Introduction to the Command Line / Linux / Shell

Some material borrow from Data Carpentry Shell presentation

R. Burke Squires

Computational Genomics Specialist

BCBB / OCICB / NIAID / NIH

Python Programming for Biologists Series

Prerequisites

- Introduction to the Command Line (and Linux)
- Introduction to Jupyter Notebooks

Python Programming

- Introduction to Programming (with python)
- The Python Programming Language (for existing programmers)
- Advanced Python Programming and Best Practices
- Bioinformatics programming with Python
- Data Analysis with Python and Pandas
- Data Visualization with Python
- Building Workflows with Python
- (Biomedical Data Storage and Retrieval (with SQL and NoSQL databases))

Learning Objectives

- Describe what the command line is
- Identify programs to access the command line (on different operating systems)
- Locate the terminal emulator on your computer
- Discuss some advantages / disadvantages to the command line
- Apply your knowledge of the command line to
- Move around in the shell
- Create, move, and manipulate files and folders

Outline

- Motivation
- From Operating Systems to the Command Line
    - Operating System, UNIX/LINUX, Terminal, Terminal Emulator, Shell, Command Line
- Navigation
- Working with Files
- Additional Commands

Motivation

Why Learn Linux (and Command line)?

- Perform analyses on computer clusters (acces remote computers)
- Analyze data that is too large for laptop / desktop
- Enable reproducible research
- Most open source bioinformatics software Linux based
- Linux programs to do one thing very well and fast
- Working with text files is especially fast and easy
- Easy to build simple pipelines

Getting to the Command Line

Operating System -> UNIX / LINUX -> Terminal -> Terminal Emulator -> Shell -> Command Line

Operating system

Operating System -> UNIX / LINUX -> Terminal -> Terminal Emulator -> Shell -> Command Line

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. All computer programs require an operating system to function. (Wikiepedia) <img src="images/hubpages_os.png",width=500,height=500> Examples of operating systems include: Windows 10, macOS, Ubuntu Linux.

UNIX / LINUX

Operating System -> **UNIX / LINUX** -> Terminal -> Terminal Emulator -> Shell -> Command Line

UNIX is an operating system. Developed at AT&T Bell Labs in 1969. Philosophy:

- Portable
- Multi-tasking
- Multi-user in a time-sharing configuration
- Everything in UNIX is either a file or a process
- Use of plain text for storing data
- A hierarchical file system
- Small utilities with master program called the ‘kernel’
- Pipeline small programs together to accomplish larger goal

UNIX / LINUX

Linux is an operating system. It is a free, open-source operating system *similiar* to UNIX.

<img src="images/sco_unix_onion.png",width=500,height=500>

Terminal (then)

Operating System -> UNIX / LINUX -> **Terminal** -> Terminal Emulator -> Shell -> Command Line

<img src="images/DEC_VT100_terminal.jpg", width=500,height=500>

Terminal emulator (now)

Operating System -> UNIX / LINUX -> Terminal -> **Terminal Emulator** -> Shell -> Command Line

The Terminal (emulator) only opens a windows into the operating system. We will need to use a program interacts with the actual hardware. We use the **shell** for that.

<img src="images/mac_terminal.png", width=500,height=500>

The Matrix Movie:

```
Neo:"Do you always look at it encoded?"
Cypher:"Well, you have to. The image translators work for the construct pr
ogram."
```

<img src="images/matrix_code_movie_wallpaper_background_33473.jpg",width=600,height=600>

Shell

Operating System -> UNIX / LINUX -> Terminal -> Terminal Emulator -> **Shell** -> Command Line

The shell is a program that presents a command line interface which allows you to control your computer using commands entered with a keyboard instead of controlling graphical user interfaces (GUIs) with a mouse/keyboard combination.

<img src="images/mac_terminal.png", width=500,height=500>

How to access the shell

The shell is already available on Mac and Linux. For Windows, you'll have to download a separate program.

Mac

- On Mac the shell is available through Terminal, *Applications* -> *Utilities* -> *Terminal*

Windows

- Windows 10 install Bash on Ubuntu on Windows (https://msdn.microsoft.com/en-us/commandline/wsl/about) or here (http://www.windowscentral.com/how-install-bash-shell-command-line-windows-10)
- All Windows, install gitbash (https://git-scm.com/downloads)

Linux

- Open Terminal

Command Line

Operating System -> UNIX / LINUX -> Terminal -> Terminal Emulator -> Shell -> **Command Line**

We have finally arrived! We are in the **shell** and have access to the **command line**.

Linux <img src="images/Visual-QuickPro-Guide-Enzer-02fig05.jpg", width=500,height=500>

Windows <img src="images/wikipedia-COMMAND_LINE.png", width=500,height=500>

Shell Navigation

Lets first see where we are located.

Type (at the prompt *$*):

```
pwd
```

(If you prefer to use Windows / DOS there is a chart at the bottom of this notebook for equivelent DOS commands. There is also a DOS / Linux Cheatsheet in this folder)

```
In [ ]:
```

Jupyter Notebook Note: Execute Shell Commands in Jupyter Notebook with "!"

Shell commands can be executes in a Jupyter Notebook by preceeding the command with an exclimation mark "!". Otherwise, go to your Teminal (emulator) and run the command.

```
pwd (Mac / Linux / UNIX)

!pwd (Jupyter Notebook)
```

```
In [ ]:
```

Navigation

UNIX directory structure

<img src="images/unix1_file_system.gif", width=500,height=500>

Macs use a Users Directory as well.

To change your location we use the **cd** or *change directory* command.

At the command prompt ($) type:

```
cd /
```

```
In [ ]:
```

```

```

If you want to go back to your home folder, type *cd* with nothing after it.

```
cd
```

```
In [ ]:
```

```

```

Getting Help

You can get help on any UNIX program by using the *--help* argument (or flag)

```
cd --help
```

or you can check the program manual by using the **man** program with the name of the program you want help with. Like this:

```
man cd
```

```
In [ ]:
```

```

```

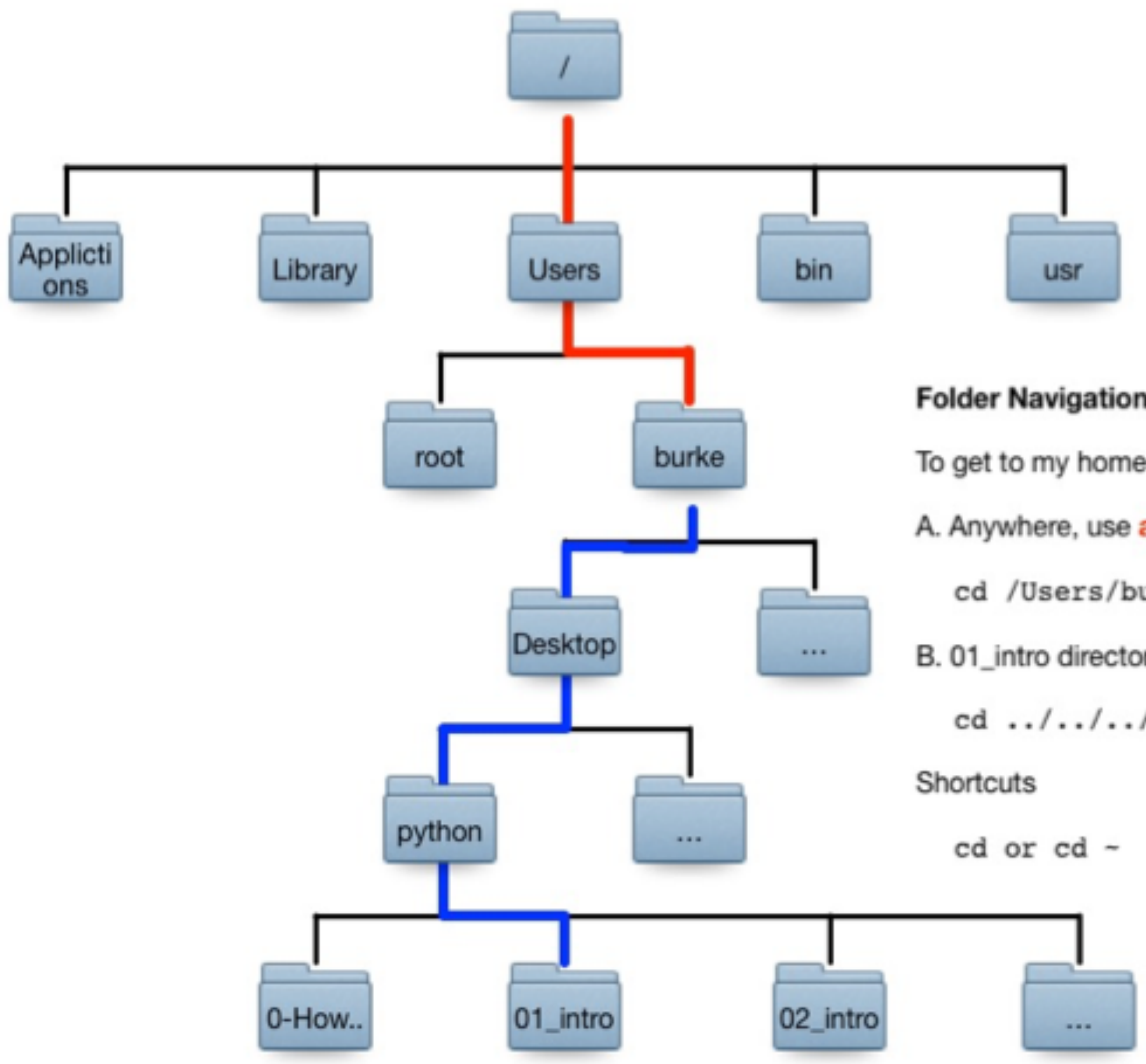
The *argument* after *cd* is the folder or directory that you want to go to.

You can specify a directory using its' **relative path** or **absolute path**.

The *relative path* is in relation to your current position in the file system. Relative directories include the current directory (shortcut "."), parent directory (shortcut ".."), child directories, or any other directory using the multiple parant directory shortcuts.

The *abolute path* is in relation to the **root directory**. A absolute path will not change unless you move the file. The relative path of a file will change everytime you change your location.

Absolute versus Relative Paths



Folder Navigation:

To get to my home directory from:

A. Anywhere, use absolute path:

```
cd /Users/burke
```

B. 01_intro directory, use relative path

```
cd ../../../
```

Shortcuts

```
cd or cd ~
```

Lets navigate into the directory that has the seminar files. If you have not already, **please copy the class files to the Desktop**.

To navigate to the desktop type:

```
cd              (makes sure you are in yor home directory)

cd Desktop
```

Details on *cd*

```
Moving around the file system:

pwd                Show the "present working directory", or current directo
ry.
cd                 Change current directory to your HOME directory.
cd /usr/STRIM100   Change current directory to /usr/STRIM100.
cd INIT            Change current directory to INIT which is a sub-director
y of the current directory.
cd ..              Change current directory to the parent directory of the
current directory.
cd $STRMWORK       Change current directory to the directory defined by the
environment variable 'STRMWORK'.
cd ~bob            Change the current directory to the user bob's home dire
ctory (if you have permission).
(Source: http://freeengineer.org/learnUNIXin10minutes.html)
```

To see if the seminar materials folder is on the Desktop lets list the contents of the Desktop folder. **ls** is short for *list* and lists the contects of the directory.

```
ls
```

Details on *ls*

```
ls -l    list a directory in long ( detailed ) format
```

for example:

```
$ ls -l
total 8
-rwxrwxrwx@  1 squiresrb  NIH\Domain Users  848 Apr  5  2016 0-How to get
started.txt
drwxrwxrwx@ 10 squiresrb  NIH\Domain Users  340 Feb  2 17:13 01_intro_comm
and_line
drwxrwxrwx@  2 squiresrb  NIH\Domain Users   68 Jan 23 16:10 02_intro_jupy
ter_notebooks
drwxrwxrwx@  2 squiresrb  NIH\Domain Users   68 Jan 23 16:10 03_intro_prog
ramming
drwxrwxrwx@  2 squiresrb  NIH\Domain Users   68 Jan 23 16:10 04_python_pro
gramming
```

```
drwxrwxrwx@  3 squiresrb  NIH\Domain Users  102 Feb  1 10:46 05_advanced_p
ython
drwxrwxrwx@  2 squiresrb  NIH\Domain Users   68 Jan 23 16:11 06_bioinfo_py
thon
drwxrwxrwx@  2 squiresrb  NIH\Domain Users   68 Jan 23 16:12 07_data_analy
sis_python
drwxrwxrwx@  2 squiresrb  NIH\Domain Users   68 Jan 23 16:12 08_data_vis_p
ython
drwxrwxrwx@  3 squiresrb  NIH\Domain Users  102 Jan 28 08:07 09_workflows_
python
drwxrwxrwx   8 squiresrb  NIH\Domain Users  272 Jan 30 10:58 long_term_evo
lution_project
drwxrwxrwx   7 squiresrb  NIH\Domain Users  238 Feb  1 09:47 new_project_n
ame
^ ^  ^  ^    ^     ^             ^           ^     ^     ^      ^
| |  |  |    |     |             |           |     |     |      |
| |  |  |    |     owner         group       size  date  time   name
| |  |  |    number of links to file or directory contents
| |  |  permissions for world
| |  permissions for members of group
| permissions for owner of file: r = read, w = write, x = execute -=no per
mission
type of file: - = normal file, d=directory, l = symbolic link, and others.
..

ls -a        List the current directory including hidden files. Hidden fil
es start
             with "."
ls -ld *     List all the file and directory names in the current director
y using
             long format. Without the "d" option, ls would list the conten
ts
             of any sub-directory of the current. With the "d" option, ls
             just lists them like regular files.
             (source: http://freeengineer.org/learnUNIXin10minutes.html#Na
vigating)
```

You should see a directory or folder named **python_programming_for_biologists**.

Challenge: How do we get into that directory?

Answer: Right, we use the *cd* command with the name of the directory!

Time Saver:

Use **tab-completion** to quickly complete entering the name of a directory or file. To do this start typing the name of a file or directory and then hit the *tab* key to fill in the rest of the name. If there are many files or directories with similia rnme you may have to type more characters.

Move into the seminar directory by typing:

```
cd python_programming_for_biologists
```

confirm this move by typing:

```
pwd
```

```
In [ ]:
```

Creating Directories

Now that we are in the seminar directory, lets create a directory structure that we will use for hte rest of the seminars. We will use a directory structure based loosly off of the structure put forth in Noble's 2009 article *A Quick Guide to Organizing Computational Biology Projects* (http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424).

We begin by making a parent directory that will hold all of our other directories. We will name the directory after the analysis we are replicating, namely *long term evolution analysis*. To make a new directory we use the the bash command **mkdir**.

```
mkdir long_term_evolution_analysis
```

```
In [ ]:
```

Details on moving and removing directories:

```
mkdir dir1 [dir2...]    create directories
mkdir -p dirpath        create the directory dirpath, including all implie
d directories in the path.
rmdir dir1 [dir2...]    remove an empty directory
(Source: http://freeengineer.org/learnUNIXin10minutes.html)
```

We will use underscores in place of spaces as it makes typing and tab-completion easier and faster in the shell. We can confirm that we have created the new directory by listing or *ls* the contents of teh crrent directory.

```
$ls
```

In [ ]:

```
```

We should see a directory names _long_term_evolution*analysis*.

Next, we will move into the new directory. To do this we type:

```
cd lon<tab>
```

In [ ]:

```
```

Remember the tab-completion should complete the project name. To confirm we have moved into the newly created folder we check out preent working directory by typing:

```
pwd
```

In [ ]:

```
```

Now that we are in the project directory lets create the directories for each part of our analysis. Do this my typing:

```
mkdir 01_doc 02_data 03_code 04_results 05_output
```

```
In [ ]:
```

You are of course welcome to organize your projects in any way you choose but for the purpose of these seminars, we will use this structure.

We want to create subfolders in the data directory for raw and cleaned data. Can could *cd* into the *data* directory and create the directories, but we can also create them from our current location. We do this by typing:

```
mkdir ./02_data/raw_data ./02_data/cleaned_data
```

The ./ tells the make directory program (*mkdir*) to look in the current directory for a subdirectory named 02_data.

Complete this task by creating a manuscript directory in our current directory. We will them move it into place. To create the manuscript directory type:

```
mkdir ./05_output/manuscript
cd ./05_output/manuscript
mkdir figures references tables
```

Lets go back to our main project directory.

```
cd ../../
```

Challenge: How to you KNOW you are in the main project directory. Remeber with relative paths we could be anywhere! :-)

Answer: Use *pwd* to show the present working directory. NOTE: You can alos look at your promp for the location.

```
In [ ]:
```

NOTE: To 'clear' the screen (really create a screenfull of empy lines) type:

```
clear
```

Making and moving files

Lets get started with files by creating a README file at the top level of our project. In macOS or Windows we would open a word processing program like Microsoft® Word, TextEdit or NotePad. In the UNIX environment we will use one of may text editors. There are a number of text editors, including emacs, vim, nano, etc. While emacs and vim are very powerful I find that they also have a steep learning curve. So I prefer to use **nano**.

Lets first just open the *nano* text editor by typing:

```
nano
```

You should see something like this <img src="images/nano_text_editor.jpg", width=500,height=500>

In the _nano_text editor type the title of the document "README". Then lets save it. We save by typing:

```
<control>X
```

nano will ask is we want to save. Type

```
"Y"
```

Then at the prompt "File Name To Write" type:

```
"README.txt"
```

and hit Enter.

We can see out file by listing the directory contents with *ls*. You should see:

```
01_doc         03_code        05_output
02_data         04_results    README.txt
```

```
In [ ]:
```

Now that we have a generic README file, lets make a copy of it and then move the copy into a subfolder. To copy a file in UNIX, we use the copy or **cp** command.

```
cp source_file copied_file
```

Specifically, in our case we will type:

```
cp README.txt README1.txt
```

We can then us *ls* to see that we have a copy. Type

```
ls
```

You should see:

```
01_doc        03_code        05_output    README1.txt
02_data        04_results    README.txt
```

```
In [ ]:
```

Since we do not need two README files in the parent directory, lets move on the copy into the documents directory.

To move a file in UNIX we use the **mv** command. We will also change teh name, to remove the "1" from the file at the same time we move it. To accomplish this we type:

```
mv README1.txt ./01_doc/README.txt
```

```
In [ ]:
```

Details on moving, renaming, and copying files:

```
cp file1 file2          copy a file
mv file1 newname        move or rename a file
mv file1 ~/AAA/         move file1 into sub-directory AAA in your home dir
ectory.
rm file1 [file2 ...]    remove or delete a file
rm -r dir1 [dir2...]    recursivly remove a directory and its contents BE
CAREFUL!
```

Now that we have the file moved we can use a few other UNIX commands to look at it. To view the file, without actually openning it, type:

```
cat README.txt
```

Details on viewing and editing files

```
cat filename      Dump a file to the screen in ascii.
more filename     Progressively dump a file to the screen: ENTER = one lin
e down
                  SPACEBAR = page down  q=quit
less filename     Like more, but you can use Page-Up too. Not on all syste
ms.
vi filename       Edit a file using the vi editor. All UNIX systems will h
ave vi in some form.
emacs filename    Edit a file using the emacs editor. Not all systems will
have emacs.
head filename     Show the first few lines of a file.
head -n  filename Show the first n lines of a file.
tail filename     Show the last few lines of a file.
tail -n filename  Show the last n lines of a file.
(Source: http://freeengineer.org/learnUNIXin10minutes.html
```

Additional UNIX commands that are important to know

Searching for strings in files: The grep command

```
grep string filename    prints all the lines in a file that contain the st
ring
```

Searching for files : The find command

```
find search_path -name filename

find . -name aaa.txt    Finds all the files named aaa.txt in the current d
irectory or
                        any subdirectory tree.
find / -name vimrc      Find all the files named 'vimrc' anywhere on the s
ystem.
find /usr/local/games -name "*xpilot*"
                        Find all files whose names contain the string 'xpi
lot' which
                        exist within the '/usr/local/games' directory tree
.

(Source: http://freeengineer.org/learnUNIXin10minutes.html)
```

Redirection:

```
grep string filename > newfile           Redirects the output of the above
grep
                                         command to a file 'newfile'.
grep string filename >> existfile        Appends the output of the grep co
mmand
                                         to the end of 'existfile'.
```

The redirection directives, > and >> can be used on the output of most commands to direct their output to a file.

Pipes:

The pipe symbol "|" is used to direct the output of one command to the input of another.

For example:

```
    ls -l | more   This commands takes the output of the long format directory
    list command
                   "ls -l" and pipes it through the more command (also known a
    s a filter).
                   In this case a very long list of files can be viewed a page
    at a time.

    du -sc * | sort -n | tail
                   The command "du -sc" lists the sizes of all files and direc
    tories in the
                   current working directory. That is piped through "sort -n"
    which orders the
                   output from smallest to largest size. Finally, that output
    is piped through "tail"
                   which displays only the last few (which just happen to be t
    he largest) results.
```

(Source: http://freeengineer.org/learnUNIXin10minutes.html (http://freeengineer.org/learnUNIXin10minutes.html))

Learning Objectives

- Describe what the command line is
- Identify programs to access the command line (on different operating systems)
- Locate the terminal emulator on your computer
- Discuss some advantages / disadvantages to the command line
- Apply your knowledge of the command line to
- Move around in the shell
- Create, move, and manipulate files and folders

Resources

- The Linux Command Line (http://linuxcommand.org/tlcl.php (http://linuxcommand.org/tlcl.php))
- Ian Korf's Lab Unix Bootcamp (http://korflab.ucdavis.edu/bootcamp.html (http://korflab.ucdavis.edu/bootcamp.html))
- Cliff at FreeEngineers' UNIX in Ten Minutes (http://freeengineer.org/learnUNIXin10minutes.html (http://freeengineer.org/learnUNIXin10minutes.html))
- Repository of bash commands (https://github.com/jarv/cmdchallenge (https://github.com/jarv/cmdchallenge))
- TutorialsPoint UNIX tutorial (https://www.tutorialspoint.com/unix/ (https://www.tutorialspoint.com/unix/))

Shell cheat sheets:

- http://fosswire.com/post/2007/08/unixlinux-command-cheat-sheet/ (http://fosswire.com/post/2007/08/unixlinux-command-cheat-sheet/)
- https://github.com/swcarpentry/boot-camps/blob/master/shell/shell_cheatsheet.md (https://github.com/swcarpentry/boot-camps/blob/master/shell/shell_cheatsheet.md)

Explain shell - a web site where you can see what the different components of a shell command are doing.

- http://explainshell.com (http://explainshell.com)
- http://www.commandlinefu.com (http://www.commandlinefu.com)

Addendum

In [ ]:

```
# Wrapping it all up...in the future we can just run this script.
# The advantages of this script it is that it (should) work on Mac, Windows, Lin
ux

import os

project_name = '../long term evolution analysis'
project_name = project_name.replace(' ', '_')

# Create the documents folder with a planning folder for pre-project discussions
os.makedirs('%s/%s/%s' % (project_name, '01_doc', 'planning'), exist_ok=True)

# Create the data folder with a raw data and cleaned data sub-folders
os.makedirs('%s/%s/%s' % (project_name, '02_data', 'raw_data'), exist_ok=True)
os.makedirs('%s/%s/%s' % (project_name, '02_data', 'cleaned_data'), exist_ok=Tru
e)

# Create the source code folder for our python scripts
os.makedirs('%s/%s' % (project_name, '03_code'), exist_ok=True)

# Create the results folder for our intermediate results
os.makedirs('%s/%s' % (project_name, '04_results'), exist_ok=True)

# Create the output folder with a manuscript sub-folder
os.makedirs('%s/%s/%s/%s' % (project_name, '05_output', 'manuscript', 'figures')
, exist_ok=True)
os.makedirs('%s/%s/%s/%s' % (project_name, '05_output', 'manuscript', 'reference
s'), exist_ok=True)
os.makedirs('%s/%s/%s/%s' % (project_name, '05_output', 'manuscript', 'tables'),
exist_ok=True)
```

DOS Equivelent to UNIX / LINUX Commands

<img src="images/dos_linux_commands.png", width=500,height=500>

Q & A

In [ ]:

```
```