Terminals

Learn C With Babbo

Terminals, as explained in the previous lesson, are textual user interfaces where the user may navigate and manipulate their file system or execute programs. Indeed, when we create programs in C we will be executing those programs in some terminal. The terminal you will use will depend on your operating system (Windows, Mac, or Linux). In the following sections, I will describe the **Command Prompt**, which is the terminal primarily used on Windows systems, and **Bash**, the terminal primarily used on UNIX (Mac and Linux) systems. You only need to read the section corresponding to your operating system. I will use the verb "enter" to mean "to type and then press enter."

Command Prompt

The Command Prompt may be opened by pressing the Windows key + R, typing "cmd", and then pressing enter. Immediately, you may enter `dir` to see a list of the files and directories (which have `<DIR>` next to them) inside your current directory, which has its name given at the top of the output. `dir` is what is called a **command**, which is a named routine that may be run in your terminal. Also extremely useful is the command `chdir`, short for "change directory", which may be used to go to a different directory. To do so, enter, for instance,

```
chdir Documents
```

to navigate to the Documents directory, from which you may enter `dir` again to list its contents. Here, `Documents` is given as an **argument** or synonymously **parameter** to the command `chdir`, arguments being inputs appearing after the name of the command that are used to determine what the command should do. Without the argument `Documents`, the `chdir` command couldn't know where you wanted to change directories to. You may also change to a directory using that directory's full path, formally called its **absolute path**. It's analogue is a **relative path**, as in how `Documents` was used earlier. To navigate to your home directory with an absolute path, enter

```
chdir \Users\<your username>
```

Note that I use `<your username>` to denote where you should put your actual username, and I don't mean for you to enter the less than and greater than signs literally. It is common to use the less than and greater than signs in this way, where they are called **angle brackets**, to denote a **placeholder** for a variable input. Continuing, if you wanted to change to the outermost directory or **root** of your C drive, you could enter

```
chdir \
```

Enter `dir` again. Notice at the top of the list are two directories named `.` (a single period) and `..` (two periods). These represent your current and parent directories, respectively. Thus, if you enter

```
chdir .
```

you will not change directories at all. However, you may enter

```
chdir ..
```

to navigate to your **parent** or enclosing directory, which can be very useful. With this knowledge, you now know how to use the Command Prompt to navigate your file system.

Bash

If you are on Linux, you have probably already interacted with your terminal: if you have not, you can probably find it as an App which has a sort of black box as its icon. On Mac, you can open the terminal by pressing ⌘+  and entering "Terminal". The terminal you are using is called "Bash", despite the fact that the application you opened was probably not named Bash.

Now that your terminal is open, enter `ls`, which stands for "list status". You will see a list of files and directories inside your current directory. `ls` is called a **command**, which is a named routine that may be run in your terminal. To see what directory we're actually in, enter `pwd`, which stands for "print working directory." You will likely see something like `/Users/<your username>` or `/home/users/<your username>`. Here, I use **angle brackets**, which are a pair of less than and greater than signs which enclose something, as a **placeholder** for something. This is a common usage of angle brackets.

We can change our current directory using the `cd` command, which stands for "change directory". You probably have, for example, a directory called "Documents" in your home directory. To change into it, enter

```
cd Documents
```

Here, `Documents` is an **argument** or synonymously **parameter** to the `cd` command, an argument being some input following the name of the command. In this case, the argument tells `cd` what directory you want to change into. You may change back into your home directory using `cd ..` (two periods). `..` is a symbol representing your **parent** or enclosing directory, whereas `.`, just a single period, is your current directory. Therefore,

```
cd .
```

does not change directories at all. Now enter

```
cd /
```

This takes you to your **root** directory, which is the directory from which all files and directories in your file system "branch off" from. To return to your home directory from here, you may simply enter `cd` with no arguments. Alternatively, you could enter the full or **absolute path** of your home directory in the `cd` command, which may look something like

```
cd /home/users/<your username>
```

In contrast to an absolute path is a **relative path**, such as `Documents` in `cd Documents`. You now know the basics of navigating your file system with Bash.