Lecture 10 (2-4-26)

Pipelines, Regex, Filters (continuation from last lecture)

Motivating Questions

1. Why is the pipeline pattern so powerful?
2. How do we use regular expressions to match text?
3. How do we translate characters?
4. How do we extract fields?
5. How do we search for patterns?
6. How do we modify text streams?
7. How do we perform more complex text processing?

Filters

- A **filter** reads in input, performs some sort of transformation, and the outputs to stdout.
  - could be the identity transformation
    - EX: `cat` spits out what it reads

Examples:

- `grep`
- `tr`

`tr` command examples

```
echo notre dame our mother #data is being passed in as command line
arguments

---

echo notre dame out mother | tr a-z A-Z # mapping from lowercase to
uppercase
```

```
---

echo notre dame our mother | tr 4310 aeio #vowals to numbers 4-->a, 3-->e,
etc.

---

echo notre dame our mother | tr -d ' ' #deleting spaces

---

echo notre dame our mother | tr -d [:space:] #deleting spaces
```

`cut` command examples

```
echo notre dame our mother | cut -d ' ' -f 2,4
#outputs "dame mother"

---

echo notre dame our mother | cut -d ' ' -f 2-
#outputs "dame our mother"

---

echo notre dame out mother | cut -c 12-15
#outputs "our"
```

`awk` command examples

```
echo notre dame our mother | awk -F ' ' '{print $4,$2}'
#outputs "mother dame"
```

```
echo notre dame our mother | awk '{print $4, $2}'
#outputs "mother dame", notice delimeter by default is a space in `awk`
```

`grep` command examples

```
cat /etc/passwd
#this file contains all the user accounts on the machine
#each line is a seperate user


---

cat /etc/passwd | grep root
#fitlers for root

---

cat /etc/passwd | cut -d : -f 1
#this outputs only the first field when pased with a delimeter of :

---

cat /etc/passwd |  tr ':' ','
#this translates colons to commas
#now the text is good in case you want in in csv format

---

cat /etc/passwd | grep -E '1[0-9]*[13579]+:'
#highlights all the lines that have numbers that start with 1 and end with
an odd number
# the colon at the end is because we KNOW there will be a colon after the
last number due to the colon being the delimeter

---

cat /etc/passwd | grep -Eo '1[0-9]*[13579]+:' | grep -Eo '[0-9]*'
#this outputs just the nunmbers WITHOUT the colon since we used `grep` again
#the `-o` only outputs the pattern itself, not the line
```

`grep` for webscraping

```
curl -sL some_website_url | grep -Eo '[0-9]{3}-[0-9]{3}-[0-9]{4}'
#this retuns all the phone numbers off the website
```

```
---

curl -sL some_website_url | grep -Eo '[0-9]{3}-[0-9]{3}-[0-9]{4}' | sort |
uniq

#this outputs only the unique phone numbers

---

curl -sL some_website_url | grep -E '[[:alnum:]]+@[[:alnum:]\.]+'
#looking for all emails

---

curl -sL some_website_url | grep -v meta | grep -Eo
'[[:alnum:]]+@[[:alnum:]\.]+'
#remember [[:alnum:]\.] is a set with all "alnums" and a "."
    # we used the backslash in "\." to escape the special meaning of the
period
#only displaying the emails itself
```

`sed` command

```
echo "monkeys love bananas" | sed 's/monkeys/joshua'
#replacing "monkeys with joshua"

---

echo "monkeys love bananas and monkeys climb on trees" | sed
's/monkeys/joshua/g'
# replacting all "monkeys" with "joshua"

---

echo "      monkeys love bananas" | sed 's/^ *//'
#replacing all the spaces at the start of the line with NOTHING
    # `tr` would not work for this purpose.

---
```

```
echo "      monkeys love bananas   " | sed 's/^ *//' | cat -E
# the `sed -E` shows you that the spaces at the end have not been repalced
```