Unix shell commands

Ling 250/450: Data Science for Linguistics

C.M. Downey

Spring 2025

Why learn the command line?

- Commands are **more exact** than the Graphical User Interface (GUI)
- Many commands are **more powerful** than the GUI
- Some software libraries are **only useable** from the command line
- Some computers (such as “super computers” and research clusters) are **only accessible** via the command line! (No GUI available)

Basic navigation and paths

- **`pwd`** : “print working directory”, prints the path to the current directory
- **`ls`** : prints the contents of the current directory
  - **`ls -a`** : includes hidden files in directory contents
  - **`ls -l`** : includes more detailed information on contents (can combine both options with **`ls -la`**)
  - **`ls <dir>`** : prints the contents of the given directory
- **`cd <dir>`** : “change directory”, navigate to the given directory
  - **`cd ..`** : navigate to the parent directory of this one (go “one level up”)

Basic navigation and paths

- **Relative path**: path to a file or directory **relative to the current directory**
  - Ex: if the current directory contains a sub-directory called `Documents`, then `Documents/file.txt` is a **relative path**
- **Absolute path**: the **full path** to a directory or file, relative to the “root” directory of the computer
  - **`pwd`** gives the absolute path to the current directory
  - Absolute paths usually **start with /**
- **Home folder**: a directory serving as the starting point for a user’s shell
  - Always accessible with the **`~`** character (ex: **`cd ~`**)

Moving and creating files

- **`touch <file>`** : creates the given file (**only** if it doesn't yet exist)
  - I don't tend to use this command much; it just creates an empty file
- **`mkdir <dir>`** : "make directory", creates a directory with the given name
- **`mv <target> <destination>`** : moves a file or directory to the destination
  - Also used to **rename** files and folders
    - ex: **`mv old_filename.txt new_filename.txt`**
    - **Warning:** if the destination filename already exists, it will be **overwritten!**
  - `.` is a shortcut for "here", so it can be used to move something into the current directory

Moving and creating files

- **`cp <target> <destination>`** : copies a file or directory to the destination
  - Works much like `mv`, except it leaves the **original target in place**
  - Need to add the option **`-r`** to **copy a directory**
    - ex: **`cp -r my_dir some_other_dir`**
- **`rm <file>`** : remove a file (**permanently; does not go to trash**)
- **`rm -r <dir>`** : remove a directory (also permanently)
- **BEWARE:** adding the `-f` option makes this an extremely powerful command
  - f is for “force”, and **suppresses warnings** about removing important files

Viewing and manipulating files

- **`cat <file>`** : print the contents of the given file
- **`head <file>`** : print the first few lines of the given file
- **`tail <file>`** : print the last few lines of the given file
  - **`tail -n <num_lines> <file>`** : print the last n lines of the file (works for `head` too)
- **`more <file>`** : print the first portion of a file for viewing
  - [enter] : scroll slowly through file
  - [space] : scroll fast through file
  - q : stop scrolling file

Viewing and manipulating files

- **`>`** : special symbol for **writing the output of a command to a file**
  - ex: **`head file.txt > file_intro.txt`**
  - **Warning**: this **overwrites** the file that is written to
- **`>>`** : special symbol for **appending** the output of a command to a file
  - Adds the content to the **end of the file**, rather than overwriting it
- **`wc <file>`** : “word count”, print the count of lines, words, and characters in a file
  - **`wc -l`** : print just the number of lines (**`-w`** for words, **`-c`** for characters)

Viewing and manipulating files

- **`grep “pattern” <file>`** : find and print all lines from the file that contain the pattern in quotes
  - The literal command above would return all lines with the word `pattern`
  - `grep` has **wayyy more options** than this. This is just the most basic usage!
- **`diff <file1> <file2>`** : give a printout of all the differences between first and second file
  - This printout can be **pretty hard to read**. Best used simply to sanity-check that two files are identical

Other tips and tricks

- Pressing the **tab key** will **auto-complete** the path or command you are currently typing, as long as it is **unambiguous**
  - ex: if you type **`~/Documents/fil`**, and the only file in that folder is **`~/Documents/file.txt`**, it will complete the name of the file for you
  - If what you've typed matches **multiple completions**, it will **list them**
- The "star" character (`*`) can be used to mean **"everything"** (in the current directory)
  - ex: **`*.txt`** can be used to indicate **all text files**
- **ctrl+c** : force the current command to stop (good when something is out of control)

Other tips and tricks

- **ctrl+a** : move cursor to beginning of line
- **ctrl+e** : move cursor to end of line
- **ctrl+p** : get the previously run command (keep pressing for history)
- **`du -h <file>`** : get the size (i.e. “disk usage”) of a file
- **`ping <url>`** : send a basic signal to a website to test connection
  - ex: **`ping www.google.com`** is a good way to sanity-check that your internet is working