**Table of Contents**

**Red Hat Enterprise Linux Network**

**Red Hat Linux Essentials**

**Unit 1 - Linux History, Ideas and Releases**

What is Linux OS 2

Linux History & Ideas 2

Linux Vendors and Distributions 3

Free Linux Distributions 3

Red Hat Linux Variants 5

**Unit 2 - Linux Usage Basics**

Desktop Environment 7

Red Hat Desktop Environment 8

Accessing the Linux Machine 23

**Unit 3 - Basic Commands of Linux**

Basic commands 25

Working With Files & Directories 26

**Unit 4 - Text Processing Tools**

Basic Commands 31

I/O Redirection 31

Finding Files in Linux 42

**Unit 5 - Basic Linux Networking Tools**

Hostname and FQDN Settings 45

Local Resolver 46

Network Clients 47

SSH & Telnet 48

Rsync 49

**I. Linux History, Ideas and Releases**

**What is Linux OS?**

Linux (often pronounced LIH-nuhks with a short "i") is a Unix-like operating system that was designed to provide personal computer users a free or very low-cost operating system comparable to traditional and usually more expensive Unix systems. Linux has a reputation as a very efficient and fast-performing system.

**Linux History & Ideas:**

* **1984:** The GNU Project and the Free Software Foundation.

Creates Open Source Version of Unix utilities. Creates the General Public License (GPL) Software License enforcing Open Source Principles.

* **1991:** Linus Torvalds

Creates Open source, Unix-like Kernel, released under the GPL.Posts some GNU utilities , Solicits assistance online

**Today:**

Linux kernel + GNU utilities = complete, open source, UNIX-like operating system.

Packaged for targeted audiences as distributions.



**fig:** Linus Torvalds

**Linux Vendors and Distributions:**

If you're running a Linux OS on enterprise servers, support matters. There is no single best Linux distribution for every enterprise's servers. It all depends on what your company needs. Today, Linux is more than a free OS to mess around with -- it runs core business applications. When comparing the most popular Linux distributions, corporate Linux users care about support throughout the stack, not just an attractive feature set.

Server administrators worry about the maintenance lifetime and support on an OS. Maintenance lifetime is the length of time that the Linux distribution provider patches and updates the product. Support comes in three different forms: support for applications, support for hardware and support for troubleshooting. Linux users in an enterprise data center willingly pay for the best maintenance and support.

**Free Linux distributions:**

Any company can run a free and open source Linux distribution in an enterprise environment, but most have serious restrictions. In particular, the maintenance lifetime of most free distributions is limited. The popular OpenSUSE distribution, for instance, has an 18-month lifecycle, meaning that a security issue that occurs 24 months after the initial release date will go un-patched. And it takes about 18 months to get a Linux distribution approved for use in the corporate world. Distributions with such a short life cycle aren't even worth considering.

Also avoid distributions with maintenance periods that might change. An enterprise-grade Linux distribution needs to be well organized. If it only relies on an open source project, the project might split up or be acquired by a vendor, eliminating your support lifetime.

Settle your software differences -- ask for support

The popular supported enterprise Linux distributions are Red Hat Enterprise Linux (RHEL), Ubuntu Long Term Support (LTS) from Canonical, SUSE Linux Enterprise Server (SLES) and Oracle Unbreakable Kernel.

It's not really about software differences when selecting the best Linux distribution for your organization. All Linux operating systems use more or less the same open source components, especially where it really matters.

What does make a difference is support on these features. Verify that the main features your enterprise uses are supported by -- not just technically feasible on -- the distribution. For example: Your server infrastructure relies on Linux container virtualization (LXC).SLES offers LXC to run many instances of the operating system on the same kernel. LXC isn't supported on RHEL, so even if technically you can run it, Red Hat would not offer support for the containers.

An enterprise Linux distribution must offer stable software for the version your organization uses. Check the distribution's website to see if the latest version of a software package is included by default. These lists change frequently so check whenever you are making an OS decision, even if you checked the lists a few months prior.

Enterprise Linux distribution pricing can be confusing: You don't pay for the software, but for the support, which differs for each distribution. Ubuntu offers free patches, whereas other vendors charge for such support. All vendors offer a base support package through full-scale premium support. Oracle or Red Hat support tend to cost the most, but the price you pay in the end depends on many circumstances.

Red Hat Enterprise Linux. Red Hat has an estimated market share between 65% and 80% of enterprise distributions. RHEL entered many data centers via support deals with hardware and software vendors, an approach emulated by other Linux distributions.

The company employs more developers than other supported-Linux vendors. Therefore, Red Hat plays an important role in many open source projects. If your company wants to adopt cutting-edge open source platforms, it makes sense to standardize on Red Hat.

Red Hat offers many products that can be used to provide a complete open source infrastructure, notably Red Hat Enterprise Virtualization and Red Hat Cloud. Red Hat also provides the JBoss middleware platform, which corporate IT teams use to develop applications.

Red Hat makes it hard to access its software for free, so it has spawned rip-off distributions like CentOs and Scientific Linux. Now that Red Hat controls CentOS, its future as a free recompiled Red Hat operating system is uncertain.

SUSE Linux Enterprise Server. SUSE estimates that about 25% of corporate Linux users run its SLES distribution, making it the second-largest supported Linux OS.

SUSE focuses on specific business verticals and partners with other major industry vendors like SAP and VMware. VMware customers get SLES free with VMware ESXi, and SUSE is the preferred platform for SAP. Microsoft also endorses SLES for its customers that need to use Linux, and has renewed that support through 2016. That interoperability partnership started in 2006 and led to Microsoft introducing SUSE Linux to its customer Wal-Mart Stores Inc.

From a technical perspective, SUSE Linux is more accessible than the other supported Linux distributions. SUSE is also the most administrator-friendly. Its integrated YaST platform makes complicated management tasks on Linux easy to perform. SUSE is the main developer on Pacemaker High Availability, the standard tool for high-availability clusters in all current Linux distributions.

Unlike Red Hat, SUSE doesn't offer its own virtualization platform. It does offer SUSE Cloud, which is based on OpenStack, and SUSE Manager, which helps patch and manage dozens of SUSE and Red Hat deployments from a centralized interface.

Ubuntu LTS. Ubuntu started as a free distribution. Canonical, the company backing Ubuntu, came along later to offer professional services around the distribution. Companies that want to run Ubuntu LTS in a supported environment should contract with Canonical, but those that are solely interested in using the software and getting updates for a guaranteed amount of time can download and install Ubuntu for free.

When installing Ubuntu as a server platform, use the Long Term Support version. Patches will be available for at least seven years for LTS, where the availability of patches for regular versions, which come out on a faster update cycle, is much more limited.

Ubuntu's desktop OS is famously easy to use, but that is not so with the server version. Most management tasks are performed the hard way, by modifying configuration files, running commands and starting processes. For some users, this is a real disappointment.

Ubuntu is based on Debian Linux, a distribution popular with developers and in educational environments. Debian doesn't offer enterprise support, but by using Ubuntu LTS, customers can get it anyway.

Oracle Linux. Oracle's Unbreakable Kernel started by modifying the open source RHEL software to make a platform for Oracle databases.

Apart from companies that are running Oracle databases, Oracle's distribution hasn't been a huge success. It typically comes into the data center if the database administrator decides which Linux OS to use. Even still, many companies already have a Linux policy and a standard distribution that predates Oracle's open source initiative. This means plenty of Oracle databases still run on SUSE and Red Hat distributions, where it is supported.

Apart from Oracle VM, a Xen-based virtualization platform, Oracle doesn't have many open source products to complete its platform. Oracle also owns the Solaris operating system, which makes Oracle Linux less important.

**Red Hat Linux Variants:**

* **Red Hat Enterprise Linux Server**

Red Hat Enterprise Linux Server is a versatile platform that can be deployed on physical systems, as a guest on the major hypervisors, or in the cloud.Subscriptions are available with flexible guest entitlements of one, four, or unlimited guests per physical host.

* **Red Hat Enterprise Linux for IBM System z**

Red Hat Enterprise Linux, deployed on IBM System z, provides flexibility, performance, and scalability for server consolidation or for mainframe reliability. Built on a decade-plus industry collaboration, Red Hat Enterprise Linux for IBM System z brings Linux applications to the mainframe, and delivers efficient management for your Linux workloads.

* **Red Hat Enterprise Linux for IBM POWER**

Deploy applications with confidence using the advanced features in IBM Power Systems and the consistency and flexibility of the market-leading Linux operating platform. Standardizing on Red Hat Enterprise Linux across your x86, Power, and Mainframe systems reduces complexity for operations and development - without compromising reliability. Applications can be responsive to business needs with Red Hat Enterprise Linux and support for the latest features in POWER7 and LPAR hardware partitioning.

* **Red Hat Enterprise Linux for SAP Business Applications**

Red Hat Enterprise Linux for SAP Business Applications includes the entire software stack required for optimal operation of SAP applications, as well as enterprise-class service and support.

* **Red Hat Enterprise Linux Desktop**

Designed for task workers. Typically requires a limited set of applications. End user has limited administrative control over the system. System administrators need to be able to pre-configure and remotely manage consistent images across a large number of systems.Primary applications in use are productivity applications like Firefox, Evolution/Thunderbird, Open Office, Planner/Task Juggler. Primary services in use would be network file systems like SMB/CIFS, NFS, printing services, IM.

* **Red Hat Enterprise Linux Workstation**

Workstations are designed for advanced Linux users. In addition to the tools provided in the Desktop variant, the Workstation variant supports a stand-alone development environment. End users are expected to have local super-user privileges or select super-user privileges.

***$cat /etc/issue*** *:- To know the OS type and release of the Operating system.*

**II. Linux Usage Basics**

**Desktop Environment:**

There are two kinds of Desktops available in Red hat Linux. 1. **GNOME** 2. **KDE**

GNOME, KDE and Xfce are the most popular desktop environments for Linux. Most people end up using the default desktop environment that ships with their preferred Linux distribution. But experienced users prefer one over the year for either usability, performance, design or customizability considerations. GNOME began as a project to develop a free and open-source desktop environment and corresponding applications in August 1997. Its design philosophy can be best described as streamlined and easy to use.

KDE community started in October 1996. Its design philosophy is dedicated to functionality and expansion of its features.

**Difference between GNOME and KDE**

|  |  |  |
| --- | --- | --- |
| Introduction (from Wikipedia) | GNOME (GNU Network Object Model Environment) is a desktop environment—a graphical user interface that runs on top of a computer operating system—composed entirely of free and open source software. | KDE is a desktop environment for an integrated set of cross-platform applications designed to run on Linux, FreeBSD, Microsoft Windows, Solaris and Mac OS, designed by the KDE Community. |
| Focus | Free software | Free software |
| Developer | The GNOME Project | KDE |
| Website | GNOME.org | kde.org |
| Founded | 1999 | 1996 |
| Current Stable Release | 3.10 (September 26, 2013) | 4.11 (August 14, 2013) |
| Minimum System Requirements | 700 Mhz CPU, 768 MB RAM | 1 Ghz CPU, 615 MB RAM |
| Design Goals | A top-to-bottom free desktop environment designed for simplicity, accessibility, and ease of  internationalization and localization. | A one-stop, integrated desktop environment; carrying out day-to-day tasks without reliance on command-line interface. |
| Motto | An intuitive and attractive desktop for users | Experience Freedom! |
| Toolkit | GTK+ | Qt |
| Founder(s) | Miguel de Icaza and Federico Mena | Matthias Ettrich |

***Table:****Differences between GNOME and KDE*

**Red hat Linux Desktop Installation:**

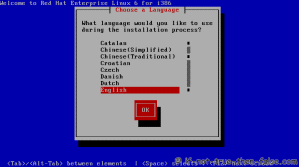
**Before we start the installation of Linux, we have to follow below steps:**

1. Insert the CD/DVD of the OS (Red hat Linux ) and reboot the system.
2. Hit on DEL or F2 button to go into the BIOS setup.
3. In BIOS select the First boot device as CD/DVD and save and exit by pressing the F10 key.
4. It will load the OS and follow below steps.

1**. Select *Install or upgrade an existing system* option on Grub Menu**



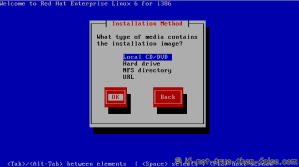
**2. Choose a language**



**3. Choose a keyboard type**



**4. Choose a installation media**

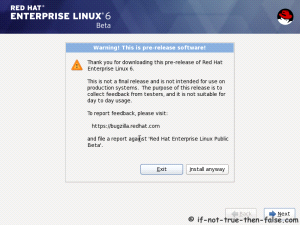
**5. Skip DVD media test (or select media test, if you want to test installation media before installation)**



**6. Red Hat 6 graphical installer starts, select next**



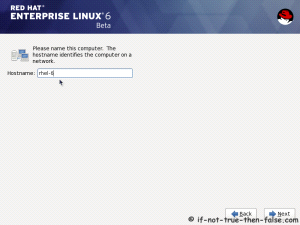
**7. Accepct Pre-Release Installation**



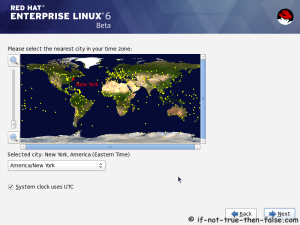
**8. Select storage devices**



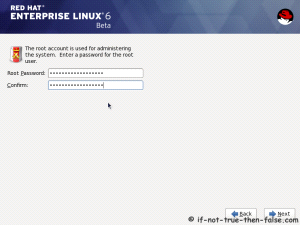
**9. Insert computer name**



**10. Select time zone**



**11. Enter a password for *root* user**



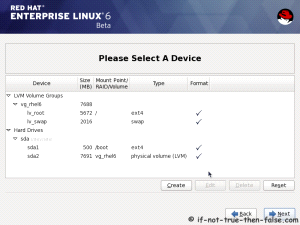
**12. Select type of installation**

Read every options info carefully. And select encrypting if needed and option to review and modify partition layout.

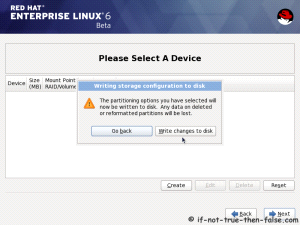


**13. Review partition layout**

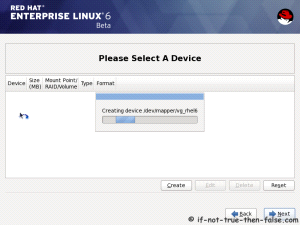
Modify if needed. Default setup with ext4 and LVM looks good for desktop machine.



**14. Accept write changes to disc**

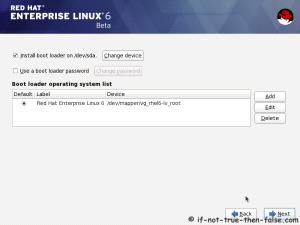


**15. Writing changes (creating partitions) to disc**



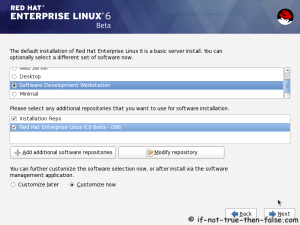
**16. Configure boot loader options**

Select device to install bootloader and check/create boot loader operating system list.



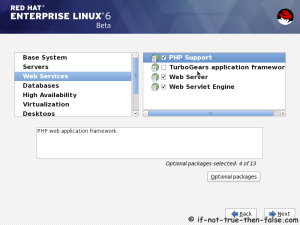
**17. Select softwares to install and enable repositories**

This case we select *Software Development Workstation* and enable Red Hat Enterprise Linux 6.0 Beta Repository and select Customize now.

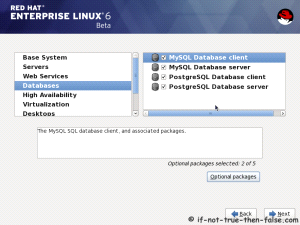


**18. Customize package selection**

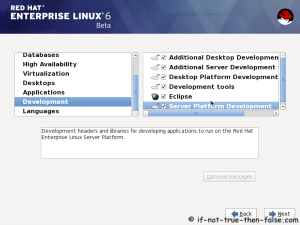
Select PHP and Web Server to installation.



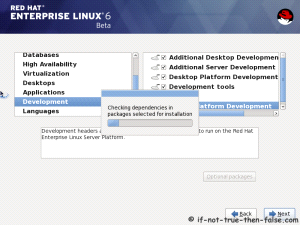
Select MySQL and PostgreSQL Databases.



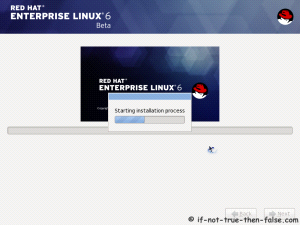
Select set of Development tools like Eclipse IDE.



**19. Checking dependencies for installation**

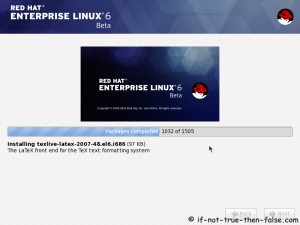


**20. Starting installation process**



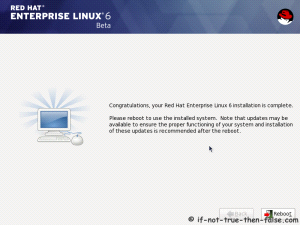
**21. Installing packages**





**22. Installation is complete**

Click reboot computer and remove installation media.



Red Hat 6 RHEL Finishing Installation

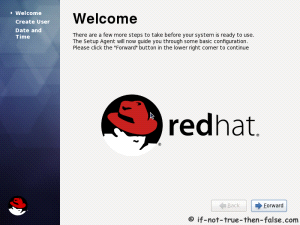
**23. Selecting RHEL 6 from grub**



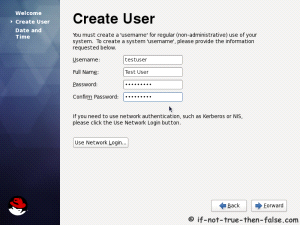
**24. Booting Red Hat 6**



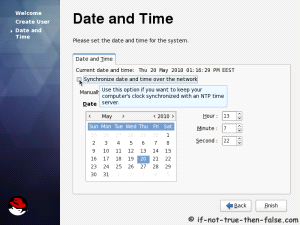
**25. Red Hat 6 Welcome screen**

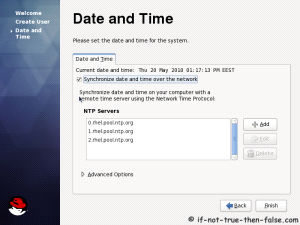


**26. Create normal user**

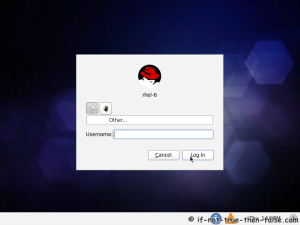


**27. Setup date and time and keep up-to-date with NTP**

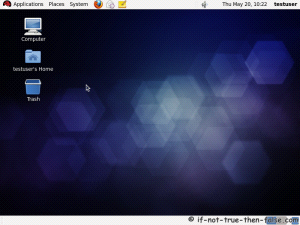




**28. Login Red Hat 6 Gnome Desktop**



**29. Red Hat (RHEL) 6 Gnome Desktop, empty and default look**



**Switching Between Virtual consoles and Graphical Console:**

In Linux, there are 6 virtual and 1 Graphical console. We can switch between by using CTRL+F1 to CTRL + F7. CTRL + F7 is for Graphical Console.

**Accessing the Linux Machine:**

We can access the Linux machine by using two users.

**1. root user :** We can call root as administrator user or super user. When we access the machine with root user, we can get # login prompt. root is having full permissions and access the machine and he can have full control over the system.

**root user id is 0.**

**2. Normal user :** When we login as normal user, we can we get the $ prompt and normal user is having limited permission to access the system.

**Normal user’s user ids will be start from 500**.

**Switching Between Users:**

We can switch users by using su command.

***$su root :*** switching user as root without changing his environment .

***$su - root :*** switching user as root with new login environment.

***$id :*** Print real and effective user and group IDs

Ex: ***$ id***

uid=1001(yugandhar) gid=1001(yugandhar)

groups=1001(yugandhar),4(adm),27(sudo),108(lpadmin),124(sambashare)

**The hyphen (-) has two effects:**

1) switches from the current directory to the home directory of the new user (e.g., to /root in the case of the root user) by logging in as that use

2) changes the environmental variables to those of the new user as dictated by their ~/.bashrc. That is, if the first argument to su is a hyphen, the current directory and environment will be changed to what would be expected if the new user had actually logged on to a new session (rather than just taking over an existing session).

**To open the terminal :**

**Applications -> System Tools -> Terminal.**

* **history** : It is used for to see the history of the executed commands on the terminal.

***#echo $HISTSIZE*** : We can see the history size

We can set the history size in the file .bashrc for user and /etc/bashrc is for entire system.

***#echo $HISTFILE*** : It displays the history file size.

* ***<TAB>*** : tab key will complete you command while executing the commands.

Ex:- $ifco<tab> = $ifconfig

**Getting Help:**

There are few commands to help you about to know the commands.

* **whatis** - display one-line manual page descriptions (Short description of a command).

Syntax : ***$whatis*** <command>

Ex:- ***$ whatis*** *date*

date (1) - print or set the system date and time

* **man** - an interface to the on-line reference manuals.

man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed.

The table below shows the section numbers of the manual followed by the types of pages they contain.

1 Executable programs or shell commands

2 System calls (functions provided by the kernel)

3 Library calls (functions within program libraries)

4 Special files (usually found in /dev)

5 File formats and conventions eg /etc/passwd

6 Games

7 Miscellaneous (including macro packages and conventions), e.g.

man(7), groff(7)

8 System administration commands (usually only for root)

9 Kernel routines [Non standard]

Syntax : ***$man*** *<command>*

*Ex:-* ***$man*** *cal*

* **info** : read info document.

Syntax: ***$info*** < command>

Ex:- ***$info*** date

* **help:** By using the help also, we can get the help about a command

Syntax:- ***$<command> --help***

Ex:- $date --help

**III.BASIC COMMANDS of Linux**

* **pwd:** The Linux pwd command show the name of the present working directory

Example:

***[root@localhost bin]# pwd***

/usr/bin

* **date:** The Linux date command print or set the system date and time

Example:

***[root@localhost bin]# date***

Sat Dec 26 11:49:39 IST 2009

* **cal:** Displays a calendar and the date of Easter

***$ cal***

January 2015

Su Mo Tu We Th Fr Sa

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30 31

If you want see the complete year calendar,

Syntax: ***$cal 2015***

If you want see the specific month of a particular year

***$ cal oct 2015***

October 2015

Su Mo Tu We Th Fr Sa

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30 31

* **logname:** Prints the login name of the current user.

***# logname***

root

* **whoami:** The Linux "whoami" command displays who is the current user. Print the user name associated with the current effective user ID

Example:

***[root@RHEL01 ~]# whoami***

root

* **who am i :** This command will let you who you are and logged in time.

***# who am i***

yugandhar pts/7 2015-01-19 15:23 (:0)

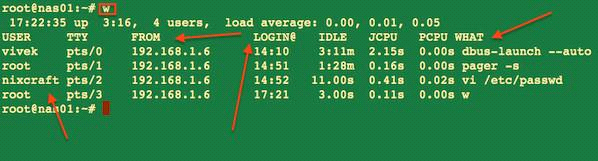
* **who:** This command shows currently logged in users with time details

ex: ***# who***

root tty1 2012-10-12 17:56 old 1960 (:0)

root pts/0 2012-10-12 17:57 06:51 2376 (:0.0)

* **w:** To see who is currently logged in and what they are doing on your Linux/Unix-based server,

Sample outputs:

* **finger:** The Linux finger command allows you to see who else is on the computer or get detailed information about a user who has access to the system.The following Linux finger command displays the who else are logged in on the computer currently.

***[root@localhost ~]# finger***

Login Name Tty Idle Login Time Office Office Phone

tintin tty3 Dec 27 12:20

The following Linux finger command displays detailed information about the user tintin.

***[root@localhost ~]# finger*** tintin

Login: tintin Name: (null)

Directory: /home/tintin Shell: /bin/bash

Never logged in.

No mail.

No Plan.

**Working with Files and Directories:**

* **touch command**

**touch:**

The “touch” command stands for (Update the access and modification times of each FILE to the current time). touch command creates the file with zero byte size, only if it doesn’t exist. If the file already exists it will update the timestamp of a file.

Syntax : **$touch** <filename>

Ex: - $touch file1

* **cat command**

**cat :** The cat command displays the contents of a file

Note: “>>” and “>” are called append symbol. They are used to append the output to a file and not on standard output. “>” symbol will delete a file already existed and create a new file hence for security reason it is advised to use “>>” that will write the output without overwriting or deleting the file.

***#ca*t** > 1.txt  
 Hi my name is x .  
 I am friend of y   
……….

***Ctrl + D***

* **ls command**

**$ls**: list all the files and directories in current working directory

|  |  |
| --- | --- |
| Option | description |
| [ls -a](http://www.rapidtables.com/code/linux/ls/ls-a.htm) | list all files including hidden file starting with '.' |
| ls --color | colored list [=always/never/auto] |
| ls –d | list directories - with ' \*/' |
| ls –F | add one char of \*/=>@| to enteries |
| ls –i | list file's inode index number |
| [ls -l](http://www.rapidtables.com/code/linux/ls/ls-l.htm) | list with long format - show permissions |
| [ls -la](http://www.rapidtables.com/code/linux/ls/ls-l.htm) | list long format including hidden files |
| [ls -lh](http://www.rapidtables.com/code/linux/ls/ls-l.htm) | list long format with readable file size |
| [ls -ls](http://www.rapidtables.com/code/linux/ls/ls-l.htm) | list with long format with file size |
| [ls -r](http://www.rapidtables.com/code/linux/ls/ls-r.htm#reverse) | list in reverse order |
| [ls -R](http://www.rapidtables.com/code/linux/ls/ls-r.htm#recursive) | list recursively directory tree |
| [ls -s](http://www.rapidtables.com/code/linux/ls/ls-s.htm#size) | list file size |
| [ls -S](http://www.rapidtables.com/code/linux/ls/ls-s.htm#sort-size) | sort by file size |
| [ls -t](http://www.rapidtables.com/code/linux/ls/ls-t.htm) | sort by time & date |
| ls –X | sort by extension name |

***Table:*** *Options list of ls command*

* **rm command**

**rm:** Linux command is used to remove/delete the file from the directory.the rm command can be used to remove one file, like this:

***$rm*** oldfile.txt  
 You can also use the rm command to delete multiple Linux files at one time.

Ex: ***$rm*** file1 file2 file3

If you prefer to be careful when deleting files, use the -i option with the rm command. The -i stands for "inquire", so when you use this option the rm command prompts you with a yes/no prompt before actually deleting your files:

**$rm** -i files file2 file3

* **mkdir command**

**mkdir:** To create the directory

Syntax: ***$mkdir*** <Directory>

Ex: ***$mkdir*** DIR1

Ex: ***$mkdir*** -p DIR1/DIR2/DIR3

* **rmdir command:**

**rmdir:** To remove the empty directory

Syntax: ***$rmdir*** <Directory>

Ex: ***$rm -rf*** : To remove the directory which is having contents.

* **cd command**

**cd:** The Linux cd command can be used to change the working directory.

Examples:

***[root@localhost ~]#cd*** /home (Changes the working directory to /home)

***[root@localhost ~]#cd ..*** (Changes the working directory to the parent directory)

***[root@localhost ~]#cd***~(Changes the working directory to the users home directory)

* **mv command**

**mv:** The Linux mv command is used to move a file from one location to another location and renaming the file as well. Rename a File and Directory

Ex:***[root@server02 ~]# mv*** file1 file2

***[root@server02 ~]# mv*** dir1 dir2

Moving a file into some other location

Ex: ***[root@server02 ~]# mv*** file1 /tmp

* **cp command**

**cp:** copy files and directories

Syntax : ***$cp*** <source file> <dest file>

Ex: ***root@linux:~/test$ cp*** fire water

**#cp –I**

To prevent cp from overwriting existing files, use the -i (for interactive option)

***#root@linux:~/test$ cp -i*** fire water

cp: overwrite `water'? no

**#cp –r**

To copy complete directories, use cp -r (the -r option forces recursive copying of all files in all subdirectories).

***#root@linux:~/test$ cp -r*** MyDir MyDirB

**To view the contents of a large files:** more, less and pg

* **more:** To view a text file one page at a time, press spacebar to go to the next page

***$more*** *filename :* show the document one page at a time

***$more*** -num filename : show the document page few lines as specified bu (-num)

example : ***more*** -10 filename will show 10 lines for every page

* **less :** It is much the same as more command except:

You can navigate the page up/down using the less command and not possible in more command. You can search a string in less command. (use /keyword to search). “more” was fairly limited, and additional development on “more” had stopped. d) it uses same functions as vi editor the usage : ***less*** filename

* **alias:** Invoking alias with noarguments will display all currently aliased commands.

Ex: ***$alias*** ls='ls --color=auto'

Creates the alias "ls" such that using thels command will always display color output.

Ex: ***$alias*** ll='ls -la'

* **tty:** The Linux "tty" command display the name of the connected terminal.

Example: ***[root@RHEL01 ~]# tty*** /dev/pts/0

Note: The above terminal displayed is a pseudo-terminal.

* **which:** The Linux "which" command prints where in the search path an executable binary is located.

Example: ***[root@RHEL01 ~]# which*** mail /bin/mail

* **echo:** The Linux "echo" command is used to print to the terminal.

Example 1: (Prints the given string to the terminal):

***[root@RHEL01 ~]# echo*** "hell world"

hell world

Example 2 (Prints the variable $USER to the terminal):

***[root@RHEL01 ~]# echo*** $USER

root

Example 3 (Prints the variable $PWD to the terminal):

***[root@RHEL01 ~]# echo*** $PWD

/root

* **clear:** The Linux "clear" command clears the terminal.
* **logout/exit:** To logout from the current user.

Ex: ***$logout*** or ***$exit***

halt, power off, and reboot are commands you can run asroot to stop the system hardware.

halt instructs the hardware to stop all CPU functions.

power off sends an [ACPI](http://www.computerhope.com/jargon/a/acpi.htm) signal which instructs the system to power down.

reboot instructs the system to [reboot](http://www.computerhope.com/jargon/r/reboot.htm).

* **shutdown:** Shutdown the system in a safe way. You can shutdown the machine immediately, or schedule a shutdown using 24 hour format.

***#shutdown*** 8:00

Schedule the system to shut down at 8 A.M.

***#shutdown*** 20:00

Schedule the system to shut down at 8 P.M.

***#shutdown*** +15 "Upgrading hardware, downtime should be minimal"

Schedule the system to shut down in fifteen minutes. Along with the normal message alerting users that the system is shutting down, they will be given the descriptive message about a hardware upgrade.

***#shutdown*** now

Bring down the system immediately.

***#shutdown*** -r now

Bring down the system immediately, and automatically reboot it.

***#shutdown*** -P now

Bring down the system immediately, and automatically power off the system.

**IV. Text Processing Tools (Utilities)**

* **wc:** wc (word count) command in Unix/Linux operating systems is used to find out number of newline count, word count, byte and characters count in a files specified by the file arguments. The syntax of wc command as shown below.

Syntax***:# wc*** [options] filenames

The following are the options and usage provided by the command.

wc -l : Prints the number of lines in a file.  
wc -w : prints the number of words in a file.  
wc -c : Displays the count of bytes in a file.

* **wc -m :** prints the count of characters from a file.
* **Piping:**Now we'll take a look at a mechanism for sending data from one program to another. It's called piping and the operator we use is ( | ) (found above the backslash ( \ ) key on most keyboards). What this operator does is feed the output from the program on the left as input to the program on the right. In the example below we will list only the first 3 files in the directory. Ex**:*$ls***

barry.txt bob example.png firstfile foo1 myoutput video.mpeg

***$ls | head*** *-3*

barry.txt

bob

example.png

We may pipe as many programs together as we like. In the below example we have then piped the output to tail so as to get only the third file.

Ex***:$ls | head -3 | tail -1***

example.png

***$ls /etc | wc -l***

48

**I/O Redirection:**

we will explore a powerful feature used by many command line programs called input/output redirection. As we have seen, many commands such as ls print their output on the display. This does not have to be the case, however. By using some special notation we can redirect the output of many commands to files, devices, and even to the input of other commands.

Every program we run on the command line automatically has three data streams connected to it.

* STDIN (0) - Standard input (data fed into the program)
* STDOUT (1) - Standard output (data printed by the program, defaults to the terminal)
* STDERR (2) - Standard error (for error messages, also defaults to the terminal)



Piping and redirection is the means by which we may connect these streams between programs and files to direct data in interesting and useful ways. We'll demonstrate piping and redirection below with several examples but these mechanisms will work with every program on the command line, not just the ones we have used in the examples.

**1 : Output Redirection:**

Output Redirection is most commonly used, when you execute a command it’s normally appears on your terminal. Using this operator ‘>’ in shell you can redirect your output in a file.

Ex:

***[root@loadb1/]#ls >*** output.txt

***[root@loadb1/]#cat*** output.txt

bin

boot

Note: If output.txt file is already exit then ‘>’ operator will overwrite the file for append more output into output.txt use ‘>>’ instead of ‘>’.

**2: Input Redirection:**

You can redirect your input by using ‘<‘ operator. You cannot run input redirection on all programs or commands. you can use only with that programs or commands that accept Input from keyboard.

Example : you are going to send an email and u already have Template of that email. you can put your template in your email body using input redirect.

***[root@loadb1 /]# mail*** ali < mail\_template.txt

above command launch email program with mail\_template.txt contents.

Now due to advancement in GUI, and also lots of good email clients, method is rarely used.

**3 : Error Redirection:**

Error Redirection is very helpful when u are in trouble. In my case i am trying to open a file that is not readable for my user i will get permission denied errors. i will redirect these error into error.txt file.

Example :

***[H.Ali@lx1/]$cat*** ali.txt 2> /home/H.Ali/error.txt  
***[H.Ali@lx1 /]$ cat*** /home/H.Ali/error.txt  
cat: ali.txt: Permission denied

In the above command 2 is descriptor of error redirection file by typing ‘2>’ you are saying redirect any kind of error to the file error.txt

Redirecting the standard error (stderr) and stdout to file

Use the following syntax:

***$ command-name &>***file

OR

***$ command >*** file-name 2>&1

Another useful example:

***# find*** /usr/home -name .profile 2>&1 | more

Redirecting both STDERROR & STDOUT to a file

Syntax: ***$command &>*** file1.txt

Ex:- ***ls /etc &> file1.txt***

* **tee :** Tee redirects standard input to both standard output and one or more files.

***$command | tee file***

This pattern (which includes the tee command) redirects the standard output of the command to a file and overwrites its contents. Then, it displays the redirected output in the terminal. It creates a new file if the file does not already exist. In the context of this pattern, tee is typically used to view a program's output while simultaneously saving it to a file.

$wc /etc/magic | tee magic\_count.txt

Examples

***#ls -1 \*.txt | wc -l | tee count.txt***

In the above example, thels command lists all files in the current directory that have the filename extension .txt, one file per line; this output ispiped towc, which counts the lines and outputs the number; this output is piped to tee, which writes the output to theterminal, and writes the same information to the file count.txt. If count.txt already exists, it is overwritten.

* **grep command:**

The basic usage of grep command is to search for a specific string in the specified file as shown below.

Syntax: grep "literal\_string" filename

***$ grep "this" demo file***

this line is the 1st lower case line in this file.

Two lines above this line is empty.

**Another definition for grep:** grep prints lines of input matching a specified pattern.

Syntax: ***grep*** [*OPTIONS*] *PATTERN* [*FILE*...]

***#grep*** chope /etc/passwd

Search /etc/passwd for user chope.

***#grep -r*** "computerhope" /www/

Recursively search the directory /www/, and all subdirectories, for any lines of any files which contain the string "computerhope".

***#grep -w*** "hope" myfile.txt

Search the file myfile.txt for lines containing the word "hope". Only lines containing the distinct word "hope" will be matched. Lines in which "hope" is *part* of a word will *not* be matched.

***#grep -cw*** "hope" myfile.txt

Same as previous command, but displays a count of how many lines were matched, rather than the matching lines themselves.

***#grep -cvw*** "hope" myfile.txt

Inverse of previous command: displays a count of the lines in myfile.txt which do *not* contain the word "hope".

***#grep -l*** "hope" /www/\*

Display the filenames (but not the matching lines themselves) of any files in /www/ (but not its subdirectories) whose contents include the string "hope".

* **cut command:**

Cut command in Unix (or Linux) is used to select sections of text from each line of files. You can use the cut command to select fields or columns from a line by specifying a delimiter or you can select a portion of text by specifying the range or characters. Basically the cut command slices a line and extracts the text

syntax : ***cut*** [option] [source file]

***[root@server02 ~]# cat*** file1

Hi Flashsoftindia

Welcome you guys

***[root@server02 ~]# cut*** -c1-5 file1

Hi Fl

Welco

***[root@server02 ~]# cut -d***":" -f 1 /etc/passwd

root

bin

daemon

adm

***[root@server02 ~]# cut -d***":" -f1,5 /etc/passwd

root: root

bin: bin

daemon: daemon

adm: adm

* **sort command:**

Sort command in unix or linux system is used to order the elements or text. Sort command has the capability of sorting numerical values and strings. The sort command can order the lines in a text file.

syntax : ***sort*** [options] filename  
The options are:

***-b*** : Ignores leading spaces in each line  
***-d*** : Uses dictionary sort order. Conisders only spaces and alphanumeric characters in sorting  
***-f*** : Uses case insensitive sorting.  
***-M*** : Sorts based on months. Considers only first 3 letters as month. Eg: JAN, FEB  
***-n*** : Uses numeric sorting  
***-R*** : Sorts the input file randomly.  
***-r*** : Reverse order sorting  
***-k*** : Sorts file based on the data in the specified field positions.  
***-u*** : Suppresses duplicate lines  
***-t*** : input field separator

File with Ascii data:

Let us consider a file with the following contents:

***$ cat*** file  
Unix  
Linux  
Solaris  
AIX  
Linux  
HPUX

sort simply sorts the file in alphabetical order:

***$ sort*** file  
AIX  
HPUX  
Linux  
Linux  
Solaris  
Unix

sort file on the basis of 1st field:

***$ sort -t***"," -k1,1 file  
AIX,25  
HPUX,100  
Linux,20  
Linux,25  
Solaris,10

* **uniq:**

Uniq command in unix or linux system is used to suppress the duplicate lines from a file. It discards all the successive identical lines except one from the input and writes the output. The default behavior of the uniq command is to suppress the duplicate line. Note that, you have to pass sorted input to the uniq, as it compares only successive lines.

***> uniq example.txt***Unix operating system  
Unix dedicated server  
Linux dedicated server  
  
If the lines in the file are not in sorted order, then use the sort command and then pipe the output to the uniq command.

***> sort example.txt | uniq***

Count of lines.

The -c option is used to find how many times each line occurs in the file. It prefixes each line with the count.

***$uniq -c*** example.txt  
 2 Unix operating system  
 1 Unix dedicated server  
 1 Linux dedicated server  
  
3. Display only duplicate lines.

You can print only the lines that occur more than once in a file using the -d option.

***$ uniq -d*** example.txt  
Unix operating system  
  
***$uniq -D*** example.txt  
Unix operating system  
Unix operating system

* **Translate command :**

tr stands for translate or transliterate. The tr utility in Unix or Linux system is used to translate, delete or squeeze characters. The syntax of tr command is

***$tr*** [options] set1 [set2]  
  
The options of tr command are:

-c : complements the set of characters in string.

-d : deletes the characters in set1

-s : replaces repeated characters listed in the set1 with single occurrence

-t : truncates set1

Tr command Examples:

1. Convert lower case letters to upper case

The following tr command translates the lower case letters to capital letters in the give string:

***$echo*** "Linux dedicated server" | ***tr*** "[:lower:]" "[:upper:]"  
LINUX DEDICATED SERVER  
***$echo*** "Linux dedicated server" | ***tr*** "[a-z]" "[A-Z]"

* **Comparing Files:** diff is used to compare the contents of two files of differences.

Syntax : diff <file1> <file2>

**[root@server02 ~]# *cat*** file1

Hello Welcome to india

**[root@server02 ~]#** ***cat*** file2

Hello Welcome to flashsoftindia

It is incredible in ATP

**[root@server02 ~]#** ***diff*** file1 file2

1c1,2

< Hello Welcome to india

---

> Hello Welcome to flashsoftindia

> It is incredible in ATP

**[root@server02 ~]#**

Patch : Patch duplicates changes in other files.

-b option to automatically back up changed files.

**[root@server02 ~]#** ***diff -u*** file1 file2 > file.patch

**[root@server02 ~]#** ***patch -b*** file1 file.patch

patching file file1

**[root@server02 ~]#** ***cat*** file1

Hello Welcome to flashsoftindia

It is incredible in ATP

**[root@server02 ~]#** ***cat*** file2

Hello Welcome to flashsoftindia

It is incredible in ATP

* **sed command:**

Sed is a Stream Editor used for modifying the files in unix (or linux). Whenever you want to make changes to the file automatically, sed comes in handy to do this. Most people never learn its power; they just simply use sed to replace text. You can do many things apart from replacing text with sed.

***$cat file.txt***  
Unix is great os. Unix is open source. Unix is free os.  
learn operating system.  
Unix Linux which one you choose.

Replacing or substituting string

***$sed*** 's/Unix/Linux/' file.txt  
Linux is great os. Unix is open source. Unix is free os.  
learn operating system.  
Linux Linux which one you choose.

Replacing the nth occurrence of a pattern in a line.

***$sed*** 's/Unix/Linux/2' file.txt  
Unix is great os. Linux is open source. Unix is free os.  
learn operating system.  
Unix Linux which one you choose.

Replacing all the occurrence of the pattern in a line

***$sed*** 's/Unix/Linux/g' file.txt  
Linux is great os. Linux is open source. Linux is free os.  
learn operating system.  
Linux Linux which one you choose.

Duplicating the replaced line with /p flag

***$sed*** 's/Unix/Linux/p' file.txt  
Linux is great os. Unix is open source. Unix is free os.  
Linux is great os. Unix is open source. Unix is free os.  
learn operating system.  
Linux Linux which one you choose.  
Linux Linux which one you choose.

Deleting lines

***$sed*** '2 d' file.txt  
***$sed*** '5,$ d' file.txt

* **head command:**

head, by default,prints the first 10 lines of each FILE to standard output. With more than one FILE, it precedes each set of output with a header identifying the file name. If no FILE is specified, or when FILE is specified as a dash ("-"), head reads fromstandard input.

***[root@server02 ~]# head*** */etc/passwd*

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

halt:x:7:0:halt:/sbin:/sbin/halt

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin

***[root@server02 ~]# head -n*** *15 /etc/passwd*

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

halt:x:7:0:halt:/sbin:/sbin/halt

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

<gopher:x:13:30:gopher:/var/gopher:/sbin/nologin>

<ftp:x:14:50:FTP> User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/:/sbin/nologin

* **tail command:**

tail prints the last 10 lines of each FILE to standard output. With more than one FILE, it precedes each set of output with a header giving the file name. If no FILE is specified, or if FILE is specified as a dash ("-"), tail reads fromstandard input.

***[root@server02 ~]# tail*** */etc/passwd*

ntp:x:38:38::/etc/ntp:/sbin/nologin

saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin

nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

tcpdump:x:72:72::/:/sbin/nologin

oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbin/nologin

nag:x:500:500::/home/nag:/bin/bash

nagesh:x:501:501::/home/nagesh:/bin/bash

***[root@server02 ~]# tail -n*** *15 /etc/passwd*

dbus:x:81:81:System message bus:/:/sbin/nologin

vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin

rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin

abrt:x:173:173::/etc/abrt:/sbin/nologin

haldaemon:x:68:68:HAL daemon:/:/sbin/nologin

ntp:x:38:38::/etc/ntp:/sbin/nologin

saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin

nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

tcpdump:x:72:72::/:/sbin/nologin

oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbin/nologin

nag:x:500:500::/home/nag:/bin/bash

nagesh:x:501:501::/home/nagesh:/bin/bash

***$tail -f*** *myfile.txt :* Outputs the last 10 lines of myfile.txt, and monitors myfile.txt for updates; tail then continues to output any new lines that are added to myfile.txt.

[root@server02 ~]# tail -f /var/log/messages

Jan 19 17:35:02 server02 rsyslogd: [origin software="rsyslogd" swVersion="5.8.10" x-pid="1007" x-info="http://www.rsyslog.com"] rsyslogd was HUPed

* **Zipping the Files:**

Compressed files occupy less disk space and download faster than large, uncompressed files. You can compress Linux files with the open-source compression tool gzip and decompress the compressed files using gzip or gunzip tool. The Linux gzip tool reduces the size of the files using Lempel-Ziv coding (LZ77).

gzip/gunzip/zcat

You can compress a file using gzip by the following commands. The first command will create a compressed gzip file with gz extension, removing the uncompressed file and second command will create a compressed gzip file with gz extension keeping the uncompressed file. The Linux zcat command allows the user to expand and view a compressed file without uncompressing that file.

***[root@RHEL2 gzip]# gzip*** gzip.txt

Or

***[root@RHEL2 gzip]# gzip*** -c gzip.txt > gzip.txt.gz

And decompress the compressed file using the following command

***[root@RHEL2 gzip]# gzip*** -d gzip.txt.gz

A compressed file can also decompressed by using gunzip command as shown below.

***[root@RHEL2 gzip]# gunzip*** gzip.txt.gz

To list the compressed file information, use “gzip –l” option.

[***root@RHEL2 gzip]# gzip*** -l gzip.txt.gz

To view a compressed file, without uncompressing it, you can use the zcat file as shown below.

***[root@localhost ~]# zcat*** *gzip.txt.gz*

This is a test...

Hello Dear Guys,

Welcome to flash soft india

Compressed files occupy less disk space and download faster than large, uncompressed files. The Linux bzip2 command has better compression ratio compared with Linux gzip command. To create a compressed file using Linux bzip2 command, use the following command. The first command will create a compressed bzip2 file with bz2 extension, removing the uncompressed file and second command will create a compressed bzip2 file with bz2 extension keeping the uncompressed file.

***[root@RHEL2 bzip2]# bzip2*** *bzip2.txt*

OR

***[root@RHEL2 bzip2]# bzip2*** *-c bzip2.txt > bzip2.txt.bz2*

To decompress the bzip2 compressed file, use the “bzip2 –d” option.

***[root@RHEL2 bzip2]# bzip2*** *-d bzip2.txt.bz2*

If you want only to read the compressed file without uncompressing it, use the bzcat command as shown below.

***[root@localhost ~]# bzcat*** *bzip2.txt.bz2*

This is a test...

Hello Dear Guys,

Welcome to Flashsoftindia

…………………………………………………

* **tar (tape archiving) command:**

The tar program is used to create, maintain, modify, and extractfiles that arearchived in the tar format.

***#tar cf*** *archive.tar file1 file2*

Create archive archive.tar containing files file1 and file2. Here, the c tells tar you will be creating an archive; the f tells tar that the next option (here it's archive.tar) will be the name of the archive it creates. file1 and file2, the final arguments, are the files to be archived.

***#tar xf*** *archive.tar*

Extract the files from archive archive.tar. x tells tar to extract files from an archive; f tells tar that the next argument will be the name of the archive to operate on.

***#tar xzvf*** *archive.tar.gz*

Extract the files from gzipped archive archive.tar.gz verbosely. Here, the z tells tar that the archive will be compressed with gzip.

***#tar czf*** *archive.tar.gz mydir/*

Creates an gzip-compressed archive of the directory mydir.

***#tar xvf*** *archive.tar documents/work/budget.doc*

Extract only the file documents/work/budget.doc from the archive archive.tar. Produce verbose output.

***#tar xvf*** *archive.tar --wildcards '\*.doc'*

Extract only files with theextension .doc from the archive archive.tar. The --wildcards option tells tar to interpretwildcards in the name of the files to be extracted; the filename (\*.doc) is enclosed in single-quotes to protect the wildcard (\*) from being expanded incorrectly by the shell.

***#tar uvf*** *archive.tar documents/work/budget.doc*

Add the file documents/work/budget.doc to the archive archive.tar only if it is newer than the version already in the archive (or does not yet exist in the archive). Here, u is the same as the long option --update.

**Finding Files in Linux:**

* **find :**

The find command is used to locate files on a Unix or Linux system. find will search any set of directories you specify for files that match the supplied search criteria. You can search for files by name, owner, group, type, permissions, date, and other criteria. The search is recursive in that it will search all subdirectories too.

The syntax looks like this: **$find** where-to-look criteria what-to-do

basic 'find file' commands  
--------------------------  
find / -name foo.txt -type f -print # full command  
find / -name foo.txt -type f # -print isn't necessary  
find / -name foo.txt # don't have to specify "type==file"  
find . -name foo.txt # search under the current dir  
find . -name "foo.\*" # wildcard  
find . -name "\*.txt" # wildcard  
find /users/al -name Cookbook -type d # search '/users/al'  
  
search multiple dirs  
--------------------  
find /opt /usr /var -name foo.scala -type f # search multiple dirs  
  
case-insensitive searching  
--------------------------  
find . -iname foo # find foo, Foo, FOo, FOO, etc.  
find . -iname foo -type d # same thing, but only dirs  
find . -iname foo -type f # same thing, but only files  
  
find files with different extensions  
------------------------------------  
find . -type f \( -name "\*.c" -o -name "\*.sh" \) # \*.c and \*.sh files  
find . -type f \( -name "\*cache" -o -name "\*xml" -o -name "\*html" \) # three patterns  
  
find files that don't match a pattern (-not)  
--------------------------------------------  
find . -type f -not -name "\*.html" # find all files not ending in ".html"  
  
find files by text in the file (find + grep)  
--------------------------------------------  
find . -type f -name "\*.java" -exec grep -l StringBuffer {} \; # find StringBuffer in all \*.java files  
find . -type f -name "\*.java" -exec grep -il string {} \; # ignore case with -i option  
find . -type f -name "\*.gz" -exec zgrep 'GET /foo' {} \; # search for a string in gzip'd files  
  
5 lines before, 10 lines after grep matches  
-------------------------------------------  
find . -type f -name "\*.scala" -exec grep -B5 -A10 'null' {} \;  
   
find files and act on them (find + exec)  
----------------------------------------  
find /usr/local -name "\*.html" -type f -exec chmod 644 {} \; # change html files to mode 644  
find htdocs cgi-bin -name "\*.cgi" -type f -exec chmod 755 {} \; # change cgi files to mode 755  
find . -name "\*.pl" -exec ls -ld {} \; # run ls command on files found  
  
find and copy  
-------------  
find . -type f -name "\*.mp3" -exec cp {} /tmp/MusicFiles \; # cp \*.mp3 files to /tmp/MusicFiles  
  
copy one file to many dirs  
--------------------------  
find dir1 dir2 dir3 dir4 -type d -exec cp header.shtml {} \; # copy the file header.shtml to those dirs  
  
find and delete  
---------------  
find . -type f -name "Foo\*" -exec rm {} \; # remove all "Foo\*" files under current dir  
find . -type d -name CVS -exec rm -r {} \; # remove all subdirectories named "CVS" under current dir  
  
find files by modification time  
-------------------------------  
find . -mtime 1 # 24 hours  
find . -mtime -7 # last 7 days  
find . -mtime -7 -type f # just files  
find . -mtime -7 -type d # just dirs  
  
find files by modification time using a temp file  
-------------------------------------------------  
touch 09301330 poop # 1) create a temp file with a specific timestamp  
find . -mnewer poop # 2) returns a list of new files  
rm poop # 3) rm the temp file  
  
find with time: this works on mac os x  
--------------------------------------  
find / -newerct '1 minute ago' -print  
  
find and tar  
------------  
find . -type f -name "\*.java" | xargs tar cvf myfile.tar  
find . -type f -name "\*.java" | xargs tar rvf myfile.tar  
   
find, tar, and xargs  
--------------------  
find . -name -type f '\*.mp3' -mtime -180 -print0 | xargs -0 tar rvf music.tar  
   
  
find and pax (instead of xargs and tar)  
---------------------------------------  
find . -type f -name "\*html" | xargs tar cvf jw-htmlfiles.tar -  
find . -type f -name "\*html" | pax -w -f jw-htmlfiles.tar

**V. Basic Linux Networking Tools**

**Hostname and FQDN Settings :**

To set your system’s hostname and FQDN (fully qualified domain name). Your hostname should be something unique; Please note that the system’s hostname has no relationship to websites or email services hosted on it, aside from providing a name for the system itself. Thus, your hostname should not be “www” or anything else too generic.

***$hostname*** : It is the command to get the hostname

***$hostname*** <sys01.flash.com> : To set up the hostname temporarily

Red Hat based system use the file /etc/sysconfig/network to read the saved hostname at system boot. This is set using the init script /etc/rc.d/rc.sysinit

***$cat /etc/sysconfig/network***  
NETWORKING=yes  
HOSTNAME="sys01.flash.com"

Setting up the network parameters in Linux

To configure static ip address, default gateway, netmask, dns server and make a network auto start at boot on Linux host normally configured with static ip address, so that easier to maintain the configuration on its client and DNS server will always able to resolve it. See below the default RHEL/Centos, network card configuration file.

***# cat /etc/sysconfig/network-scripts/ifcfg-eth0***

DEVICE="eth0"  
BOOTPROTO="dhcp"  
HWADDR="xx:xx:xx:xx:xx:xx"  
NM\_CONTROLLED="yes"  
ONBOOT="no"  
TYPE="Ethernet"

DEVICE: Specifies the device to use for the configuration.

IPADDR: The static IP for the network interface

NETMASK: The network mask for the device

HWADDR: The MAC address for the device

NM\_CONTROLLED: Determines if the device can be controlled by the network manager.

ONBOOT: Determines if the device should be initialized when booted

DNS1: Specifies the DNS server for the device. This is needed for name resolution. You may also add DNS2 and DNS3 if you have additional servers.

Here are steps to configure static ip address and auto start of network card once system rebooted.

***# vi /etc/sysconfig/network-scripts/ifcfg-eth0***

DEVICE="eth0"  
BOOTPROTO="static"  
HWADDR="xx:xx:xx:xx:xx:xx"  
NM\_CONTROLLED="yes"  
ONBOOT="yes"  
TYPE="Ethernet"  
IPADDR=192.168.1.2  
NETMASK=255.255.255.0

Configure the gateway

***# vi /etc/sysconfig/network***

NETWORKING=yes  
HOSTNAME=host.lintut.com  
GATEWAY=192.168.1.1

Configure the DNS resolver

***# vi /etc/resolv.conf***

nameserver 8.8.8.8  
nameserver 8.8.4.4

Restart Network service:

***# service network restart***

To Set up the IP address temporarily

We can assign the ip, netmask, gateway in one command

***$ifconfig eth0 192.168.0.1 netmask 255.255.255.0 gw 192.168.0.254***

**Local Resolver**: If we don’t have a DNS server in our infrastructure, we can define all hostnames in the Local Resolver files (/etc/hosts).

The main purpose of /etc/hosts configuration file is to resolve hostnames that cannot be resolved any other way. It can also be used to resolve hostnames on small networks with no DNS server. Regardless of the type of network the computer is on, this file should contain a line specifying the IP address of the loopback device (127.0.0.1) as localhost.localdomain.

***$ cat /etc/hosts***  
127.0.0.1 dev-server localhost.localdomain localhost

***$ cat /etc/hosts***  
127.0.0.1 prod-server localhost.localdomain localhost

**Diagnostics of Network**:

ping is a simple way to sendnetwork data to, and receive network data from, another computer on a network. It is frequently used to test, at the most basic level, whether anothersystem is reachable over a network, and if so, how much time it takes for that data to be exchanged.

**Network Clients:**

**ELINKS:**

ELinks is a text mode WWW browser, supporting colors, table rendering, background downloading, menu driven configuration interface, tabbed browsing and slim code. ELinks can handle both local files and remote URLs. The main supported remote URL protocols are *HTTP*, *HTTPS* (with SSL support compiled in) and *FTP*. Additional protocol support exists for BitTorrent finger, Gopher, *SMB* and *NNTP*.

***$links http://www.google.com***

**wget:**

wget is a free utility fornon-interactive download of files from theweb. It supportsHTTP,HTTPS, andFTPprotocols, as well as retrieval through HTTP [proxies](http://www.computerhope.com/jargon/p/proxyser.htm).

Ex: ***#wget*** [***http://website.com/files/file.zip***](http://website.com/files/file.zip)

**scp:**

The scp tool is used to copy files from one computer to another while encrypting the data.

**Example 1** – Below command will copy myfile.txt securely to a remote machine:

***[root@RHEL04 ~]# scp myfile.txt tintin@192.168.1.106****:*

The authenticity of host '192.168.1.106 (192.168.1.106)' can't be established.

RSA key fingerprint is 88:8e:46:35:ac:dc:5e:73:f2:ce:9a:29:41:f5:73:fb.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '192.168.1.106' (RSA) to the list of known hosts.

tintin@192.168.1.106's password:

myfile.txt 100% 48KB 47.9KB/s 00:00

**Example 2** – Below command will copy mynewfile.txt securely from a remote machine to your machine:

***[root@RHEL04 ~]# scp tintin@192.168.1.106:mynewfile.txt mylocalfile.txt***

tintin@192.168.1.106's password:

mynewfile.txt 100% 575KB 575.1KB/s 00:00

Note: To see more details about the status of command, use the verbose mode (-v).

**SSH :** Secure Shell

SSH client utility in unix or linux server is used to logging into a remote host and execute commands on the remote machine. The rlogin and rsh commands can also be used to login into the remote machine. However these are not secure. The ssh command provides a secure connection between two hosts over a insecure network.

You can login to a remote server from the local host as shown below:

***#localhost:[~]> ssh username@remote-server******#username@remote-server password****:****#remote-server:[~]>***

Running remote commands from local host

Sometimes it is necessary to run the unix commands on the remote server from the local host. An example is shown below:

***#localhost:[~]> ssh user@remote-host "ls test"***online-backup.datoracle-storage.bat

* **telnet (teletype network):**

Telnet (teletype network) is a network protocol used on the Internet or local area networks to provide a bidirectional interactive communications facility. Telnet provides access to a command-line interface on a remote host via a virtual terminal connection. Main disadvantages of Telnet is that the data is sent over the network as clear text. It is possible for someone to use a network analyzer to peek into your data packets and see your username and password. A more secure method for remote logins would be via Secure Shell (SSH) which encrypts the data which is sent over the network.

Telnet can be used to scan a remote host for open TCP ports also.

***[root@RHEL01 ~]# telnet*** *192.168.1.109*

Trying 192.168.1.109...

Connected to RHEL09.omnisecu.com (192.168.1.109).

Escape character is '^]'.

Red Hat Enterprise Linux Server release 5 (Tikanga)

Kernel 2.6.18-8.el5 on an i686

**login**: tintin

**Password**:

* **RSYNC:**

Rsync (Remote Sync) is a most commonly used command for copying and synchronizing files and directories remotely as well as locally in Linux/Unix systems. With the help of rsync command you can copy and synchronize your data remotely and locally across directories, across disks and networks, perform data backups and mirroring between two Linux machines.

***#rsync -avR*** */foo/bar/baz.c remote:/tmp/*

then a file named /tmp/foo/bar/baz.c would be created on the remote machine, preserving its full path. These extra path elements are called "implied directories" (i.e. the "foo" and the "foo/bar" directories in the above example).