Lecture 3

Tuesday, September 22, 2015 2:35 PM

Streams:
stdout is buffered

- This is to improve performance
- Drawing to the screen is expensive

stderr is not

Find out how to output to two files at once

'>>' appends

"How many words occur in the first 10 lines of sample.txt?"
$> head -10 sample.txt

- Prints the number of words in the first 10 lines
- Can use output redirection

```
$> head -10 sample.txt > temp
$> wc -w temp
$> rm temp
```

- Instead of doing the above, we can do the following:

Piping:
Pipe: connect the stdout of program 1 to stdin of program 2

stdin 1--> [program 1] --> stdout1 --> stdin2 [program 2]

- | (pipe)
- Can pipe between multiple programs

e.g.) $> head -10 sample.txt | wc -w
e.g.) $> cat sample.txt | head -10 | wc -w

The two examples are equivalent

e.g.) Suppose files **words1.txt** and **words2.txt** contain a list of words, one per line

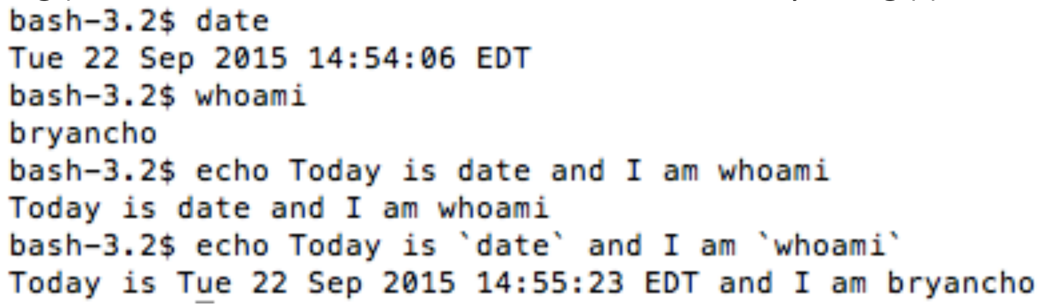
- Print a duplicate free list of words from words*.txt

1. Merge words1.txt and words2.txt
   - $> cat words*.txt
2. Sort the result
   - $> cat words*.txt | sort
3. Remove duplicates
   - $> cat words*.txt | sort | uniq

Sending the output of one program as an argument to another program:

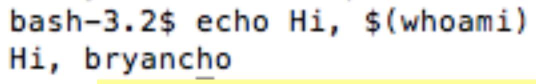
May be on the midterm!

e.g.) You can embed commands within another by using (`)



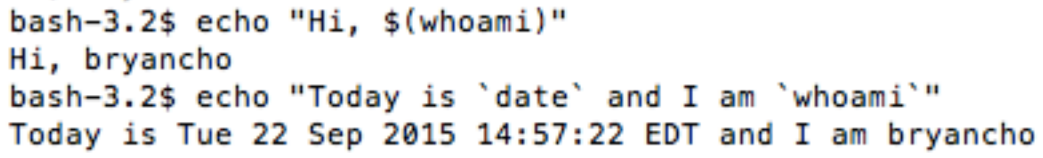
```
bash-3.2$ date
Tue 22 Sep 2015 14:54:06 EDT
bash-3.2$ whoami
bryancho
bash-3.2$ echo Today is date and I am whoami
Today is date and I am whoami
bash-3.2$ echo Today is `date` and I am `whoami`
Today is Tue 22 Sep 2015 14:55:23 EDT and I am bryancho
```

- You can also do $(cmd)



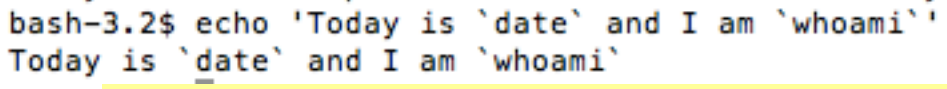
```
bash-3.2$ echo Hi, $(whoami)
Hi, bryancho
```

- If we wrap the entire argument in double quotes, the output is the same:



```
bash-3.2$ echo "Hi, $(whoami)"
Hi, bryancho
bash-3.2$ echo "Today is `date` and I am `whoami`"
Today is Tue 22 Sep 2015 14:57:22 EDT and I am bryancho
```

- However, what is happening behind the scenes is different
- This only sends a single argument to `echo`



```
bash-3.2$ echo 'Today is `date` and I am `whoami`'
Today is `date` and I am `whoami`
```

- Wrapping the argument in single quotes does not allow you to embed commands

Searching within a text file:

grep: global regular expression pattern
egrep: extended grep

```
$> egrep pattern file
```

- **Output**: prints every line in file that contains this pattern

Checkout 'man egrep' for assignment

BEWARE: REGEX AHEAD:

You can escape | by putting \ before it

```
$> egrep cs246\|CS246 file
```

OR using "

```
$> egrep "cs246|CS246" file
```

You can also "factor" out terms:

```
$> egrep "(cs|CS)246" file
```

Like how you do x(x+1) = x^2+x

[abcd]

- Any **one** character from this list
- Short form for (a|b|c|d)

[^abcd]

- Any one character that is **not** in this list

? - 0 or 1 of the preceding expression
* - 0 or more of the preceding expression
+ - 1 or more of the preceding expression

"cs_?246" = cs246, cs 246
"(cs_)?246" = 246, cs 246
"(cs_?)?246" = cs246, cs 246, 246, cs 246

- _ is space

- To match a special character escape | using \
- \|
- *
- \?
- Except within [] where characters do not have special meaning
    - [a?]
- "(cs)*246"
    - 246

    - cs246
    - cscs246
    - …
    - Infinite repetitions of 'cs'
- '.' - stands for any one character
- '.*' - 0 or more of any characters (repetition)
- + - any non-empty sequence of characters

- To get matched with lines that start with the pattern, use ^
- "^cs246" the line should start with…
    - a c
    - a c followed by an s
    - a c followed by an s followed by a 2…
    - cs246
- To get matched with lines that end with the pattern, use $
    - "cs246$"
- Lines only containing cs246:
    - "^cs246$"
    - "^cs246+$"
- Print all words of even length from /usr/share/dict/words
    - egrep "(..)+" /usr/share/dict/words
        - This will not work
        - Ex) the
        - The pattern above will match the first two characters and print it
    - egrep "^(..)+$" /usr/share/dict/words will work
- Print all words in /usr/share/dict/words that start with an 'e' and have length 5
    - egrep "^e….$" /usr/share/dict/words
- Print all files whose name contains exactly 1 a
    - ls | egrep "^[^a]*a[^a]*$"
    - Something that is not an **a** followed by an **a** followed by something that is not an **a**