UNIX COMMANDS

YOUR FIRST TERMINAL SESSION

Locate and open your terminal:

Linux Users

- Use the keyboard shortcut Ctrl-Alt-T

Mac Users

- Launch Finder
- Go to Applications > Utilities
- Click on Terminal
- Right-click on dock icon, click Options > Keep In Dock

Windows Users

- You'll use git-bash instead of the windows command prompt
- Right-click on desktop
- Select Git Bash
- If not already set, right-click taskbar icon and select 'pin this program to the taskbar'

READING THE PROMPT

The first stuff you see in your terminal is called the prompt.

It will include your username, where you are, and what machine this terminal is on.

By default:

- Mac:

```
machine_name:current_directory username$
```

- Linux:

```
username@machine_name:current_path$
```

- Window (git-bash):

```
username@machine_name current_path$
```

COMMAND: `pwd`

The pwd command shows the *present working directory*.

Mac

```
$ pwd
/Users/yourName
```

Linux

```
$ pwd
/home/yourName
```

Windows (git-bash)

```
$ pwd
/c/Users/Your Name
```

CONCEPT: THE PATH

In any computer system, a path represents a location in the filesystem.

Paths are like addresses, listing a location from the general to the specific.

A bit like addressing an envelope backwards:

USA
Seattle, WA 98105
123 Somestreet
Some Person

vs.

/home/cewing/projects/someproject

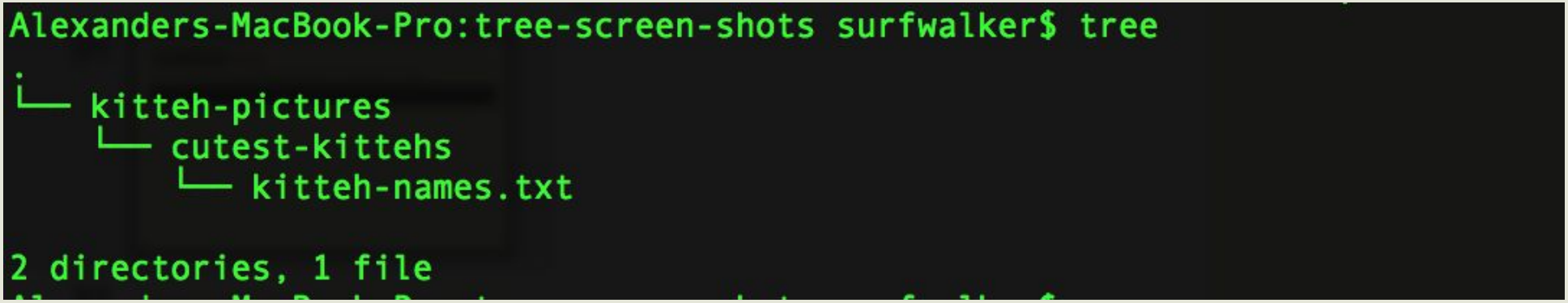
A path is absolute when it starts with /

A path is relative when it does not

COMMAND: `tree`

The `tree` command provides a visual representation of your current directory's structure.

```
$ tree
```



COMMAND: `ls`

The `ls` command shows you a listing of the contents of your present working directory.

```
$ ls
```

The behavior of this command can be altered by flags such as the `-la` which is a combination of the `-l` (long) and `-a` (all) flags.

`-l` provides more information about each directory and file.

`-a` reveals hidden files and folders .

```
$ ls -la
```

COMMAND: `cd`

The cd command allows you to *change directories*.

`cd` entered by itself it will take you to your *home directory*.

```
$ cd
```

Or it can take a path as an argument. The path can be either *absolute* or *relative.*

An absolute path always begins with a "`/`" which represents the root directory of your computer.

```
$ cd /Users/YourName/somewhere-else
```

An example of an relative path would be:

```
$ cd somewhere-else
```

COMMAND: `cd ..`

To move up a level from your present working directory simply enter `cd ..` where `..` is an alias for the parent directory.

```
$ cd ..
```

You can even chain them together, providing relative paths that go up more than one level:

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

And you can combine these with directory names to go back down into a different branch of your filesystem:

```
$ cd ../somewhere-else
```

COMMAND: `mkdir`

The terminal equivalent of new folder is `mkdir` which stands for *make directory*.

This command take an argument which is the name of the directory you want to create.

```
$ mkdir kitteh_pictures
```

To create nested directories (a new directory within a new directory) you can use the `-p` flag.

```
$ mkdir -p kitteh_pictures/cutest_kittehs
```

COMMAND: `touch`

The terminal equivalent of new file is `touch`.

This command take an argument which is the name of the directory you want to create.

```
$ touch kitteh_names.txt
```

COMMAND: `atom`

The `subl` command opens your Atom text editor. This is not a command that is native to the terminal.

With no argument it simply opens the program:

```
$ atom
```

You can provide a file as an argument:

```
$ atom kitteh_names.txt
```

You can provide a directory as an argument:

```
$ atom kitteh_pictures/
```

COMMAND: `mv`

The mv command allows moving files from place to place in a file system.

It expects two paths as arguments which are the file you want to move and the directory where you want to move it.

```
$ mv kitteh_names.txt kitteh_ideas/
```

Another use for the `mv` command is *renaming*. If an explicit filename is provided at the end of the second argument the targeted file will be moved and renamed. You can use this to change the name of a file and not move it.

```
$ mv kitteh_names.txt terrible_kitteh_names.txt
```

Trouble Spot

Depending on how you've named things (or due to the default names inherent in your operating system)your home directory file path may contain spaces which when entered as a command will result in an error:

```
$ cd /c/Users/Your Name
sh.exe": cd: /c/Users/Your: no such file or directory
```

The problem is the space between my first and last names

The command line expects paths to be a single continuous string of characters

Spaces are used to delimit one element of the command line from the next

You can fix this by escaping it with the `\` character:

```
$ cd /c/Users/Your\ Name
```

CONCEPT: Naming Conventions

Avoid spaces in the names you give to files and directories.

Use dashes and underscores to create visual separation between words in names.

Prefer lower-case letters in naming files and directories.

This is good:

- my_project_file.html

This is bad:

- My Project File.html

COMMAND: `rm`

The `rm` command is the equivalent of moving something to the trash with one important distinction: the file is completely deleted from the system and is no longer recoverable. Always take a moment to be sure before you execute an `rm` command.

```
$ rm terrible_kitteh_names.txt
```

BEWARE!!!
There `-rf` flag is commonly used in conjunction with the `rm` command. This allows you to delete directories and all files contained therein. The `f` stands for *force* which means your system will not provide any warning or request for confirmation. It is possible to delete your entire everything this way. Be extremely cautious with this command. Sam refers to this command as, “Remove with Fire”.

COMMAND: `cp`

The `cp` command allows you to copy a file and place that copy in a different location.

It takes two arguments with the first being the file to be copied and the second argument being the destination.

```
$ cp kitteh_names.txt ~/Desktop
```

This will make a new copy of the `kitteh_names.txt` file on your Desktop.

COMMAND: `history`

The history command allows you to review and revise the history of actions you've taken in a shell

```
$ history
```

COMMAND: `man`

The `man` command provides access to the built-in manual for all unix commands

Providing the command with the name of some other command will print detailed information about how that command may be used

Often these manual pages include useful examples for common and advanced usage patterns

```
$ man ls
```

For Windows users...Google is your friend!

COMMAND: `date/cal`

The `date` command provides the current date and time to the second.

```
Alexanders-MacBook-Pro:~ surfwalker$ date
Fri Sep 18 14:37:16 PDT 2015
Alexanders-MacBook-Pro:~ surfwalker$
```

The `cal` command provides a visual representation of the current month. You can also provide month and year arguments to see any future or previous months.

```
Alexanders-MacBook-Pro:~ surfwalker$ cal 7 1776
     July 1776
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

HELPFUL TIPS

Up and *Down* arrows on your keypad allow you to scroll through previously entered commands to save you from retyping them.

The *tab* key will auto complete the file, directory or command that you a currently typing.

Just as `..` is an alias for the parent directory `.` is an alias for the current directory. This is useful specifically for opening files with Sublime when you want to open all the files in a directory e.g. `subl .`

You can use the .. symbol as an element in a path (absolute or relative) as a shortcut for "one level up"

You can use the . symbol as an element in a path as a shortcut for "right here"

You can use the ~ (tilde) as a shortcut for the absolute path of your home directory

You can use the cd command without an argument to return to your home directory immediately from anywhere

ANY
QUESTIONS?

REVIEW

We've added the following unix commands to our repertoire:

COMMAND	
history	interact with your command line history
less	read large text inputs in a controlled fashion
mv	move files from one place to another, or rename them, or both
touch	create a new file, or update the modified date for an existing

REVIEW

COMMAND	
cp	copy the contents of a file or directory to a new location
rm	remove a file from the filesystem entirely
rmdir	remove a directory from the filesystem if it is empty