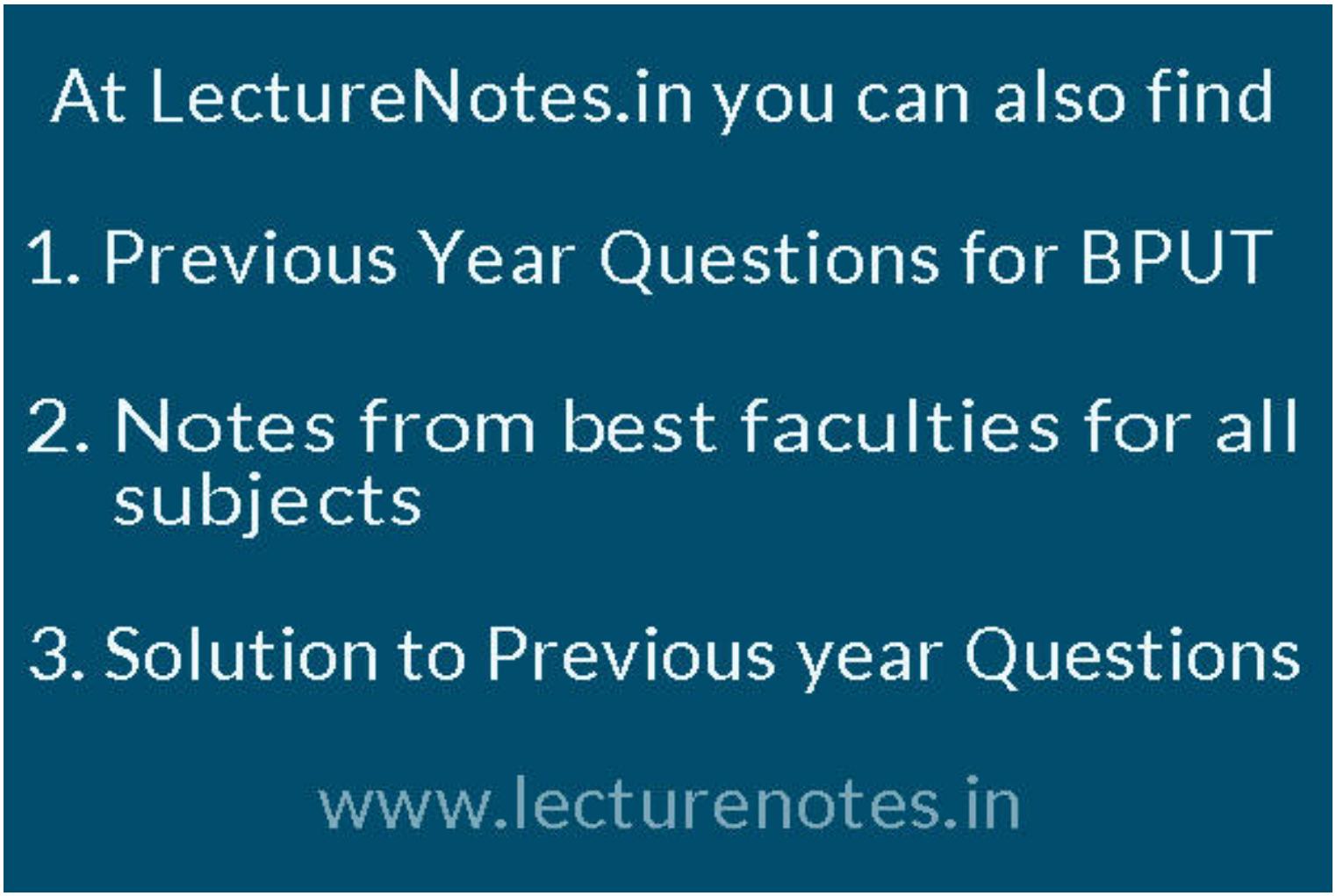
LINUX PROGRAMMING

Contributed By:
Nagi Reddy

Disclaimer



LINUX PROGRAMMING

Topic:

UNIX/ LINUX

Contributed By:

Nagi Reddy

UNIX/LINUX

→ UNIX is a CUI Operating System.

→ LINUX is not just for UNIX wizards. LINUX is a clone of O/S.

→ Linux is the most important achievement of free software, it has been developed for business, education & personal productivity.

→ Everyone has to start somewhere, and Linux administrators and engineers are no exception. If you have purchased this book, I imagine that your goal is to pass the Red Hat exams (RHCSA & RHCE) while acquiring or improving your current Linux skills.

→ These Linux skills and commands are all essential for knowing how to work with linux, not just Red Hat.

* Operating System: (O/S)

→ Operating System is a interface between User & computer (or) It is a System Software.

→ The operating System is the most important program that runs on a computer. Every general-purpose computer must have an operating System to run other programs.

→ Operating Systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

→ It is classified into two types.

① Single User Systems.

② Multi User Systems.

① **Single User Systems:** provides a platform for only one user at a time. They are popularly associated with Desk Top operating System which run on standalone Systems where no user accounts are required.

Example: DOS

② **Multi User Systems:** More than one user can access same System resources (cpu, applications, memory, printers....etc) at the same time known as multiuser.

Example: UNIX, LINUX

* Features of UNIX/LINUX:

(a) **Multiuser:** A multi-user operating System allows more than one user to share the same computer system at the same time.

(b) **Multi Tasking:** More than one program can be run at a time. The main concept of multitasking is maximum utilizing cpu resources.

(c) **Open System:** The UNIX is open source code. i.e Any user can modified unix open source code according there ideas and requirements.

→ Using UNIX open source code

Sun Micro Systems + adding additional features = SunSolaries

IBM + " = IBM-AIX

HP + " = HP-UX

Santa Cruz + " = Sco-unix

Silicon Graphics + " = IRIX

Micro Soft + " = Xenix

→ Any operating System developed based on UNIX open source code known as flavors of UNIX.

→ The Linux was given to GPL (General public Lience) Organized by GNU.

→ Linus Torvalds, who was then a student at the university of Helsinki in Finland, developed Linux in 1991. Linux is also open System

Distributors of Linux:

Red Hat	Suse	Ubuntu	Puppy	Slakware
Centos	OEL	Fedora	White box	Mandrake

(d) **Security:** One of the most valued advantages of Linux over the other platforms lies with the high security levels it ensures. Every linux user is happy to work in a virus-free environment and use the regular virus-prevention time needed when working with other operating systems for other more important tasks.

→ UNIX/LINUX has given two levels of securities.

(i) **System Level Security:** Its controlled by System Administrator.

(ii) **File Level Security:** Its controlled by owner of the file.

(e) **portability:** portability means Independent of hardware & processor

(f) **Communication:** The main concept of communication facility Exchanging of information or files from one user account to another user account.

(g) **programming facility:** UNIX o/s provides Shell. Shell works like a programming language. It provides commands and keywords.

Script language	programming language
1. It is an interpreter based language.	1. Compiler based language.
2. Enterpreter converts high level instructions into lowlevel instructions line by line.	2. The whole program in a single shot into machine language.
3. Doesn't create .exe files	3. Create .exe files.
4. No need to compile the program.	4. Need to compile the program.
5. It takes less lines of code	5. Takes numerous lines of code.
6. Reducess cost of maintainance	6. Encreses cost of maintainance.

(h) **Help facility:** It is the beautiful feature of UNIX/LINUX operating Systems. Dont know the information about given command just gothrough the help line.

Example: # man <Command name> ↵

(or)

info <Command name> ↵

(or)

<command name> --help ↵

* Architecture of O/S:

User:- User is nothing but an individual who uses the available hardware & software resources.

Shell: It is a command line interpreter. The shell access request from user and the checks command existence, if the command exist then it converts into kernel understandable language and it send the given request to kernel. The shell access interface between user and kernel.

Types of Shells:

Shell name	Developed by	prompt	Interpreter name
Bourne Shell	Stephen Bourne	$	Sh
Bash Shell	Stephen Bourne	$	bash
Korn Shell	David korn	$	ksh
Z Shell	paul	$	Zsh
C Shell	Bill joy	%	csh

Note: The advanced version of Bourne shell is Bash shell. Bash means Bourne again Shell.

Default Shell name	Flavor name
Bash Shell	Linux
Bourne Shell	Sco-unix, Solaries, Hp-UX
Korn Shell	IBM-AIX
C Shell	IRIX

Kernel: The kernel is the heart of the operating System.

→ The kernel is responsible for interacting with the hardware and producing output to the screen.

→ It handles the process, memory, file, device and network management for the operating System.

→ Linux is truly just the kernel.

* Difference between UNIX & WINDOWS:

UNIX	WINDOWS
1. It is GUI	1. It is GUI
2. It is Multiuser and Multitasking O/S	2. WINDOWS also Multiuser and Multitasking O/S.
3. To boot UNIX O/S, 2 MB RAM is Enough.	3. 12MB RAM is required.
4. UNIX is process based concept	4. It is Thread based concept.
5. In UNIX, Any user process is killed it will not Effect to others.	5. It Effects to all.
6. Unlimited users working on the Server.	6. Limited users working on the server.
7. UNIX is open System	7. It is closed System.
8. It is portable O/S.	8. Not portable.
9. No down time	9. Down time is there.

* File System:

→ It is method of storing the data in an organized fashion on the disk. Every partition on the disk except MBR and Extended partition should be assigned with some file system in order to make them store the data. file system is applied on the partition by formatting it with a particular type of file system.

Types of file systems:

(a) Disk file systems:

→ A disk file system is a file system designed for the storage of files on a data storage device, most commonly a disk drive, which might be directly or indirectly connected to the computer.

Ex: FAT, FAT32, NTFS, CDFS, HFS, ext2, ext3, ISO 9660.

(b) Network file systems:

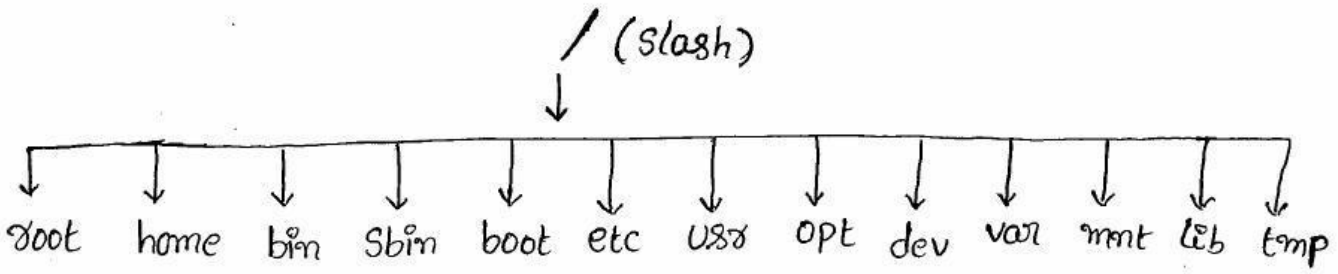
→ A Network file system is a file system that acts as a client for a remote file access protocol, providing access to files on a server.

Ex: DFS, NFS, SMB, FTP

(c) Virtual file system:

→ The purpose of VFS is to allow client applications to access different types of concrete file systems in a uniform way. It can be used to bridge the differences in windows, Mac OS and Unix file System, so that applications can access files on local file systems of those ~~file systems~~ types without having to know what type of file System they are accessing.

⇒ File System Hierarchy System:



/ : This is top level working directory. It is parent directory for all other directories. It is called as "ROOT" directory. It is represented by forward slash(/)

root: It is home directory for root user(super user). It provides working environment for root user.

home: It is home directory for other users. It provides working environment for other users (Except root).

bin: (Binary files): It contains commands used by all users.

Sbin: (Super user binary files): It contains commands used by only super user (root).

boot: It contains system bootable files, bootloader information, Kernel related information for Linux.

etc: It contains all system configuration files.

/etc/hosts, /etc/resolv.conf

usr: By default Soft wares are installed in /usr directory.

(UNIX Sharable Resources).

opt: It is a optional directory for users. It contains third party softwares.

dev: It contains all device files information. Similar to device manager of windows. In UNIX/LINUX Every device treated as a file.

var: It is containing variable files information like mails, poring, log files.

mnt: It is default removable media working directory. It is empty by default.

lib: It contains library files which are used by OS. It is similar to dll files of windows. Library files in linux are SO (Shared Object) files.

tmp: It contains temporary files information.

media: It contains all of removable media like CD-ROM, pendrive.

proc: It contain process files.
Its contents are not permanent, they keep changing
Its also called as Virtual Directory.
Its file contain useful information used by OS.
like /proc/meminfo ---
/proc/cpuinfo ---

—x—

* Basic commands:

→ root user prompt.

$ → User working prompt.

$logname → Displays current user name.

$pwd → present working directory.

$date → Display current date & time.

$cal → Displays current month calender.

$cal 2020 → particular year total months.

$cal 04 2020 → 2020 year 04th month calender.

$who → To display the information about all the users who have logged into the System.

$whoami → It displays current user name.

$finger → It displays complete information about all the users who are logged in.

$uptime → How long server up & running, how many users connected and load avg time.

$which (or) $whereis → Given command location

Ex: which date↵

$tty → Terminal position.

$df → Displays disk free size

$du → Disk usage information

$clear → To clear the screen.

* Creating files:

⇒ **cat (concatenate):** It is used to create a file and display, appending the contents of a file.

→ **To create a file:**

```
$cat > filename ↵
Hello World
Ctrl+d (To save the file)
```

→ **To display the content of the file:**

```
$ cat < filename ↵
(or)
$cat filename ↵
```

→ **To append the data in the Existing file:**

```
$ cat >> filename ↵
- - - - -
- - - - -
Ctrl+d (To save)
```

⇒ **Touch:** To create multible files but all are Empty.

Syn: $touch file1 file2 file3 - - - -filen

Ex: $touch file1 file2 file3 ↵

⇒ **ls:** Displays the contents of a directory.

Syn: $ ls [options]

Options:
- -r → Reverse
- -a → hidden
- -R → Recursively
- -i → inode
- -l → long list
- -h → human readable

⇒ **mkdir**: creates a directory.

Syn: mkdir <Dirname>

$mkdir Linux ↵

→ To create multible directories:

$mkdir Dir1 Dir2 Dir3 --- Dirn ↵

→ To create a Nested directory:

$mkdir -p world/asia/India/ap/Hyd ↵

↳ parent

→ To check: $ tree world ↵

(or)

$ ls -R ↵

⇒ Navigation commands:

cd: changes the current location.

cd .. → To go one level back

cd ../.. → To go two levels back

cd → To change user's home directory.

Ex: $cd world ↵

$ls ↵

* Note: The trailing slash (/) is optional when you're using the cd command. It indicates that the name being specified is a directory.

Ex: $cd Documents/ ↵

⇒ **cp:** copies files or directories from one location to another.

Syn: cp [options] source Destination

-R → copies recursively
-f → copies forcefully
-v → provides verbose output.

Ex: $cp file1 file2 ↵ one file to another.

$cp file1 file2 Documents ↵ Multible files into Directory.

$cp -R Documents unix ↵ one directory to another.

$ cp -rvf Documents unix world/asia/india/ap/Hyd ↵

$ cp /var/log/messages . ↵ (.) represents current location.

⇒ **mv:** Moves or renames files and directories.

Syn: mv [options] SOURCE DEST

-v → verbose

→ Rename the file by specifying the file name & new name of the file.

#mv messages messages.bak ↵

→ move it to the test directory for safekeeping:

#mv messages.bak test/ ↵

$ls test ↵

⇒ **rm:** Deletes files or directories

Syn: rm [options] FILE

-i → interactive
-r → Recursively
-f → forcefully

→ Delete the messages.bak file:

$cd test ↵

$rm -i messages.bak ↵

→ Delete the test directory:

$cd .. ↵

$rm -rf test/ ↵

⇒ file: Displays the type of a file

Syn: file <filename>

Ex: $file test1 ↵

test1: Empty

$ file /etc/passwd ↵

passwd: ASCII test.

* Meta characters (or) Wild card characters:

(a) * : It matches zero (or) more characters in the given file.

Ex: $ls a* ↵ Displaying files start with 'a'.

$ls i*g ↵ Start with 'i' End with 'g'

$ls *g ↵ Listout End with 'g' only

$rm i* ↵ removes Start with i

$rm *g ↵ removes End with g only

$ cp a* Documents/ ↵ copies start with a

$ cp -rf i*g Documents/ ↵

$ cp -ruf * /backup ↵ copies current directory all files

(b) ? : It matches any single character in the given file.

Ex: $ ls ? ↵ Display single character files.

$ ls ?? ↵ Two character files

$ ls a??? ↵ List four character files but first one is 'a'.

$rm ?? ↵ Removes two character files.

$cp ??? Documents/ ↵ Copies three character files.

(c) [ ] : It matches any single characters in the given list.

Ex: $ ls [aeiou] ↵ Displays given matching character files.

$ls [aeiou]* ↵ Displays start with a, e, i, o, u files.

$rm [aeiou]* ↵ Removing start with a, e, i, o, u

$cp [aeiou]* unix/ ↵ copying start with a, e, i, o, u

(d) [-] : It matches any single characters in the given range.

Ex: $ ls [a-f]* ↵ Displays start with a, b, c, d, e, f

$ ls [a-f, o-v]* ↵ Display start with a-e & o-v.

$rm [a-f]* ↵ Removes a-f

$cp [a-f, 1-9]* unix/ ↵ copying files a-f & 1-9

* Using a Text Editor:

Being able to use a text Editor is probably one of the most critical skills to have as a System administrator. you constantly need to Edit config files, write scripts or make changes to System files.... all of which require you to use a text Editor.

→ The three most popular editors available today include

vi (or) vim : Text editor with great flexibility.

emacs : Similar to vi, an advanced text Editor with many features.

nano : A basic text Editor for quick editing

⇒ Vi (or) Vim Editor:

→ Using this Editor to create new files, open the files and modifying the data into a Existing files.

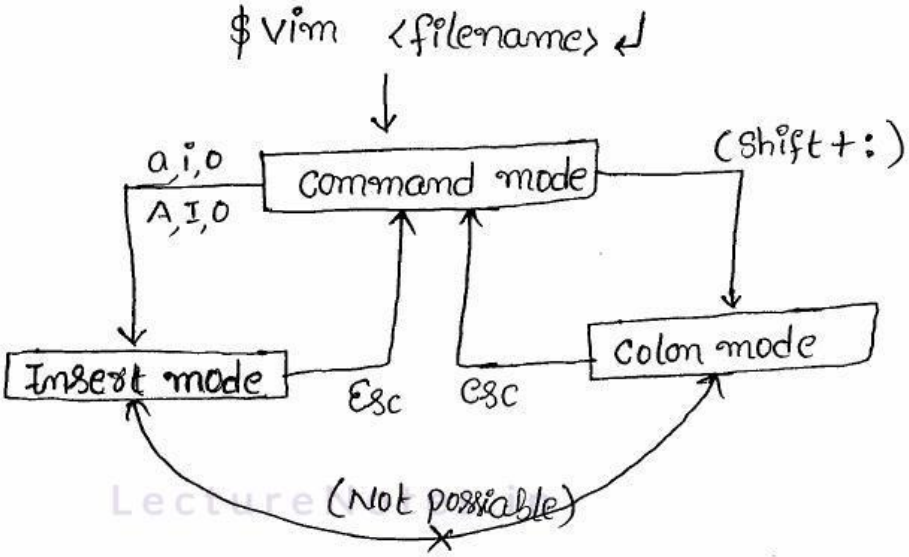
→ The vi Editor is most popular.

→ It has three modes : (a) Command mode
(b) Insert mode
(c) Execution (or) colon mode.

→ By default mode is command mode.

Syn: Vim [arguments] [file]

Arguments: -R → opens a file in read-only mode
-o → open two files at a time
+ → Starts at the end of the file
+<num> → Starts at line <num>



Insert mode options:

i → To begin insert mode at the current cursor position.

I → To insert at the beginning of the current line.

a → To append to the next word's letter.

A → To append at the end of the line.

o → To insert a new line below the cursor position.

O → To insert a new line above the cursor position.

Commands for command mode:

e → Moves to the end of a word.

b → Moves to the beginning of a word.

$ → Moves to the end of a line.

^ → Moves to the begining of a line.

H → Moves to the first line onscreen.

M → Moves to the Middle line onscreen.

L → Moves to the last line onscreen.

x (nx) → Deletes current character.

dd (ndd) → Deletes the current line.

dw (ndw) → Deletes current word.

yy (nyy) → yanks (copies) the current line.

p → paste below the cursor line.

P → paste above the cursor line.

u → Undo the last action.

gg (ngg) → go to beggining of the file.

G → End of the file.

w (n) → To move the cursor forward, word by word.

b (n) → To move the cursor backword, word by word.

Ctrl + f → To forward one page.

Ctrl + b → To backword one page.

/ → To search a word in the file.

n → find next occurence of search word.

N → find previous occurence of search word.

. → Repeat last command action.

Commands for last line mode:

:q → To quit without saving

:w → To save the changes

:wq → To save & quit

:wq! (or) :x } → To save & quit with forcefully.

: set nu
(or) } → To setting line numbers.
: se nu

: set nonu
(or) } → To remove line numbers.
: se nonu

: n → Jumps to line n

: $d → To delete last line

: ! <unixCmd> → To execute unix cmds.

: x → To give password to the file and remove password.

: /string/ → To search a word in the file.

→ **To find & Replace:**

: % s/root/dog/ ↵ To replace string "dog" for the first instance on a line.

: % s/root/dog/g ↵ for each instance of a line.

: % s/root/dog/gi ↵ To ignore casesensitive

: % s/root/dog/gc ↵ ask for confirmation.

→ **Executing unix commands in vi:**

Any unix command can be executed from the vi command line by typing an "!" before the unix command.

Ex: : ! pwd ↵

: r ! date ↵ Reads the results from the date command into a new line following the cursor.

: r ! cat file1 ↵

→ I want to copy 1,4 lines to paste after 10th line:

:1,4 co 10 ↵

→ I want to move 3,7 lines after 8th line:

:3,7 mo 8 ↵

→ I want to copy 1,30 lines create a new file:

:1,30 w test1 ↵

→ I want to append the data into a existing file:

:8,20w >> test1 ↵

→ I want to insert end of the line (or) we require line

:r /etc/passwd ↵

Managing two files at time:

$ vim -o file1 file2 ↵

(or)

$ vim file1 file2 ↵

Options:

:n → Edit next file (file2)

:rew → Rewind to the file (file1)

(or)

→ To move one file to another file (ctrl + w)

↳ press two times

— x —

* I/O Redirection :-

→ Sometimes you need to use the output from a command more than once. To accomplish this, you can redirect the output of commands using some neat command-line tricks.

→ There are also a few characters you can use to direct or redirect output of commands. These characters are

>	Directs output to a file or device (override if the file Exists)
<	Directs input from the file or device.
>>	Appends output or text to a file (Creates if the file doesn't Exist).
|	pipes the output of one command to another.
&&	Combines commands.

(or)

STDIN – file Description (FD) – 0

STDOUT – File Description (FD) – 1

STDERR – File Description (FD) – 2

Examples : ① . $cat file > backup ↵

$cat backup ↵ To verify

② $Cat > test2 < test1 ↵ Input from the test1

$cat test2 ↵ To verify

③ $cat test1 test2 test3 2> Error1 ↵
↳ file not Exist
$cat Error1 ↵ To verify

④ $cat sample test test3 > out-file 2>> Error1 ↵
↳ file not Exist
$cat out-file ↵
$cat Error1 ↵ } → To verify

⇒ **echo:** outputs or displays a string.

Ex: $echo "this is some sample text" ↵

→ To output some text to a file:

$echo "this is some sample text" > file_example ↵

~~$echo~~ $cat file_example ↵ To verify.

⇒ **cut:** Divides a string or output.

Syn: cut [option] file

-d Specifies a delimiter
-f Displays a particular field.
-c Displays a character.

→ Displays the third field of the text using space as a delimiter:

$cut -d " " -f3 file_Example ↵

→ Displays third & fourth fields:

$cut f3,4 file_Example ↵

→ Displays 1st to 5th characters:

$cut -c 1-5 file_example↵

⇒ paste: To join two or more files horizontally by using delimiters.

→ To join two files horizontally:

$paste states capitals↵

AP Hyd
MP Bopal
KN Banglore

→ To join two files using delimiter:

$paste -d ":" states capitals↵

AP: Hyd
mp: Bopal
KN: Banglore

$paste -d ":" states capitals > Example↵

⇒ WC: provides a word or line count.

Sym: wc [options] file

-l Lines
-w words
-c characters

$wc Example↵ Displays lines, words and characters.

$wc -l Example↵ only lines

$wc -w Example↵ only words.

⇒ **Diff:** Displays different lines between two files.

$diff file1 file2 ↵

⇒ **Cmp:** It compares two files character by character.

$cmp file1 file2 ↵

Note: If files are same it doesn't return any output otherwise it displays line numbers and character position.

⇒ **tr:** It translate character by character.

$tr "aeiou" "AEIOU" < sample ↵

$tr "a-z" "A-Z" < sample ↵ Translate lower to upper.

$tr "A-Z" "a-z" < sample ↵ Upper to lower

$tr -s " " < sample ↵

↳ Squeeze

$tr -d "aeiou" < sample ↵ To delete aeiou

$tr "," "\t" < sample ↵ Replaced with tab space.

⇒ **aspellcheck:** To check the spelling mistakes but not gramitical mistakes.

$aspellcheck sample ↵

$aspellcheck test ↵

⇒ **Head:** Displays top 10 lines of the file

$head Sample ↵

$head -5 Sample ↵ Top 5 lines.

⇒ **Tail:** Displays last 10 lines of the file.

$tail Sample ↵

$tail -5 sample ↵ last 5 lines

$tail -f sample ↵ file is open continuously.

⇒ **piping (|):** Combine the two or more commands in to a single line.

→ Here the first command output is taken as the next command input.

$ls -l | wc -l ↵

$cat Example | cut -d " " -f3 file_example ↵

$cat Example | head -20 ↵

⇒ **&&:** combines commands.

$echo "This is text file" > file_example && cut -f3 Example ↵

$cat file_example ↵ To verify

$echo "My original text" >> file_example && cat file_example ↵

⇒ **More:** To see the contents of a file in the form of pagewise.

$more Example ↵

⇒ **less:** To display file contents in pagewise. But we can go to all directions.

$less Example ↵

options: f → forward direction
b → Backward direction
v → Vi editor mode
q → To quit.

⇒ **Tee:** It is used to write the data into the files as well as on the screen.

$cat sample | tee file1 file2 file3 ↵

$cat file1 ↵ to verify
$cat file2 ↵

⇒ **Sort:** Sorts the output of a command or file.

Syn: Sort [options] FILE

-r → Sorts in reverse order
-b → Ignores leading blanks
-n → Compares according to numerical string value.

$ sort Example ↵

$ sort -r Example ↵ Reverse order

$ sort -n Example ↵ Display numaric

$ sort -u Example ↵ uniquee lines

$ sort -f Example ↵ Ignores case

ASCII Values
0-9 ⇒ 48-57
A-Z ⇒ 65-90
a-z ⇒ 97-122

⇒ **uniq:** Lists all the unique lines in a file or command output.

$ uniq Example ↵

$ uniq -u Example ↵ Displays non duplicated lines

$ uniq -d Example ↵ Displays only duplicated lines

$ uniq file_example > uniq_file && cat uniq_file ↵

→ On above command to view uniq lines in the sample file, create a new file based on the output, and view the contents of this new file.

⇒ **Sed (Stream Editor):** To search and replace strings or patterns in the given file.

→ Sed is a multipurpose filter command.

Syn: Sed "s/oldstring name/new string name/g" <filename>

s → Substitution

g → Global occurence in Every line.

$ sed "s/unix/linux/g" sample↵

$ sed "s/unix/linux/gi" sample↵ Ignore case.

$ sed "s/unix/linux/" sample↵

$ sed "s/^unix/linux/gi" sample↵

$ sed "s/unix//gi" sample↵ Delete a word from a file.

$ sed -e "s/unix/sas/gi" -e "s/linux/dba/gi" sample↵

$ sed -n "2p" sample↵ To print 2nd row

$ sed -n "3,5p" sample↵ To print 3rd, 4th, 5th rows.

$ sed -n "1p↵
>$p" sample↵ print 1st and last rows.

$ sed '3d' sample↵ Delete 3rd row.

$ sed '2,5d' sample↵ Delete 2 to 5 lines.

$ sed '2,5w file' sample↵ It copies 2nd to 5th rows from sample file to file.

$ sed '=' sample↵ To get line numbers.

⇒ Regular Expressions (or) Regex (Grep):

→ Globally research a regular Expression & print

→ To search a string or regular expression in a file(s).

Syn: grep [options] PATTERN FILE(S)

$ grep root sample ↵

$ grep root sample example backup ↵

$ grep root * ↵ Search all files in a current directory.

$ grep -i root sample ↵ Ignore case

$ grep -c root sample ↵ counts no. of lines

$ grep -n root sample ↵ print the lines along with line no's.

$ grep -l root sample ↵ List file names only the given patte

$ grep -r root * ↵ Search the pattern Recursively

$ grep -v root sample ↵ prints nonmatching lines.

$ grep -o root sample ↵ prints only the given pattern.

$ grep root sample --color ↵ Displays output in color

$ grep "it technology" sample ↵

$ grep "exam*" sample ↵ prints start-with exam pattern.

$ grep "b[aeiou]ll" sample ↵

o/p: ball
bell
bill
boll

$ grep "b..d" sample ↵

o/p: band
book
ba#d
ba-d

Note: "." & "*" are wild card characters, it matches any single character.

$grep c[on] Example↵

$grep [0-9] Example↵

word pattern: \< \> ⇒ word boundary

\< ⇒ Starting of the word

\> ⇒ Ending of the word

$ grep "\<root\>" sample↵

$grep "\<root" sample↵ o/p: root✓ root123x

$grep "root\>" sample↵ o/p: root✓ root123✓ xroot123x

$grep "\<[0-9][0-9][0-9][0-9]\>" sample↵ 1025✓ 112x 1356✓

Line pattern:

Anchors: ^ ⇒ Start of the line.

$ ⇒ End of the line.

$grep "^d" sample↵ line start with d

$grep "^me" sample↵ line start with me.

$ grep "me$" sample↵ line End with me.

$grep "^[^aei] sample↵ Not start with a, e, i

$grep "[0-9]$" sample↵ line Ending with digit.

$grep "^unix$" sample↵ line should contain only unix.

$grep "^$" sample↵ Displays Empty lines

$ grep "^...$" sample↵ Line should contain three characters.

fgrep: To search the string more faster than the grep command

```
$ fgrep "unix ↵
>sas ↵
>dba" sample ↵
```

egrep (Extended grep): It is a combination of grep & fgrep plus some additional regular Expressions.

```
$egrep "(unix|oracle|sas)" sample ↵
$egrep ab{3}c sample ↵   Exact occurence of preceding character
$egrep "|<[0-9]{4,7}|>" sample ↵
$egrep ab{3}c => abc  abbbc  abbbbbc
                  ×    ✓      ×
```

⇒ **Find:** This filter is used to search the results by depending on requirments may be on name, inode, permissions, user----etc.

Syn: find <Search path> <criteria> <action>

(a) **Based on name:**

```
$ find / -name passwd ↵
$ find /home -name passwd ↵
$ find /etc -name 'pass*' ↵
$ find . -name linux ↵
```

(b) Based on Size:

+n ⇒ for greater than n.
-n ⇒ for less than n.
n ⇒ for Exactly n.

$ find / -size 4c ↵ 4 character files

$ find / -size +4c ↵ More than 4 character files

$ find / -size -4c ↵ less than 4 characters.

$ find . -size +50m ↵ More than 50M.

$ find /etc/Backup -size -50m ↵ Less than 50M

$ find / -size +30M -size -50M ↵ Between 30 to 50m.

(c) Based on permissions:

$ find / -perm 644 ↵

$ find / -perm 665 ↵

$ find / -perm 777 ↵

(d) Based on Type: $ find / -type f ↵ To find files

$ find / -type d ↵ To find directories.

(e) Based on inode:

$ find / -inum 15253 ↵

$ find /root -inum 32512 ↵

$ find /home -inum 130128 ↵

(f) Based on time:

mtime → Modification time.
ctime → Change time.
atime → Access time.

```
$find / -mtime +10↵
$find / -mtime -10↵
$find / -mtime 10↵
```

⇒ To find the file with access time:

```
$find /root -atime +5↵   5 days ago.
$find /root -atime -5↵
$find /root -atime 5↵
```

⇒ To find the file with change time:

```
$find / -ctime +5↵
"      "    -5↵
"      "    5↵
```

```
$find / -amin +5↵   file was last accessed 5 minutes ago
$find /root -cmin +5↵   file's status was last changed
                        5 minutes ago.
$find /home -mmin +5↵
$find /root -amin -5↵
$find .     -amin 5↵
```

(g) Based on user:

$ find / -user <username>

$ find / -user raju ↵ particular user files.

(h) Based on group:

$ find / group <groupname>

$ find / group sales ↵ particular group files.

* path: It is the way of representing files & directories in the system.

→ There are two types of paths.

(i) Absolute path: It is the way of representing files and directories form the top of hierarchy.

Ex: $ ls /root/world/asia/india/ap/hyd/↵

$ cp /home/raju/linux /root/world/asia ↵

(ii) Relative path: It is the way of representing files and directories which are related to current directory.

Ex: $ cd /home/raju/Desktop ↵

$ cp linux unix/sas ↵

$ cd unix/sas ↵

$ ls ↵

* Monitoring System performance: (or) process Management:

→ A process is a program under Execution. (or)

→ A process is a instance of a running program.

→ process have their own address space in memory, thread of Execution, and characteristics such as security context, Environment and current priority.

→ The linux kernel tracks every aspect of a process by its process ID number information about Each process is advertised by the kernel to user program through the /process/PID directories.

→ When a process starts another program, the new process is called child process. This original process is the parent process of its child process. Child process inherit characteristics from its parent, such as its Environment and the user and groups its as which its run.

→ There are two types of process: (a) foreground process.
(b) Background process.

(a) Foreground process:

→ In foreground user can execute only one process (or) job.

Ex: $ firefox ↵

→ To kill the foreground job:

Ctrl + C

Ex: ② $ cp file1 file2 ↵

(b) Background process:

→ In Background user can execute many jobs at a time.

Ex: $ firefox & ↵

$ cp file1 file2 & ↵

→ To check the Jobs list: $ jobs ↵

ps: Displays information about running process.

$ ps ↵

→ To view process with more detailed information:

$ ps u ↵

→ To can detailed about a particular process:

$ ps aux | grep ssh ↵

(or)

$ ps aux ↵

kill: Terminates a process.

Syn: $ kill PID

$ kill 4286 ↵

Sometimes if the kill command doesn't work the way you intended it do, you can also call it with the -9 option to give it priority on the System.

$kill -9 4286 ↵
↳ signal.

Signal: The operating system communicates to process through signals. These signals report events or error situations to processes. In many cases signals will result in the process existing

→ one typical signal is SIGTERM, which terminates the process; it asks it to exit cleanly.

→ Another is SIGKILL, which kills the process; the process is required to exit immediately.

→ To find the PID(s) belonging to the SSH service:

$pidof sshd ↵

(or)

$pgrep sshd ↵

top: Monitors system resources (Similar to Task Manager in Windows)

$top ↵

options:

S → To change the time interval for updating top results (sec's)

R → To sort by PID number

U → username to get only that user process details.

P → To sort by CPU utilization.

M → To sort by RAM utilization.

C → To display or hide command full path.

r → To renice a process

K → To kill a process

W → To save the modified configuration

'q' → To quit.

→ When you're comfortable working with processes, you can then make some more advanced adjustments, such as changing the priority of a particular process.

renice: Adjusts the priority of a particular process.

Syn: renice <priority> [options]

-p changes process priority for a particular PID.

-u changes process priority for a particular user(s).

→ The priority value range from -20 (first priority) to 20 (dead last priority). only the root user may set processes to use a priority under 0.

#renice -2 3874 ↵

* **Note:** If all ready processes have the same priority, they will share the processor equally. priority only has an Effect when two process at different priority levels are computing for cpu time, in which case the lower priority process will get less time & apper to run more slowly.

nohup: The nohup jobs will create in server account so nohup jobs will execute even the user disconnects from his account.

Ex: $nohup cp file1 file2 & ↵

$nohup firefox & ↵

—x—

* Communication commands :

→ The main concept of communication facility exchanging of information or files from one user to another user.

write : It is used for to write message to another user account, but he should be logged into the user.

```
$ write username/terminalname ↵
  -----
  -----
  ctrl+d (save & quit)
```

→ To deny messages : $mesg n ↵

→ To allow messages : $mesg y ↵

wall : It is used for to send broadcast message to all users, who are connected to server

```
$ wall ↵
    Welcome to Linux.....
    ctrl+d (save & quit)
```

Mail : using mail command you can quickly and efficiently circulate memos and other written information to your co-workers you can even send and receive mails from people outside your organization.

```
$ mail user1@server1254.Example.com ↵   Single user
$ mail user1 user2 user3 ↵              Multible users at a time
$ mail user1 < Stud ↵
$ mail user2 < backup_file ↵            } It translates files to user.
```

→ Mails are stored in mailbox : (/var/spool/mail/username)

→ To open the mail box : $mail ↵

Note: By default all mails will store in primary mail box (/var/spool/mail). It will opened mails in primary mail box transfered to secondary mail box (i.e mbox).

→ To open secondary mail box : $mail -f ↵

→ The primary mailbox only maintains unread mails.

Mail box options: q → quit d → delete
r → reply d 2 → delete 2nd mail
P → print w filename → It writes to new file.

* One Bit Equals to how many bytes :

→ Bit is the smallest component of data and byte is larger than bit size. The size 1000 can be replaced with 1024 and still be correct using the other acceptable standards. Both of these standards are correct depending on what type of storage you are refferring.

Processor (or) Virtual Storage		Disk Storage	
1 bit	= Binary digit	1 bit	= Binary digit
8 bits	= 1 Byte	8 bits	= 1 Byte
1024 bytes	= 1 Kilobyte	1000 bytes	= 1 Kilo
1024 Kilo	= 1 Mega	1000 Kilo	= 1 Mega
1024 Mega	= 1 Giga	1000 Mega	= 1 Giga
1024 Giga	= 1 Tera	1000 Giga	= 1 Tera
1024 Tera	= 1 peta	1000 Tera	= 1 peta
1024 peta	= 1 Exa	1000 peta	= 1 Exa
1024 Exa	= 1 Zetta	1000 Exa	= 1 Zetta
1024 Zetta	= 1 Yotta	1000 Zetta	= 1 Yotta
1024 Yotta	= 1 Bronto	1000 Yotta	= 1 Bronto
		1000 Bronto	= 1 Geo

* **Links:** To give a pointer to the source file called as a link.

→ In unix/Linux two types of Links.

(a) Soft Links

(b) Hard Links.

(a) Soft links: → The inode number of the source file, Link file are different.

→ It can be created across the file system.

→ Editing of original file will be replicate in the Link files.

→ Size of soft link file Equals to number of characters in original file path.

→ If source file is deleted the link file will not be accessible.

→ It is also be called as Shortcut link.

Syn: $ ln -s <source file> <link file>

Ex: $ ln -s /Backup/linux /root/Desktop/linux ↵

(b) Hard links: → Source file, link file has same inode numbers.

→ It can't be created across file system.

→ Editing of original file will replicate in the link files.

→ Size of hard link file is same as original file.

→ The source file is deleted the link file we can access.

→ It is a backup link.

Syn: $ ln <source file> <link file>

Ex: $ ln /root/backup /root/Desktop/backup ↵

—×—

* **Shell concept:** Shell is a command line interpreter. The Shell access request from user and the checks command existence if the command exist then it converts into kernel understandble language (machine language) and it send the given request to kernel.

→ The shell access interface b/w user and kernel.

⇒ **Types of shells:**

Shell name	Developed By	prompt	Interpretername
Bourne shell	Stephen Bourne	$	Sh
Korn Shell	David korn	$	ksh
C Shell	Bill Joy	%	csh
Bash Shell	Stephen Bourne	$	bash
Z Shell	paul	$	zSh

Note: The advanced version of Bourneshell is bash shell.

→ Bash means "Bourne again Shell".

→ Bash shell is default in Linux.

Default Shell name	Flavour name
Bash Shell	Linux
Bourne Shell	SCO-unix, Solaries, HP-UX
Korn Shell	IBM-AIX
CShell	IRIX (Silicon Graphics)

Features of Shells:

- Word completion.
- Command History.
- Command alias.

→ To check the shells:

```
#cat /etc/shells ↵
```

→ To check parent shell of current user:

```
#echo $SHELL ↵
```

→ To view the available shells:

```
#cd /bin ↵
#ls *sh ↵
```

→ To Shift from bash shell to sh shell:

```
#sh ↵
```

→ To shift from sh shell to k shell:

```
#ksh ↵
```

→ To check current working shell:

```
#echo $0 ↵
```

→ To Exist the shell:

```
#exist ↵
```

Command completion: Linux automatic command completion is a tool or program that can identify what you are typing in the Linux command line terminal and can complete that command, words or sentence for you. This is really cool feature in Linux.

→ When <TAB> key is pressed, any command starting with the given string will be completed by the system automatically.

→ For multible commands that starting with the given string, pressing <TAB> key twice will list down all those matched files or commands.

→ If there are no matched of any command, files or folders. then, the automatic word completion will not shown, a 'ting' sound will buzzed.

Ex: $ cd /var/l<tab>
lib/ lock/ log/

Command History: The history command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is reffered to as an 'event' when specifying an event to the history command.

$history ↵
$history 10 ↵
$history -c ↵ Lock the history.
$history -r ↵ Unlock the history.
$ rm .bash_history ↵ Removes the history.

Command alias:

→ Alias is a built in shell command in Linux/unix operating systems.

→ It can save you a lot of typing by assiging a name to long commands.

→ The alias command can be useful if you want to create a 'shortcut' to a command.

Syn: aliasname='command'

Ex: $ alias u=useradd ↵
$ alias c=clear ↵
$alias ↵ Display alias list
$unalias u ↵ Disable alias
$vim .bashrc ↵ permanent alias names

—x—

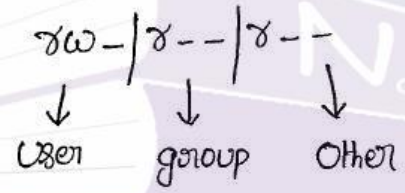
File permissions

→ Just like every operating System, Linux comes with a set of permissions that it uses to protect files, directories, and devices on the System.

→ These permissions can be manipulated to allow (or) disallow access to files and directories on different parts of the System.

Basic file permissions:

→ let's look at how permissions work first. Linux permissions are implemented through the properties of files and defined by three separate categories.



User: person who owns the file.

Group: Group that owns the file.

Other: All other users on the System.

→ permissions in Linux can be assigned one of two ways. you can use the mnemonic or a single digit to represent the permission level.

Operation	Digit	Mnemonic	Description
Read	r	4	View file contents.
Write	w	2	Write to or change.
Execute	x	1	Run the file.

Default file permissions:

umask: Universal mask is a default value that always gets dedicated from maximum file permission allocated for every file & Directory.

→ For Super user umask value is #022.

→ For Normal user umask value is $002.

For Super user:

→ Maximum permission of a file 666

umask 022

(-)

→ Default file permission → 644

→ Maximum permission of a directory 777

umask 022

(-)

→ Default directory permission 755

For Normal user:

→ Maximum permission of a file – 666

umask – 002

(-)

664

→ Maximum permission of a directory – 777

umask – 002

(-)

775

→ To see the umask #umask↵

→ To change the umask #umask 222↵

→ To view umask value from the file

#vim /etc/bashrc↵

Operators:

+ ⇒ To add a permission.

− ⇒ To remove a permission.

= ⇒ To override the permission.

→ Here are some of the commands you can use to work with permissions:

(a) chmod:- It is used to change the permission of a file and directory. It can be used by the owner of the file (or) by root.

Syn: chmod [options] [permission] [File]

-R → Acts recursively.

-v → provides verbose output.

Ex: ① # chmod u+rw,g+r,o+x linux ↵

(or)

chmod 641 Linux ↵

② # chmod ugo =rw backup ↵

(or)

chmod 666 backup ↵

③ # chmod u-w, g-r, o-x linux ↵

④ # chmod -R u+w, g+r, o+x Linux ↵

⑤ # chmod -R 777 unix ↵

⑥ # chmod 755 unix ↵

(b) chgrp :

→ By using this command we can change group of the file.

Syn: chgrp [options] [groupname] [file]

-R → Recursively

-v → Verbose

Ex: # ls -l linux ↵

chgrp sales linux ↵

(c) chown :- This command is used to we can change the owner of the file, as well as owner & group at a time.

Syn: chown [options] [user:group] [file]

-R → Recursively

-v → Verbose.

Ex: # chown raju linux ↵ To change only owner.

chown raju:sales linux ↵ To change owner & group.

chown -R ramu:color unix ↵ Recursivly to change.

→ To view the symbolic as well as numaric mode of permission

stat linux ↵

→ To change the permissions in GUI mode

nautilus & ↵

→ Assign the permissions in GUI mode:

Right click on file → properties → permissions.

—x—

Automation Jobs

→ On any operating system, it is possible to create jobs that you want to reoccur. This process known as job scheduling, is usually done based on user-defined jobs.

→ As an administrator, however, you can define your own jobs and allow your users to create them as well.

→ The Importance of the job scheduling is that the critical tasks like taking backups, which the clients usually wants to be taken in nights, can easily be performed without the intervention of the administrator by scheduling a cron job. If the cronjob is scheduled carefully than the backup will be taken at any given time of the client and there will be no need for the administrator to remain back at nights to take the backup.

→ For Red Hat or any other linux, this process is handled by the cron service or a daemon called crond, which can be used to schedule tasks (also called jobs).

→ By default, Red Hat comes with a set of predefined jobs that occur on the system (hourly, daily, weekly, monthly, and the with arbitrary periodicity).

→ There are two tools to Scheduling jobs.

(a) at

(b) crontab

⇒ **AT Jobs:** "at" is used to schedule the job for a particular time or interval, in other words it is used only for one time or only for one interval.

Syn: at [options]

-l Lists all jobs in the queue

-d Removes a job from the queue.

-f Reads input from the file

-m Sends mail to the user when the job is complete.

Examples:

① $at 9am↵

at> date

ctrl+d (Save & quit)

② $at now + 3 days ↵

at> /bin/echo "Hello world"

ctrl+d (Save & quit)

③ $at 03222013 ↵

at> ls

ctrl+d (Save & quit)

④ $at 1:30 3/22/2013 ↵

at> cp file1 file2

Ctrl+d (Save & quit)

→ View the currently queued jobs:

$at -l ↵

(or)

$atq ↵

⑤ $at -f filename 11pm ↵

at> /bin/echo "Hello world"

Ctrl+d (save & quit)

→ Delete the job from the queue:

$at -d 1 ↵

(or)

$atrm 1 ↵

→ Verify that the job is truly gone:

$atq ↵

→ To view the job details:

$at -c 2 ↵

⇒ Restricting a user from using at jobs:

→ The atd service uses two files to control access to the services.

/etc/at.allow

/etc/at.deny

The /etc/at.allow file:

- If it exists, only these users are allowed (at.deny is ignored).
- If it doesn't exist, all users except at.deny are permitted.

The /etc/at.deny file:

- If it exists and is empty, all users are allowed (Red Hat default)

for both files:

- If neither file exists, root only.

⇒ cron jobs:

→ The default setting for Red Hat allows any user to create a cron job. As the root user, you also have the ability to edit and remove any cron job you want.

→ Let's jump into creating a cron job for the system. you can use the crontab command to create, edit, and delete jobs.

Syn: crontab [-u user] [option]

options: -e Edits the user's crontab.

-l Lists the user's crontab.

-r Deletes the user's crontab

-i prompts before deleting the user's crontab.

→ Before you start using the crontab command, however, you should look over the format it uses so you understand how to create and edit cron jobs. Each user has her own crontab file in /var/spool/cron, based on the username of each user. Any "allow" actions taken by the cron service are logged to /var/log/cron.

→ To view the /etc/crontab file to understand its Syntax:

$grep ^# /etc/crontab↵

Examples:

crontab -e↵

*/01	*	*	*	*	date
*/30	11	*	*	*	cp file1 file2
45	10,22	03	*	*	/bin/echo "Hello world"
50	22	01, 11	10	*	logname
59	23	31	12	*	/bin/echo "Happy New year"
*/01	*	*	*	0,6	/bin/echo "Today is weekend"

:wq!↵

→ To check assigned cronjobs:

crontab -l↵

→ To check the crond job service:

```
# service crond status ↵
```

→ Restart the cron service:

```
# service crond restart ↵
# chkconfig crond on ↵
```

→ To setup user01's crontab:

```
# crontab -u user01 -e ↵
* * * * * /tmp/sample_script
:wq!
```

→ To remove a user's crontab jobs:

```
# crontab -u user01 -r ↵
```

→ You can verify the log file:

```
# tail /var/log/cron ↵
```

*

Note:

→ What do you think happens if you set up cron jobs to run during the night (say, to run some reports) and you shut down the system right before you go home? well, it turns out that there is another great feature of cron. the /etc/~~cron~~anacrontab file defines jobs that should be run every time the system is started. If your system is turned off during the time that a cron job should have run, when the system boots again, the cron service will call /etc/anacrontab to make sure that all missed cron jobs are run.

→ Let's look at the /etc/anacrontab file:

#cat /etc/anacrontab ↵

Control access to the cron service:

→ To start working with cron, first need to know at the two config files that control access to the cron service:

→ These two files are

/etc/cron.allow.

/etc/cron.deny.

The /etc/cron.allow file:

- If it exists, only these users are allowed (cron.deny is ignored)
- If it doesn't exist, all users except cron.deny are permitted.

The /etc/cron.deny file:

- If it exists and is empty, all users are allowed (Red Hat default)

For both files:

- If neither file exists, root only.

LINUX PROGRAMMING

Topic:

User Administration

Contributed By:

Nagi Reddy

User Administration

→ User is nothing but an individual who uses the available H/w & S/w resources

→ On Red Hat, there are three different types of users.

(a) Super user (or) root user:

→ The root user account is the equivalent of the Administrator (or) Enterprise Admin account in the Windows world.

→ It is the most powerful account on the System and has access to everything.

(b) System users:

→ A System account is similar to a normal user account. The main difference is that System users normally don't have a home directory and can't login the way normal users do.

→ Many system users are created or associated with applications or service to help run them more securely.

→ For Example if we install 'Apache it will create a user apache. these kinds of users are known as System users.

(c) Normal user's:

→ Normal users are the users created by the root user. they are normal users like Rahul, Sara, Rafu----etc.

→ Normal user accounts have no write access to anything on the System except their home directory (they can read and explore much of the System, however), which is created when the user account is added.

→ As an Administrator, you can assign access rights to different files & directories, allowing your users to gain access to different areas of the System (outside their home directory).

Some important points related to users:

→ Users & Groups are used to control access to files and resources.

→ Users login to the system by supplying their username & password.

→ Every file on the system is owned by a user and associated with a group.

→ Every process has an owner and group affiliation, and can only access the resources owner or group can access.

→ Every user of the system is assigned a unique user ID number.

→ Users name and UID are stored in /etc/passwd.

→ Users password is stored in /etc/shadow in encrypted form.

→ Users are assigned a home directory and a program that is run when they login (usually a shell).

→ Users can't read, write or execute each other's files without permissions.

Types of users in Linux & their attributes:-

Type	Example	UID	GID	Home Directory
Super user	root	0	0	/root
System user	ftp, ssh, nobody	1-499	1-499	/var/ftp
Normal user	visitor, sara	500-60,000	500-60,000	/home/user

→ Whenever a user is created in Linux things created by default:

- → A home directory is created (/home/username)
- → A mail box is created (/var/spool/mail)
- → Unique UID & GID are given to user.

User private Group (UPG) :-

On Red Hat Linux uses user private Group (UPG) schema. According to UPG Scheme, you create the any user account, the user consists the primary group with same name and same ID.

→ For Example if a user is created with the name Raju, then a primary group of that user will be Raju only.

→ The user information maintained by the two database files:

(a) /etc/passwd :→ This file maintains user related information.

Syn: <username>:<password>:<UID>:<GID>:<Comments>:<Homedir>:<shell>

(b) /etc/shadow: → This file maintains user related password information.

Syn: <username>:<encrypted password>:<Last passwd change>:<min>:<max>:<warn>:<inactive>:<expires>:<not used>

Complexity Requirements of password :-

→ A root user can change password of self and any user in the system, there are no rules for root to assign a password. Root can assign any length of password either long or short, it can be alphabet or numeric or both. on the whole there is no limitation for root for assigning a password.

→ A normal user can change only its password. Valid password for a normal user should add here to the following Rules.

- → It should be at least 7 characters but not more than 255 character
- → At least one character should be upper case.
- → At least one character should be lower case.
- → At least one character should be a symbol and number.
- → It should not match the previous password
- → The login name and the password can't be same.

→ To manage user accounts, you can use the following commands:

Useradd:- Create user (or) System accounts.

Syn: useradd [options] LOGIN

options:

-u	userid
-c	Comment
-e	Expire_Date
-S	SHELL
-r	Creates a System account
-d	Home directory
-g	primary group id
-G	Secondary group id.

Ex: → Create a user

```
#useradd raju↵
```

→ To check the user you just created

```
# cat /etc/passwd | grep raju↵
```

→ Let's create a user with our own attributes

```
#useradd -u 555 -c "Linux user" -d /opt/india -s /bin/sh india↵
```

→ To check the user:

```
# cat /etc/passwd | grep india↵
```

Tip:- As a good practice, you should provide a label or some description for each account; otherwise, after time, you will forget what it is for.

Usermod:- Modifies user accounts.

Syn: usermod [options] LOGIN

Note:- All the options which are used with useradd command can be used and,

-l TO change login name

-L TO Lock account

-U TO Unlock account.

Ex: → Changing the name of the user

#usermod -l newname oldname ↵

→ TO lock the user account

#usermod -L username ↵

→ TO unlock the user account

#usermod -U username ↵

Note: When an account is locked it will show ! (Exclamation mark) in /etc/shadow file.

Userdel: Removes a user or System account.

Syn: userdel [options] LOGIN

-f forces deletion of the user even if he's still loged in.

-r Removes the user's home directory and mail spool.

Ex: #userdel username ↵

passwd: Sets a password or resets a password for a user account.

Syn: passwd [options] [LOGIN]

-l → Locks a user's account

-u → Unlocks a user's account

Ex: #passwd raju ↵

→ Let's look at how the password files work.

#cat /etc/shadow | grep raju ↵

Chage: Enables you to modify the parameters surrounding passwords (complexity, age, expiration)

Syn: Chage [options] user

-d Indicates the day the password was last changed.

-E Sets the account expiration date

-I Change the password in an inactive state after the account expires.

-l Shows account aging information.

-m Sets the minimum number of days between password changes.

-M Sets the maximum number of days a password is valid.

-W Sets the number of days to warn befor the password expires.

Ex: → find the user's password information.

#Chage -l user ↵

→ Sets user account to expire in one week

#Chage -E 2013-03-28 raju ↵

pwck: Verifies the consistency of passwords across database files.

→ When you create or delete users, sometimes things don't always work out properly. This can cause the password file to become inconsistent you can use the pwck command to verify the consistency between the /etc/passwd & /etc/shadow file.

#pwck ↵

Group Administration

→ Group is nothing but collection of users using which one can reduce the administration task in the o/s environment.

→ Groups are divided into two types.

(a) **primary group:-** It is a group in which a user initially belongs in this group the user can access the resource with default permissions.

(b) **Secondary group:-** A part from primary, if a user have an account in the other group i.e then it is called as secondary group to the user.

group → The group information maintained by the two database files.

(i) **/etc/group:** This file maintains group related information.

Syn: <group name> : < password placeholder> : <GID> : < members>

(ii) **/etc/gshadow:** This file maintains group password related information.

Syn: < group name> : < password placeholder> : < group admin> : < members>

groupadd: Creates a group.

Syn: groupadd [options] groupname

-r Creates a system group

-g groupid

Ex: → Let's create a group called sales.

#groupadd sales ↵

→ To verify the group:

cat /etc/group | grep sales ↵

→ To add the user:

#usermod -G Sales user1 ↵

#cat /etc/group | grep sales ↵

→ To add another user to the sales group

#usermod -G Sales user2 ↵

→ Now if you verify, you should see two user accounts in the last field:

#cat /etc/group | grep Sales ↵

→ Another way you can verify what groups a user belongs to is to use

Syn: id [options] [username]

-G Shows the GID

-n Shows the name instead of the ID

-u Shows the UID.

Ex: #id -Gn user1 ↵

→ If the ID command is called without any options, you can also see what UID & GID the user has: #id user1 ↵

groupmod: Modifies the properties of a group.

Syn: groupmod [options] groupname

-g groupid

-n newgroupname

→ To change the groupid:

groupmod -g 888 sales ↵

→ To change groupname:

#groupmod -n <new-name> <Existing name>

gpasswd:

→ Assign the password to the group:

#gpasswd <groupname>

→ Adding and Removing Members to a Group:

Syn: gpasswd [options] <arguments> <groupname>

-a Add single user

-M Add Multible users

-A Group Administrator

-d Removing a user from group

Ex: #gpasswd -a user1 Sales ↵

#gpasswd -M user2, user3, user4 color ↵

#gpasswd -d user2 color ↵

#gpasswd -A user3 color ↵

groupdel : Deletes a group.

→ To removes a group :

#groupdel <groupname>

Note: If the group has Empty (or) Secondary users you can delete the group. In case the group maintains single primary user, then you can't delete the group account.

Switching Accounts:

su : Enables you to run a command as another user or switch user accounts.

→ To switch accounts, use this command :

#su user1 ↵

→ you can also login as the root user using the su command :

#su - root ↵

* Tip :- you can log in to the root user account using the su command with no parameters. So what is the difference between using su and su - The su command moves you into the root user's account without initializing any of root's path or shell variables. When you use su -, everything is initialized as if you were logging in from the console.

User Account Initialization : When a user is created, everything from the /etc/skel directory is copied to the user's newly created home directory (usually /home/username).

→ you can modify these "skeleton" files or can add your own custom files. The benefit here is that user creation becomes standardized, ensuring that policies are adhered to. The customizable files are broken down into two different sections:

(a) User-Specific files:

→ After a user is created and his home directory is populated, that user can now customize those files to fit his own personal needs.

- .bashrc — Defines functions and aliases
- .bash_profile — Sets environment variables
- .bash_logout — Defines any commands that should be executed before the users log out.

(b) Global User Configuration:

- /etc/bashrc — Defines functions and aliases
- /etc/profile — Sets environment variables
- /etc/profile.d — Specifies a directory that contains scripts that are called by the /etc/profile file.

→ One last file to look at is /etc/login.defs. This file controls specifies relating to system wide user logins and passwords.

```
# grep -v ^# /etc/login.defs ↵
```

(or)

```
# cat /etc/login.defs ↵
```

Group Collaboration:

Group collaboration is an essential part of any business and for any system administrator who deals with users. Here we look at three key features about file and directory permissions;

(a) Setuid : This flag is used to allow multiuser access.

→ For example, if you have a script that generates reports for your company, but the script must be run as user1 to succeed, you can set the setuid bit to enable other users to run this command as though they were user1.

→ Create a file to hold the report script:

```
# touch reporting_script
```

→ Set the setuid bit:

```
# chmod 4755 reporting_script
```

(or)

```
#chmod u+s reporting_script
```

→ Now view the permissions of the file:

```
# ls -l reporting_script
```

→ On the file's owner permissions, notice that there is an 's' in in place of the x. This shows that this file has the setuid flag set.

*Tip: → To find all setuid files:

```
# find / -perm 4000 ↵
```

(b) Setgid: This flag is used to allow multigroup access.

→ which is similar to setuid but set at the group level instead. With this bit set, all users of the group are able to execute the file instead of just the user who owns it. The setgid bit allows users to collaborate on files.

Step ①: As root, create the directory:

#mkdir /tmp/oracle ↵

Step ②: create the group and add users to it.

#groupadd Sales ↵

#usermod -G Sales user1 ↵

#usermod -G Sales user2 ↵

Step ③: Assign the permissions for collaboration:

#chown root: Sales /tmp/oracle

#chmod 2770 /tmp/oracle

Step ④: verify #ls -ld /tmp/oracle ↵

→ Now all members of the Sales group are able to read/write to files within this folder. Also, notice that access to this folder is denied for anyone who isn't a members of the Sales group.

(c) Sticky bit: This flag prevents accidental delete by users & groups

Step ①: Set the Sticky bit on the /tmp directory:

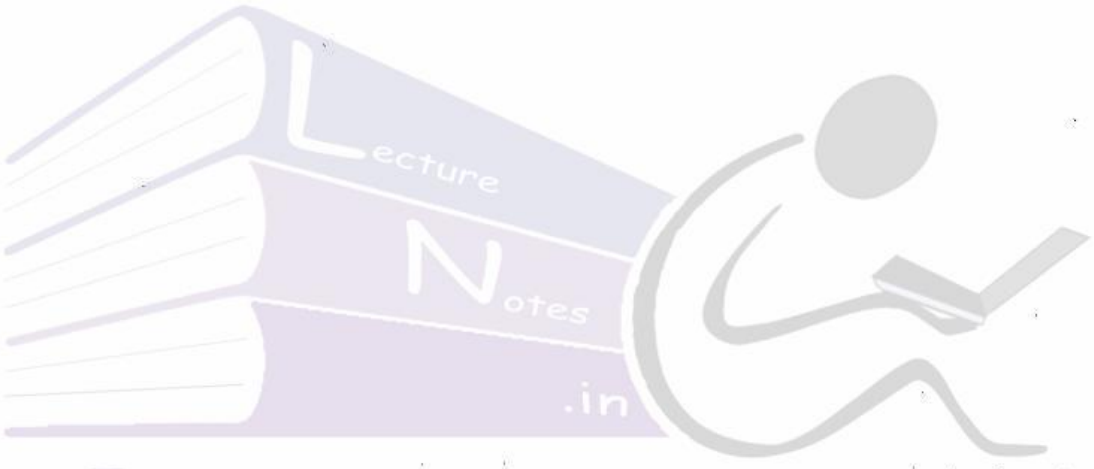
#chmod 1777 /tmp ↵

Step ②: verify # ls -ld /tmp ↵

→ For the Sticky bit, there is a 't' on the end of the permissions listed. Now Otherusers are not able to delete your files; only you can.

→ This feature might be helpful when you're sharing files and there are particular files you don't want other users to delete.

—x—



Networking

→ One of the key elements of connecting to different systems in the network configuration involved.

→ A network is a set of hardware devices connected together, either physically or logically to allow them to exchange information.

→ Network management is fairly easy when it comes to Red Hat.

→ Most of the network configuration is kept in files; therefore, adjusting these settings is simple.

→ In the network system maintains the fully qualified Domain name

∴ F.Q.D.N = Hostname + Domain name

→ Let's start by looking at the information about hostname & Networking

→ To check the hostname: # hostname↵

→ To Managing hostname temporarily:

hostname Server1254↵

→ To Managing permanently: #vim /etc/sysconfig/network↵

NETWORKING=yes
HOSTNAME = Server1254
NETWORKING_IPV6 = yes

→ for avoiding geographical problems:

#vi /etc/hosts↵

192.168.0.254 Server1254.Example.com Server1254

:wq!↵

→ TO Managing IP-Address:

→ TO check the ipaddress:

Syn: ifconfig [options] [interface]

options: netmask
up
down

Ex: #ifconfig eth0↵

→ TO display all interfaces on the System

#ifconfig↵

→ TO change ip address temporarily:

Syn: #ifconfig eth0 <ip-address> netmask <Subnetmask> <up/down>

#ifconfig eth0 192.168.0.254 netmask 255.255.255.0 up↵

#ifconfig eth0↵ TO verify:

→ TO change permanently:

(a) #setup↵ (b) #System-config-network-tui↵

→ you could also check the output of the interface config file

#vim /etc/sysconfig/network-scripts/ifcfg-eth0↵

DEVICE=eth0
BOOTPROTO=NONE
ONBOOT=yes
IPADDR=192.168.0.254
NETMASK=255.255.255.0

:wq↵

→ To check the interface detected (or) not:

Syn: #ethtool <interface>

#ethtool eth0↵

→ To bring the single interface down:

#ifdown eth0↵

→ To restore the interface that you just brought down:

#ifup eth0↵

* Note:

→ Any time you make a change to an interface's settings, you need to bring down that interface and then bring it back up again.

WARNING: Restarting the network service interrupts all network connections and any client that is currently connected.

→ Restart the network service as follows:

Syn: # service <service name> <stop/start/restart/status>

service network restart↵

→ To check the all services status:

#service --status-all↵

→ To Manage peranently:

Syn: #chkconfig --level <runlevels> <servicename> <on/off>

chkconfig network on↵

→ To check the status of the service:

#chkconfig --List network↵

→ To disable the service at boot time: #chkconfig network off↵

Routing: When you have a system that has two or more network interfaces, they are called dual-homed (or) multihomed systems. You need to make sure that each interface has a gateway that it can route through.

Syn: route [options]

add	Add a net route
del	Deletes an existing route
flush	Flushes any temporary routes

→ Let's look at the current routes on the system:

#route ↵

→ To set default gateway:

#route add default gw 192.168.0.1 eth0 ↵

→ To verify: #route ↵

Networking utilities:

ping: Tests the connectivity between two hosts

#ping 192.168.0.254 ↵

→ When you ping something on a linux host, unlike in windows, the ping continues until you cancel it. You can limit the number of ping requests sent by prefixing -c number_count in front of the destination host.

#ping -c 3 192.168.0.254 ↵

netstat: Shows information about connections (open, closed and listening).

→ Using netstat command to obtain information on routing tables, listening sockets, and established connections.

Syn: netstat [options]

Options:

-r	Displays the routing table.
-I	Displays interface statistics
-t	Shows tcp connections
-u	Shows UDP connections
-a	Displays all sockets (tcp, udp, or local)
-p	Displays process ID's
-e	Displays Extended information.

→ To check that the connection is available for your clients:

```
#netstat -tuape | grep ssh ↵
```

Client DNS Troubleshooting:

/etc/sysconfig/network	: Contains the hostname of the System.
/etc/hosts	: Contains the local IP to hostname mappings
/etc/resolv.conf	: Contains the IP address of the DNS servers
nslookup	: Queries (or) looks up a domain name or system
ping	: Test connectivity between two hosts.

Ethernet Bonding :

Ethernet bonding is used to combine multiple interfaces into one, creating an increase in available bandwidth and redundancy. This is done by creating a special network interface file called as a channel bonding interface.

Package Management

→ Software is the basic of any operating System, allowing you to install and use different utilities.

→ In Linux, Software is distributed through the use of packages, which contain the actual software files.

→ Each distribution of Linux has its own package management System.

→ for RedHat, there are two package management Systems : RPM & YUM.

Working with RPM:

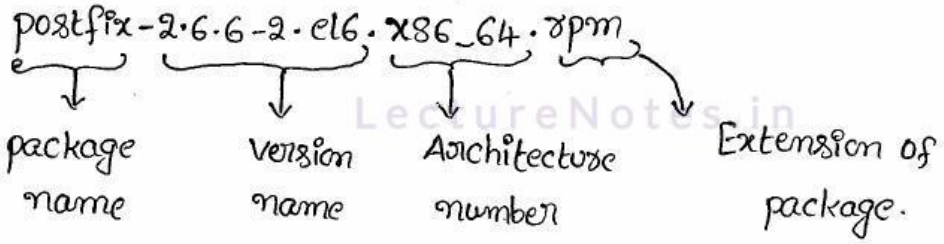
→ RPM stands for (Red Hat package Manager)

→ RPM is default package installation tool in Linux operating System.

→ By using Rpm we can install, upgrade, query, verify and Remove the packages.

→ Before diving into package management, let's look at the naming convention used by the System to describe packages.

package parameters:



Methods of Installation:

→ Two ways of Installations.

- Standalone
- Network.

Standalone: In this type, we can install the packages through any removable media (or) through dump.

Installing and Removing packages:

Syn: #rpm <options> <pkg-name> --force --aid

options: i → Installs a given package

v → verbose output.

h → Shows hash progress when installing

U → Upgrades a given package

e → Removes a given package.

--force → Install the package with forcefully.

--aid → Install the package along with Dependencies

--nodeps → To Erase an package without dependencies.

Quary Options (with -q):

Syn: #rpm <options> <pkg-name>

c → Lists all config files.

d → Lists all documentation files.

i → Displays information about the package.

l → Lists the files in a package.

s → status of the package.

verify Options (with -v):

-a → Queries all packages

f → Displays information about the specified file.

→ for installing an application.

```
#rpm -ivh nano-2.2.6-1.x86_64.rpm ↵
```

→ for installing an application with forcefully.

```
#rpm -ivh nano-2.2.6-1.x86_64.rpm --force ↵
```

→ If you want to upgrade this package (because you know that it is already installed), you can substitute the -i option for -U.

```
#rpm -Uvh nano-2.2.6-1.x86_64.rpm ↵
```

→ for uninstalling a package

```
#rpm -e nano ↵
```

→ you can always reinstall the package at a later date if you keep the .rpm file on your system

```
#rpm -ivh -replacepkgs nano-2.2.6_1-x86_64.rpm ↵
```

Tip: A common situation to run into on the job is having to install a package that is already installed. you don't have to go through the trouble of uninstalling the package first only to reinstall it. you can use the -replacepkg option alongside the regular install options to override an existing installation.

→ Query the instal'ed System package for nano:

```
#rpm -qa | grep nano ↵
```

→ Query the information from the nano package:

```
#rpm -qi nano ↵
```

→ Suppose you are looking around on your new Red Hat Installation and a file but aren't sure what it does. you can use the -f option to query the package that the file belongs to, possibly giving you a better idea of what that file might be used for.

→ find out where the /etc/Syslog.conf file came from by doing the following

#rpm -qf /etc/Syslog.conf ↵

→ Use the -c option to find all config files

#rpm -qc rsyslog ↵

→ To find the documentation files for a given package.

#rpm -qd rsyslog ↵

→ To listing of all files that come with the package.

#rpm -ql rsyslog ↵

→ To find out whether a package has any dependencies:

#rpm -qR rsyslog ↵

Network Installation: In this level we can install the package from Server through NFS (or) FTP services.

NFS Service: → first check the communication.

#ping 192.168.0.254 ↵

→ for accessing the data shared from Server

#mount 192.168.0.254:/var/ftp/pub/Server /mnt ↵

#cd /mnt ↵

#rpm -ivh vsftpd* --force --aid ↵

FTP Service: In this method to install the package the ftp server should have the dump of o/s under the ftp default shariable location

→ first check the communication

#ping 192.168.0.254 ↵

→ for accessing the data shared from the server.

#rpm -ivh ftp://192.168.0.254/pub/Server/<pkg-name> --force --aid

Working With YUM:

→ YUM stands for yellowdog update Modifier.

→ In this section, we look at the exact same tasks, except this time we use the more flexible YUM utility.

→ The YUM command has access to repositories where tons of packages are kept and can install, upgrade, or remove them for you automati-cally.

→ yum also takes care of resolving and installing any dependencies for you, which the rpm command can't do.

→ yum is an interactive tool which waits for the confirmation of a user.

→ yum is a default package management tool in Red Hat o/s.

→ Using this tool we can install the required packages with dependencies.

Syn: yum <options> <Commands> <package name>

Options:

c → Specifies the location of the config file.

q → Specifies quit, no output.

y → To always answer yes to prompts.

v → provides verbose output.

Commands:

clean → Removes cached data.

erase → Removes a package from the System.

List → Displays available packages

Install → Install a package on the System.

Search → Enables you to search for a package.

Update → Updates a package.

grouplist → Displays available packages groups.

groupInstall → Install a packages with in a group.

groupremove → Removes a packages with in a group

YUM Server Configuration:

→ Mount the dvd

#mount /dev/dvd /mnt↵

→ Install RPM package(vsftpd)

#rpm -ivh vsftpd* --force --aid↵

→ Copy the Dump of o/s into the default shareable location ftp.

#cp -rvf /mnt/* /var/ftp/pub ↵

* → means all the data under the mount point /mnt.

→ Install the createrepo package:

#rpm -ivh createrepo* --force --aid ↵

→ Create the new Repository:

#createrepo -g /~~mnt~~/var/ftp/pub/Server/repodata/comps-rhel5-server-core.xml /var/ftp/pub ↵

Note: If any errors are showing then remove

#rm -rf /var/ftp/pub/repodata ↵

→ Open the yum configuration file:

#vim /etc/yum.repos.d/linux.repo ↵

1. [linux]
2. name=yum Server
3. baseurl=ftp://192.168.0.254/pub
4. Enabled=1
5. gpgcheck=0

:wq! ↵

→ Restart the ftp service

service vsftpd restart ↵

yum clean all ↵

→ To list out packages

```
#yum list ↵
```

→ To install the package

```
#yum install postfix -y ↵
```

→ you could also update the postfix package

```
#yum update postfix -y↵
```

→ To remove the package

```
#yum remove postfix ↵
```

Note: one great feature about yum is that instead of updating a single package, you can list all updates that need to be installed for the system.

```
#yum list updates ↵
```

→ Fonom this list, you can choose to update packages individually or as a whole. If you want to install all the updates

```
#yum update ↵
```

→ To get a listing of all available "groups":

```
#yum grouplist↵
```

→ you can then install that "groups":

```
#yum groupinstall "Development Tools" ↵
```

Note: Don't forget that Linux doesn't handle whitespace the way that Windows does. If you specify a group name, you need to enclose it in quotation marks (" ").

Searching for packages:

→ find the postfix package to install:

```
#yum search postfix↵
```

→ To find out more information about the postfix package:

```
#yum info postfix↵
```

→ To flush the cache, do the following

```
#yum clean all↵
```

Configuring Additional Repositories:

→ Some times you might want to install a package that isn't in the repositories that come preconfigured with Red Hat. If the package is available in someone else's repository, you can add that person's repository to your yum config file.

→ you can either add your own custom repositories to the main config file /etc/yum.conf (or) create a .repo file in the /etc/yum.repos.d directory which will be added automatically. Here is what a sample entry for a custom repository looks like:

```
[unique title]
name=My custom yum Repository.
baseurl=ftp://192.168.0.250/opt/yum/myrepos
enabled=1
gpgcheck=0
:wq!↵
```

→ One neat trick that you can do is to create a repository based on an ISO.

→ This trick can be useful in an environment whether there is no Internet access and you'd like to install packages from the Red Hat DVD.

Step ①: Create a folder for the temporary mount:

```
#mkdir /mnt/cd ↵
```

Step ②: Mount the ISO image (or) cd:

```
#mount -o loop /dev/cdrom /mnt/cd ↵
```

Note: → To create a ISO image.

```
#dd if=/dev/scd0 of=/OS/Rhel5.iso ↵
```

↳ To check it which device is mount

→ Read the image file

```
#mount -o loop /OS/Rhel5.iso /DVD ↵
```

Step ③: Create a repository in the temporary directory:

```
#cd /mnt ↵
#createrepo . ↵
```

This creates an XML file of all the packages from the mounted CD.

Step ④: You also need to clean the current repository cache:

```
#yum clean all ↵
```

Step ⑤: Create a .repo file for your temporary repository;

#vim /etc/yum.repos.d/iso.repo↵

[ISO Repo]

name = My Repository.

Baseurl = file:///mnt/cd

enabled = 1

gpgcheck = 0

:wq! ↵

Adding your custom packages:

→ Just as you have already created two different types of custom repositories, you can add packages you have created to them as well.

Step ①: copy the file over to your private directory.

#cp /usr/src/redhat/RPMS/X86_64/mysample-1.0-0.X86_64.rpm

/opt/yum/myrepos↵

Step ②: Update the repository to recognize your new package:

#createrepo -update ↵

→ Now you should be able to search for your package in your private repository along with other software.

Registering your System:

To register your system to the Red Hat Network, you must have an active subscription with Red Hat. you can register during the installation of your system or manually after the system has already been installed. Use the 'rhn_register' command to begin the registration process. After you finish, you can visit http://rhn.redhat.com to start managing your system(s) through the web. you need to make sure that the rhnsd daemon is running in order for it to be managed.

Step ①: Set the daemon to boot on system start

```
# chkconfig rhnsd on ↵
```

Step ②: you should verify whether the service is currently running.

```
# service rhnsd status ↵
```

Step ③: If it is, you are all set; otherwise, start the service manually:

```
# service rhnsd start ↵
```

—— x ——

LINUX PROGRAMMING

Topic:

The Kernel

Contributed By:

Nagi Reddy

The Kernel

→ The heart of Red Hat is the Linux kernel. The kernel is responsible for interacting with the hardware and producing output to the screen. There is also a virtual file system that gets created in the /proc directory to hold information and parameters for the kernel.

→ Linux is truly just the kernel. Red Hat and the other distributions in existence today are software and configuration files packaged with the Linux kernel to bring you an entire operating system. Because the kernel is really what runs everything, understanding how it works is essential.

→ The kernel can be used to load new drivers, support new hardware, or even offer a custom kernel for individual needs.

→ The Linux kernel is modular, and because of this, you can load and unload kernel modules even after the system has booted.

→ Let's start with the uname command to find out some info, about the kernel.

uname: Displays information about the kernel.

Syn: uname [option]

- -a prints all information relating to the kernel
- -s Show the kernel name
- -r Requests kernel release information
- -v Requests the kernel version.

→ Let's check & see which version of the kernel is currently running:

```
#uname -a ↵
Linux root 2.6.32-71.el6.x86_64 .....
```

*Note: The kernel version numbering is important here. The first number is the major version of the kernel. The second number is the major release of the first number. If the release number is even, which it is (6), it means that this is a stable release of the kernel. Odd numbers are development kernels and should not be used for production systems. The third number is the patch version of the kernel. The last number (71) is added by Red Hat to represent its release version of the kernel. Also note the el6, which tells you that you are running Red Hat Enterprice Linux 6. If you couldn't tell, this is an x64-bit version of the operating system.

→ To get the version of the currently installed kernel:

```
#rpm -qa | grep kernel ↵
```

→ you could also use the following:

```
#rpm -q kernel ↵
```

When it comes to working with kernels, you should be familiar with different locations. Let's look at four of these locations:

/boot : place where the kernel and boot files are kept.

/proc : current hardware configuration & status.

/usr/src : Source code of the kernel.

/lib/modules : kernel modules.

lsmod : Lists currently loaded kernel modules

Syn: lsmod

→ To look at what is currently loaded by the kernel since you booted the system, you use the following command:

```
#lsmod ↵
```

Module	Size	Used by
autofs4	29253	3
hidp	23105	2
rfcomm	42457	0
l2cap	29505	10 hidp, rfcomm
ext4	353979	2

[output truncated]

→ The preceding output is truncated due to size.

→ Shows the details of the ext4 kernel module listed previously:

modinfo: Displays information about a kernel module

```
#modinfo ext4 ↵
#modinfo cdrom ↵
```

→ To find all the kernel Modules:

→ All the kernel modules will be residing in /etc/lib/modules directory.

```
# cd /etc/lib/modules ↵
# ls ↵
```

→ To search all the kernel modules in the system using find Cmd:

```
# find / -name *.ko ↵ (Modules in the System will be ending with .ko extension)
```

→ To Remove

modprobe :→ To remove the loaded module

Syn: modprobe <modname>

#modprobe -r vfat ↵

→ Now to check

#lsmod | grep -i vfat ↵

→ To install/re-install a module:

#modprobe vfat ↵

#lsmod | grep -i vfat ↵

* Updating the kernel:

→ View the current version of the kernel:

#uname -r ↵

→ To view kernel package information

#yum info kernel ↵

(or)

#rpm -qa | grep kernel ↵

→ Using the package manager, you can upgrade the kernel to the latest version:

#yum update kernel -y ↵

(or)

#rpm -ivh kernel-2.6.18-194.3.1el5 ↵

*
Note: When updating a kernel with the rpm command, never use the -U option to update. The reason behind this is that the update option erases the prior kernel when updating, whereas the -i option installs the newer kernel alongside the old kernel. If something doesn't work or goes wrong, you have an older kernel to revert to.

Tuning the kernel with /proc/sys:

→ The kernel has a virtual file system, /proc/sys, that allows you to tune the kernel while the system is running.

→ The kernel creates the /proc/sys virtual file system when the system boots up, which holds all the parameters of the kernel. This virtual file system is then used to manipulate kernel parameters for testing purposes (these changes are valid only until the system reboots).

→ When you have the kernel tuned the way you'd like, you can simply have your settings applied when the system boots (through a special config file), or you can compile your own kernel to have them built in permanently

→ As you are testing kernel changes, make sure you don't rely on any settings made within the /proc/sys file system because they are erased when the system reboots. During testing, you can use the echo cmd to change the values of the kernel while the system is running.

Step ①: View the current value in the kernel:

```
#cat /proc/sys/net/ipv4/ip_forward ↵
```

Step ②: change the kernel option that controls packet forwarding:

```
#echo 1 > /proc/sys/net/ipv4/ip_forward ↵
```

Step ③: verify the value has changed:

```
#cat /proc/sys/net/ipv4/ip_forward ↵
```


→ If you want the changes to be persistent across system reboots, you can put the parameters you'd like to remain during boot in the /etc/sysctl.conf file.

Sysctl: Enables you to tune kernel parameters.

→ Before you start tuning things, let's look at all the available options:

```
#Sysctl -a ↵
```

→ Now, let's make the same changes to the kernel as before.

Step ①: Query the parameter responsible for forwarding packets within the kernel

```
#Sysctl -a | grep ip_forward ↵
```

Step ②: using Sysctl to change the option:

```
#Sysctl -w net.ipv4.ip_forward=1
```

↳ Enables you to change a settings in the sysctl config file.

Step ③: verify the value has been changed:

```
#Sysctl -a | grep ip_forward ↵
```

Step ④: Return the parameter to its original value:

```
#Sysctl -w net.ipv4.ip_forward=0 ↵
```

* Disks & partitioning *

→ Disk partitioning is one of the many steps you must take when preparing a system for use.

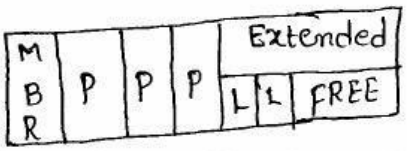
→ partitioning means to divide a single hard drive into many logicall drives.

→ In Each System the physical disk drivers are divided ~~into~~ up logically into partitions that allow you to store data on them.

→ there are also special types of partitions such as RAID that allow for increased performance, redundancy, or both. LVM, like RAID, is an advanced form of partitioning that eases management of partitions and makes growing them for increased storage capacity simple.

→ One of the key points to remember when walking with partitions is to always plan ahead

⇒ Disk partitioning Criteria:



MBR → Master Boot Record.

P → primary partition

Extended → Extended partition

L → Logical partition

Free → Free space.

→ Every disk can have only 3 primary partitions.

→ primary partition is a partition which usually holds the operating system

→ Extended partition is a special type of primary partition which can be subdivided into multiple logical partitions.

→ Logical partitions are the partitions which are created under Extended partitions

→ As in the real world, it is the results that matter. It doesn't matter whether you use Disk Druid, fdisk, or parted to create partitions. you can create new partitions at the command line or use GUI front ends to these tools such as the Disk Utility.

→ Remember Disk Druid is available only during the installation process.

→ you can use two different utilities when partitioning disks:

fdisk → Disk partitioning utility

parted → Another Disk partitioning utility.

→ while fdisk is more common, it is slowly being replaced by parted, which is more flexible.

⇒ **Disk identification:**

→ Different types of disks will be having different initials in linux

IDE drive will be shown as /dev/hda

SCSI/SATA drive will be shown as /dev/sda

Virtual drive will be shown as /dev/vda

Note: The first two letters represent whether the disk is a SCSI (Sd) (or) IDE (hd) disk. The third letter represents which disk it actually is. If there is a number after the three letters, it is the number of the partition.

→ To view information about the current partition layout,

#cat /proc/partitions | grep hd ↵ for IDE

#cat /proc/partitions | grep Sd ↵ for SCSI/SATA

→ you need to view their current partitions to see if any exist.

Syn: fdisk [options] [Device]

options: -b Specifies the sector size of the disk
-h number of heads on the disk
-l Lists current partition table.

Note: There are some limitations when it comes to working with partitions. you can have only four partitions to a physical disk with one exception. If you want to make more than the four, you need to create three primary partitions and one Extended partition, although the primary partitions aren't required for extended partition creation. The Extended partition can then hold 11 logical partitions (5-16) on it.

⇒ Creating a partition:

Step①: #fdisk /dev/sda ↵

Step②: view all the options available to you

Command (m for help): m ↵

p	print the partition table
n	add a new partition
d	Delete a partition
m	print this menu
q	quit without saving changes
t	change a partition's System id
w	write table to disk and exit

→ Create a new partition

#fdisk /dev/sda ↵

: n ↵

First Cylinder (1-1044, default 1): ↵

Using default value 1

Last cylinder or +Size ..... (1-1044, ...): +500M ↵

→ Create a second partition

: n ↵

→ Verify newly created partitions

:p ↵

→ write the changes to disk

:w ↵

→ Now that two new partitions have been created and written to disk, you should verify their existence. Before doing that, however, you want the kernel to reread the partition table to make sure that it recognizes all disks and partitions correctly. To do this, you use the partprobe cmd.

Syn: partprobe [options] [Device]

-d Does not actually inform the operating system

-s prints a summary of contents.

→ Now call the partprobe command

#partprobe /dev/sda ↵

→ Now that you have created partitions with the fdisk utility, do it again using the parted command

→ Create a partition:

#parted /dev/sda ↵

→ Menu (parted) help ↵

→ Create your first partition in a similar manner to fdisk:

(parted) mkpart ↵

partition type? primary/Extended/logicall?

File System type? [Ext2]?

Start?

End?

→ Make your second partition again

→ Before writing changes to disk, you should verify that they have been created the way you want them:

(parted) print

→ Exit the program to save your changes

(parted) quit↵

→ There are a few things you should notice here. First, you need to specify exactly where you want the start and end of the partition to be. If you don't plan this out ahead of time, you will end up with incorrect partition sizes. You should also take note of the fact that you don't have to write the changes to disk manually; this is done for you when you quit the parted program.

→ Again, you need to force the kernel to reread the partition table

partprobe

→ Once again, verify that your partitions have been created successfully:

parted -l ↵

⇒ Deleting a partition: Deleting a partition is much easier than creating one because you need to specify only the partition number that you want to delete.

→ Start the fdisk utility: # fdisk /dev/sda ↵

:p ↵ printout the current partition

:d↵ Delete a partition

:6↵ & want to delete 6th partition

:w↵ write changes to disk

→ Don't forget to reread the partition table

#partprobe /dev/sda↵

→ Start the parted utility: # parted /dev/sda ↵

(parted) print ↵

(parted) rm 5 ↵

(parted) quit ↵

* File System *

→ It is method of storing the data in an organized fashion on the disk. Every partition on the disk except MBR and Extended partition should be assigned with some file System in order to make them store the data.

→ File System is applied on the partition by formatting it with a particular type of a file System.

→ The number of file system types may exceed the number of operating systems. while RHEL can work with many of these formats, the default is Ext4. while many users enable other other file systems such as ReiserFS, Redhat may not support them.

→ Before the partitions can be used, however you need to create a file system for each one.

→ The default file System for RHEL5 is Ext3 and has been changed to Ext4 for RHEL6. Both of these file systems offer a journaling option, which has two main advantages.

(i) It can help speed up recovery if there is a disk failure because journalling file systems keep a "journal" of the file system's metadata.

(ii) It can check drivers faster during the system boot process.

→ The journaling feature isn't available on older file systems such as Ext2.

→ The first Linux operating systems used the Extended file System (ext) until the last few years, Red Hat linux operating systems formatted their partitions by default to the seconded Extended file System (Ext2). For RHEL5, the default was the third Extended file System (Ext3). The new default for RHEL6 is the fourth Extend file System (Ext4).

→ Ext file system is the widely used file system in Linux, whereas vfat is the file system to maintain a common storage between Linux and windows (in case of multiple o/s)

Ext2	Ext3	Ext4
① Stands for second Extend file System.	→ Third Extend File System	→ Third Extended file System.
② Introduced in 1993	→ Introduced in 2001	→ Introduced in 2008.
③ Does not have journaling	→ Support journaling	→ Support journaling
④ Maximum file size can be from 16GB to 2TB	→ 16GB to 2TB	→ 16GB to 16TB
⑤ Maximum Ext2 file System size can be from 2TB to 32TB.	→ 2 TB to 32 TB	→ Maximum Ext4 file System size is 1 EB (Exabyte). 1 EB = 1024 PB (peta byte) 1 PB = 1024 TB (Tera byte)

→ There are many types of file Systems

Swap: The Linux swap filesystem is associated with dedicated swap partitions. you've probably created at least one swap partition when you installed RHEL.

MS-DOS & VFAT: these file Systems allow you to read MS-DOS ~~command~~ formatted File Systems. MS-DOS lets you read pre-windows 95 partitions, or regular Windows partitions within the limits of short filenames. VFAT lets you read windows 9x/NT/2000/vista/7 partitions formatted to the FAT16 or FAT32 file Systems.

ISO 9660: The standard file System for CD-ROMs. It is also known as the High sierra file System, or HSFS, on the unix systems.

/proc: A Linux virtual filesystem. Virtual means that it doesn't occupy real disk space. Instead, files are created as needed. Used to provide information on kernel configuration and device status.

/dev/pts: The Linux implementation of the Open Group's Unix98 PTY Support.

JFS: IBM's Journaled filesystem, commonly used on IBM Enterprice Servers.

ReiserFS: The ReiserFS file system is resizable and supports fast journaling. It's more efficient when most of the files are very small and very large. It's based on the concept of "balanced trees". It is no longer supported by RHEL, or even by its former main proponent, SUSE.

xfs: Developed by Silicon Graphics as a journaling filesystem, it supports very large files; as of this writing, xfs files are limited to 9×10^{18} bytes. Do not confuse this file system with the X font server, both use the same acronym.

NTFS: The current Microsoft Windows file system.

⇒ **Creating a file System:**

→ When you're creating a file System, there are many different ways to complete the same task.

→ They're all based on the mkfs command, which works as a front end to filesystem-specific commands such as mkfs.Ext2, mkfs.Ext3, and mkfs.Ext4.

Syn: mkfs [options] [Device]

-j creates a journal option

-m Specifies a reserved percentage of blocks on a file system.

→ There are two ways to apply formatting on a volume. For Example, If you've just created a partition on /dev/sda5

#mkfs -t Ext4 /dev/sda5

#mke2fs -t Ext4 /dev/sda5

#mkfs.Ext4 /dev/sda5.

* → If you want to reformat an Existing partition, Logical volume, or RAID array, take the following precautions.

- Backup any Existing data on the partition
- Unmount the partition.

→ you can format partitions, Logical volumes, and RAID arrays to other filesystems. The options available in RHEL 6 include:

- mkfs.cramfs — creates a compressed Rom filesystem.
- mkfs.Ext2 — formats a volume to the Ext2 filesystem.
- mkfs.Ext3 — formats a volume to the Ext3 filesystem.
- mkfs.Ext4 — formats a volume to the Ext4 filesystem.
- mkfs.msdos (or) - mkfs.vfat (or) - mkdosfs — formats a partition to the microsoft compatible VFAT filesystem; it does not create bootable filesystems.
- mkfs.xfs — formats a volume to the xfs filesystem developed by the former Silicon Graphics.
- mkswap — formats a volume to the Linux swap file System.

→ one advantage of some rebuild distributions is the availability of useful packages not supported by or available from Red Hat. For Example, CentOS 6 includes the ntfsprogs package, which supports the mounting of NTFS partitions.

⇒ Creating a Swap :-

→ In Linux, a swap space is used as a "scratch space" for the System. When the System runs low on memory, it uses the swap as a virtual memory area to swap items in and out of physical memory. Although it should not be used in place of physical memory because it is much slower, it's critical piece of any system.

→ there are two different types of swaps that you can have:

① File swap
② partition swap

partition swap:

Step ①: → Create a partition

```
#fdisk /dev/sda ↵
```

Step ②: → update to kernel

```
#partprobe /dev/sda ↵
```

Step ③: → Use the mkswap command to create a swap space

Syn: mkswap [options] [Device]

-c checks the device for bad blocks before creating the swap area.

```
#mkswap /dev/sda8 ↵
```

Step ④: → Enable the swap partition

```
#swapon /dev/sda8 ↵
```

Step ⑥: → Verify the swap is running correctly

```
#swapon -s ↵
```

Syn: Swapon [options] [Device]

-a Enables all swap devices

-e Silently skips devices that don't Exist

-s Verifies that the swap is running

Step ⑦: → If you want to turn off the swap, you can use the Swapoff command.

Syn: Swapoff [options] [Device]

-a Enables all swap devices

-e Silently skips devices that don't Exist

-s Verifies that the swap is running

File swap: you can use the dd command to reserve space for another swap on the /dev/sda9 partition.

→ The dd command can be used for many different purposes and has a huge syntax

Step ①: → Reserve 1GB of space for the swap

```
#dd if=/dev/zero of=/mnt/file_swap bs=1024 count=1000000
```

Step ②: → Just as with partition swaps, you can now create a swap space specifying the device file just created

```
#mkSwap /mnt/file_swap↵
```

Step ③: → Enable the swap

```
#Swapon /mnt/file_swap
```

Step ④: → Again you can verify that the swap is enabled

```
#Swapon -s↵
```

Note:

* → The big difference between the two swap types is that file swap is easier to manage because you can just move the swap file to another disk if you want. The swap partition would need to be removed, re-created, and so on. Although Red Hat recommends using a partition swap, file swaps are fast enough these days with less administrative overhead to not use them instead. One word of caution, though, is that you can use only one swap (of either type) per physical disk.

⇒ Mounting a file System:

→ After formatting a partition we cannot add the data into the partition. In order to add the data in the partition it is required to be mounted.

→ Mounting is a procedure where we attach a directory to the file system.

→ they can be mounted to any directory, which is refferred to as a mount point. Every mount point before is a directory.

→ If you mount a file system on a directory that is not Empty, Everything within that directory becomes inaccessible. Therefore, you should create a new directory as a mount point for each of your file systems.

→ there are only two commands for mounting a file systems:

mount	Mounts a file System.
umount	Unmounts a file System.

Step①:→ Start by going to the /opt directory, where you can make some directorys to serve as a mount points.

```
#cd /opt↵
#mkdir company_data ↵
#mkdir backup ↵
```

Syn: mount [options] [Device] [Mount_point]

-r	Mounts as read-only
-w	Mounts as read/write (the default)
-L LABEL	Mounts the file System with the name LABEL
-v	provides verbose output.

Step②: → Mount the two file systems

```
#mount /dev/sda6 /opt/company_data ↵
#mount /dev/sda7 /opt/backup ↵
```

*→ Notice that you don't specify a file system type or any mount options. The reason is that the mount command automatically detects the file system type and mounts it for you. By default, the file system is also mounted with the defaults option (rw).

Step③: → To unmount a file systems:

Syn: umount [options] [Mount_point]

-f force unmount.

-v provides verbose output.

Step④: → you can use the "fuser and lsof" commands to check for open files and users that are currently using files on a file system

Syn: fuser [options] [Mount_point / file system]

-c checks the mounted file system

-k kills processes using the file system

-m shows all processes using the file system.

-u Displays user IDs

-v Verbose output.

→ Check to see what users are currently using the file system

```
#fuser -cu /dev/sda6 ↵
```

(or)

```
#lsof /dev/sda6 ↵
```

→ To kill the open connections, you can use the fuser cmd again:

```
#fuser -ck /opt/backup ↵
```

→ Now you should be able to unmount the file system:

```
#umount /opt/backup ↵
```

→ * Now you know how to mount and unmount file systems, but there is something else you need to look at. If you reboot your system right now, all the file systems that you just mounted will no longer be available when the system comes back up. Why? The mount command is not persistent, so anything that is mounted with it will no longer be available across system reboots. I suppose you want to know how to fix that, right? The system looks at two config files

/etc/mtab contains a list of all currently mounted file systems

/etc/fstab Mounts all listed file systems with given options at boot time.

→ View the /etc/mtab file:

```
#cat /etc/mtab ↵
```

Every time you mount or unmount a file system, this file is updated to always reflect what is currently mounted on the system

→ you can also query to check whether a particular file system is mounted

```
#cat /etc/mtab | grep backup ↵
```

→ you can use the mount command with no options to also view the currently mounted file systems:

#mount ↵

→ Go through the /etc/fstab file. The file follows this syntax:

<device> <mountpoint> <file system type> <mount options> <write data during shutdown>

<check sequence>

→ View the /etc/fstab file:

#cat /etc/fstab ↵

*→ The first three fields should be fairly obvious because you have been working with them throughout the chapter. The fourth field defines the options that you can use to mount the file system. The fifth field defines whether data should be backup (also called dumping) before a system shutdown or reboot occurs. This field commonly use a value of 1. A value of 0 might be used if the file system is a temporary storage space for files, such as /tmp. The last field defines the order in which file system checking should take place. For the root file system, the value should be 1; everything else should be 2. If you have a removable file system (CD-ROM (or) External) you can define a value of 0 and skip the checking altogether. Because you want the two file systems created earlier to be mounted when the system boots, you can add two definitions for them here.

→ open the /etc/fstab file for editing:

#vim /etc/fstab ↵

/dev/sda6 /opt/backup Ext3 defaults 0 0

:wq! ↵

→ you can use the mount command with -a options to mount all file systems defined in the /etc/fstab file

#mount -a ↵

⇒ Extra file System commands:

Label: Labels enable you to determine a specific file system more easily with a common name, instead of /dev/sda6. An added benfit is the system's being able to keep its label even if the underlying disk is switched with a new one.

Step①: → Take your file system offline

#umount /dev/sda6 ↵

Step②: → Let's label the file system ~~CDdata~~ ~~CD~~data to denote that it's the company_data file system.

#e2label /dev/sda6 CData ↵

Step③: → you can use the same command to also verify:

#e2label /dev/sda6 ↵

Step④: → Find the file system you just labelled:

Syn: findfs LABEL=<label> | UUID=<uuid>

#findfs LABEL=CData ↵

→ you can also query more information about the device using the blkid command.

Syn: blkid [options]

-s Shows specified tag(s)

dev Specifies the device to probe.

Step⑤ :→ Combine the blkid cmd with grep for specific results

#blkid | grep cdata↵

Step⑥ :→ When you finish your maintenance, you can remount the file system with the new label instead of the device path:

#mount LABEL=CData /opt/company_data↵

→ you could even update the /etc/fstab file to use the label information instead of the device path.

#vim /etc/fstab↵

LABEL=CData /opt/company_data Ext3 defaults 0 0

:wq!↵

→ you can use the mount command to verify the label names

#mount -l↵

→ you also can use the df command to view the usage information for your file systems:

Syn: df [options]

-h specifies human-readable format

-l Local file systems only

-T print the file system type

#df -h↵

#df -Th↵

⇒ Managing file System quotas :-

→ Quotas are used to restrict the amount of disk space occupied by users or groups.

→ Quotas regulates disk consumption of users. It improves System performance.

→ Quotas are two types

① user level

② Group level

→ If we apply quotas on a group level it will Effected to only the primary users of that group.

→ quotas can be applied only quotas enabled partitions.

→ you need to Install the required packages before you can use quotas on your System.

Step ①: → To Install the quota package

#yum Install -y quota ↵

Step ②: → Verify that the package was Installed Successfully

#rpm -qa | grep quota

Step ③: → you can query quota support from the kernel with the following command

#grep -i config_quota /boot/config-`uname -r`

→ Now that you have a listing of the commands you can use, you first need to edit the /etc/fstab file to specify which file systems you want to utilize quotas.

Step ①: → open the /etc/fstab file, Edit the following line

/dev/sda6 /opt/company_data Ext3 defaults, usrquota, grpquota

:wq! ↵

→ Now you need to remount the /opt/company_data file System before the changes take effect.

Step ②: → You can accomplish this by using the mount command:

#mount -o remount /opt/company_data ↵

Step ③: → You can verify that the mount and quota options took correctly

#mount | grep company_data ↵

→ There are two files that maintain quotas for users and groups.

aquota.users	users quota file.
aquota.group	Group quota file.

→ These two files are automatically created in the top-level directory of the file System where you are turning on quotas - in this case, the /opt/company_data file System.

Step ④: → To start the quota System, you use the quotacheck cmd.

Syn: quotacheck [options] [partition]

-c	Don't read Existing quota files
-u	Checks only user quotas
-g	Checks only group quotas
-m	Doesn't remount the file System as read-only.
-v	provides verbose output.

#quotacheck -ugm /opt/company_data ↵

→ To verify that the quota files were created successfully

#ls /opt/company_data ↵

→ Enabling quotas: Normally, you would have to call the quotaon and quotaoff cmds to have the quota system enforced, but they are automatically called when the system boots up and shuts down.

Step ⑤: → Run the cmd manually the first time just to make sure that quotas turned on:

```
#quotaon -v /opt/company_data↵
```


→ Let's briefly discuss the two different limits you can have when dealing with quotas:

Soft Limit: Has a grace period that acts as an alarm, signaling when you are reaching your limit. If your grace period expires, you are required to delete files until you are once again under your limit. If you don't specify a grace period, the soft limit is the maximum number of files you can have.

Hard Limit: Required only when a grace period exists for soft limits. If the hard limit does exist, it is the maximum limit that you can hit before your grace period expires on the soft limit.

→ To work with quotas for users and groups, you need to do some conversions in your head here. Each block is equal to 1KB. If you aren't good at the conversions, remember that 1,000KB = 1MB.

Step ⑥: → Set the limits for user1 by using the edquota cmd.

Syn: edquota [-u/-g] [username/groupname]

```
#edquota -u user1↵
```

filesystem	blocks	soft	hard	inodes	soft	hard
/dev/sda6	0	20000	25000	0	0	0

Step ⑦: Again, you use the edquota Cmd, but with a different option:

#edquota -t ↵

→ Here, the current value is seven days for the block grace period. you should not give your users that much time to get their act together, so drop that limit to two days.

Tip: The edquota Cmd offers a pretty cool feature. After you configure a quota and your limits for a single user, you can actually copy this over to other users as if it were a template. To do this, specify the user you want to use as a template first and call the edquota Cmd with the -p option.

#edquota -up user1 user2 user3 ↵

Step ⑧: Quota usage Reports:

Syn: repquota [options] [partitions]

-a Reports on all non-NFS file systems with quotas turned on

-u Reports on user quotas.

-g Reports on group quotas.

-v verbose output.

#repquota -uv /opt/company_data ↵

—X—

⇒ **File System Security:** Linux, like most operating systems, has a standard set of file permissions. Aside from these, it also has a more refined set of permissions implemented through access control lists.

→ This section covers both of these topics and how they are used to implement file system security for files, directories, and more.

Step①: Installing the required package

```
#yum install -y acl ↵
```

Step②: Verify the package installation:

```
#rpm -qa | grep acl
```

Step③: Before you can even use ACL's however, you need to make sure that the file system has been mounted with ACL parameter:

```
#mount | grep acl ↵
```

Step④: You can accomplish this using the following

```
#mount -t ext3 -o acl,remount /dev/sda7 /opt/backup ↵
```

Step⑤: If your file system isn't already mounted, you could also use the following:

```
#mount -t Ext3 -o acl /dev/sda7 /opt/backup ↵
```

Step⑥: To verify, you can use the previous cmd:

```
#mount | grep acl
```

Step⑦: Adjust the following line in your /etc/fstab file.

```
/dev/sda7   /opt/backup   Ext3   defaults,acl   1 2
```

```
:wq! ↵
```

Step ⑧: → To make the changes take effect, you need to remount the filesyst

#mount -o remount /opt/backup ↵

→ Now verify that your file System has the ACL options:

#mount | grep -i acl ↵

→ The file System is now mounted properly with the ACL option, so can start to look at the management cmds that pertain to ACL's:

getfacl Obtains the ACL from a file or directory

setfacl Sets or modifies an ACL.

Step ①: Create a sample file on which you can test an ACL in the /opt/backu

#cd /opt/backup ↵

#touch file1 ↵

→ Now you can use the getfacl cmd to view the ACL currently associate with the file.

Syn: getfacl [options] file

-d Displays the default ACL

-R Recurses into Subdirectories.

#getfacl file1 ↵

Syn: setfacl [options] file

-m Modifies an ACL

-x Removes an ACL

-n Doesn't recalculate the mask

-R Recurses into Subdirectories.

Step ②: → Set the test file so that user1 also has access to this file

```
#setfacl -m u:user1:rwx /opt/backup/file1↵
```

→ To check the ACL permissions again:

```
#getfacl file1 ↵
```

Step ③: To remove the ACL for user1:

```
#setfacl -x u:user1 /opt/backup/file1
```

→ Verify the ACL has been removed:

```
#getfacl file1↵
```

Step ④: → If you have multiple ACL set up on a single file, you can remove them all with the -b option instead of removing them one by one:

```
#setfacl -b testfile ↵
```

File permissions and ACLs can get really complex if they aren't thought out ahead of time.

—x—

⇒ Logical volume Manager (LVM):

→ LVM is a form of advanced partition management. The benfit to using LVM is ease of management due to the way disks are setup.

→ LVM is a method of allocating hard drive space in to Logical volumes that can be easily resized of partition with LVM, the hard drive (or) set of hard drives are allocated to one (or) more physical volumes.

→ The physical volumes are combined into volume groups such volume group is divided into logical volumes which are assigned mount points as "/home', '/' etc. These logical volumes are formatted to Ext3 file System.

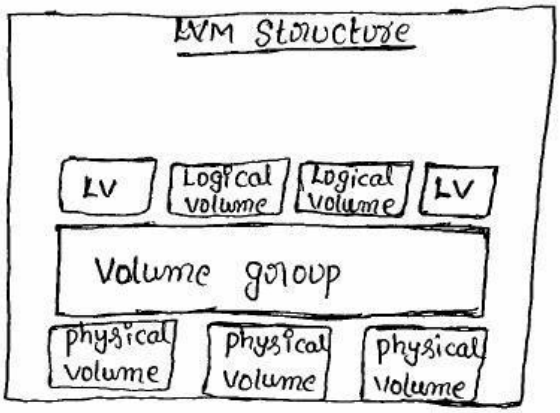
→ The LVM must follow the bellow sequence:

① physical volume.
② Volume group.
③ Logical volume.

physical volume: The collection of individual physical drives are called as physical volumes.

Volume group: It is a collection of physical volumes and assign a name through which we can create logical volumes.

Logical volume: → The logical volumes are specified from the volume group. These are logical partitions which can resize, format, mount etc.



Implementation of LVM:

Step①: → Install the required packages:

```
#yum install -y lvm* ↵
```

Step②: Verify that it is installed

```
#rpm -qa | grep lvm
```

Step③: creating an Lvm partitions: (four partitions)

```
#fdisk /dev/sda ↵
```

→ To update to kernel for rereading

```
#partprobe /dev/sda ↵
```

creating an LVM partition:

Step ④: To create physical volumes:

```
#pvcreate /dev/sda{10,11,12,13}
```

→ verify that the physical volume was created successfully:

```
#pvdisplay /dev/sda10 ↵
```

Step⑤: → To create volume group:

```
#vgcreate India /dev/sda{10,11,12}
```

→ verify that the volume group was created successfully:

```
#vgdisplay -v India ↵
```

→ When volume groups are created and initialized, the physical volumes are broken down into physical Extends (the unit of measurement for LVM). This is significant because you can adjust the size of how data is stored based on the size of each physical extend, defined when the volume group is created (the default is 4MB).

Step⑥: Create logical volumes:

→ To create a logical volumes, use the lvcreate cmd and specify the size of the partition that you'd like to create. The size can be specified in kilobytes, megabytes, gigabytes, or logical extents (LE). Like physical extents, logical extents are a unit of measure when dealing with logical volumes.

→ create a partition 2GB in size

#lvcreate -L 2000 -n india ap (-n → new)

→ To verify logical volume info,

#lvdisplay ↵ (or) #lvs

→ To create one more logical volume

#lvcreate -L 3000 -n india mp↵

→ Using the Lvrename cmd you can change the name of a logical partition

#lvrename /dev/india/mp /dev/india/up↵

→ verify with the following cmd

#lvdisplay↵

Adjusting the size of Lvm partitions:

→ The single best feature of Lvm is that you can reduce or Expand your logical volumes and volume groups. If you are running out of room on a particular logical volume (or) volume group, you can add another physical volume to the volume group and then expand a logical volume to give you more room.

Step①: Add 2GB more to the ap logical volume

#lvextend -L +2000 /dev/india/ap↵

#lvExtend -L (or) 2000 /dev/india/ap↵

→ verify the change with the following cmd:

#lvdisplay /india↵

Step②: → To decrease a logical volume

#lvresize -L 2000 /dev/india/ap↵

(or)

#lvreduce -L-2000 /dev/india/ap↵

Step③: → Suppose, through, that you want to add a new physical volume so that you can extend your volume group.

→ Create a new physical volume somewhere

#pvcreate /dev/sda15↵

→ Now Extend your volume group to incorporate that new physical volu

#vgextend /india /dev/sda15↵

→ Now verify the details of the newly increased vg:

#vgdisplay -v /india↵

Step④: To reduce the volume group to no longer include the physical volume /dev/sda15, you can use the vgreduce cmd:

#vgreduce /india /dev/sda15↵

→ Now verify expansion or reduction of volume groups

#vgdisplay /india↵

LINUX PROGRAMMING

Topic:

Migrating Data

Contributed By:

Nagi Reddy

Migrating Data: Suppose you have a drive that is old or dying and you'd like to remove it from the system. on a system with normal partitions, you would have to copy all the data from one disk to another while the disk is offline (because of files locks). Having LVM makes this easier because you can migrate your data from one disk to another, even while the disk is online! This capability is very useful when you need to replace a disk.

→ If you want to replace /dev/sda14 (dev/sda) because it's failing, you can use the pvmove cmd to migrate the physical extents (which is really your data) to an other physical volume (/dev/sda15). (/dev/sdb)

Step①: → To create physical volume

#pvcreate /dev/sda15↵

Step②: → you need to add back /dev/sda15 to the vg:

~~vgs~~ vgextend fedia /dev/sda15↵

Step③: → Also create a logical volume to hold the migrate data:

#lvcreate -L 3000 fedia -n ban↵

Step④: → verify all logical volumes are in place

#lvdisplay fedia↵

Step⑤: → Migrate the data from the "dying" drive

#pvmove /dev/sda14 /dev/sda15↵
(/dev/sda) (dev/sdb)

Note: Make sure that you have more than one physical volume; otherwise there will be nowhere for the data to move.

Step⑥: verify that physical volume is Empty

#pvdisplay /dev/sda14↵
(dev/sda)

Deleting an Lvm partition:

It just as important to understand how to delete Lvm partitions as it to create them. This is a common task when you are upgrading or redesigning a file system layout.

Step ①: → To remove a logical volume

```
#lvremove /dev/India/ap ↵
```

* → Although this advice should common sense, make sure you back up any data before deleting anything within the Lvm structure.

Step ②: → To remove the volume group:

```
#vgremove @ndia
```

→ you can also do both steps in one cmd by using the -f option

```
#vgremove -f @ndia ↵
```

Step ③: Wipe all the current physical volumes:

```
#pvremove /dev/Sda10 ↵
#pvremove /dev/Sda11 ↵
```

Note: use the resize2fs cmd to Extend the file System. Before Extending the file System, however, you should always ensure the integrity of the file System first with the e2fsck cmd.

Step ①: Syn: e2fsck [options] [Device]

-p	Automatically repairs (no questions)
-n	Makes no changes to the file System
-y	Assumes "yes" to all questions
-f	Force checking of the file System
-v	provides verbose output.

→ Check the file system

#e2fsck -f /dev/Sndia/ap ↵

Step②:

Syn: resize2fs [options] [Device]

-p points percentage as task completes

-f force the cmd to proceed.

→ Extend the underlying logical volume:

#lvextend -L 3000 /dev/Sndia/ap ↵

→ Now you can extend the file System

#resize2fs -p /dev/Sndia/ap ↵

Step③:

→ Now that your maintenance is complete, remount the file System:

#mount /dev/Sndia/ap /mnt ↵

→ you can use the mount cmd to verify it mounted successfully:

#mount ↵

→ Now you can use the df cmd to view the usage information for your file systems. this should also reflect the additional space that you just added to the /mnt file System.

Syn: df [options]

-h specifies human-readable format

-T points the file System type

#df -h ↵

RAID: Now let's move on to the final type of advanced partitioning: RAID

→ RAID means Redundant Array of Independent Disk

→ RAID partitions allow for more advanced features such as redundancy and better performance.

→ Mainly we implement the Raid inorder to increase the storage capacity along with data security.

→ There are two types of RAID's.

① Hardware raid.

② Software raid.

→ While RAID can be implemented at the hardware level, the Red Hat exams are not hardware based and therefore focus on the Software implementation of RAID through the MD driver.

→ Before we describe how to implement RAID, let's look at the different types of RAID:

RAID 0: (Striping) Disks are grouped together to form one large drive. This offers better performance at the cost of availability. Should any single disk in the RAID fail, the entire set of disks becomes unusable.

- Minimum 2, Max 32 Hard disks
- Data is written alternatively
- No fault tolerance.
- Read & write speed is fast.

RAID 1: (Mirroring) Disks are copied from one to another, allowing for redundancy. Should one disk fail, the other disk takes over, having an exact copy of data from the original disk.

- Min 2, Max 32 hard disks.
- Data is written simultaneously.
- fault tolerance available
- Read fast, write slow.

RAID 5 : (Striping with parity) Disks are similar to RAID 0 and are join together to form one large drive. The difference here is that 25% of the disk is used for a parity bit, which allows the disks to be recovered should a single disk fail.

- Min 3, Max 32 hard disks
- Data is written alternatively
- parity is written on all disks
- Read & write speed is fast.
- fault tolerance is available.

Implementation of RAID 5 :

Step ①: Install the following package

```
#yum install -y mdadm ↵
```

Step ②: Verify the install

```
#rpm -qa | grep mdadm ↵
```

→ To start, you first need to create partitions on the disk you want to use. you start with a RAID 5 setup, so you need to make partitions on at least three different disks.

Creating a RAID Array :

→ create three partitions `#fdisk /dev/sda ↵`

→ To verify when you're done `#fdisk -l ↵`

→ Now you can begin to set up the RAID 5 array with the three partitions

Step ①: **Syn:** mdadm [options]

-a	Add a disk into a current array
-C	Create a new RAID array
-D	prints the details of array
-f	fails a disk in the array
-l	Specifies level of RAID array to create
-n	Specifies the devices in the RAID array.

-S Stops an array

-A Activate an array

-v provides verbose output.

#mdadm -Cv /dev/md0 -n3 /dev/sda1 /dev/sdb1 /dev/sdc1 -l5↵

Step ②: Again to verify that the RAID array has been created successfully

#mdadm -D /dev/md0↵

Step ③: view the status of the newly created RAID array:

#cat /proc/mdstat↵

This output shows that you have an active RAID 5 array with three disks in it. The last few lines here show the state of each disk and partition in the RAID array. You can also see that the RAID is in "Recovery" mode, or creating itself.

Step ④: If you wait the estimated 2.9 minutes and then query again, you see the following

#cat /proc/mdstat↵

You now see that the RAID is good to go as it has finished building itself.

What to DO When a Disk fails:

Suppose that a disk in the array fails. In that case, you need to remove that disk from the array and replace it with a working one.

Step ①: → Manually fail a disk in the array,

#mdadm /dev/md0 -f /dev/sdc1↵

Step ②: verify that the disk in the array has failed

#mdadm -D /dev/md0↵

Step ③: To remove a disk from the array

#mdadm /dev/md0 -r /dev/sdc1↵

Step ④: Look at the last few lines of the RAID details again

#mdadm -D /dev/md0↵

→ If you want, you could combine the previous two commands

#mdadm -v /dev/md0 -f /dev/sdc1 -r /dev/sdc1↵

Step ⑤: When the disk is partitioned, you can add it back to the array

#mdadm /dev/md0 -a /dev/sdd1↵

→ verify that it has been added properly

#mdadm -D /dev/md0↵

Step ⑥: Query the kernel

#cat /proc/mdstat↵

Step ⑦: Should something go seriously wrong and you need to take RAID array offline completely

#mdadm -vS /dev/md0

Deleting a RAID Array:

Step ①: To delete an array, first stop it

#mdadm -vS /dev/md0

Step ②: Then remove the RAID array device

#mdadm -r /dev/md0↵

—×—

System Initialization

Boot process: Boot process consists the set of processes from power on the pc to login prompt comes.

→ when a computer boots up, the BIOS is the first program that is run. After it is loaded, the BIOS begins to test the system through the power on Self Test (POST) and then starts loading peripheral devices. The BIOS then looks for the boot device and passes control to it. The boot device contains the Master boot record (MBR), which starts to boot the system via the bootloader. From here, the Grand unified Bootloader (GRUB) looks to boot into the kernel that is labeled as the default. Finally, the kernel calls the init process, which boots up the rest of the system.

→ The GRUB has become the default bootloader for Red Hat, Ubuntu, and many other versions of Linux as well.

→ When GRUB loads, you are given a list of kernels and additional operating systems from which you can choose to boot.

→ By default, there is a configurable 5-second timeout value that choose the default kernel if you don't make a selection and the timeout threshold is reached. After GRUB loads the kernel, it passes control over to the kernel, which it run begins to initialize and configure the computer's hardware.

*→ During the boot process, everything is logged to the /var/log/dmesg file. you can also use the dmesg cmd to query information about the boot process after the system has booted.

→ When the system's drivers are in place, the kernel executes the /sbin/init program.

*→ In RHEL6, the boot process has been replaced by a new utility called upstart instead of the traditional SysV init style scripts. This utility decreases the time that it takes the system to boot and is already currently being used on other versions of Linux such as Ubuntu.

*→ The init program is the first process created by the kernel. It is responsible for the rest of the boot process and setting up the environment for the user.

→ First, it consults the /etc/inittab file, which defines how the rest of the boot process will go. The /etc/inittab file lists the default runlevel to boot into and the system initialization script (/etc/rc.d/rc.sysinit).

→ Let's look at the /etc/inittab file to see what the init process goes through

```
#cat /etc/inittab ↵
```

→ You can see that the default runlevel is set to 5, although six different runlevels are listed. The /etc/inittab file also defines how to handle power failures and virtual terminals. After the init process is done consulting the /etc/inittab file, the /etc/rc.d/rc.sysinit script is run, which handles setting the system clock, networking, setting up the user environment & more.

→ On Red Hat Linux, the default run level is 5. This default runlevel is passed to the /etc/rc.d/rc script, which calls all the programs in the /etc/rc.d/rc#.d directory.

→ The last thing that you should see is the login prompt. If you have a desktop manager installed such as Gnome, you should see a GUI login screen where you can login to the system; otherwise, you see a text mode login.

⇒ **Working with Grub:** The GRUB bootloader is broken down into different stages. The code contained on the MBR is considered GRUB stage 1. It loads GRUB stage 1.5, which tries to identify the file system type (optional), or it can call GRUB stage 2 directly. stage 2 is what calls the kernel and loads it into memory. On stage 1, GRUB needs to search the MBR looking for an active partitions from which to boot the kernel. GRUB has its own format for looking through hard disks.

→ The Syntax of this format is

(xdn[,m]) where xd is the drive
n is the number of the disk
m denotes the partition number.

→ The Syntax is very useful when troubleshooting issues with GRUB because you need to know how GRUB searches for disk drives when trying to find the primary partition. When the primary partition found, GRUB loads the kernel, which is where you move on to stage2. Stage2 is the place where you will tend to spend the most time troubleshooting boot issues with the system. As stage2 is starts, it presents you ~~which~~ with a list of kernel's that you can boot from, along with a listing of options that you can use to modify the parameters passed to the kernel during bootup.

Grub Boot options:

e	Edit the cmds before booting
a	Modify (or) append the kernel arguments before booting
c	Open the GRUB cmd line.

→ you can use 'a' option to modify any parameters you want to pass to the kernel. This includes changing the runlevel that the system will boot into.

→ After choosing the 'a' option, you can pass a mode as a parameter to enter into the mode. Here are the different modes that you can boot into:

Single-User Mode	perform maintenance tasks (or) forget the root password.
Runlevel 2 (or) 3	Load only partial services during the boot process
Emergency Mode	Used to perform tasks on an unbootable system.
Rescue Mode	Used to fix boot issues (or) reinstall GRUB.

The config file:

→ GRUB has only a single config file, /boot/grub/grub.conf. Two other files actually have soft links to this main config file as well: /boot/grub/menu.lst and /etc/grub.conf. When GRUB starts, it reads its configuration from the main config file

```
#cat /boot/grub/grub.conf
```


→ Using the c option to enter the GRUB cmd line, you can make changes to the config file, reinstall GRUB, or repair a broken config file.

The GRUB Command Line:

→ How to repair a broken MBR.

You need to make use of the rescue environment, which can be found by booting from the RHEL installation DVD. When you are in the rescue environment, you can repair your broken MBR.

Step ①: Load up the GRUB cmd line to find the disk and partition that contain the grub.conf file using the find cmd:

```
grub> find /grub/grub.conf ↵
      (hd0,0)
```


→ you could also run:

```
grub> root ↵
      (hd0,0)
```

Step ②: Install GRUB on the drive is returned:

```
grub> setup (hd0)
```

→ Now that your MBR is fixed, you should be able to boot the system once again.

⇒ Runlevels:

→ When the system boots up, it queries for the default runlevel, which is defined in the /etc/inittab file. When the default runlevel is located, the system boots into that particular runlevel.

→ There are six runlevels in total, which are shown in the /etc/inittab file. Each runlevel also has a directory called /etc/rc.d/rc#.d, where # is the runlevel (from 0 to 6).

→ Let's look at the different runlevels:

0	Halt
1	Single user mode
2	Multiuser with partial services
3	Full multiuser with networking (text mode)
4	Not used
5	Full multiuser graphical mode (GUI desktop login)
6	Reboot

→ The easiest runlevels to understand are 0 and 6. These two runlevels are called by the same cmd with different input. In runlevel 0, essentially the system is off. In runlevel 6, the system is restarting. Runlevel 1 is used to enter single-user mode, which you would enter if there are issues with the system and you'd like to perform maintenance. You can also reset the root user's password in this runlevel. The remaining runlevels provide various states for different services to run in.

→ Don't forget the the upstart is the program that actually starts & stops services at each runlevel now. The /etc/init/rc.conf file shows how each set of scripts is called:

```
#cat /etc/init/rc.conf↵
```

Runlevel Utilities:

→ Let's now look at the many system utilities that help you manage the system in different runlevels. These management cmds are

Syn: Shutdown [options] time

-k	Doesn't shutdown; just warns
-h	Halts the system after shutdown
-r	Reboots instead of turning off the system.
-F	forces a file system check on reboot.
-n	Kills all processes quickly (not recommended)
-t secs	Sends a shutdown message but delays shutdown by x seconds.

Ex: #Shutdown -h now ↵

#Shutdown -r now ↵

#reboot ↵

#Shutdown -h 120 ↵ delay the shutdown by 2 Minutes.

Halt: powers down the system.

→ To turn of the system #Shutdown now (or) #halt ↵

poweroff: works the same as the halt cmd.

→ To check the current runlevel #runlevel ↵

(or)

#who -r ↵

→ To change runlevel 3 #init 3 ↵

Troubleshooting

I lost My Root user password:

Step ①: Boot into Single-user mode by appending the command line during boot with the following

Single

Step ②: When you are presented with a command prompt, change the root user password:

passwd root ↵

Step ③: Reboot the system and validate that the new root password works correctly:

#reboot ↵

password change Not Available in Single-user Mode:

When you enter Single-user mode, you may encounter an issue where the root user's password can't be changed.

Step ①: verify the existence of the /etc/shadow file:

#ls /etc/ | grep shadow ↵

Step ②: If the /etc/shadow file doesn't Exist, use the pwconv Cmd to re-create the /etc/shadow file:

#pwconv ↵

Step ③: Now execute the passwd command to reset or change the root user's password:

```
#passwd root ↵
```

Step ④: Reboot the system and validate that the new root password works correctly:

```
#reboot ↵
```

The MBR is corrupt:

If you are having trouble booting the system and you have determined that the master boot record (MBR) is corrupt, you need to boot into rescue mode. Use the Red Hat DVD, boot from it, and choose the option to enter rescue mode.

Step ①: After you boot, enter the GRUB shell:

```
#grub ↵
```

Step ②: Locate the root drive:

```
grub> root ↵
```

Step ③: Reinstall the MBR from the GRUB shell:

```
grub> setup (hd0)
```

Step ④: Reboot the system to validate that the system boots properly:

```
#reboot ↵
```

Repair the file System :

When we have a inconsistency to the file /etc/fstab and the file systems listed by blkid utility.

Then the system will go to rescue mode

→ To troubleshoot this problem provide the root password for the maintainence

→ When we provide the root password correctly then a shell prompt will be opened

→ At this movement the file System is mounted in read only mode hence we can't change the file system table (fstab file)

→ So mount the file system in rw mode

```
#mount -o remount,rw / ↵
```

→ Now open the file /etc/fstab

```
#vim /etc/fstab ↵
```

Now remove (or) change the file systems which leads to inconsistency and save the fstab file and reboot the pc

```
#init 6 ↵
(or)
#reboot ↵
```

Assign the grub password:

The grub password is to be maintained in the file "/boot/grub/grub.conf".

#grub-md5-crypt >> /boot/grub/grub.conf

when we type the above command then a encrypted password comes to the last line of the file /boot/grub/grub.conf

we have to copy that encrypted password and paste in the same file under the line "hidden menu"

password --md5 <encryptedpasswd>

here we have to paste the encryptedpassword

Recover root & grub password:

Step①: use the Red Hat DVD, boot from it, and choose the option to enter Rescue mode.

boot: Linux rescue ↵

Select keyboard. => select language => select networking

Then a shell will be opened.

at this moment to give the previlage to the root user to maintain the file system

#chroot /mnt/sysimage ↵

Here we can follow the above mentioned procedures of recovering root password and grub password.

The Superblock Has Become corrupt:

If the superblock on your system has become corrupt, you can re-create it with one of the backup superblocks. If the primary file system has the corruption, you may need to use single-user mode or the rescue environment to perform the recovery.

Step ①: check the state of the file system:

```
#dumpe2fs  h  /dev/sda1 ↵
```

Step ②: find a valid backup superblock:

```
#dumpe2fs /dev/sda1 | grep -i superblock ↵
```

Step ③: Repair the file system with a backup superblock:

```
#e2fsck -f -b 8193 /dev/sda1 ↵
```

Backup & Restore

→ In information technology, a backup or the process of backing up is making copies of data which may be used to restore the original after a data loss event.

→ Backup have two distinct purposes.

→ The primary purpose is to recover data after its loss, be it by data deletion or corruption. Data loss is a very common experience of computer users. 67% of internet users have suffered serious data loss.

→ The secondary purpose of backups is to recover data from an earlier time, according to a user-defined data retention policy, typically configured within a backup application for how long copies of data are required.

→ Backup is the most important job of a System administrator, as a System admin it is your duty to take backup of the data everyday.

→ Many companies have gone out of the market because of poor backup planning.

→ The easiest way to backup your first is just copying. But if you have too many files to backup, copying and restoring may take too long time and it is not convenient. If there is a tool that can put many files into one file, the world will be better. Fortunately, 'tar' is used to create archive files.

Compression and Archiving:

→ As you will learn when you become a System Administrator, backups are the number one priority.

→ If something should crash or become corrupt and you can't restore it because you aren't keeping up with your backups or you just don't keep any, you may be looking for a new job. Although we don't address backup programs here, this is good lead into archiving and compression.

tar: Tar means tape archiving

→ It is used for compressing and archiving files and directories.

→ A more common use for tar is to simply combine a few files into a single file, for easy storage distribution.

Syn: tar [options] [FILE]

Options:
- -c Creates a new archive
- -v provides verbose output
- -f Specifies the archive file to use
- -t Lists the files in an archive
- -x Extract the backup
- -z Zipping

Step ①: create some random blank files:

#touch file1 file2 file3 another_file ↵

Step ②: create a simple archive containing these files:

#tar -cvf Sample.tar file1 file2 file3 another_file ↵

when an archive is created, you can also apply compression to reduce the amount of space the archive files takes up. Although multiple types of compression are supported with the use of tar, we look only at gunzip (.gz) and bzip2 (bz2) here.

Step ③: Let's re-create the archive using the gunzip compression:

tar -cvzf Sample.tar.gz file1 file2 file3 another_file↵

Step ④: view the current directory to see the two current archive files:

#ls ↵

Step ⑤: To see all the contents within the build file:

#tar -tvf Sample.tar ↵

Step ⑥: Now extract this build file verbosely:

#tar -xvf Sample.tar ↵

Step ⑦: To extract files on different location

#tar -xvf Sample.tar -C /root/Desktop ↵

cpio : cpio is a tool for creating and extracting archives or copying files from one place to another.

→ It handles a number of cpio formats as well as reading and writing tar files.

→ cpio like tar but can read input from the "find" command.

→ The basic structure is

: find -name file | cpio [options] [controller] <Dest>

options: -o (out)
-i (in)

controllers:
O (or) > -out
I (or) < -in

Step ①: To take the backup files

#ls file* | cpio -ocvf > /root/backup.cpio ↵

Step ②: To see the backup content:

#cpio -it < /root/backup.cpio ↵

#cpio -it -I /root/backup.cpio ↵

Step ③: To restore the backup file:

#cpio -icuvd < /root/backup.cpio ↵

o → Reads the standard ~~output~~ input
i → Extract files from the standard input
c → Read or write header information in ASCII character
d → Creates directories as needed
u → copy unconditionally (older file will not replace a new file)

DD : (Disk to Disk)

Used to take the backup of one partition to another, here source partition should be given to "if", destination partition should be passed to "of".

Step ① : To take the backup :

```
#dd if=/dev/hda6  of=/dev/hda7 ↵
```

Step ② : To recovery :

```
#dd if=/dev/hda7  of=/dev/hda6 ↵
```

Scp : (Secure copy)

→ Scp is used to copy data from one unix or linux System to another unix or Linux Server.

→ Scp uses Secured Shell (ssh) to transfer the data between the remote hosts.

→ The features of scp are:

- copies files with in the same machine.
- copies files from local machine to remote machine
- copies files from remote machine to local machine
- copies files between two different remote servers.

Syn: scp [options] [user from_Host: source_file] [user to_Host :Destination_file]

Options:
- -r Recursively
- -q progress bar not displayed
- -v verbose mode
- -p copy files using the specified port number.

→ Copy file from local host to remote server:

scp filename root@server254.Example.com:/root ↵

→ Copy files from remote host to local server:

scp root@server254.Example.com:/root/backup* . ↵
(. ↓ current directory)

→ Copying a directory:

scp -r directory root@server254.Example.com:/root ↵

→ Improving performance of scp command:

Using blowfish or arcfour encryption will improve the performance of the scp command

scp -c blowfish filename root@server254.Example.com:. ↵

→ Specifying the port number:

scp -p 6001 backup_file root@server254.Example.com:/tmp ↵

—x—