


The Bash Shell

This lesson will teach you the basics of using **Bash**, a terminal for UNIX systems (Mac and Linux). If you are using Windows, it is highly recommended that you download and install Bash for windows, as usage of Windows' command prompt will not be shown here. If you are on Mac, you may open a Bash terminal by pressing **⌘** + , typing "Terminal" and then pressing enter. If you are on Linux, I'm just going to assume you already know how to open a terminal.

Navigation

Now that you have launch your Bash terminal, we will see how you may use it to navigate your **file system**, that is, your computer's **files** and **directories** (which are often called "folders"). First, type **pwd**, which stands for "print working directory", and press enter (from now on, I will use "enter" as verb meaning to type a command and then press enter). You will probably see something similar to `/Users/MyUserName:` this is your **home directory**. You have probably navigated your file system on your operating system before through some visual interface, but you can also do it through Bash. For example, enter

```
ls
```

and you should see a list of all the files and directories in your home directory. You likely see a directory called "Documents" or something similar. To enter it or another directory, enter

```
cd <directory name>
```

which in this case is

```
cd Documents
```

to navigate to it. Note that I put `<directory name>` as a placeholder for the directory name: a pair of angle brackets `<>` is commonly used to denote a placeholder for an input that could vary. **cd** stands for "Change Directory", and **cd**, **ls**, and **pwd** are examples of Bash **commands**. The directory name appearing after **cd** is called an **argument**. Commands may have 0, 1, or several arguments. **ls** may take an argument as well: **ls <directory name>** will list the contents of the given directory instead of the current directory.

Now that you've navigated to a directory, you may want to navigate to the previous directory: to do so, enter

```
cd ..
```

In Bash, `..` is a special name for a directory's **parent directory**. Finally, you may want to eventually return to your home directory in your Bash session. To do so, simply enter

```
cd
```

with no arguments, and you will be returned to your home directory.

Brief Overview of Common Bash Commands

This course is about C, so I will not spend too much time going into detail on how to use Bash. However, some of the basic commands may be useful to you while you are leveraging Bash to compile and run C programs, or you may just find it interesting. Note that I will neither give a full or details version of the commands and their arguments, just a basic description.

Command	Description
<code>cat <file name></code>	Output the contents of the given file
<code>cd <directory name></code>	Navigates to the given directory. If the directory is omitted, navigates to the home directory instead.
<code>echo <arguments></code>	Outputs the given arguments to the console
<code>ls <file or directory></code>	If no argument is given, lists the contents of the current directory. If an existing file is given, outputs the name of the file. If a directory is given, lists the contents of that directory
<code>man <command name></code>	Outputs a detailed manual of a given command. Useful if you want to learn more about the commands listed here.
<code>mkdir <directory name></code>	Creates a directory with the given name
<code>pwd</code>	Outputs the full path of the current directory
<code>rm <file name></code>	Removes the given file. Does not work for directories
<code>rmdir <directory name></code>	Removes the given directory, which must be empty
<code>vim <file name></code>	Opens the given file in the text-based editor <code>vim</code> . You can use this to create and write to files using Bash. To learn how to use it (recommended), see http://www.openvim.com/