1.1.3 Get Started with the Command Line

In the old days, entering text-based commands on a black screen (known as the **command-line interface**, or **CLI**) was the only way to interact with a computer. But now that graphical operating systems like Windows and macOS exist, most people never have to even look at a CLI.

For developers, though, the CLI (also called the **terminal**) is still frequently used for many tasks. For now, we'll use it to set up our project.

But first, let's figure out how to access it!

Access the Command Line

Follow the directions for accessing the command line depending on which operating system you're using.

On macOS

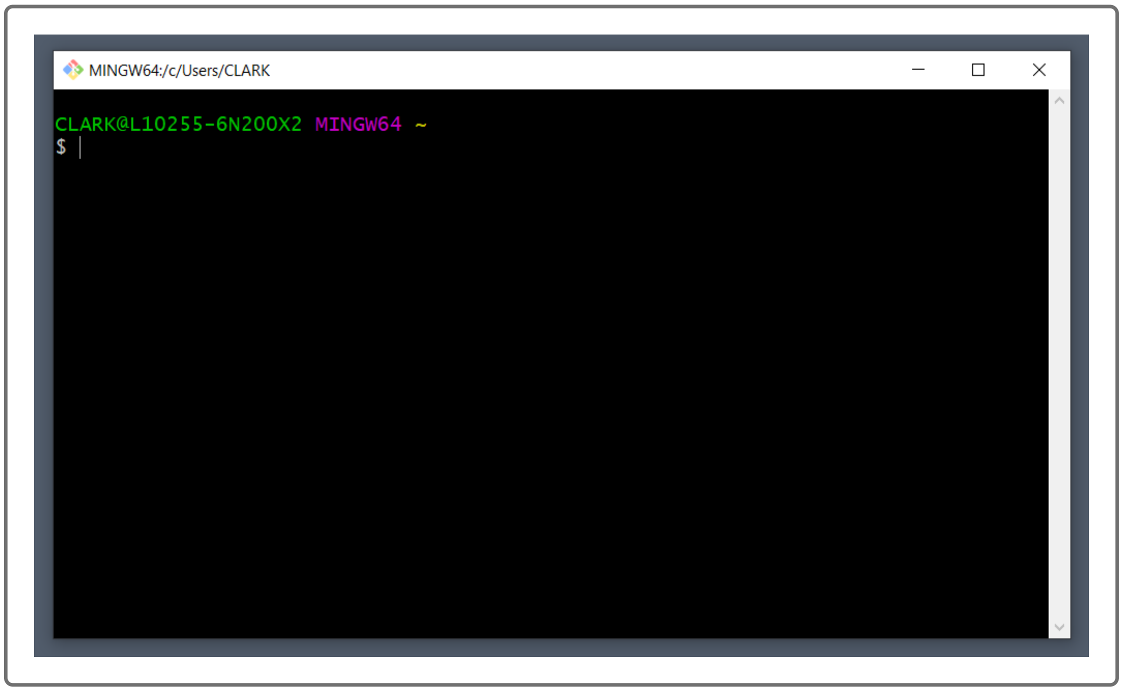
macOS users, you already have a built-in Terminal app, which you can launch by searching for "terminal" in the Spotlight Search (click the 🔍 icon in the top-right corner or use the keyboard shortcut "Command + Spacebar").

On Windows

Windows users, you installed Git Bash as part of the prework. Open Git Bash by using the search box in the bottom-left corner (or by using the keyboard shortcut "Windows key + S") to search for "git bash".

Use the Command Line

When you open the command line, you should see a window that looks like this image:



```
CLARK@L10255-6N200X2 MINGW64 ~
$ 
```

In this window, type `pwd` and press Enter. The `pwd` command means "print working directory" ("print" in this context means "display to the screen") and lets you know which directory (aka folder) the terminal is currently in. Typing `pwd` is like asking "Where am I?"

IMPORTANT

> To succeed in this boot camp, type out every line of code instead of copying and pasting. This will quickly build the muscle memory you

> need to become an expert coder.

The terminal will display something like `/c/Users/<username>` on Windows or `/Users/<username>` on macOS. This is because when you first open Terminal or Git Bash, you start in your user directory.

Now let's look at the contents of this directory using another command: `ls`.

The `ls` command lists the files and folders within the current directory. It's like asking "What's here?"

Some folders you might see in your user directory are `Documents`, `Pictures`, `Music`, and `Desktop`.

Fun fact: the graphical "desktop" you see when your computer starts is this `Desktop` folder in your user directory! Let's "go" there now in the command line by typing the following commands, pressing Enter after each line:

```
cd Desktop
pwd
```

PAUSE

What do you think `cd` stands for?

Show Answer

We can use the `cd` command to move in and out of directories. Try the following commands, noting what `pwd` prints each time:

```
cd ..
pwd
cd Desktop
pwd
```

So `cd ..` took us back a level (or "up" a directory). Good to know!

Now that we're in the `Desktop` directory again, run the command `ls`. Do you recognize the listed files and folders as things that are already on your desktop? Let's add to the clutter:

```
mkdir projects
ls
```

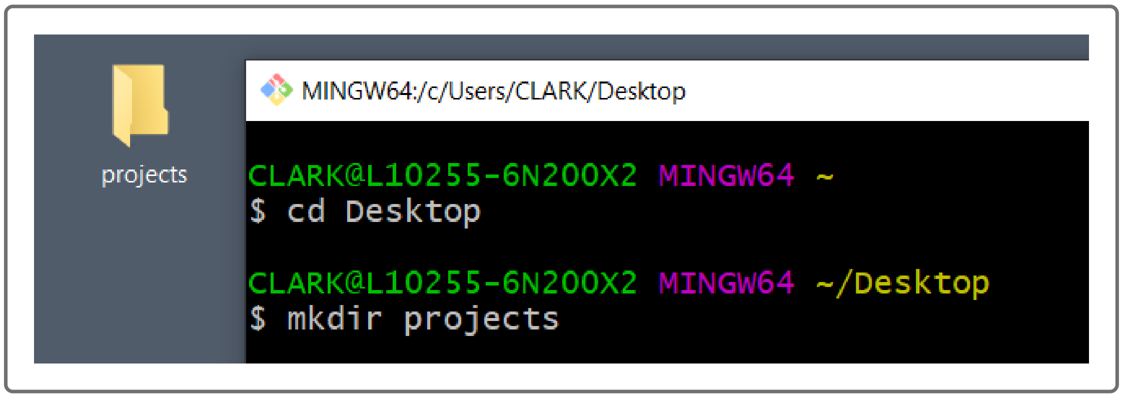
The `mkdir` command allows us to create new folders (or literally, "make directories"). In this case, we created a folder called `projects`. The second command, `ls`, should let us verify that the folder was created.

IMPORTANT

> You can clear the contents you just typed into the command line by simply typing `clear` then pressing the Enter key. It won't undo your work, but it'll keep your command line interface from getting cluttered and hard to read!

Now, try something neat: minimize the command line for a second and look at your computer's desktop again. Do you see the `projects` folder we just made (on the left in the picture below)? What you do on the command line has direct consequences on your computer!

The result looks something like this image (this will vary depending on your operating system!):



```
CLARK@L10255-6N200X2 MINGW64 ~
$ cd Desktop

CLARK@L10255-6N200X2 MINGW64 ~/Desktop
$ mkdir projects
```

This `projects` folder is where we'll keep all of our work throughout the boot camp.

From the command line, let's step inside the `projects` folder and create another folder for our Run Buddy project:

```
cd projects
mkdir run-buddy
cd run-buddy
pwd
```

If you were to run the `ls` command while in the `run-buddy` directory, nothing would happen because we don't have any files in here to list.

Let's go ahead and make a few files using the `touch` command:

```
touch hello.txt
touch index.html
ls
```

IMPORTANT

> Notice that we're using lowercase for our filenames. A file named `index.html` would be an entirely different file than one named `Index.html`.

We use the `touch` command to create files. Here, we've just made two new files: a text file (`hello.txt`) and an HTML file (`index.html`).

On second thought, we don't really need that text file. So let's (permanently) remove it by using the `rm` command:

```
rm hello.txt
ls
```

Everything we've done up to this point could also be accomplished with mouse clicks and menus, but once you're used to it, you'll find that the command line allows you to work much faster. Plus it'll make you feel really cool.

The following list recaps the commands you just learned:

- `pwd`: Print working directory
- `ls`: List files in directory
- `cd <directory-name>`: Change to directory
- `cd ..`: Go up/back a directory
- `mkdir <directory-name>`: Make a new directory
- `touch <file-name>`: Make a new file
- `rm <file-name>`: Remove a file

DEEP DIVE