Lecture 01 - Command Line Basics (last updated: 10/27/22)

Note to the reader: this lecture note assumes you have zero prior exposure to the UNIX command line. If you are already familiar with this material or just need a light refresher, skip to the end to see a cheat sheet of common/useful UNIX commands.

We will do everything in this course using the command line, a textual interface for computers. You can interact with the command line on your computer using a terminal emulator, e.g., Terminal.app (macOS) or Windows Terminal (Windows). A terminal emulator is our bridge between the graphical user interface (GUI) and the command line.

On the command line, we typically perform tasks using a shell, e.g., Bash. We use a shell to execute commands; the shell will then show us the output of running that command (if any). For example, here is a snippet from a shell session where I ran echo, a Bash command that prints its argument to the terminal:

```
$ echo 'hello world'
hello world
```

Breaking it down:

- "$" represents the shell prompt. You do not need to type this.
- "echo" is the command you are executing.
- "'hello world'" is the argument you give to "echo".
- (newline) hit enter to run "echo".
- "hello world" is the output of running "echo".

When you are in a shell, the prompt indicates that it is ready to receive more commands from you. Note that real shell prompts are often longer than just "$". For example, some shell prompts will show your username and the name of the computer you are running on, among other information:

```
j-hui@clac:~ $
```

In lecture notes, exams, and other written material, I will usually shorten the shell prompt to just "$", unless the other information is relevant. I also like to leave an extra blank line between consecutive prompts, e.g.:

```
$ echo hello
hello

$ echo world
world
```

The blank line between "hello" and "$ echo world" is NOT part of the output of running "echo hello".

Sometimes, you will also see a shell command written with a comment:

```
$ echo 'hello world'    # output hello world
hello world
```

My course materials will often use this syntax to comment on what is happening in the shell. If you run that command line verbatim, your shell will ignore anything after the '#' character.

The directory tree

UNIX file systems are organized as trees, e.g.:

```
/
|-- bin/
|   |-- cat
|   |-- echo
|   |-- ls
|-- etc/
|   |-- hosts
|-- home/
    |-- edward/
    |   |-- taxes.xlsx
    |-- j-hui/
    |   |-- labs/
    |   |   |-- lab1/
    |   |-- lecture-notes/
    |       |-- 01-cli-basics.txt
    |-- karen/
        |-- photo1.jpg
        |-- photo2.jpg
        |-- photo3.jpg
```

Directories may contain files and other directories. The location of each directory and file is determined by the directory names encountered on the path from the root directory; we call this location a "file path".

Note a trailing "/" (e.g., home/) indicates that the path refers to a directory rather than a file.

Navigating file systems

When you are using a shell, your location in the directory tree is known as your "current working directory" (cwd) or "present working directory" (pwd). We can ask for the current working directory using the pwd command:

```
$ pwd
/home/j-hui
```

You can list (ls) the files in the current directory:

```
$ ls
labs   lecture-notes
```

We can change directory (cd):

```
$ cd lecture-notes
```

Running pwd and ls now will show the location and contents of lecture-notes/:

```
$ pwd
/home/j-hui/lecture-notes

$ ls
01-cli-basics.txt
```

Note: ls might not show every file on the same line. In some cases, it may also show every file on a separate line, e.g.:

```
$ ls
labs
lecture-notes
```

The actual output format may depend on which version of ls you are using, or the size of your terminal window, among other factors.

. and ..

There are two special directory names in UNIX: "." and "..". They refer to the current and parent directory.

You can use ".." to navigate to the parent directory:

```
$ pwd
/home/j-hui/lecture-notes

$ cd ..

$ pwd
/home/j-hui
```

You can also navigate through multiple directories at once:

```
$ pwd
/home/j-hui

$ cd labs/lab1

$ pwd
/home/j-hui/labs/lab1

$ cd ../..

$ pwd
/home/j-hui
```

The paths we've seen so far are all relative paths, i.e., relative to the cwd. You can also use absolute paths (beginning with "/") that do not depend on cwd:

```
$ cd /home/j-hui/labs

$ pwd
/home/j-hui/labs
```

"." doesn't seem very useful for now, but know that we can use it to make explicit that we are talking about a relative path:

```
$ cd ./lab1

$ pwd
/home/j-hui/labs/lab1
```

Hidden file names

In UNIX, all file names beginning with '.' are "hidden." This means that they will not appear when you run ls. However, we can tell ls to list all files and directories using the -a flag:

```
$ ls        # shows no output
$ ls -a     # shows all files and directories
.   ..   .bashrc   .ssh
```

With -a, ls shows the current (".") and parent ("..") directories, as well as the hidden file ".bashrc" and the hidden directory ".ssh". Because this directory contains .bashrc and .ssh, it is not empty.

In an empty directory, you will still see "." and ".." listed:

```
$ ls -a     # in an empty directory
.   ..
```

By convention, hidden file names are typically used for configuration or temporary files that don't need to be shown all the time. For example, from above, .bashrc is used to configure your shell, Bash, while the contents of .ssh are used to configure SSH, which allows you to login to a remote system.

Aside from not being visible by default, hidden files and directories are the exact same as regular files and directories. Do not confuse this concept with file permissions, which we will discuss later in this course.

Learning UNIX commands

There are more UNIX commands than anyone ever needs to know. You can find information about most commands by looking them up in the manual (man) pages:

```
$ man ls
```

Note that cd doesn't have a man page, so man cd does not work.

In case you were wondering, man even works on itself:

```
$ man man
```

I recommend learning commands by first seeing how they're used, then reading about them in their man pages, before trying them for yourself. The next section contains some commands you should know for the rest of this course.

UNIX commands cheatsheet

I want to...	File	Directory
show the contents of a...	cat or less	ls
delete a...	rm	rmdir (when empty) or rm -r
rename a...	mv	mv
copy a...	cp	cp -r
create a...	(depends)	mkdir

You will encounter, and are expected to know, the following commands:

- echo: print the arguments
- pwd: show the name of the current/present working directory
- cd: change directory
- ls: list directory contents
- mv: rename/relocate a file or directory
- cp: copy a file (recursively copy a directory when -r flag is given)
- rm: remove a file or directory
- mkdir/rmdir: make or remove an empty directory
- cat: print out the contents of a file
- less: show the contents of a file in a viewer that allows you to scroll
- clear: clear the terminal screen
- exit: log out of the shell
- man: look up manual pages for commands