Tutorial Objective

• This is designed to teach you enough to start using the Generic Mapping Tools
• Just the tip of the iceberg!
Open the Terminal application

• On a Mac: Applications -> Utilities -> Terminal
Open the Terminal application

• On a Mac: Applications -> Utilities -> Terminal
Open the Terminal application

• On a Mac: Applications -> Utilities -> Terminal
Navigating in the terminal

• `pwd`: print present working directory
Navigating in the terminal

- `pwd`: print present working directory

/Users/mherman is my “home” directory. When you open your terminal, by default you start in your home directory. “Directory” is another term for a Folder.
Navigating in the terminal

- `ls`: list contents of current directory

Now type "ls" to see what files or subdirectories are in my home directory.
Navigating in the terminal

• `ls`: list contents of current directory

These are the files and directories in my home directory.
Navigating in the terminal

• **cd:** change directory

Let's change directory to the Desktop. Type in “cd Desktop”

The syntax is always “cd <directory>”. In other words, cd requires the name of a directory as an “argument.”
Navigating in the terminal

- `cd`: change directory

Check that directory was changed with "`pwd`"
You can see these match the items on my Desktop.

Navigating in the terminal

- `cd`: change directory
Navigating in the terminal

• **cd**: change directory

Two special directories are “.” and “..”
- “.” is the current directory
- “..” is the directory above the current one.
Navigating in the terminal

- `cd`: change directory

Here I used “cd ..” to go up one directory, ending up back in my home directory.
Making a new directory

- `mkdir`: make new directory

Back in the home directory, I typed “`mkdir unix_tutorial`” to make a new folder called `unix_tutorial`.

Note: avoid using spaces for file or directory names.
We will work in this new directory. Change into unix_tutorial and verify the directory is clean and empty.

Making a new directory
• mkdir: make new directory
Manipulating files

- touch: make a new blank file

Type in the command “touch file1” to create an empty file named “file1.” Verify that this file is now in your directory.
Manipulating files

• `cp`: copy file

To duplicate a file, use the copy command: "cp file1 file2"

This copies everything in file1 into file2, and keeps file1.
To rename a file, deleting the original, use the move command: “mv file2 file3”
Manipulating files

- `rm`: remove file (IRREVERSIBLE!)

To delete a file, use the `rm` command: “rm file3”
Manipulating files

- `rm`: remove file (IRREVERSIBLE!)

You cannot undo the “rm” operation. Once a file is deleted, it is gone forever (equivalent to emptying trash). Use with caution.

To delete a file, use the `rm` command: “rm file3”
I am starting with a clean `unix_tutorial` directory. I removed the last file in the directory with “`rm file1`”

Input and output
- `echo`: print following values to terminal screen
The echo command prints its arguments (whatever follows it) to the terminal screen. For example, typing

“echo 1 2 3 4”

prints “1 2 3 4” on the next line.

Input and output

- echo: print following values to terminal screen
You can take this output and put it into a file instead of printing it to the terminal. This is done by using “>” (the redirect arrow). For example, to save the previous output into file1, use

“echo 1 2 3 4 > file1”
To verify that the file contains what we expect, type “cat file1”. This prints the contents of a file to the terminal screen.

Input and output

- `cat`: concatenate (print contents of) file
Input and output

• \texttt{\>\>}: redirect output to a file (append)

The single redirect arrow ("\>") overwrites the contents of the file. Verify this by typing in the command

\texttt{"echo 5 6 7 8 > file1"}
To append (add on) to a file, use the double redirect arrow (">>").

"echo 9 10 11 12 >> file1"
Examining files

- less/more: open a file for reading in the terminal

For very long files, it can be annoying to print everything to the terminal.
The commands “less” and “more” open a text file instead of printing its contents (like “cat”). I prefer “less” but try both. To exit, press “q”.

Examining files

• less/more: open a file for reading in the terminal
Imagine that you want to run several commands in a row, like shown above. It would be a pain to type them all into the terminal, especially if you wanted to run the series more than once.

Writing shell scripts

• List commands in a text file, run all at once.
Imagine that you want to run several commands in a row, like shown above. It would be a pain to type them all into the terminal, especially if you wanted to run the series more than once.

We can do this by typing the commands into a text file and running the file as a "shell script."

Writing shell scripts

- List commands in a text file, run all at once.
Side note: there are a lot of different plaintext editors out there, and they all have different features. The default Mac editor (TextEdit) is terrible. Use something else like TextWrangler (the editor shown here).

Writing shell scripts

- List commands in a text file, run all at once.
The first line should always be “#!/bin/sh”. Don’t worry too much about what this means now. Just do it.

Writing shell scripts

• List commands in a text file, run all at once.
Writing shell scripts

- List commands in a text file, run all at once.

Type in the commands you want to run, in order.
Save the file into the folder where you are working. Call it “script.sh”

Writing shell scripts
- List commands in a text file, run all at once.
Writing shell scripts

• List commands in a text file, run all at once.

Return to your terminal, and type in “sh script.sh” (or whatever the name of your script is).
Writing shell scripts

- List commands in a text file, run all at once.

Making changes to a sequence of commands is much easier with scripting.
Writing shell scripts

- List commands in a text file, run all at once.

Making changes to a sequence of commands is much easier with scripting.

I wrote “17” instead of “16” like I wanted to.
Variables make scripts very powerful and flexible. Use them often.

Writing shell scripts
• List commands in a text file, run all at once.
Variables make scripts very powerful and flexible. Use them often.

Define variables with an "=" (no spaces allowed)

Writing shell scripts

• List commands in a text file, run all at once.
Writing shell scripts

- List commands in a text file, run all at once.

Variables make scripts very powerful and flexible. Use them often.

Call variables with a “$” sign in front of the name.
Writing shell scripts

- List commands in a text file, run all at once.
Writing shell scripts

• List commands in a text file, run all at once.

Try changing the variables, saving the file, and running again.
Writing shell scripts

• List commands in a text file, run all at once.

```bash
#!/bin/sh

echo 1 2 3 4
echo 5 6 7 8
echo 9 10 11 12
echo 13 14 15 16

echo Matt wants to go to bed
VAR1="Frodo"
VAR2="Mt. Doom"
echo $VAR1 wants to go to $VAR2
```

Try changing the variables, saving the file, and running again.
Tutorial Complete

• You now have enough basic knowledge to begin learning GMT through our GMT tutorials.
• Remember, this is just the beginning. There is so much more to the Unix operating system and the more you know, the more cool things you can accomplish.

Good luck!