SOFTENGER
Strive to Excel

Regular Expressions

Grep

- Searching for Pattern grep scans its input for a pattern
- It can display:
  - The selected pattern
  - The line number
  - The filenames where the pattern occurs.
- Syntax :
  - grep options pattern filename(s)
- Example :

```
grep “sales” emp.lst
grep president emp.lst
who | grep pratham > me.txt
grep “director” emp1.lst  emp2.lst
grep ‘jai sharma‘ emp.lst
```

Options	***Significance***
-i	Ignore case for matching
-v	Doesn't display lines matching expression
-n	Display line numbers along with lines
-c	Displays count of number of occurences
-l	Displays list of filenames only
-e exp	Specifies expression exp with this option. Can use multiple times. Also used for matching expression beginnig with a hypen
-x	Matches pattern with entire line
-f file	Takes patterns from file , one per line
-E	Treats pattern as an extended regular expression (ERE)

- Example :

```
grep  -i ‘agarwal’ emp.lst
grep –v ‘director’ emp.lst > otherlist
grep –n ‘marketing’ emp.lst
grep –c ‘director’ emp1.lst
grep –c ‘director’ emp*.lst
grep –l ‘manager’ *.lst
grep –e “Agarwal” –e “aggarwal” –e “agrawal” emp.lst
grep –f pattern.lst emp.lst
```

- A regular expression is a string that describes or matches a set of strings, according to certain syntax rules.
- Regular expressions are used by many text editors and utilities to search and manipulate bodies of text based on certain patterns.
- Many programming languages support regular expressions for string manipulation.
- Example, Perl have a powerful regular expression engine built directly into their syntax.
- The set of utilities (including the editor sed and the filter grep) provided by Unix distributions were the first to popularize the concept of regular expressions.
- A regular expression, often called a pattern, is an expression that describes a set of strings. They are usually used to give a concise description of a set, without having to list all elements.

Basic Regular Expressions (BRE)

SOFTENGER
Strive to Excel

BRE Character Subset

Symbols or Expressions	Matches
*	Zero or more occurences of previous character
g*	Nothing or g , gg , ggg , etc
.	A single character
.*	Nothing or any number of characters
[p q r]	A single character p , q , r
[c1 - c2]	A sinlge character within the ASCII range represented by c1 and c2

Symbols or Expressions	Matches
[1 - 3]	A digit between 1 and 3
[^ p q r]	A single character whivh is not p , q , r
[^ a - z A - Z]	A nonalphabetic character
^pat	Pattern pat at the beginning of the file
pat$	Pattern pat at the end of the file
^bash$	bash as the only word in line
^$	Lines containing nothing

Symbols or Expressions	Matches
ch+	Matches *one* or *more* occurrences of character *ch*
ch?	Matches *zero* or *one* occurrences of character *ch*
exp1|exp2	Matches *exp1* or *exp2*
GIF|JPEG	Matches GIF or JPEG
(x1|x2)x3	Matches *x1x3* or *x2x3*
(lock|ver)wood	Matches lockwood or verwood

Example :

```
grep “[aA]g[ar][ar]wal” emp.lst
grep “[aA]gg*[ar][ar]wal” emp.lst
grep “j. *saxena” emp.lst
grep “^2” emp.lst
grep “7…$” emp.lst
grep “^[^2]” emp.lst
ls –l | grep “^d”

grep  -E “[aA]gg?arwal” emp.lst
grep –E ‘sengupta|dasgupta’ emp.lst
grep –E ‘(sen|das)gupta’ emp.lst
```

sed

- sed stands for stream editor, works as a filter processing input line by line
- sed is a non-interactive editor used to make global changes to entire files at once
- An interactive editor like vi would be too cumbersome to try to use to replace large amounts of information at once
- sed command is primarily used to substitute one pattern for another

- Typical Usage of sed:
- edit files too large for interactive editing
- edit any size files where editing sequence is too complicated to type in interactive mode perform “multiple global” editing functions efficiently in one pass through the input
- edit multiples files automatically good tool for writing conversion programs

Syntax:

sed –e ‘command’ file(s)
sed –e ‘command’ –e ‘command’ … file(s)
sed –f scriptfile file(s)

^ matches the beginning of the line
$ matches the end of the line
. matches any single character
(character)* match arbitrarily many occurences of (character)
(character)? Match 0 or 1 instance of (character)
[abcdef] Match any character enclosed in [] (in this instance, a b c d e or f)
[^abcdef] Match any character NOT enclosed in [] (in this instance, any character other than a b c d e or f)

SUBSTITUTE	s
DELETE	d
APPEND	a
CHANGE	c
INSERT	i

SUBSTITUTE(s)

[address1[, address2]]s/pattern/replacement/[flags]

Flags:

n	replace nth instance of pattern with replacement
g	replace all instances of pattern with replacement
p	write pattern space to STDOUT if a successful substitution takes place
w	file write the pattern space to file if a successful substitution takes place

- If one address is given, then the substitution is applied to lines containing that address.

- An address can be a regular expression enclosed by forward slashes /regex/ , or a line number.

The $ symbol can be used to denote the last line.

Ex 1.

```
sed 's/Tx/Texas/' foo
```

replaces Tx with Texas in the file foo

Ex 2. cat file

I have three dogs and two cats

sed -e 's/dog/cat/g' -e 's/cat/elephant/g' file

I have three elephants and two elephants

Ex 3. cat file

the black cat was chased by the brown dog.
the black cat was not chased by the brown dog

sed -e '/not/s/black/white/g' file

the black cat was chased by the brown dog.
the white cat was not chased by the brown dog

DELETE(d)

[address1[, address2]]d

sed -e 6d foo — deletes line 6.

sed -e '1,10d' foo — delete lines 1-10 from the file foo

sed '11,$d' foo — A dollar sign ($) can be used to indicate the last line in a file. For example, delete lines 11 through the end of myfile.

sed -e /^$/d foo — deletes all blank lines

sed '/^Co*t/,/[0-9]$/d' foo — deletes from the first line that begins with Cot, Coot, Cooot, etc through the first line that ends with a digit

cat file

line 1 (one)
line 2 (two)
line 3 (three)

sed -e '/^line.*one/s/line/LINE/' -e '/line/d' file

Output:

LINE 1 (one)

- sed can also delete lines based on a matching string. Use /string/d For example, sed '/warning/d' log deletes every line in the file log that contains the string warning.
- To delete a string, not the entire line containing the string, substitute text with nothing. For example, sed 's/draft//g' foo removes the string draft everywhere it occurs in the file foo.