**Environment**

Each time we launch the terminal application, it creates a new session. The session immediately loads settings and preferences that make up the command line *environment*.

We can configure the environment to support the commands and programs we create. This enables us to customize greetings and command aliases, and create variables to share across commands and programs.

You can reference the filesystem for this lesson [here](https://s3.amazonaws.com/codecademy-content/courses/learn-the-command-line/img/LCL-fileTrees-04.png).

# nano

Nice. You just edited a file in the nano text editor. How does it work?

$ nano hello.txt

nano is a command line text editor. It works just like a desktop text editor like TextEdit or Notepad, except that it is accessible from the command line and only accepts keyboard input.

1. The command nano hello.txt opens a new text file named **hello.txt** in the nano text editor.
2. "Hello, I am nano" is a text string entered in nano through the cursor.
3. The menu of keyboard commands at the bottom of the window allow us to save changes to **hello.txt** and exit nano. The ^stands for the Ctrl key.
4. Ctrl + O saves a file. 'O' stands for output.
5. Ctrl + X exits the nano program. 'X' stands for exit.
6. Ctrl + G opens a help menu.
7. clear clears the terminal window, moving the command prompt to the top of the screen.

In this lesson, we'll use nano to implement changes to the environment. You can learn more about nano [here](http://www.nano-editor.org/).

**Bash Profile**

You created a file in nano called **~/.bash\_profile**and added a greeting. How does this work?

$ nano ~/.bash\_profile

**~/.bash\_profile** is the name of file used to store environment settings. It is commonly called the "bash profile". When a session starts, it will load the contents of the bash profile before executing commands.

* The ~ represents the user's home directory.
* The . indicates a hidden file.
* The name **~/.bash\_profile** is important, since this is how the command line recognizes the bash profile.

1. The command nano ~/.bash\_profile opens up **~/.bash\_profile** in nano.
2. The text echo "Welcome, Jane Doe" creates a greeting in the bash profile, which is saved. It tells the command line to echo the string "Welcome, Jane Doe" when a terminal session begins.
3. The command source ~/.bash\_profileactivates the changes in **~/.bash\_profile** for the current session. Instead of closing the terminal and needing to start a new session, source makes the changes available right away in the session we are in.

**Aliases I**

What happens when you store this alias in **~/.bash\_profile**?

alias pd="pwd"

The alias command allows you to create keyboard shortcuts, or aliases, for commonly used commands.

1. Here alias pd="pwd" creates the alias pdfor the pwd command, which is then saved in the bash profile. Each time you enter pd, the output will be the same as the pwdcommand.
2. The command source ~/.bash\_profilemakes the alias pd available in the current session.

Each time we open up the terminal, we can use the pd alias.

# Aliases II

What happens when you store the following aliases in **~/.bash\_profile**?

alias hy="history"

hy is set as alias for the history command in the bash profile. The alias is then made available in the current session through source. By typing hy, the command line outputs a history of commands that were entered in the current session.

alias ll="ls -la"

ll is set as an alias for ls -la and made available in the current session through source. By typing ll, the command line now outputs all contents and directories in long format, including all hidden files.

**Environment Variables**

What happens when you store this in **~/.bash\_profile**?

export USER="Jane Doe"

*environment variables* are variables that can be used across commands and programs and hold information about the environment.

1. The line USER="Jane Doe" sets the environment variable USER to a name "Jane Doe". Usually the USER variable is set to the name of the computer's owner.
2. The line export makes the variable to be available to all child sessions initiated from the session you are in. This is a way to make the variable persist across programs.
3. At the command line, the command echo $USER returns the value of the variable. Note that $ is always used when returning a variable's value. Here, the command echo $USER returns the name set for the variable.

**PS1**

What happens when this is stored in **~/.bash\_profile**?

export PS1=">> "

PS1 is a variable that defines the makeup and style of the command prompt.

1. export PS1=">> " sets the command prompt variable and exports the variable. Here we change the default command prompt from $ to >>.
2. After using the source command, the command line displays the new command prompt.

# HOME

What happens when you type this command?

$ echo $HOME

The HOME variable is an environment variable that displays the path of the home directory. Here by typing echo $HOME, the terminal displays the path **/home/ccuser** as output.

You can customize the HOME variable if needed, but in most cases this is not necessary.

**PATH**

What happens when you type this command?

$ echo $PATH /home/ccuser/.gem/ruby/2.0.0/bin:/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin

PATH is an environment variable that stores a list of directories separated by a colon. Looking carefully, echo $PATH lists the following directories:

1. /home/ccuser/.gem/ruby/2.0.0/bin
2. /usr/local/sbin
3. /usr/local/bin
4. /usr/bin
5. /usr/sbin
6. /sbin
7. /bin

Each directory contains scripts for the command line to execute. The PATH variable simply lists which directories contain scripts.

For example, many commands we've learned are scripts stored in the **/bin** directory.

/bin/pwd

This is the script that is executed when you type the pwd command.

/bin/ls

This is the script that is executed when you type the ls command.

In advanced cases, you can customize the PATH variable when adding scripts of your own.

# env

What happens when you type this command?

env

The env command stands for "environment", and returns a list of the environment variables for the current user. Here, the env command returns a number of variables, including PATH, PWD, PS1, and HOME.

env | grep PATH

env | grep PATH is a command that displays the value of a single environment variable. Here the standard output of env is "piped" to the grepcommand. grep searches for the value of the variable PATH and outputs it to the terminal.

**Generalizations**

Congratulations! You learned to use the bash profile to configure the environment. What can we generalize so far?

* The *environment* refers to the preferences and settings of the current user.
* The *nano* editor is a command line text editor used to configure the environment.
* **~/.bash\_profile** is where environment settings are stored. You can edit this file with nano.
* *environment variables* are variables that can be used across commands and programs and hold information about the environment.
  + export VARIABLE="Value" sets and exports an environment variable.
  + USER is the name of the current user.
  + PS1 is the command prompt.
  + HOME is the home directory. It is usually not customized.
  + PATH returns a colon separated list of file paths. It is customized in advanced cases.
  + env returns a list of environment variables.