

# 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

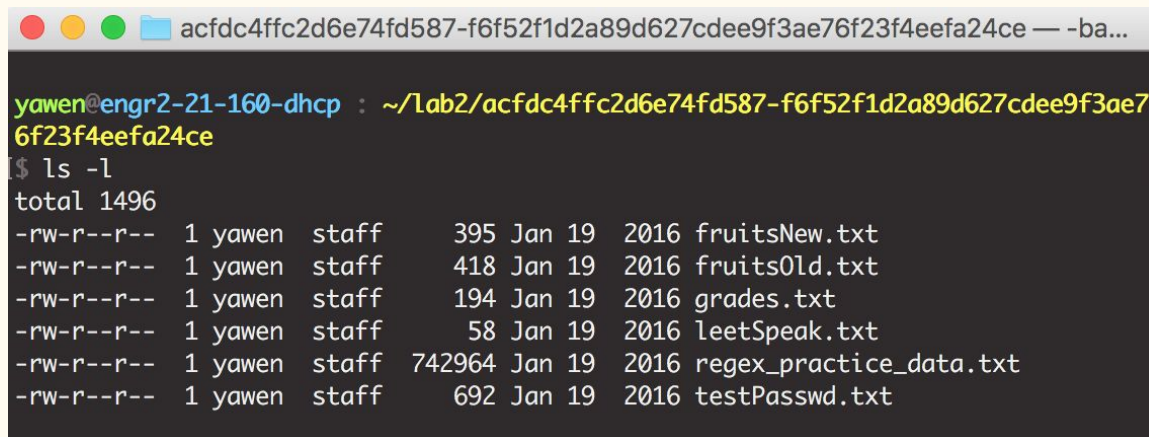
- ❖ <http://www.lousymedia.com/csci-3308/labs/lab-2>

# Preparation: download practice files (step 1)

```
curl -L  
https://gist.github.com/dgrah  
am/acfdc4ffc2d6e74fd587/ar  
chive/f6f52f1d2a89d627cdee  
9f3ae76f23f4eefa24ce.zip >  
lab2.zip
```

```
unzip lab2.zip -d lab2
```

```
cd  
lab2/acfdc4ffc2d6e74fd587-f  
6f52f1d2a89d627cdee9f3ae7  
6f23f4eefa24ce
```

A terminal window with a dark background and light text. The title bar shows a folder icon and the path 'acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce' followed by a prompt character and '-ba...'. The prompt is 'yawn@engr2-21-160-dhcp : ~/Lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce'. The user has entered 'ls -l' and the output shows a directory listing with permissions, owner, group, size, date, and filename.

```
acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...  
yawn@engr2-21-160-dhcp : ~/Lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce  
$ ls -l  
total 1496  
-rw-r--r--  1 yawn  staff    395 Jan 19  2016 fruitsNew.txt  
-rw-r--r--  1 yawn  staff    418 Jan 19  2016 fruitsOld.txt  
-rw-r--r--  1 yawn  staff    194 Jan 19  2016 grades.txt  
-rw-r--r--  1 yawn  staff     58 Jan 19  2016 leetSpeak.txt  
-rw-r--r--  1 yawn  staff 742964 Jan 19  2016 regex_practice_data.txt  
-rw-r--r--  1 yawn  staff    692 Jan 19  2016 testPasswd.txt
```

# Practice Unix Commands

❖ **diff** : `diff file1 file2`

❖ **wc** : `wc -l file1`

❖ **cut** : `cut -d : -f 3 file1`

❖ **pipe** : `cut -d : -f 3 file1 | sort > file2`

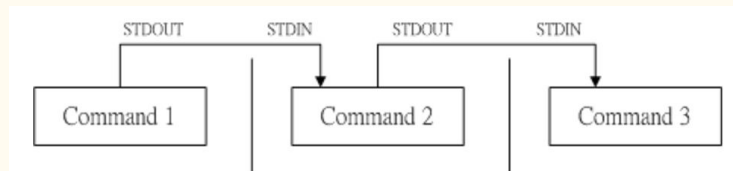
❖ **sed** : `sed s/yourname/myname/g file1`

❖ **awk** : `awk 'NR > 1{print $1}' file1`

❖ **grep/egrep** : `grep -c ‘^[0-9]’ file1`

**General Unix command format:**

`command -option1 argument -option2 argument ...`



# awk (step 7)

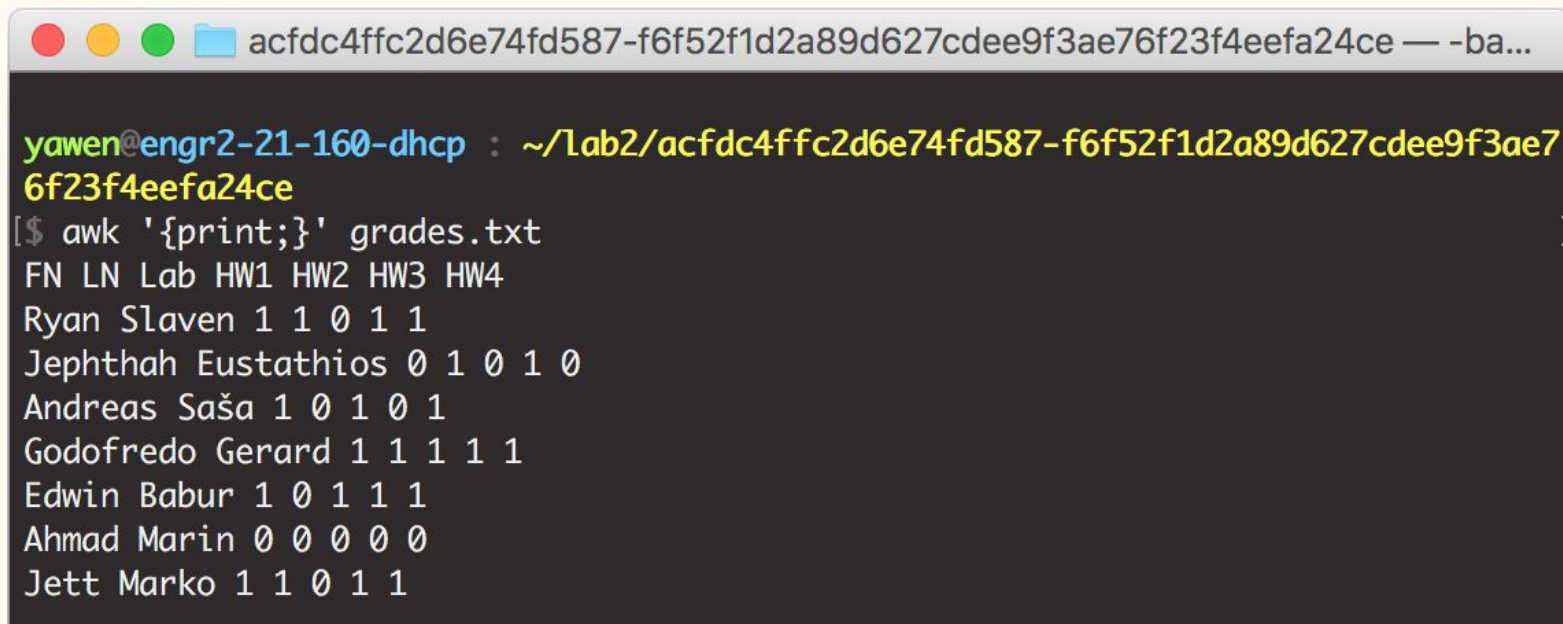
- ❖ a little more difficult, but very powerful text processing command
- ❖ work with lines in a file
- ❖ build-in variables: NR, NF (number of fields), FS (field separator, space by default)

Syntax:

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

# awk (step 7)

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

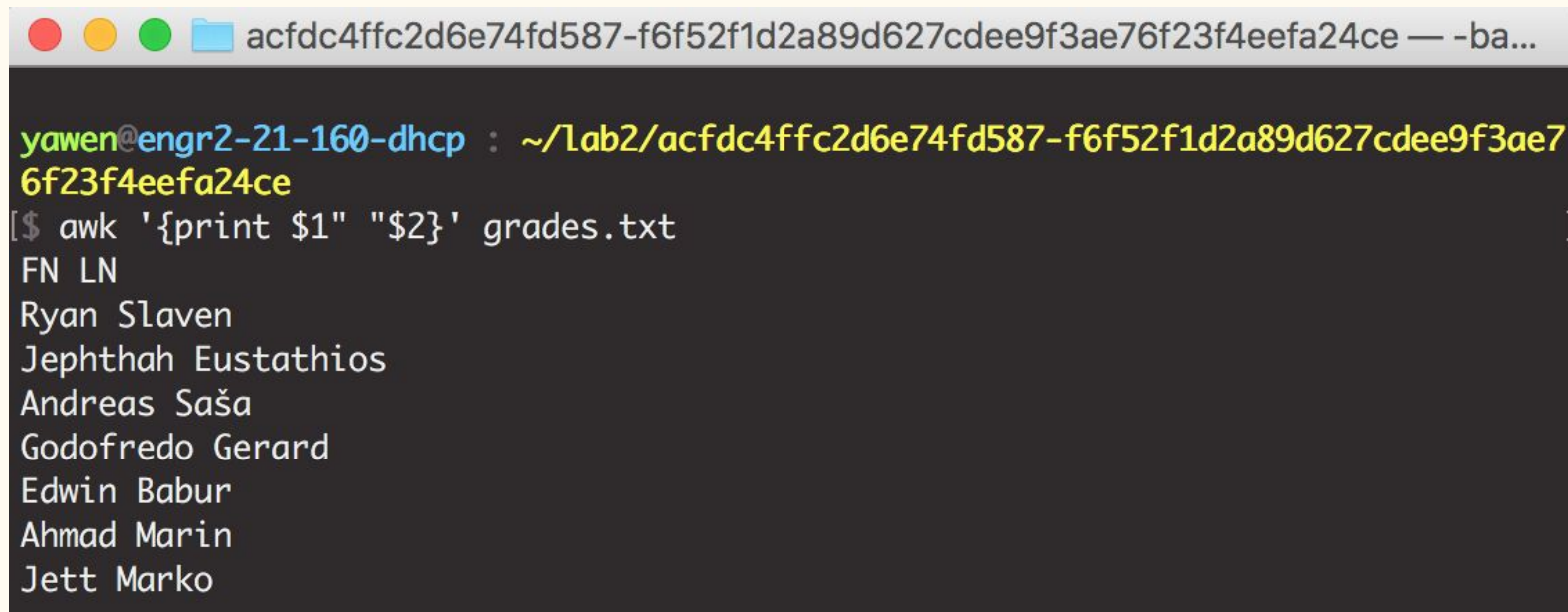


A terminal window with a title bar showing a file path and a window icon. The prompt is `yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce`. The command `awk '{print;}' grades.txt` has been executed, resulting in the following output:

```
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-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce  
[$ 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**

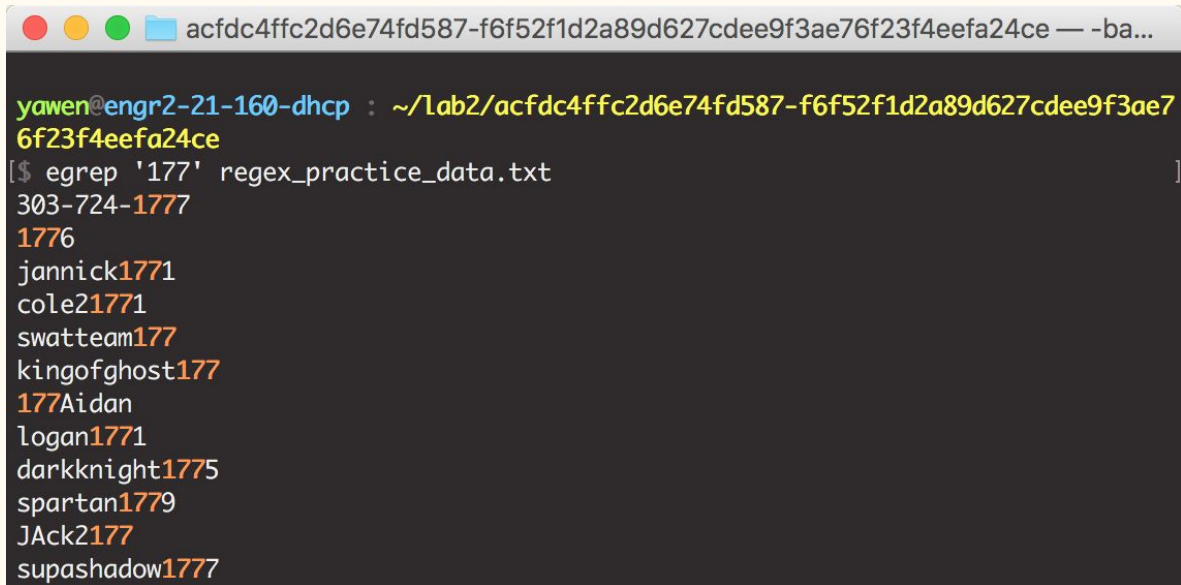


A terminal window with a title bar containing window control buttons and a file path. The prompt shows the user is yawen@engr2-21-160-dhcp in the directory ~/lab2. The command awk 'NR > 1{print \$1" "\$2}' grades.txt has been executed, resulting in a list of names.

```
acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...  
yawen@engr2-21-160-dhcp : ~/lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce  
[$ 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)

- Q: search a word within a file
- A: **egrep '\com' regex\_practice\_data.txt**

A terminal window with a dark background and light-colored text. The window title bar shows a file path and the text '-bash — 107x30'. The prompt is 'yawen@engr2-21-160-dhcp : ~/Lab2/acf4dc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce'. The command '[ \$ egrep '\com' regex\_practice\_data.txt' has been entered. The output lists 15 email addresses, all ending in '.com', such as 'humantvguide@geocities.com' and 'benz@uole.com'.

```
yawen@engr2-21-160-dhcp : ~/Lab2/acf4dc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce
[ $ egrep '\com' regex_practice_data.txt
humantvguide@geocities.com
benz@uole.com
axluigi@geocities.com
garibaldi@uole.com
janet@uole.com
goose70@geocities.com
fuchs@uole.com
singe@uole.com
cog@uole.com
airport@uole.com
mathematics@geocities.com
amj@uole.com
posy@uole.com
syo@geocities.com
velvetfurs@geocities.com
```

# 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)

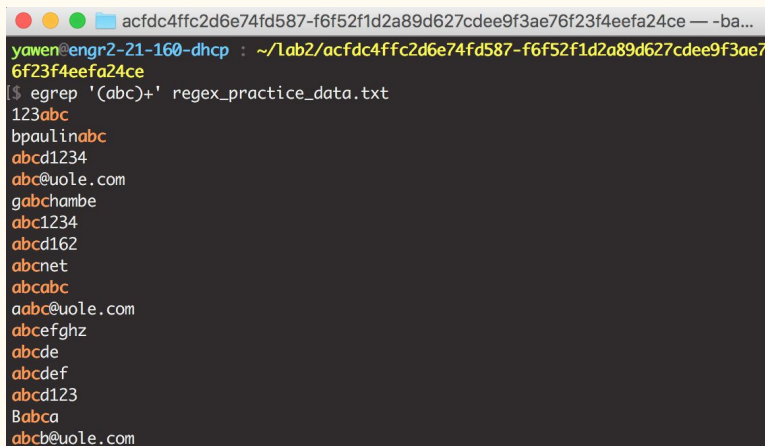
- Q: find lines **beginning** (**ending**) with numbers in file
- A: egrep “**^[0-9]**”  
regex\_practice\_data.txt
- (egrep “**[0-9]\$**”  
regex\_practice\_data.txt)
- 
- Q: find lines with **3** numbers
- A: egrep “**[0-9]{3}**”  
regex\_practice\_data.txt

```
acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -bash — 99x30
[$ 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 — 99x30
yawn@engr2-21-160-dhcp : ~/Lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce
[$ egrep '[0-9]{3}' regex_practice_data.txt
303-724-1777
Jordy3000
v01c0m172
4271
nitrokillaz1220
westking123
bob1734
sunny1414
wired2000
1337dman1337
Wolfy42345
WhiteWolf1423
srbucks5266
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



```
acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae76f23f4eefa24ce — -ba...
yawen@engr2-21-160-dhcp : ~/Lab2/acfdc4ffc2d6e74fd587-f6f52f1d2a89d627cdee9f3ae7
6f23f4eefa24ce
$ egrep '(abc)+' regex_practice_data.txt
123abc
bpaulinabc
abcd1234
abc@uole.com
gabchambe
abc1234
abcd162
abcnet
abcbabc
aabc@uole.com
abcefgzhz
abcde
abcdef
abcd123
Babca
abcb@uole.com
```

# 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,n}**

m Repetitions  
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://regexpr.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](https://github.com/yawenz/csci_3308)