mode and it will prompt you for a phrase to search for in the bottom left-hand corner. An example of this can be seen in Fig. 12

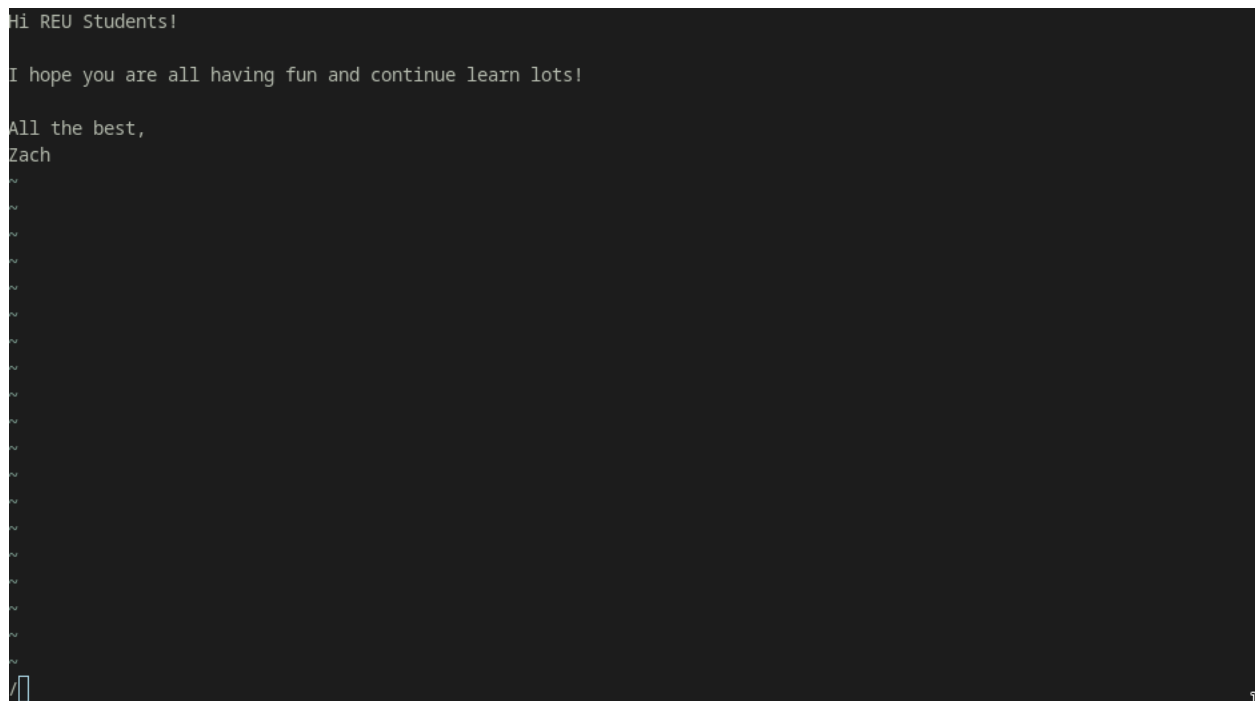


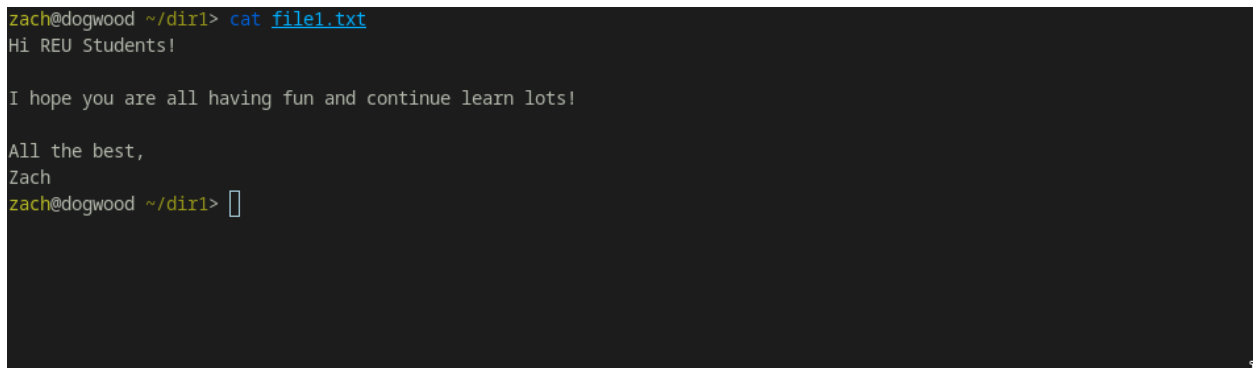
Figure 12: The search bar in “Vim”

When you press **Enter**, it will take you to the first iteration of the searched phrase that comes after your cursor. In order to move to the next searched phrase, simply press the **n** key. Conversely, to search for the last iteration before your cursor, pressing **Shift** + **n** should do the trick! What if you’re at the bottom of a file and want to search from the bottom up? Luckily, the **?** key will do just that! While in normal mode, you can press that key and it do a “reverse” search that can be cycled through using the **n** and **Shift** + **n** keys!

So, the basic tutorial of “Vim” is swiftly coming to a close. With that being said, how are you supposed to exit a file once you are finished with it? After saving your file and all of its sweet, sweet contents, what do we do now? Remember our pal the `:` menu? When you are finished editing a document and its time to quit, you can use `:q` in order to easily close a file that has been just been written too. Now you can actually chain these together to save and quit a file at the same time with `:wq!` Alternatively, if you did a lot of manipulation to a file and don’t want to save the changes, but want to quit, you have to use the `:q!` command to “really quit” the file. If you don’t it will tell you that you have unsaved changes and it will refuse to close the file. Thanks, “Vim!” Once you have exited a file, you will be in the same directory that you were in when you first opened your file.

2.4 File viewing with “cat” and “less”

Now that you have finished writing to your files, you want to view the contents but don’t want to open it in “Vim” again. How can we view the file without “Vim,” though? Well there are two common tools that CLI can provide us. First, the simple “cat” program will concatenate your file and present it to you on the screen as can be seen in Fig. 13.

A terminal window with a black background and light-colored text. The prompt is 'zach@dogwood ~/dir1>'. The user enters 'cat file1.txt'. The output is: 'Hi REU Students!', 'I hope you are all having fun and continue learn lots!', 'All the best,', 'Zach', and 'zach@dogwood ~/dir1>'.

```
zach@dogwood ~/dir1> cat file1.txt
Hi REU Students!

I hope you are all having fun and continue learn lots!

All the best,
Zach
zach@dogwood ~/dir1> █
```

Figure 13: The output of “cat” on file1.txt

As you can see, the contents of the file have been printed onto your terminal for you to read! That is very handy for short files to see what their contents are. What if you had a file that contained lots of contents? Well for all intents and purposes, “cat” would print

the whole file and you would only see the last page of it printed to your screen. So in order to combat this, you can use the “less” command. “Less” allows you to view the contents of a really long document, gives you the ability to scroll through with the arrow keys and to search with the search keys, / and ?, but doesn’t allow you to manipulate any text. If you used the man page from earlier to look at “ls” or any of the other commands, you may notice that “less” works a lot like that. “Man” uses “less” to print its pages to the screen for you to read. Thus, when you are done viewing something in less, you can press **q** and it will exit the document window. This can be especially useful if you are viewing the contents of a calculations output file that is still running. If you accidentally save and quit one of these files, its curtains for ye olde calculation.

2.5 File and Directory Manipulation with “cp,” “mv,” and “rm”

Let’s say you want to copy a file or directory from one directory to another one, but all you have is this silly CLI that you can’t drag things around in. The way to copy these files or directory is with the “cp” command. The “cp” command works by copying a selected file from one directory and pasting it to a new location, as can be seen in Fig. 14.

```
zach@dogwood ~/dir1> ls --color=auto
dir2/ file1.txt
zach@dogwood ~/dir1> cp file1.txt dir2/
zach@dogwood ~/dir1> cd dir2/
zach@dogwood ~/d/dir2> ls --color=auto
file1.txt
zach@dogwood ~/d/dir2> █
```

Figure 14: The copied file1.txt from dir1 to dir2

Notice how I’ve strung together the different commands we have learned so far. I used “ls” to list the files in dir1, I used “cp” on file1.txt and told it to copy to dir2, then I used “cd” to get dir2, and I used “ls” to view the contents of that directory to make sure the file made it in there. This is a common even that happens when you start learning the CLI. Be

sure to always check these things to ensure you aren't making any mistakes that may come to bite you later. Now back to “cp,” this can also work the backwards way. What if I was already in dir2 and wanted to copy file1.txt into the current directory? Well, just give “cp” the file path to the file and what you want it to be called as seen in Fig. 15.

```
zach@dogwood ~/d/dir2> ls --color=auto
zach@dogwood ~/d/dir2> cp ../file1.txt .
zach@dogwood ~/d/dir2> ls --color=auto
file1.txt
zach@dogwood ~/d/dir2> █
```

Figure 15: The copied file1.txt from dir1 to dir2 from within dir2

I provided “cp” with the pathway to the previous directory with the “../” portion, but then followed the “/” with the name of the file that I wanted. Something else you may notice is that instead of giving it a name after the file path I just gave it “.” This just means that I want to copy this file to the current the directory but to keep the current name of the file. Of course, if you wanted to make a full copy of a file but change the name of it you could have given it a new name, but for the purposes of training we’ll keep it the same. This same exact usage can be done on whole directories even. If you want to copy full directories you can do so! However, in order to do that you need to give the “cp” command a “flag” that will allow it to treat all contents in the directory. In order to copy full directories, you must use “cp -r.” the “-r” flag means that it will recursively copy the directory and all of the files inside of the directory into the directory you want it to be copied to. Without this flag, an attempt at copying a directory will result in an error saying that it is omitting the directory for being copied.

Now, what if instead of copying the file or directory, and I wanted to move it to another directory entirely? You would follow almost the exact same steps before hand as you did with “cp,” but now you would use the “mv” command instead. This will move the file or directory from one directory to another as long as you give it a file path and the name of the file or directory. However, this can also just be used if you want to change the name of

a file or directory and not make a copy. Just provide the “mv” command with the file or directory you want to change the name of and the name you want it to change to. When working with directories, you can move entire directory structures to other directories and do not have to provide the “-r” flag like you would for the “cp” command.

Now for the scary part... Let’s say you want to remove a file or directory from your computer. There is an easy command for that that will work on most files and directories. The “rm” command gives you the ability to remove said files and directories. Just provide “rm” with the file path for a file or directory and it will remove it from your computer. Now like with “cp” in order for this command to remove a directory, you have to provide it with the “-r” flag.⁸ One word of warning, the CLI does not have a “recycle bin” when you remove something from your computer it is gone for good. Gone for good. If you are removing a file from your computer, you better be 100% sure you know what you’re removing. Sometimes, it may feel nice to do some like spring cleaning and remove any old files you may have lying around... Just be careful please.

Alright... With that out of the way, I can safely say that I shared the most basic commands I could come up with you in this section. The things that you have learned may seem like a lot and I covered a large swath of topics, but just know that there is plenty more tips and tricks that can be learned elsewhere if you are interested. With just the commands I have taught you, you know about 90% of the commands I use to get my research done. If you are interested in learning more I would suggest watching YouTube videos or using your preferred search engine to ask specific questions.

3 Interacting with the MCSR supercomputers

Hopefully by this point you are comfortable enough with the CLI basics that you won’t have much problem getting acclimated to the supercomputers. This section may also seem rela-

⁸If you know that a directory is empty and you want to remove it you can use “rmdir” and it won’t need to be used with any flags

tively short, yet it will be vitally important for your success with the MCSR supercomputers. First things first, how do you actually get on the supercomputer? Hopefully, you have your “hpcwoods” account ready to go and can easily follow these commands. In order to actually log into the “hpcwoods” supercomputer, type “ssh r#####@hpcwoods.olemiss.edu,” where the ##### are from the email you received from assist@mcsr.edu, and the return of this will ask if you trust the network.⁹ Just say yes and it will take you through to a password prompt. Here you can start typing your password, but don’t freak out when it looks like nothing is happening as it won’t show that you are typing anything, but I promise that it is working! Once you are in, you will see a prompt... Imagine that! Now you can interact as you normally would with the CLI. This area that you have logged into is the “hpcwoods” supercomputer and is the hub point for three other supercomputer known as “maple, sequoia, and catalpa.” More than likely you will be using maple, so to log into maple from here type “ssh r#####@maple.” Once you login you will be ready to run jobs in the queue.

Submitting calculations is the most important thing you will be doing for computational research. Without this crucial, yet what seems trivial, step you cannot do anything. So how does one submit a calculation? Well that depends on what program it is you are using. The people in charge of training you to do your research should have all of the sample files that you will need in order to submit a job to the queue that will interact with the right programs. In order to submit a job to the queue, however, you must be in a directory that contains a “.pbs” file extension. In your prompt you’ll want to type “qsub opt.pbs.” This will submit your job to the queue and print your job number to the screen. When there are enough resources to allocate your calculation, it will run the job using the program you specify in your “.pbs” file. If for any reason you must kill a job, you can use the command “qdel #” in order to kill that specific job. A sample “.pbs” file for running a MOLPRO calculation can be seen in Fig. 16.

There is a lot of pertinent information that can be changed and updated in this file to

⁹This is only the case the first time. Anytime after this you will no longer see this message unless you log in from a new computer.


```
#!/bin/sh
#PBS -N hcn
#PBS -S /bin/bash
#PBS -j oe
#PBS -W umask=022
#PBS -l ncpus=4
#PBS -l mem=1gb

module load pbspro molpro

export WORKDIR=$PBS_O_WORKDIR
export TMPDIR=/tmp/$USER/$PBS_JOBID
mkdir -p $TMPDIR

cd $WORKDIR

date
hostname
molpro -t 4 hoof.F12-TZ.com
qstat -f $PBS_JOBID
date

rm -rf $TMPDIR
```

Figure 16: Sample Molpro file

suit the needs of the jobs you are running. The “#PBS -N hcn” line can be edited to be any name other than “hcn.” The “#PBS -l ncpus=4” can be used to change the numbers of cpus used for your job. Lastly, the “#PBS -l mem=1gb” line can be edited to increase or decrease the amount of memory that your jobs have access to while they’re running. You must be careful with how many resource you request as this will sharply increase the amount of time your job sits in the queue before actually running. Additionally, you must change the name of the job to match whatever the input file is that you will be running on the “molpro -t 4 hoof.F12-TZ.com” line. The “hoof.F12-TZ.com” can be changed to whatever your input file name is. If you keep it something neutral, like “opt.com” you will never need to change the “.pbs” file between jobs.

When you have submitted your job with the “qsub” command from before, you can check on the progress of your job by typing “qstat -u r#### -r” to see what jobs you have in the queue that are actually running. This will be helpful in determining what has completed, or died..., overnight if you check it first thing in the morning for example. Additionally, “removing the “-r” flag will show you all of the job that you have waiting in the queue. This

one is useful in seeing how much more you have to wait before you get to the good stuff of data analysis! Those are really the most useful commands for the supercomputer and will be the most beneficial for conducting your computational research.

4 Conclusion

Welp, my job here is done. You have been shown how to interact with one of the most unfriendly interfaces for beginners and came out on top! No longer are you beginners and novices! Nay, for you have been initiated, inducted into the society of CLI apprentices! With your new found skills, traverse the CLI with confidence in knowing that you learned an important life skill for your time as a computational researcher. Of course there will always be more advanced skills to learn when it comes to the CLI and even “Vim.” Perhaps in a future document those exalted words will make an appearance, but ye verily I say unto thee: thou needst it not for thine success. Anyways, I hope this was helpful and I hope that it will make your experiences as a computational researcher that more enjoyable and efficient. Now my young grass-hoppers... FLY!¹⁰

“One might forget that panda bears are still bears.” - Zach Palmer

¹⁰You fools!

Table 1: Table of Important Commands

Command	Purpose	Usage
ls	Lists contents in a directory	ls
cd	Changes into the specified directory	cd dir1
mkdir	Makes a new directory with a given name	mkdir dir1
touch	Creates a new, empty file with a specified name	touch file1.txt
vi	Opens a given file in the “Vim” editor	vi file1.txt
cat	Prints the contents of a file to the terminal	cat file1.txt
less	Puts the contents of a file on the terminal window and give you more control	less file1.txt
cp	Copies a given file or directory to a new location	cp file1.txt dir2/ or cp ../file1.txt .
mv	Moves a given file or directory to a new location	mv file1.txt dir2/ mv ../file1.txt .
	Also allows name changes of files or directories	mv file1.txt file2.txt
rm	Permanently deletes a file or directory	rm file1.txt
rmdir	Removes an <i>EMPTY</i> directory from the computer	rmdir dir1/
ssh	Connect to the supercomputer	ssh r#####@hpcwoods.olemiss.edu
qsub	Submits a .pbs script to the queue	qsub opt.pbs
qdel	Kills the job specified	qdel #
qstat	Allows you to see what is in your queue	qstat -u r#####

Table 2: Table of Important Vim Commands

Command	Purpose
i	Enter insert mode
Esc	Exits a given mode back to normal mode
o	Open a new line in insert mode
Shift + O	Opens a new line above the cursor in insert mode
y	Copy current line
p	Paste copied line
x	Deletes character
dd	Deletes current line
Ctrl + V	Enters visual block mode
Shift + V	Enters visual line mode
gg	Moves cursor to the top of the file
Shift + G	Moves cursor to the bottom of the file
Ctrl + F	Moves cursor up one page
Ctrl + B	Moves cursor down one page
Ctrl + U	Moves cursor up half a page
Ctrl + D	Moves cursor down half a page
Shift + H	Moves cursor to the top of the current page
Shift + M	Moves cursor to the middle of the current page
Shift + L	Moves cursor to the bottom of the current page
u	Undo
Ctrl + r	Redo
/	Searches from top/down
?	Searches from down/up
n	Moves to next searched phrase
Shift + n	Moves to the previous searched phrase