Day 07 - The Command Line Interface (CLI)

Sept. 29, 2020

From Pre-Class Assignment

There are a variety of experiences across our class:

- Some have never used the CLI while some use it daily.

Challenging bits

- Unable to get the CLI commands to work
- What are these commands doing; how to remember them all
- Getting `vim` and `nano` to work properly

Why use the CLI at all?

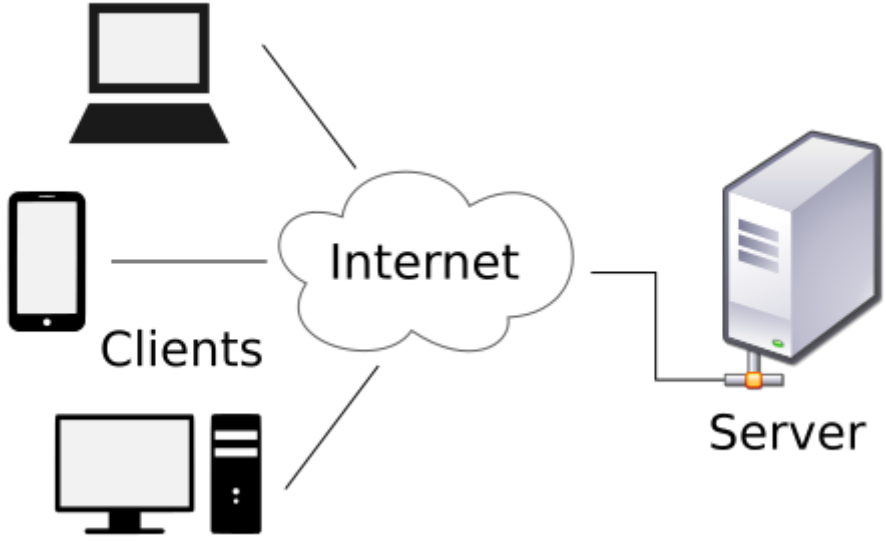
- Not all data science work can be done using a Graphical User Interface (GUI)
- Installing and configuring packages, libraries, and modules are typically done via CLI
- Using the CLI is typically more efficient than the equivalent GUI approach
- Interfacing with other systems through CLI ensures similarity across systems
- Many production computing systems provide no GUI at all

Your future work

Right now, you do all your computations on your own computer or on EGR's JupyterHub. In the future, you might:

- Work for a data science firm with its own computational systems
- Use cloud computing resources for analysis and modeling
- Administer high performance computing systems for a team of data scientists
- ...

These kinds of production systems typically make use of a client-server relationship.



MSU Institute for Cyber-Enabled Research (ICER)

- 900 compute nodes with more than 23,000 cores
- 142 TB of RAM
- 6.5 PB of persistent storage

CLI is used to access ICER resources.

Structure of a typical CLI command

- `command -f variable(s)` - The `command` is the main action you are attempting. The flag (`-f`) is a modifier that slightly changes the action. The `variable(s)` is/are passed to the action.
- Many CLI commands execute without any visual feedback.

CLI commands that you will use daily

- `ls` - lists the contents of a directory (`ls -a` modifies the command to list all files including hidden ones).
- `pwd` - prints the working directory to screen (shows where you are)
- `cd <directory>` - changes the working directory (moves you to a different location)
- `mkdir <new_directory>` - creates a new directory (must have name that doesn't yet exist in the working directory)
- `touch <new_file>` - creates a new blank file (again, must have a name that doesn't yet exist)
- `cp <file> <new_file>` - copies a file to a new file (will overwrite `<new_file>` with telling you)
- `cp -r <directory> <new_directory>` - the `-r` flag means recursive; copies full contents of a directory

Editing text files

Why did we introduce `vim` and `nano`?

To make edits to the kinds of files used in data science, we need plain text editors. MS Word, Google Docs, and the like make binary files; they contain additional metadata and markup for origin, formatting, et cetera.

`vim` and `nano` are the most commonly available CLI plain text editors.

I'd be more comfortable with a GUI

There are a number of text editors that make use of a GUI, which might be more comfortable right now.

- Atom (https://atom.io/) - Windows, Mac, Linux
- Sublime Text (https://www.sublimetext.com/) - Windows, Mac, Linux
- Visual Studio Code (https://code.visualstudio.com/) - Windows, Mac
- Notepad++ (https://notepad-plus-plus.org/) - Windows
- TextMate (https://macromates.com/) - Mac
- BBEdit (https://www.barebones.com/products/bbedit/) - Mac

Some of these can be set up as programming environments. For example, Danny uses Atom almost exclusively.

Questions, Comments, Concerns?