WEEK 2

COMP(2041|9044)

SHELL = CRAB = RUST???

AGENDA

Week 1	Regex, Grep
Week 2	Pipelines and Filters (sort, cut, sed, etc)

REDIRECTING INPUT & OUTPUT

- The > command redirects output from the command on the left and passing to the file on the right.

```
echo "Hello World" > output.txt
```

- The < command redirects the right file as input for the left command.

```
./hello.c < input.txt
```

- The | command is called a pipe. It can transfer standard output from the command on the left into standard input of the command on the right.

```
echo "Hello World" | grep "Hello"
```

HEAD

- Output the first part of files
- head [OPTION] ... [FILE] ...
- Important Flags to remember:
  - -n, --lines=[-]NUM: print the first NUM lines instead of the first 10; with the leading ‘-’, print all but the last NUM lines of each file.

```
$ head -3 hello.txt
```

PIPES

2. Consider this Unix password file
   (usually found in `/etc/passwd`):

```
root:ZHolHAHZw8As2:0:0:root:/root:/bin/dash
jas:iaiSHX49Jvs8.:100:100:John Shepherd:/home/jas:/bin/bash
andrewt:rX9KwSSPqkLyA:101:101:Andrew Taylor:/home/andrewt:/bin/cat
postgres::997:997:PostgreSQL Admin:/usr/local/pgsql:/bin/bash
oracle::999:998:Oracle Admin:/home/oracle:/bin/bash
cs2041:rX9KwSSPqkLyA:2041:2041:COMP2041 Material:/home/cs2041:/bin/bash
cs3311:mLRiCIvmtI9O2:3311:3311:COMP3311 Material:/home/cs3311:/bin/zsh
cs9311:fIVLdSXYoVFaI:9311:9311:COMP9311 Material:/home/cs9311:/bin/bash
cs9314:nTn.JwDgZE1Hs:9314:9314:COMP9314 Material:/home/cs9314:/bin/fish
cs9315:sOMXwkqmFbKlA:9315:9315:COMP9315 Material:/home/cs9315:/bin/bash
```

   Provide a command that would produce each of the following results:

   a. Display the first three lines of the file

   b. Display lines belonging to class accounts
      (assuming that class accounts have a username that starts with either "cs", "se", "bi" or "en", followed by four digit)

   c. Display the username of everyone whose shell is `/bin/bash`

   d. Create a tab-separated file `passwords.txt` containing only the username and password of each user

CUT

- Remove sections from each line of files
- cut OPTION ... [FILE] ...
- Important Flags to remember:
  - -d, --delimiter=DELIM: use DELIM instead of TAB for field delimiter
  - -f, --fields=LIST; select only these fields

```
$ cut -d' ' -f3 file.txt
```

TR / TRANSLATE

- Translate or Delete Characters
- tr [OPTION] ... SET1 [SET2]
- Important Flags to remember:
  - -d, --delete: delete characters in SET1, do not translate
  - -c, -C, --complement: use the complement of SET1
  - -s, --squeeze-repeats: replace each sequence of a repeated character with a single occurrence

```
$ tr -d ' ' < file.txt
$ tr 'abc' 'def' < file.txt
```

UNDERSTANDING TR 1

Consider this fairly standard split-into-words technique.

```
$ tr -cs 'a-zA-Z0-9' '\n' < someFile
```

Explain how this command works.
What does each argument do?

wc

- Print newline, word and byte counts for each file
- wc [OPTION] ... [FILE] ...
- wc [OPTION] ... --files0-from=F
- Important Flags to remember:
  - -l, --lines: print the newline counts
  - -w, --words: print the word counts

```
$ wc -l < text.txt
```

UNDERSTANDING TR 2

4. What is the output of each of the following pipelines if the text:

```
this is big Big BIG
but this is not so big
```

is supplied as the initial input to the pipeline?

a. `$ tr -d ' ' | wc -w`

b. `$ tr -cs '[:alpha:]' '\n' | wc -l`

c. `$ tr -cs '[:alpha:]' '\n' | tr '[:lower:]' '[:upper:]' | sort | uniq -c`

SORT

- Sort lines of a text file
- Important Flags to remember:
  - -n, --numeric-sort: compare according to string numerical value
  - -r, --reverse: reverse the result of comparisons
  - -t, --field-seperator=SEP: splits lines by the seperator
  - -k, --key=KEYDEF: sort via a key

```
$ sort -t'|' -k4,4 k2,2rn file.psv
```

SED

- Stream editor for filtering and transforming text
- Important Flags to remember:
  - -E, -r, --regexp-extended: use extended regular expressions in the script
  - -n, --quiet, --silent: suppress automatic printing of pattern space
  - -i[SUFFIX], --in-place[=SUFFIX]: edit files in place

```
$ sed -E 's/H[el]lo/world/' file.txt
```

SED

- sed is VERY powerful but also very complex.
- Here are some useful options for sed in this course:
  - s for substitution => s/oldText/newText/
  - /g - global replacement => s/oldText/newText/g
  - /d - delete => /deletedText/d
  - /regex1/ , /regex2/ => selects all lines between lines matching regex1 and regex2

LAB TIME!!!

Code and Slides Avaliable here

https://github.com/jeremyle56/tutoring/tree/24T2/24T2/cs2041