Bash / Command Line

Q: What is Bash?

Bash = "**B**ourne **A**gain **SH**ell".

Q: Who/what is Bourne?

Stephen Bourne, created Bourne shell `sh` — the predecesor to `bash` — in the 1970s.

Q: What is a shell?

Shell is a layer around the operating system; it exposes the operating system services (file management, process management, configuration) to the user.

There are generally two types of shells:

- Command Line Interface (CLI), in which you type commands in the *command line*. You make computer do things by **typing**.
- Graphical User Interface (GUI), which provides a *graphical* way for manipulating programs. You make computer do things by **clicking**, **dragging**, **dropping**.

Somewhat confusingly, when we talk about *shells*, we generally refer to the command line interface only, and talk about it as opposed to GUI.

Q: Why would we want to do this?

Q: Why should we type, when we can click, drag, drop?

A: (Other than getting to feel like a hacker from a movie)

- Faster on a larger scale (automation)
- Safer (no mistakes in dragging and dropping)
- Can turn it into a script / repeatable
- Often there's no GUI for stuff

`bash` shell on your machine

- ~~Windows: `git bash` (`WSL`).~~
- Linux: `Terminal`
- Mac: `Terminal`, or `iTerm`

Type `ls`. You just executed a command equivalent to opening your Finder, Files, or File Explorer.

Basic `bash` commands

- *Navigation*: `pwd`, `cd`, `ls`
- Moving: `cp`, `mv`, `rm`, `mkdir`
- *Inspecting files*: `less`, `cat`

🎉 bash tutorial 🎉

1. Navigate the file system with `cd` and `pwd`
2. Create a folder for your work at Spiced with `mkdir`
3. Navigate into that folder and create a folder for the first week
4. Download bash tutorial zip file, navigate to *Downloads*, inspect content with `ls`
5. `mv` files to the created directory
6. (Unzip if needed)
7. Inspect a file with `cat` or `less`

BONUS: Other cool things for you to see/learn/google that may make your life easier:

- `~/.bashrc` or `~/bash_profile` file: configuration file for `bash`; you can set up `bash` to your liking here and configure things like how `bash` looks, environment variables, aliases...
- aliases: useful ways to save yourself a lot of typing for commands you use often, e.g. instead of typing `cd Documents/spiced_projects` every time you want to access your SPICED folder, you can add a line:

```
alias spiced='cd Documents/spiced_projects'
```

in your `~/.bashr` or `~/bash_profile` file and you'll only have to type `spiced` to get to your SPICED folder.

- print out a help page for a command: `man <command>`
- navigating to beginning/end of line: `CTRL-a`, `CTRL-e`
- wildcards:
  - `*` matches zero or more characters,
    - e.g. `rm *` would remove all files in a folder. ⚠️ Be VERY careful when doing this! ⚠️
  - `?` matches a single character,
    - e.g. `copy week_0?.ipynb ../` would copy Jupyter notebooks `week_01`, `week_02`, `week_03` into the folder one level up
  - `[]` matches any of the characters within the brackets,
    - e.g. `ls l[aeiou]st.txt` will list `last.txt`, `lest.txt`, `list.txt`, `lost.txt`, `lust.txt`
  - `{,}` matches any of the terms inside the curly brackets separated by comma,
    - e.g. `cp {*.pdf, *.ipynb} week_01/` will copy all `.pdf` *and* Jupyter notebook files into the `week_01` folder.
- counting lines, words, characters in a file: `wc -l`, `wc -w`, `wc -m`
- pipe, `|`: takes the output of first command as an input to the second. A common usecase is wanting to find out the id of a process/program slowing your computer down: `ps aux | grep chrome`. The first part lists all processes, the second gets only those that have the word `chrome` in them and prints them out to the terminal. You can then look up the process id (PID) and kill it with `kill -9 PID`.
- `reverse-i-search`: `CTRL-r`
- history expansion: `!$` gets the last parameter of the previous command,
  - e.g. type `ls <directory-name>`, then `cd !$` and it will take you to the directory you `ls`ed previously.