File Expansion and quoting

*	zero or more characters
?	one of any character
\	escape following special character
[xyz]	matches any of the enclosed
{a,b,c}	matches any of the enclosed strings
‘	literal string
“	variable interpretation permitted

Piping and redirection

<table>
<tr><td>A | B</td><td>output of A is sent to B</td></tr>
<tr><td>A > f</td><td>output of A overwrites file f</td></tr>
<tr><td>A >> f</td><td>output of A appends file f</td></tr>
</table>

Parameter Expansion - ${<varname>}

```
$ x=’toy’
$ echo ${toy}s
toys
```

Command substitution - $(<command>)

```
$ echo “date: $(date)”
```

Arithmetic substitution - $((<expression>))

```
$ echo $(( 3 + 5 ))
8
$ x=1; y=2
$ echo $(( x + y))
3
```

Bash for-loop syntax

```
for x in <list>
do
        <code>
done
```

```
for x in <list; do <code>; done
```

Bash if statement

```
if [[ -r file.txt ]]
then
        <code>
Fi
```

-r	file is readable
-f	file exists
-d	directory exists
-z	test variable is empty

wc – word counter

-l, --lines	line count
-w, --words	word count
-m, --chars	character count

grep – pattern searcher

--help list of options and brief explanations

-c, --count	-A, --after-context
-v, --invert-match	-B, --before-context
-i, --ignore-case	-C, --context
-w, --word-regexp	-h, --no-filename
-l, --files-with-match	-L, --files-without-match

sort

-g, --general-numeric-sort

-n, --numeric-sort

-r, --reverse

-u, --unique

-k, --key=POS

-t, --field-separator=SEP

```
sort -k3g a.tab   #sort 3rd column
```

uniq

-c, --count	count occurrences of each line
-d, --repeated	print only duplicated lines
-u, --unique	print only uniq lines

Regular Expressions

<table>
<tr><td>.</td><td>matches any character except a newline</td></tr>
<tr><td>*</td><td>matches 0 or more of previous character</td></tr>
<tr><td>+</td><td>matches 1 or more of previous character</td></tr>
<tr><td>[xyz]</td><td>matches characters x, y, z</td></tr>
<tr><td>[^xyz]</td><td>matches characters OTHER than x, y, z</td></tr>
<tr><td>^</td><td>anchors match at BEGINNING of line</td></tr>
<tr><td>$</td><td>anchors match at the END of the line</td></tr>
<tr><td>\</td><td>escapes the following special character</td></tr>
<tr><td>(…)</td><td>captures the enclosed sequence</td></tr>
<tr><td>\n</td><td>recalls nth captured sequence</td></tr>
<tr><td>+</td><td>matches 1 or more of previous character</td></tr>
<tr><td>|</td><td>OR</td></tr>
</table>

Sed syntax – by example
sed ‘2d’
sed ‘2,5d’
sed ‘/pattern/,$’
sed ‘/pattern/,/pattern/’
sed ‘[ADRESS]d’
sed ‘[ADDRESS] s/pattern/replacement/’

Sed flags	
g	global
I	ignore case
P	print

AWK – comparison operators	
~	regular expression match
!~	regular expression non-match
==	equals (don’t use ‘=’)
!=	not equals
<	less than
>	greater than
>=	greater than or equal to
<=	less than or equal to
/a/,/b/	TRUE between matches (like in sed)

AWK – logical operators	
||	logical OR
&&	logical AND
!	logical NOT

AWK - builtin variables	
NR	Current line number

less and man	
q	quit
/	find
u	half page up
d	half page down
k	one line up
j	one line down

vi – Command Mode	
:q	quit
:q!	force quit (LOSE UNSAVED INFO)
:w	save
:wq	save and quit
:! <command>	execute a shell command
/	find
i	enter INSERT mode (ESC to exit)
v	enter VISUAL mode (ESC to exit)
Shift-v	enter VISUAL-LINE mode
Ctrl-Shift-v	enter VISUAL-BLOCK mode

vi – Command Mode – basic editing commands	
dd	delete a line
x	delete character or selection
r	replace a character or selection
p	paste deleted/yanked text after cursor
P	paste deleted/yanked text before cursor
u	undo
Ctrl-r	redo

vi – Command Mode - Navigation	
^	move to the beginning of a line
$	move to the end of a line
h	left
j	down
k	up
l	right
{	move to previous blank line
}	move to next blank line

For more info: **vimtutor**