```
cat words.txt                          Just print the whole file

---- Simple Use & Flags ----

grep -E 'Sue' words.txt                all lines where 'Sue' occurs
grep -E -c 'Sue' words.txt             counts number of matched lines
grep -E -n 'Sue' words.txt             now with line numbers
grep -E -o 'Sue' words.txt             just print the match - 'Sue' each on different line
grep -E -v 'Sue' words.txt             all lines where 'Sue' does NOT occur

---- Anchors ----

grep -E '^Sue' words.txt               lines that START with 'Sue'
grep -E 'Sue$' words.txt               lines that END with 'Sue'

---- Special characters - ( ?  \  .  [  ]  ^  $ * ) ----

grep -E '*' words.txt                  not really what you expect
grep -E '\*' words.txt                 Special characters (*) must be '\'

---- Quantifiers ----

grep -E 'a b. c' words.txt             '.' any ONE character
grep -E 'a b.* c' words.txt            * = 0 or more (of the previous character)
grep -E 'a b.? c' words.txt            ? = 0 or 1 (of the previous character)
grep -E 'a b.+ c' words.txt            + = 1 or more (of the previous character)
grep -E 'a b.{3} c' words.txt          {3} exactly 3 (of the previous character)
grep -E 'a b.{2,3} c' words.txt        {2,3} between 2 and 3 (of the previous character)

---- Matches are greedy - They will match as much as possible ----

grep -E -o 'TH.*S' words.txt           Matches are greedy
grep -E -o 'TH[^S]*S' words.txt        [^S] everything that is NOT 'S'
grep -E -o '<a.*>' words.txt           probably not what you want
grep -E -o '<a[^>]*>' words.txt        pick out the 'anchor' tag

---- Ranges of Characters [] ----

grep -E 'a b[a-z]{2} c' words.txt      [a-z] - any lower case letter
grep -E 'a b[0-9]{2} c' words.txt      [0-9] any number
grep -E 'a b[0-9,a-z]{3} c' words.txt     any number or letter
grep -E '[0-9]{3}-[0-9]{4}' words.txt      Phone Numbers
grep -E '[0-9]{3}-[0-9]{2}-[0-9]{4}' words.txt       SSN

---- Words and boundries ----

grep -E ' Bob ' words.txt              Want single word 'Bob' - Does not always work.
grep -E '\bBob\b' words.txt            Want single word 'Bob' - Use word boundaries
grep -E 'a \b\w+\b c' words.txt        \b\w+\b will match a single word

---- Groupings () and | ----

grep -E '(Bob|Eve)' words.txt            | = or
grep -E '(Eve ){3}' words.txt          The string 'Eve ' 3 times in a row
grep -E '(ba){2}' words.txt            the string 'ba' 2 times in a row
grep -E 'a b(1|a) c' words.txt
grep -E '(From|Subject|Date):' words.txt             () groupings

sed 's/Sue/*SUE*/g' words.txt          simple global subtitution (g = global)
sed 's/[0-9]/(&)/g' words.txt          sub with backreference '&'
sed '1,22d' words.txt                   deleting lines

tr - transliterate

cat words.txt | tr A-Z a-z             quick upper/lowercase exchange
cat words.txt | tr Sa  xy              change all S to x and a to y

uniq - report or omit repeated lines  (-i ignore case, -c count)
uniq -c -i words.txt

cat words.txt | tr A-Z a-z | grep -o -E '[a-z]' | sort | uniq -c -i | sort -n

cat words.txt | tr A-Z a-z | grep -o -E '\b\w{4}\b' | sort | uniq -c -i | sort -n
```

regularexpressions

Anchors

^	Start of line +
\A	Start of string +
$	End of line +
\Z	End of string +
\b	Word boundary +
\B	Not word boundary +
\<	Start of word
\>	End of word

Character Classes

\c	Control character
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word
\xhh	Hexadecimal character hh
\Oxxx	Octal character xxx

POSIX Character Classes

[:upper:]	Upper case letters
[:lower:]	Lower case letters
[:alpha:]	All letters
[:alnum:]	Digits and letters
[:digit:]	Digits
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation
[:blank:]	Space and tab
[:space:]	Blank characters
[:cntrl:]	Control characters
[:graph:]	Printed characters
[:print:]	Printed characters and spaces
[:word:]	Digits, letters and underscore

Assertions

?=	Lookahead assertion +
?!	Negative lookahead +
?<=	Lookbehind assertion +
?!= or ?<!	Negative lookbehind +
?>	Once-only Subexpression
?()	Condition [if then]
?()|	Condition [if then else]
?#	Comment

Note Items marked + should work in most regular expression implementations.

Sample Patterns

([A-Za-z0-9-]+)	Letters, numbers and hyphens
(\d{1,2}\/\d{1,2}\/\d{4})	Date (e.g. 21/3/2006)
([^\s]+(?=\.(jpg|gif|png))\.\2)	jpg, gif or png image
(^[1-9]{1}$\|^[1-4]{1}[0-9]{1}$|^50$)	Any number from 1 to 50 inclusive
(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)	Valid hexadecimal colour code
((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})	8 to 15 character string with at least one upper case letter, one lower case letter, and one digit (useful for passwords).
(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Email addresses
(\<(/?[^\>]+)\>)	HTML Tags

Note *These patterns are intended for reference purposes and have not been extensively tested. Please use with caution and test thoroughly before use.*

Quantifiers

*	0 or more +
*?	0 or more, ungreedy +
+	1 or more +
+?	1 or more, ungreedy +
?	0 or 1 +
??	0 or 1, ungreedy +
{3}	Exactly 3 +
{3,}	3 or more +
{3,5}	3, 4 or 5 +
{3,5}?	3, 4 or 5, ungreedy +

Special Characters

\	Escape Character +
\n	New line +
\r	Carriage return +
\t	Tab +
\v	Vertical tab +
\f	Form feed +
\a	Alarm
[\b]	Backspace
\e	Escape
\N{name}	Named Character

String Replacement (Backreferences)

$n	nth non-passive group
$2	"xyz" in /^(abc(xyz))$/
$1	"xyz" in /^(?:abc)(xyz)$/
$`	Before matched string
$'	After matched string
$+	Last matched string
$&	Entire matched string
$_	Entire input string
$$	Literal "$"

Ranges

.	Any character except new line (\n) +
(a|b)	a or b +
(...)	Group +
(?:...)	Passive Group +
[abc]	Range (a or b or c) +
[^abc]	Not a or b or c +
[a-q]	Letter between a and q +
[A-Q]	Upper case letter + between A and Q +
[0-7]	Digit between 0 and 7 +
\n	nth group/subpattern +

Note *Ranges are inclusive.*

Pattern Modifiers

g	Global match
i	Case-insensitive
m	Multiple lines
s	Treat string as single line
x	Allow comments and white space in pattern
e	Evaluate replacement
U	Ungreedy pattern

Metacharacters (must be escaped)

^	[	.
$	{	*
(	\	+
)	|	?
<	>	

Available free from AddedBytes.com