CSC209 Notes

Software Tools and Systems Programming

https://github.com/icprplshelp/

Last updated January 10, 2023

1 Introduction

This course is about:

- Software tools
  - Using the command line and not clicking on UIs (apparently linux addicts hate them)
- Systems programming
  - No, this course isn't learning to program
  - It's about the pieces of C we haven't seen yet
  - And the systems part of programming in C (the file system, the notion of processes, and communications over the network)

1.1 Unix Principles

- Unix is different fundamentally than an IDE or an application with a UI.
  - An IDE gives you a button for everything.
- Unix has simple tiny tools that can be combined to do interesting tasks.
  - A lot of them are programs that do one things, maybe with a few variations, but all the same thing
  - We have a way to connect these tools together in different combinations to do more interesting tasks
- To do everything together, **they work with plain text files.**
- **None of the UNIX tools (should) require human-interactive input** (e.g. `input()` in python)
  - Why? So we can automate things and put commands into scripts, so they can run without having us involved
  - We want a simple output format, so the output format is the simple format that the next tool can take the input

  - I/O streams
    * In Java, we have `System.out.println()` (standard output) and `System.err.print()`
    * For every process, we have a `stdin` (most of the time, it's a keyboard).

1.2 Commands

UNIX is short on vowels and you'll rarely see them. Commands are short.

- `cd` for change directory
- `ls` is for list all the commands on the path or the current directory
  - a.k.a. all your files… probably I'm wrong on this
  - `ls -F` lists them with an extra slash, listing all files with a `/` at the end if it is a directory. For example: `file1.txt file2.txt folder/`
  - `-lF` gives us the long listing (and combines the effects of `-F` – more verbose and looks like file explorer but with text)
- `cat <file>` (stands for concatenate) shows me the files content… but
  - The `cat` command is running. It takes the argument `<file>` and uses it as the filename it should open and read for standard input.
  - `cat` takes what comes in standard input and push it to standard output
  - Then the shell displays the standard output in the window for us.
  - Multiple arguments? `cat` stands for concatenate, so it concats all the inputs and sends the output to standard output.
    `cat <file> <file> <file>` puts file three times into the standard output.
  - `cat document document > double_document` – you know what this does. That's exactly how you concat stuff into a new file.
- `sort` sorts each line in lexical order and puts it in standard output.
- `wc document` (gives me the word count. It gives me `LINE_COUNT WORD_COUNT CHARACTERS` (words are tokens?)
- `sed "<regex>"` (stream editor)

  - Applies the regex to ALL lines and sends it into the standard output.
  - Example: `sed 's/, /XXX/' document`
    * Search for `,` and replace it with `XXX` (similar to find and replace)
  - Let's do this again: `sed 's/\(.*\), \(.*\)/\2 \1/' document`... good luck figuring that out (basically transforms `LastName, FirstName` to `FirstName LastName`

- Piping: the pipe symbol – whatever process is on the LEFT, the standard output from THE LEFT becomes the standard input on the RIGHT. For example:
  - `sed 's/\(.*\), \(.*\)/\2 \1/' document | sort`
    * We don't sort by first name, so maybe sort first?
  - `sort document | sed 's/\(.*\), \(.*\)/\2 \1/'`
  - Now, this command, if `document` were a text file filled with `LastName, FirstNames` on each line, it sorts by last name first THEN does the swap.

- `man <command>` gives us the manual page for the command. For example, `man sort` gives us a help page.
- `cut -d " " -f 1` has the same effect as `line.split(' ')[0]` for every line in the file.
- `unique`... you know this... **NOT**.
  - Filters out **ADJACENT** matching lines.
  - If you really want to get rid of duplicates, SORT FIRST

> UNIX is user-friendly; it's just choosy about its friends.
>
> Become a friend of UNIX.

1.2.1 The Grep Command

> Grep: search for (a string of characters) using grep.

It's in the dictionary. It means **global regular expression and print.** The idea of gripping is you're searching an RE on a file and you're sending in standard output the lines that match.

For example, find all the lines that match this RE and print it.

And do you want to grep but get the numbers of occurrences? Pipe it into `wc`.