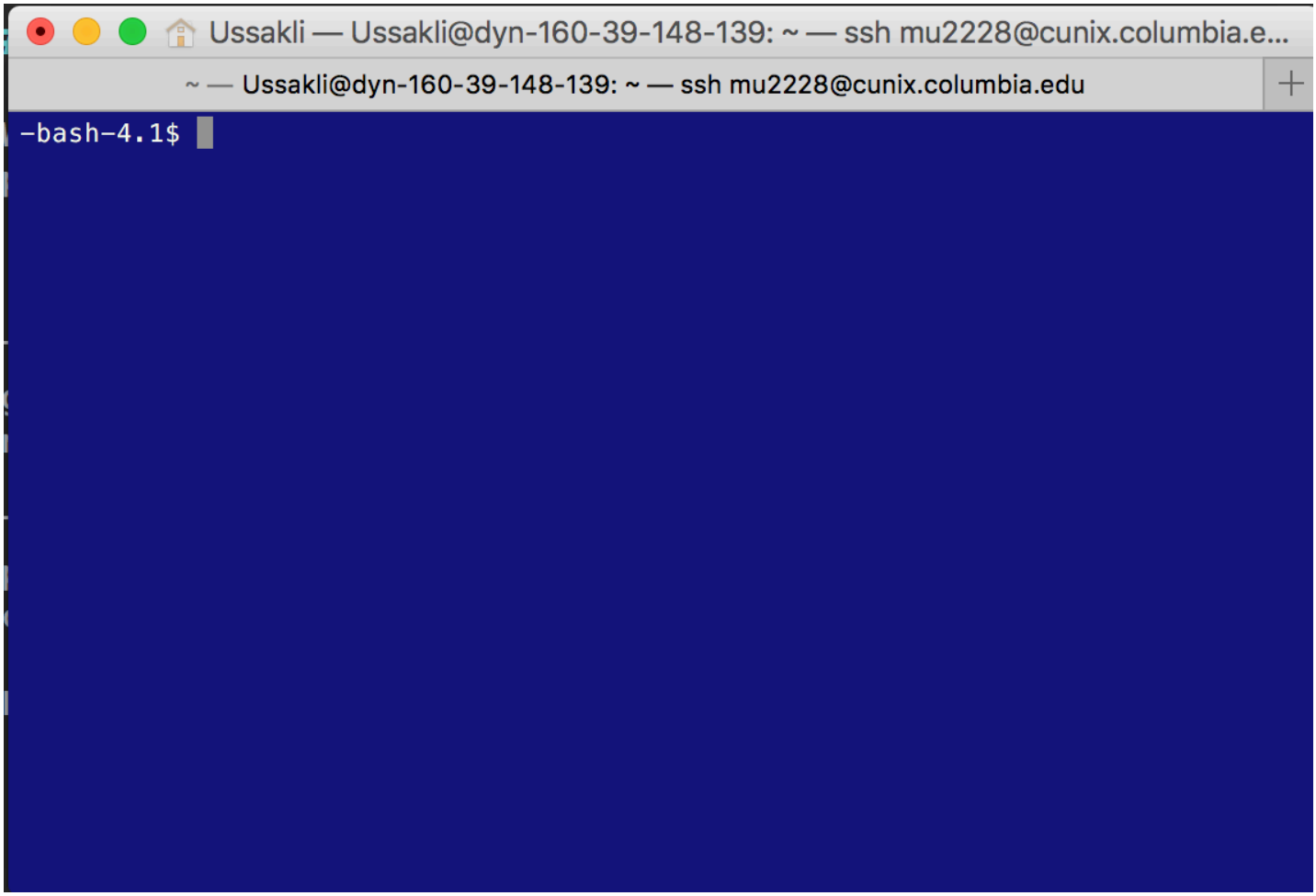
Written by Mert Ussakli.

Compiling Java Code in Cunix / Command Line

Knowing the basics of command line / terminal and being able to compile and run your code is pretty important. Not only will it make your life easier in 3134, but it is also an incredibly important skill to have if you are planning to move on and take more Computer Science classes at Columbia. So let's hop right into the basics.

The Basics

When you load up your terminal, or log in to your Cunix machine, you probably see something like this:



To be able to run and compile your Java code, you need to be able to go to the directory in which your code lives. How do we do that? We need to learn a bunch of commands.

The first one is `ls`. If you type `ls` into the command line and then press enter, it will give you a list of all the files that live in the current directory you live in.

Let's type `ls` in our CUNIX machine and press enter.

```
-bash-4.1$ ls
fred  fred.txt  javaCode  public_html  wkcr  wkcr_arts  wkcr_ial  wkcr_new
-bash-4.1$
```

Here, we can see a list of all the files in our directory. Now, we have to change our current directory to the one in which our Java code lives in. To do this we need to use the `cd` command. Type `cd` followed by the name of the directory you want to go to.

In this example, we are going to type `cd javaCode`. Your code may be living a couple directories down. In that case, you need to keep using `ls` and `cd` until you find your code.

```
-bash-4.1$ ls
fred  fred.txt  javaCode  public_html  wkcr  wkcr_arts  wkcr_ial  wkcr_new
-bash-4.1$ cd javaCode/
-bash-4.1$
```

Awesome. So which files are in this directory that we just moved in to? Let's `ls`

```
-bash-4.1$ ls
fred  fred.txt  javaCode  public_html  wkcr  wkcr_arts  wkcr_ial  wkcr_new
-bash-4.1$ cd javaCode/
-bash-4.1$ ls
HelloWorld.java
-bash-4.1$
```

I see a little guy named `HelloWorld.java`. I think we can compile him now, since we are in the directory where it's located.

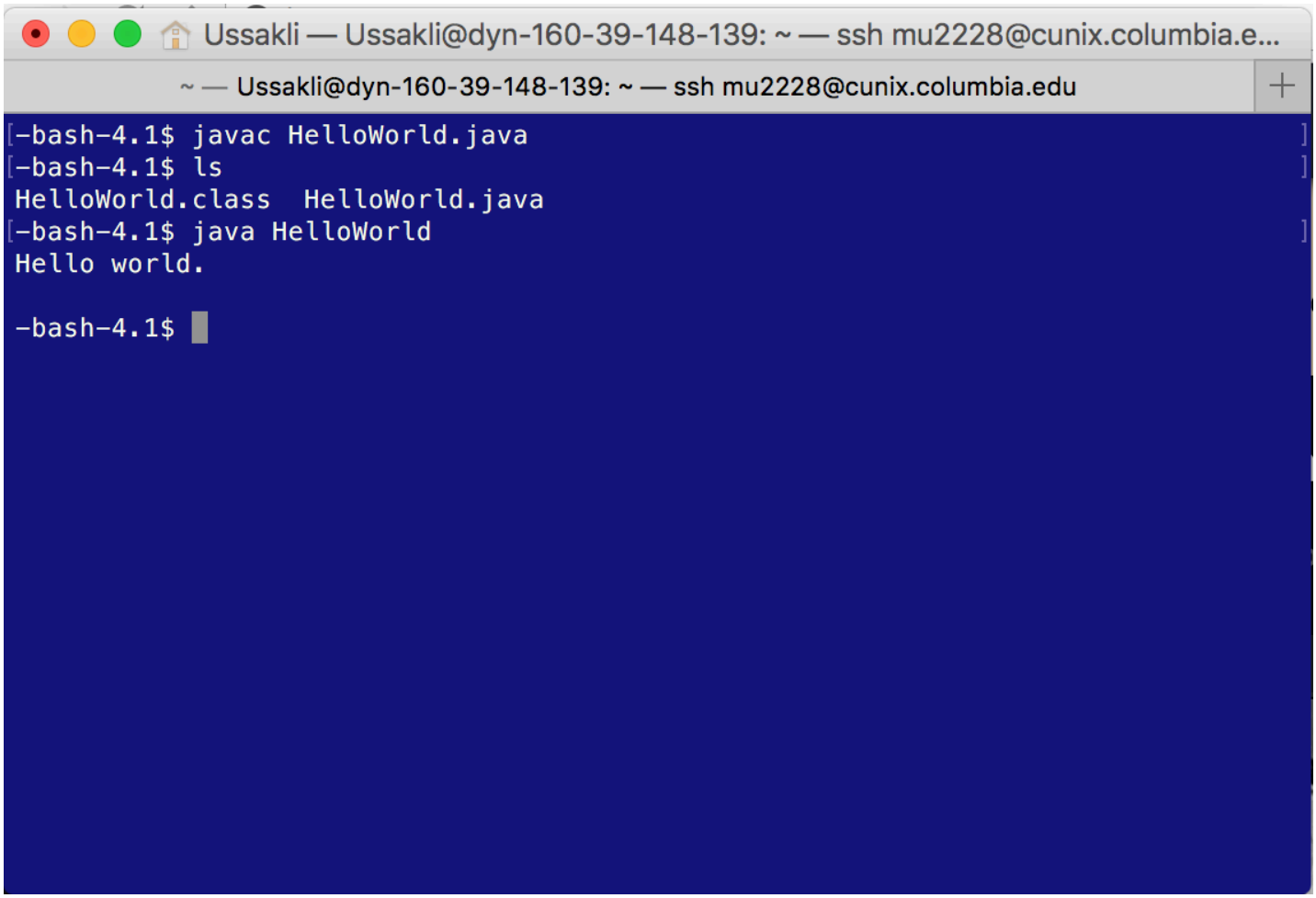
To compile in the terminal, you need to type `javac HelloWorld.java`. If you have multiple Java files linked to each other, you still need to call `javac NameOfProgram.java` on the file where your main method is located. The Java compiler is smart enough to figure out all the dependencies for you.

Let's compile our code by running `javac HelloWorld.java`

```
-bash-4.1$ javac HelloWorld.java
-bash-4.1$ ls
HelloWorld.class  HelloWorld.java
-bash-4.1$
```

Now, after we `ls` we can see that a new file called `HelloWorld.class` was created. This is a binary file containing your Java executable. To run this, we should type `java HelloWorld` on the terminal. Remember, `java HelloWorld` and NOT `java HelloWorld.class` or `java HelloWorld.java`.

Let's run `java HelloWorld`



Look! It's one of those programs that print "Hello world". We have successfully run and compiled our code.

Command Line Arguments

Passing in command line arguments is a lot easier than messing with those Eclipse preferences and this is something you should probably get used to. They are called "command line arguments" for a reason.

Let's consider this following piece of code:

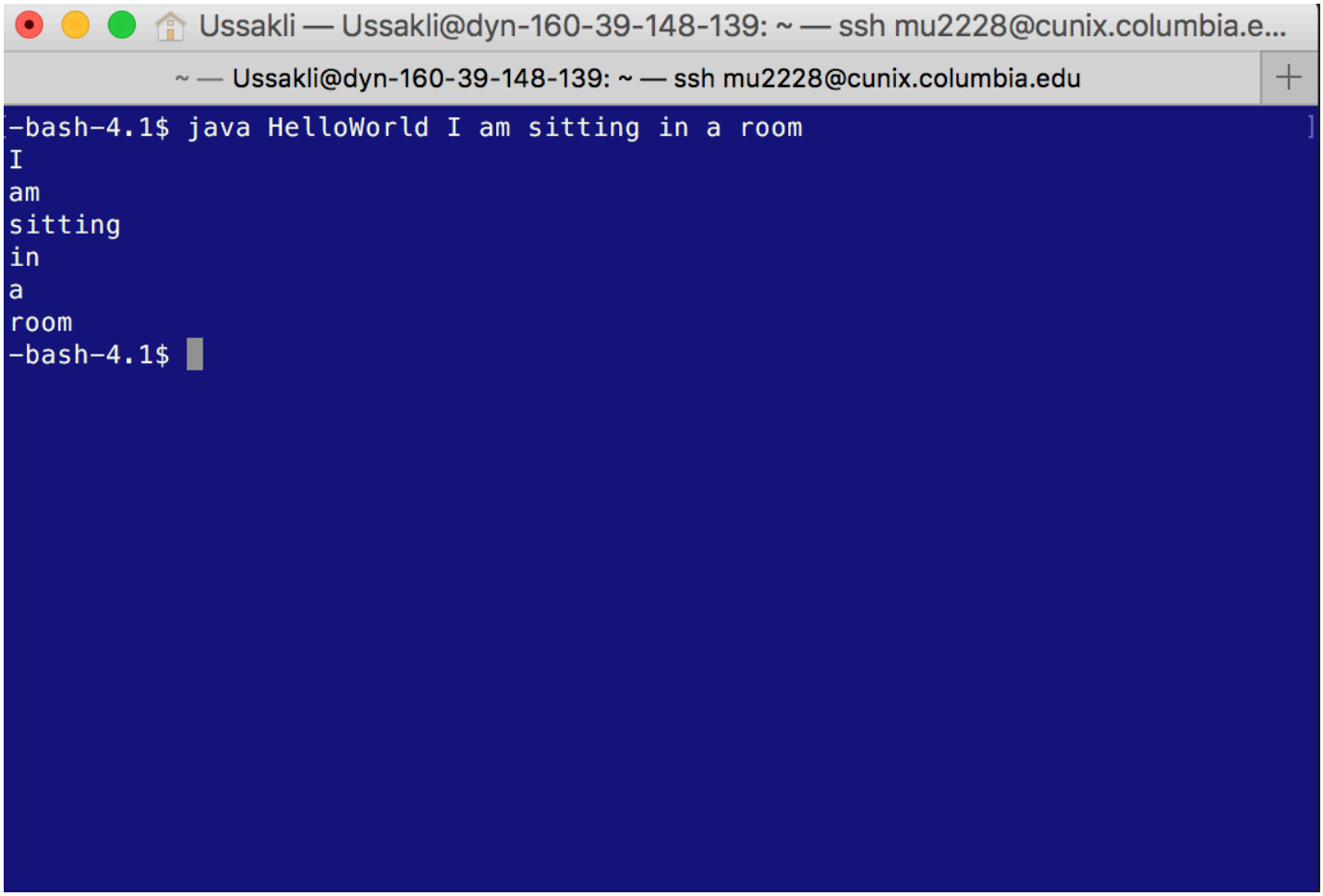
```
public class HelloWorld {
    public static void main(String[] args)
    {
        for(String s : args)
            System.out.println(s);
    }
}
```

This is a program that prints out the name of all the command line arguments we pass in.

To do that, we compile our code in the way we just did, and then run it in the following format:

```
java HelloWorld argument1 argument2 argument3 ...
```

So `argument1` becomes the String at `args[0]` in your main method. Let's try this and see what our program prints.



Every single argument we passed in (I, am, sitting, in, a, room) got printed on a new line. If we wanted to just pass a single argument, let's say the name of a text file, all we needed to do would be:

```
java HelloWorld someTextFile.txt
```

So now you know how to pass in command line arguments on the command line.

A Few Things To Make Your Life Easier

I will show you to more things that are super useful on the command line. One is called **tab complete**.

Suppose we want to go to this directory called `floccinaucinihilipilification`.

```
-bash-4.1$ ls
floccinaucinihilipilification  fred.txt  public_html  wkcr_arts  wkcr_new
fred                           javaCode  wkcr         wkcr_ial
-bash-4.1$
```

Isn't it tedious to type the entire word to go to that directory? Instead of doing that, we can start typing the name of the directory and then hit the **tab** button. What this will do is to complete the name of the directory for us.

In our case, we will type something like `cd floc` and then press tab. Here is how the tab complete will look like after:

```
-bash-4.1$ ls
floccinaucinihilipilification  fred.txt  public_html  wkcr_arts  wkcr_new
fred                           javaCode  wkcr         wkcr_ial
-bash-4.1$ cd floccinaucinihilipilification/
```

I swear I did not write that word out. Well OK maybe on the first time when I created the directory.

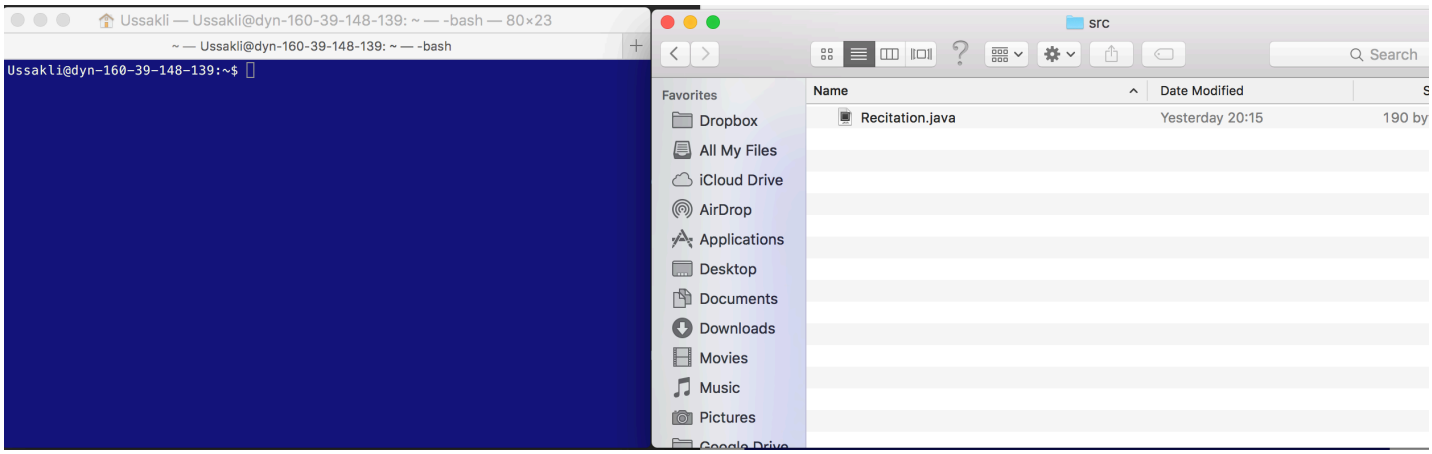
cd'ing into your directory faster

This is something that will make your life super easy if you are a Mac user.

A reason why a lot of people don't use the command line is because their code is 10 directories deep from their home directory and they have to type `cd` and `ls` 10 times to get there.

But there's a shortcut.

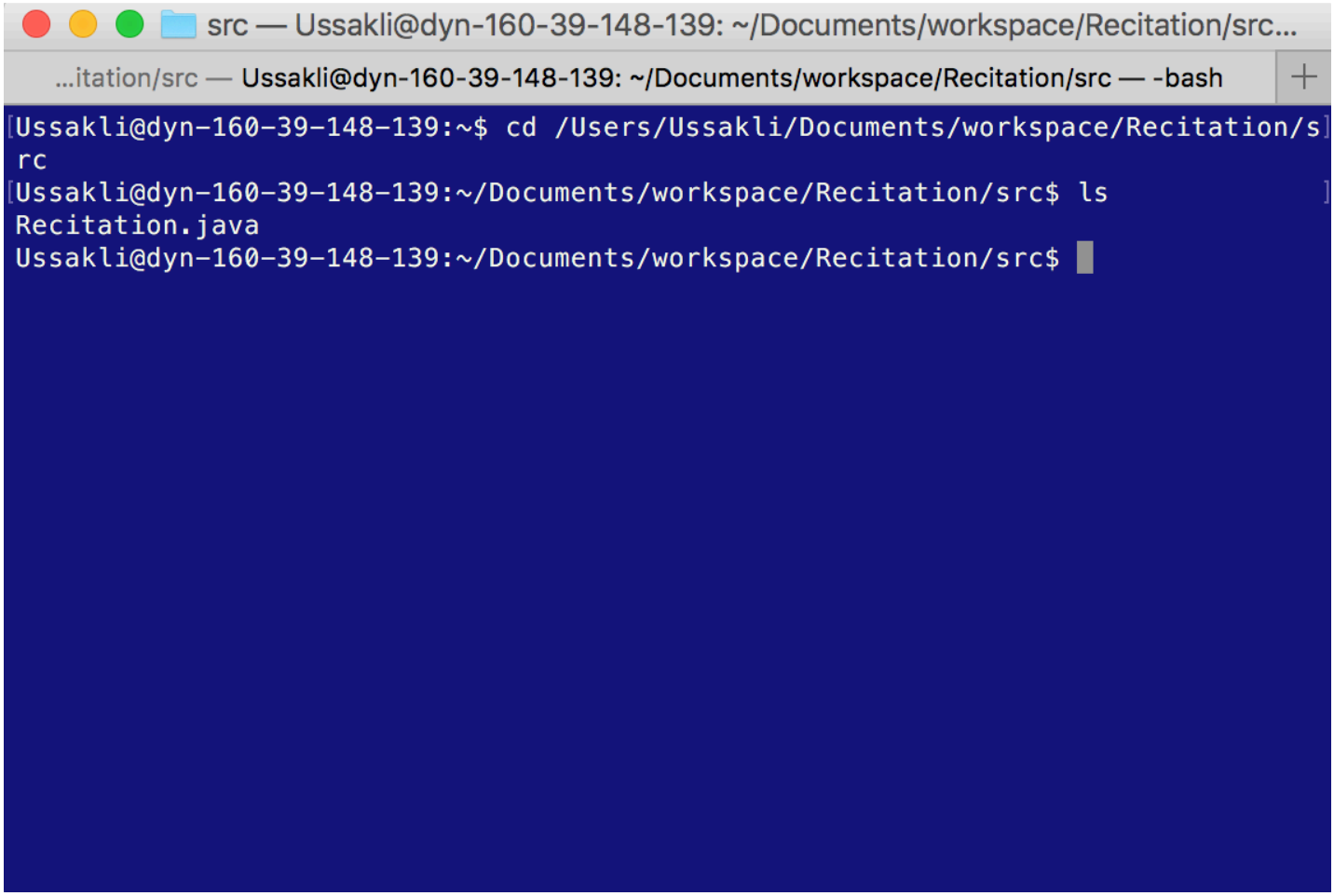
Open up a terminal and also open up where your code lives in a Finder window.



Ok so how do we get there to compile and run the code?

First, type `cd` on your terminal and leave a space. Don't enter anything.

Do you see the blue folder icon right next to the word **src** on top of that finder window? You want to click on that icon, drag it and drop it on your Terminal window. The result will look something like this.



Wow! I typed `ls` after and can see that I am in the directory I wanted to be. Wasn't that easy? Now I can compile & run my code as I wish.