Intro Linux/Unix Shell Tutorial

Author: Francisco McGee

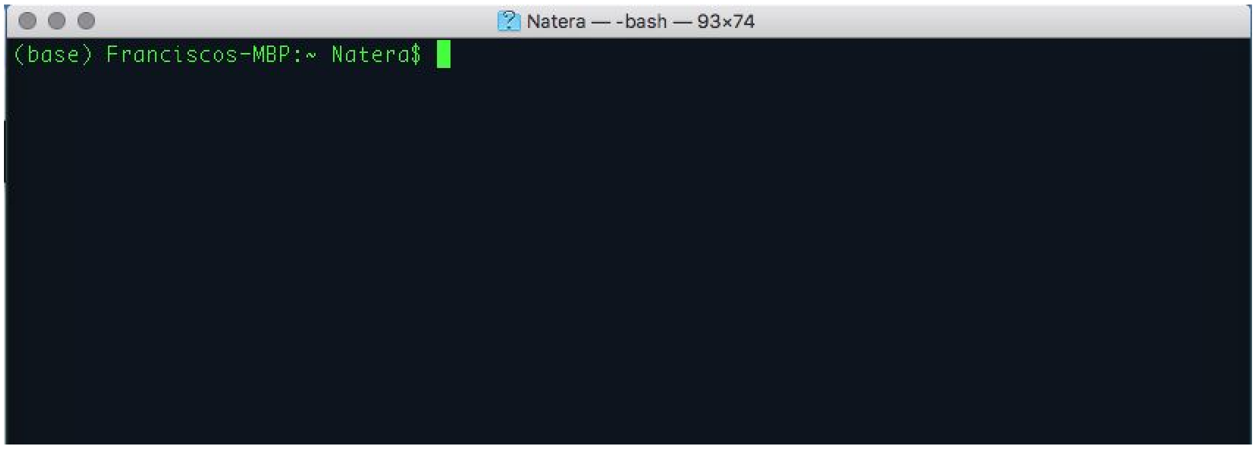
I. Welcome!

This tutorial is designed for newcomers to the Linux/Unix command line. It assumes you already have access to a Unix “terminal” or “shell”, which are alternate names for the command line in which you will enter your commands to the computer.

Usually, the default “shell” is actually a particular flavor of shell called “bash.” That stands for the “Bourne-again shell”, because it’s the second version of shell initially created by Stephen Bourne. This tutorial is actually just for bash, and may not work with other shell versions.

We’ll be showing you how to navigate a Unix file system, how to manipulate text files, and how to make/run basic scripts. With these skills, you should be able to get started in Linux/Unix and branch out in whichever direction you need!

Anaconda gives you a Unix terminal on any platform, though Mac and Linux distributions have one built-in. Once you find it and open it on your platform, then we can get started. It should look something like this:



```
(base) Franciscos-MBP:~ Natera$
```

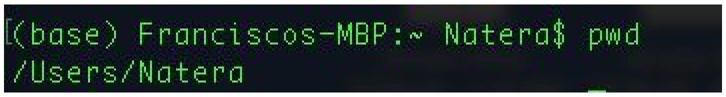
II. Navigating a Unix file system

Where am I ?

When you first open your terminal, its default location will probably be your "home directory."

You can always check where you are in the file system using the command:

```
pwd
```



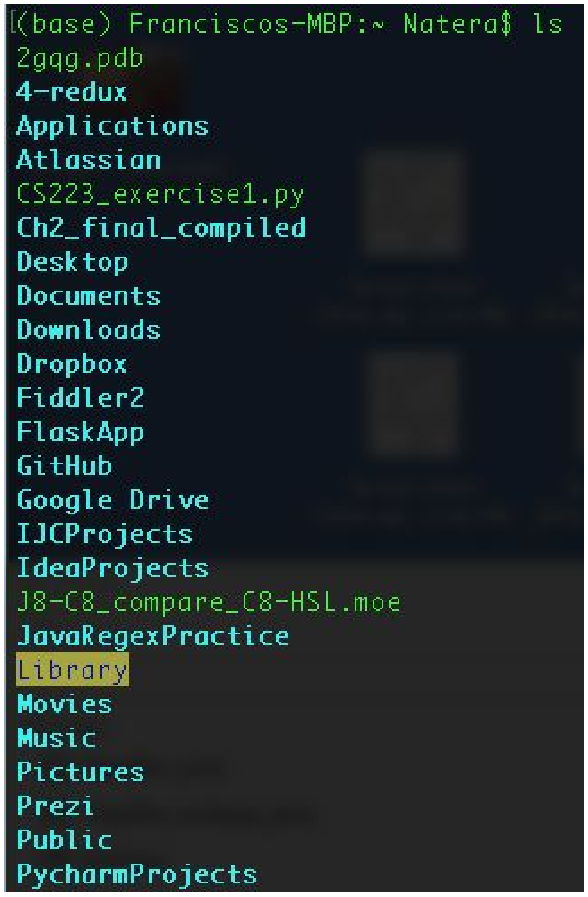
```
[(base) Franciscos-MBP:~ Natera$ pwd
/Users/Natera
```

Indeed, my home directory has the path "/Users/Natera". This "path" is a path from my root directory, "/", to the current directory that I'm in. Every file and folder on a Unix file system has a path, which is basically like its "address" in the file system. Each step in a path on Mac and Linux systems are separated by a "/", called a "forward slash" because it starts on the left and goes "forward" to the right. On Windows, you might see a "\", or "back slash", used instead.

What's in this place where I'm currently at?

You can find out what's in your current location by using the "list" command, "ls":

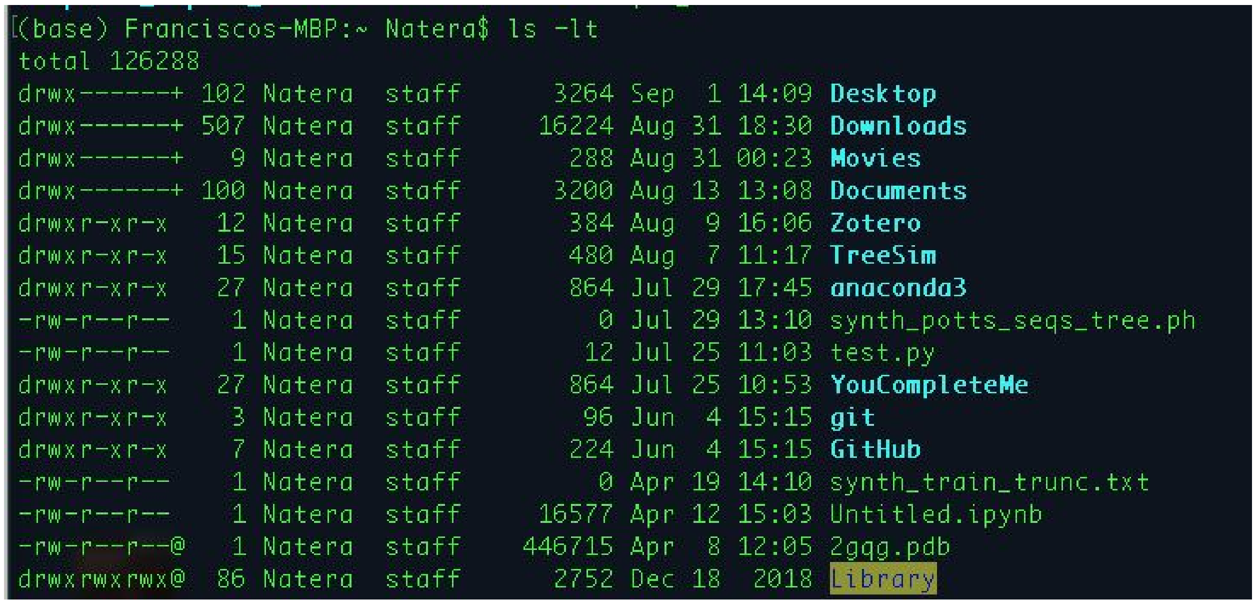
```
ls
```



```
(base) Franciscos-MBP:~ Natera$ ls
2gqg.pdb
4-redux
Applications
Atlassian
CS223_exercise1.py
Ch2_final_compiled
Desktop
Documents
Downloads
Dropbox
Fiddler2
FlaskApp
GitHub
Google Drive
IJCProjects
IdeaProjects
J8-C8_compare_C8-HSL.moe
JavaRegexPractice
Library
Movies
Music
Pictures
Prezi
Public
PycharmProjects
```

Yikes! That’s a lot of stuff! Sometimes it’s hard to find stuff with just “ls”. You can add “flags” to certain commands that bring added functionality to the command. For example, I often use the flags “-lt” to give me longer file names and then sort them with the most recent files at the top. This makes it easier to find things.

```
ls -lt
```



```
(base) Franciscos-MBP:~ Natera$ ls -lt
total 126288
drwx------+ 102 Natera  staff     3264 Sep  1 14:09 Desktop
drwx------+ 507 Natera  staff    16224 Aug 31 18:30 Downloads
drwx------+   9 Natera  staff      288 Aug 31 00:23 Movies
drwx------+ 100 Natera  staff     3200 Aug 13 13:08 Documents
drwxr-xr-x   12 Natera  staff      384 Aug  9 16:06 Zotero
drwxr-xr-x   15 Natera  staff      480 Aug  7 11:17 TreeSim
drwxr-xr-x   27 Natera  staff      864 Jul 29 17:45 anaconda3
-rw-r--r--    1 Natera  staff        0 Jul 29 13:10 synth_potts_seqs_tree.ph
-rw-r--r--    1 Natera  staff       12 Jul 25 11:03 test.py
drwxr-xr-x   27 Natera  staff      864 Jul 25 10:53 YouCompleteMe
drwxr-xr-x    3 Natera  staff       96 Jun  4 15:15 git
drwxr-xr-x    7 Natera  staff      224 Jun  4 15:15 GitHub
-rw-r--r--    1 Natera  staff        0 Apr 19 14:10 synth_train_trunc.txt
-rw-r--r--    1 Natera  staff    16577 Apr 12 15:03 Untitled.ipynb
-rw-r--r--@   1 Natera  staff   446715 Apr  8 12:05 2gqg.pdb
drwxrwxrwx@  86 Natera  staff     2752 Dec 18  2018 Library
```

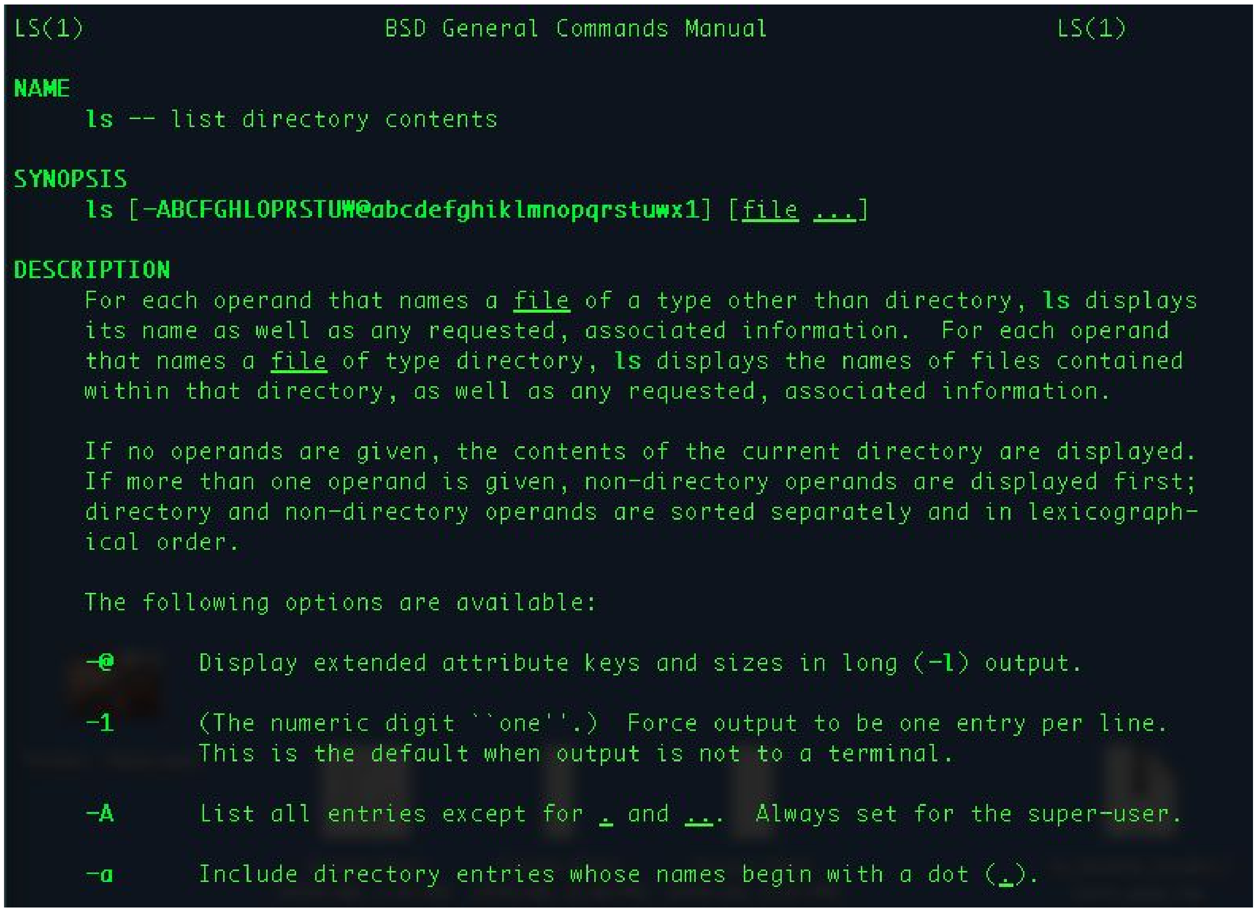
How can I find out more about a specific command?

You can find more information on any command, including all the different flags you can use, using the “man” command:

```
man <name of command>
```

Here, I’ll run “man” on “ls”:

```
man ls
```



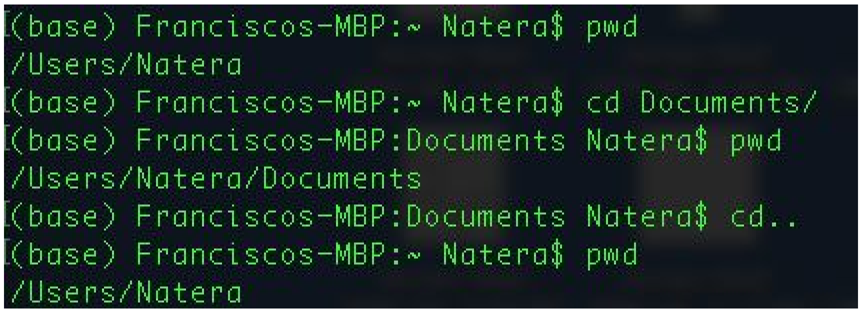
```
LS(1)                    BSD General Commands Manual                   LS(1)

NAME
     ls -- list directory contents

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls displays
     its name as well as any requested, associated information.  For each operand
     that names a file of type directory, ls displays the names of files contained
     within that directory, as well as any requested, associated information.

     If no operands are given, the contents of the current directory are displayed.
     If more than one operand is given, non-directory operands are displayed first;
     directory and non-directory operands are sorted separately and in lexicograph-
     ical order.

     The following options are available:

     -@      Display extended attribute keys and sizes in long (-l) output.

     -1      (The numeric digit ``one''.)  Force output to be one entry per line.
             This is the default when output is not to a terminal.

     -A      List all entries except for . and ...  Always set for the super-user.

     -a      Include directory entries whose names begin with a dot (.).
```

The information about the command should pop up right inside your shell. You can exit the “man” page by typing the letter “q”.

How do I go to different folders?

Let's say you're in your home directory, and there's a folder in there called "Documents", and you want to go there. You can go there using the "change directory" command or "cd":

```
cd Documents
```



```
[(base) Franciscos-MBP:~ Natera$ pwd
/Users/Natera
[(base) Franciscos-MBP:~ Natera$ cd Documents/
[(base) Franciscos-MBP:Documents Natera$ pwd
/Users/Natera/Documents
[(base) Franciscos-MBP:Documents Natera$ cd..
[(base) Franciscos-MBP:~ Natera$ pwd
/Users/Natera
```

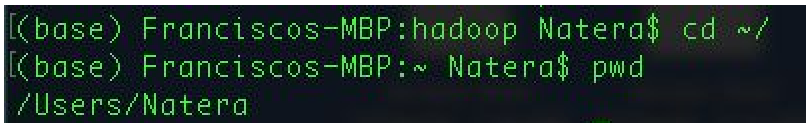
In the commands used above, I first checked where I was with "pwd", changed directories to "Documents" with "cd"... then went *backwards* or "up" one directory using a special "cd" command:

```
cd ..
```

I got lost in my file system!! How do I get back home?

No worries! There's a quick command to get back home! Use "cd" with "~/":

```
cd ~/
```



```
[(base) Franciscos-MBP:hadoop Natera$ cd ~/
[(base) Franciscos-MBP:~ Natera$ pwd
/Users/Natera
```

III. File manipulation - read, copy, move text files

How do I just see what's inside of a text file?

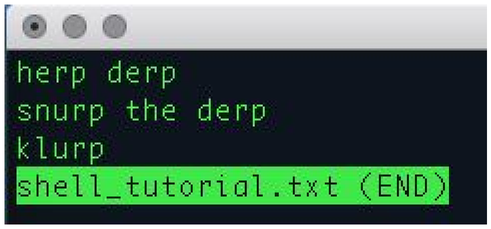
You have a couple options to visually read a file within your shell. I usually just use the command "less":

```
less <name of file>
```

For example, if there's a file called "shell_tutorial.txt" that I want to read, I just type:

```
less shell_tutorial.txt
```

The text within the file pops up in my shell like this:



```
herp derp
snurp the derp
klurp
shell_tutorial.txt (END)
```

Typing the letter "q" allows you to exit.

How do I copy a file?

Use the "cp" command to copy a file into the same directory that you are currently in:

```
cp <name of old file to be copied> <name of new copy>
```

For example, to make a copy of "shell_tutorial.txt" named "shell_tutorial_copy.txt":

```
cp shell_tutorial.txt shell_tutorial_copy.txt
```

How do I move a file?

Let's say I'm in my home directory, but I want to move a copy of a file into another directory. You can use the "move," or "mv" command:

```
mv <name of file> <path where I want to put the file>
```

For example, if I have a file named "shell_tutorial.txt" in the folder I'm currently in, and I want to move it to my Documents folder:

```
mv shell_tutorial.txt /Users/Natera/Documents
```

How do I make a new directory?

You can make a new directory using:

```
mkdir <name of new directory>
```

If I want to make a new directory named "new_stuff", I type:

```
mkdir new_stuff
```

Notice I put a "_" character in between "new" and "stuff." The empty space character " " requires special treatment, and can cause certain commands to not work as expected or at all. Thus, using "_" instead of " " is generally a good convention to follow on a Unix file system for naming files.

IV. Make/run shell scripts

How do I make a “bash script”?

Sometimes you have several commands that you want to weave together to perform a series of instructions. Instead of typing them all out individually, you can string them together in a “script,” which is a special file with the extension “.sh”.

To make a bash script, go to your favorite text editor, open up a new file and call it “test_script.sh”. On a Mac, I prefer to use Text Wrangler.

At the top of your script, you need to put in a special line:

```
#!/bin/bash
```

This special line has no spaces. It indicates to your shell that this file is a bash script. Without it, your shell will not be able to run your script.

Now, in the next line, put in a comment indicating that you are the author:

```
# Author: <insert your name>
```

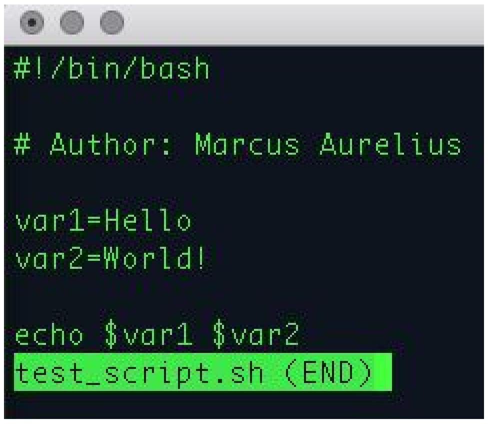
The “#” symbol indicates to the shell that this is a comment line, and should not be executed as a command in your script.

On the next several lines, define some variables in your script, like this:

```
var1=Hello
var2=World!

echo $var1 $var2
```

The “echo” command will print a variable into the shell so that you can see it when you run your script.

Finally, save your file in your editor to your home directory, go back to your shell, and then read the file in your shell using the “less” command to make sure it looks like this:



```
#!/bin/bash

# Author: Marcus Aurelius

var1=Hello
var2=World!

echo $var1 $var2
test_script.sh (END)
```

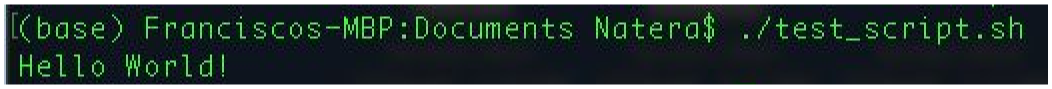
<u>How do I run or execute a bash script?</u>

You can run any bash script using “./”, then the name of the script:

```
./<name of script>
```

In our case, we type:

```
./test_script.sh
```



```
[(base) Franciscos-MBP:Documents Natera$ ./test_script.sh
 Hello World!
```

If everything works, you should see the phrase, “Hello World!” printed out in your shell.

If it worked, then congratulations! And welcome to the wonderful world of bash scripting!