# Regular expressions (Re)

1. A sequence of characters used to find a pattern
2. Extensively used in many computer languages
3. mostly used to find patterns in text -- by using pipe after cmds
4. can be used to validate user input
    - email address, postal/zip codes , phone numbers

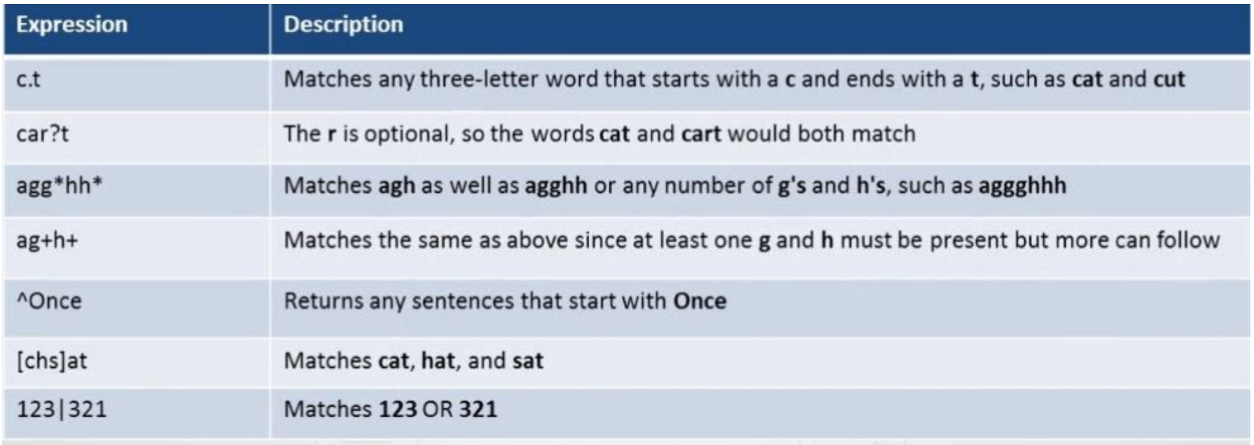
Re is build with metachars

Wildcards
? --> 0 0r 1 of pervios element
* --> 0 or more of pervious element
+ --> 1 or more
. means 1 char at that location

Anchors
^ --> beginning of the string
$ --> ending of the string

Containers
``{} --> used to group or contain sub expressions
``[] --> used to select a particular set of items

others
``or --> verical bar/pipe symbol (|)
``\ --> escape the metachars if they are required for the pattern



| Expression | Description |
|---|---|
| c.t | Matches any three-letter word that starts with a **c** and ends with a **t**, such as **cat** and **cut** |
| car?t | The **r** is optional, so the words **cat** and **cart** would both match |
| agg*hh* | Matches **agh** as well as **agghh** or any number of **g's** and **h's**, such as **aggghhh** |
| ag+h+ | Matches the same as above since at least one **g** and **h** must be present but more can follow |
| ^Once | Returns any sentences that start with **Once** |
| [chs]at | Matches **cat**, **hat**, and **sat** |
| 123\|321 | Matches **123** OR **321** |

# Filtering and formatting

Normally used after the pipe (|)
grep --> line filtering
awk --> programmatic filtering and string manipulation

**grep** examples :

```
cat greptest.txt | grep -i test
``case insensitive filter for the text "test"
ls -l | grep -E 'j.+s'
`` match items with the letter "j" , any number of chars . then "s"
grep "^[0-9]"*
``Begins with atleast one numeric
grep -v "^[0-9]"
``shows lines do not begin with a number
```

**awk** examples

```
ls -l | awk '{print $1}'
`` shows only permission column  from the long listing
ls -l | awk {print$9,"->",$3}'
``shows filename,-> followed by the owner name
``since column is last one , $NF (number of fields) could have been used instead
awk '$1>2000' invoice.txt
``check column one is the listed file for amounts greater than 2000
```

# Background process

```
service <toolname> status
```

```
find /
```

start a backgroudprocess

```
``send output to root/output.txt
find / > /output.txt &

#to see what jobs are running
jobs

#to stop the jobs
jobs -r 1

#readinput.sh

read -p "write something" var
echo "you have entered  " $var

./readinput.sh

fg
bg
```

# File editing using sed

*Cat test.txt*

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a
powerful.
```

### Replacing or substituting string

The below simple sed command replaces the word “unix” with “linux” in the file.

```
$sed 's/unix/linux/' test.txt
```

### Replacing the nth occurrence of a pattern in a line

The below command replaces the second occurrence of the word “unix” with “linux” in a line.

```
$sed 's/unix/linux/2' test.txt
```

**Replacing all the occurrence of the pattern in a line**

The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.

```
$sed 's/unix/linux/g' test.txt
```

**Replacing from nth occurrence to all occurrences in a line**

The following sed command replaces the third, fourth, fifth… “unix” word with “linux” word in a line.

```
$sed 's/unix/linux/3g' test.txt
```

**Parenthesize first character of each word :**

This sed example prints the first character of every word in paranthesis.

```
$ echo "Welcome To The World" | sed 's/\(\b[A-Z]\)/\(\1\)/g'
```

Output:

```
(W)elcome (T)o (T)he (W)orld
```

**Replacing string on a specific line number :** You can restrict the sed command to replace the string on a specific line number. An example is

```
$sed '3 s/unix/linux/' test.txt
```

**Duplicating the replaced line with /p flag :** The /p print flag prints the replaced line twice on the terminal. If a line does not have the search pattern and is not replaced, then the /p prints that line only once.

```
$sed 's/unix/linux/p' test.txt
```

**Printing only the replaced lines :** Use the -n option along with the /p print flag to display only the replaced lines. Here the -n option suppresses the duplicate rows generated by the /p flag and prints the replaced lines only one time.

```
$sed -n 's/unix/linux/p' test.txt
```

**Replacing string on a range of lines :** You can specify a range of line numbers to the sed command for replacing a string.

```
$sed '1,3 s/unix/linux/' test.txt
```

**Deleting lines from a particular file :** SED command can also be used for deleting lines from a particular file. SED command is used for performing deletion operation without even opening the file

```
Examples:

1. To Delete a particular line say n in this example

Syntax:

$ sed 'nd' filename.txt

Example:

$ sed '5d' filename.txt

2. To Delete a last line


Syntax:
```

```
$ sed '$d' filename.txt
```

**3. To Delete line from range x to y**

**Syntax:**

```
$ sed 'x,yd' filename.txt
```

**Example:**

```
$ sed '3,6d' filename.txt
```

**5. To Delete from nth to last line**

**Syntax:**

```
$ sed 'nth,$d' filename.txt
```

**Example:**

```
$ sed '12,$d' filename.txt
```

**6. To Delete pattern matching line**

**Syntax:**

```
$ sed '/pattern/d' filename.txt
```

**Example:**

```
$ sed '/abc/d' filename.txt
```

# File system commands

mkdir – Creates a directory

rmdir – Deletes a directory

ls – Lists contents of given path

cat – Read from given file and output to STDOUT or given path

find – Search for a given file (find <path> -name <filename>)

chmod – Change mode/permissions

cp - Copy files (cp sourcefile destfile)

mv – Move/rename files (mv oldname newname)

scp – Secure copy (Remote file copy) (scp <filename> <host>:<path>)

I/O Commands

echo – To print to stdout

read – To obtain values from stdin

I/O Redirection

> - Output to given file

< - Read input from given file

>> - Append output to given file