# zINTRODUCTION TO LINUX



**What is Operating System?**

Operating system is an interface between user and the computer hardware. The hardware of the computer cannot understand the human readable language as it works on binaries i.e. 0's and 1's. Also it is very tough for humans to understand the binary language, in such case we need an interface which can translate human language to hardware and vice-versa for effective communication.

**Types of Operating System:**

* Single User - Single Tasking Operating System
* Single User - Multitasking Operating System
* Multiuser - Multitasking Operating System

**Single User - Single Tasking Operating System**

In this type of operating system only one user can log into system and can perform only one task at a time.

**E.g.: MS-DOS**

**Single User - Multi tasking operating System**

This type of O/S supports only one user to log into the system, but a user can perform multiple tasks at a time, browsing internet while playing songs etc.

**E.g.: Windows -98, XP, vista, etc**.

**Multiuser - Multi Tasking Operating System**

This type of O/S provides multiple users to log into the system and each user can perform various tasks at a time. In a broader term multiple users can log in to system and share the resources of the system at the same time.

**E.g.: UNIX, LINUX etc.**

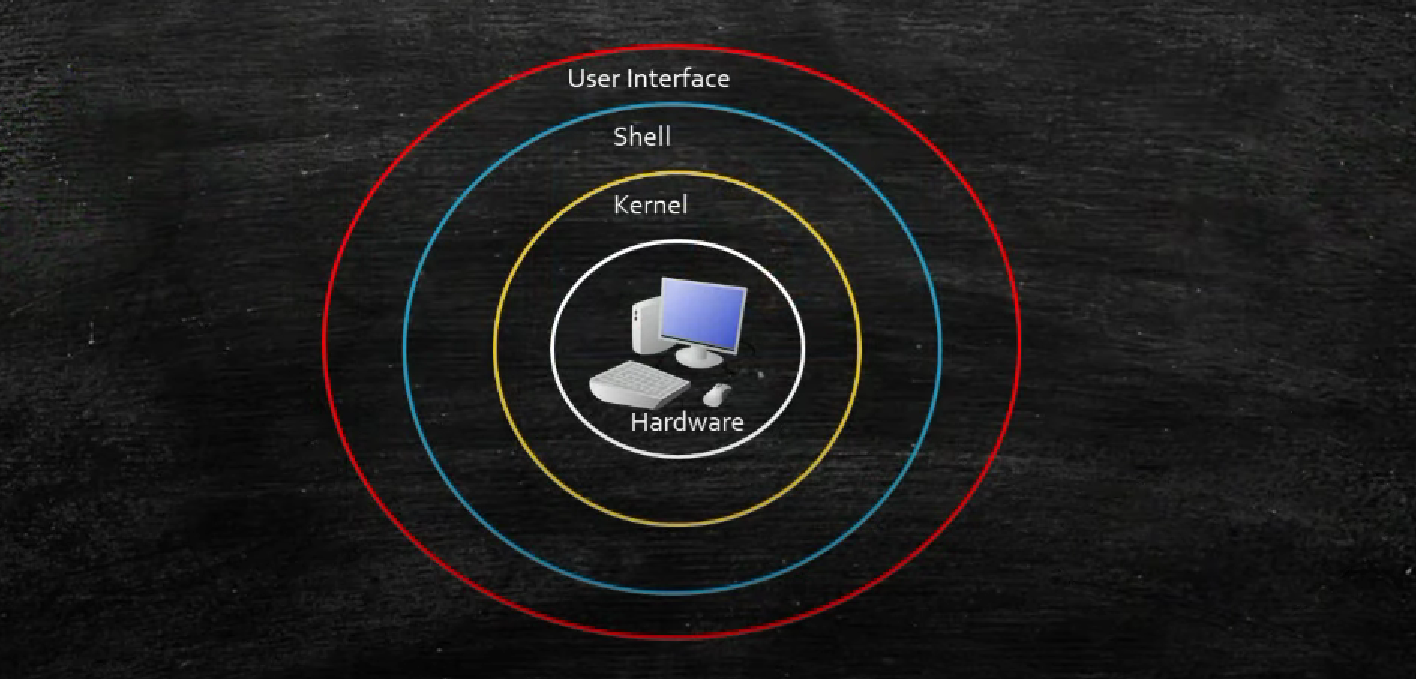
# Unix vs Linux

* Unix was developed for multi-use and multi-tasking in mid of 1970 in bells labs by ATT and GE.
* Then Linux born in 1991 by Linux Torvalds.
* Linux is mostly free
* Linux is open source

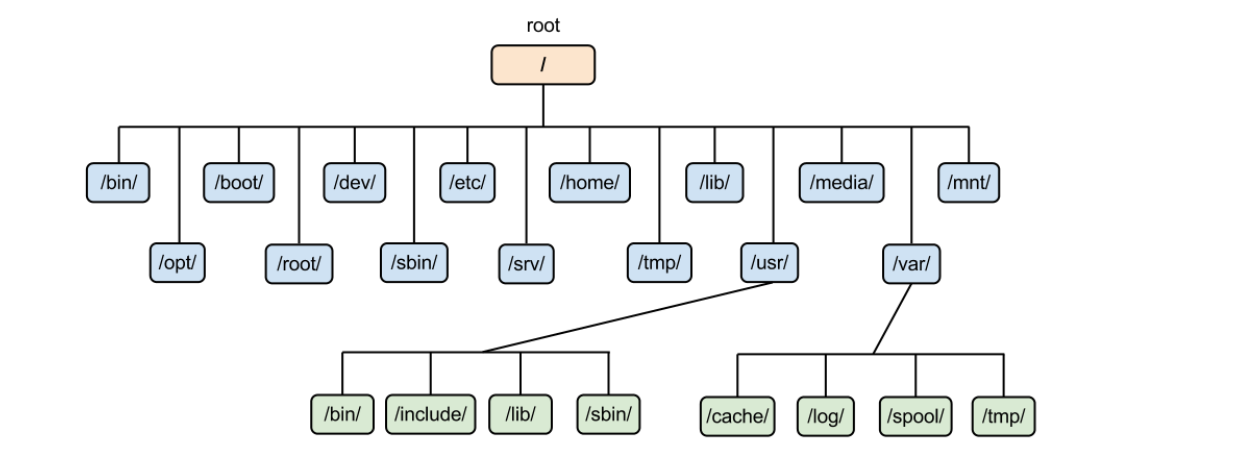
# What is Linux:

Linux is an operating system which sits in the middle of your hardware and users.

**ARCHITECTURE OF LINUX**

****

## **FILESYSTEM HIERARCHY SYSTEM**



**/** This is top level directory

It is parent directory for all other directories It is called as ROOT directory

It is represented by forward Slash (/)

C:\ of windows

.

**/root** it is home directory for root user (super user) It provides working environment for root user.

C:\users\Administrator

**/home** it is home directory for other users

It provide working environment for other users (other than root) c:\users\kishor

**/boot** it contains bootable files for Linux

Like vmlinuz (kernel). ntoskrnl

Initrd (INITial Ram Disk)and

GRUB (GRand Unified Boot loader). boot.ini, ntldr

**/etc** it contains all configuration files

Like /etc/passwd..... User info

/etc/resolv.conf... Preferred DNS

/etc/dhcpd.conf DHCP server

C:\windows\system32\dirvers\

**/usr** by default soft wares are installed in /usr directory (UNIX Sharable Resources)

c:\program files

**/opt** It is optional directory for /usr It contains third party softwares c:\program files

**/bin** it contains commands used by all users (Binary files)

**/sbin** it contains commands used by only Super User (root) (Super user's binary files)

**/dev** it contains device files

Like /dev/hda ... for hard disk

/dev/cd rom ... for cd rom

(Similar to device manager of windows)

**/proc** it contain process files

Its contents are not permanent, they keep changing It is also called as Virtual Directory

Its file contain useful information used by OS

like /proc/meminfo ... information of RAM/SWAP

/proc/cpuinfo ... information of CPU

**/var** it is containing variable data like mails, log files

**/mnt** it is default mount point for any partition It is empty by default

**/media** it contains all of removable media like CD-ROM, pen drive

**/lib** it contains library files which are used by OS It is similar to dll files of windows.

Library files in Linux are SO (shared object) files

/tmp This directory for temporary files.

**What is root ?**

There are 3 types of root on Linux system.

1. Root account: Root user
2. Root as /: The first directory.
3. Root home directory: Root user directory /root

**Basic Commands**

**Creating files and directories**

**Creating files:**

1. Touch
2. Cp
3. Vi, nano and vim

**`**

**Creating and copying directory**

1. Mkdir
2. Cp (command to copy a directory)
3. cp -r <source\_folder> <destination\_folder>

(-r is for recursive)

Cp -r config /tmp/

1. touch abcd{1..11}-xyz
2. rm \*xyz

**Adding text to Files(3 ways):**

1. vi
2. Redirect command output > or >>
3. Echo > or >>

**Find files and directories**

Two main commands are used to find files/directories

1. find
2. locate

Ex: find / -name ‘index.html’

find .

ex: locate index.html

what is the difference between find and locate commands:

**locate simply looks its database and reports the file location.** **find does not use a database, it traverses all the directories and their sub directories and looks for files matching the given path**.

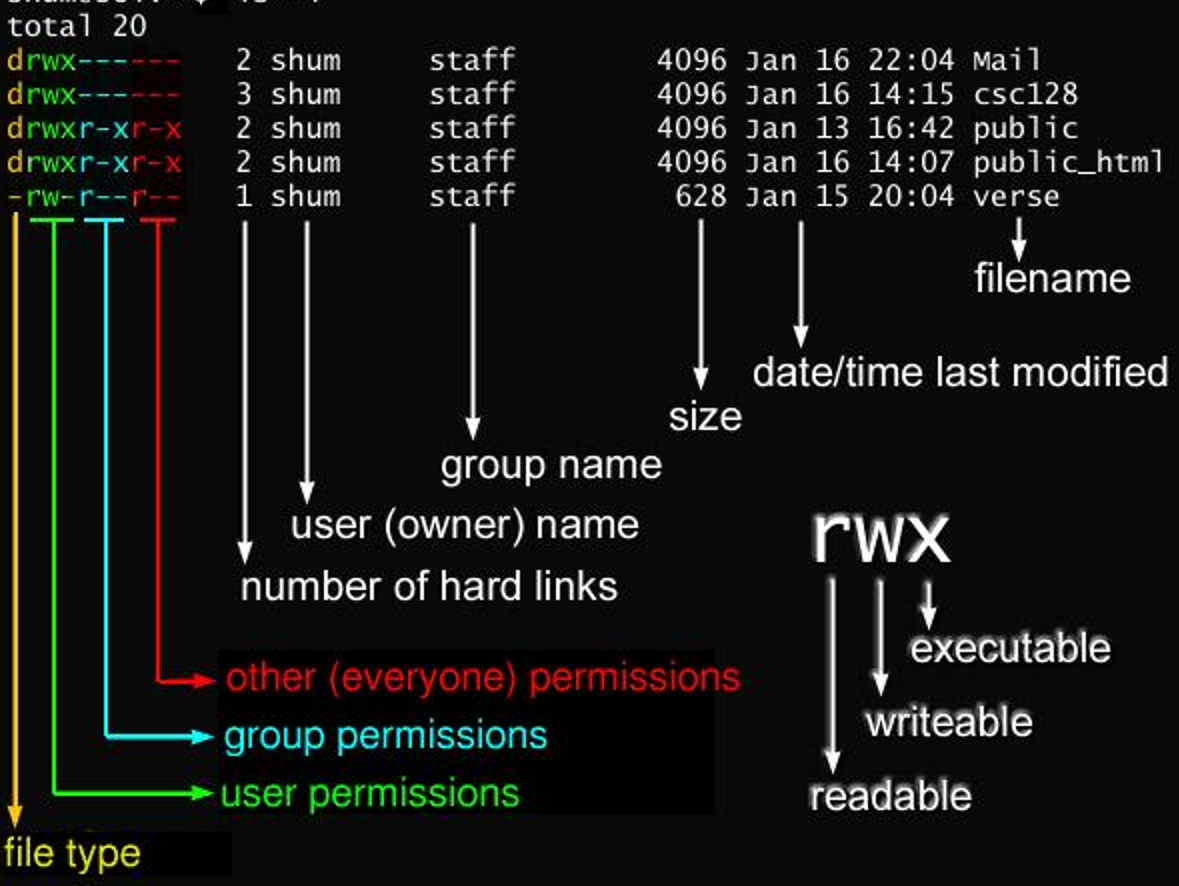
**Here locate is much faster than find**

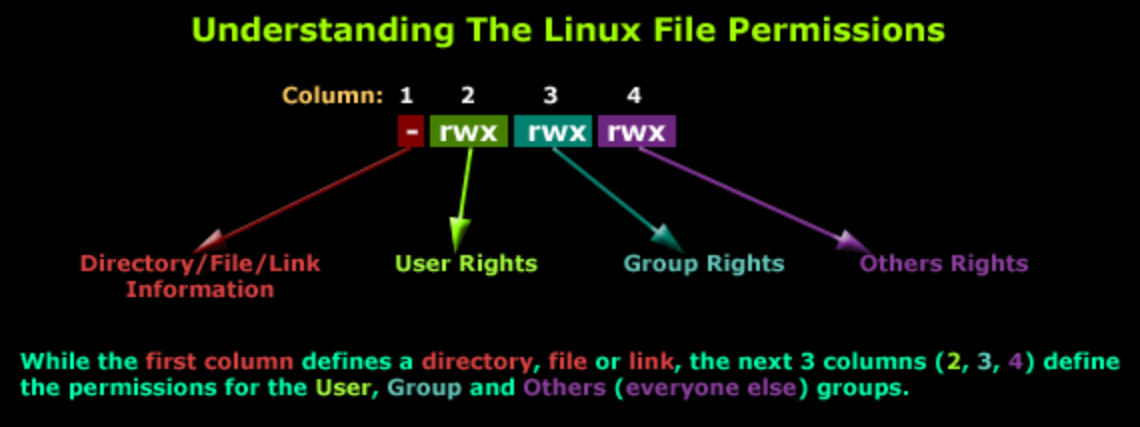
**To update local database run: updatedb**

**Renaming a File and directories:**

**#mv <old name> <new name> File types:**

**Files and directory permissions:**





**We can provide permissions in 2 ways:**

1. **Numeric method**
2. **File and directory permission**

**File ownership commands(chown and chgrp):**

1. **Change owner**

**chown username <file or dir>**

**Ex: Chown kishor file1**

1. **Change group**

**Chgrp groupname <file or dir>**

Chgrp ec2-user file1

**Access control list(ACL)**

* Define more fine-grained discretionary access rights for files and directories.
* Often, you want to share files among certain groups and specific users. It is a good practice to designate a directory for that purpose. You want to allow those groups and users to read, and write files in that directory, as well as create new files into the directory. Such special permissions can be given using ACL.

#### setfacl <option> < argument > < file or directory name >

* **The options are,**
* **-m** Modifies an ACL
* **-x** Removes an ACL
* **-R** Recurses into subdirectories

#### **The possible arguments are**

* **u: user**
* **g: group**
* **o: others**

#### **To assign read write and execute permission to a particular group**

----------------------------------------------------

#### setfacl –m u: <username>: <permissions> <file or dir name>

setfacl –m u:ec2-user:rx /home/ec2-user/file1

Verify it by using getfacl file1

#### **To assign read write and execute permission to a particular group**

-----------------------------------------------------------------------------------------------------

* **#setfacl –m g:<group name>:<permissions> <file or directory name>**
* **#setfacl –m g:ec2-user:rwx /folder1**

Verify it by using getfacl folder1

#### **Assigning read and execute permission to a user and a group at same time.**

-----------------------------------------------------------------------------------------------------------------------

* **#setfacl –m u:ec2-user:rx,g:ec2-user:rx /folder2**

Remove ACL access

* **setfacl –x u:<username> <file or dir name>**
* **setfacl –x g:<groupname> <file or dir name>**

**setfacl -x u:ec2-user kk.sh**

**setfacl -x g:ec2-user folder1**

**getfacl kk.sh**

**getfacl folder190**

**Recursive permission:**

**setfacl –Rm u:ec2-user:rx,g:ec2-user:rx /folder2**

**standard output to a tee file:**

“tee” command is used to store and view(both at same time) the output of any command.

Ex; echo “welcome to beseant” | tee kk.txt

**Pipes:**

A pipe is used by the shell to connect the output of the command directly to the input of the another command.

Command1 [argument] | command2 [arguments]

Eg: history | grep yum

**Listing files and directories:**

#ls list the file names

#ls -l long listing of the file

#ls –l filename to see the permissions of a particular file

#ls -al shows the files in ascending order of modification.

#ls b\* All the files start with b.

#ls -ld d\* Directory listing only

#ls –ld directory name to seepermissions of a particular directory

**File display commands:**

Cat, less, more, head, tail

**Cat:**

Cat commands display the entire file

Ex: cat /etc/passwd

**less:-**

The **less** command is used to see the output line wise or page wise.

Ex: less /var/log/messages

**Note: -**press **Enter** key to scroll down

line by line (or) Use **d** to go to next page

Use **b** to go to previous page

Use **/** to search for a word in the file

Use **v** to go vi mode where you can edit the file and once you save it you will back to less command

**more:-**

more is exactly same like less

Ex: #more /etc/passwd

Note: -press Enter key to scroll down line by line (or) Use d to go to next page

Use / to search for a word in the file

Use v to go vi mode where you can edit the file and once you save it you will back to more command

**head:**

By default, it used to display the top 10 lines of the file.

Ex:# head /etc/passwd

To display the custom lines

#head -n /etc/passwd (where n can be any number)

head -5 /etc/passwd (It will display top 5 lines)

**tail:**

It is used to display the last 10 lines of the file #tail /etc/passwd

To display the custom lines

#tail -n /etc/passwd (where n can be any number)

tail -3 /etc/passwd here it will display only 3 lines

**Filters / Text processors commands**

# USER AND GROUP ADMINISTRATION

In Linux/Unix user is one who uses the system. There can be at least one or more than one user in Linux at a time. Users on a system are identified by a username and a userid. The username is something that users would normally refer to, but as far as the operating system is concerned this is referred to using the user id (or uid). The username is typically a user-frfriendly string, such as your name, whereas the user id is a number. The words username and userid are often (incorrectly) used interchangeably. The user id numbers should be unique (one number per user). If you had two usernames with the same user id, effectively their permissions would be the same and the files that they create would appear to have been created by the same user. This should not be allowed and the **useradd** command will not allow usernames to share the same userid.

* Every user of the system is assigned a unique user ID number ( the UID)
* Users name and UID are stored in **/etc/passwd**
* User’s password is stored in **/etc/shadow** in encrypted form.
* Users are assigned a **home director**y and a program that is run when they login (**Usually a shell**)
* Users cannot read, write or execute each other’s files without permission.

**Types of users In Linux and their attributes:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TYPE** | **EXAMPLE** | **USER ID (UID)** | **GROUP ID**  **(GID)** | **HOME**  **DIRECTORY** | **SHELL** |
| **Super User** | **Root** | **0** | **0** | **/root** | **/bin/bash** |
| **System User** | **ftp, ssh,**  **apache nobody** | **1 to 499** | **1 to 499** | **/var/ftp , etc** | **/sbin/nologin** |
| **Normal User** | **Visitor, kishor**  **,etc** | **500 to 60000** | **500 to 60000** | **/home/user name** | **/bin/bash** |

**In Linux there are three types of users.**

**1.Super user or root user**

Super user or the root user is the most powerful user. He is the administrator user.

#### **System user**

System users are the users created by the software’s or applications. For example if we install Apache it will create a user Apache. These kinds of users are known as system users.

#### **Normal user**

Normal users are the users created by root user. They are normal users like kumar, kishor etc. Only the root user has the permission to create or remove a user.

**Whenever a user is created in Linux things created by default:-**

* + A home directory is created(/home/username)
  + A mail box is created(/var/spool/mail)
  + unique UID & GID are given to user

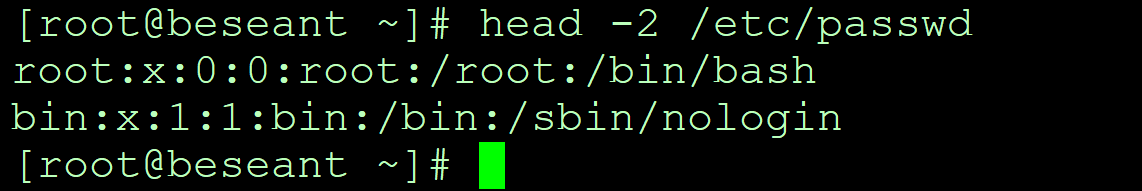
**Linux uses UPG (User Private Group) scheme**

* It means that whenever a user is created is has its own private group
* For Example, if a user is created with the name **Kishor,** then a primary group for that user will be Kishor only

**There are Three important files a user administrator should be aware of.**

1. **"/etc/passwd"**
2. **"/etc/shadow"**
3. **“/etc/group”**

/etc/passwd



#### The above fields are

* **root** =name
* **x**= link to password file i.e. /etc/shadow
* **0** = UID (user id)
* **0** =GID (group id)
* **root** or **bin** = comment (brief information about the user)
* **/root** or **/bin** = home directory of the user
* **/bin/bash** or /**sbin/nologin** = shell

#### /etc/shadow:

|  |
| --- |
| **root:$1fdsfsgsdfsdkffefje:14757:0:99999:7:::** |

**The fields are as follows,**

1. **root** = User name
2. **:$1fdsfsgsdfsdkffefje** = Encrypted password
3. **14757 =** Days since that password was last changed.
4. **0 =** Days after which password must be changed.
5. **99999 =** Days before password is to expire that user is warned.
6. **7 =** Days after the password is expires that the user is disabled.

**Creating a user**

* The syntax for creating a user in Linux is

#### # useradd <option> <username>

* **options are**
* **-u user id**
* **-G Secondary group id**
* **-g primary group id**
* **-d home directory**
* **-c comment**
* **-s shell**

Useradd  
groupadd

Userdel

Groupdel

Usermod

Ex: Useradd kumar

Ex: useradd -g cloud -s /bin/bash -c "cloud Architect" -m -d /home/kishor kishor

Ex: usermod -G user2 kishor

**Assigning password to the user:**

**passwd <user name>** to assign a password to a specific user, only root can assign password to other user.

Ex: Passwd kishor

**Modifying the user’s attribute**

* After creating a user if we need to modify the attributes of user like changing uid, changing secondary group id or adding a comment, locking or unlocking the user account, can be done by following command

#### Syntax. # usermod <options> <username>

* **options are:**
* **all the options which are used with useradd command can be used and the following,**
* **-L to LOCK account**
* **-U to UNLOCK account**
* **ex. # usermod -L newname** to lock the user account
* **ex. # usermod -U newname** to unlock the user account

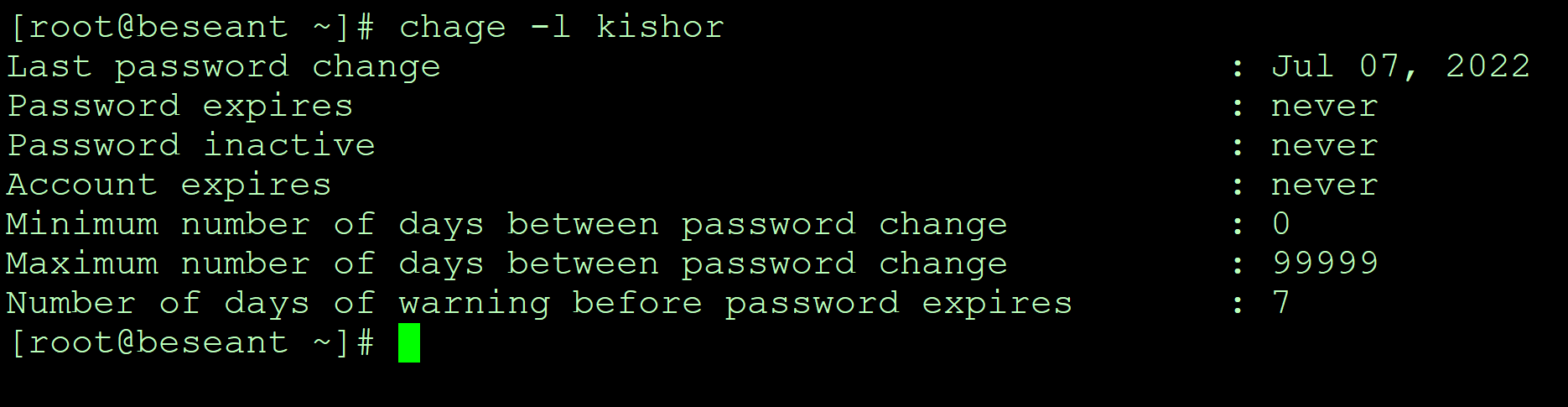
#### Note: - when an account is locked it will show! (Exclamation mark) in /etc/shadow file.

**The password parameters.**

* For any user we can set the parameters for the password, like **min** and **max password age, password expiration warnings** and **a/c expiration date** etc.
* To view the advanced parameters of the user, use

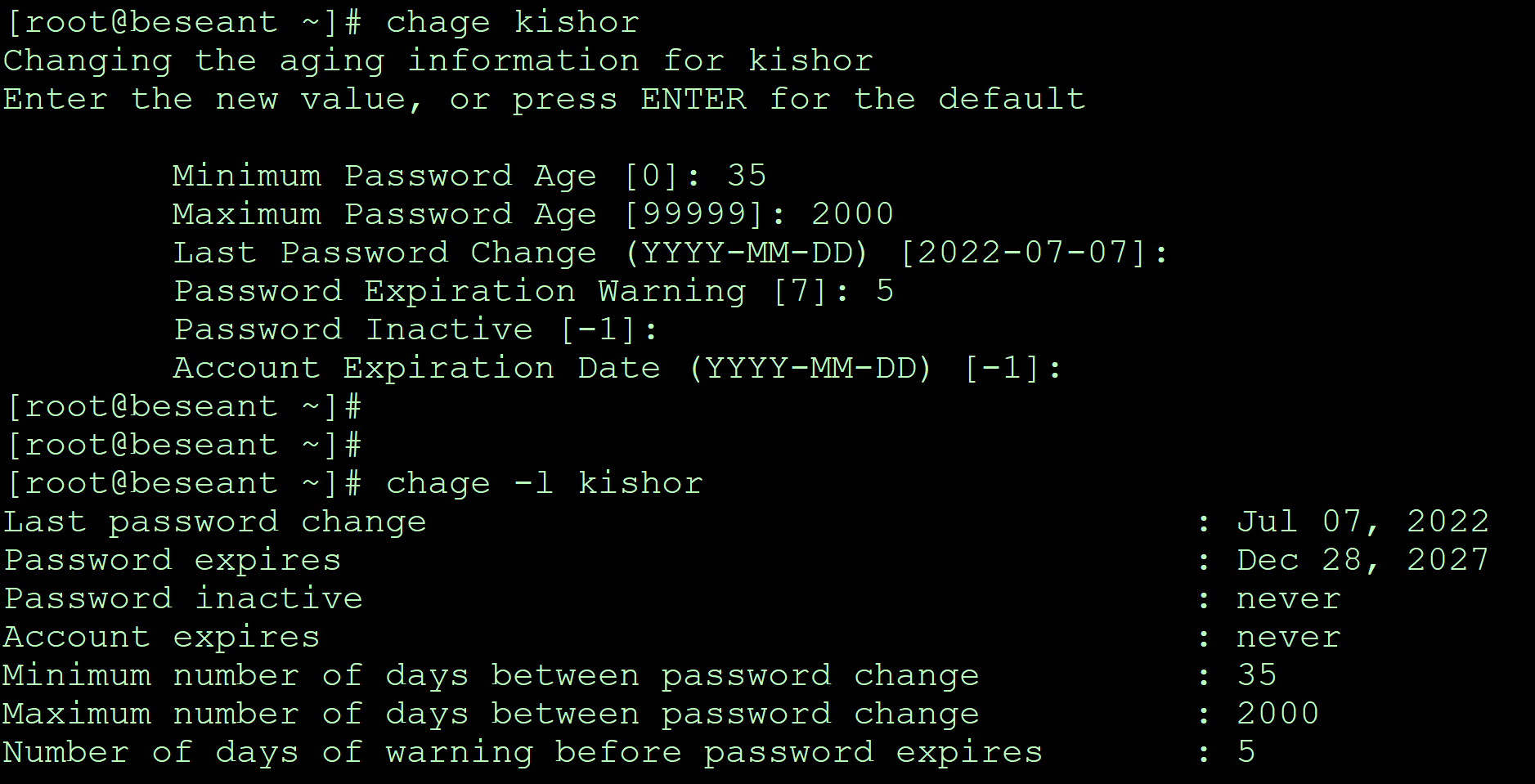
#### #chage -l < user name>

Change -l kishor



* **Last password change:** When the password was change last time.
* **Password expires:** Password expiry date
* **Password inactive:** After password expiry grace period before the account gets locked.
* **Account expires:** Date on which the account expires.
* **Minimum number of days b/w password change:** once the password is changed, it cannot be changed until a min period of specified date. **[0]** means never.
* **Max number of days b/w password change:** After changing the password how long it will be valid for.
* **Number of days of warning before password expires:** start of warnings to change the password, no. of days before the password expires.

If we want to modify the parameters of password age(min and max)



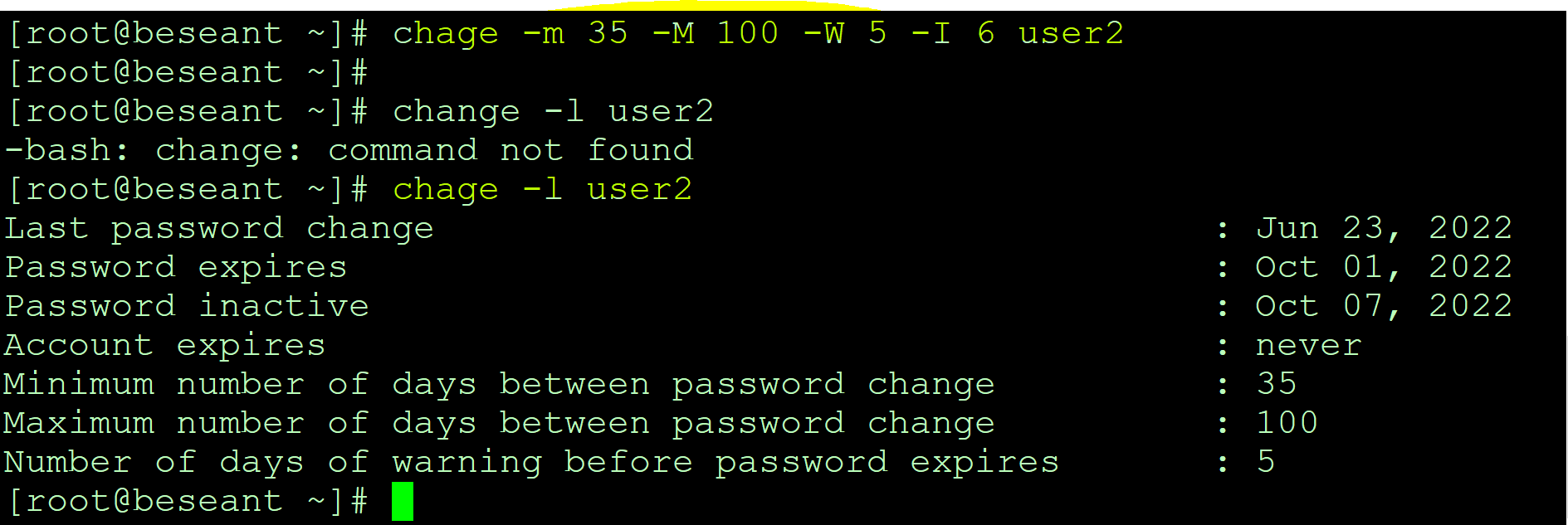
* The second method is for, if you want to change a particular field of password aging policy

#### #chage <option> <value> <username>

* **The options which can be used are as follows**
  + **-m** for Min password age
  + **-M** for Max password age
  + **-d** for last time the password is changed.
  + **-W** Password expiration warnings
  + **-I Password inactive** [-1 means inactive]**.**

#### -E A/C expiration date

* Let’s see how to change only the account expiration date



Default file: /etc/login.def

**Deleting a User:**

* To delete a user the syntax used is
* **#userdel <username>** it will only delete the user but home directory will be there. To delete the user with its home directory use the following command.

#### #userdel –r < user name >

* **#userdel –r kishor**
* **Remove user: gpasswd -d user4 linuxbatch**

**Monitor users:**

Who: who all are opened the terminal?

Last: every user logged in and, what time logged in, IP and reboot all can capture

W: who and w both same, here ideal time, cpu, Ip and process can be show here.

Id: it will give uid and gid and all groups which user belongs.

**System utility commands:**

* date
  1. timedatectl list-timezones
  2. sudo timedatectl set-timezone Asia/Kolkata
* uptime
* hostname
* uname
* which

which touch, pwd,

* cal

cal 1999, cal 8 1947

* bc

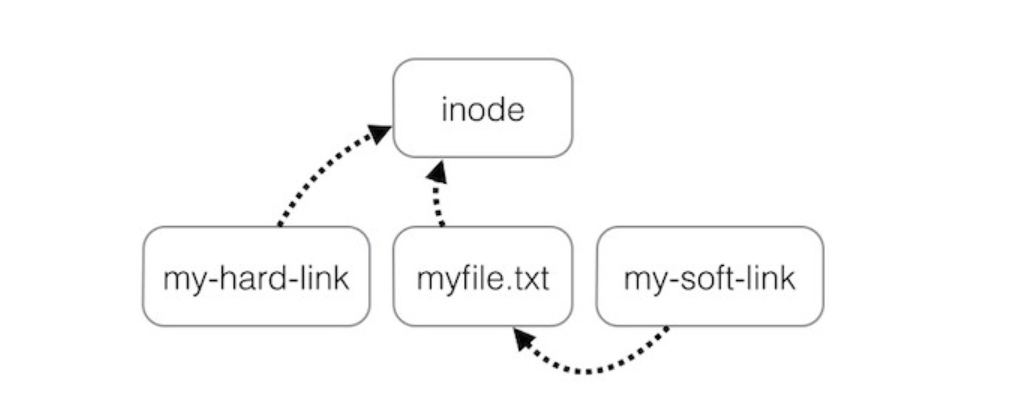
calculator and do 34\*333 and quit

**Symbolic Link**

**There are two types of Links:-**

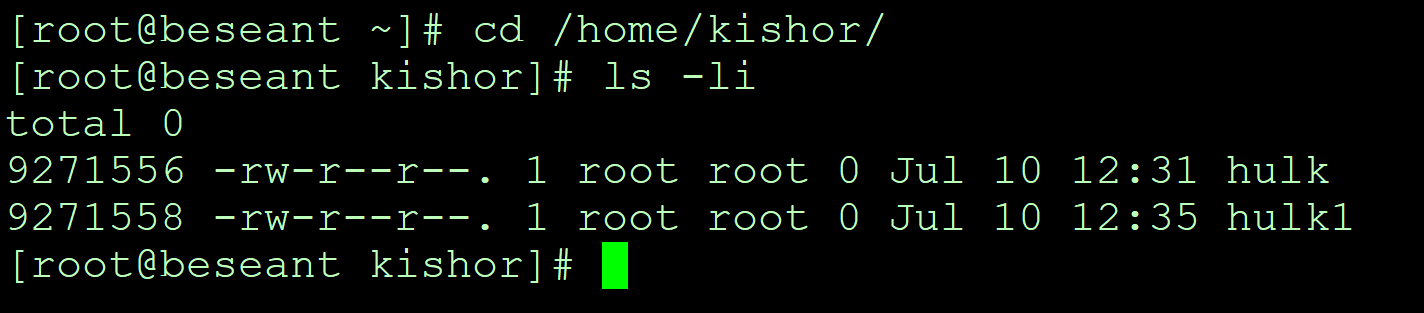
|  |  |  |
| --- | --- | --- |
|  | **Soft Link** | **Hard link** |
| **1** | **Size of link file is equal to no. of**  **characters in the name of original file** | **Size of both file is same** |
| **2** | **Can be created across the Partition** | **Can't be created across the partition** |
| **3** | **Inode no. of source and link file is**  **different** | **Inode no. of both file is same** |
| **4** | **if original file is deleted, link is broken**  **and data is lost** | **If original file is deleted then also link**  **will contain data** |
| **5** | **SHORTCUT FILE** | **BACKUP FILE** |

**what is soft link and hard link.:**



**Creating a soft link:**

**# ln –s <source file> <destination>**



Ex: touch hulk (created in /home/kishor)

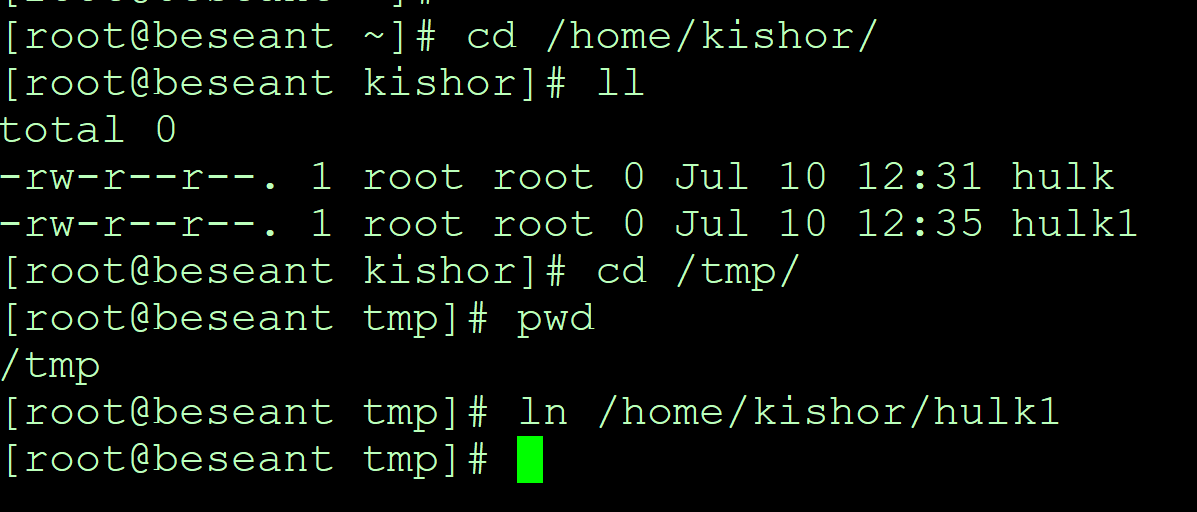
Cd /tmp

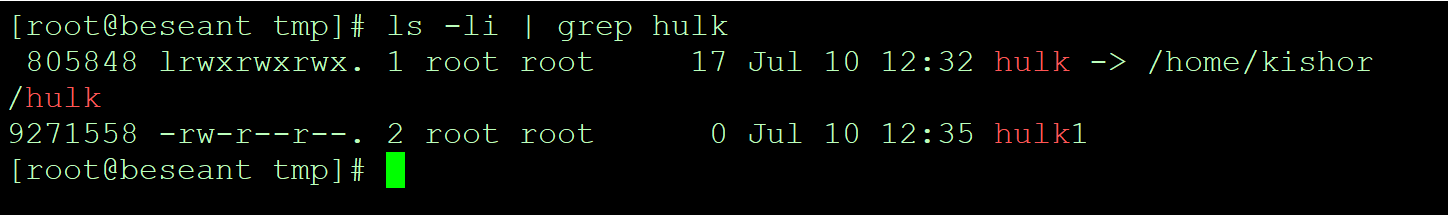
ln -s /home/kishor/hulk

**Creating a Hard link:**

**#ln <source file> <Destination>**

hard link: ln /home/kishor/hulk





**AWK (Text processor command):**

Awk is a utility/language designed for data extraction. Most of the time it is used to extract fields from a file or from an output.

Awk –version

awk ‘{print $1}’ file (print 1st coloumn)

ls -l | awk ‘{print $1, $3}’ (output will give 1st and 3rd column. )

ls -l | awk ‘{print $NF}’ (last column output)

awk '/searchingword/ {print}' file

awk ‘/home/ {print}’ file ( search for specific word )

awk -F: '{print $1}' /etc/passwd (separated with : and print 1st column )

and awk -F: '{print $6}' /etc/passwd (if want only user home directory)

echo “Hello tom” | awk ‘{$2=”kishor”; print $0’)’ (replace words field words and $0 for print all information)

cat file | awk ‘{$2=”kishor”; print $0}’ (replace 2nd column with kishor)

awk ‘length($0) > 15’ file (Get the lines which we have more than 15 byte size )

ls -l | awk '{if($9 == "folder2") print $0;}' (the only matching file can check and get output)

**grep and egrep:**

grep –version

syantax: grep keyword file

eg: grep kishor /etc/passwd

grep -c kishor /etc/passwd (c will number of counts)

grep -i kishor /etc/passwd (-i ignore for uppercase and lowercase )

grep -n keyword file (it will give exact matching lines)

ls -l | grep beseant (beseant will search first output )

egrep -i “keyword|keyword2” file (search for 2 keywords in same time)

**sort and uniq commands:**

sort:It is used to sort the output in numeric or alphabetic order

sort filename

sort -r filename (to reverse order)

**sort –u names (to remove dumplicates)**

**sort filename | uniq -c (it will show how many counts are available)**

**sort filename | uniq -d (it will show duplicate lines)**

**uniq:** uniq commands is filters and removed the duplicated lines

uniq filename

**WC (word count)**

The command reads either standard input or a list of files and generates: newline count, word count, and byte count.

wc –version

wc file

wc -l file (it will give lines count)

wc -w file (it will give word count)

wc -b file (it will give byte count)

ls -l | wc -l

**compare files (diff and cmp)**

compare files in linux

diff (Line by line)

syntax: diff file1 file2

cmp (byte by byte)

syntax: cmp file1 file2

**compress and un-compress files:**

tar

gzip

gzip -d or gunzip

#### 2.To back up the file or compressing will using tar

Syntax: tar –cvf <destination and name to be > < source file>

Ex: tar –cvf /opt/etc.tar /etc

1. Check the size of tar file by using du –h <file name > command

#### #du –h /opt/etc.tar

1. Apply gzip on tar file and check the size.

* To apply gzip on a tar file, the syntax is

#### #gzip <file name> #gzip /opt/etc.tar

1. Un-compressing

Syntax: gzip -d etc.tar.zip

To gunzip a file the syntax is: #gunzip <file name>

Ex: #gunzip etc.tar.gz

1. Untar the file and check for the size of the file/directory

To untar a file the syntax is

#tar -xvf <file name>

Ex: tar –xvf etc.tar

**Combining and splitting files:**

Cat file1 file2 file3 > file4

Split file4

Ex: split -l 300 file.txt childfile

Split file.txt created 300 lines into one file and output to a childfileaa, childfileab and childfileac

Disk Partion and LVM

**Networking, Services and system updates:**

**Network Configuration & Trouble Shooting**

**Networking:**

It is a connection between two or more machines to communicate with each other.

**The basic requirements for Networking are:**

1. **NIC (Network Interface Controller or Card)**
2. **Media**
3. **Topology**
4. **Protocol**
5. **IP Addresses**
6. **NIC (Network Interface Controller or Card)**

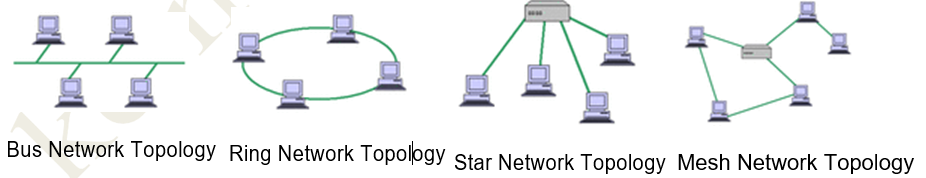
A **network interface controller** (also known as a **network interface card**, **network adapter**, **LAN adapter** and by similar terms) is a computer hardware component that connects a computer to a computer network. Each NIC will be having a unique MAC addresses (**Media Access Control address)** to avoid conflicts between same NIC adapters. In Linux these NIC adapter is represented by the word “**eth”.** Example if there are two Ethernet adapters in the system then it will be denoted as eth0, eth1, etc.

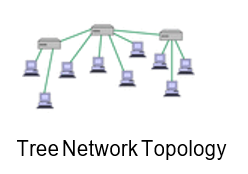
1. **Media**

Media is the medium via which two different computer’s NIC card will be connected. The best example for media is Cable. Example **RJ 45, CAT 5** etc.

1. **Topology**

Topology is the scheme or design in which the computers in the network will be connected to each other. Example for topology is Bus, Ring, star, mesh, tree topologies. The following pictures explain it better.





1. **Protocol**

A **network protocol** defines rules and conventions for communication between network devices. Protocols for computer networking all generally use packet switching techniques to send and receive messages in the form of *packets*.

Network protocols include mechanisms for devices to identify and make connections with each other, as well as formatting rules that specify how data is packaged into messages sent and received. Some protocols also support message acknowledgement and data compression designed for reliable and/or high-performance network communication. Hundreds of different computer network protocols have been developed each designed for specific purposes and environments.

Example for Protocols are **TCP/IP (Transmission Control Protocol), UDP (User Datagram Protocol), HTTP.** The most widely and regularly used protocols for transferring data are TCP and UDP. Let’s analyze some basic differences between **TCP/IP** and **UDP.**

|  |  |
| --- | --- |
| **TCP/IP** | **UDP** |
| Transmission Control Protocol | User Datagram Protocol |
| It is connection Oriented | Connectionless |
| Reliable | Non-Reliable |
| TCP Acknowledgement will be sent/received | No Acknowledgement for UDP |
| Slow Communication | Faster Communication |
| Protocol Number for TCP is **6** | Protocol Number for UDP is **17** |
| HTTP, FTP, SMTP uses TCP | DNS, DHCP uses UDP |

1. **IP ADDRESS**

An IP address can be thought of as being similar to a phone number. Just as every person who communicates with a telephone is using a phone with a unique phone number, every computer that is on the Internet has a unique IP address. Not only on internet but within an organization every computer is assigned an IP address so that they can communicate with each other. Basically,32 IP addressing is very deep concept. To understand the concept of IP address we need to understand some important aspect of IP Address which is

* + IP Address Classes
  + Subnet mask
  + Gateway

The above concepts in IP Addressing are very important to understand networking clearly.

**IP Address classes:**

The IP addresses are further broken down into classes. These classes are A, B, C, D, E and their possible ranges can be seen in Figure below.



* + **Subnet Mask**

A subnet mask allows users to identify which part of an IP address is reserved for the network and which part is available for host use. By looking at the IP address alone, especially now with classless inter-domain routing, users cannot tell which part of the address is which. Adding the subnet mask or netmask gives users all the information needed to calculate network and host portions of the address with ease. In summary, knowing the subnet mask can allow users to easily calculate whether IP addresses are on the same subnet or not.

A commonly used netmask is a 24-bit netmask as seen below.

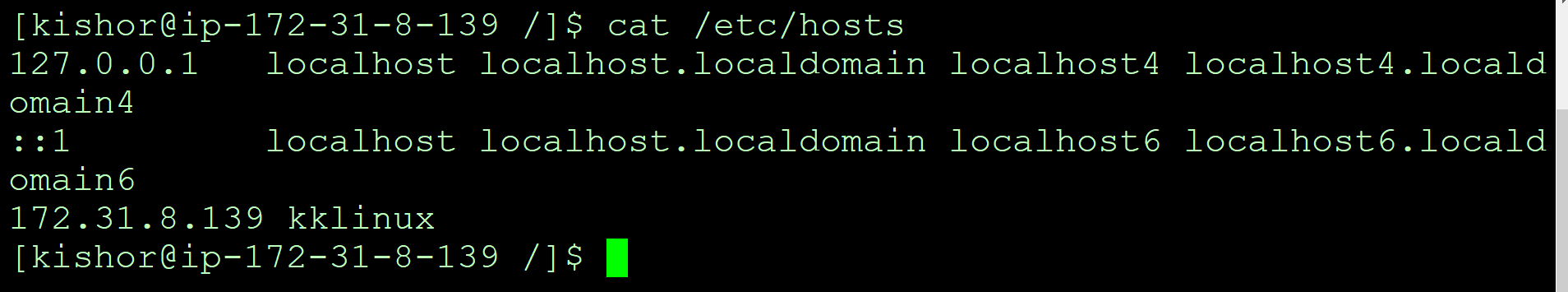
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Netmask: | 255. | 255. | 255. | 0 |
| Binary: | 11111111 | 11111111 | 11111111 | 00000000 |
| Netmask length | 8 | 16 | 24 | -- |

* + **Gateway**

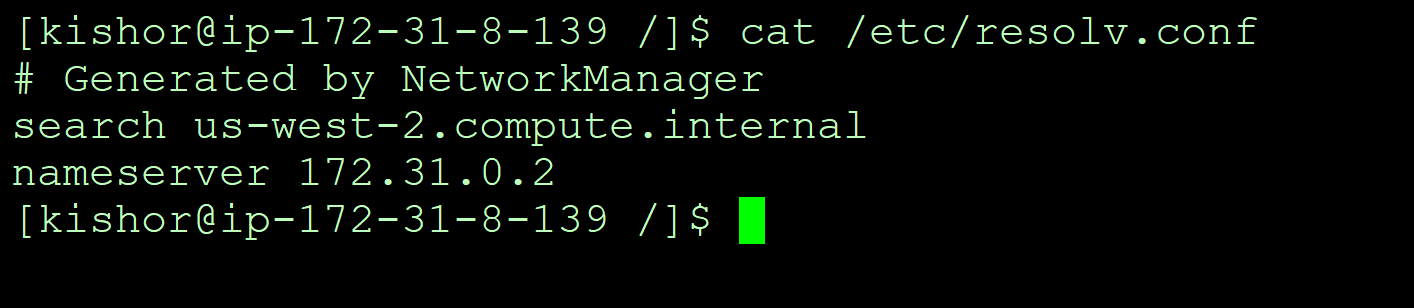
A gateway is a network point that provides entrance into another network. On the Internet, a node or stopping point can be either a gateway node or a host (end-point) node. Both the computers of Internet users and the computers that serve pages to users are host nodes. The computers that control traffic within your company's network or at your local Internet service provider (ISP) are gateway nodes.

For example, let’s say our network is 192.168. something and we want to send a file to other computer on 10.10.network, so we need a gateway to communicate between two computers of different networks.

**#/etc/hosts** a file which is responsible for resolving hostname into IP locally, in other word it acts as local DNS if DNS server is not accessible.

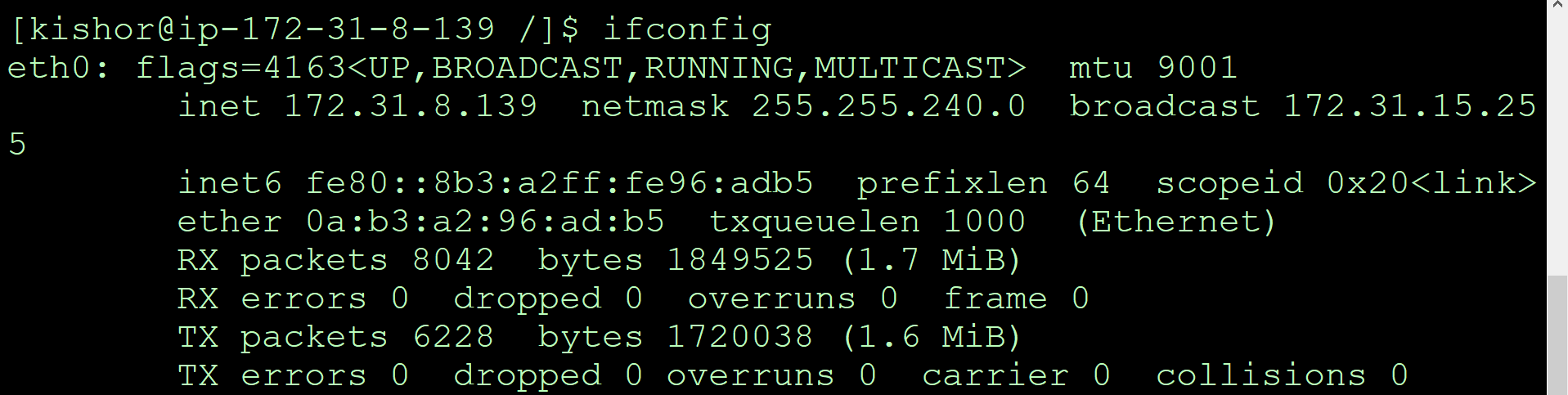


**#/etc/resolv.conf** is a file which keeps the address of DNS server to which the clients will be accessing to resolve IP to hostname and hostname to IP.



LAB Work:

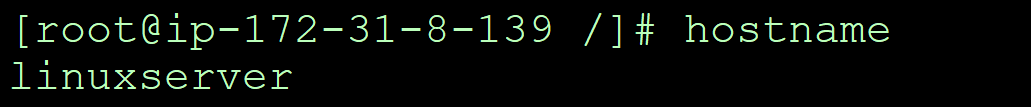
#### **To check the ip address assign to all the interfaces #ifconfig**

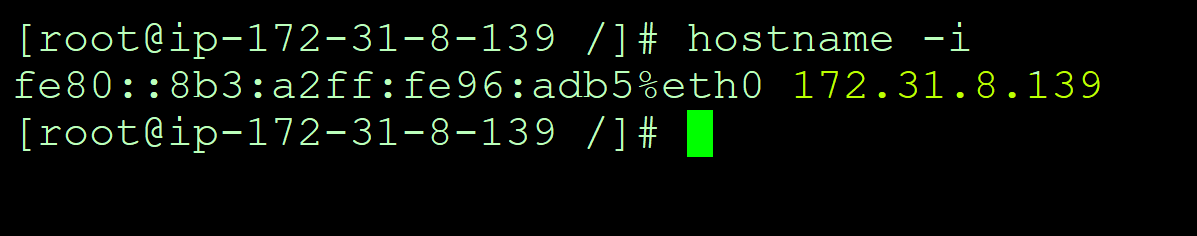


* **To chech the ip of a particular interface #ifconfig < adapter name >**

**#ifconfig eth0**

* **To check the hostname of the system. #hostname**

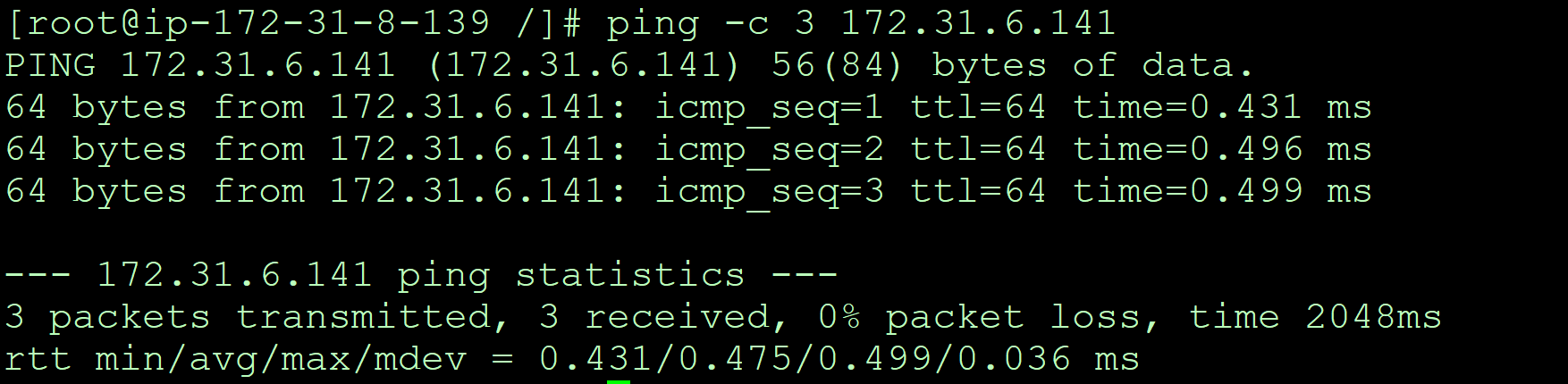


* **To check ip of the host #hostname –i**
* 
* **“nslookup” command #nslookup < ip address > #nslookup < hostname >**
* **Checking network connectivity using ping command #ping < ip address >**

**#ping 192.168.10.95**

* **To limit the pinging for specific number of counts #ping –c <counts> <ip address>**

**#ping –c 2 192.168.10.95**

* 

**Changing the hostname**

* Check the current hostname with **hostname** command
* The syntax for changing the hostname is

#### #hostname <new name> #hostname Linuxserver

This is temporary

Parment entry: /etc/hostname

***downloading files and apps***

***cmd: wget***

***wget https://website.com/file***

# SOFTWARE MANAGEMENT

**To manage the software in Linux, two utilities are used**,

1. **RPM – REDHAT PACKAGE MANAGER**
2. **YUM – YELLOWDOG UPDATER MODIFIED**

**RPM –REDHAT PACKAGE MANAGER**

RPM is a package managing system (collection of tools to manage software packages). RPM is a powerful software management tool for installing, uninstalling, verifying, querying and updating software packages. RPM is a straight forward program to perform the above software management tasks.

#### Features:

* + RPM can verify software packages.
  + RPM can be served as a powerful search engine to search for software’s.
  + Components, software’s etc can be upgraded using RPM without having to reinstall them
  + Installing, reinstalling can be done with ease using RPM
  + During updates RPM handles configuration files carefully, so that the customization is not lost.

### LAB WORK:-

**To check all the installed packages in the system**

* To check all the installed packages in the system, the syntax is
* **#rpm –qa** (where **q** stands for query, and **a** stands for all)

|  |
| --- |
|  |

#### Note:- The output of above command will be very lengthier.

**To check whether a particular package is installed or not**

* To check whether a package is installed or not out of the list of installed package, the syntax is

#### #rpm –qa <package name> or #rpm –q < package name>

**#rpm –qa vsftpd** or **#rpm –q vsftpd**

|  |
| --- |
|  |

* One more method of checking the installed package, when you are not sure about the package name, like whether it starts with capital letter and full name etc.

#### #rpm –qa | grep –i < package name> #rpm –qa |grep –i vsft\*

|  |
| --- |
|  |

**To check whether a package is consistent or not, before installing it. (Testing the installation)**

* To check the package’s consistency,
* Move to the directory where you have kept the rpm package which you wish to install

|  |
| --- |
|  |

* The command used to check the package’s consistency is

#### #rpm –ivh - -test <package name>

Where i = install, v= verbose view, and h = hash progress.

#### #rpm –ivh - - test finger-0.17-39.el6.i686.rpm

|  |
| --- |
|  |

**If the installation status shows 100%, then the package is good or consistent.**

**But while showing the hash progress if it shows any error, then the package is inconsistent.**

**To install a package using rpm command and check whether it is installed properly or not.**

* To install the package first we need to be in the directory of the package

|  |
| --- |
|  |

* To install the package the syntax is

#### #rpm –ivh <package name>

**#rpm –ivh finger-0.17-39.el6.i686.rpm**

|  |
| --- |
|  |

* To check whether it is installed or not

#### #rpm –qa finger

|  |
| --- |
|  |

* Check the installed package by using it command, finger is used to check user’s details.

#### 

**To remove a package or uninstall the package**

* To remove a package the syntax is

#### #rpm –e < package name> #rpm –e finger

Verify it by **#rpm –q or rpm –qa command**

|  |
| --- |
|  |

* To see the info of a package, the syntax is

**#rpm –qip <package name> (**where **q** is for query, **i** is for install and **p** is for package**) #rpm –qip finger-0.17-39-el6.1686.rpm**

|  |
| --- |
|  |

## To see the information about the installed package

* To see the information or details about the installed package, the syntax is

#### #rpm –qi < package name > #rpm –qi vsftpd

|  |
| --- |
|  |

**YUM – YELLOWDOG UPDATER MODIFIED**

* The Yellow dog Updater Modified (YUM) is a package management application for computers running Linux operating systems.
* Yum is a standard method of managing the installation and removal of software. Several graphical applications exist to allow users to easily add and remove packages; however, many are simply friendly interfaces with yum running underneath. These programs present the user with a list of available software and pass the user's selection on for processing. It is yum that actually downloads the packages and installs them in the background.
* Packages are downloaded from collections called **repositories**, which may be online, on a network, and/or on installation media. If one package due to be installed relies on another being present, this **dependency** can usually be resolved without the user needing to know the details. For example, a game being installed may depend on specific software to play its music. The problem of solving such dependencies can be handled by yum because it knows about all the other packages that are available in the repository.
* Yum will work only from Cantos 5 / Red hat 5 and latest versions of fedora. For Old releases like RHEL 4 you need to use up2date command to update your rpm based packages.
* Yum uses a configuration file at **/etc/yum.conf**

## Working with YUM commands.

#### To list the available packages in the repository

* **#yum list (** or) **#yum list all** (or) **#yum list |more (**to view line wise**)**

#### As we have seen first command, second will also gives exactly same output. let us see the third command

**#yum list |more**

|  |
| --- |
|  |

**To list all the installed packages in the system.**

* To view all the installed packages in the system, the syntax is

#### #yum list installed

|  |
| --- |
|  |

**To check a particular package is installed or not**

* To check whether a package is installed or not the syntax is

#### #yum list installed <package name> #yum list installed vsftpd

|  |
| --- |
|  |

**To install a package using yum**

* Installing a package using yum does not requires full package name as in the case of rpm, and it also automatically resolves the dependencies as well.
* The syntax for installing a package is

#### #yum install <package name>

**#yum install finger\* (where \* means anything with name “finger”)**

|  |
| --- |
|  |

**It will prompt you for y/n to continue, type y and continue installing the package**

* installing a package without being prompt for **y** or **n**, the syntax is

#### #yum install <package name > -y #yum install finger\* -y

|  |
| --- |
|  |

**To remove the package with yum command**

* To remove the package using yum command, the syntax is

#### #yum remove <package name> #yum remove finger -y

|  |
| --- |
|  |

**To update the package using yum**

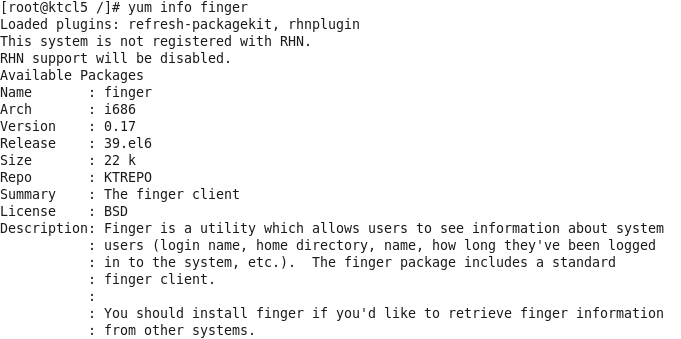
* To update the package using yum command, the syntax is

#### #yum update <package name> #yum update httpd

|  |
| --- |
|  |

**To see the information about the package #yum info <package name>**

**#yum info finger**



System upgrade and patch management:

Yum update -y

Yum update vs upgrade

Roll back changes:

Yum history undo “ID”

## Remote file transfer with SCP and RSYNC

**SCP (SECURE COPY protocol)**

* scp stands for secure cp (copy), which means that you can copy files across an ssh connection that will be encrypted, and therefore secured. As scp will be using ssh protocol to transfer the data, hence it is termed as the safest method of transferring data from one location to another.
* The syntax for SCP a file from source location.

#### # scp <file name > <remote hosts IP >:/<location to copy the file >

#### #scp file1 192.168.10.95:/home/kishor

Reverse:

scp <source system’s IP>:/<location of file to be copied> <destination location to copy>

#### scp 192.168.10.95:/home/kishor /tmp/

**RSYNC (REMOTE SYNCHRONIZATION)**

* rsync is a very good program for backing up/mirroring a directory tree of files from one machine to another machine, and for keeping the two machines "in sync." It's designed to speed up file transfer by copying the differences between two files rather than copying an entire file every time.
* For example, Assume that we are suppose to take the backup of a system and copy the same to another system. For first time we will copy entire directory, but every day if we copy entire directory it will kill lots of time. In such situation if rsync is used it will only copy the updated files/directories rather than copying all files/directories inside main directory, which saves lots of time and speedup the transfer
* If rsync is combined with ssh it makes a great utility to sync the data securely. If rsync is not used with ssh, the risk sniffing will always be there.

### LAB WORK:-

#### Copy a directory using SCP, then update it and try rsync with SSH and check if the data is synced.

* As we have already copy a directory earlier using SCP from **ktlinux** to **ktcl5** system, let’ s use it for rsync.
* Update the directory with some files in **ktlinux** system

|  |
| --- |
|  |

* Check the content of same directory in **ktcl5**

|  |
| --- |
|  |

* Use rsync to sync the directory on **ktcl5** machine, with the one in **ktlinux** machine
* The syntax to rsycn a directory is

#### #rsync <options> <encryption> <source dir> <destination IP>:/<location of destination dir>

**#rsync –rv -e ssh /ktdir 192.168.10.95:/root/**

|  |
| --- |
|  |

**Observe that it is only copying the files which are not there in destination’s folder.**

rsync -avzh /root/putty-0.77.tar.gz [kishor@172.31.24.188:/home/kishor/](mailto:kishor@172.31.24.188:/home/kishor/)

rsync -avzh /root/\* [kishor@172.31.24.188:/home/kishor/](mailto:kishor@172.31.24.188:/home/kishor/)

# JOB AUTOMATION

## Automation with cron and at

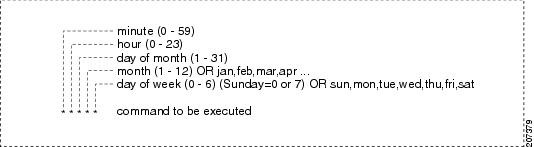
* In any operating system, it is possible to create jobs that you want to reoccur. This process, known as ***job scheduling***, is usually done based on user-defined jobs. For Red Hat or any other Linux, this process is handled by the cron service or a daemon called **crond**, which can be used to schedule tasks (also called *jobs*). By default, Red Hat comes with a set of predefined jobs that occur on the system (hourly, daily, weekly, monthly, and with arbitrary periodicity). As an administrator, however, you can define your own jobs and allow your users to create them as well.
* The importance of the job scheduling is that the critical tasks like taking backups, which the clients usually wants to be taken in nights, can easily be performed without the intervention of the administrator by scheduling a cron job. If the cron job is scheduled carefully than the backup will be taken at any given time of the client and there will be no need for the administrator to remain back at nights to take the backup.

**Important Files related to cron and at**

* **/etc/crontab** is the file which stores all scheduled jobs
* **/etc/cron.deny** is the file used to restrict the users from using cron jobs.
* **/etc/cron.allow** is used to allow only users whose names are mentioned in this file to use cron jobs. (this file does not exist by default)
* **/etc/at.deny** same as cron.deny for restricting at jobs
* **/etc/at.allow** same as cron.allow for allowing user to use at jobs.

**Crontab format**

* To assign a job in the Crontab file the format used is the following



|  |  |
| --- | --- |
| **Options** | **Explanation** |
| **\*** | **Is treated as a wild card. Meaning any possible value.** |
| **\*/5** | **Is treated as ever 5 minutes, hours, days, or months. Replacing the 5 with another numerical value will change this option.** |
| **2,4,6** | **Treated as an OR, so if placed in the hours, this could mean at 2, 4, or 6 o-clock.** |
| **9-17** | **Treats for any value between 9 and 17. So if placed in day of month this would be days 9 through 17. Or if put in hours it would be between 9 and 5.** |

**Crontab Commands**

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| **crontab –e** | Edit your crontab file, or create one if it doesn’t already exist. |
| **crontab –l** | Display your crontab file. |
| **crontab –r** | Remove your crontab file. |
| **crontab -u** | If combined with **–e,** edit a particular user’s Crontab file and if combined with **–l,** display a particular user’s crontab file. If  combined with **–r,** deletes a particular user’s Crontab file |

***LAB WORK:-***

**CRON JOBS:**

**To check the assigned cron jobs of currently logged in user**

* To check the cron jobs the command is

#### #crontab –l

|  |
| --- |
|  |

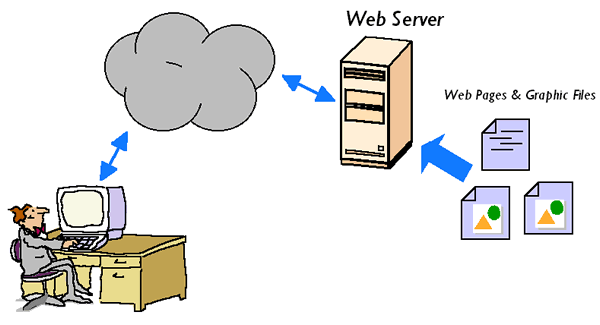
**To check the cron jobs of a particular user**

* To check a user’s cron jobs, the syntax is

#### #crontab –l –u <user name> #crontab –l –u ktuser

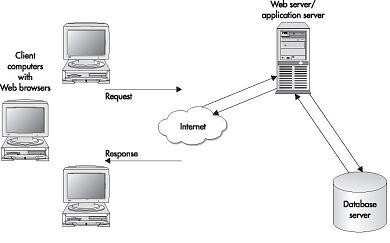
|  |
| --- |
|  |
|  |

# WEB SERVER (APACHE)



* + Every Web site sits on a computer known as a Web server. This server is always connected to the internet. Web servers are computers that deliver (serves up) Web pages. Every Web server has an IP address and possibly a domain name.
  + A web server can mean two things - a computer on which a web site is hosted and a program that runs on such a computer. So the term web server refers to both hardware and software.
  + A web server is what makes it possible to be able to access content like web pages or other data from anywhere as long as it is connected to the internet. The hardware houses the content, while the software makes the content accessible through the internet.
  + The most common use of web servers is to host websites but there are other uses like data storage or for running enterprise applications. There are also different ways to request content from a web server. The most common request is the Hypertext Transfer Protocol (HTTP), but there are also other requests like the Internet Message Access Protocol (IMAP) or the File Transfer Protocol (FTP).

## How a Web Server Works



#### A simple exchange between the client machine and Web server goes like this:

1. The client's browser dissects the URL in to a number of separate parts, including address, path name and protocol.
2. A Domain Name Server (DNS) translates the domain name the user has entered in to its IP address, a numeric combination that represents the site's true address on the Internet (a domain name is merely a "front" to make site addresses easier to remember).
3. The browser now determines which protocol (the language client machines use to communicate with servers) should be used. Examples of protocols include FTP, or File Transfer Protocol, and HTTP, Hypertext Transfer Protocol.
4. The server sends a GET request to the Web server to retrieve the address it has been given. For example, when a user types [http://www.example.com/1.jpg,](http://www.example.com/1.jpg) the browser sends a GET 1.jpg command to example.com and waits for a response. The server now responds to the browser's requests. It verifies that the given address exists, finds the necessary files, runs the appropriate scripts, exchanges cookies if necessary, and returns the results back to the browser. If it cannot locate the file, the server sends an error message to the client.
5. The browser translates the data it has been given in to HTML and displays the results to the user.

## Profile for Apache Server

#### Use : Hosting a web site.

**Package : httpd**

**Port : 80/http, 443/https Configuration Files : /etc/httpd/conf/httpd.conf**

**/etc/httpd/conf.d/ssl.conf (https)**

**Document Root : /var/www/html**

**Daemon : httpd**

**Script : /etc/initd/httpd**

**Steps to Configure a simple web server**

**Step1: Install the package**

* **The package for apache web server is httpd. #yum install httpd\* -y**

|  |
| --- |
|  |

**Step2: Navigate to /etc/httpd/conf/httpd.conf and edit it.**

* **Navigate to the configuration file for http i.e. /etc/httpd/conf/httpd.conf and copy the last 7 lines as shown below**

**#vim /etc/httpd/conf/httpd.conf**

|  |
| --- |
|  |

**Copy these lines and paste it at the end of the page, then edit it with your preferences.**

**Step2: Navigate to the document root folder i.e. /var/www/html/ and create an index.html file which will be accessed through a web browser**

* **#vim /var/www/html/index.html**