CSCI 3308 Software Development Methods and Tools
Instructor: David Graham

Lab 2 - Regular Expression

—

TA: Yawen Zhang

Lab 2 - Regular Expressions

Objectives

- Use regular expressions with common Unix commands
- Practice using some useful Unix commands
- Practice creating and running bash shell scripts
- Practice using pipes

Lab Link

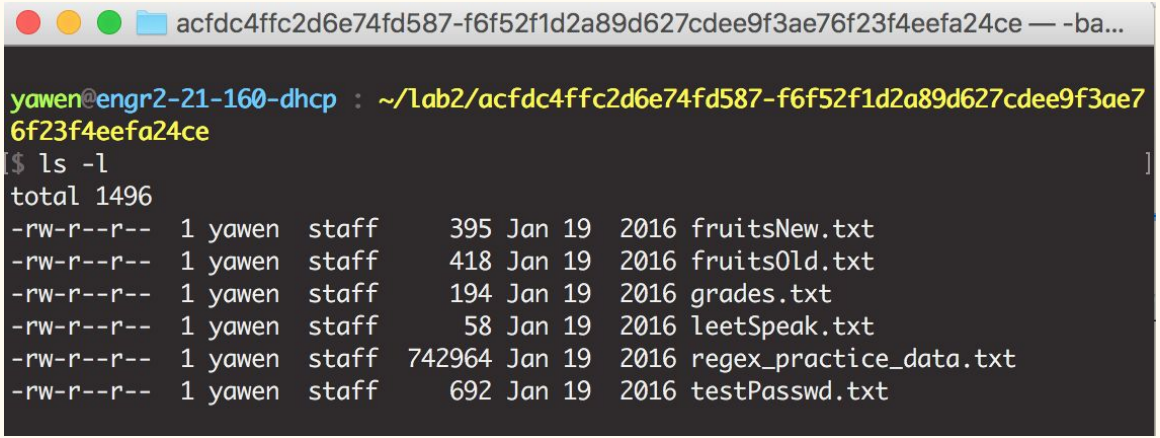
- http://www.lousymedia.com/csci-3308/labs/lab-2

Preparation: download practice files (step 1)

curl -L https://gist.github.com/dgraham/acfdc4ffc2d6e74fd587/archive/f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce.zip > lab2.zip

unzip lab2.zip -d lab2

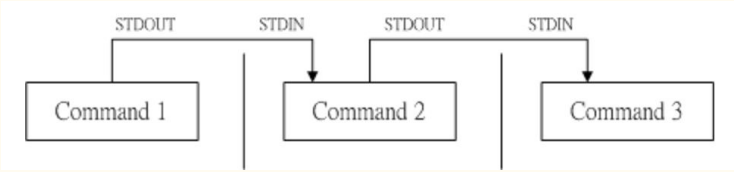
cd lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce



Practice Unix Commands

General Unix command format:
command –option1 argument -option2 argument ...

- **diff :** diff file1 file2
- **wc :** wc -l file1
- **cut :** cut -d : -f 3 file1
- **pipe** : cut -d : -f 3 file1 | sort > file2
- **sed :** sed s/yourname/myname/g file1
- **awk :** awk ‘NR > 1{print $1}’ file1
- **grep/egrep** : grep -c **‘^[0-9]’** file1



awk (step 7)

- a little more difficult, but very powerful text processing command
- work with lines in a file
- build-in variables: NR, NF(number of fields), FS (field separator, space by default)

```
Syntax:


awk '/search pattern1/ {Actions}
     /search pattern2/ {Actions}' file
```

awk (step 7)

- awk Example 1. Default behavior of awk
- command: **awk '{print;}' grades.txt**

acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...

```
yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae7
6f23f4eefa24ce
[$ awk '{print;}' grades.txt                                                    ]
FN LN Lab HW1 HW2 HW3 HW4
Ryan Slaven 1 1 0 1 1
Jephthah Eustathios 0 1 0 1 0
Andreas Saša 1 0 1 0 1
Godofredo Gerard 1 1 1 1 1
Edwin Babur 1 0 1 1 1
Ahmad Marin 0 0 0 0 0
Jett Marko 1 1 0 1 1
```

awk (step 7)

- awk Example 2. Print only specific field
- command: **awk ‘{print $1“ ”$2}’ grades.txt**

```
acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...

yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae7
6f23f4eefa24ce
$ awk '{print $1" "$2}' grades.txt
FN LN
Ryan Slaven
Jephthah Eustathios
Andreas Saša
Godofredo Gerard
Edwin Babur
Ahmad Marin
Jett Marko
```

awk (step 7)

- awk Example 3. Print the lines which matches with the pattern
- command: **awk '$3 >= 1{print $1“ ”$2}' grades.txt**

```
acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...

yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce
$ awk '$3 >= 1{print $1" "$2}' grades.txt
FN LN
Ryan Slaven
Andreas Saša
Godofredo Gerard
Edwin Babur
Jett Marko
```

awk (step 7)

- awk Example 4. Initialization and Final Action (BEGIN + END)
- command: **awk 'BEGIN{count = 0}$3 == 1{count++}END{print "number of students with lab grad equal to 1: "count}' grades.txt**

```
Syntax:

BEGIN { Actions}
{ACTION} # Action for everyline in a file
END { Actions }

# is for comments in Awk
```

acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...

```
yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce
$ awk 'BEGIN{count = 0}$3 == 1{count++}END{print "number of students with lab grad equal to 1: "count}' grades.txt
number of students with lab grad equal to 1: 5

yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce
$ awk '$3 == 1' grades.txt
Ryan Slaven 1 1 0 1 1
Andreas Saša 1 0 1 0 1
Godofredo Gerard 1 1 1 1 1
Edwin Babur 1 0 1 1 1
Jett Marko 1 1 0 1 1
```

awk (step 7)

- awk Example 5. Using build-in variables
- command: **awk 'NR > 1{print $1" "$2}' grades.txt**

acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...

```
yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae7
6f23f4eefa24ce
$ awk 'NR > 1{print $1" "$2}' grades.txt
Ryan Slaven
Jephthah Eustathios
Andreas Saša
Godofredo Gerard
Edwin Babur
Ahmad Marin
Jett Marko
```

Regular Expression (with **egrep**)

- Q: find lines containing numbers in file
- A: **egrep '177' regex_practice_data.txt**

```
acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...

yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae7
6f23f4eefa24ce
[$ egrep '177' regex_practice_data.txt                                          ]
303-724-1777
1776
jannick1771
cole21771
swatteam177
kingofghost177
177Aidan
logan1771
darkknight1775
spartan1779
JAck2177
supashadow1777
```

Regular Expression (with **egrep**)

- Single char matching
- [ ] : matching any character defined by contents of [ ]
  - [abc]: any char of a, b or c
  - [a-z]: any letter between a, z
  - [^a-z]: any char except a-z (^ is a “not” operator for elements in [ ])
  - [0-9]: any char of 0-9 (i.e. numbers)
  - [0-9a-zA-Z]: any char of 0-9, a-z, A-Z (i.e. numbers or letters)
  - [0-9a-zA-Z_]: any char of 0-9, a-z, A-Z and _
  -
- . : matching any character

Regular Expression (with **egrep**)

- Q: find lines beginning (ending) with numbers in file
- A: egrep "^[0-9]" regex_practice_data.txt
- (egrep "[0-9]$" regex_practice_data.txt)
-
- Q: find lines with 3 numbers
- A: egrep "[0-9]{3}" regex_practice_data.txt

acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -bash — 99×30

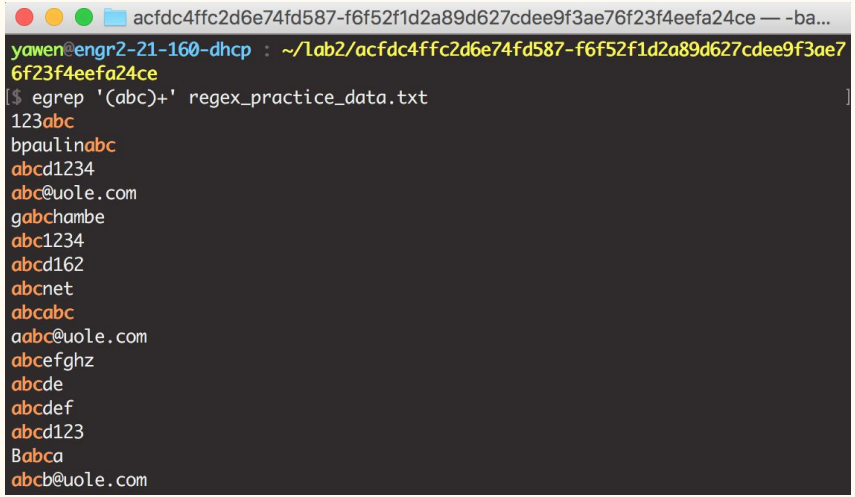
```
$ egrep '^[0-9]' regex_practice_data.txt
303-724-1777
4271
1337dman1337
8bitrules
1monkey
363636
303-441-3330
100hamburger
9black9
5ravemaster
303-724-9623
215-305-2003
303-724-6725
1369
303-724-3203
66666
123abc
```

acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -bash — 99×30

```
yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce
$ egrep '[0-9]{3}' regex_practice_data.txt
303-724-1777
Jordy3000
v0lc0m172
4271
nitrokillaz1220
westking123
bob1734
sunny1414
wired2000
1337dman1337
Wolfy42345
WhiteWolf1423
srbuckey5266
HaWk083
zombieslayer2011
```

Regular Expression (with **egrep**)

- Repetition of last character
  - *: 0 or 1 or more repetition ([0-9]* : 0 or 1 or more numbers)
  - ?: 0 or 1 repetition ([0-9]? : 0 or 1 numbers)
  - +: 1 or more repetition ([0-9]+ : 1 or more numbers)
  - {n}: n repetition ([0-9]{3}: 3 numbers)
  - {n,m}: n to m repetition ([0-9]{3,5}: 3 or 4 or 5 numbers)
  - {n,}: n or more repetition



Regular Expression (with **egrep**)

- Tips:
  - Search “regular expression egrep” for details
  - Other editors/tools (vim/sed/python/..) also support regular expression, but the syntax may differ a little
  - Use backslash “\” to transfer meaning for special chars
    - Example: egrep “\.” test.txt (search dot (.))

Basic Syntax in RE (step 8)

abc...	Letters
123...	Digits
\d	Any Digit
\D	Any Non-digit character
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
\s	Any Whitespace
\S	Any Non-whitespace character
[abc]	Only a, b, or c
[^abc]	Not a, b, nor c
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9

{m}	m Repetitions
{m,n}	m to n Repetitions
.	Any Character (one)
*****	Zero or more repetitions
+	One or more repetitions
?	Optional character (zero or one)
\.	Period
^a	Starts with a
a$	Ends with a (the end of a string)
(abc)	Capture Group
(a(bc))	Capture Sub-group
(abc|def)	Matches abc or def

- **On-line testing with your regular expression commands**

http://regexr.com/

Lab Task

- Task: **step 1 - 8**
- For submission, prepare a .txt file (you may try editing it with **vim** or **any text editor** you like)
- Save all the commands you use for **step 2 - 8** in the .txt file
- When you **finish**, show me your **.txt file** and get grade for the lab

Search the Internet !

when you get puzzled

Link to slide

https://github.com/yawenz/csci_3308