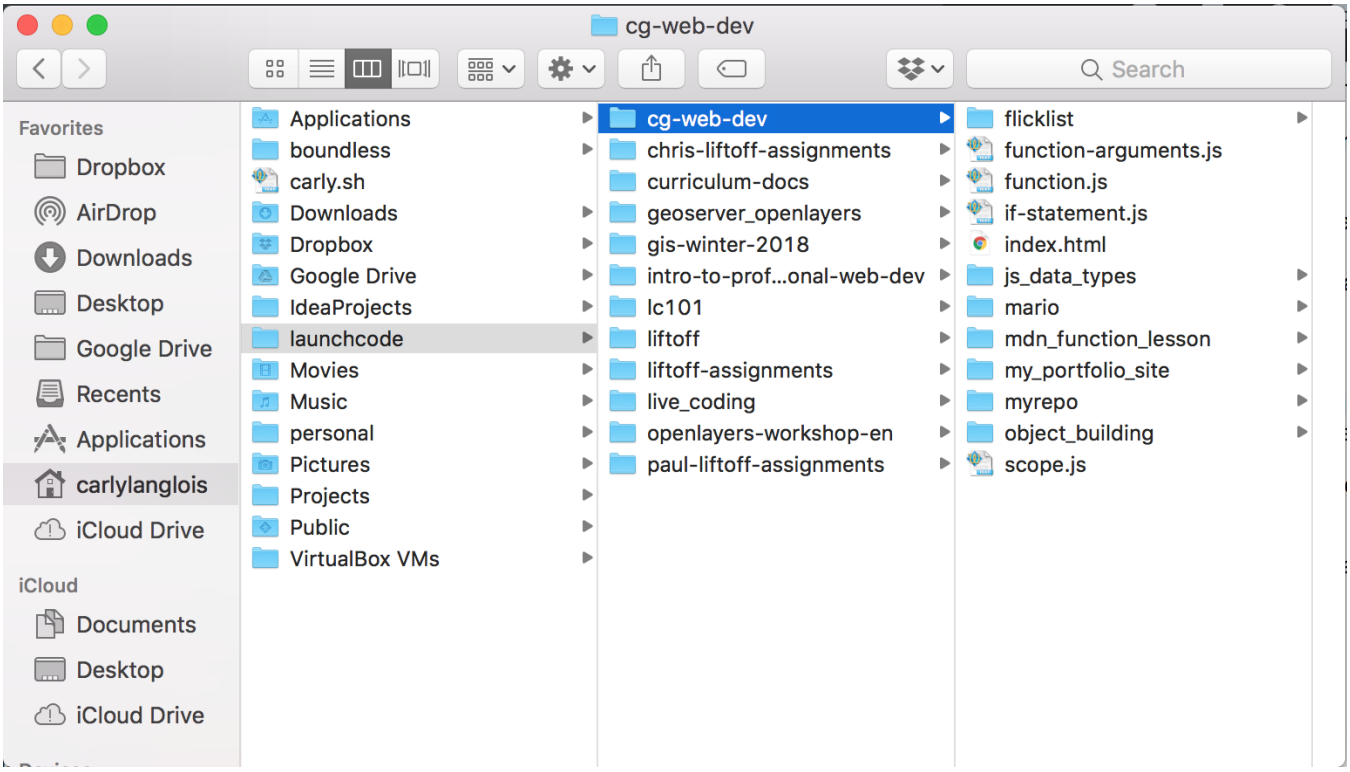
19.1. What is a terminal?

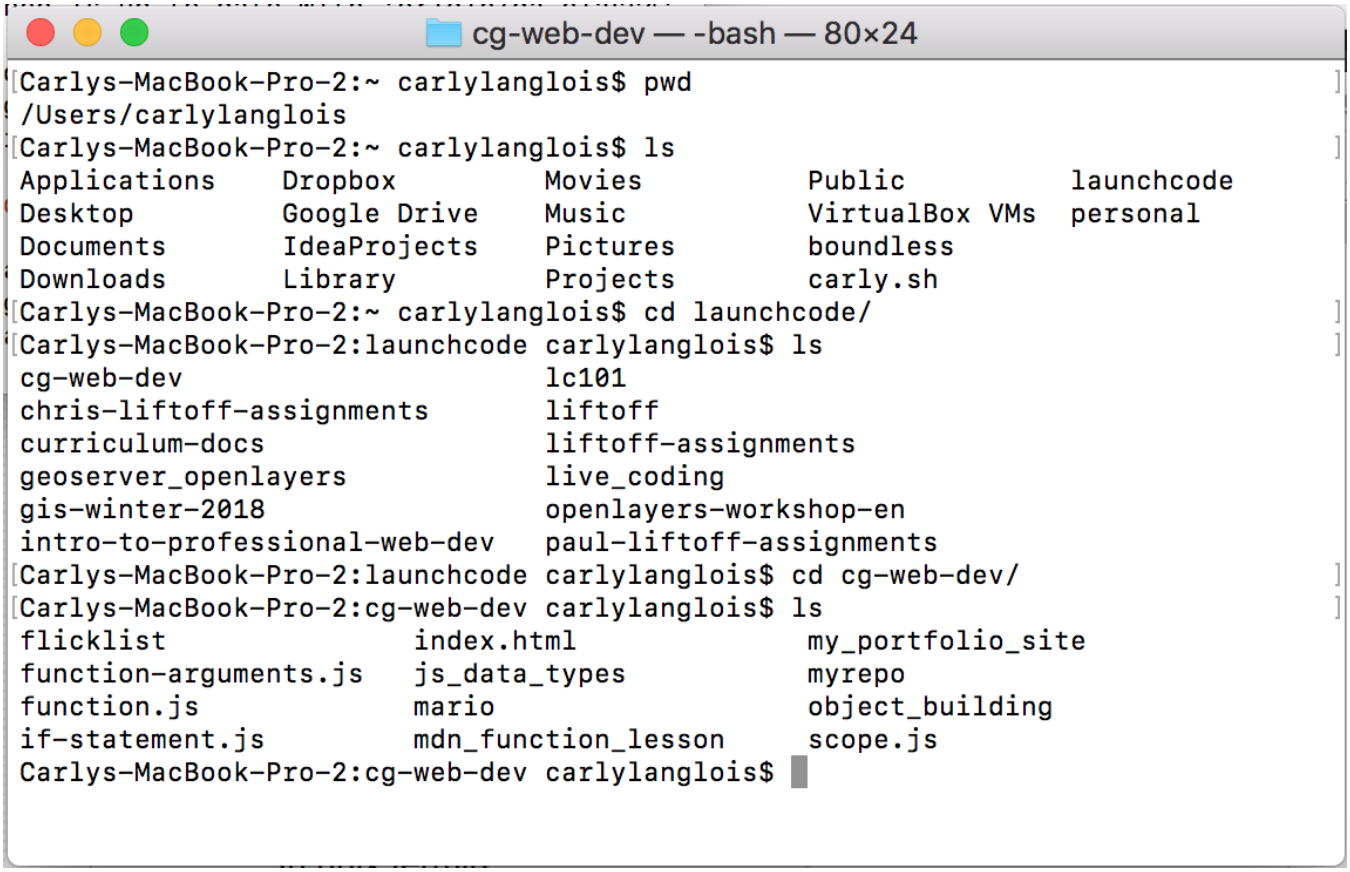
19.1.1. GUIs and CLIs

Most of the time when we use our computers, we do so through a **graphical user interface**, or **GUI** for short. A GUI is a system designed with icons and visual representations of the machine's file systems.



A GUI with file icons and columns representing folder structure.

Programmers often use another kind of interface, called the **command line**. A **CLI**, or command line interface, uses textual commands, rather than dragging and dropping icons, to give the computer instructions.



A CLI with commands navigating the same file paths as the GUI above.

The application responsible for running a CLI is called a **terminal** and the program interpreting the commands is called the **shell**.

Note

The terms “command line”, “terminal”, and “shell” are often used interchangeably.

19.1.2. Why use the terminal?

Both of the images above represent the same file structure. While the GUI may now appear more user-friendly, as you grow more familiar with the commands available, you’ll find there can be advantages to using the terminal.

In the terminal, you will be able to:

- quickly move throughout your computer’s file structure
- make new files and directories
- remove items from folders
- install software
- open programs
- run programs directly

19.2. Filesystem and Paths

A **filesystem** is a structure for the computer to store the files and folders that make up the data of the operating system.

Inside a filesystem, folders are referred to as **directories**. Your **root directory** is your Home folder or C drive. When you open a new terminal window to work in, it opens to your root directory. The root directory is the **parent directory** for the folders stored inside of it.

Example

Oftentimes, there is a `Desktop` folder inside the root directory. If there is a folder on your Desktop called "LC101_Homework", then the parent directory of `LC101_Homework` is `Desktop`. The parent directory of `Desktop` is `LC101_Homework`.
A **path** for files and folders is the list of parent directories that the computer must go through to find that particular item.

Computers have two different types of paths: absolute and relative. The **absolute path** is the path to a file from the root directory. The **relative path** is the path to a file from the current directory. When working with a relative path, you may find yourself wanting to go up into a parent directory to find a file in a different child directory. In order to do so, you can use `..` in the file path to tell the computer to go up to the parent directory.

Example

We have a file inside our `LC101_Homework` directory from the above example. We named that file `homework.js`. The absolute path for `homework.js` is `/Users/LaunchCodeStudent/Desktop/LC101_Homework` for Mac users and `C:\windows\Desktop\LC101_Homework` for Windows users. If the current directory is `Desktop`, then the relative path for `homework.js` is `/LC101_Homework` for Mac users and `\LC101_Homework` for Windows users.

If `homework.js` were in a different directory called `CoderGirl_Homework`, which is inside the `Desktop` directory, and the current directory was `LC101_Homework`, then we would use the `..` syntax in our relative path. The relative path would then be `/../CoderGirl_Homework` for Mac users and `\..\CoderGirl_Homework` for Windows users.
Many programmers use paths to navigate through the filesystem in the terminal. We will discuss the commands to do so in the next section.

19.3. How to Do Stuff in the Terminal

19.3.1. Navigating the Terminal Window

Moving from a GUI to a CLI can be difficult when we are so used to dragging our files from one folder to another. One of the difficulties is simply figuring out where we are in the filesystem! Here are some key indicators that the terminal gives us to show where we are:

```
LaunchCode-Super-Computer:~ lcstaffmember$
```

This line is called the **prompt**. The prompt lets us know that the terminal is ready to accept commands. `LaunchCode-Super-Computer` is the name of the computer. The `~` tells us we are currently in the Home directory. The Home directory is the folder that contains everything in the computer. `lcstaffmember` is the username of the person who has logged onto the terminal. We will be typing all of our commands after the `$`.

As we navigate through our filesystem, the terminal will rarely output a line to let us know that the change has occurred. We have to keep our eye out on our prompt as we enter our commands. The name of the computer and the username will not change, however, the space where the `~` is, will. That indicates our current directory.

19.3.2. Basic Commands

There are many commands you can use in the terminal to move through the filesystem of your computer and projects.

Basic Terminal Commands

Command	Result
`ls`	Lists all files and folders in the current directory.
`cd <new-directory>`	Navigates from the current directory to `new-directory`.
`pwd`	Prints the path of the current directory.
`mkdir <new-folder>`	Creates `new-folder` inside the current directory.
`touch <new-file>`	Creates a file called `new-file` in the current directory.
`rm <old-file>`	Removes `old-file` from the current directory.

Basic Terminal Commands

Command	Result
`man <command>`	Prints to the screen the manual pages for the command. This includes the proper syntax and a description of how that command works.
`clear`	Empties the terminal window of previous commands and output.
`cp <source-path> <target-path>`	Copies the file or directory at `source-path` and puts it in the `target-path`.
`mv <source-path> <target-path>`	Moves the file or directory at `source-path` from its current location to `target-path`.

Note

`rm` will permanently remove items from the computer and cannot be undone.

Beyond these basic commands, there are some shortcuts if you don't want to type out the full name of a directory or simply can't remember it.

Directory Shortcuts

Shortcut	Where it goes
`~`	The Home directory
`.`	The current directory
`..`	The parent directory of the current directory

For an in-depth tutorial of how to use a CLI to move through your daily life, refer to the terminal commands tutorial.

19.3.3. Check Your Understanding

Question

What line in a CLI indicates that the terminal is ready?

a. prompt
b. command
c. shell
d. There isn’t a line that does that.

Question

Which shortcut takes you to the parent directory?

a. `.`
b. `~`
c. `..`

19.4. Running Programs in the Terminal

Quickly navigating through our filesystems is just one benefit of using the terminal for programmers. We can also quickly run our code inside of the terminal to see the outputs.

The commands used to run a program in the terminal vary widely based on type of program you want to run. However, no matter what language you are coding in, the documentation will include, in some format, ways to run the program in the terminal.

Example

So far, in repl.it, we have been running our programs by hitting the "Run" button. If we type `node <file-name>` into our terminal, we would be doing the same thing as the "Run" button!
Let's say there is an error in our program like an infinite loop. How then do we get it to stop running so we can go back and fix our code?

In many cases, typing *ctrl+c* into the terminal will stop a process that is currently running. However, if that doesn't work, the `exit` command can also stop a currently running process.

19.5. Exercises: Terminal

1. If you haven't done so already, set up your command line environment with instructions from the Setting Up Your Terminal appendix.
2. Using your terminal, navigate to your Home directory using `cd ~`.
3. Use `ls` to view the contents of your Home directory.
4. Use `cd` to move into your Desktop directory. For most, the command to do this is `cd Desktop/` since the Desktop is most often a child of the Home directory.
5. In the terminal, use `mkdir` to create a folder on the Desktop called 'my_first_directory'. Look on your Desktop. Do you see it?
6. Use `cd my_first_directory/` to move inside that directory.
7. `pwd` to check your location.
8. There, make a file called 'my_first_file.txt' with `touch my_first_file`.
9. Open the file and write yourself a message!
10. Back in the terminal, list the contents of your current directory from the terminal with `ls`.
11. Make a copy of your 'my_first_file.txt' from it's current spot to directly on the Desktop with `cp my_first_file.txt ../my_first_copy.txt`.
12. Move back out to your Desktop directory from the terminal with `cd ..`.
13. Use `ls` in the terminal to verify your 'my_first_copy.txt' on your Desktop. Open it up. Is it the same as your first file?
14. Move your copied file into your 'my_first_directory' with `mv my_first_copy.txt my_first_directory/`.
15. Use `ls` to see that the the copied file is no longer on your Desktop.
16. Type `cd my_first_directory/`, followed by `ls` to confirm that your copy has been moved into 'my_first_directory'.
17. `cd ..` to get back out to your Desktop.
18. Type `rm -r my_first_directory/` and do a visual check, as well as `ls` on your terminal, to verify that the directory has been removed.