**Logging into a remote server**

To access a Linux machine from a **Windows** machine while on campus you will need to download and use Putty or another SSH (Secure Shell) client. If you use Putty, the start up screen will ask you for the host name (IP), your user name and port (commonly 22). After entering that into the window, a terminal for the other computer will open and ask for your password. The first time you access the server or any other machine via SSH, you will be warned that your computer does not recognize the other and asked if you want to proceed (In this case, you do).

To access a Linux machine from another **Linux** *or* **Mac** machine while on campus, enter 'ssh [usename]@[IP]' in a terminal and then enter password. The first time you connect you will be asked if you are sure you want to proceed.

To connect directly to a machine with fixed IP, your machine must be set to a similar IP (to connect to machine with IP 10.2.7.17 and subnet 255.255.255.0 use 10.2.7.XX with a subnet of 255.255.255.0). To do so in **Windows**, go to Network and Sharing → Network connections → Wired (or *local area*) connection → IPv4 → Properties → Static IP (or *use the following IP*). . To do so on a **Mac**, go to System Preferences → Network → Ethernet → Configure IPv4 Manually. In **Linux**, you can use the following command in a terminal to change the IP of the eth0 port:

ifconfig eth0 10.2.7.XX netmask 255.255.255.0 up

Using the ifconfig command without any options displays your current networking configurations, and ifconfig eth0 will only show the configurations of the eth0 port. Linux also has GUIs which can be used instead of the command line for connection settings.

Using SSH, you will be able to access the other computer through a terminal window, so you will need to use Linux commands to do things that you might be more used to doing through a GUI (Graphical User Interface). Necessary commands will be explained in each section. There are also plenty of free online tutorials if you want to find out more about using a Linux platform.

Similarly *SCP* copies files from one machine to another. In **Windows**, the program WinSCP, which will need to be installed, is the most convenient method. Use file protocol *SFTP* with the IP as the *Host name*, then use the GUI to select the files or folders to be copied and the transfer location. In **Linux** or on a **Mac** use the SCP command:

scp [username]@[IP]:[full filename and path of files to copy] [path of where the files should be copied]/.

This would copy files from a remote machine to the one being used while keeping the filenames the same. The same command can be used to transfer from the current machine to a remote one by moving the [username]@[IP] to the copied path portion, or to rename the files in the process by replacing the ‘.’with a new name. The SCP command also handles normal Unix wildcards or copies multiple files if they are surrounded by quotation marks.

**Useful Linux commands**

Wildcards: '\*' is a wild card for any number of characters while '?' is for only one character.

*pwd* Print Working Directory

The absolute path of the current working directory is given on the screen. All commands can be executed using relative paths (meaning the location with respect to your current working directory) or absolute paths(given with respect to the root directory, referenced by a '/' in Linux, similar to a 'C:' drive folder in Windows).

*cd* [directory] - Change Directory

Changes the working directory to the directory given after the command, which can be given by absolute path (starting with the root directory '/') or by relative path (where '..' means go up one directory to the “parent” directory). Also, '~' references the user's home directory

*ls* Lists all files and directories within the current working directory

*ls* [directory]

*ls* can also be used to only list only files fitting a pattern or files in another directory.

*less* [file]

The contents of the file given after the command will be displayed in the terminal. Typing 'q' will quit the less program and return to the normal terminal window. *less* will allow you to scroll both up and down in the file using arrow keys or the 'pg up/down' keys. Another common command *more* will only allow for scrolling down.

*cp* [original file] [new file]

Copies one file to another location or name. Typing '.' after the location of the new file is a short cut to keeping the same name (ex: *cp madeup.txt ../.* will copy 'madeup.txt' if it exists in the current directory to the parent directory and will name the new file 'madeup.txt')

*mv* [original file] [new file]

Moves a file from one location or name to another in the same manner as *cp*, but removes the original file. This is also a way to rename a file in Linux.

*mkdir* [new directory]

Makes a new directory of the name given after the command (ex: *mkdir ~/notneeded* will make a new directory named 'notneeded' in your home directory)

*rm* [file]

Removes a given file. To remove a folder and its contents **CAREFULLY** use: rm –rf [folder]

*man* [command]

Opens the manual on any command in through less with optional arguments and examples

*cat* [file]

Displays the text of a file in the terminal window

*chmod* [code] [file or directory]

Changes the permission status for a file or directory

Ex: chmod u+x script.py

Ex: chmod a+w script.py

Ex: chmod u+r script.py

A few shortcuts:

* In a terminal, you can use the arrow keys to scroll through previous commands.
* 'Tab' will auto complete file names or directory names. If there are multiple options, pressing 'Tab' twice will display all options.
* Note: In Linux, capitalization counts.
* Highlighted text can be pasted elsewhere using the center mouse button in a Linux environment, but this is not guaranteed to work through Putty on Windows and neither is *ctrl+v.*

**To edit a file in a terminal**

Make a copy of the file before you edit it! It is much easier to have a backup of the file than attempt to fix a file that was mistakenly saved and now has errors.

There are two common Linux terminal text editors: nano and vi

nano [file] This will open a file to edit in nano. Instructions will be on the bottom of the terminal.

vi [file] This will open a file to edit in vi, using vi requires some memorization, but it has more functionality than nano. If the file does not exist, it will be created.

Using vi:

* There are two full modes in vi. Command Mode will let you search for text, select words/lines/section, copy and paste, save the file, etc. By pressing 'Esc' you can return to Command Mode at any time. To enter Text Entry mode to edit text in a more normal text editing way, type 'i' for insert mode.
* In insert mode you can type and delete text normally.
* Note: You cannot scroll using the mouse. In fact, the mouse is nearly worthless, but the 'pg up/down', 'home', 'end' and arrow keys all work as normal.
* When you are done editing text, return to Command Mode, where:
* *:w* Write the file (saves it)
* *:q* Quits
* *:q!* Quits without saving
* *:wq* Writes the file and exits
* To search for in vi, by typing */string* while in command mode. The */string* command will search through the file for a given string, highlight it, and bring you to that portion of the file. If multiple strings match the input then pressing ‘n’ will take you from one match to the next.

**Running scripts in Linux automatically at another time**

You can run any scripts through the crontab to execute at a later point in time. Crontab contains a table of programs to run and the times to do so (in UTC). To view the table use *crontab –l.* To edit the table, use the command *crontab –e* which will open it in vi. The table is set up in rows of minute, hour, day of month, week, day of week, and program. Here the commas delineate separate times of the same category (i.e. at hours 3 and 5) at which to run the given program and the asterisks mean to run the program for each of the possible values in that category.

You can also replace the existing crontab with another file using *crontab [file].*

The simplest method is to write a shell script where commands are written exactly as they would be typed into the terminal, but saved within a file (normally given a “.sh” extension). To make the file executable use: *chmod 755 [file].sh.* Then add the file and an execution time to the crontab, check later to make sure that it executed correctly and remove it from crontab. Note that if using a shell script or within a terminal, following a command with ‘&’ makes it run as a background process, allowing other programs to run at the same time in the same terminal.