# VI and SED shortcuts

If not already done you can alias the vi command to vim.

sudo yum install vim -y

echo "alias 'vi=vim'" >> ~/.bashrc && ~/.bashrc

Once done you can create a .vimrc file with vi options - for example you can turn hlsearch off by adding the following line:

set nohlsearch

In VI the following are useful

gg - top line

G - bottom line

w - next work

b - previous word

e - end of word

0 - start of line

^ - first character of line

5 G - fifth line

ctrl + f - forward one page

ctrl + b - back one page

H - first line of current screen

M - middle line of current screen

L - last line of current screen

I - insert text at the beginning of line

A - insert text at the end of line

O - add line above current line

o - add line below current line

:0,$d - delete from top to bottom line

6yy - copies 6 lines

6dd - cuts 6 lines

P - paste from line above

p - paste from line below

cw - change word

C - change to end of line

cc - change a whole line

u - undo

ctrl + r - redo last undone change

. - repeat last change

J - append the line below to the current one

:1,30w newfile - save line 1 to 35 to newfile

:1,30 >> newfile - append line 1 to 30 to newfile

:! - allows you to run linux commands

!! - allows you to run linux commands and the result is appended to the current file

:sh - allows you to temporarely exit to a shell on quitting the shell you are popped back into vim

You can search and replace text in VI by using the following syntax:

:[line address]s/<search-text>/<replacement-text>/[g]

* If "line address" is omitted only the current line is searched
* you can specify a range of lines such as 1,35, % indicates all lines which is the same as 1,$.
* the "g" at the end indicates to replace all words being searched, if omitted only the first word on each line will be affected.

search and replace also support pattern matching which support regular expressions that are also used in sed, grep, awk.

sed search and replace is quite similar to VI - sed will never overwrite an existing file it will do the search and replace and then create a new file.  The syntax is:

sed '[line address]s/<search-text>/<replacement-text>/[g]' file > new\_file

# Manage Processes and Systemd

The kill command can be used to kill processes - root can kill any processes and any user can kill its own processes.

kill -l will list all the parameters that can be passed to the kill command.

If you type kill process\_id by default you will be running kill -15 which means "die when you are ready"

kill -KILL or kill -9 both do the same and it means "die no matter what"

in normal circumstance child processes are killed when the parent is killed.

For example say you want to run a process in your terminal and want to carry on working, you can append an & at the end of the command which will run it in backgrouond mode:

sort bigfile > ofile 2> efile &

Once that is started you will be able to see the job with its job id and via its PID. Background jobs run at a slightly lower priority.

In some shells when a user logs out of its shell any background commands which are running will be killed off. the way to prevent that from happening is by using nohup (bash shell

uses the nohup option by default):

nohup sort bigfile > ofile 2> efile &

Foreground jobs can be suspended user ^Z (or using kill -STOP PID), Korn and Bash shell willl allow a user to manage jobs using the jobs command - for example to resume a suspended foregroung job:

fg % job\_number

to resume a suspended job in backgrouong mode:

fg % job\_number

Jobs can be killed via their jobs id or via their PID.

As we have said earlier bg jobs have lower priority, the nice value controls the CPU priority of a job (-20 HIGH, 19 SLOW). The nice command command will alter this value for any new processes, the renice command will alter the value for any currently running processes. Only root can make a process got faster and by default the nice value is 0. For example:

nice -15 prog1

renice 5 -p 1352

You can group commands together with () and these will be run as a single process - for example:

(date; ls -l; echo "end of listing")

When this has been done a new parent process is started. you can check a PID parent process by running ps -ef and looking at the PPID column.

In the Linux world system v which was also known as init has now been replaced by systemd. In order to interact with systemd you now use the systemctl command allows you to:

* Change runlevels
* enable processes
* start/stop and check the status of processes

The configuration for systemd is kept under /etc/systemd

systemd main purpose is to control the boot process therefore you can see how long it took your system to boot and which was the slowest process to start:

systemd-analyze - will show you the boot time

systemd-analyze blame - will tell you which was the slowest process to start

systemd organizes tasks into control groups - so for example to see all of the control groups for httpd you can run:

systemd-cgls | grep httpd

Systemd will deal with units and in turn units will have unit-files which can thought as config file for the services. Unit files are kept in multiple places, however

if you wish to change a particular unit file the best place to do it is under /etc/systemd/system as this location supersedes any other location.

You can run systemctl to see all of the systemd units, mount points loaded etc - or you can run systemctl list-unit-files to show all enabled, disabled and static units.

Other useful commands that can be run are:

hostnamectl

localectl

timedatectl

Systemd also handles a log file which can be used to troubleshoot units:

journalctl

or you can do:

whereis crond

journalctl /usr/sbin/crond

useful journalctl option:

-b since last reboot

--since=today just today

-p <sev level>

-f like tail -f