RS=100/=

UNIX/LINUX

→ UNIX is a CUI operating system.

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

→ Linux is not just for unix wizards. Linux is a clone of O/S.

⟹ UNIX History:

The journey of unix operating system had been started from the project MULTICS (Multiplex Information and Computing System) at AT&T (American Telephone & Telegraph) Bell labs. The software team lead by ken Thomson, Dennis Ritchie & Rudd Canby worked on this project. The aim of this project is to share the same data by 'n' number of users at the same time. Initially, MULTICS was developed for only two users. Based on the same concept in 1969, UNICS operating system was developed for 100's of users. UNICS stands for (Uniplexed Information Computing System). Initially UNICS was written Assembly language. In 1973, they rewritten in 'c' language named as UNIX

MULTICS — In 1965
↓
UNICS — 1969
↓
UNIX — 1973

⟹ Operating System (O/S):

→ It is a System software

→ It is a collection of System programs

→ Operating System is a interface between User & Computer

→ It is classified into two types.

(1) CUI (Character User Interface) : DOS, UNIX

(2) GUI (Graphical User Interface) : WINDOWS, LINUX

→ GUI is a user friendly, CUI is a not user friendly.

1

⇒ Features of UNIX:
- Multiuser
- Multitasking
- open system
- progaramming facility
- Security
- portability
- Communication
- Help facility

Multiuser: More than one user can access same system resources (cpu, applications, memory, pointers ... etc) at the same time known as multiuser.

Multitasking: Execution of more than one task (or) application simultaneously known as multitasking. The main concept of multitasking is maximum utilizing cpu resources.

Open System: UNIX is a open source code. i.e any user can modified Unix open source code according there ideas & requirments.

→ It is developed "AT&T" Bell labs Employess in 1973 by using 'C' language

→ Using UNIX open source code

Redhat + Adding additional features ⇒ LINUX
SUN microsystem + " ⇒ Sun Solaries
IBM + " ⇒ IBM-AIX
HP + " ⇒ HP-UX
Santa cruz + " ⇒ SCO-UNIX
Silicon graphics + " ⇒ IRIX
Microsoft + " ⇒ Xenix

→ Any operating system developed based on UNIX open source code known as flavour of UNIX.

Advantages of Linux:
- Free Software
- GUI
- New hardware
- C, C++, perl, PHP, python, my SQL ..... (by default availa
- open system.

2

→ Linux is also open System

→ Flavours of LINUX :
- ~~Redhat~~ Fedora
- Suse
- Obuntu
- debain
- Bhart o/s solutions
- Oracle Enterprice Linux.
- Linux mandrake
- puppy linux
- turbo linux
- cent o/s
- Slackware linux

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	- Compiler based language
2. Interpreter converts high level instructions into machine language linebyline.	- The whole programe in a single shot into machine language
3. It doesn't create .exe files.	- create .exe files.
4. No need to compile the program	- Need to compile the program
5. It takes less lines of code	- Takes numerous lines of code
6. Reducess cost of maintainance	- Increses cost of maintainance

Security : UNIX has given two levels of security
(1) System level security
(2) File level security

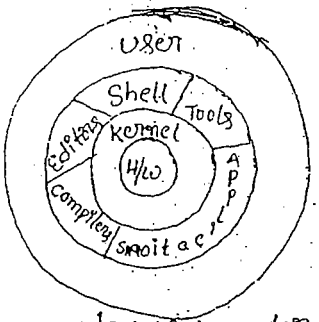
(1) System level security : System level security controlled by System Administrate
(2) File level security : File level security controlled by owner of the file.

portability : portability means independent of hardware & processors.

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

Help facility : Man is the command to see the help.
#man perl ↵

⇒ Structure of UNIX:



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 understandble language (machine language) and it send the given request to kernel

→ The shell access interface b/w user and kernel

kernel : kernel is nothing but the bundle of software that makes an o/s for making coordination possible b/w user and hardware.

→ It is a group of hundreds of system calls

→ System call means low level programming

→ Each & every system call written in 'c' language

Eg: open(), fork(), copy(), proc(), kill(), create ------ etc

→ Jobs of kernel: - File Management
- Memory Management
- process Management
- Device management

⇒ D/B UNIX & WINDOWS:

	UNIX	WINDOWS
9.	It is CUI	- GUI
1.	UNIX is multi-user o/s	- Windows is a multiuser - o/s
2.	UNIX is multitasking o/s	- windows is also multitasking o/s
3.	To boot the UNIX o/s, 2mB RAM is Enough	- To boot the UNIX o/s 12mB RAM is required.
4.	UNIX is process based concept	- thread based concept
5.	In UNIX, any user process is killed It will not effect the other	- It effects to all.
6.	Can run more than 1,00,000 transactions per minute	— 80,000 per minute
7.	There is no limit for number of users working with the server	- Limited users.
8.	UNIX is an open system	— closed system

⇒ UNIX File System: File System is a mechanism used in the C/S Environment for storing the data in a systematical order into a storage device.

(or)

→ It is the way of storing and retrieving the files and directorys into the secondary storage devices.

→ Types of file systems are: - Disk based
- Network based
- Virtual based

Disk based 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:	WINDOWS	UNIX
	FAT	ext2
	FAT32	ext3
	NTFS	HFS+
		UDF

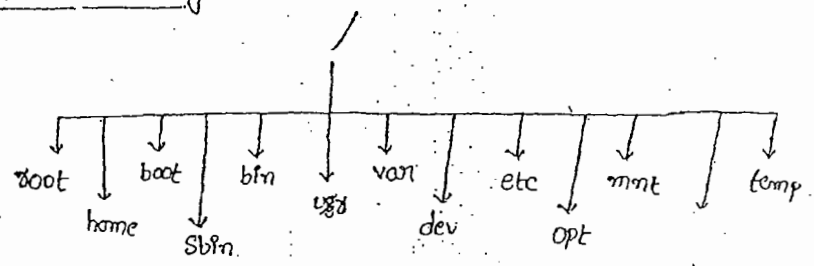
Network based File Systems: For sharing any resource from one PC into a network we require network base file system.

Ex: NFS, DFS, SMB protocols, FTP.

Virtual based File Systems: For improving the system performance we can go for virtual based file system.

Ex: Swap, RAMFS

→ Filesystem Hierarchy:



/ : Slash is a parent directory from where the Entire file System is going to start. It can also be called as Top level working directory.

/root : This is the default workspace allocated for the privilage user.

Ex: Super user :.

* /mnt → Mount.

5

/home : This is the default work space allocated for non-privilage user

Ex: normal user :

*raju ⟶ /home/raju.

/boot : All the bootable files, boot loader information and kernel information can be formed under /boot directory.

Ex: /boot/grub/grub.conf

/bin : It maintains all user executable commands.

/sbin : It maintains all Administrator executable commands.

Ex: fdisk
useradd

/usr : It means unix System resources. It contains the programs and applications which are available for both normal user & administrator.

Ex: man pages

/var : The variable data that keeps on changing like log files, mail accounts, and printspool can be found under /var.

/dev : The entire device which are connected to the machine are store under /dev directory. Every device in linux and unix is treated as a file

Ex: /dev/sda

/opt : In this directory we can install only the third party applications

like - openoffice
- webbin
- oracle

/etc : It contains all configuration files.

Ex: /etc/exports
/etc/hosts

/mnt : It is mount point provided for removable media. Such as Floppy Drive, CD, DVD, Ram disk etc.

/tmp : This directory contains temporary files used by the System.

/lib : It consists library files. which are needed by no. of different applications as well as linux kernel.

6

* Basic commands : Commands are nothing but the program or the Script or the Executable file used for assigning a task to the

logname → current username
pwd → present working directory
uname → Display o/s name
uname -r → Display kernal version
date → Display systemdate & time
clear → To clean the screen
cal → Display current month calender
cal 2011 → Display 2011 cal
cal 2 2011 → Display Feb 2011 cal
hostname → Display Server name
hostname -i → Display Server ip address
tty → Display terminal name
who → To display how many users connected to the Server
whoami → current username (or) parrent user name
who am i → Display switched username (or) child username
finger username → display given user information

Ex: # finger root ↵

uptime → Display how long Server is "up & running", no. of users connected & avg load on the Server
Su → It is used for switch prompt
which <command name>
(or)
whereis <command name> } → Displays location of the command
Exit → To Exit the terminal (or) session

7

⟹ Viewing List of files & directorys :

Syn: # ls <options> < source>

options: -a → To check the hidden object
-i → To view inodes
-l → To list the properties of an object
-s → To check the size in block
-r → To check reverse order
-R → Recursively

Ex: # ls -a
ls -la
ls -lr

⟹ How to identify Files & Directorys :

Files	colour
Files	Black
Directorys	Blue
Link	Sky blue
exe	green
zip	Red

⟹ Working with Files :

cat :- It is used for to create new files (or) to open existing files (or) to append data to the existing files

→ create a file

#cat > wisdom

---
ctrl+d (Save & quit)

→ open the file

#cat < filename
(or)
#cat <file name>

8

→ To append the file

#cat >> file name

→ To open multible files

#cat f1 f2 f3 ↵

Dis adv: Using this command we can't create multible files and we can't Edit the text of the file.

Touch: It is used for create Empty files and also create multible files. (i.e zero byte file)

→ To create multiblefiles

#touch file1 file2 file3 ....

Dis adv: Using this command we can't modify the text of the data base file.

→ Deleting the files:

#rm <file name>

#rm -i <file name> → asking confirmation

#rm -f <file name> → Delete file forcibility

⇒ Working With Directorys:

→ To create a directory

#mkdir <dir name>

→ creating multible directorys

#mkdir dir1 dir2 dir3

→ creating Nested tree structure

#mkdir -p world/asia/india/ap/hyd/wisdom

→ How to check

9

→ Removing directorys

#rmdir <dir name> [But directory must be Empty]

#rm -r <dir name> → Deletes recursively Entire directory.

#rm -ri <dir name> → asking confirmation msg

#rm -rf <dir name> → remove forcefully

⇒ Navigation commands:

#cd → To change a directory

#cd .. → To comeout from current working directory
(or)
To move one level back

#cd ../.. → To move two levels back

#cd ~ → To move to the users home directory

#cd <dir name> → To move to a particular directory.

⇒ Copying files & directorys:

cp: → To copy a file

#cp file1 file2
↳ may be Existing (or) not

→ To copy a file with confirmation

#cp -i file1 file2

→ copying multible files into a directory

#cp f1 f2 f3 .... fn dir name

→ copying a directory

#cp -R <Source dir> <dest dir>

mv: To rename a files (or) directorys
(or)
To move a file (or) directory

#mv file1 file2 ↵

#mv dir1 dir2 ↵

10

ls -l → Display longlist format i.e with 9 fields.

(i) File type (-) → Regular file
(ii) File permissions (d) → Directory file
(iii) No. of Links (l) → Link file
(iv) Owner file
(v) group name
(vi) File size in bytes
(vii) Date
(viii) Time
(ix) File name

b → block special file
c → character special file
} device files

⇒ Wild card characters (or) Meta characters:-

→ Also called global characters.

(1) * → It matches zero (or) more characters
(2) ? → It matches any single character
(3) [ ] → It matches any single character in the given list
(4) [-] → It matches any single character in the given range.

Ex: # ls t* ↵ Starting with file name 't'
ls ra* ↵ Starting with 'ra'
ls b*k ↵ Starting with b & Ending with k.
ls a? ↵ (one ? mark is one character)
ls a?? ↵
ls [bkat]* ↵ Starting with b, k, a, t
ls [bkat] ↵ file names b, k, a, t
ls [a-h]* ↵ Starting with 'a-h'
ls [1-9]* ↵ file name starting with digits [1-9]

11

→ Creating hidden files :- To hide a file start file name with . character

#cat >.filename↵

#mv Empl .Empl ↵ To hide existing file

#mv .Empl Empl ↵ To unhide a file

#mkdir .abc ↵ To hide a directory

#mv .abc abc ↵ To unhide a directory.

→ Viewing List of files :

ls → displays current directory all files & sub directorys in the assending order based on ASCII values.

#ls -a → displays all files & sub directorys along with hidden files

#ls -r → display in reverse order

#ls -R → Recursively

#ls -t → based on time & date of creation (which file most recently created)

#ls -F → along with file type (how to identify list of files & directorys)

→ How to identify files & directorys :

#ls -F ↵

Files – Black

Directorys – Blue

Link – Sky blue

exe – green

zip – Red.

12

* SHELL CONCEPT:

→ 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 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
1. Bourne shell	Stephen Bourne	$	sh
2. Korn shell	David Korn	$	ksh
3. cshell	Bill Joy	%	csh
4. Bash shell	Stephen Bourne	$	bash
5. Zshell	paul	$	zsh

Note: The advanced version of Bournshell is Bash shell.

→ Bash means "Bourne Again shell".

Default Shell Name	Flavour Name
1. Bash shell	Linux.
2. Bourne shell	Sco-unix, solaries, Hp-ux
3. Korn shell	IBM-AIX
4. cshell	IRIX (Silicon Graphics)

→ Shell Features:

- Word completion
- Command alias
- Command History
- Shell Scripting

→ To check the shells List:

#cat /etc/shells ↵

→ To check the parent shell of current user:

#echo $SHELL ↵

→ What is the shell location:

/bin

→ To view the available shells:

#cd /bin

#ls *sh ↵

→ how to shift from bash shell to sh shell:

#sh ↵

→ how to shift from Bourne shell to korn shell:

#ksh ↵

→ To check current child shell (or) sub-shell:

#echo $0

→ To check the history: #history ↵

→ To Lock the history: #history -c ↵

→ To unlock the history: #history -r ↵

→ To remove the history: #rm -rf .bash_history ↵

→ Applying the alias name:

#alias u=useradd ↵

→ To display the alias List: #alias ↵

→ To make alias permanent: #vi .bashrc ↵

Ex: u wisdom
↳ useradd

→ To remove alias temporarily: #unalias u ↵

—x—

14

⇒ Filter commands:-

WC: It counts no. of words, lines, characters in a given file

#cat filename ↵

Syn: #wc filename ↵

7 35 196 filename

↳ Line no. → words characters → this is file name

#wc -l Emp ↵ Count lines only

#wc -w Emp ↵ Counts words only

#wc -c Emp ↵ Count characters only.

#wc -lw Emp ↵ Count lines, words.

diff: Displays different lines b/w 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 no. & character position.

File: It displays the given file type

Syn: #file filename ↵

#file t1 ↵

t: Empty

#file t2 ↵

t2: ASCII text

#file Sample ↵

cut: It is used for to retrieve specific fields & characters in a given file

#cut -f 2,4 Stud↵ It displays second & fourth field from Stud file

#cut -f 2-5 Stud↵ displays 2-5 fields from Stud file

#cut -d",," -f 2,4 Stud↵ displays 2nd & 4th fields from stud

#cut -c 1-5 Stud↵ displays every line 1st to 5th character
↳ character

#cut -c 1-5, 15-20 Stud↵ 1-5 & 15-20 characters displaying.

paste: It is used for join two or more files horizontally by using delimiter

#paste states cities ↵

Ap	Hyd
TN	Che
KN	Ben
MR	mum

#paste -d":" states cities↵

Ap : Hyd
TN : che
KN : Ben
MR : mum

tr: It translate character by character

#tr "aeiou" "AEIOU" < Sample↵

Sample
i am learning unix

#tr "," "\t" < Emp↵ In Emp file all "," replaced with tab space

#tr -d "a" < Sample↵ delete "a" from Sample file.

** #tr "a-z" "A-Z" < Sample↵ To convert small keys converted into upper keys.

#tr -d "aeiou" < Sample↵

#tr -s " " < Sample↵
↳ squeeze

Sample
this is sample file

16

Sort: It is used for to sort the file contains by default sort the file contains based on ASCII values

→ The default order is Assending order

```
#sort sample↵
```

Sample

te-
a-
he-
s-
ha-

ASCII values

0-9 ⇒ 48-57
A-Z ⇒ 65-90
a-z ⇒ 97-122

→ reverse order

```
#sort -r sample↵
```

→ To display unique lines (↳ eliminate duplicate lines.)

```
#sort -u sample↵
```

```
#sort -n file1↵ To display numaric
```

file1

456
125
23
1025
456
15
225

```
#sort file1↵
```

→ To sort field by field:

```
#sort -f +pos1 -pos2 filename
```

(+pos1 ↳ including; -pos2 ↳ Excluding)

```
#sort -f +1 -2 stud↵ complete data assending order of name
#sort -f +3 -4 stud↵
#sort -f +1 -3 stud↵
#sort -fn +2 -3 stud↵
#sort -t "," -f +1 -2 stud↵
```

Uniq: It displays uniquelines in the given files but the file contains must be in sorted order.

```
#uniq sample↵
```

aaa
ccc
ddd
hhh
ppp

Sample

aaa
aaa
ccc
ddd
hhh
hhh
hhh
ppp

```
#uniq -u sample↵ display non duplicated lines.
```

ccc
ddd
ppp
LLL

17

⇒ #uniq -c sample ↵ counts how many times the lines repeated in the file

#uniq -d sample ↵ displays only duplicated lines.

⇒ To delete duplicated records (or) lines in the given file

#uniq -u sample > temp ↵

#mv temp sample ↵

head: It displays the first n lines from sample files

→ By default displays the first 10 lines from sample file.

#head -5 sample ↵

head sample ↵

tail: It displays the last n lines from file

→ By default displays the last 10 lines

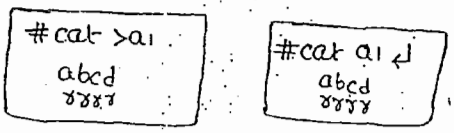
#tail sample ↵

#tail -5 sample ↵

#tail -f filename ↵ used for to monitor file growth

open two terminals (or) two user terminals

#tail -f sample ↵



quit: ctrl+c

tee: It is used for to write data to the file as well as to the screen.

#cat Emp | tee Emp1

→ To copy single file data to multible files

#cat Emp | tee file1 file2 file3

18

⇒ piping : It is used for combined two (or) more Cmds.

→ we can give the piping (or) connection b/w two Cmds.

→ Here first command output is taken as the second Cmd input.

who | wc -l ↵ count total no of users connected to the server

ls | wc -l ↵ count total no of files in the current directory.

→ To count total no of subdirectorys in the current directory

#ls -l | grep "^" | wc -l ↵

date | cut -c 1-3 ↵ displays todays date

ls -l | tr -s " " | cut -d" " -f 3,5,9 ↵ Display current directory all files owner name file size & file name.

Sed (Stream Editor) : It is used for to search & replace strings (or) patterns and it is multipurpose filter command.

Syn: #sed "s/oldstring/newstring/g" filename ↵

s → substitution, g → globally & all occurence in every line

#sed "s/unix/Linux/g" sample ↵

#sed "s/unix/Linux" sample ↵

cat > sample

unix

-unix-

unix-unix-

-unix

* #sed "s/unix/Linux/gi" sample ↵ → ignore case sensitive

* #sed "s/^unix/Linux/gi" sample ↵

* #sed "s/unix//gi" sample ↵ To delete a word from a file

* #sed -n '3p' stud ↵ displays 3rd record from stud file

#sed -n '3,5p' stud ↵ print 3rd record to 5th record

#sed -n '$p' stud ↵ prints last record.

* #sed -n '1p
> $p' stud ↵ print 1st record & last record

#sed '3d' stud ↵ delete 3rd record from stud file.

#sed '2,5d' stud ↵ delete 2nd to 5th record.

#sed '2,5 w f1' stud ↵ It copyes 2nd record to 5th record from stud file to f1 file.

Note: sed command deles with rows
cut command deles with columns.

#sed '=' sample ↵ displays Emp file record with line numbers
↳ to get line numbers.

⟹ compress/uncompress the files:

zip :→ It is used for to zip the file ~~gzip file~~.

Syn: #gzip <file name>

#gzip sample ↵

sample.gz.

gunzip :→ To unzip the file

#gunzip sample.gz ↵

sample

compress: To compress the file

#compress filename ↵

sample.z

uncompress: To uncompress the file

#uncompress sample.z ↵

sample

more: To see the contents of a file in the form of pagewise

#more <file name>

#more f1 ↵

→ But it will work only in forward direction

#more /etc/passwd

less: To display file contents in pagewise

→ we can go to all the directions

#less /etc/passwd ↵

f – Move to the one page forward direction
b – Move to the one page back
h – display the help options
v – to open the vi Editor
q – quit

aspellcheck: It is used to check the spelling mistakes but not gramitical mistakes.

#aspellcheck <filename>

#aspell check t1 ↵

* grep: [Globally research a regular Expression & print]

→ It is used for to search a string (or) regular expression in a given file (or) files.

Syn: #grep pattern filename(s)

#grep Raju sample ↵

#grep Raju f1 f2 f3 ↵

#grep Raju * ↵ search all files in a current directory

#grep " Raju Rongali " sample ↵

21

Options:

#grep -i Raju Sample ↵ To ignore case sensitive

#grep -c Raju Sample ↵ counts no. of lines patterns in give file

#grep -n Raju Sample ↵ prints the lines along with line numbers containing the given pattern.

#grep -l Raju * ↵ gives file names only the given pattern

** #grep -r Raju * ↵ Searches the given pattern recursivily Entire dir---

#grep -v "Raju Ron" Sample ↵ print the lines do not containing the given pattern.

#grep -o Raju Sample ↵ prints only the given pattern.

#grep --color ↵ with higlights of pattern with color.

#grep Raju sample --color ↵

Regular Expression: Any String contains wildcard characters known as regular Expression (or) pattern.

→ The patterns are classified in to three types.

1. character pattern
2. Word pattern
3. Line pattern.

1. character pattern: The default pattern is character pattern.

#grep "Raju*" Sample ↵

#grep "b[ae iou]ll" Sample ↵

ball
bell
bill
boll
bull

#grep "b..d" Sample ↵

band book ba#d ba_d

Note: '.' is a wild card character it matches any single character.

22

2. Word pattern :

\< \> ⟹ word boundary

\< ⟹ Starting of the word

\> ⟹ Ending of the word

#grep "\< Raju \>" sample ↵

(or)

#grep -w "Raju" sample ↵ Raju ✓ Raju123 x Raju12 x

#grep "\< Raju" sample ↵ Raju ✓ Raju123 ✓

#grep "Raju\>" sample ↵ TRaju123 x

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

3. Line pattern :

Anchors :

^ ⟶ Start of the line

$ ⟹ End of the line

#grep "^d" sample ↵ line start with d

#grep "^the" sample ↵ line start with the

#grep "^\<the\>" sample ↵ there - - x the - - ✓ d - - x

#grep "^[grep]" sample ↵ g - ✓ e ✓ t - x p ✓ n - ✓

#grep "^[^sdr]" sample ↵ not start with s -- d -- r --

\^ \. * \[

\$ \] \[

No meaning Special Character

* #grep "[0-9]$" sample ↵ Line Ending with digit

#grep "[^a-z A-Z 0-9]$" sample ↵

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

** #grep "^...$" sample ↵ line should contain three characters

#grep -c "^$" sample ↵ display no. of Empty lines

#grep "^$" sample ↵ blank lines

23

fgrep : (Faster grep) : It is used to search multible strings but not regular Expression.

→ It searches the string more faster than the grep command.

#fgrep " unix
> perl
>oracle " stud ↵

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

(i) | (pipe) : It matches any one string in the given list

#egrep " (unix|oracle|perl) " stud ↵

(ii) {m} : It matches exact occurence of it preceding character

#egrep ab{3}c sample ↵

#egrep " \< [0-9]{4} \> " sample ↵

(or)

#egrep " \< [0-9][0-9][0-9][0-9] \> " sample ↵

(iii) {m,n} : It matches minimum 'm' occurencess of maximum 'n' occurences of its preceding character

#egrep " \< [0-9]{4,7} \> " sample ↵

(iv) {m,} : It matches minimum 'm' occurencess of its preceding characters

ab{3}c ⇒ abc, abbbc abbbbbc

#egrep " \< [0-9]{4} \> " sample ↵

Note : Fgrep means ⇒ grep -F

24

* find: this filter is used to search the results by depending on requirements may be on name, inode no, user, permissions etc.....

Syn: #find <location> <requirement> <value>

.....(or)

#find <searchpath> <criteria> <action>

→ To find the file with name Raju in /

#find / -name Raju

→ To find the file with name Raju in /root directory

#find /root -name Raju

→ To find the file test1

#find / -name test1 ↵

→ To find the file starting with pass

#find / -name 'pass*' |more

Based on size:

#find -size 4c (here '4c' means 4 characters)

#find -size +4c (here '+4c' means more than 4 characters)

#find -size -4c (here '-4c' means less than 4 characters)

#find / -size +50m ↵ more than 50m.

#find / -size -50m ↵ less than 50m.

#find / -size +50m -size -100m (more than 50, less than 100m)

#find / -name test1 -size +10m

#find / -name test1 -o -size +10m

↳ or operation.

#find / -name test1 -not -size +10m ↵

↳ not operation

#find / -name test1 -not -size -10m ↵

Based on time:

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

→ To find the file with modification time

#find -mtime 100 (files exactly modified on 100th day back)

#find -mtime +100

#find -mtime -100

#find / -mtime -10 /more

10 → Exactly 10th day
+10 → 10 days back
-10 → Exactly last 10 days

#find / -mtime -10 /more

#find / -mtime +10 /more

→ To find the file with access time:

#find . -atime 100

#find . -atime +100

#find . -atime -100

Based on permissions:

#find / -perm 777 ↵ To find the file with full permissions.

#find / -perm 755 ↵ To find the file with 755 permission.

Based on inode no's:

#find . -inum 551074 (here '.' means current directory)

→ To find the files only

#find . -type f ↵

→ To find the directory

#find . -type d ↵

→ To find the particular users file only

#find / -user <username>

→ To find the particular groups files only

#find . -group <group name>

26

⇒ Vi Editor (visual interface):

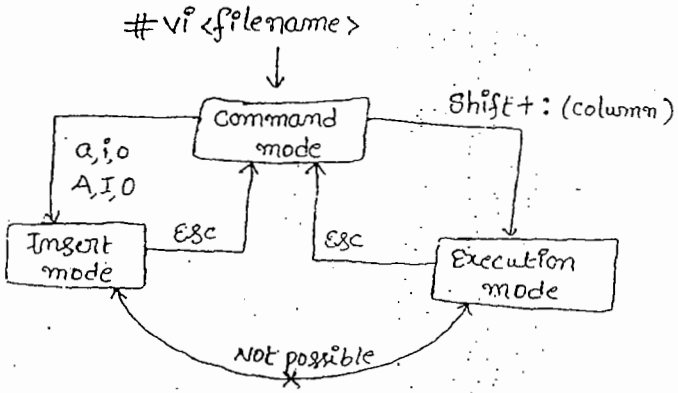
→ It is used for to create new files (or) to modify existing files.

→ In vi Editor totally 3 modes are available.

1. Command mode
2. Insert mode
3. Execution mode

→ vi is command to open vi Editor

→ The default mode is command mode.



→ To shift from command mode to Insert mode:-

X — deletes one character

i → It places cursor at left side of the cursor line

I → It places cursor at beginning of the current line

a → It places cursor at right side of the cursor line.

A → It places cursor at end of the current line

o → It inserts new line below of the cursor.

O → It ~~places~~ inserts new line above of the cursor.

→ Command mode commands:-

H → Beginning of the current page

M → Middle of the current page

L → End of the current page

27

$ → End of the current line (End key)

^ → Beginning of the current line (home key)

x (nx) → Delete current character (del key)

ctrl f → foreward one page (pgdn)

ctrl b → backward one (pgup).

dd (ndd) → Delete the current line (delete n lines)

YY (nYY) → copy the Entire line (copy n lines)

P → paste below the cursor

P → paste above the cursor

U → undo

ZZ → Save & quit

Y$ → It copies current position to End of the line

Y^ → It copies current position to beginning of the line.

gg → Move to beginning of the line

G → Move to End of the line

99G → goto 99th line

/<find word> → Find the particular word.

h → move to left k → move to up

l → move to right j → move to down.

→ Execute Mode options:-

:w → Save without quit

:w filename → save with given file name

:q → quit without saving changes.

:q! → quit with forcefully

:wq → Save & quit

:wq! → Save & Exit forcefully.

:x → Save & Exit

:set nu → Set line numbers

:set nonu → To remove line numbers

:nd → deletes nth line

:$d → deletes last line

:4,7d → deletes 4th line to 7th line

:! <unix cmd> → Execute unix commands

:/String/ → Top to bottom Search

:?String? → Bottom to top Search.

→ How to Manage the Terminal :-

Alt + F10 → Maximize
Alt + F5 → Restore
Alt + F9 → Minimize
Alt + Tab → open terminal
F11 → Enable & Disable full screen
Ctrl + Shift + + → Increase the zoom
Ctrl + - → Decrease the zoom
Ctrl + 0 → Normal size
Ctrl + Shift + t → opens the new tab at some terminal
Ctrl + Swift + pageup → move to the previous tab
Ctrl + Swift + pagedown → Move to the after tab
Ctrl + Shift + n → open the new terminal
Exit → close the terminal

→ Command Line Editing :-

Ctrl + a → move to the beginning of the line
Ctrl + e → Move to the End of the Line
Ctrl + b → move to the one char back
Ctrl + f → Move to one char forward
Ctrl + w → Delete the before line current cursor position
Ctrl + k → Delete the after current cursor position line
Ctrl + c → skip from command
Ctrl + m → Execute the command without pressing enter
Ctrl + p → Display the previous command
Ctrl + n → Display the next command

→ To see the default shell #echo $SHELL
→ Super user prompt is (#)

29

⇒ process management:-

→ To check the process #PS (or) PS -f

→ To see the all process information #ps -aux

→ By default Jobs come under foreground

$cp file1 file2 ↵ foreground Job

$cp file1 file2 & ↵ Background Job

$sort sample > a1 & ↵ Background Job

513 (PID)

Note: In foreground, user can execute only one Job but in the background user can execute many Jobs.

→ To kill foreground Job

ctrl c (or) ctrl z

→ #ps -ef ↵ It displays current running process list in the Linux Server

→ To kill background Job

kill pid ↵

kill -9 pid ↵

→ signal

→ To see the signals #kill -l ↵

→ nohup : the nohup Jobs will create in server account so nohup Jobs will execute even the user disconnects from his account

Eg: $ nohup cp file1 file2 & ↵

→ To display only background Jobs with Job ids

$Jobs ↵

→ How to bring background Job to foreground

$fg Jobid ↵

→ To check the cpu status in CUI mode # top

GUI mode # gnome-system-monitor

⇒ Redirecting Input/Output/Errors :-

STDIN – File Description (FD) – 0
STDOUT – (FD) – 1
STDERR – (FD) – 2

→ < ⇒ Redirecting input

→ > (or) /> ⇒ Redirecting output

→ 2> ⇒ Redirecting Errors

Eg: #cat Emp > File1 ↵

#cat >test6 <test2 ↵

#cat test6 ↵

#cat >test1 <test2 (Input from test2)

STDERR:-

#cat t1 t2 t3 2> Error ↵

#cat Error ↵

#cat t1 t2 t3 2>> Error ↵

#cta Sample 2> file1

#cat file1

#cat Sample a3 Sample2 >out-file 2> Error file ↵

#cat out-file ↵

#cat err-file ↵

⇒ Flat file:-

→ In a file data entered by using delimiter know as a flat file.

→ The default delimiter is tab key

→ Delimiter means field seperator (like : ; , * - |)

→ Enterkey means record seperator.

31

#cat > Stud ↵ (Default delimiter flat file)

Sid	SName	phone	course	location
101	Raju	9848363431	Linux	Vij
102	Mahi	9965894342	Ora	Sec
103	Rao	9923125253	Unix	Hyd
104	Raja	9290916905	Java	Sec

#cat > Studi (Custom delimiter flat file)

Sid, SName, phone, course, location
101, Raju, 9848363431, Linux, Vij
102, Mahi, 9965894342, Ora, Sec
103, Rao, 9923125253, Unix, Hyd
104, Raja, 9290916905, Java, Sec

⇒ Links :- To give a pointer to the source file, called a link.

→ In linux two types of links

(i) Soft Link : → Inode no. of the source file, link file are different

→ It can be created across partition

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

→ Size of soft link file equals to no. of chars in original file path

→ If original file is deleted the link file will not be accessible

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

(ii) Hard link : → Source file & link file have same inode numbers.

→ It can't be created across partitions

→ Editing of original file will replicate in the link files

→ Size of hardlink file is same as original file

→ If the original file is deleted the link file we can access.

Syn: #ln <original file> <link file>

32

* path: → It is the way of representing files & directories in the System.
→ Two types of paths.

(a) Absolute path: It is the way of representing files & directories from the top of hierarchy is called Absolute path.

Ex: #cp /root/Emp /world/Asia/India/Ap/Hyd↵

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

#cd /root/Desktop↵
Ex: #cp wisdom /var/wisdom1

⇒ 1 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 referring.

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 Megabyte	1000 kilo = 1 mega
1024 mega = 1 Gigabyte	1000 mega = 1 Giga
1024 Giga = 1 Terabyte	1000 giga = 1 Tera
1024 Tera = 1 petabyte	1000 Tera = 1 peta
1024 peta = 1 Exabyte	1000 peta = 1 Exa
1024 Exa = 1 zettabyte	1000 Exa = 1 zetta
1024 zetta = 1 yottabyte	1000 zetta = 1 yotta
1024 yotta = 1 Brontobyte	1000 yotta = 1 Bronto
1024 Bronto = 1 Geobyte	1000 Bronto = 1 Geo

33

...munication Commands:

(a) **write:** → It is used for to write message to another user account, but he should be logged into the server

```
$write username/terminalname ↵
.....
Ctrl+d (save & quit)
```

→ To deny messages: $mesg n ↵

→ To allow messages: $mesg y ↵

(b) **wall:** → It is used for to send broadcast message to all users, whoever connected to server and mesg is 'y'.

```
$wall ↵
    welcome to wisdom
ctrl+d (save & quit)
```

(c) **Mail:** → Using mail 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 ↵ — to send mail one user

$mail user1 user2 user3 ↵ — multible users at a time

$mail user1 < stud ↵ It translates stud file to user1.

→ mails are stored in mail box: (/var/spool/mail/user1)

→ mail is the command 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 transffered to Secondary mail box (i.e mbox)

→ To open Secondary mailbox:

$mail -f ↵

→ The primary mailbox only maintains unread mails.

· Mail box options:

& q → quit

& r → reply

& p → print

& d → delete current mail

& d 2 → delete 2nd mail

& w filename → It writes to

—X—

34

* User Administrator :-

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

→ UNIX/LINUX is a multi user & multitasking O/s. Red hat Linux uses UPG (User private group) Scheme. According to UPG scheme, you create the any user account, the user consists the primary group with same NAME and same ID.

→ User always get created with primary group. one primary user has one primary group only. But the primary group may have no of primary users. The users home directory defaultly created under "/home" directory with his own name

→ The mail account defaultly created "/var/spool/mail".

→ Every user and group in the system are identified by a unique number ID.

→ Users are classified into two types.

① System Users

② Normal Users.

① System users : This type of users accounts are generally created by the ~~Super user (root)~~. O/s (or) By an application

→ The System users having ID value from 0 to 499

② Normal users : This type of accounts are generally created by the Super user (root).

→ For this type of users have UID's & GID's will be ranging in b/w 500 to 60,000

→ The users & group information maintained by the four database files.

(i) /etc/passwd : This database file maintains the user related information

Raju : x : 500 : 500 : : /home/raju : /bin/bash

- Raju → username
- x → Encrypted password
- 500 → UID
- 500 → GID
- (empty) → comment
- /home/raju → home directory
- /bin/bash → login shell

(ii) /etc/shadow : This file maintains the user related information like uname, Encrypted pwd, etc. To Encrypt the passwords MD5, DES algorithms are used

Raju : $1kj$12sfdhs : 12502 : 0 : 99999 : 7 : : : → extra field.

- Raju → username
- $1kj$12sfdhs → Encrypted pwd
- 12502 → ex
- 0 → ....
- 99999 → ....
- 7 → ....
- : : : → inactive days

35

(iii) /etc/group : It maintains group related information like group name, GID etc.

```
Team : x : 600 : user1, user2
 ↓     ↓    ↓        ↓
gname  mask gid   member's of a group.
       password
```

(iv) /etc/shadow : It maintains the group password related information.

```
Team : x12$abc.... : Aust : India, sri, pak
 ↓        ↓          ↓          ↓
gname   Encrypted   group    member's of a group.
        password of admin
        group admin
```

→ Tools : - useradd
- usermod
- passwd
- userdel
- chage

useradd : This cmd is used to create the users with default values. If you want to create the user A/c with your required values then mention the options.

Syn : #useradd <options> <username>

options :
- -u — UserID
- -g — groupID (primary)
- -G — groupID (secondary)
- -c — comment
- -d — directory
- -s — shell
- -f — Inactivedays
- -e — expire date (YYYY MM DD)

Ex: #useradd -u 555 u1
#useradd -g 1555 u2
#useradd -G 555 u3
#useradd -c "Sales" u4
#useradd -d /mnt/u5 u5
#useradd -f 1 u6
#useradd -u 777 -g 500 -G 501 -c "Linux-Admin" -d /opt/ug -c /bin/ksh -f

36

usermod: In usermod command also we use the same of useradd cmd, the extra options in this command are,

- `-l` - To change the username
- `-L` - To lock the user account
- `-U` - To unlock the user account

Syn: `# usermod <options> <username>`

```
#usermod -l Raju Rao
#usermod -L Raju
#usermod -U Raju
```

passwd: using this command we can give the passwords to the user accounts.

Syn: `#passwd <user-name>`

Ex: `#passwd Raju`

→ To disable password `# passwd -d <user-name>`

userdel: **Syn:** `# userdel <user-name>`

```
# userdel Raju↵
# userdel -r Raju ↵
```

→ using this command we can delete the user along with his properties.

chage: using this command we can change the (or) see the user password aging information.

Ex: `#chage -l U7 ↵`

→ To change the password information

Ex: `#chage U7 ↵`

—x—

→ To add users without overriding

`#usermod -G <existing group> <user>`

37

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.

(i) primary group.

(ii) Secondary group.

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

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

→ Tools: - groupadd
- groupmod
- groupdel
- gpasswd

groupadd: with this command we can create the group account

Syn: #groupadd <options> <groupname>

#groupadd colors ↵

#groupadd -g colors → To change gid

Ex: #groupadd -g 888 purchase

groupmod:

Syn: #groupmod <options> <groupname>

→ To change the gid

#groupmod -g 888 sports

→ To change group name

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

#groupmod -n sports1 sports

38

gpasswd: → using this command we can do two tasks

(i) Assign the password to the group:

#gpasswd <gname>

(ii) Add (or) remove the secondary users of the group:

-a – to add single user

-M – to add multiple users at a time

-d – to delete the user

-A – to make one user as a admin of group.

Ex: #gpasswd -a u1 Sales

#gpasswd -M u2,u3,u4 Sales

#gpasswd -d u1 Sales

#gpasswd -A u2 Sales

groupdel:

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 del the group account.

Syn: #groupdel <gname>

#groupdel team

→ If you want to login as a normal user

#su - u3 ↵

→ If you want to quit

#Exit ↵

→ The user home directory defaultly contains the below hidden files.

(i) .bash_logout (ii) .bash_profile (iii) .bashrc

→ To manage the user & group administration with graphically

#System-config-user &

—x—

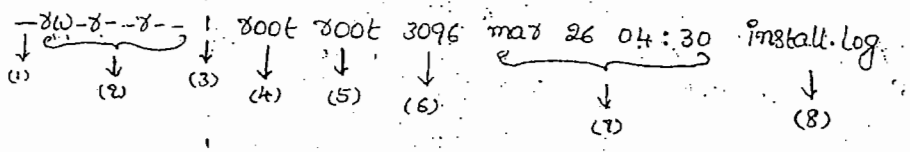
39

* permissions: permission is a mechanism used in the o/s for allowing the access (or) denying the access and particular resource for users and groups.

→ Types of permissions: (i) Basic file permission
(ii) Advanced filepermission
(iii) ACL (Access control List)

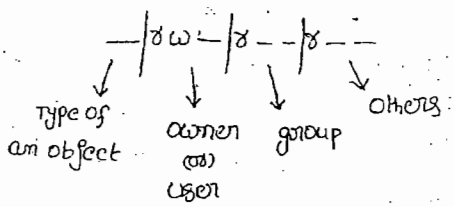
(i) Basic file permission:

→ In Linux/unix the file (or) Directory has 8 attributes



(1) Type of an object (2) Access permission (3) Links (4) User (5) group
(6) Size in bytes (7) Last modification Date & Time. (8) file-name.

→ Basic file permissions:



- Normal file
d - directory
l - Linkfile
b - block device file
c - character based file
S - Socket file

weight	Access mode's	Symbol	Access level	Symbolic
4	read	r	owner	u
2	write	w	group	g
1	Execution	x	Other	o

→ Default file permission:

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 #022
→ For Normal user umask value #002

→ Maximum permission of a file: -rw-rw-rw-

	6	6	6
(umask)	0	2	2
Default file permission	6	4	4

d rwx rwx rwx
7 7 7
0 2 2
7 5 5

→ Maximum permission of a directory:
Default directory permission 4 0 7 5 5

→ Default file permission of a file 644

→ Default Directory permission is 755

→ To see the UMASK

#umask

→ To view UMask value from file

#vi /etc/bashrc

→ Tools: - chmod
- chown
- chgrp

→ Operators: + → To add a permission
- → To remove a permission
= → To override the permission.

→ Weight of access mode's:

Access mode's	Symolic	Numaric values
(a) Execution	X	1
(b) Write	W	2
(c). Excution + write	[X+W]	3
(d) Read	R	4
(e) Read + execution	[R+X]	5
(f) Read + Write	[R+W]	6
(g) Read + Write + Execution	[R+W+X]	7

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

→ Use this command we can assign and remove the permissions.

Syn: #chmod <permissions> <file/directory name>

Ex: #chmod 766 Raju (or) #chmod R+W+X, R+W, R+W Raju

To full permission # chmod 777 Raju

#chmod ugo =RW Raju

→ To removing the permissions and group

41

Chown & chgrp: This commands are used to change the ownership, and also we can change the group of the file.

Chown: This Cmd is used to we can change the owner of the file, as well as owner & group at a time

#chown <username> <file name>

#chown ram /Linux

#chown Raju:Raju dir1 → Assign owner & group name

#chown -R Raju:Raju dir1

Chgrp: By using this Cmd we can change the group of the file

Syn: #chgrp <groupname> <file(or)d name>

#chgrp color /Linux

#chgrp sales sun

→ To ~~change~~ view the status of a file

#file install.log

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

#stat install.log

→ To change the permissions in GUI mode:

#nautilus & ↵

→ Assign the permissions in GUI mode: Right click on source - properties - permis

(II) Advanced file permission: (a) Sticky-bit
(b) Suid (Set-userid)
(c) Sgid.

Suid: By using this mode of permission the admin can share the administrative command to the normal users.

#whereis fdisk

#chmod u+s /sbin/fdisk

#ls -l /sbin/fdisk → To verify

#whereis ping

#chmod 4755 /bin/ping

42

Sgid : It is a type of permission used for inheriting the parent directorys group ownership. It is mainly used for making the group ownership constant for the files & directorys created under a parent directory.

→ create a directory

#mkdir /oracle

→ create a group

#groupadd Sales

→ To assign full permission to the directory

#chmod 777 /oracle (or) #chmod 2777 /oracle

#chmod g+s /oracle

→ To change the ownership

#su - Ramesh

$cd /oracle

Sticky-bit : It is a type of advance file permission. using an admin can assign full control on a particular directory a part from deleting

→ create a directory

#mkdir /oracle

1 → sticky bit

0 → remove sticky bit

#chmod 1777 /oracle

#ls -ld /oracle

→ login to the ramesh

#su - Ramesh

$cd /oracle

$rm -rf file1

→ To remove the sticky bit

#chmod 0777 /oracle

43

(iii) ACL (Access control List):

Using ACL admin can assign multiple set of permissions on users as well as groups on same resource.

→ ACL can only applied on ACL Enable partition.

→ Tools: -setfacl
-getfacl

→ Implementation of ACL's:

→ create the partition
#fdisk /dev/sda

→ without restarting to update the kernel
#partprobe /dev/sda

→ format the drive & mount with ACL support
#mkfs.ext3 /dev/sda8

→ create a directory #mkdir /acl

→ mount the partition with acl option
#mount -o acl /dev/sda8 /acl

→ For mounting an existing drive
#mount -o remount,acl /dev/sda2 /var

→ create the users and group
#useradd u1 #groupadd sales
#useradd u2 #groupadd -g sales u3

→ To assign the permissions at user level
#setfacl -m u:user1:r /acl/test

→ To assign the permissions at group level
#setfacl -m g:sales:r /acl/test
#setfacl -m u:user2:rw, u:user3:- /acl/test

→ To checking the acl permission
#getfacl /acl/test

→ To remove the users from the list

* Disk Management :-

⟹ **partitions**: We can create the partition at the time of installation and also after installation.

→ partition is a mechanism used for specifying sizes (or) Boundarys with in a single device

→ If we create the partition at the time of installation we have the tool is Disk Druid

→ To create the tool after installation

fdisk, parted, sfdisk etc.....

→ **For creating a partition:**

Syn: #fdisk <device name>

options: m – To get the help
p – To point the partitions table
n – To add a partition
d – To delete a partition
w – To save the modification
q – To quit the modification.

→ To update the partition table information to the kernal without restarting the system

#partprobe /dev/sda ↵

→ To checkout the hard disk

#fdisk -l

45

⇒ Formatting: It is a concept used for laying a structure on a raw storage device so that we can store the data in a systematical manar in a storage device

#mkfs <partition no>
#mke2fs <partition no>
#mkfs.ext2 <partition no>
#mkfs -t ext2 <partion no>
} To format the partition ext2 file system

#mke2fs -j <partition no>
#mkfs -j <partition no>
#mkfs -t ext3 <partion no>
#mkfs.ext3 <partion no>
} To format the partition ext3 file system.

⇒ Mounting :- Allocation of access points for storage device is called as mounting

→ Logical linking b/w physical device to a directory is also called as mounting

Syn: #mount <partition name> <mount point>

Ex: #mount /dev/sda5 /mnt

→ To check the mounting partition information

#mount

→ To check the used & Freee space

#df -h (or) #df -H

→ For unmounting a partition

#umount <mount point>

→ For making a partition permanently

#vi /etc/fstab

/dev/sda5 /mnt ext3 defaults 0 0

Note: Temporary mounting point information maintains in /etc/mtab, permanent information maintains in /etc/fstab.

46

⇒ Tuning :- Converting the file system from one type to another type is called as tuning.

→ On Linux we can convert file system type from ext2 to ext3 and viceversa without loss of data

→ Before converting we must unmount the partition

```
#umount /wisdom
```

→ converting from ext2 to ext3 :

```
#tune2fs -j /dev/sda6
```

→ converting from ext3 to ext2 :

```
#tune2fs -O ^has_journal /dev/sda6
```

⇒ Labelling :- Labels are only assign to the o/s partition.

→ It is mainly used for providing individual identification for the storage device.

→ how to see the label of a partition

```
#e2label /dev/sda1
```

→ To assign the label

```
#e2label /dev/sda6 /songs
```

→ To see the mount point information along with labels

```
#mount -l
```

→ at a time all file systems are unmounted except root(/)

```
#umount -a
```

⇒ Swap :- (virtual memory)

→ For improving the system performence we can dedicate a part of hard drive has virtual memory

→ while installation we can assign the swap memory as double size of the RAM. After installation if you need the more swap memory

→ To create the partition

#fdisk /dev/sda8

→ To update the partition

#Partprobe fdisk /dev/sda

→ Format the partition with swap file system

#mkswap /dev/sda8

→ To Enable the swap file system

~~#swapon -s~~

#swapon /dev/sda8

→ To disable the swap file system

#swapoff /dev/sda8

→ To check the swap partition

#vmstat

→ To mount the floppy drive

#mount /dev/fd0 /mnt

→ To mount the pendrive

#mount /dev/sda1 /mnt

→ To mount the CD: #mount /dev/cdrom /mnt

→ To mount the DVD: #mount /dev/dvd /mnt

→ To mount the IDE tape drive

#mount /dev/ht0 /mnt

→ To mount SCSI tape drive

#mount /dev/st0 /mnt

—— x ——

48

* Quotas:

→ Quotas is a mechanism used in the o/s for specific size of a hard disk for a particular user (or) group.

→ It is mainly used for restricting users so that they should not occupied the entire available disk space.

→ Quotas improve system performance

→ Quotas can be applied only on quotas Enable partition.

→ It can be applied in two types.

→ on the basic of no.of blocks

→ on the basic of no.of inodes (or) files

→ 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 the group.

→ To implement quotas on partition, it should be support quotas features (usrquota/grpquota)

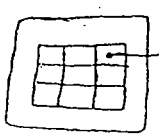
edquota: By using this we can apply restrictions for the users & groups. The edquota can automatically identify the quota Enabled partitions.

→ In edquota programe we can apply restrictions in terms of inodes (or) blocks.

→ Tools: - Quotacheck
- Edquota
- repquota

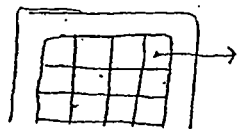
→ Sizes allocated for Every block By the file System:

→ less than 1gb [partition /dev/sda3]



1block = 1kb.

→ More than 1gb [partition /dev/sda9]



1block = 4kb

→ creating the partition

#fdisk /dev/sda ↵

→ without rebooting to update the kernel

#partprobe /dev/sda ↵

→ format the partition

#mkfs.ext3 /dev/sda10

→ create the mount point

#mkdir /quota

→ mount the partition with quota options

#mount -o usrquota,grpquota /dev/sda10 /quota

→ To assign the fullpermissions to the mount point

#chmod 777 /quota

→ To scan the file system (or) generate the database files

#quotacheck -cugv /quota

→ To Enable the quota on the file System

#quotaon /quota

→ Create users and group

#useradd u1 #groupadd Sales

#useradd u2 #useradd -g Sales p1

→ To assign the restrictions at userlevel

#edquota -u u1

filesystem	blocks	soft	hard	inodes	soft	hard
/dev/sda10		20	30			

→ To assign the quota at group level

#edquota -g sales

→ To see the report of quotas #repquota /quota

→ To disable the quota's

#quotaoff -/quota

→ To change the all users grace period #edquota -u -t ↵

→ To change the all groups grace period #edquota -g -t ↵

—x— .50

* JOB SCHEDULING *

→ To Schedule the tasks we use the Schedule task (or) Automation of Jobs.

→ We can automate the jobs using Three utilities.

(a) Batch

(b) At

(c) Crontab

(a) Batch : The batch jobs will execute, when server is free and server load is less.

```
#batch ↵
at> ls
at> ctrl+d
```

(b) At : The job will execute only once. After execution the job was killed automatically

Ex: (i) #at now ↵ → It Executes the job immediately.

```
at> mkdir wisdom
at> ctrl+d (Save & quit)
```

(ii) #at 10 am (or) pm

```
at> sh script1 ↵
    ctrl+d
```

(iii) #at 12302012 (mm dd YY)

```
at> sh script2
```

(iv) #at now + 3minutes ↵ after 3minutes

(v) #at 4pm tomarrow ↵ Tomarrow at 4'o clock.

(vi) #at 8pm + 5days

→ To see the at job's list : #atq (or) #at -l ↵

→ To see the job details: #at -c <job id>

→ To remove job : #atrm jobid ↵

51

→ To restrict a user to generate at job, his name has to be mentioned in the file /etc/at.deny. Defaultly only /etc/at.deny file is available. If /etc/at.allow is created then /etc/at.deny file doesn't consider and the users who mentioned in the /etc/at.allow file only can generate the at jobs.

(c) crontab: It executes the given jobs repeatedly in server account.

→ To generate a crontab we have to type

#crontab -e ↵

→ When we type the above command a fill will be opened in vi editor

→ crontab file fields:

*	*	*	*	*	*
minutes (0-59)	hours (0-23)	day of month (1-31)	month of year (1-12)	weekdays (0-6)	Job/task

Ex: (i) * * * * * date → Every minute it execute date cmd

(ii) 30 10 * * * sh script1 → every day at 10:30 AM

(iii) 30 10 * * 0 sh script2 → Every Sunday at 10:30 AM

(iv) 30 22 * * 1,3,5 sh script3 → Every MON, WED, FRI at 10:30 pm.

(v) 30 22 1,15 * * sh script4 > file1 → It Executes script4, Every month 1st and 15th at 10:30pm and writes output into file1.

(vi) 30 15 * * * /bin/echo "Good Evening" → Display msg at 3:30pm.

(vii) 59 23 31 12 * /bin/echo "HAPPY NEW YEAR"

(viii) * 20 * * * dump -0uf /dev/st0/dump /dev/hda1 → Every 20 min to take the backup.

→ To see the crontab jobs: #crontab -l ↵

→ To remove all crontab jobs: #crontab -r ↵.

→ To remove particular job then open the crontab file and delete job.

→ crontab jobs location: /var/spool/cron.

→ To check the cronjobs:

#rpm -qa | grep cron ↵

→ How to control the crontab:

→ To control the cronjobs using two files

(a) cron.deny

(b) cron.allow

→ To restrict a user to generate cronjob, his name mentioned in the file /etc/cron.deny. Defaultly only /etc/cron.deny file is available. If /etc/cron.allow is created then /etc/cron.deny file doesn't consider and the users who mentioned in the /etc/cron.allow file only can generate the cron jobs.

#useradd Raju ↵

#vi /etc/cron.deny

Raju

→ Now try to open crontab job user Raju

→ To open the crontab:

#crontab -e -u username ↵

→ System level crontab files:

The format of /etc/crontab and the files in /etc/cron.d are different from user crontabs. The sixth field is a username which will be used to execute the command in the seventh field.

vi /etc/crontab ↵

02 4 * * * root run-parts /etc/cron.daily

Every morning, at 4:02, all of the executables in the /etc/cron.daily directory will be run as root.

→ A common cmd in these files is the "run-parts" shell script. This

53

* System Initilization *

→ Redhat Enterprise Linux Release:

#cat /etc/redhat-release ↵

→ How to identifying your kernel:

#uname -r ↵

→ How to see available kernels:

#yum list installed kernel*

(or)

#rpm -qa kernel*

⇒ **Runlevels:** The runlevels determines which services are started automatically on your Linux System.

→ Most Linux desktop Systems are set to boot up to runlevel 5 (multiuser networking, GUI).

→ Many server systems boot to runlevel 3 (multiuser, networking, no GUI).

→ you can change the runlevels immediately using the init command, at boot time from the boot loader, or permanently in the inittab file.

→ The following types of runlevels are available.

(i) Init 0: halt - Shutdown to a poweroff state (or) #Shutdown -h now ↵

(ii) Init 6: reboot - shutdown and soft reboot (or) #Shutdown -r now ↵

(iii) Init 1: Single usermode. In this normal user can't login to the pc.

(iv) Init 2: Multiuser mode without NFS.

(v) Init 3: full multiuser text mode, typical for Servers.

(vi) Init 4: officially undefined.

(vii) Init 5: Giraphical login, typical for Desktops and Laptops.

→ Runlevels are maintained in the file:

#/etc/inittab

→ To check the runlevel:

#runlevel (or) #who -r ↵

* Boot Sequence *

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

→ Linux crosses the 4 levels of booting.

(a) Hardware Boot
(b) Boot Loader
(c) Kernel
(d) Initprocess

(a) Hardware Boot :

- BIOS Initialization.
- performs first POST (power on self Test)
- It checks all h/w connectivity, if all things are correct then it gives a health beep.
- Boot strap finds the device from where to boot

Ex: Hard-disk, CD-ROM, Floppy.

(b) Boot Loader :

→ The Linux we have default boot loader called GRUB.
→ GRUB means "Grand Unified Boot Loader)
→ In older versions we used to have LILO (Linux Loader).
→ GRUB supports non-multiboot operating system, via chain Loading.
→ GRUB can be used with a variety of different interfaces.
→ In this stage it read's the file "/boot/grub/grub.conf" and Initrd images which are next in the process of booting.

(c) Kernel :

→ Kernel initializes the devices.
→ It mounts the root file System in read only mode.
→ It starts first process called as init process.
→ In this stage it reads the file "/etc/fstab"
→ Initrd is used by kernel as temporary root file System untill kernell in bootload and the root file System is mounted

(d) Initprocess:

→ init reads "/etc/inittab" file.

→ this file contains what programs (or) services should be run at different run levels.

→ Run levels script files are available under /etc/rc.d

→ When Redhat boots, the boot process (or) will run a number of scripts located in sub directories under /etc/rc.d. The boot process first runs the scripts found in /etc/rc.d/rc1.d which provides only the most basic functionality and the ability to only handle a single user. This stage is known as "single user mode". After completing this first phase, the boot process will run scripts in only one of the other directories depending on the startup mode. ~~these are listed bellow (i.e called Run levels)~~.

—x—

⇒* Console role: while Installation:

Alt+ctrl+F1 ⇒ Text Based Installation.
Alt+ctrl+F2 ⇒ Maintains RAM info,.
Alt+ctrl+F3 ⇒ Debugging/Error info,.
Alt+Ctrl+F4 ⇒ Kernel related info,.
Alt+Ctrl+F5 ⇒ Boot Loader info,.
Alt+Ctrl+F6, Alt+ctrl+F7 ⇒ Graphical Based Installation.

After Installation:

Alt+Ctrl+F1 ... Alt+Ctrl+F6 ⇒ CUI Console modes
Alt+Ctrl+F7 ⇒ GUI Console modes
} defaultly

→ How to Increase the virtual consoles:

→ open the file: #vi /etc/inittab ↵

go to Line no 14: 8:2345:respawn:/sbin/mingetty tty8
9:2345:respawn:/sbin/mingetty tty9

56

* RAID: (Redundant Array of Inexpensive (or) Independent Disks)

→ Collection of individual physical disks as a single ~~metadevice~~ disk, called as a metadisk.

→ Mainly we implement the RAID in order to increase the storage capacity along with data security.

→ Features of RAID:
- Backup
- Fault tolerance
- Redundency

→ Types of RAID: → Two types of RAIDS.
① Software RAID
② Hardware RAID

① Software RAID: A simple way to describe software RAID is that the RAID tasks runs on the CPU of your computer system.

→ Software RAID is RAID implemented as kernel (or) driver-level software for a particular operating system.

② Hardware RAID: A hardware RAID solution has its own processor and memory to run the RAID application. In this implementation, the RAID system is an independent small computer system dedicated to the RAID application, off-loading this task from the host system. Some hardware vendors implements RAID data protection directly in their hardware as with disk array controller card

→ Comparision of Software & Hardware RAID:

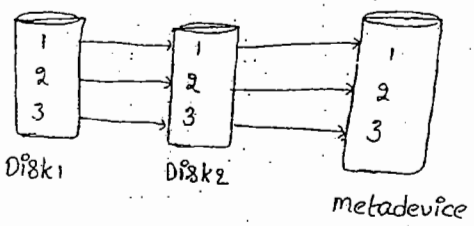
→ Hardware fault tolerance is more expensive than Software fault tolerance.

→ Hardware fault-tolerance generally provides faster disk I/O than Software fault tolerance.

→ RAID levels:
- RAID0 [Striped volumes]
- RAID1 [Mirrored volumes]
- RAID4 [parity]
- RAID5 [Striped volumes with parity]

57

RAID1 : → minimum 2 harddisks, maximum 32 hard disks
→ Data is written simultaneously
→ Read speed is fast and write speed is slow
→ Fault tolerance is available
→ Overhead is 50%



Implementation :-

→ create two partitions

#fdisk /dev/sda↵

→ Without restarting to update the kernel

#partprobe /dev/sda↵

→ create the metadevice

#mdadm -C /dev/md0 -n2 /dev/sda{11,12} -l1 ↵

→ For checking the major & minor number

#cat /proc/partitions ↵

→ For displaying the properties

#mdadm -D /dev/md0 ↵

→ For accessing the metadevice

#mkfs.ext3 /dev/md0

#mkdir /raid1

#mount /dev/md0 /raid1 ↵

#cd /raid1

#~~touch 1 2 3 4~~ touch 1 2 3 4 ↵

→ For stopping the RAID device

#mdadm -S /dev/md0

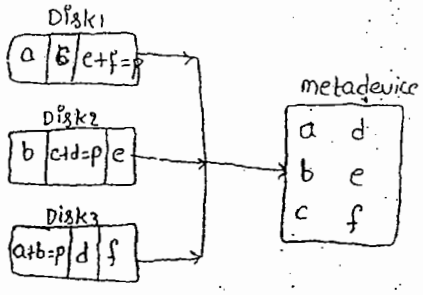
→ To add the partition to the meta device

#mdadm -a /dev/md0 /dev/sda13

58

RAID5 : [Striped volume with parity]

→ minimum 3 hard disks, maximum 32 disks
→ Data is written simultaneously and evenly accross multiple harddisks.
→ the parity is written Equally on all disks.
→ the reading & writing speed is fast
→ Faulttolerance is available



Implementation :

→ Create the four partitions
#fdisk /dev/sda↵
→ To update the kernel without restarting
#partprobe /dev/sda↵
→ Create the metadisk
#mdadm -C /dev/md1 -n3 /dev/sda{8,9,10} -l5
→ For displaying the properties
#mdadm -D /dev/md0
→ For accessing the metadevice
#mkfs.ext3 /dev/md1
#mkdir /raid5
→ Mount the metadevice & store the data
#mount /dev/md1 /raid5
#cd /raid5
#touch 1 2 3 4 5
→ For checking the raid level
#mdadm -f /dev/md1 /dev/sda9

59

→ For adding the spare device

```
#mdadm -a /dev/md1 /dev/sda11
```

→ To remove the faulty partition

```
#mdadm -r /dev/md1 /dev/sda8
```

Note: Before stopping the RAID we must note the RAID partitions list. Because while activation we must mention the partition list.

→ To stop the RAID

```
#mdadm -S /dev/md1
```

→ To activate the metadevice /Raid

```
#mdadm -A /dev/md1 /dev/sda{8,9,10}
```

→ To see the RAID config file

```
#cat /proc/mdstat ↵
```

——— x ———

60

* LVM (Logical volume Manager)

→ Lvm is a mechanism use for providing specility of Extending (or) reducing the sizes of an Existing partitions.

→ In this concept we must follow the bellow sequence

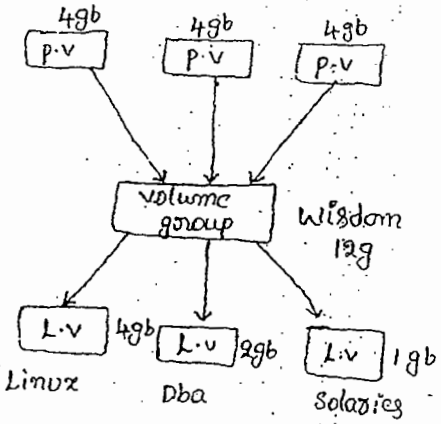
- physical volume
- Volume group
- Logical volume

physical volume: The Storage device (or) physical device is known as physical volumes under Lvm's.

volume group: The collection of physical volumes is called as volume group

Logical volume: The sizes specified from a volume group is known as Logical volume.

→ The volume group is divided into logical volumes.



→ Implementation of Lvm after Installation

→ To create partitions

```
# fdisk /dev/sda
```

→ To update the kernal without restarting

```
# partprobe /dev/sda
```


61

→ Create the physical volume:

```
#pvcreate /dev/sda{1,2,3}
```

→ To see the physical volume details

```
# pvdisplay (or) # pvs
```

→ Create the volume group:

```
#vgcreate wisdom /dev/sda{1,2,3}
```

→ To see the volume group info:

```
#vgdisplay (or) #vgs
```

→ Create the Logical volumes:

```
# lvcreate -L+6gb -n /dev/wisdom/Linux.
#lucreate -L+2gb -n /dev/wisdom/ Dba
#lvcreate -L+2gb -n /dev/wisdom/Solaries
```

→ To see the logical volume info:

```
#lvdisplay (or) #lvs
```

→ format the logical volumes:

```
#mkfs-ext3 /dev/wisdom/Linux
#mkfs.ext3 /dev/wisdom/dba
#mkfs.ext3 /dev/wisdom/Solaries
```

→ To mount the logical volumes:

```
#mount /dev/wisdom/Linux. /mnt
#mount /dev/wisdom/dba /mnt
#mount /dev/wisdom/Solaries /mnt
```

→ To increase the Logical volume size

```
# lvresize -L+4g -n /dev/wisdom/Linux
```

→ To decrease the Logical volume size

```
#lvresize -L-1g -n /dev/wisdom/dba
```


62

→ To increase the volume group size:

```
#vgextend wisdom /dev/sda4
```

→ To decrease the volume group size:

```
#vgreduce wisdom /dev/sda4
```

→ Before increase the volume group, create the physical volume

→ To remove the logical volume:

→ Before removing logical volume first unmount it

```
#umount /mnt
#lvremove /dev/wisdom/Salaries
```

→ To remove the volume group:

```
#vgremove wisdom /dev/sda1 /dev/sda2 /dev/sda3
```

→ To remove the physical volumes:

```
#pvremove /dev/sda1 /dev/sda2 /dev/sda3
```

→ To manage the LVM in graphical mode:

```
#System-config-lvm &
```

———x———

63

* package management *

=> Tools for installing any application

- rpm
- yum

=> rpm :- (Redhat 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.

→ package parameters:

ex: vsftpd-2.0.5-0.el5.i386.rpm

- vsftpd → package name
- 2.0.5-0.el5 → version name
- i386 → Architecture no
- rpm → Extension of package

→ 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.

→ To install the package:

Syntax: #rpm -ivh <pkg-names --force --aid

-i = install
-v = verbose
-h = installation process display in 50 hashes
--force = install the pkg with forcefully
--aid = install the pkg along with dependencies

65

→ For installing the package with rpm :-

→ First mount the media

#mount /dev/dvd /mnt

#cd /mnt/Server

→ For installing an application

#rpm -ivh firefox-1.5.0.9-10.el5.i386.rpm

→ For installing the application forcefully

rpm -ivh vnc* --force

⇒ For uninstalling a package :-

Syntax : #rpm <option's> <pkg name>

-e = erase a package

--Nodeps = To erase an package without dependencies

→ For removing application without dependencies

#rpm -e vsftpd

→ For removing an application

#rpm -e vsftpd --nodeps

⇒ To querying the package :

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

-q = checks the pkg installed or not

-qa = query the availability of the pkg

-qd = query the documentation files of the pkg

-qc = query the configuration files of the pkg

-ql = query the list of files information

-qi = query the information of the pkg

-qs = query the status of the pkg

Ex. #rpm -qi vsftpd (To display information of the pkg)

⇒ Network Installation :

→ In this level we can install the packages form Server through NFS (or) FTP Services.

NFS Service : → First check the communication

#ping 192.168.0.250

→ For accessing the data shared from Server

#mount 192.168.0.250 : /var/ftp/pub/Server /mnt

#cd /mnt

#rpm -ivh bind* cash*

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

Syn :- #rpm -ivh ftp://192.168.0.250/pub/Server/<pkg names> --force --aid

ex: #rpm -ivh ftp://192.168.0.250/pub/Server/nfs* --force --aid↵

⇒ Yum (yellowdog update Modifier)

→ yum is a default pkg management tool in Redhat o/s.

→ By using yum tool we can install, upgrade, remove the packages and also we can see the information about package's.

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

→ using this tool we can install the required package with dependencies

⇒ For installing / uninstalling a pkg with yum :

Syn : #yum <options> <pkg name>

options :

install	group install
remove	group remove
list	group list
info	

67

→ To mount the dvd

```
#mount /dev/dvd /mnt
#cd /mnt/Server
#rpm -ivh vsftpd* --force --aid
```

→ copy the dump of o/s to the default sharable location ftp

```
#cp -rvf /mnt/* /var/ftp/pub
```

here '*' means all the data under the mount point /mnt

→ Install the create repo pkg

```
#rpm -ivh createrepo* --force --aid
```

→ create the new repo:

```
#createrepo -g /mnt/Server/repodata/comps-rhels-server-core.xml
/var/ftp/pub/Server
```

Note: If any Errors are showing then remove

```
#rm -rf /var/ftp/pub/Server/.olddata
#yum clean all
```

→ open the yum configuration

```
#vi /etc/yum.repos.d/rhel-debuginfo.repo.
```

1. Server
2. baseurl = ftp://192.168.0.250/pub/Server
3. Enabled = 1
4. gpgcheck = 0

→ Restart the ftp service

```
# service vsftpd restart ↵
```

→ To install the package

```
#yum install nfs* ↵
```

→ To upgrade the package

```
#yum upgrade nfs*
```

→ To remove the package

```
#yum remove nfs*
```

* Network Administration *

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

→ In the n/w System maintains the FQDN (Fully qualified Domain Name)

∴ FQDN = Host Name + Domain Name

→ To Change the Host Name: → To check the hostname #hostname↵

(a) Temporarily: #hostname Station254.wisdom.com

(b) permanently: #vi /etc/sysconfig/network

Networking=yes

Hostname=Station254.wisdom.com

→ For avoiding graphical problems

#vi /etc/hosts

192.168.1.254 Hostname Station254
(Station254.wisdom.com)

→ To change the IP address:

→ To check the IP address

#ifconfig ↵

(a) Temporarily:

Syn: #ifconfig eth0 <ipaddress> netmask <default submask> up↵

Ex: #ifconfig eth0 192.168.1.254 netmask 255.255.255.0 up↵

(b) permanently:

For GUI: (i)#neat↵ (ii)#System-config-network↵

For CLI: (i) #Setup↵ (ii) neat-tui↵

(iii) #System-config-network-tui↵

69

→ To Check the NIC card detected (or) not:

Syn: #ethtool <ethernetcard>

ex: #ethtool eth0 ↵

→ To Disable the NIC card:

#ifdown eth0 ↵

→ To Enable NIC card:

#ifup eth0 ↵

→ To Manage the Services:

(a) Temporarily:

Syn: #Service <Service name> <stop/start/restart>

Ex: # Service vsftpd restart ↵

→ To check the particular service status:

#Service vsftpd status ↵

→ To check the all services status:

#service --status-all ↵

(b) permanently:

Syn: #chkconfig --level <run levels> <service name> <on/off>

Ex: #chkconfig --level 146 vsftpd on ↵

→ To check the particular service status:

Syn: #chkconfig --list <servicename>

Ex: #chkconfig --List vsftpd ↵

→ To check the all services status:

#chkconfig --List ↵

* → To See the port numbers of all services:

#vi /etc/services

70

* NFS *

(Network file System)

→ The network file System is certainly one of the most widly used network services.

→ NFS is based on the Remote procedure call.

→ It allows the client to automount and transparently access the remote file systems on the network.

→ Using NFS we can share the files among homogenous Environment hence we can't share the files between unix to windows Environment

→ NFS was developed by Sun Micro Systems.

→ NFS was developed to allow the user to access remote directory as a mapped directory.

→ NFS is ~~a~~ not a single program, It is a suite of related programs

Features:

(i) Every one can access same data this eliminates the need to have a redundant data on Server Systems.

(ii) Reduces storage cost.

(iii) provides data consistency.

(iv) Reduces the System administrator overhead.

(v) Data Transfer rate in 3.0 is 32 kb bit but in 2.0 it is only 8kb.

prerequisites:

(i) verify NFS daemon Service:

Here we assume that the NFS Service daemon is already on our System, including portmap daemon on which NFS setup depends. one more thing, our System needs to Support the NFS file System. For this we needs to ~~Support the NFS file~~ & Issue the following command:

#cat /proc/filesystems

71

(ii) Server export file:

→ All NFS server exports need to be defined in /etc/exports file.

Most common exports options:

(a) /home/nfs/ 192.168.1.254(rw,sync) : Export /home/nfs directory for host with IP 192.168.1.254 with read, write permissions, and Synchronized mode.

(b) /home/nfs/ 192.168.1.254(ro,sync): Export /home/nfs directory for network 192.168.1.254 netmask 255.255.255.0 with read only permissions and Synchronized mode.

(c) /home/nfs/ 192.168.1.254(rw,sync,no_root_squash):

Export /home/nfs directory for host with IP 192.168.1.254 with read, write permissions, Synchronized mode and the remote root user will be treated as a root and will be able to change any file and directory.

(d) /home/nfs *(ro,sync): Export /home/nfs directory for any host with a read only permission and Synchronized mode.

(e) /home/nfs *.wisdom.com(rw,sync): Export /home/nfs directory for any host within wisdom.com domain with read, write permissions and Synchronized mode.

Requirements: pack: nfs *....rpm
portmap*...rpm

port no: 2049 - NFS
111 - PORT MAP

Config File: /etc/exports

Service : portmap and NFS

Daemon : nfsd mountd statd locked

⇒ Serverside configuration:

→ Install the packages:

#rpm -ivh nfs* portmap* --force --aid (or)

#yum install nfs* portmap* -y

→ Create the directory for sharing:

#mkdir /home/nfs ↵

#cd /home/nfs ↵

#cat > wisdom ↵ Create some files

→ Open the config file:

#vi /etc/exports

/home/nfs 192.168.1.0/255.255.255.0 (rw,Sync)

/home/nfs 192.168.1.254/255.255.255.0 (ro,Sync)

/home/nfs 192.168.1.254/255.255.255.0(rw,Sync,no_root_Squash)

/home/nfs *(ro,Sync)

/home/nfs *.wisdom.com(ro,Sync)

→ Restart the service:

#Service nfs restart

#Service portmap restart

→ To make a ~~permanent~~ permanent of this service

#chkconfig nfslock on ↵

#chkconfig nfs on ↵

⇒ Clientside configuration:

→ To check the nfs server sharing information:

#Showmount -e <nfs server ip address)

→ Create the local mount point:

```
#mkdir /nfsclient ↵
```

→ Now mount to the server sharing information:

```
#mount 192.168.1.254:/home/nfs /nfsclient ↵
#cd /nfsclient ↵
# ls
```

Limitations of NFS:

(i) We can only exports the directories which are started with "/".
(ii) We can Exports the files only to the homogeneous environment.
(iii) only local file systems can be exported.
(iv) It uses only in private network.

* /var/lib/nfs/rmtab: → this file contains exported list which are mounted by list
* /var/lib/nfs/etab: → this file contains currently exported file system information.

74

* SAMBA *

→ SAMBA Server is used to communicate the shares on windows.

→ Samba is a free software re-implementation of SMB/CIFS networking protocol, originally developed by Australian Andrew Tridgell.

→ As of version 3, Samba provides file and print services for various microsoft windows clients and can integrate with a windows server domain either as a primary Domain Controller (PDC) (or) as a domain member.

→ It can also be part of an Active Directory domain.

→ Samba using Smb protocol. It also called as "Server Message Block" also known as "netbios/tcp-ip"

Features of SAMBA:

(i) File/Directory Sharing
(ii) Browsing
(iii) Resource Sharing
(iv) User Authentication & Authorization.

Requirements:

packages: samba*....rpm

port no: 137 - Net BIOS name service.
138 - Net BIOS Datagram service.
139 - Net BIOS Session service.

Config File: /etc/samba/smb.conf

Service: Smb

Daemons: Smbd (Server message block daemon)

75

→ Configure SAMBA Server:

→ Before configuring the Samba server, first we check the Samba server is installed or not.

```
#rpm -qa | grep -i samba ↵
```

Note: If it is not installed, then it will not show any output. If the Samba server software is installed in the machine, it will show you the output as

```
#rpm -qa | grep -i samba ↵
    samba-3.0.33
```

→ Install the packages:

```
#rpm -ivh samba* --force --aid ↵ (or)
#yum install samba* -y ↵
```

→ Create the source for sharing:

```
#mkdir /common
#cd /common
#touch 1 2 3
```

→ open the configuration file:

```
#vi /etc/samba/smb.conf ↵
```

→ go to Last Line:

```
[common]
    comment      = This is Samba sharing info
    path         = /common
    valid users  = smb1 smb2
    public       = no
    writable     = yes
```

→ Test the configuration :

#testparm ↵

→ Testparm will parse your configuration file and report any unknown parameters or incorrect Syntax. It also performs a check for common misconfigurations and will issue a warning if one is found. So always run testparm again whenever the smb.conf file is changed!

→ After checked the Syntax, don't forget to restart the service. without restarting the service, action will not get effect.

→ Create Samba users and assign samba password :

#useradd smb1 ↵

#useradd smb2 ↵

#smbpasswd -a smb1 ↵

#smbpasswd -a smb2 ↵

→ Restart the Service :

#Service smb restart ↵

⇒ Clientside (Linux) :

→ To check the shares are visible or not :

#Smbclient -L //192.168.1.254 -N ↵

(or)

#Smbclient -L localhost ↵

→ We can access the sharing information in two ways :

(i) NFS (or) mount

(ii) FTP

77

→ NFS method:

#mkdir /Smbclient ↵

#mount //192.168.1.254/common /Smbclient -o username=Smb1↵

password : < Smb user password >

#cd /Smbclient ↵

#ls ↵

→ FTP method:

#Smbclient //192.168.1.254/common -U Smb2

password : < Smb user password >

Smb > quit

WINDOWS Client:

Open run prompt: And type the Samba server ip with path

\\192.168.1.254/common (or) \\192.168.1.254 ↵

Note: This share will not accessible after reboot. For permanent mount just make an Entry in /etc/fstab file.

//servername/sharename mountpoint smbfs defaults 0 0

78

FTP
(File Transfer protocol)

→ This protocol is used to download and upload the files over the internet. (or)

→ It is a standard method for sharing the files over the internet for many years.

→ FTP servers are still the most common way to make directories of documents and software available to the public over the interne

→ Types of FTP servers:

(i) FTP : Defaultly available on Solaries.
(ii) proFTP : For anonymous logins, it is a third party tool.
(iii) SFTP : Secure FTP
(iv) VSFTP : very Secure FTP.
(v) WU-FTP : Washington FTP.
(vi) Cute FTP

→ In linux defaultly SFTP, VSFTP.

Requirements:

packages : vsftpd....rpm

port no : 20 - FTP Data transfer
21 - FTP control Connection.

Config file : /etc/vsftpd/vsftpd.conf

Sharing loc: /var/ftp

Service : vsftpd

Daemon : vsftpd

79

→ Configuration :

→ Install the package :

```
#rpm -ivh vsftpd* --force --aid
(or)
#yum install vsftpd* -y ↵
```

→ Testing to see if vsftp is running :

```
#netstat -a | grep ftp ↵
```

→ open the configuration file :

```
#vi /etc/vsftpd/vsftpd.conf
```

anonymous_enable=yes #Line no 12 :

Defaultly anonymous users are enable. If you want to disable login permission to anonymous users then

anonymous_enable = no #Line no 12 :

→ once logged into a VSFTP server, you'll automatically have access to only the default anonymous FTP directory /var/ftp and all its subdirectories.

→ Go to sharing location :

```
#cd /var/ftp
#mkdir upload [for uploading files]
```

→ create some files in pub directory to download :

```
#cd pub ↵
#touch f1 f2 f3 ↵
```

→ Restart the service :

```
#service vsftpd restart ↵
#chkconfig vsftpd on ↵
```

→ Now go to client side & we can check it.

→ Accessing the private users: → open the config file
→ To disable the anonymous users

#vi /etc/vsftpd/vsftpd.conf

local_enable=yes Line no:15

anon_upload_enable=yes no:27

ftpd_banner = Welcome to ftp no:85

→ Create the Local users:

#useradd ftp1 #passwd ftp1

#useradd ftp2 #passwd ftp2

→ Create a user group and shared directory. In this case we'll use "/home/ftp-users" and a user group name of "ftp-users" for the remote users:

#groupadd ftp-users

#mkdir /home/ftp-docs

→ make the directory accessible to the ftp-users group:

#Chmod 750 /home/ftp-docs↵

#Chown root:ftp-users /home/ftp-docs↵

→ Add users, and make their default directory /home/ftp-docs:

#useradd -g ftp-users -d /home/ftp-docs user1

#useradd -g ftp-users -d /home/ftp-docs user2

#passwd user1

#passwd user2

→ Change the permissions of the files in the /home/ftp-docs directory for read only access by the group

#Chown root:ftp-users /home/ftp-docs/*

81

→ Users should now able to login via ftp to the server using their new user names and passwords.

→ If you absolutely don't want any FTP users to be able to write to any directory then you should comment out the write_enable line in your config file

#write_enable=YES Line no:18

→ Now Restart the Service:

#Service vsftpd restart

→ To Restriction local user:

→ If we want to restrict any normal users to login into the FTP server then mention their names in the file.

#vi /etc/vsftpd/ftpusers

ftp1 [restrict ftp1]
user1 [restrict user1]

→ Accessing ftp service by specified users:

#vi /etc/vsftpd/user_List↵

ftp2
user2

→ Open the config file:

#vi /etc/vsftpd/vsftpd.conf

go to last line: userlist_deny=NO

→ FTP users can't login telnet:

#usermod -s /sbin/nologin ftp1
#usermod -s /sbin/nologin user2

82

⇒ Clientside:

→ To connect FTP server:

ftp <ftp-server-ip>

ex: # ftp 192.168.1.254 ↵

→ Type username & password

⇒ FTP prompt cmds:

ftp > pwd → Display serverside directory

ftp > ! pwd → Display clientside working directory.

ftp > ls → List the serverside info

ftp > ! ls → List the clientside info

ftp > cd → change directory at serverside

ftp > ! cd → change directory at clientside.

ftp > get <filename> → To download single file

ftp > mgt → To download multible files.

ftp > put → To upload a single file

ftp > mput → To upload multible files.

ftp > help → display the prompt cmds.

ftp > bye → To quit ftp prompt

—x—

83

* Scp : (Secure copying)

→ Scp is a command which copies/pushes the files in remote locations.

→ Scp is available as a part of the Secure Shell package that is normally installed by default on Redhat.

→ There is a windows scp client called WinScp.

→ Secure copy is installed in parallel with SSH and they always run simultaneously on the same tcp port.

→ Scp doesn't support anonymous downloads like FTP.

→ To pull the file:

```
#scp 192.168.1.254:/root/file1 .↵
                              ↑ current location
```

→ To push the file:

```
#scp file1 192.168.1.254:/root/Desktop↵
```

* Create ISO image of your O.S:

→ Insert the DVD:

→ Mount the DVD:

```
#mount /dev/dvd /mnt
```

→ To create 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 /mnt↵
#cd /mnt↵
#ll
```

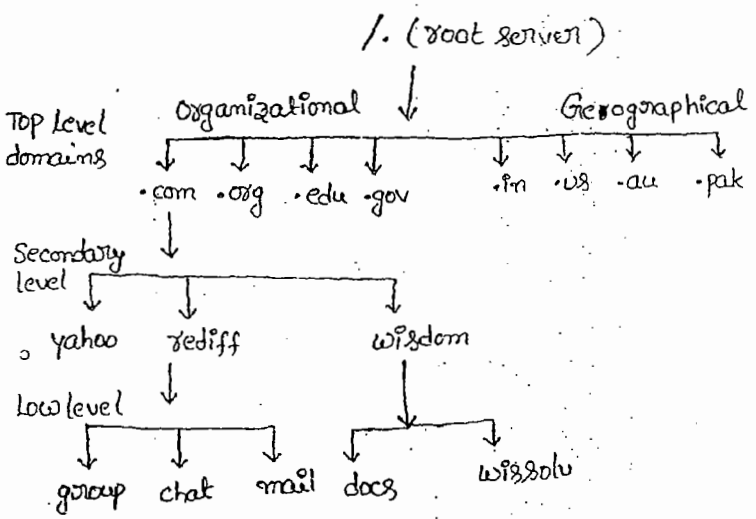
— X — 84

DNS
(Domain Naming System)

→ The Domain Naming System is the way in which a URL (or) domain like www.wisdom.com is converted to an ip address.

→ Domain Name System, provides resolution of names to ip address and ip address to Names.

→ Defines a hierarchical name space where each level of the name space is separated by a "."



→ Top level domains classified into two types.

1. Organizational : Based on purpose (or) function of the domain
2. Geographical : Based on physical location

→ All the root servers are maintained by IANA

→ Top level domains are maintained by Inter NIC

→ Secondary level domains are maintained by com levels like yahoo, rediff

Host file:

→ It provides resaultion of Hostnames to ip address.

→ It can only resolve the name provided in the local host file.

→ you can add the name and ip addresses in /etc/hosts file but we should have to maintain it all the systems in the Nlw.

85

The "/etc/resolve.conf File:

This file is used by DNS clients to determine both the location of their DNS server and the domains to which they belong.

(a) Nameserver: IP address of your DNS nameserver. There should be only one entry per "nameserver". If there is more than one nameserver, you'll need to have multiple "nameserver" lines.

(b) Domain: The local domain name to be used by default. If the server is station254.wisdom.com, then the entry would just be wisdom.com

Ex: # nameserver 192.168.1.254

The Zone files:

→ In all zone files, you can place a comment at the end of any line by inserting a semi-colon ";" character then typing in the text of your comment.

→ By default, your zone files are located in the directory "/var/named".

Zone: Zone is a storage database. which contains all zone records.

→ Two types of zone records are available

(a) FLZ (Forward Lookup Zone): Used for resolving host name to ip address

→ It maintains host to ip address mapping information.

(b) RLZ (Reverse Lookup Zone): Used for Resolving ip address to host name.

→ It maintains ip address to host mapping information.

→ Each zone file contains a variety of records (eg: SOA, NS, MX, A and CNAME)

SOA (Start of Authority): The very first record is the start of Authority record which contains general administrative and control information about the domain.

NS (Name Server): The NS resource record identifies all name servers that can perform name resolution for the zone.

(or)

Point the name of the nameserver for the domain.

A (Address) : This record resolves from host name to ip address.

PTR (pointer) : This record resolves from ipaddress to hostname.

CNAME (Canonical Name) : provides additional alternate "alias" names for servers listed in the "A" record.

MX (Mail Exchange) : Lists the mail servers for your domain

Requirements :

packages : bind*.....rpm
Caching*.....rpm

port no : 53 – DNS

service : named

configuration files : /etc/named.caching-nameserver.conf
/etc/named.rfc1912.zones

Zone file location : /var/named/chroot/var/named.

Configuration :

→ Install the packages

#rpm -ivh bind* caching* --force --aid

#yum install bind* caching* ↵

→ Configure the hostname

vi /etc/sysconfig/network

Hostname = wisdom.com

vi /etc/hosts

192.168.1.254 station254.wisdom.com

→ open the first configuration file

#vi /etc/named.caching-nameserver.conf

87

options {

listen-on port 53 { 127.0.0.1; 192.168.1.254; }; # Line no:15

allow-query { localhost; 192.168.1.0/24; }; # Line no:23

match-clients { localhost; 192.168.1.0/24; }; # line no:32

→ open the second config file:

vi /etc/named.~~caching-nozone~~ rfc1912.zones

go to last line:

```
zone " wisdom.com" IN {
        type master;
        file " wisdom.for";
        allow-update {none;};
};

zone "1.168.192.in-addr.arpa" IN {
        type master;
        file " wisdom.rev";
        allow-update {none;};
};
```

→ goto zones Location:

cd /var/named/chroot/var/named ↵

→ configure the FLZ:

cp localhost.zone wisdom.for ↵

vi wisdom.for

```
$TTL 86400
@   IN   SOA   station254.wisdom.com.   root.wisdom.com. (
                                        :
            IN      NS      station254.wisdom.com.
254         IN      ~~PTR~~ A       88station254
```

#Vi wisdom.for ↵

```
$TTL 86400
@       IN      SOA     Station254.wisdom.com. root.wisdom.com. (
                                42      ; Serial
                                3H      ; Refresh
                                15m     ; Retry
                                1w      ; Expiry
                                1D)     ; Minimum
        IN              NS      Station254.wisdom.com.
Station254      IN              A       192.168.1.254
Station240      IN              A       192.168.1.240
www             IN              CNAME   Station254
```

→ Configuration the RLZ:

#cp named.local wisdom.rev

#vi wisdom.rev

```
$TTL 86400
@       IN      SOA     Station254.wisdom.com. root.wisdom.com. (
                1997022700 ; Serial
                    28800 ; Refresh
                    14400 ; Retry
                  3600000 ; Expire
                    86400 ) ; Minimum
        IN      NS      Station254.wisdom.com.
254     IN      PTR     Station254
240     IN      PTR     Station240
```


89

→ Change the ownership of zone files:

```
# chown named wisdom*
```

→ Add the DNS ip address:

```
# vi /etc/resolv.conf

nameserver 192.168.1.254
```

→ To check the test configuration files:

```
#named-checkconf /etc/named.caching-nameserver.conf
#named-checkconf /etc/named.rfc1912.zones
```

→ To check the errors at zone files location

```
#named-checkzone wisdom.com /var/named/chroot/named/wisdom.fwd
#named-checkzone wisdom.com /var/named/chroot/named/wisdom.rev
```

→ Restart the service:

```
# service named restart
```

→ To test the DNS:

(i) To test the DNS with hostname

```
# dig Station240.wisdom.com
```

(ii) To test with ip-address

```
#dig -x 192.168.1.254
```

→ Test at NSLookup prompt:

```
#nslookup
> station254.wisdom.com
> Exit
```

→ Test the pinging with hostname:

```
# ping <hostname>
#ping station254.wisdom.com
```


90

client side:

vi /etc/resolv.conf

nameserver 192.168.1.254 #add DNS ip address

Digging:

#dig station254.wisdom.com

#dig -x 192.168.1.254

dig: dig is a slightly more flexible tool, providing the ability to trace DNS queries and to use keys for verifying transactional signatures (TSIGs).

nslookup: which has been the standard DNS query tool in UNIX and Linux for years. It has both an interactive and non-interactive mode which include a number of useful features. See man nslookup for more options.

—x—

* SLAVE DNS *

primary DNS: It is the master copy of all zones information. It is read/write copy.

Slave DNS: It is backup of master's zones. It is read only copy.

→ configure the primary DNS:

→ go to zone location #cd /var/named/chroot/var/named

#vi wisdom.for

add IN NS station240.wisdom.com
(A)

#vi wisdom.rev

add 240 IN NS station240.
(PTR)

→ Restart the service

#service named restart ↵

→ Configure the slave DNS:

→ Install the packages

#rpm -ivh bind* caching* --force --aid ↵

yum install bind* caching* ↵

→ open the first configuration file:

#vi /etc/named.caching-nameserver.conf

Listen-on port 53 {127.0.0.1; 192.168.1.240; }; #line no 15:

allow-query {localhost; 192.168.1.0/24; }; #line no 23:

match-clients {localhost; 192.168.1.0/24; }; #line no 32;

→ open the second config file:

#vi /etc/named.rfc1912.zones

goto last line:

```
zone "wisdom.com" IN {
type slave;
file "slaves/wisdom.for.backup";
masters {192.168.1.254; }; #master DNS ip
zone "168.192.in-addr.arpa" IN {
type slave;
file "slaves/wisdom.rev.backup";
masters {192.168.1.254; }; #master DNS ip
};
```

→ create a directory under zones location:

#mkdir /var/named/chroot/var/named/slaves ↵

→ Add the DNS ip address:

#vi /etc/resolv.conf

```
nameserver 192.168.1.254
nameserver 192.168.1.240
```

92

Digging:

```
#dig station254.wisdom.com
#dig station240.wisdom.com
#dig -x 192.168.10.254
#dig -x 192.168.10.240
```

```
#nslookup ↵
> station254.wisdom.com
> exit
```

→ Test with pinging:

```
#ping station254.wisdom.com
#ping station240.wisdom.com
```

Serial (42): Based on this serial number only, the zone transfer between master DNS and slave DNS will be occured. Generally the serial no will be displayed in (YYYYmmdd s.no) 2007092201

Refresh (3H): for every 3 hours the secondary DNS contacts the primary DNS.

Retry (15m): Retry every 15 minutes. If failure of the previous request occures.

Expiry: This value is the max limit the information stored in the primary.

Minimum (1D): Storage of information in one day in cache DNS.

—x—

93

WEB SERVER

→ By using web server we can host the webpages.

→ No. of web servers are available

IIS : Microsoft

Apache : For all platforms

Tomcat, weblogic : Third party Tools

Apache:

In the year 1995 "httpd" daemon was popular to host a webserver. This was developed by NCSA (National Centre for Super Computing Applications).

→ Apache is a freeware and is the most popular and widly used webserver, which consumes 60% of webmarket that can be configured in both windows & Linux.

HTTP:

Requirments:

- packages : http....rpm
- port no : http - 80
- config file : /etc/httpd/conf/httpd.conf
- Service : httpd

Note:

Configure the DNS:

Remember that you will never receive the correct traffic unless you have configured DNS for you domain to make your new Linux box web server the target of the DNS domain's www entry. See either the Static DNS or Dynamic DNS pages on how to do this.

94

⇒ Configure the web Server:

→ Install the package:

#rpm -ivh http* --force --aid (or)

#yum install http* -y

→ open the configuration file:

#vi /etc/httpd/conf/httpd.conf

Go to last line:

```
<VirtualHost *:80>
    ServerAdmin root@Station254.wisdom.com
    ServerName Station254.wisdom.com
    DocumentRoot /var/www/html
    DirectoryIndex file.html
</VirtualHost>
```

→ Go to website location:

#cd /var/www/html

#vi file.html

here we can write html code.

→ Restart the service:

#Service httpd restart

→ open the webbrowser:

CLI Based: #elinks ↵

GUI Based: #firefox ↵

→ Type URL: http://Station254.wisdom.com

(or)

www.wisdom.com

95

⇒ Authentication of the user:

→ open the config file:

```
#vi /etc/httpd/conf/httpd.conf
```

→ go to lastline:

```
<Directory /var/www/html>
    AuthUserFile /etc/httpd/conf/.htpasswd
    AuthName "web authentication"
    AuthType Basic
    Require valid-user
</Directory>
```

→ Now create the user & assign the http password:

```
#useradd user1
#htpasswd -c /etc/httpd/conf/htpasswd user1
```

→ Restart the service:

```
#service httpd restart
```

→ Now open the web browser:

```
#elinks (or)
#firefox ↵
```

→ Type URL:

```
http://www.wisdom.com
        (or)
http://station254.wisdom.com
```

Now type the Authenticated user name and password in the dialouge box.

—x—

96

* VIRTUAL HOSTING *

→ Virtual Hosting is used to host multiple websites on a single machine with one or more public IP address.

→ There are three types of Virtual Hosting

(a) Name Based.

(b) IP Based.

(c) port Based.

(a) Name Based: You can make your webserver host more than one site per IP address by using Apache's "Named Virtual Hosting".

→ Install the package:

```
#yum install httpd* -y
```

→ Open the config file:

```
#vi /etc/httpd/conf/httpd.conf
```

→ Go to the line 972 remove # :

→ Go to last line:

```
<VirtualHost *:80>
        ServerAdmin root@station254.wisdom.com
        ServerName station254.wisdom.com
        DocumentRoot /var/www/html
        DirectoryIndex file.html
</VirtualHost>
<VirtualHost *:80>
        ServerAdmin root@station254.india.com
        ServerName station254.india.com
        DocumentRoot /var/www/html
        DirectoryIndex file.html
```

97

=> (ii) IP Based: The other virtual hosting option is to have one IP addresses per website which is also known as IP based virtual hosting.

→ In this case you will not have a NameVirtualHost directive for the IP address and you must only have a single <VirtualHost> section per IP address.

→ Create the virtual IP:

→ go to location

```
# cd /etc/sysconfig/ifcfg-eth0 ↵
# cp ifcfg-eth0  ifcfg-eth0:0
# vi ifcfg-eth0:0
        DEVICE = eth0:0
        ONBOOT = YES
        IPADDRESS = 192.168.1.253
# Service network restart
```

→ Open the config file:

```
# vi /etc/httpd/conf/httpd.conf
```

go to Last Line:

```
<VirtualHost 192.168.1.253:80>
        ServerAdmin root@Station254.wisdom.com
        ServerName Station254.wisdom.com
        DocumentRoot /var/www/html
        DirectoryIndex file.html
</VirtualHost>
```


98

(iii) port Based:

→ In this we can host more than one website which works on different portno's.

```
Listen 8000
    <VirtualHost 192.168.1.254 :8000>
        ServerAdmin root@Station254.wisdom.com
        ServerName Station254.wisdom.com
        DocumentRoot /var/www/html
        DirectoryIndex file.html
    </VirtualHost>
```

→ go to website location:

```
#cd /var/www/html
#vi file.html
    write html code
```

→ Now restart the service:

```
#Service httpd restart
#Chkconfig httpd on↵
```

→ Open the web browser:

```
#elinks (or) #firefox ↵
```

→ To testname based: http://Station254.wisdom.com
http://Station254.india.com

→ To test Ip based: http://192.168.1.253

→ To test port based: http://Station254.wisdom.com:8000↵

—x—

99

* TCP WRAPPER *

→ TCP Wrappers are used to restrict the Systems to access the services.

(or)

→ TCP wrappers gives the possibility to control and protect the network services, limiting the access access and registering all the connections to make the work of detecting and resolving problems easier.

→ To setup TCP wrappers you work with two access control text files.

→ They are: (a) /etc/hosts.allow

(b) /etc/hosts.deny

(a) /etc/hosts.allow: This file is used to give the permissions to access the service.

(b) /etc/hosts.deny: This file is used to deny the permissions to access the service.

Syn: deamon-List : client-List [: options]

→ Here we can use two keywords.

(i) ALL: This keyword is used to specify all the deamons (or) clients.

(ii) EXEPT: To specify only to a perticular deamon.

Examples:

→ To deny the System 192.168.1.254 to access 192.168.1.240 through SSh:

```
# vi /etc/hosts.deny
sshd : 192.168.1.240
```

→ To restrict all the hosts in wisdom.com:

```
# vi /etc/hosts.deny
sshd : *.wisdom.com
```

→ To restrict all the hosts in wisdom.com except station241.wisdom

#vi /etc/hosts.deny

sshd : *.wisdom.com EXEPT station241.wisdom.com

→ To restrict all TCP wrappered services to every one:

#vi /etc/hosts.deny

ALL : ALL

→ The above example denies access to all TCP wrappers services to every one except those who are explisitily allowed

→ It allows all the systems in the N/w 192.168.1.0:

#vi /etc/hosts.allow

vsftpd : 192.168.1.0

→ Allows hosts in the wisdom.com to access telnet, ssh:

#vi /etc/hosts.allow

telnet,sshd : .wisdom.com

—x—

* Telnet: Telnet is a program that allows users to log into your server and get a command prompt just as if they were logged into the VGA console.

→ Telnet is installed and enabled by default on Redhat Linux.

Disadvantage: Telnet is that the data is sent as clear text. This means that it is possible for someone to use a network analyzer to peek into your data packets and see your username & password.

101

→ A more secure method for remote loging would be via Secure shell (SSh) which uses varying degrees of Encryption.

→ To login the Server using telnet

Syn: #telnet <server ip>

#telnet 192.168.1.254 ↵

* ~~SSH of Secure Shell~~ → configuration of Telnet:

#vi /etc/xinetd.d/krb5-telnet

disable = no [Defaultly disable = yes]

Service xinetd restart

#chkconfig telnet on

* SSH (Secure Shell):

→ Secure Shell is a replacement for telnet.

→ Redhat Linux comes standard with Secure shell installed.

→ SSH provides an Encrypted data Stream for you to use when you login from one machine to another.

→ When logging in from another Linux/Unix machine you use SSh cmd.

→ Testing to see if SSH is Running (or) not:

#pgrep sshd ↵

→ To see the configuration:

#vi /etc/ssh/sshd_config

→ By default SSH listens on all your NICs and uses TCP port 22.

→ To Restart the Service:

Service sshd restart

#chkconfig sshd on

* PROXY SERVER *

→ proxy server controls client computers access to the Internet.

→ It blocks the users from accessing undefinable website and hides the internal identify of the network.

→ It improves performance by storing webpages only.

→ It is generally used to share internet connection to more than one client

→ Squid proxy is widly used as proxy because it provides many features and mainly it is a open source.

Requirments:

packages: squid*...rpm

port no: proxy - 3128

icp - 3130 (Internet cache protocol)

config file: /etc/squid/squid.conf

Service: squid

Daemon: squid

Syntax:

{ acl <acl-name> <keywords> <requirments>
  http_access <aclname> <deny/allow> }

Configuration:

→ Install the packages:

#rpm -ivh squid* --force --aid ↵

(or)

#yum install squid* -y

→ open the config file:

#vi /etc/squid/squid.conf ↵

103

→ Now Mention the Rules in config file:

Below the Line no 627:

```
acl Mynetwork src 192.168.1.0/24
http_access allow Mynetwork
```

```
acl badsite url_regex yahoo.com
http_access deny badsite.
```
} To restrict particular site

```
acl restrictedhost src 192.168.1.241
http_access deny restrictedhost
```
} To restrict particular machine.

```
acl blocksites url_regex "/etc/squid/squidblock.com"
http_access deny blocksites
```

#vi /etc/squid/squidblock.com

youtube.com
gmail.com
Torrents.com

```
acl mornings time 08:00-12:00
http_access allow mornings
```

→ Restart the Service:

#service squid restart ↵

Clientside: → open the browser:

FOR GUI: #elinks & ↵ press 'esc' key

Setup → optionsmanager → expand the protocols http → proxy configuration → edit the hostname and portno → put the IP address: portno of proxy server → ok → close

FOR GUI: → open the browser:

Edit → preferences → connection settings → general → Select the radiobutton ⊙ manualproxy configuration → put in IP & portno → Ok → close

104

* DHCP *

(Dynamic Host Configuration protocol)

Types of IP address:

1. Static IP address: Addresses that are manually assigned and do no change over time.
2. Dynamic IP address: Addresses that are automatically assigned for a specific period of time and might change.

Boot p Server: In this method the administrator used to collect the MAC address of the system and assign corresponding IP address to them. The list of MAC address and IP address was maintained in a server called as Boot p Server. when ever the client request for IP address Boot p server assign the IP. In this method the administrator used to collect the MAC address.

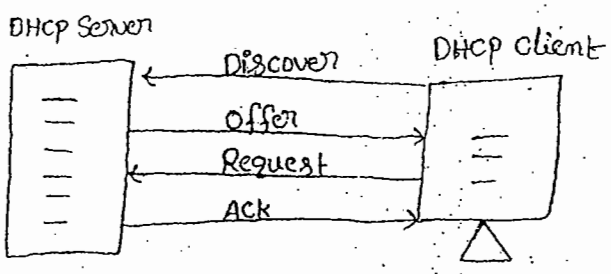
DHCP:

→ It gives IP addresses automatically to the clients who is requesting for an ipaddress.

→ It provides Centralized IP address management

→ DHCP reducess the complexity and amount of administrative work by assigning TCP/IP configuration.

→ DHCP follows the DORA process.



105

Requirements:

packages: dhcp*

port no: 67 - Bootp
68 - Dhcp

Config file: /etc/dhcpd.conf

Service: dhcpd

Deamon: dhcpd

Configuration at Serverside:

→ Install the package

#rpm -ivh dhcp* --force --aid (or)

#yum install dhcp* -y

→ Copy the sample configuration file to main config file

#cp /usr/share/doc/dhcp-3.0.5/dhcpd.conf.sample /etc/dhcpd.conf

→ open the configuration file

#vi /etc/dhcpd.conf

Subnet 192.168.1.0 netmask 255.255.255.0 #line 4

range dynamic-bootp 192.168.1.50 192.168.1.100 #line 21

→ Restart the Service:

#Service dhcpd restart

Client side:

→ Connect to the DHCP Server

#neat-tui (or) #Setup

Select DHCP option ⇒ ok ⇒ save ⇒ quit

106

DHCP Reservation: Assiging ip address dynamically has some problem that every time a client system boots it is not sure that it will get the same ip, so it will be tedious task for other systems to find the particular system. To solve the above problem we can do MAC address binding of the ip address for this provide its entity in the fixed address portion.

→ How to assign fixed ip address:

→ open the configuration file

```
#vi /etc/dhcpd.conf
```

Go to last line

```
host station254 {
        netxt-server station254.wisdom.com;
        hardware ethernet 00:11:1D:F3:26:37;
        fixed address 192.168.1.80;
};
```

→ Restart the service

```
#Service dhcpd restart
```

client side:

→ Restart the service

```
#Service network restart
```

→ check the ip address:

```
#ifconfig eth0
```

—x—

107

* KICKSTART INSTALLATION *

→ Kickstart allows the installer to read information from a designated file rather than prompting the person doing the installation.

→ It especially facilitates the setup of a number of machines that have similar hardware that the installer can autoprobe successfully.

→ Anaconda automatically generates a kickstart file during installation and saves it under /root/anaconda-ks.cfg. This file can be used as a basic for disaster recovery of this same machine (or) as a starting point for your own custom installs.

→ "System-config-kickstart" is a graphical tool for creating and modifying kickstart files.

→ We can install the O.S in two ways.

(i) Standalone Installation.

(ii) Network installation.

→ In N/w installation we can install 3 ways.

(i) NFS

(ii) FTP

(iii) KICKSTART

Requirements:

(i) O.S-Dump

(ii) pykickstart and System-config-kickstart packages.

(iii) NFS (or) FTP service

(iv) DHCP service.

→ Configure the kickstart through FTP service:

→ Install the ftp package and copy O.S.Dump to default shareble location of ftp service (i.e /var/ftp/pub)

→ mount the DVD.

#mount /dev/dvd /mnt

#rpm -ivh vsftpd* --force --aid ↵

#cp -rvf /mnt/* /var/ftp/pub ↵ here '*' means all the data in the mountpoint /mnt

→ Install the kickstart package:

#rpm -ivh pykickstart* system-config-kickstart* --force

(or)

#yum install pykickstart* system-config-kickstart* -y ↵

→ Now open the kickstart dialogue-box

#System-config-kickstart &

→ After configure of the above dialogue box save the file with name ks.cfg under the directory /var/ftp/pub.

→ To give full permission to ks.cfg file

#chmod 777 ks.cfg

→ Configure the DHCP Service:

→ Restart the FTP & DHCP Services:

Clientside configuration:

Step1: Boot the pc with Linux bootable DVD.

Step2: At the boot prompt type

Use FTP: boot: Linux ks=ftp://192.168.1.254/pub/ks.cfg

Use NFS: boot: Linux ks=nfs://192.168.1.254:/var/ftp/pub/ks.cfg

—x—

109

110

SHELL SCRIPTING

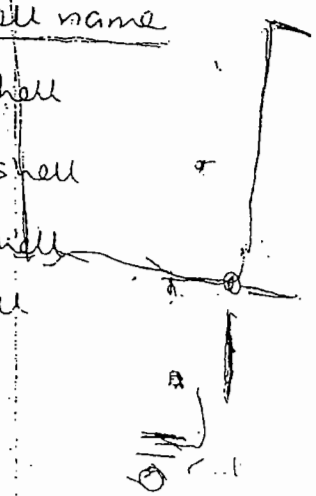
* It is a group of unix commands and shell keywords.
* The main concept of shell scripting is
  * To handle text files i.e It has given very rich text processing features and text utilities.
  * It saves lot of work time
  * To create new unix commands.

Different types of shells

Shell name	Developed By	Prompt	Interpreter name
1. Bourne shell	Stephen Bourne	$	sh
2. Korn shell	David Korn	$	Ksh
3. C shell	Bill Joy	%	Csh
4. Bash shell (Bourne Again shell)		$	bsh (or) sh
5. Z shell	Paul	$	ZSh

* The advanced version of Bourne shell is Bash shell

Flavour name	Default shell name
1. Linux, Macintosh	Bash shell
2. SCO-unix, Sun solaris, HP-UX	Bourne shell
3. IBM-AIX	Korn shell
4. IRIX (silicon Graphics)	C shell



* How to see Parent shell of current user?

```
$echo $SHELL ↵
  /bin/bash
```


* What is the location shell ↵

  /bin

* How to view available shells? (2)

$cd /bin ↵

$ls *sh ↵

(or) $cat /etc/shells

* how to shift to Korn shell ↵

$Ksh ↵

* How to shift to Bourne shell ↵

$Sh ↵

* How to see current child shell (or) sub shell ↵

$echo $0 ↵

* write a shell script to display todays date, present working directory, No. of users connected and calender.

$vi welcome ↵

```
clear
date
Pwd
who | wc, -l
cal

:wq
```

$sh welcome ↵

(or) $chmod 755 welcome ↵

$./welcome ↵

How to execute shell script

$sh scriptname

(or) $chmod 755 scriptname

$./scriptname ↵

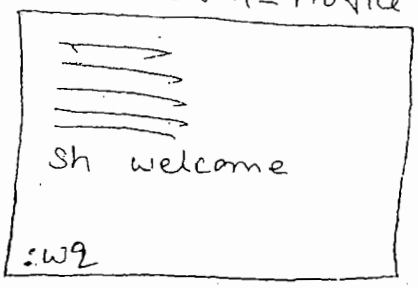
Note :- extension is optional for shell scripts.

* what is meant by Profile file (or) Startup file?

- bash_Profile is a predefined file, it executes at the time of login

Note :- In Sunsolaris, the name of the file is .Profile

$vi .bash_Profile ↵

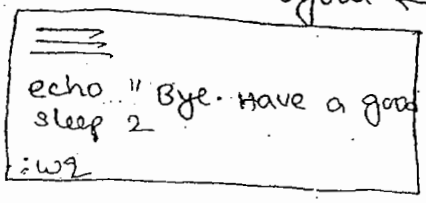


(3)

* what is meant by logout file?

.bash_logout is a predefined file, it executes at the time of logout.

$vi .bash_logout ↵



* echo :- It is a shell keyword. It is used for to write data to the output screen.

Eg:- ① $echo Tecno ↵
Tecno

② $echo welcome to Tecnosoft ↵
welcome to Tecnosoft.

③ $echo Today date is : date ↵
Today date is : date.

command substitution operator
`commandname` (or) $(commandname)

→ It is known as backquote (or) backticks

④ $echo Today date is : `date` ↵
Today date is : Sun 01 08 12:20:21 IST 2010

⑤ $echo -e "Hello \n Tecno"
Hello
Tecno

113

⑥ $echo -e "\t Tecno\b\bsoft" ④
Tecsoft

* Variables

1. It is a dataname (or) memory location name.
2. It is a temporary storage location.
3. It value can change during execution of the program
4. No datatypes in shell scripting.
5. Each and Every variable, it treated as string variable

Variables are classified into 2 types

1. User defined variables
2. system defined variables (or) environment variables.

User defined variables :- It is classified into 3 types

1. Local variables
2. Constant variables
3. Global variables

Local variables Examples

```
a=100
b=1.5
course=unix
name="Tecno soft"
```

① $echo a
a

② $echo $a
100

③ n=a
echo $n
a

④ n=$a
echo $n
100

⑤ K=ls
echo $K
ls

⑥ K="ls"
echo $K
==

⑦ C="cat emp"
echo $C
= 114

Variable substitution operator

$variablename
(or)
${variablename}

⑧ mypath="/home/tecno/a"
cd $mypath

(9) x = "Tecno"
x = $xsoft
$echo $x (It prints empty value)
$

(10) x = "Tecno"
x = ${x}soft
$echo $x
Tecnosoft
$

(5)

Note :- To add some text to existing variable, use { }.

Constant variables :- "readonly" is the keyword to create constant variables.

x = 100
readonly x.

Global variables :- "export" is the keyword, to create global variables.

How to take input from user :- "read" is the keyword to take input from user.

Syntax: read variablename

How to take input from user with prompt

read -p "prompt :" variablename

Eg:- (1) $read -p "Enter a name :" name ↵
Enter a name : Tecnosoft

(2) $read -s -p "Enter a password :" name ↵
Enter a password: ↵

System defined variables :- "set" is the command, to see all system defined variables along with its values.

$ set ↵
HOME = /home/tecno
SHELL = /bin/bash
LOGNAME = tecno

115

```
PATH =
MAIL = /var/spool/mail/tecno
MAILCHECK = 60
PS1 = $
PS2 = >
```

* How to change system prompt ?

PS1 = "Tecno $"

* How to change Input prompt ?

PS2 = " ---> "

* PATH, system defined variable is used for to set application software paths like Java, oracle, oracle apps.

Operators

1. Arithmetic operators

+
-
*
/
%

2. Relational operators

a) Numeric comparision operators

-lt (less than)
-le (less than or equal to)
-gt (greater than)
-ge (greater than or equal to)
-eq (equal to)
-ne (not equal to)

b) String comparision operators

<
>
=
!=

3. Logical operators

-a (logical and)
-o (logical or)
! (logical not)

4. Assignment operator

=

Note :- Each and Every operator, Should contain space before and after operator except assignment operator.

```
a=10
b=4
$echo $a + $b ↵
10 + 4
```

expr :- expr is the keyword to convert string expression to integer expression.

Eg:① `$echo ` expr $a + $b ` ↵`

14

1) expr integer expression
   Eg:- `expr $a + $b`
2) $((integer expression))
   Eg:- $(($a + $b))
3) let integer expression
   Eg:- let c=$a+$b

② `$echo ` expr $a * $b ` ↵`

40

Float arithmetic

```
a=10.4
b=3.2
$echo $a + $b
10.4 + 3.2
$echo `echo $a + $b | bc`
13.6
$echo `echo $a - $b | bc`
7.2
```

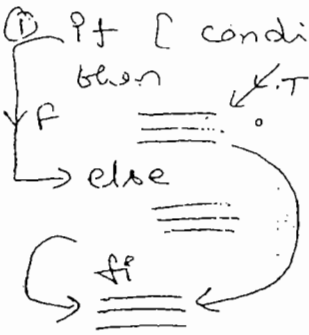
Note :- bc is the command, to perform any float related calculations.

$bc ↵ (binary calculator)
10+4 ↵
14
10.3+4.2 ↵
14.5
10>4 ↵
1
10<4 ↵
0
10/4 ↵
2
sqrt(20) ↵
14
scale=2
sqrt(20)
14.14
10/4

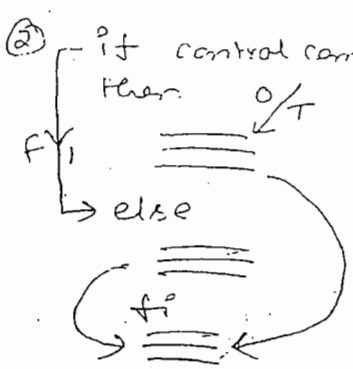
117

if

(1) if [condition]
then
F ↙ T
else
fi



(2) if control command
then
F ↙ T
else
fi



file test commands

1) -f ⇒ TRUE, if it is regularly file
2) -d ⇒ TRUE, " " " directory file
3) -l ⇒ TRUE, " " " link "
4) -b ⇒ TRUE, " " " block special file
5) -c ⇒ TRUE, " " " character " "
6) -e ⇒ TRUE, if file exist
7) -s ⇒ TRUE, if file is not empty.
8) -r ⇒ TRUE, if file has read permission
9) -w ⇒ TRUE, if file has write permission
10) -x ⇒ TRUE, if file has execute permission.

String test commands

1) -z ⇒ TRUE, if string is empty
2) -n ⇒ TRUE, if string is not empty.

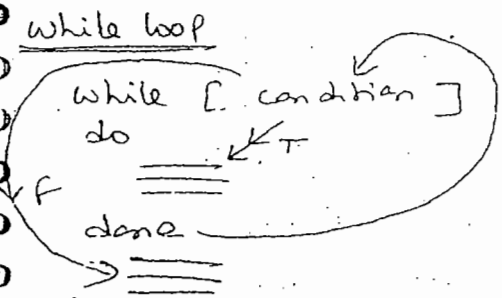
is a single line comment.

118

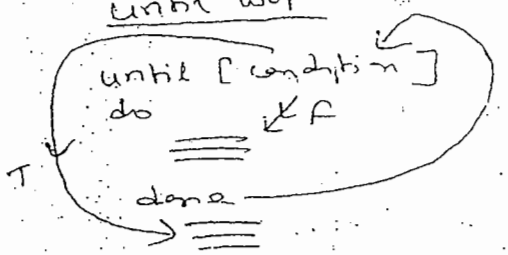
case

```
case variable in
Pattern 1) ===
           ;;  # to terminate the case
Pattern 2) ===
           ;;
   :
Pattern n) ===
           ;;
*)  ===
    ;;
esac
```

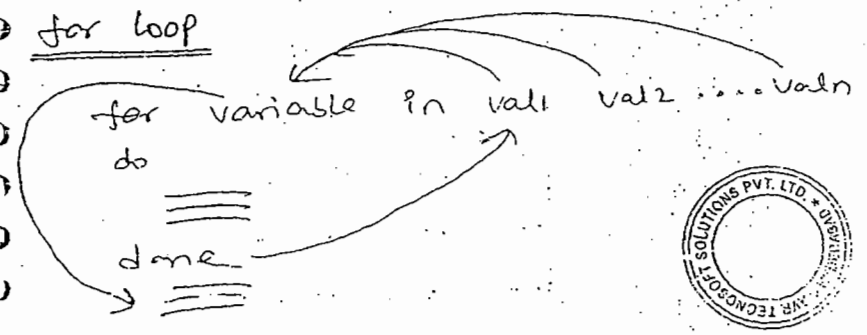

while loop



until loop



for loop



break :- break is the keyword, to terminate the loop

```
while [ condition ]...
do
  ===
  break
  ===
done
```


119

Continue :- Continue is the keyword, to start the loop again

```
while [ condition ]
do
  ==
  continue
  ==
done
  ==
```

** → The 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 referring.

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 megabyte	1000 kilo = 1 mega
1024 mega = 1 Gigabyte	1000 mega = 1 giga
1024 giga = 1 Terabyte	1000 giga = 1 Tera
1024 tera = 1 petabyte	1000 Tera = 1 peta
1024 peta = 1 Exabyte	1000 peta = 1 Exa
1024 Exa = 1 zettabyte	1000 Exa = 1 zetta
1024 zetta = 1 yottabyte	1000 zetta = 1 yotta
1024 Yotta = 1 Brontobyte.	1000 yotta = 1 Bronto
1024 Bronto = 1 Geopbyte	1000 Bron = 1 geophy

120

TECNOsoft solutions

Comunication commands

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

$write username/terminalname ↵
message
ctrl d

Note:- Default mesg is "y"

tecno1	tecno2
$write tecno2 ↵	
Hello	message from tecno1
ctrl d →	Hello
message from tecno2	$write tecno1 ↵
Hi ←	Hi
	ctrl d
$mesg n ↵ [To deny messages]	$write tecno1 ↵
→	tecno1 user disabled messages.
$mesg y ↵ [To allow messages]	

wall :- It is used for to send broadcast message to all users, whoever connected to server and mesg is y.

$wall ↵
welcome to tecnosoft
ctrl d
$

mail :- To send the mail.

Note $mail username ↵
ctrl d
$

① $mail tecno1 ↵ This mail is transfered to tecno1 user
Subject: Hello ↵ mail box. (/var/spool/mail/tecno1)
Hi, How r u?
ctrl d

$mail tecno1 tecno2 tecno3 ↵ This mail is transfered to
Subject: from Tecnosoft ↵ tecno1, tecno2, tecno3 user
mail boxes.

121 This the contents to tecno

$mail is the command to open mail box.

eg:- $mail ↵

mail no.	sender's name	Date	time	Subject
1 >	tecno	mar 15	8:30	Hello
2 >	hari	--	--	very urgent
3 >	tecno	--	--	shd file
4 >	root	--	--	Hi

& 2 ↵ [It opens 2nd mail]

& 3 ↵ [It opens 3rd mail]

& q ↵ [To quit from mail box]

$

(a)

Mail box options:-

&filename ⇒ It writes the current mail contents to given file.

&r ⇒ To reply.

&p ⇒ To print out

&d ⇒ delete current mail

&d 2 ⇒ deletes 2nd mail

&d 1-10 ⇒ deletes 1st to 10th mail.

Note:- By default all mails will store in primary mail box (/var/spool/mail). All opened mails in primary mail box transfferes to secondary mail box (mbox).

* How to open secondary mail box :- $mail -f ↵
* The primary mail box only maintains unread mails.

Networking commands

① Telnet :- Telnet protocal is in-built in windows/Linux. It is used for to connect to remote servers.

Syntax telnet ipaddress ↵
login: tecno1 ↵
Password: xxxxxx ↵
$

$hostname ↵ It displays servername

$hostname -i ↵ It displays server ipaddress

$uname ↵ It displays o/s name

$uname -r ↵ It displays kernel ve

② ftp :- ftp stands for file transfer protocol.

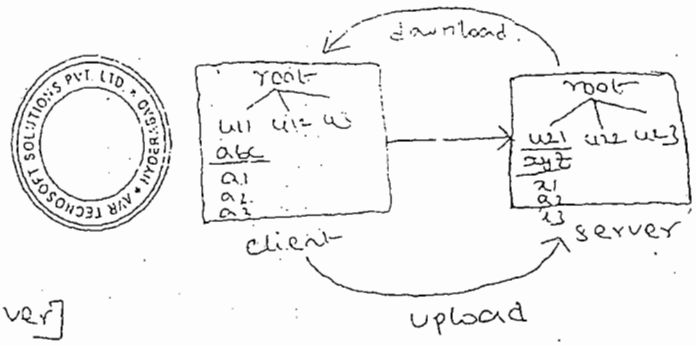
It is used for to transfer files from one server to another server account.

Syn ftp ipaddress ↵
login: _____ ↵
Password: _____ ↵
ftp>

122

Technosoft solutions

ftp commands



ftp > ls ↵ [server]
ftp > !ls ↵ [client]
ftp > pwd ↵ [server]
ftp > !pwd ↵ [client]
ftp > cd dirname ↵ [server]
ftp > lcd dirname ↵ [client]
ftp > get filename ↵ [TO download a file]
ftp > mget file1 file2 .... filen ↵ [TO download multiple files]
ftp > put filename ↵ [TO upload a file]
ftp > mput file1 file2 ... filen ↵ [TO upload multiple files]
ftp > ? ↵ [It list all ftp commands]
ftp > bye ↵ [TO Quit from FTP]

zip files

zip :- It is used for to zip the file
$gzip filename ↵
$gzip sample ↵
sample.gz

gunzip :- TO unzip the file.
$gunzip sample.gz ↵
sample

compress :- TO compress the file
$compress filename ↵
$compress sample ↵
sample.Z

uncompress :- TO uncompress the file
$uncompress sample.Z ↵

Disk status

1. $df ↵ It displays disk space in bytes.
2. $df -h ↵ It displays disk space in kilo bytes
3. $du ↵ It displays directorywise disk usage in form of blocks.

123

2) Background Jobs:

* By default jobs come under foreground.

① $cp file1 file2 ↵ Foreground Job.

② $cp file1 file2 & ↵ Background Job
S12 (PID)

③ $sort sample > a1 & ↵ Background Job.
S13 (PID)

Note: In foreground, user can execute only one job. But in the background user can execute many jobs.

* How to kill foreground Job? ctrl C (or) ctrl Z.

* $PS (or) $PS -f ↵ It displays currently running process list in the current user acco

* $PS -ef ↵ It displays currently running process list in the linux server.

* How to kill background Job? $kill pid (or) $kill -9 pid
↳ sure

* nohup - The nohup jobs will create in server account. So nohup jobs will execute even the user disconnects from his account.

Eg:- $nohup cp file1 file2 & ↵

* $jobs ↵ It displays only background jobs with job id's

* How to bring background job to foreground?

$fg jobid ↵

124

Job scheduling

1) crontab
2) at
3) batch

crontab :- It executes the given jobs repeatedly in server account

crontab -e ⇒ To open the crontab editor

$crontab -e ↵

```
* * * * *  date                  ⇒ Every minute, it executes date command and it writes output into your mail box.
30 10 * * * sh script1           ⇒ It executes script1, every day at 10:30 AM.
30 10 * * 0 sh script2           ⇒ It executes script2 every Sunday at 10:30 AM.
30 22 * * 1,3,5 sh script3       ⇒ It executes script3 every Monday, Wednesday & Friday at 10:30 PM.
30 22 1,15 * * sh script4 > file1 ⇒ It executes script4, every month 1st and 15th at 10:30 PM and it writes output into filed file.
```

(0-59) ⇒ Minutes
(0-23) ⇒ Hours
(1-31) ⇒ Days
(1-12) ⇒ Month
(0-6) ⇒ weekday

0 Sunday
0-S
1-M
2-T
3-W
4-T
5-F
6-S

crontab -r ⇒ To remove all crontab jobs.
crontab -l ⇒ It list all crontab jobs.

at :- It executes the given jobs only once.

$at now ↵ ⇒ It executes the given jobs immediately.
at > ls ↵
at > sh script1 ↵
at > ctrl d

at now + 3 minutes ↵ after 3 minutes
at now + 2 hours ↵ after 2 hours
at now + 3 days ↵ after 3 days
at now + 1 month ↵ after 1 month
at 4pm tomorrow ↵ tomorrow at 4'O clock
at 10:30am July 10 ↵ July 10th at 10:30am

at -l ⇒ It displays all the at command jobs with job id's

batch :- The batch jobs will execute, when server is free and server load is less.

$batch ↵
at > ls ↵
at > ctrl d ↵
$

125

Path: It is the way of representing files & directorys in the system.

→ Two types of paths.

1. Absolute path
2. Relative path

1. Absolute path: It is the way of representing files & directorys from the top of hierarchy is called absolute path.

touch /f1

cp Emp /world/Asia/india/AP/Hyd.

2. Relative path: It is the way of representing files & directorys which are related to current directory.

dir2

Emp wisdom/wisdom1 ↵

→ To create a Nested directory.

#mkdir world/Asia/india/AP/HYd ↵

→ To see the nested tree structure

#tree world ↵

→ 1 Bit Equals to how many bytes?

↳ Bit is the smalest componat of data and byte is longer then bit's

- 8 bits (2^3 bits) = 1 byte
- 1024 byte (2^{10} bits) = 1 kilobyte
- 1024 kilobyte (2^{20} bits) = 1 megabyte
- 1024 megabyte (2^{30} bits) = 1 gigabyte

awk

* The awk utility, which takes its name from the initials of its authors (Alfred V. Aho, Peter J. Weinberger, and Brain W. Kernighan), is a powerful programming language

* How to run awk command

```
$awk 'Pattern {action}' input-file
```

* How to run awk program

```
$awk -f scriptfile.awk input-file
```

* In awk programming, the field numbers begin with one,
  the first field is $1, and
  the second field is $2, and
  the third field is $3, and so on.

* In awk, $0 represents the entire record

* Variables are two types
  a) System variables
  b) User defined variables.

* System Variables

Variable	function	Default
FS	Input field separator	space or tab
RS	Input record separator	newline
OFS	output field separator	space or tab
ORS	output record separator	newline
NF	Number of nonempty fields in current record	
NR	Number of records read from all files	
FILENAME	Name of the current file	

* User-defines variables
  1. Variable should begin with an alphabet
  2. It shouldn't contain any space character

127

* Structure of awk Program

BEGIN {Begin's Action} — Preprocessing

Pattern {Action}
Pattern {Action} — Body
Pattern {Action}

END {End's Action} — Post-processing

* Expressions

* Regular Expressions :- The awk regular expressions are those defined in egrep. In addition to the expression, awk requires one of the two operators.

(a) match (~)
(b) not match (!~)

Note:- when using a regular expression, remember that it must be enclosed in /slashes/.

Eg:- $0 ~/^A.*B$/ # Record must begin with A and end with B

$3 !~/^Unix/ # Third field must not start with a unix

$4 ~/unix$/ # Fourth field must end with unix.

Eg:- $awk '$0 ~/unix/ {print $0}' stud

Operators

Arithmetic operators :- +, -, *, /, %, ++, --

Relational operators :- <, >, <=, >=, ==, !=

Logical operators :- &&, ||, !

Assignment operators :- =, +=, -=, *=, /=, %=

Note: Each and every operator must contain space before and after operator

Eg:- (1) $awk '{print}' stud ← It prints stud file contents

(2) $awk '{print $1, $2, $4}' ← It prints 1st, 2nd & 4th fields from stud file.

(3) $awk 'BEGIN {OFS="\t"}; {print $1, $2, $4}' stud | head -5

It prints the first 5 records of 1st, 2nd & 4th fields

④ $awk '$4=="unix" {print $2,$3}' stud ↵ ②

It prints only unix students names & Phno's.

⑤ $awk -f";" '$3>=5000 {print $0}' emp ↵

It prints all employee records whose salaries more than 5000.

⑥ $awk -f";" '$4==10 && $3>=5000 {print $0}' emp ↵

It prints all employee records who belongs deptno 10 and salary more than 5000.

If statement

```
if (condition)
{
  ===
  ===
}
```

if else

```
if (condition)
{
  ===
}
else
{
  ===
}
```

while loop

```
while (condition)
{
  ===
}
```

do...while

```
do
{
  ===
}
while (condition);
```

for loop

```
for (initialization; condition; incr/decr)
{
  ===
}
```

129

How to use unix commands in a awk

① BEGIN {
```
"date" | getline
Print ($1, $4)  # Day of week & time of day
}
```

② BEGIN {
```
"who" | getline
Print ($1)  # only usernames who are
            connected to the server
}
```

awk and grep

```
grep 'regular expression' filename
awk '$0 ~/regular expression/ {Print $0}' filename
```

Eg: ① $grep '^[A-Z].*[A-Z]$' file1
```
$awk '$0 ~/^[A-Z].*[A-Z]$/ {Print $0}' file1
```

② $grep -v '^unix' file1
```
$awk '$0 !~ /^unix/ {Print $0}' file1
```

sed and awk


```
sed '=' shd
awk '{Print NR; Print $0}' shd
```
The commands prints shd file records with line numbers.

```
$sed '2,4d' shd
$awk 'NR<2 || NR>4 {Print $0}' shd
```
The commands deletes 2nd line to 4th line from shd file.

130

Vi editor

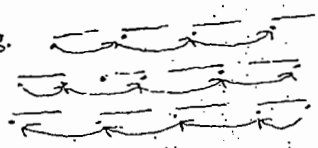
- It is used for to create new files or to modify existing files.
- It is classified into 3 types.
  1. command mode
  2. Input or Insert mode
  3. Ex command mode
- The default mode is command mode.
- Vi is command to open vi editor.
- The following are the commands to shift from command mode to insert mode.
  1. A → It places cursor at end of the current line.
  2. a → It places cursor at right side of the cursor line.
  3. I → It places cursor at begining of the current line.
  4. i → It places cursor at left side of the cursor line.
  5. O → It inserts new line of the cursor.
  6. o → It inserts new line below of the cursor.
- "ESC" is the key to shift from Insert mode to command mode.
- " : " is the command to shift to Ex command mode from command mode.

Command mode commands:

1. k(nk)
   ↑
   h← → l (nl)
   (nh) ↓
   j(nj)

   n means any number.

2. w(nw)→ next word starting.
   e(ne) → word ending.
   b(ne) → word beginning.



3. $ → end of the current line (end key)
   ^ → beginning of the current line(home key)

4. H → Beginning of the current page.
   M → Middle of the current page.
   L → End of the current page.

5. ctrl f → forward on page (pgdn)
   Ctrl b → backward one (pgup)

6. x(nx)→ Delete current character (del key).

131

7. X→ Delete previous character (backspace key)
8. dw(ndw) → Delete current word.
9. dd(ndd) → Delete current line
10. d$ → Delete current position to end of the line.
11. d^ → Delete current position to beginning of the line.
12. yw(nyw) → To copy a word.
13. yy(nyy) → To copy a line.
14. y$ → It copies current position to end of the line.
15. y^ → It copies current position to beginning of the line.
16. p → paste below the cursor
    P → paste above the cursor
17. J → To join a line.
18. cc → To clear a line.
19. u→ undo
20. zz → Save and Quit.

EX command mode commands:

: w → Save without quit.
: w file name → Save with given file name
: q! → quit without save
: wq → Save and quit
: n→ places cursor at nth line.
: $ → places cursor at last line in the file.
: Set nu → Set line numbers.
: Set nonu → To remove line numbers.
: ! <unix command> → Execute unix command.
: /string/ → Top to bottom search. n → next occurence
: ?string? → Bottom to top search. N → previous occurence
: starting line no, ending line no s/old string/new string/gi → Search And Replace string
Eg: 1,$ s/unix/linux/gi

:nd → It deletes nth line
Eg: :3d → It deletes 3rd line
:$d → It deletes last line
:4,7d → It deletes 4th line to 7th line.

: 1,$ s/^/unix → It adds unix at begining of each line
: 1,$ s/$/; → It adds ; at end of each line.

Installing Oracle 10g on RedHat Enterprise Linux 4 AS

1. Login as root.
2. # uname –r /* To check the Kernel Version.*/
3. Check the packages for Oracle on RedHat
   # rpm –q gcc make binutils openmotif setarch compat-db
   The versions should be

gcc	3.4.3-9.EL4
make	3.80-5
binutils	2.15.92.0.2-10.EL4
openmotif	2.2.3-6.RHEL4.2
setarch	1.6-1
compat-db	4.1.25-9

4. # grep MemTotal /proc/meminfo /* To check for available memory. It is advised to have a 512MB+ of memory. */
5. # grep SwapTotal /proc/meminfo /* For checking the SWAP space. The rule is SWAP should be twice the amount of RAM installed on the system. */
6. # df-h /* To check for the Disk Space on the hard drive(s). */
7. Create groups "oinstall" and "dba"
   # /usr/sbin/groupadd oinstall
   # /usr/sbin/groupadd dba
8. Now create the user "oracle".
   # /usr/sbin/useradd –m –g oinstall –G dba oracle
9. Check for the details of the user "oracle":
   # id oracle
10. Change the password for the user "oracle".
    # passwd oracle
    Note: The command is "passwd" and not "password".
11. Make the directories /u01 and /u02
    # mkdir –p /u01/app/oracle /u02/oradata
12. Change the ownerships for the above 2 directories.
    # chown –R oracle:oinstall /u01/app/oracle /u02/oradata
13. Change the access permissions for the above 2 directories.
    # chmod –R 775 /u01/app/oracle /u02/oradata

133

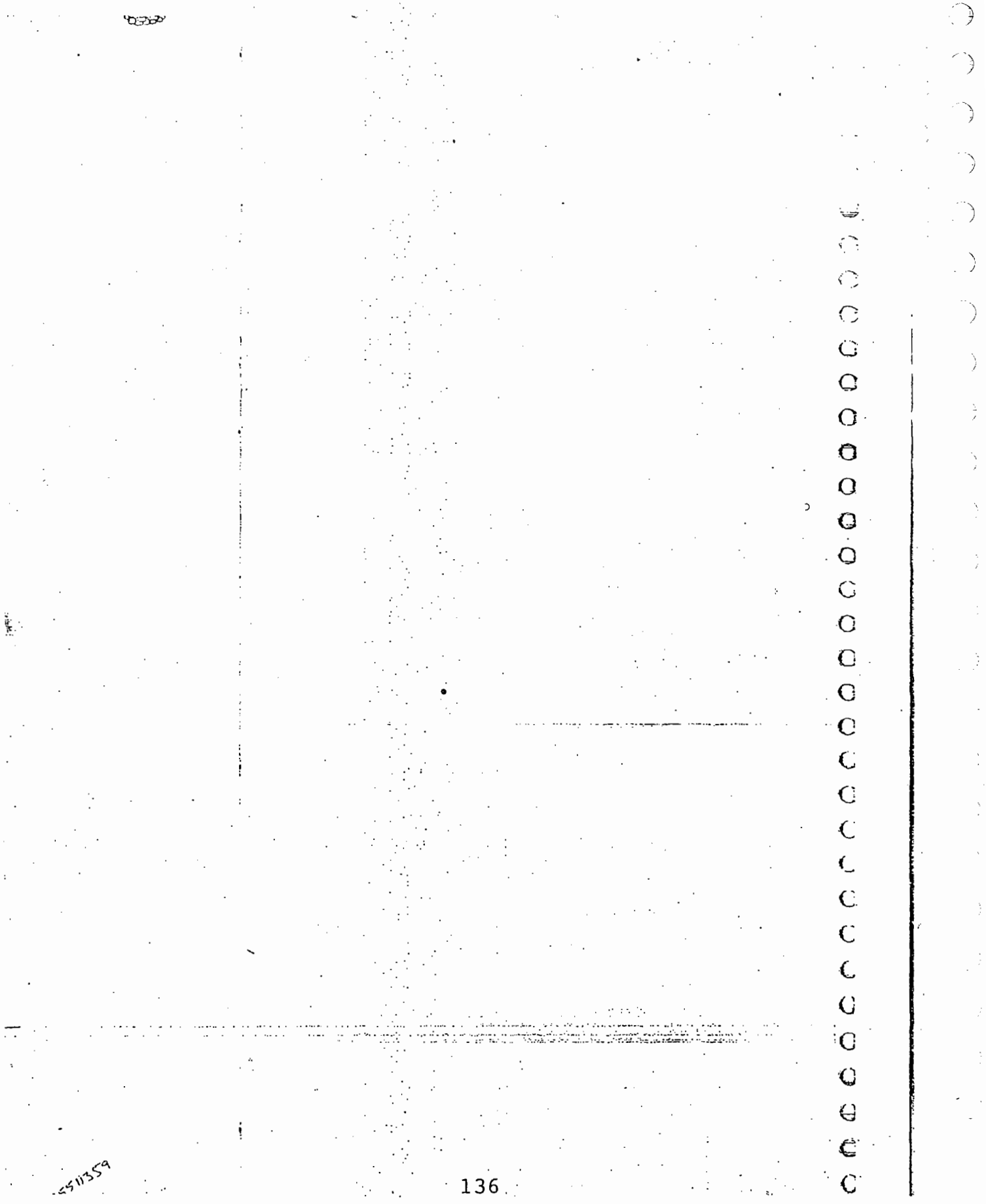
14. Setting the Linux KERNEL parameters:
    # cat >> /etc/sysctl.conf << EOF
    > kernel.shmall = 2097152
    > kernel.shmmax = 2147483648
    > kernel.shmmni = 4096
    > kernel.sem = 250 32000 100 128
    > fs.file-max = 65536
    > net.ipv4.ip_local_port_range = 1024 65000
    > EOF
15. Execute it.
    # /sbin/sysctl -p
16. Log off the "root" user and login as "oracle" user.
17. Setting Shell Limits for the "oracle" user:
    Login in as "root" from the "oracle" login by using, "$ su root"

    cat >> /etc/security/limits.conf <<EOF
    oracle soft nproc 2047
    oracle hard npoc 16384
    oracle soft nofile 1024
    oracle hard nofile 65536
    EOF
18. cat >> /etc/pam.d/login <<EOF
    session required /lib/security/pam_limits.so
    EOF
19. REBOOT
20. Login as "oracle" and set the .bash_profile as:
    ORACLE_BASE=/u01/app/oracle; export ORACLE_BASE
    ORACLE_HOME=$ORACLE_BASE/product/10.1.0/db_1; export
    ORACLE_HOME
    ORACLE_SID=orcl; export ORACLE_SID
    LD_ASSUME_KERNEL=2.4.19; export LD_ASSUME_KERNEL
    PATH=$PATH:$ORALCE_HOME/bin; export PATH
21. Activating the .bash_profile:
    $. .bash_profile
22. For RedHat 4 EL AS, make these changes:
    $ cp /etc/redhat-release /etc/redhat-release.orig
    $ cat >> /etc/redhat-release << EOF
      Red Hat Enterprise Linux AS release 3 (Taroon)
      EOF
23. Installing Oracle:
    a) Download the Oracle CD's from the Oracle.com site.
    b) #gunzip ship.db.cpio.gz
    c) #cpio –idmv < ship.db.cpio
    d) $./runInstaller (or) $sh runInstaller
      To ignore the system prerequisites check, use "-ignoreSysPrereqs".
      The OS group while installing oracle has to be "oinstall".

134

Database Creation Script

```
connect sys/&&password as SYSDBA

set echo on

spool /u01/app/oracle/product/10.1.0/db_1/admin/orcl/create/CreateDB.log

startup nomount pfile="/u01/app/oracle/product/10.1.0/db_1/admin/orcl/scripts/init.ora";

CREATE DATABASE "orcl10g"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 16
MAXLOGMEMBERS 3
MAXDATAFILES 100

DATAFILE '/u02/oradata/orcl10g/system01.dbf' size 300M reuse autoextend on next
10240K maxsize unlimited extent management local

SYSAUX DATAFILE '/u02/oradata/orcl10g/sysaux01.dbf' size 120M reuse autoextend
on next 10240K maxsize unlimited

DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE
'/u02/oradata/orcl10g/temp.dbf' size 300M reuse next 640K maxsize unlimited

UNDO TABLESPACE "UNDOTBS1" DATAFILE
'/u02/oradata/orcl10g/undotbs01.dbf' size 200M reuse next 5120K maxsize unlimited

character set AL32UTF8
NATIONAL CHARACTER SET UTF8

LOGFILE GROUP 1 ('/u02/oradata/orcl10g/REDO01.log') size 10240K,
LOGFILE GROUP 2 ('/u02/oradata/orcl10g/REDO02.log') size 10240K,
LOGFILE GROUP 3 ('/u02/oradata/orcl10g/REDO03.log') size 10240K

USER SYS identified by "&syspassword"
USER SYSTEM identified by "&syspassword";

SPOOL OFF
```



file save to .sh...

sh filename

Shell Scripting Programs

```
#Example of variables
a=10
b=20
echo "a is : $a"
echo "b is : $b"
```

```
#Write a script accept 2 intgere no's and display
echo -e "Enter a number1 : \c"
read a
echo -e "Enter a number2 : \c"
read b
echo "a value is : $a"
echo "b value is : $b"
```

```
#Write a script accept 2 intgere no's and find sum
echo -e "Enter a number1 : \c"
read a
echo -e "Enter a number2 : \c"
read b
c=`expr $a + $b`
echo "Sum of $a and $b is : $c"
```

```
#Write a script accept a filename and open
echo -n "Enter a filename to open:"
read fn
echo "----------o----------"
cat $fn
echo "----------o----------"
```

```
# Write a script accept a filename and delete all empty lines
echo -n "Enter a filename to delete blank lines:"
read fn
grep -v "^$" $fn > temp
mv temp $fn
echo "$fn file empty lines deleted."
```

```
#Write a script accept a filename and delete all duplicate lines:
echo -n  "Enter a filename to delete duplicate lines:"
read fn
sort $fn | uniq -u > temp
mv temp $fn
echo "$fn file Duplicate lines are deleted."
```

```
#write a script accept a number and check the given no is +ve or -ve.
echo -n "Enter a number :"
read n
if [ $n -gt 0 ]
then
echo "$n is a +ve no."
else
echo "$n is a -ve no."
fi
```

```
#Write a script accept a intger no and check the given no is even or odd number
echo -n "Enter a number:"
read n
if [ `expr $n % 2` -eq 0 ]
then
echo "$n is  an Even no."
else
echo "$n is  an Odd no."
fi
```


```
#Write a script accept 2 strings and check the given 2 strings are equal or not
echo -n "Enter a string1 : "
read str1
echo -n "Enter a string2 : "
read str2
if [ "$str1" = "$str2" ]
then
echo "Strings are Equal"
else
echo "Strings are not Equal"
fi
```

```
#Write a script accept a filename and delete given file
echo -n "Enter a filename:"
read fn
if rm $fn
then
echo "$fn file deleted."
else
echo "No such file"
fi
```

```
#Write a script check today is Sunday or not
x=`date +%a` # x=`date | cut -c 1-3`
if [ $x = "Sun" ]
then
echo "Yes.Today is Sunday"
else
echo "Sorry. Today is $x day"
fi
```

```
#Write a script accept a user and check the given user exist or not
echo -n "Enter a username : "
read un
if grep -w $un /etc/passwd > /dev/null
then
echo "$un user exist"
else
echo "$un user doesn't exist"
fi
# /dev/null is a special file.It is used for to write unwanted output.
```

```
#Write a script accept a user and check the user is connect to the server or not.
echo -n "Enter user name:"
read un
if grep -w $un /etc/passwd > /dev/null
then
 if who | grep -w $un > /dev/null
  then
    echo "Logged In"
  else
    echo "Not Logged In"
  fi
else
 echo "$un user doesn't exist",
fi
```

```
#Write a script accept a filename and open
echo -n "Enter a filename : "
read fn
if [ -e $fn ]
then
  if [ -f $fn ]
  then
    if [ -r $fn ]
    then
     cat $fn
    else
    echo "No read permission"
    #chmod 644 $fn
    #cat $fn
    # echo "No read permission"
    fi
  else
   echo "It is not a file"
  fi
else
 echo "$fn file dosen't exist"
fi
```

#Write a script accept a filename and check the file is regular file or directory file

```
echo -n "Enter a filename : "
read fn
if [ -e $fn ]
then
 if [ -f $fn ]
 then
  echo "$fn is a regular file"
 elif [ -d $fn ]
 then
  echo "$fn is a directory file"
 else
 echo "It is not a file or directory"
 fi
else
 echo "$fn file dosen't exist"
fi
```

#Write a script accept 2 filenames and chjeck the given 2 files are same or not

```
echo -n "Enter a filename 1 :"
read fn1
echo -n "Enter a filename 2 :"
read fn2
x=`cmp $fn1 $fn2`
if [ -z "$x" ]
then
echo "Given 2 files are same"
else
echo "Given 2 files are not same"
fi
```

#write a script aceept a string and check the given string is empty or not

```
echo -n "Enter a string : "
read str
if [ -z "$str" ]
then
echo "Given string empty"
else
echo "Given string not empty"
fi
```

---

using case:

```
#case example
echo "enter a number b/w 1 to 4:"
read n
case $n in
1)echo "one";;
2)echo "two";;
3)echo "three";;
4)echo "four";;
*)echo "invalid number";;
esac
```

#Write a script accept a single character and check the given character is alphabet or digit or special character.

```
echo -n "Enter a single character: "
read ch
case $ch in
[a-zA-Z])echo "Alphabet";;
[0-9])echo "Digit";;
[^a-zA-Z0-9])echo "Special Character";;
*)echo "You enetered more than one character";;
esac
```

#write a script accept a single character and check the given character is special character or digit or vowel or consonent

```
echo -n "Enter a single character: "
read ch
case $ch in
[^a-zA-Z0-9])echo "Special character";;
[0-9])echo "Digit";;
[AEIOUaeiou])echo "Vowel";;
[^AEIOUaeiou])echo "Consonent";;
*)echo "You enetered more than one character";;
esac
```

#write a script accept a month and display quarter of the given month

```
echo -n "Enter a month[mon] : "
read mm
case $mm in
jan|feb|mar)echo "1st Quarter";;
apr|may|jun)echo "2nd Quarter";;
jul|aug|sep)echo "3rd Quarter";;
oct|nov|dec)echo "4th Quarter";;
*)echo "Invalid month.";;
esac
#[Jj]an|[Ff]eb|[Mm]ar)echo "1st Quarter";;
#[Aa][Uu][Gg]
#[Aa][Uu][Gg]*
```

```
#Example of Menu Program
clear
tput cup 6 10
echo "MAIN MENU"
tput cup 7 10
echo "*********"
tput cup 8 10
echo "1.Date"
tput cup 9 10
echo "2.List of users"
tput cup 10 10
echo "3.Open a file"
tput cup 11 10
echo "4.delete a file"
tput cup 12 10
echo "5. Exit"
tput cup 20 5
echo "enter a choice[1-5] : "
read choice
case $choice in
1)echo "Today date is : `date`";;
2)who ;;
3)sh fopen.sh;; # ./fopen.sh
4)sh del.sh;; #./del.sh
5)echo "Thank You"
exit ;;     # to terminate the program
*)echo "choice wrong. try again";;
esac
```

```
#Write a script print no's from 1 to 10
i=1
while [ $i -le 10 ]
do
echo $i
i=`expr $i + 1`
done
```

```
#Write a script accept a string and display reverse of the given string
echo -n "Enter a string : "
read str
l=`echo $str | wc -c` # length of the string
while [ $l -gt 0 ]
do
ch=`echo $str | cut -c $l`
temp=$temp$ch
l=`expr $l - 1`
done
echo "Reverse of $str is : $temp"
```

→ TO open 'n' number of files

```
#Example of while loop
ans="y"
while [ $ans = "y" ]
do
echo "Enter a filename to open:"
read fn
if [ -e $fn -a -f $fn ]
then
cat $fn
else
echo "no such file"
fi
echo "Do u want to open one more file [y/n] : "
read ans
done
```

exit for open the file & exit the file

```
#Example of while loop
while true #until false
do
echo "Enter a filename to open:"
read fn
if [ -e $fn -a -f $fn ]
then
cat $fn
break
else
continue
fi
done
```

give correct file name, if not give correct file name asking no. of times Enter a file name

```
#Example of sleep
# sleep is used for to stop the execution specified no of seconds.
while true
do
clear
tput cup 5 8
echo "WELCOME TO"
sleep 2
clear
tput cup 5 8
echo "TECNOSOFT"
sleep 2
done
```

```
#Write a script create "n" no of users
echo -n "Enter no of users to create : "
read n
$i=1
while [ $i -le $n ]
do
$x=tesno$i # It creates users with tecno name.
useradd $x
i=`expr $i + 1`
done
```

```
#Example of for loop
for i in 1 2 3 4 5
do
echo $i
done
```

(or) for i in {1..5}
do
echo $i
done

```
#Example of for loop
a=10
b=20
c=30
for i in a b c
do
echo $i
done
```

```
#Example of for loop
a=10
b=20
c=30
for i in $a $b $c
do
echo $i
done
```

```
#Write a script to display all sub directories of current directory
for i in *
do
if [ -d $i ]
then
echo $i
fi
done
```

-s → non Empty files

```
#Write a script to display all empty files in the current directory
for i in *
do
if [ ! -s $i ]
then
echo $i # rm $i => To delete empty files
fi
```

```
done
```

```
#Write a script to display alll exe files in the current directory
for i in *
do
if [ -f $i -a -r $i -a -x $i ]
then
echo $i
fi
done
```

#Write a script to display details of employees who are receiving salary more than 5000 from emp file

```
#101,hari,9000,10
#102,madhu,4000,20
#103,anu,000,30
#104,priya,8000,10

for i in `cat emp`
do
j=`echo $i | cut -d"," -f 3`
if [ $j -ge 5000 ]
then
echo $i
fi
done
```

→ anything containing bank code (*) is called Command.

#Write a script retrieve details of employees who are receiving salary more than 5000 in deptno 10 from emp file and insert into emp1 file

```
#101,hari,9000,10
#102,madhu,4000,20
#103,anu,666666000,30
#104,priya,8000,10

for i in `cat emp`
do
j=`echo $i | cut -d"," -f 3`
k=`echo $i | cut -d"," -f 4`
if [ $j -ge 5000 -a $k -eq 10 ]
then
echo $i >> emp1
fi
done
```

↓ And function

146

Shell Scripting Examples

```
do
j=`echo $i | cut -d"," -f 3`
k=`echo $i | cut -d"," -f 4`
if [ $j -ge 5000 -a $k -eq 10 ]
then
echo $i >>emp1
fi
done
```

Command line Arguments (or) Positional Parameters

* At the time of execution of shell script, if user passes any argurments known as Command line Arguments (or) Positional Parameters

* The Special variables holds positional parameters values. The special variables are $0,$1,$2,$3,$4,$5,$6,$7,$8,$9,$#,$*,$#,$?,$$
  - $0 => Name of the Program
  - $1 => 1st parameter value
  - $2 => 2nd parameter value
  - $3 => 3rd parameter value
  - $4 => 4th parameter value
  - $5 => 5th parameter value
  - $6 => 6th parameter value
  - $7 => 7th parameter value
  - $8 => 8th parameter value
  - $9 => 9th parameter value
  - $# => Counts no of arguments
  - $* => all parameter values
  - $@ => all parameter values but each and every parameter encloses within double quotes.
  - $? => It holds last executed command status , If the command executed successfully it holds 0(zero) otherwise non-zero value.
  - $$ => It holds user parent shell process id.

$ cd abcd (Executed)

$ cd bacd (not Executed)
0

```
$vi hello
for i in $*
do
echo -n "$i "
done
:wq
$chmod 755 hello
$./hello Tecno soft Solutions
```

3rd parameter (or) arguments

1st parameter 2nd parameter

```
$vi calc
if [ $# -eq 3 ]
then
c=`echo $1 $2 $3 | bc`
echo $c
else
echo "Invalid no of arguments"
fi
:wq
$chmod 755 calc
$./calc 10 + 4
or
$sh calc 10 + 4
```

```
$vi checkuser
if [ $# -eq 1 ]
then
 if who | grep $1> /dev/null
then
echo "Logged In"
else
echo "not Logged In"
fi
else
  echo "Invalid no of arguments"
fi

:wq
$chmod 755 checkuser
$./checkuser tecno
```

#109, Annapurna Block, Aditya Enclave, Ameerpet, Hyderabad.
☎040-66839666 , 9966422225

Database connectivity

1) write a shell script to connect to oracle db.

```
$vi a1.sh ↵
 sqlplus  scott/tiger
 :wq
$sh a1.sh ↵
```

2) write a shell script to insert data into oracle emp table.

```
$vi a2.sh ↵
clear
x=101
y="Hari"
sqlplus -s scott/tiger<<Eof
insert into emp(empno,ename)
values ($x,'$y');
commit;
Eof
:wq
$sh a2.sh ↵
```

3) write a shell script to retrieve data from oracle emp table.

```
$vi a3.sh ↵
sqlplus -s scott/tiger<<Eof
Select * from emp;
Eof
:wq
$sh a3.sh ↵
```

4) write a shell script to call oracle stored procedure

```
$vi a4.sh ↵
sqlplus -s scott/tiger<<Eof
set serveroutput on
 exec square(9)
Eof
:wq
```

output: 81

5) write a shell script to load flat file data into oracle table.

```
$cat > emp ↵
101, Hari, 80000, 10 ↵
102, Sai, 75000, 20 ↵
103, Siva, 60000, 30 ↵
104, Lakshmi, 90000, 10 ↵
ctrl d
$vi a5.sh ↵
 for i in `cat emp`
 do
  a=`echo $i | cut -d"," -f 1`
  b=`echo $i | cut -d"," -f 2`
  c=`echo $i | cut -d"," -f 3`
  d=`echo $i | cut -d"," -f 4`
 sqlplus -s scott/tiger<<Eof
 insert into emp(empno,ename,sal,deptno)
 values ($a,'$b',$c,$d);
 commit;
 Eof
 done
 :wq
$sh a5.sh ↵
```


How to connect to oracle

```
$sqlplus ↵
 username : scott ↵
 password : tiger ↵
SQL> !<unixCommand> ↵
Eg:- SQL>!who ↵
     SQL>!ls ↵
SQL>! ↵ [To return to unix]
$ exit ↵ [To return to SQL]
```

$vi hello Tecnosoft solutions

```
#!/bin/ksh  → this is KSh Script
Print "Hello In"
Print "welcome to Tecnosoft"
```

:wq
$./hello ↵

Note: #! is a shebang statement. It is used for to invoke shell interpreter path.

Functions

It is a group of statements to perform specific task and it returns a value.
The main concept of function is
① To reduce length code of the program
② To reduce cost of maintenance
③ Easy to maintain the application.

Syntax

```
functionname arg1, arg2, .... # calling function

function functionname()
{
==
==
}
```

The function call arguments will store in

$1, $2, $3, ... $9, $#

149

Example of function (12)

① $vi fun1.sh ↵

```
function hello()
{
 echo "welcome to functions"
}
hello
```

:wq

$sh fun1.sh

② write a script to add integer no's using function.

$vi fun2.sh ↵

```
function add()
{
 if [ $# -eq 2 ]
 then
    c=`expr $1 + $2`
    echo "$1 + $2 = $c"
 else
    echo "Invalid arguments"
 fi
}
add 15 2
```

:wq

$sh fun2.sh ↵

150

SHELL SCRIPTING

* It is a group of unix commands and shell keys
* The main concept of shell scripting is
  * To handle text files i.e It has given very rich text processing features and text utilities.
  * It saves lot of work time.
  * To create new unix commands.

* Different types of shells

Shell name	Developed By	Prompt	Interpreter name
1) Bourne shell	Stephen Bourne	$	sh
2) Korn shell	David Korn	$	ksh
3) C shell	Bill joy	%	csh
4) Bash shell (Bourne Again shell)		$	bsh (or) sh
5) Z shell	Paul	$	zsh

The advanced version of Bourne shell is Bash shell

Flavour name	Default shell name
1) Linux, Macintosh	Bash shell
2) SCO-Unix, Sun, Solaris, HP-UX	Bourne shell
3) IBM-AIX	Korn shell
4) IRIX (Silicon Graphics)	C shell

151

```
$ echo $SHELL ↵
/bin/bash
```

what is the location shell ↵

/bin

* How to view available shells?

$cd /bin ↵

$ls *sh ↵

(or) $cat /etc/shells

* How to shift to korn shell ↵

$ksh ↵

* How to shift to Bourne shell ↵

$sh ↵

* How to see current child shell (or) sub shell ↵

$ echo $0 ↵

* Write a shell script to display today's date, present working directory, no. of users connected and calender.

$ vi welcome ↵

```
clear
date
pwd
who| wc -l
cal

:w2
```

$ sh welcome ↵

How to execute shell script

$ sh scriptname

(or) $chmod 755 scriptname

$./scriptname ↵

152

$chmod 755 welcome ↵

$./welcome ↵

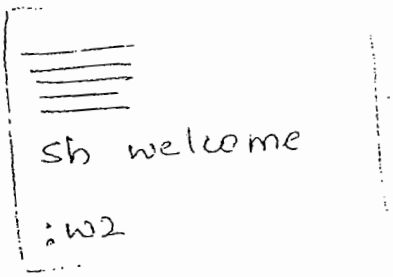
Note: extension is optional for shell scripts

* what is meant by profile file (or) start up file?

. bash-profile is a predefined file, it executes at the time of login.

Note: In sun solari, the name of the file is .profile

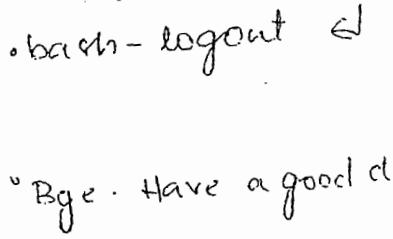
$vi .bash-profile ↵



* what is meant by logout file?

. bash-logout is a predefined file, it executes at the time of logout

$vi .bash-logout ↵



echo: It is a shell keyword. It is used for to write data the output screen.

153

topno

② $ echo welcome to topnotch ↵

welecome to topnotch

③ $ echo Today date is : date ↵

Today date is : date

command substitution operator

`command name` ——→ It is known as back quote (or) backticks.

(or)

$(command name)

④ $ echo Today date is : `date` ↵

Today date is : Sun 25 11 12:20:21 IST 20

⑤ $ echo -e "Hello \n Topnotch"

Hello
top notch

⑥ $ echo -e "top \b\ notch"

topnotch

* Variables

1. It is a data name (or) memory location name
2. It is a temporary storage location
3. It value can change during execution of the program
4. No datatypes in shell scripting
5. Each and Every variable, it treated as string variable

154

1. User defined variables
2. System defined variables (or) environment variables

User defined variables:- It is classified into 3 types

1. Local variables
2. Constant variables
3. Global variables

Local variables Examples

```
a=100
b=1.5
course=unix
name="top notch"
```

i) $echo a ↵
   a

2) $echo $a ↵
   100

3) n=a
   echo $n
   a

4) n=$a
   echo $n
   100

5) k=ls
   echo $k
   ls

6) k='ls'
   echo $k
   ≡

Variable substitution operator

$variable name
(or)
${variable name}

7) c='cat emp'
   echo $c
   ≡

8) mypath='/home/top/abc
   cd $mypath

155

x = $x info

$ echo $x (It prints empty value)

$

x = ${x} info

$ echo $x

topnotch info

$

Note:- To add some text to existing variables, use { }.

Constant Variables: "read only" is the keyword to create constant variables

x=100

read only x

Global variables: "export" is the keyword, to create global variables

How to take input from user:- "read" is the keyword to create take input from user.

Syntax: read variable name

How to take input from user with prompt:

read -p "prompt :" variable name

Eg: ① $ read -p "enter a name :" name ↵

Enter a name : topnotch info

② $ read -s -p "enter a password :" name ↵

Enter a password : ↵

156

To see all system defined variables along with its values

```
$ set ↵
HOME = /home/topnotch
SHELL = /bin/bash
LOGNAME = topnotch
PATH =
MAIL = /var/spool/mail/topnotch
MAILCHECK = 60
PS1 = $
PS2 = >
```

* How to change system prompt ?

```
PS1 = "topnotch $"
```

* How to change Input prompt ?

```
PS2 = "__ -->"
```

* PATH, system defined variables is used to set applicat
software paths like java, oracle, oracle apps

Operators

1. Arithmetic Operators

```
+
-
*
/
%.
```


157

a) Numeric comparision operators

- -lt (less than)
- -le (less than or equal to)
- -gt (greater than)
- -ge (greater than or equal to)
- -eq (equal to)
- -ne (not equal to)

b) String comparison op

<
>
=
!=

3) Logical Operators

- -a (logical and)
- -o (logical or)
- ! (logical not)

Note: Each and Every operator should contain space before an after operator except assignment operator

4) Assignment Operator

=

```
a=10
b=4
$echo $a +$b ↵
10+4
```

expr: expr is the keyword to convert string expression to integer expression

Eg: ① $ echo `expr $a + $b` ↵

158

1) expr integer expression

Eg:- 'expr $a + $b'

2) $((integer expression))

Eg: $(($a + $b))

3) let integer expression

Eg: let c= $a + $b

2) $echo 'expr $a * $b' ↵

40

Float authmetic

a=10.4

b=3.2

$echo $a + $b

10.4 + 3.2

$echo 'echo $a + $b |bc'

13.6

$echo 'echo $a - $b |bc'

7.2

Note :- bc is the command, to peform any float related calculations

$bc ↵ (binary calculator)

10+4 ↵

14

10.3 + 4.2 ↵

14.5

10>4 ↵

1

10<4 ↵

0

10/4 ↵

2

sqrt(200) ↵

14

Scale = 2

sqrt(200)

14.14

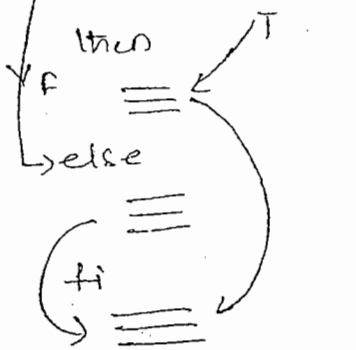
10/4

2.5

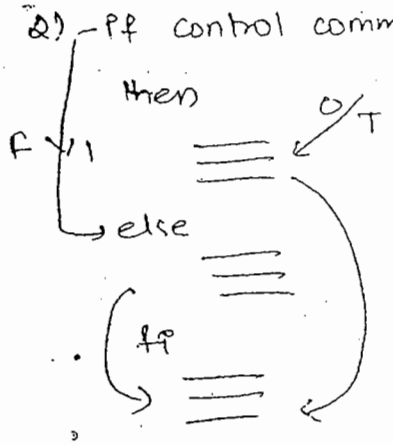
ctrl

159

1) if [condition]



2) if control command



File test commands:

1) -f => True, if it is regular file
2) -d => True, if it is directory file
3) -l => True, if it is link file
4) -b => True, if it is block special file
5) -c => True, if it is character file
6) -e => True, if file exist
7) -s => True, if file is not empty
8) -r => True, if file has read permission
9) -w => True, if file has write permission
10) -x => True, if file has execute permission

String test commands:

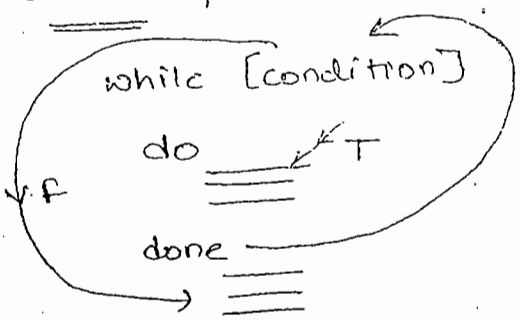
1) -z => True, if string is empty
2) -n => True, if string is not empty

is a single line comment

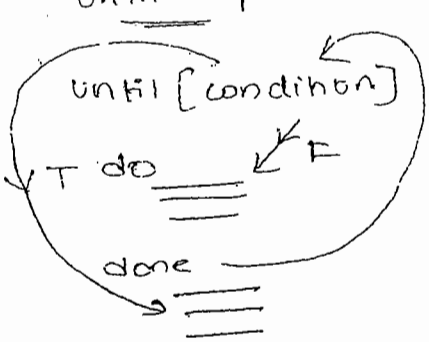
160

```
case variable in

Pattern 1) ===
        ;;  # to terminate the case

Pattern 2) ===
        ;;
   .
   .
   .
Pattern n) ===
        ;;

   *) ===
      ;;
esac
```

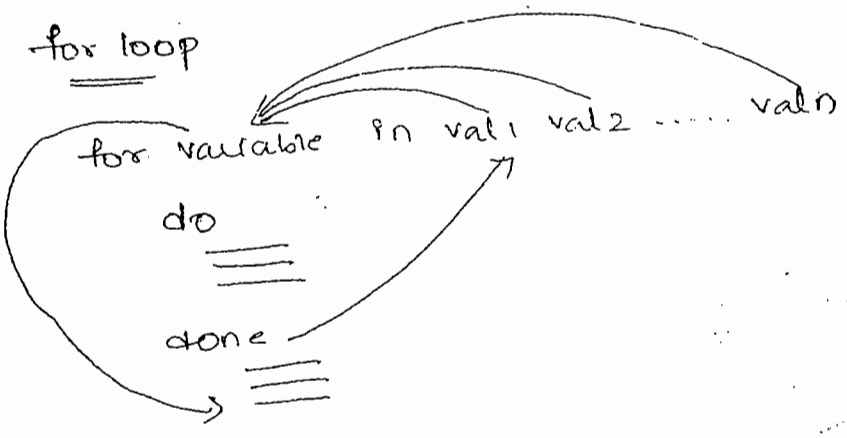
while loop



Until loop



for loop



161

```
while [condition]
do
  ==
  ==
  break
  ==
done
```

Continue : Continue is the keyword, to start the loop again

```
while [condition]
do
  ==
  ==
  continue
  ==
done
```

→ The 1000 can be replaced with 1024 and still be (using the other acceptable standards. Both of these standards are coined depending on what type of Storage you are refering.

Processor (or) virtual storage	Disk Storage
1bit = Binary digit	1bit = Binary digit
8bits = 1Byte	8bits = 1Byte
1024 bytes = 1 kilobyte	1000 bytes = 1kilo
1024 kilo = 1 mega byte	1000 kilo = 1mega
1024 mega = 1 Giga byte	1000 mega = 1giga
1024 giga = 1 Tera byte	1000 giga = 1 Peta
1024 tera = 1 petà byte	1000 tera = 1 peta
1024 peta = 1 Exabyte	1000 peta = 1 exa
	1000 exa = 1 zetta

162

1024 zetta = 1 yotta byte
1024 yotta = 1 Bronto byte
1024 Bronto = 1 Geobyte

1000 yotta = 1 Bronto
1000 Bron = 1 geophy

Communication commands:

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

$ write username / terminal name ↵
=
Ctrl d

Note: Default mesg is "y"

top 1	top 2
$write topnot2 ↵	message from topnot2:
Hello	Hello
Ctrl d	
message from topnot2	$ write topno1 ↵
Hi	Hi
	Ctrl d
$ mesg n ↵ (To deny message)	$ write topno1 ↵
	topno1 user disabled messages.
$ mesg y ↵ (To allow messages)	

Wall: It is used for to send broadcast message to all users whoever connected to server and mesg is y.

$ wall ↵
Welcome to topnotch
Ctrl d

163

Mail

Syntax: $ mail username ↵

ctl d

$

$ mail topnotch1 ↵ This mail is transfered to topnotch1 mail box (/var/spool/mail/topnot

Subject: Hello ↵

Hi, how r u?

ctl d

$ mail topnotch1 topnotch2 topnotch3 ↵ This mail is

Subject: from topnotchinfo ↵ transfred to topnotch1, topnotch2, topnotch3, User mail boxes.

ctl d

$ mail topnotch < stud ↵ It transfers stud file content to topnotch user

→ $ mail is the command to open mail box.

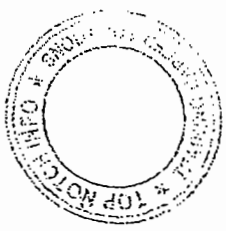
Eg: $ mail ↵

mailno	sender's name	Date	Time	subject
1	> top	mar 15	8:30	Hello
2	> sagar	–	–	very urgent
3	> krith	–	–	stud file
4	> root	–	–	Hi

& 2 ↵ [It opens 2nd mail]

& 3 ↵ [It opens 3rd mail]

& q ↵ [To quit from mail box]



(14)

2) ftp: ftp ...

It is used to transfer files from one server to another server account

ftp ipaddress ↵
login : ____ ↵
password : ____ ↵
ftp >

ftp commands:

ftp > ls ↵ [server]
ftp > !ls ↵ [client]
ftp > pwd ↵ [server]
ftp> !pwd ↵ [client]
ftp> cd dirname ↵ [server]
ftp> lcd dirname ↵ [client]
ftp > get filename ↵ [To download a file]
ftp > mget file1 file2 .... filen ↵ [To download multiple files]
ftp > put filename ↵ [To upload a file]
ftp> mput file1 file2 .... filen ↵ [To upload multiple files]
ftp > ? ↵ [It list all FTP commands]
ftp > bye ↵ [To quit from FTP]

gZip files

Zip: It is used for to zip the file

Ex: $ zip filename ↵

(15)

165

&w file name => It copies current mail co
to given file.

&r => To reply

&p => To print out

&d => delete current mail

&d 2 => deletes 2nd mail

&d 1-10 => deletes 1st mail to 10th mail

Note: By default all mails will store in primary ma
(/var/spool/mail) all opened mails in primary mai
transferred to secondary mail box (mbox)

* How to open secondary mail box :- $mail -f ↵

* The primary mail box only maintains unread mails.

Networking Commands:

1) Telnet: Telnet protocol is in built in windows/Lin
It is used for to connect to remote servers.

Syntax: telnet ipaddress ↵
login : toprnot ↵
password : xxxxx ↵
$

$hostname ↵ It displays
name

$host name -i ↵ It displa
Server ip add

$uname ↵ It displa
O/s name

$uname -r ↵ It dis
kernel valu

166

=g gzip sample ↵

sample.gz

Gunzip: To unzip the file

gunzip: To unzip the file

$ gunzip sample.gz ↵

sample

Compress: to compress the file

$ compress filename ↵

$ compress sample ↵

Sample Z

Uncompress: To uncompress the file

$ uncompress Sample.Z ↵

Sample

Disk Status

1) $ df ↵ It display disk space in bytes

2) $ df -h ↵ It displays disk space in kilo bytes

3) $ du ↵ It displays directory wise disk usage in form of blocks.

Background jobs

* By default jobs come under foreground

① $ cp file1 file2 ↵ foreground job

② $cp file1 file2 & ↵ Background job

$_{12}$ (PID)

③ $ sort sample >a1 & ↵ Background job

$_{13}$ (PID)

Note: In foreground, user can execute only one job. But in the background user can execute many jobs.

167

* $PS (or) $PS -f ↵ It displays currently running process list in the current account.

* $PS -ef ↵ It displays currently running process list the linux server.

* How to kill background job? $kill pid (or) $kill -9 ↳

* nohup: The nohup jobs will create in server account. So nohup jobs will execute even the user disconnects for his account.

Eg: $nohup cp file1 file2 & ↵

* $jobs ↵ It displays only background jobs with job id

* How to bring background job to fore ground?

$fg jobid ↵

Job Scheduling:-

1) cron tab
2) at
3) batch

Crontab: It executes the given jobs repeated by in server account

crontab -e ⇒ to open the crontab editor

168

* (0-23) => Hours
* (1-31) => Days
* (1-12) => month
* (0-6) => weekday

$ crontab -e ↵

1) * * * * * date
2) 30 10 * * * sh script1
3) 30 10 * * 0 sh script2
4) 30 22 * * 1,3,5 sh script3
5) 30 22 1,15 * * sh script4 > fi

1) => Every minute, it executes date command and it writes output into your mail box.

2) => It executes script1, every day at 10-30 Am

3) => It executes script2, every sunday at 10.30 Am

4) => It executes script3, every monday, wednesday & friday 10:3

5) => It executes script4, every month 1st and 15th at 10.30 pm an

it writes output into file1 file.

crontab -r => To remove all crontab jobs.

crontab -l => It list all crontab jobs.

at: It executes the given jobs only once.

at now ↵ => It executes the given jobs immediately

at > ls ↵

at > sh script1 ↵

at > ctrl d

* at now + 3minutes ↵ after 3minutes
* at now + 2 hours ↵ after 2hours
* at now + 3 days ↵ after 3days

169

* at 4pm tomorrow ↵ tomorrow at 4'o clock

* at 10.30 am July 10 ↵ July 10th at 10.30am

at q => It displays all "at" command jobs with

3) batch: The batch jobs will execute when server is free and server load is less.

```
$ batch ↵
at > ls ↵
at > child ↵
$
```

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

Two types of paths

1) Absdute path
2) Relative path

1) Absolute Path: It is the way of representing files directories from the top of hierarchy is called absolute path

```
# touch /root/f1
# cp Emp /world/Asia/india/Aplhyd
```


170

2) Relative path : It is the [illegible] directories which are related to current directory

```
# mkdir dir1 dir2
# cp Emp wisdom/wisdom1 ↵
```

→ To create a nested directory :

```
# mkdir -p world/Asia/india/AP/Hyd ↵
```

→ To see the nested tree structure

```
# tree world ↵
```

⇒ 1 Bit equals to how many bytes:

→ Bit is the smallest component of data and byte is larger then bit

8bits (2^3 bits) = 1byte

1024 bytes (2^{10} bits) = 1 kilo byte

1024 kilobyte (2^{20} bits) = 1 megabyte

1024 megabytes (2^{30} bits) = 1 gegabyte

171

awk

* The awk utility which takes its name from the initials of its authors (Alfred V. Aho, Peter J. Weinberger and Brain W. Kernighan) is a powerful programming lan

* How to run awk command

$awk 'pattern{action}' input-file.

* How to run awk program

$ awk -f scriptfile.awk input-file

* In awk programming, the field numbers begin with one

the first field is $1, and
the second field is $2, and
the third field is $3, and so on

* In awk, $0 represents one entire record

* variables are two types

a) System variables
b) User defined variables

* User defined Variables

1) Variable should begin with an alphabet
2) It shouldn't contain any space character

172

Built-in Variables

Variable	Function	Default
FS	Input field separator	space or tab
RS	Input record separator	newline
OFS	Output field separator	space or tab
ORS	Output record separator	newline
NF	Number of non empty fields in current record	
NR	Number of records read from all files	
FILENAME	Name of the current file	

Structure of awk program:

```
BEGIN {Begin's Action}  Preprocessing
    pattern {Action}
    pattern {Action}    Body
    pattern {Action}
END {End's Action}      postprocessing
```

* Expressions:

* **Regular Expression:-** The awk regular expressions are the defined in egrep. In addition to the expression and requires one of the two operators

a) match (~)

b) not match (!~)

173

Note: ... be enclosed in /slashes/:

Eg: $0 ~/ ^ A*B $/ # Record must begin with A & end with B

$3 !~/^ unix/ # third field must not start with unix.

$4 ~/unix $/ # Fourth field must end with unix.

Eg: $awk '$0 ~/^unix/ {print $0}' stud

Operators

Arithmetic Operators :- +, -, *, /, %, ++, --

Relational Operators : <, >, <=, >=, ==, !=

Logical Operator :- &&, ||, !

Assignment Operator :- =, +=, -=, *=, /=, %=

Note: Each and every operator must contain space bef and after operation

Eg: 1) $awk '{print}' stud ← It prints stud file conte

2) $awk '{print $1,$2,$4}' ← It prints 1st, 2nd & 4th fi from stud file

3) $awk 'BEGIN {OFS="\t"}; {print $1, $2, $4}' stud ;h

It prints the first 5 records, 1st, 2nd, & 4th fields with tab output field seperator.

174

It prints only unix students names & phnols.

5) $awk -f"," '$3>=5000 {print $0}' emp ↵

It prints all employee records whose salaries more than 50

6) $awk -f"," '$4==10 && $3>=5000 {print $0}' emp ↵

It prints all employee records who belongs dept no 10 and salary more than 5000

If statement

```
{
if (condition)
  {
  ===
  }
}
```

if else

```
{
if (condition)
  {
  ===
  }
else
  {
  ===
  }
}
```

while loop

```
while (condition)
  {
  ===
  }
```

do...while

```
do
  {
  ===
  }
while (condition);
```

for loop

```
for (initialization; condition; incr/decr)
  {
  ===
  }
```

175

* How to ...

Eg: 1) BEGIN {

"date" | get line

print ($1, $4) # Day of week & time of da

}

2) BEGIN {

"who" | get line

print ($1) # only usernames who are connected

} to the server.

* awk and grep

grep 'regular exprenion' filename

awk '$0 ~/ regular exprenion / {print $0}' filename

Eg 1) $ grep '^[A-Z].*[A-Z]$' file1

$ awk '$0 ~/^[A-Z].*[A-Z]$/ {print $0}' file

2) grep -v '^unix' file1

$ awk '$0 !~/^unix/ {print $0}' file1

* sed and awk

sed '=' stud

awk '{print NR ; print $0}' stud

The command prints stud file records with line n

(26)

176

$ cat 2,4d' shd

$awk 'NR<2 || NR>4 {print $0}' shd

The commands delete 2nd line to 4th line from Stud f

Vi Editor

→ It is used for to create new files or to modify existing files.

→ It is classified into 3 types.

1. Command mode
2. Input or Insert mode
3. Ex command mode

→ The default mode is command mode.

→ Vis command to open vi editor.

→ The following are the commands to shift from command mode to insert mode

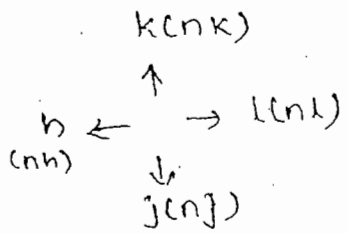
1) A → It places cursor at end of the current file
2) a → It places cursor at right side of the cursor line
3) I → It places cursor at beginning of the current line
4) i → It places cursor at left side of the cursor line
5) O → It inserts new line of the cursor.
6) o → It inserts new line below of the cursor.

→ "ESC" is the key to shift from Insert mode to command mode.

→ ":" is the command to shift to Ex command mode from command mode

177

1)



n means any number

2) w(nw) → next word starting

e(ne) → word ending

b(ne) → word begining

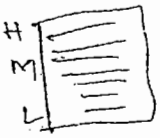
3) $ → end of the current line (end key)

^ → begining of the current line (home key)

4) H → Begining of the current page

M → Middle of the current page

L → End of the current page



5) ctrl f → forward on page (pgdn)

ctrl b → backward one (pgup)

6) x(nx) → Delete current character (del key)

7) X → Delete previous character (backspace key)

8) dw (ndw) → Delete current word

9) dd (ndd) → Delete current line

10) d$ → Delete current position to end of the line.

11) d^ → Delete current position to begining of the line.

178

13) yy (nyy) → To copy a line.

14) y$ → It copies current position to end of the line.

15) y^ → It copies current position to beginning of the line.

16) p → paste below the cursor

P → paste above the cursor

17) J → To join a line

18) cc → To clear a line

19) u → undo

20) ZZ → Save and Quit

EX command mode commands:

:w → Save without quit

:w filename → Save with given filename

:q! → quit without save

:wq → Save and quit

:n → places cursor at nth line

:set nu → set line numbers

:set nonu → To remove line numbers

:! <unix command> → Execute unix command

:/string/ → Top to bottom search n → next occurence

:Starting line no, ending line no s/old string/new string/gi →

Search and replace string.

Eg: 1, $ s/unix/linux/gi

179

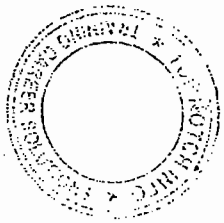
:1,$ s/$/;/ → It adds ; at end of each line

Eg: :nd → It deletes n^{th} line

:3d → It deletes 3rd line

:$d → It deletes last line

:4,7d → It deletes 4th line to 7th line



180

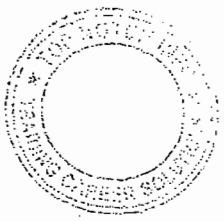
Shell Scripting Programs

#Example of variables.

```
a=10
b=20
echo "a is: $a"
echo "b is: $b"
```

Write a script accept 2 integer nos and display.

```
echo -e "Enter a number 1: \c"
read a
echo -e "Enter a number 2: \c"
read b
echo "a value is: $a"
echo "b value is: $b"
```



Write a script accept 2 integer no's and find sum

```
echo -e "Enter a number 1: \c"
read a
echo -e "Enter a number 2: \c"
read b
c=`expr $a + $b`
echo "sum of $a and $b is: $c"
```

Write a script accept a file name and open

```
echo -n "Enter a filename to open: "
read fn
echo "- - - -"
```


181

```
esac
echo "-------"
```

Write a scipt accept a filename and delete all empty lines

```
echo -n "Enter a filename to delete blank lines:"
read fn
grep -v "^$" $fn > temp
mv temp $fn
echo "$fn file empty lines deleted"
```

Write a scipt accept a filename and delete all duplicate lines.

```
echo -n "Enter a filename to delete duplicate lines:"
read fn
grep -v "^$"
sort $fn | uniq -u > temp
mv temp $fn
echo "$fn file Duplicate lines are deleted"
```

Write a scipt accept a number and check the given no is +ve or -ve

```
echo -n "Enter a number:"
read n
if [ $n -gt 0 ]
then
echo "$n is a +ve no:"
else
echo "$n is a -ve no"
fi
```


182

Write a script accept a number and check given no is even or odd number

```
echo -n "Enter a number:"
read n
if [`expr $n % 2` -eq 0]
then
echo "$n is a Even no:"
else
echo "$n is a odd no."
fi
```

Write a script accept a filename and delete given file

```
echo -n "Enter a file name:"
read fn
if rm $fn
 then
  echo "$fn file deleted"
else
echo "No such file"
fi
```

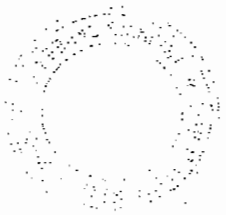
Write a script check today is Sunday or not

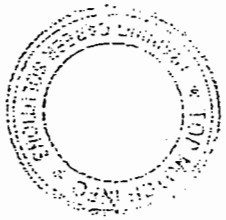
```
x= `date +%a` # x=`date | cut -c1-3`
if [$x = "Sun"]
then
 echo "Yes Today is Sunday"
else
echo "Sorry Today is $x day"
fi
```


183

connect to the server or not.

```
echo -n "Enter user name:"
read un
if grep -w $un /etc/passwd > /dev/null
then
if who|grep -w $un > /dev/null
 then
   echo "Logged In"
 else
   echo "Not logged In"
 fi
 else
 echo "$un user doesn't exist"
fi
# Write a script accept a filename and Open
echo -n "Enter a filename:"
read fn
if [-e $fn]
then
 if [-f $fn]
 then if [-r $fn]
 then
  cat $fn
else
echo "No read permision"
#chmod 644 $fn
# cat $fn
 # echo "No read permission"
```





```
fi
else
echo "It is not a file"
fi
else
echo "$fn file doesn't exist"
fi
```

write a script accept a filename and check the file is regular file or directory file.

```
echo -n "Enter a filename:"
read fn
if [-e $fn]
then
 echo "$fn is a regular file"
elif [-d $fn]
then
 echo "$fn is a directory file"
else
echo "It is not a file or directory"
 fi
else
echo "$fn file doesn't exist"
fi
```

write a script accept a string and check the given string is empty or not.

```
echo -n "Enter a string :"
read str
if [-z "$str"]
```


185

```
echo "Given string empty"
else
echo "Given string not empty"
fi
```

Using Case

```
# case example.
echo "enter a number b/w 1 to 4:"
read n
case $n in
1) echo "one";;
2) echo "two";;
3) echo "three";;
4) echo "four";;
*) echo "invalid number";;
esac
#write a script accept a single character and check the given character is alphabet or digit or special character.
echo -n "Enter a single character:"
read ch
case $ch in
[a-z A-Z]) echo "Alphabet";;
[0-9]) echo "Digit";;
[^a-z A-Z 0-9]) echo "special character";;
*) echo "You entered more than one character";;
esac
```

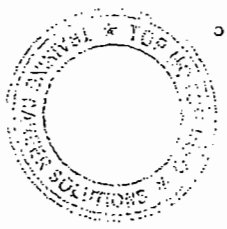

186

Write a script accept a month and display quarter of the given month.

```
echo -n "Enter a month[mon]:"
read mm
case $mm in
jan|feb|mar) echo "1st Quarter";;
apr|may|jun) echo "2nd Quarter";;
jul|aug|sep) echo "3rd Quarter";;
oct|nov|dec) echo "4th Quarter";;
esac
# [Jj]an|[Ff]eb|[Mm]ar) echo "1st Quarter";;
# [Aa][Uu][Gg]
# [Aa][Uu][Gg]*
```

Example of Menu Program

```
clear
tput cup 6 10
echo "MAIN MENU"
tput cup 7 10
echo "*****"
tput cup 8 10
echo "1. Date"
tput cup 9 10
echo "2. List of users"
tput cup 11 10
echo "3. Open a file"
```



187

```
tput cup 11 10
echo "4. delete a file"
tput cup 12 10
echo "5. Exit"
tput cup 20 5
echo "enter a choice [1-5]:"
read choice
case $choice in
1) echo "Today date is: 'date'";;
2) who;;
3) sh fopen.sh;; # ./fopen.sh
4) sh del.sh ;; # ./del.sh
5) echo "Thank You"
exit;;
*) echo "choice wrong try again";;
esac
```

```
#write a script accept a string and display reverse of the
given string
echo -n "Enter a string:"
read str
l='echo $str|wc -c' #length of the string
while [$l -gt 0]
do
ch='echo $str|cut -c$l'
temp=$temp$ch
l='expr$l-1'
done
echo "$ctr is $temp"
```

```
# To open a filename
#
# Example of while loop
ans="y"
while [$ans="y"]
do
echo "Enter a filename to open:"
read fn
if [-e $fn -a -f $fn]
then
cat $fn
else
echo "no such file"
fi
echo "Do u want to open one more file [Y/N]:"
read ans
done
#Example of while loop
while true # until false
do
echo "Enter a filename to open:"
read fn
if [-e $fn -a -f $fn]
then
cat $fn
break
else
continue
fi
done
```

```
# Example of ...
# sleep is used for to stop the execution specified no of seconds
while true
do
clear
tput cup 5 8
echo "WELCOME TO"
sleep 2
clear
tput cup 5 8
echo "TOPNOTCH"
Sleep 2
done
```

```
# write a script create "n" no of users.
echo -n "Enter no of users to create:"
read n
$i=1
while [ $i -le $n ]
do
$x=topno$i # It creates users with topno name
useradd $x
i=`expr $i+1`
done
```

```
# Example of for loop:
for i in 1 2 3 4 5
do
echo $i
done
```


190

```
# Example of for loop

a=10
b=20
c=30
for i in $a $b $c
do
 echo $i
done

# write a script to display all sub directories of current
directory

for i in *
do
if [-d $i]
then
echo $i
fi
done

# write a script to display all empty files in the current
 directory

for i in *
do
if [! -s $i ]
then
 echo $i #rm $i => To delete empty files
fi
done
```

Write a script to [illegible] are receiving salary more than 5000 from emp file

```
#101, Ram, 9000, 10
#102, Sagar, 4000, 20
#103, pratap, 000, 30
#104, Akshay, 8000, 10

for i in 'cat emp'
do
j= `echo $i |cut -d "," -f 3`
if [$j -ge 5000]
then
echo $i
fi
done
```

Write a script retrieve details of employees who are receiving salary more than 5000 in dept no 10 from emp file and insert into empl file

```
#101, Ram, 9000, 10
#102, Sagar, 4000, 20
#103, pratap, 666666000, 30
#104, priya, 8000, 10

for i in 'cat emp'
do
j = ' echo $i |cut -d "," -f3'
k = ' echo $i |cut -d "," -f4'
```

```
if [$j -ge 5000 -a $k -eq 10]
then
 echo $i >> emp1
fi
done
do
j=`echo $1 | cut -d',' -f3`
k=`echo $1 | cut -d',' -f4`
if [$j -ge 5000 -a $k -eq 10]
then
echo $i >> emp1
fi
done.
```

Command line Arguments (or) Positional Arguments

* At the time of execution of shell script, if user passes any arguments known as command line arguments (or) positional arguments.
* The special variables holds positional parameter value. The Special variables are $0, $1, $2, $3, $4, $5, $6, $7, $8, $9 $#, $*, $?, $$

=> $0 => Name of the program
=> $1 => 1st parameter value
=> $2 => 2nd parameter value
=> $3 => 3rd " "
=> $4 => 4th " "

=> $5 => 5 parameter ...

=> $6 => 6th " "

=> $7 => 7th " "

=> $8 => 8th " "

=> $9 => 9th parameter value.

=> $# => counts no of arguments

=> $* => all parameter values

=> $@ => all parameter values but each and every param encloses within double quotes

=> $? => It holds last executed command status,
If the command executed succesfully it holds 0 otherwise non-zero value.

=> $$ => It hold user parent shell processid.

```
$vi hello
for i in $*
do
echo -n "$i"
done
:wq
$chmod 755 hello
$./ hello Top Notch Info
```

```
$vi calc
if [ $# -eq 3]
then
c= 'echo $1 $2 $3 |bc'
```


194

```
echo $c
else
echo "Invalid no of arguments"
fi
wq.
$chmod 755 calc
$/calc 10 + 4
or
$ sh calc 10+4
```

$vi checkuser

```
if [$# -eq 1]
then
if who |grep $1 >/dev/null
then
 echo "Logged In"
else
echo "not logged In"
fi
else
echo "Invalid no of arguments"
fi

:wq
$chmod 755 checkuser
$./checkuser toprocth
```

195

1) Write a shell sript to connect to oracle db

```
$vi a1.sh ↵
  sqlplus scott/tiger
  :wq
$ sh a1.sh ↵
```

2) write a shell sript to insert data into oracle ~~emp ta~~

```
$vi a2.sh ↵
clear
x=101
y="Ram"
sqlplus -s scott/tiger <<EOf
insert into emp (empno, ename)
values ($x, '$y');
commit;
EOf
:wq
$ sh a2.sh ↵
```

3) write a shell sript to retrieve data from oracle emp table

```
$vi a3.sh ↵
sqlplus -s scott/tiger <<EOf
select * from emp;
EOf
:wq
$ sh a2.sh ↵
```


196

write a ...
table.

cat >emp ↵
101, Ram, 8000, 10 ↵
102, Sai, 7500, 20 ↵
103, Sagar, 6000, 30 ↵
104, Akshay, 9000, 10 ↵
ctrl d

$vi as.sh ↵

```
for i in 'cat emp'
do
a = 'echo $i | cut -d"," -f 1
b = 'echo $i | cut -d"," -f 2
c = 'echo $i | cut -d"," -f 3
d = 'echo $i | cut -d"," -f 4
sqlplus -s scott/tiger <<EOF
insert into emp(empno, ename, sal, dep values('$a','$b',$c,$
commit ;
EOF
done
:wq
```

$ sh as.sh ↵

197

$ sqlplus ↵

Username : scott ↵

password : tiger ↵

sql > ! <unix command> ↵

eg: sql > ! who ↵

sql > ! ls ↵

sql > ? ↵ (To return to unix)

$ exit ↵ [To return to sql]

$ vi hello

```
#! /bin/ksh
print "Hello In"
print "welcome to topnotch info"
```

:wq

$ ./hello ↵

Note #! is a shebang statement. It is used to invo
shell interpreter path.

198

It is a group of statements to perform specific ta
and it returns a value.

The main concept of function is

1) To reduce length code of the program
2) To reduce cost of maintenance
3) Easy to maintain the application

Syntax:

function name arg1, arg2 .... # calling function

function functionname ()
{
==
}

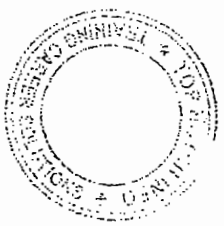
The function call arguments will store in $1, $2, $3.. &

° Example of function

1). $vi fun1.sh ↵

function hello()
{
echo "welcome to function"
}
hello.

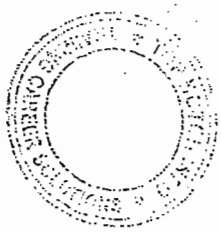
:wq
$ sh fun1.sh



199

2) Write an script [illegible]

```
$vi fun2.sh ↵
function add()
{
if [$# -eq 2]
then
   c='expr $1 + $2'
  echo "$1 + $2 = $c"
else
echo "Invalid arguments"
fi
}
add 15 2
:wq
$ sh fun2.sh ↵
```



200