BASIC LINUX TERMINAL COMMANDS

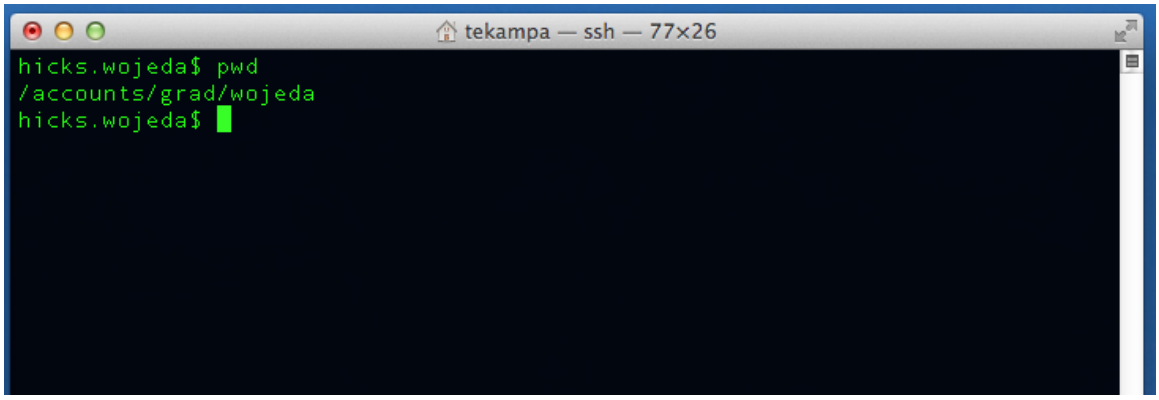
This little handout is just a quick introduction to basic commands in the Terminal that should get you going to use it on your own for your projects.

Why should I learn how to navigate the terminal?

- In empirical work, you're bound to run into a problem where your computer will not be able to handle a process. Let the server deal with it (or even multiple ones with parallel processing – see Daniel's presentation last year).
- Running different processes at the same time could also bog your personal computer down. You can send all of them to the server and not worry about it.
- Projects can get really tiring with all that pointing and clicking especially when it involves multiple programs. You can write a shell script that does all that for you and keep things organized.
- Here and at most other places you'll end up at, the most recent software and other goodies will be on the server.
- One more thing to write on your CV …
- Probably many more reasons so just try to at least be familiar with it.

pwd

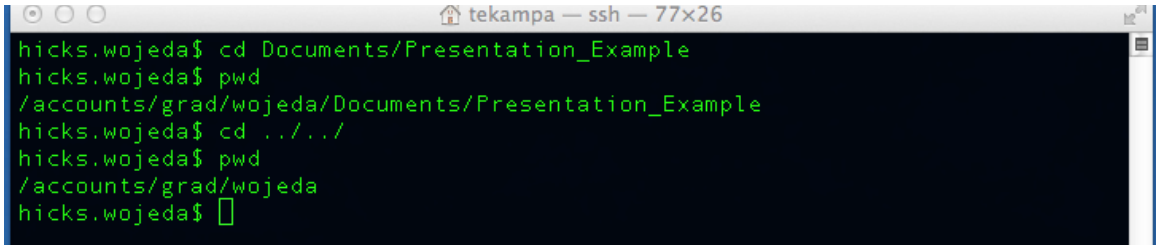
pwd stands for present working directory. Once you log-in to the server you should be automatically located in your home eml directory.



cd

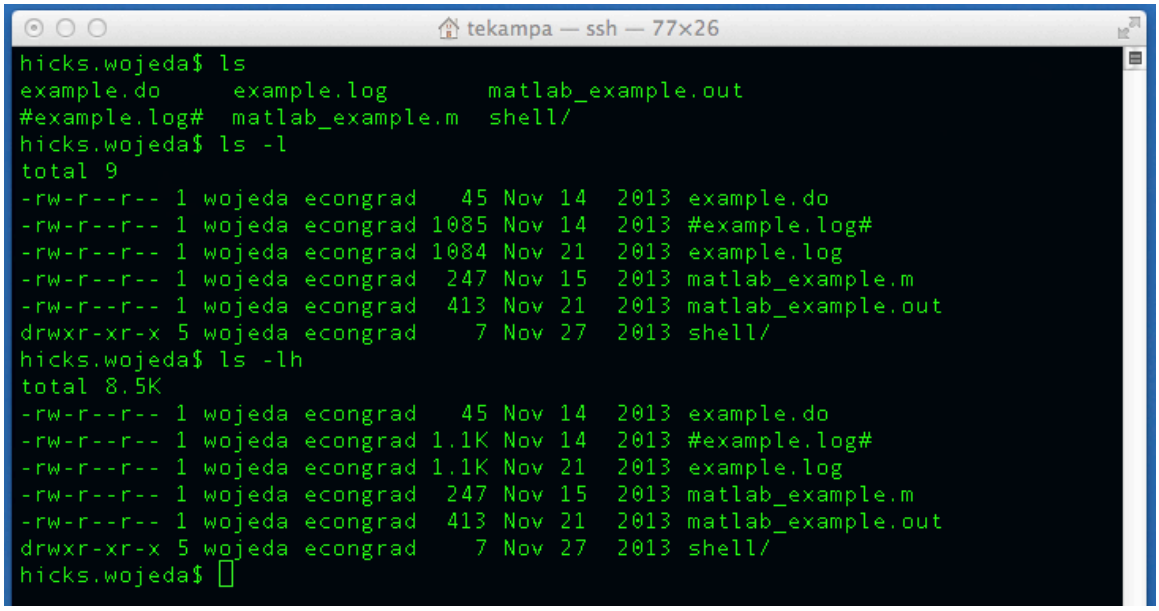
cd stands for change directory so you can start moving around directories. The syntax follows:
cd directory_path

You also can use **../** as part of the directory path to move up a directory.



ls

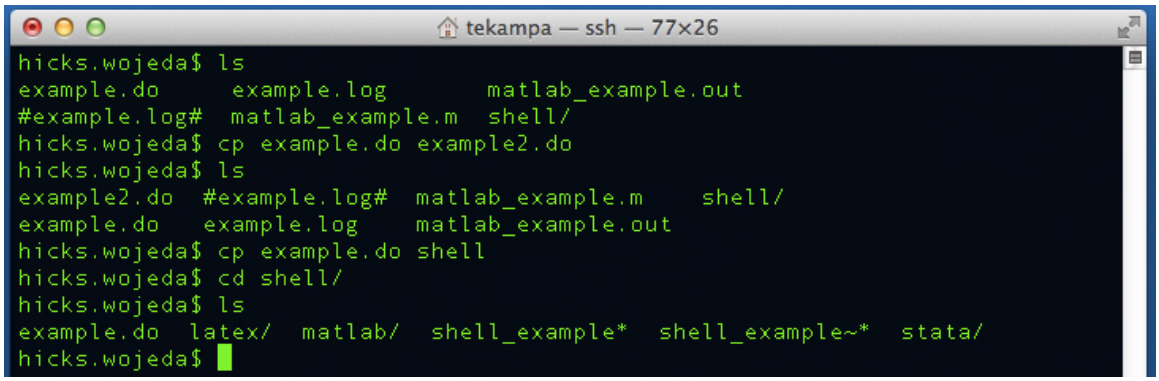
ls will list out what's under the directory your in. You can use the options **-l** and **-lh** to get more information. The first one will tell you the size of files in bytes while the second one gives it to you in human readable format. If it had a **d** in front of it then this means it's a directory.



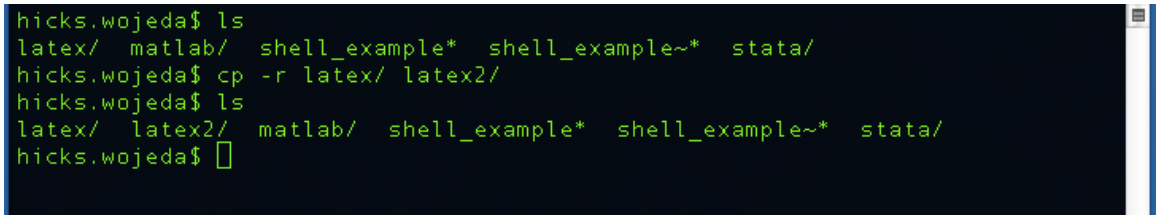
cp

cp will allow you to copy files following the syntax:
cp filename new_filename directory_path

If directory path is empty then it will copy the file to the present working directory. You don't have to write a new filename if you're coping the file another directory.

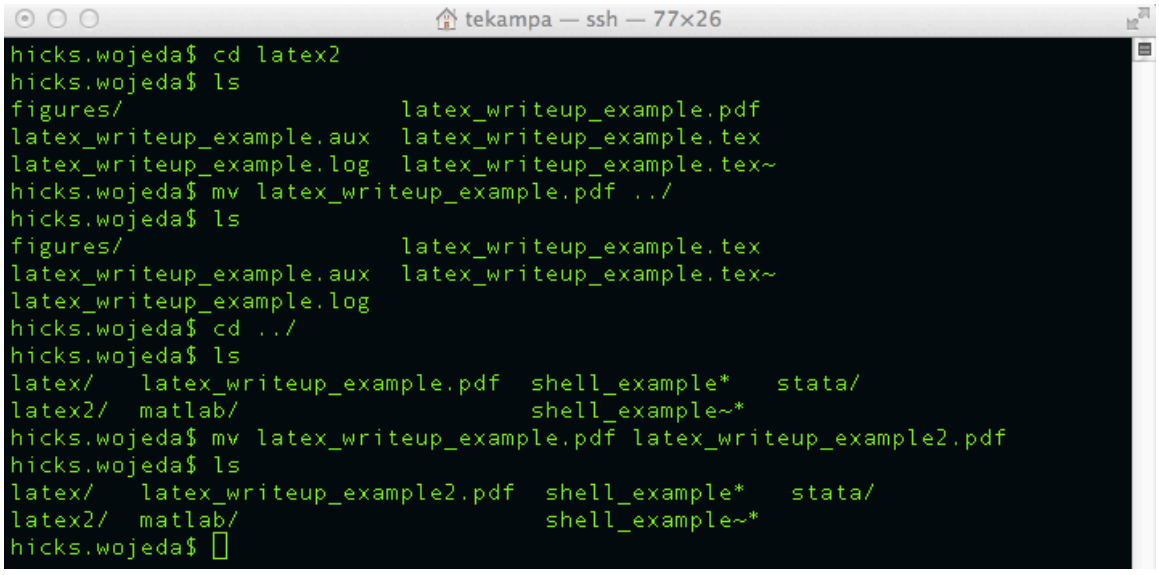


To copy a whole directory you would follow the syntax (the **-r** here stands for recursively):
cp -r directory_path new_directory_path



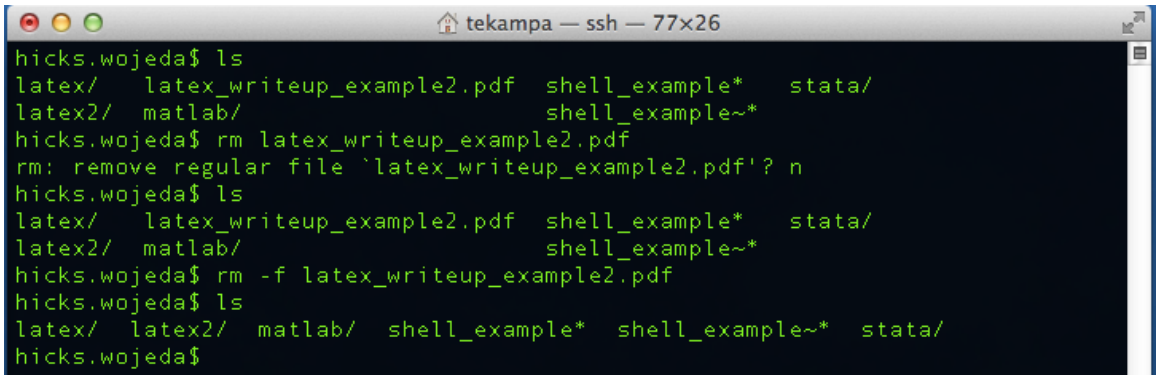
mv

mv is the command to move files and also works to rename files when needed. Follows the syntax:
mv filename directory_path
or to change filename: ***mv filename new_filename***

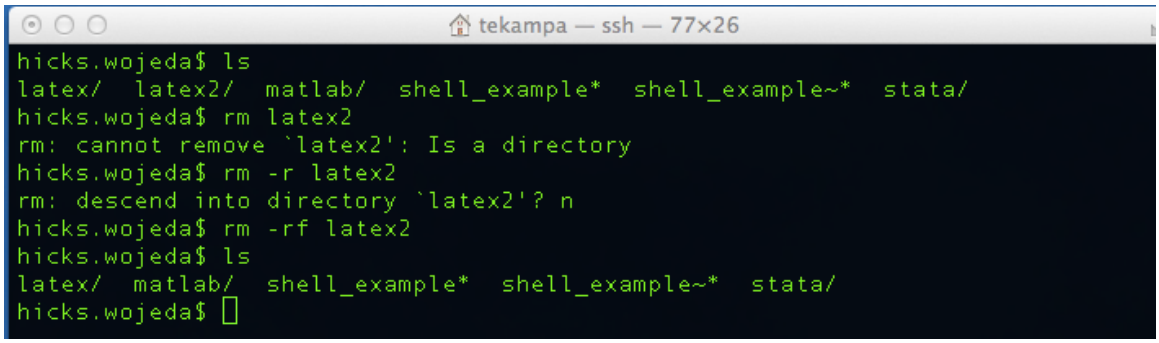


rm

rm will allow us to remove things so let's delete a few things that we created following the syntax:
rm filename
and you can add the option **-f** so it doesn't ask you that buggy question if you want to delete.

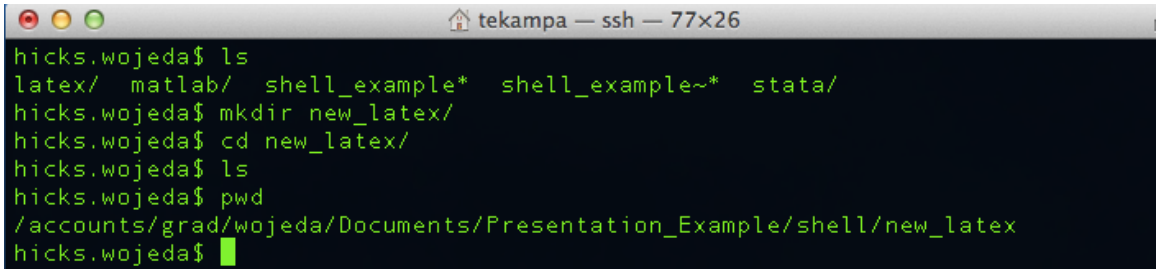


To remove a directory you would add the option **-r** so it deletes recursively within the folder and the folder itself.



mkdir

mkdir allows us to make a new directory.



*

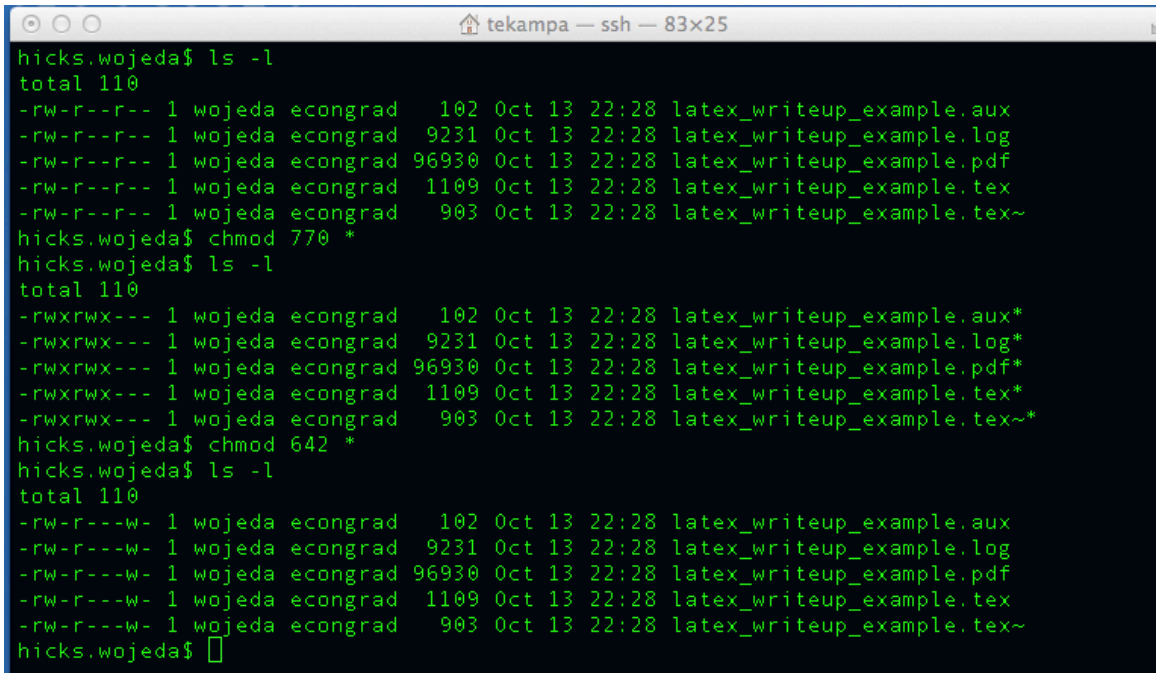
The asterisk is not really a command but counts as the wildcard character in the terminal. Below are a couple of examples of how to use it. It can help list files that end with a certain extension or move an entire set of files. Basically you're telling the terminal to fill in the command for any names permissible substituting anything for the asterisk.



BE VERY CAREFUL REMOVING THINGS WITH *. YOU COULD POTENTIALLY REMOVE A LOT OF THINGS. The good news is that the EML servers are backed up quite frequently so stuff can be restored but let's try to avoid that.

chmod

chmod is used to change user access permissions. If you notice the first dashes in the front when you type in the **ls -l** command it will tell if you have r (reading), w (writing) and x (executing) permissions. The first three letters tell you what permissions you have, the second set of the three letters tells you what permission your group has and the last is overall permission access. You'll sometimes have to edit these permissions to access your files or when writing a shell or python script.
The syntax is:
chmod nnn filename
where nnn stands for three numbers each assigning permissions to the three levels mentioned above. 4 = read, 2 = write and 1 = execute. If you want to give yourself reading, writing and executing permissions then the first n will be 7 = 4 + 2 + 1. For not giving anyone else any permissions then the syntax would be: ***chmod 700 filename***

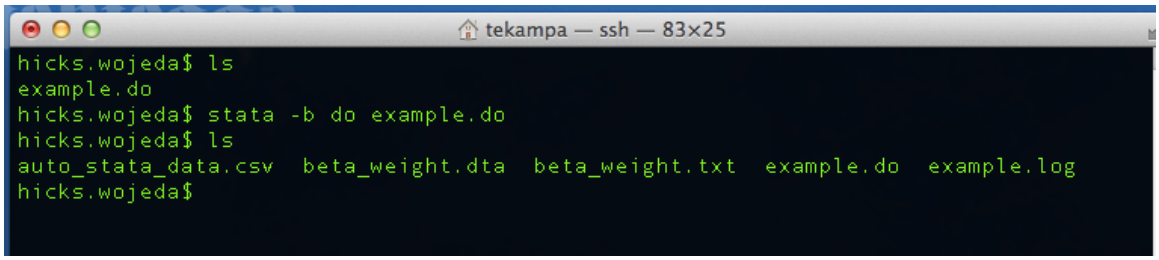


Opening UI Stata

You can open Stata from any directory you are in by typing: **xstata &**
Notice that your starting directory once you open Stata is where you were located when you typed in the command. As an important aside, the **&** tells the terminal to run Stata in the background so you can keep typing on the terminal command line.

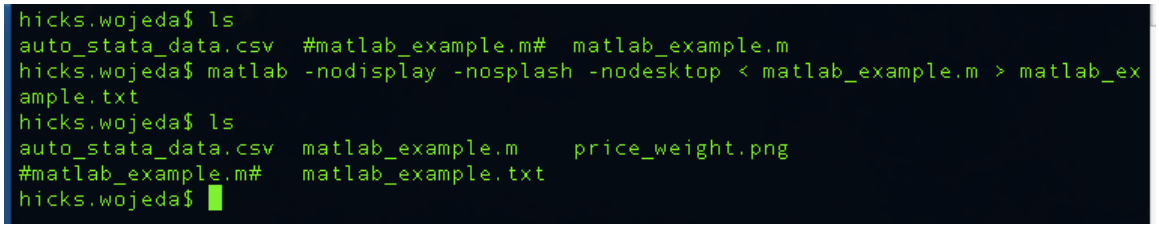
Opening UI Matlab
Similarly you can open Matlab from any directory you are in by typing: **matlab &**

Running a .do file in the command line
You can run a .do file from the command line by following the syntax: **stata –b do filename.do**
The **-b** part tells Stata to run in batch mode. Notice also the .log file is created automatically for us to view it.



Running a .m file in the command line
To run a Matlab .m file from the command line follow the syntax: **matlab -nodisplay -nosplash -nodesktop < filename.m > output_filename.txt**

The text preceded by a **-** (dash) gives some Matlab options to not open the UI version. The < character inputs the filename given next while the > character tells Matlab where to send the output.



More info on Matlab options when running from command line:
http://people.sc.fsu.edu/~jburkardt/m_src/matlab_commandline/matlab_commandline.html

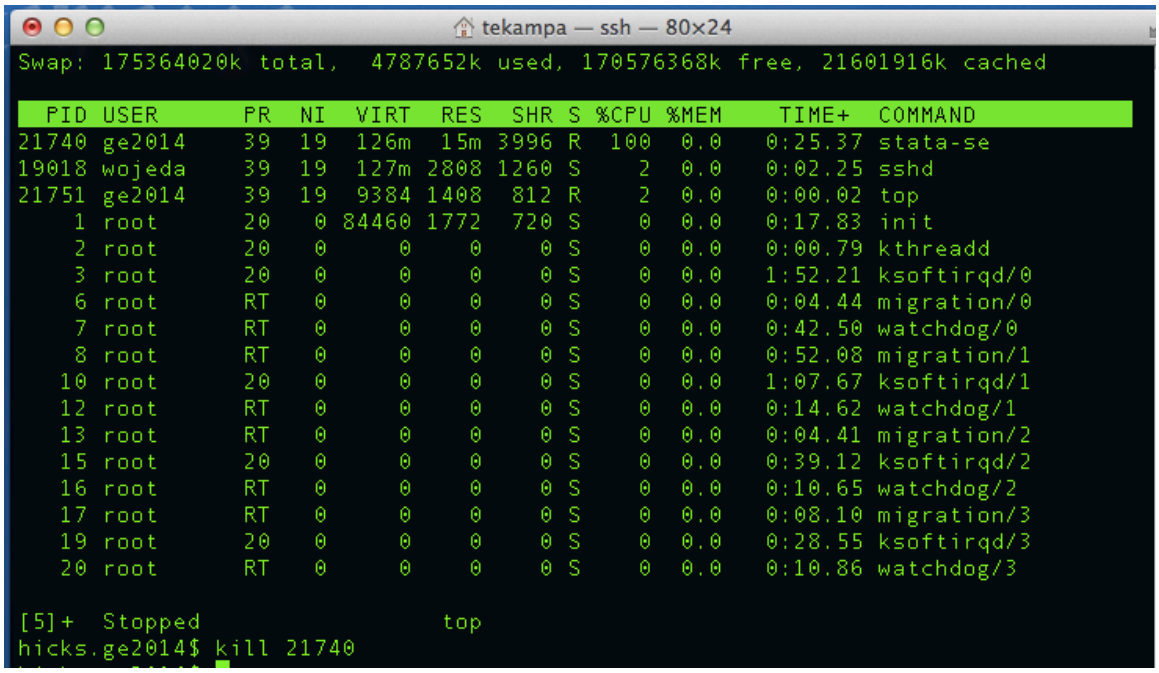
nohup

The beauty of nohup is that it will tell the terminal to keep on running your program even after you logout. So for instance following the Stata example above you would simply have in the syntax:
nohup stata –b do filename.do &

NOTE: For Stata-MP, you can't use the **nohup** command. It will close down when you log-out. Follow Daniel's instructions from last year to use Stata-MP.

top
How can you check that your program is indeed running? The **top** command will show what processes are currently running on the server you logged onto. To exit out of it type in CTRL + Z.

kill
Now what happens if you realized you made a mistake in your program that will have it running forever? **kill** allows you to terminate a process that you left running the background. When you run the **top** command you'll see you process is identified with a PID number. Type it in as: **kill PID_number**
That should terminate the process you had on the server.



Emacs
Emacs is the text editor I use on the server. You can open basically any file with text by typing: **emacs filename &**
It does take a while to get used to it so maybe another person more comfortable with it can present it in the next productivity seminar? But I'm just putting it here so you can use to at least open some files and make minor edits as needed.

evince
evince allows you to open pdfs in the terminal.
Syntax: **evince filename.pdf &**

display
display is the command to view images such as .png or .jpg files you might have created.
Syntax: **display filename.jpg &**

st
st is the command for StatTransfer to convert between different programs' data formats. For instance, you can transform a .csv file into Stata following the command.
Syntax: **st filename.csv filename.dta**
You can also move between .dbf and .sas7bdat and many more. I will say sometimes it's better to do these things directly from the programs instead of using StatTransfer since some data could be lost. Just be careful when using it for its convenience.

OTHER SHORTCUTS:

- Pressing TAB should have the terminal attempt to autofill the command, filename or directory
- Pressing the UP ARROW should have the terminal bring up previous commands
- Typing **clear** and ENTER in the command line should clear up the window and start from the top again.
- Sometimes you might get stuck in the command line where it doesn't allow you to type in more commands because it was expecting more characters. The usual way to exit out is CTRL + Z or CTRL + C.
- Typing **--help** (two dashes next to each plus help) will give you some help on what the command does and its options.

OTHER ADVICE:

- Of course, I google basically everything and it mostly answers all my questions. Thank you folks in stackoverflow.com.
- computerhope.com is usually my first step to check the exact syntax for commands and some examples.
- I recommend moving at least a minor project to the server and force yourself to run into all the problems we're bound to run into and try to solve them so you at least get familiar with the servers.