Basic Commands in the Terminal: Traversing and Creating New Directories

Author: Michelle Franc Ragsac (mragsac@eng.ucsd.edu)

Notebook Information:

This Jupyter Notebook contains the `UNIX` commands that were used in Module 1a: Bench to Terminal.
It's running with a `Bash` kernel so that all of the code blocks will have the output from running a command directly below them.

Where are you currently? ¶

When using a `terminal`, you will always be inside a folder or directory. But at any given time, you may lose track of what directory you are currently in.

To figure out where you are currently located, you can use the `pwd` command to "print working directory":

```
In [1]: pwd
```

```
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal
```

When I run the `pwd` command, my current—or "working"—directory is `/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal`.

This is also the location of where this notebook is located on my computer!

> **Note:** The location that is output by the `pwd` command will reflect your own location, so it should have a different output from this notebook.

The structure of the `pwd` command's result is what we call our **absolute path**, or your absolute location on your computer. An absolute path is a way of navigating to a specific location regardless of your starting point.

Where can we go?

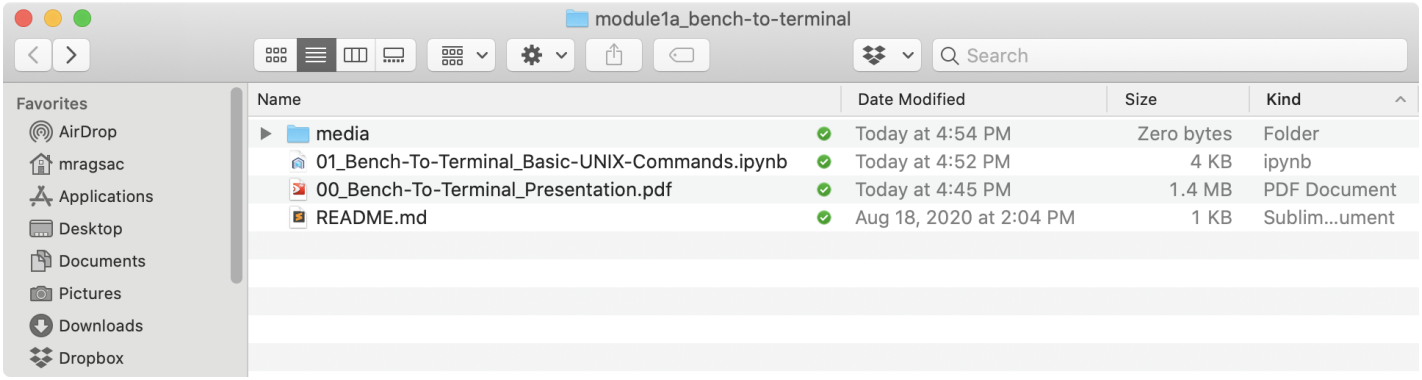
Now that we know where we are, we can try and see (or "list") what folders are in my working directory with the command `ls`.

```
In [2]: ls
```

```
00_Bench-To-Terminal_Presentation.pdf
01_Bench-To-Terminal_Basic-UNIX-Commands.ipynb
README.md
media
```

From this command, we can see the files and folders that are located in my current location.

This command should also reflect what is present when you look at the folder in a graphic user interface or "GUI", such as the macOS `Finder` application:



How can we make a new directory?

Say we wanted to create a new folder called " `new_directory_1` ". We can use the command `mkdir` to "make directories":

In [3]:
```
# First, make a new directory
mkdir new_directory_1

# After, immediately list out the contents of our current location to check if the new d
irectory was made
ls
```

```
00_Bench-To-Terminal_Presentation.pdf
01_Bench-To-Terminal_Basic-UNIX-Commands.ipynb
README.md
media
new_directory_1
```

Huzzah! I was able to make a new directory called `new_directory_1` within my current location.

> **Warning:** When naming a directory, it is always a good idea to separate words with an underscore character (_) to prevent unusual notation that you wouldn't expect.
>
> For example, if you ran the `mkdir` command to create the `new_directory_1` folder with spaces instead of underscores (e.g., `mkdir new directory 1`), it would create **three** separate directories named " `new` ", " `directory` ", and " `1` " instead of the single directory we wanted called " `new_directory_1` ".

How can we make nested directories?

You can also make many levels of subdirectories simultaneously using the `mkdir` command, but you need to add an additional note, or **flag**, to do it in one command. This is the `-p` flag, which adds additional "parents" to the directories you want to create.

Say we wanted to create the following nested folder structure, " `new_directory_2/this/is/nested` ". We can use the command `mkdir` with the `-p` flag to create this nested structure:

In [4]:
```
# First, make the directories with the -p flag
mkdir -p new_directory_2/this/is/nested

# After, immediately list out the contents of our current location to check if the new d
irectory was made
ls
```

```
00_Bench-To-Terminal_Presentation.pdf
01_Bench-To-Terminal_Basic-UNIX-Commands.ipynb
README.md
media
new_directory_1
new_directory_2
```

From this, we can see that we made our second directory, `new_directory_2`. Let's check to see if the nested directories were made:

In [5]:
```
# Check if the "this" folder is present within new_directory_2/
ls new_directory_2
```

```
this
```

In [6]:
```
# Check if the "is" folder is present within new_directory_2/this/
ls new_directory_2/this
```

```
is
```

In [7]:
```
# Check if the "nested" folder is present within new_directory_2/this/is/
ls new_directory_2/this/is
```

```
nested
```

We were able to generate all of the directories with the `-p` flag of the `mkdir` command!

From this portion, we were also able to demonstrate how `ls` can be used to list out the directories within a particular folder that we're interested in.

Learning More about Commands with the `man` Command

Any command you use on the terminal will probably have multiple flags associated with it; for example, the `mkdir` command has more flags than just the `-p` flag! To learn more about a command and see what available flags there are, you can use the `man` function in front of the command name (e.g., `man mkdir`) to access that command's "manual".

In [8]:
```
# Learn more about the mkdir command with the man command
man mkdir
```

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

NAME
     mkdir -- make directories

SYNOPSIS
     mkdir [-pv] [-m mode] directory_name ...

DESCRIPTION
     The mkdir utility creates the directories named as operands, in the order
     specified, using mode rwxrwxrwx (0777) as modified by the current
     umask(2).

     The options are as follows:

     -m mode
             Set the file permission bits of the final created directory to
             the specified mode.  The mode argument can be in any of the for-
             mats specified to the chmod(1) command.  If a symbolic mode is
             specified, the operation characters ``+'' and ``-'' are inter-
             preted relative to an initial mode of ``a=rwx''.

     -p      Create intermediate directories as required.  If this option is
             not specified, the full path prefix of each operand must already
             exist.  On the other hand, with this option specified, no error
             will be reported if a directory given as an operand already
             exists.  Intermediate directories are created with permission
             bits of rwxrwxrwx (0777) as modified by the current umask, plus
             write and search permission for the owner.

     -v      Be verbose when creating directories, listing them as they are
             created.

     The user must have write permission in the parent directory.

DIAGNOSTICS
     The mkdir utility exits 0 on success, and >0 if an error occurs.

SEE ALSO
     rmdir(1)

COMPATIBILITY
     The -v option is non-standard and its use in scripts is not recommended.

STANDARDS
     The mkdir utility is expected to be IEEE Std 1003.2 (``POSIX.2'') compat-
     ible.

HISTORY
     A mkdir command appeared in Version 1 AT&T UNIX.

BSD                             January 25, 1994                             BSD
```

How can we change our working directory?

We don't want to always stay in the same folder, so let's use the `cd` command to "change directories" to another one on the command line:

In [9]:
```
# Change our current directory
cd new_directory_1

# List the directory that we're located in to see what happened
pwd
```

```
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal/new_di
rectory_1
```

And just like that, I'm now in the directory that I created in the previous section, `new_directory_1` !

Navigating through Directories with the `..` Notation

With the previous command, we moved forward through a directory by selecting a folder that was present at our current location.

We can move back up a level in our directory hierarchy using two periods (" `..` ") with the `cd` command:

In [10]:
```
# List the directory that we're currently located in
pwd

# Move up a directory with the .. notation
cd ..

# List the directory that we're in now
pwd
```

```
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal/new_di
rectory_1
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal
```

Previously, I was located in the directory `/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal/new_directory_1` , then after running the `cd` command, I moved up in the directory hierarchy to the directory `/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal` !

We can also move up multiple levels by separating the two periods with a forward slash (" `/` "):

In [11]:
```
# List the directory that we're currently located in
pwd

# Move up two directories using the .. notation
cd ../..

# List the directory that we're in now
pwd
```

```
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020
```

Note: The `..` notation allows us to change our relative path, or location relative to where we are now. Compare this to our **absolute path**, where the entire path is defined in identifying our absolute location. We can also change location to any other directory, such as the `module1a_bench-to-terminal` directory from our current location, so as long as we provide an absolute path.

In [12]:
```
# List the directory that we're currently located in
pwd

# Move to the specific directory that we're interested in changing to
cd /Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal

# List the directory we're in now
pwd
```

```
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal
```

Warning: If you're ever typing out an absolute path, you might introduce mistakes due to typos. Computers need to follow a set of instructions precisely, so typos aren't tolerated! Thus, we can use something called **tab-completion** to fill in what is already known by the computer—such as existing folder names in a directory.

All you need to do is press the `TAB` key to auto-complete the file or folder name you're interested in. If there's a file or folder that has similar naming to each other, you can press the `TAB` key **twice** to see all the objects that fit the description; afterwards, you can add additional characters and use tab-complete to go to the file or folder of interest. Use this technique to reduce typos and navigate through your files in the terminal faster!

The Tilde (~) Notation

The tilde (" ~ ") is a `UNIX` shortcut to denote your "home directory", or the directory that you're automatically redirected to whenever you open up a terminal window.

Note: On your keyboard, the key with the tilde (~) is shared with the back quote key ("`" ). This key should be located to the **left** of the `1/!` key on the number row. To type it, press `SHIFT + ` .

In [13]:
```
# List the directory we're curently located in
pwd

# Move to the home directory using the ~ notation
cd ~

# List the directory we're in now
pwd

# List the contents of this directory
ls
```

```
/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal
/Users/mragsac
Applications                Google Drive File Stream School
Desktop                     Library                 Zotero
Documents                   Movies                  bin
Downloads                   Music                   miniconda3
Drive                       Pictures
Dropbox                     Public
```

From this code cell, I started out in the `/Users/mragsac/Dropbox/School/BISB-Bootcamp-2020/day1/module1a_bench-to-terminal` directory, and then ended up in the `/Users/mragsac` directory after using the tilde in my `cd` command. When I list out the contents of this directory, we can see that it reflects the same directory that appears when I open a new `Finder` window or open a new terminal!