

www.jrktrainings.in

**JRK TRAININGS**

**Ph: 9059309846**

**SAP BASIS ADMINISTRATION**

**Linux basic commands**

**By JRK**

**Computer (**Common Operating Machine Purposely Used for Technological and Educational Research)

A **compute**r is a machine for manipulating data according to a list of instructions or an electronic device for the storage and processing of information or a programmable machine which runs with two principal characteristics as

* + It responds to a specific set of instructions in a well-defined manner.
  + It can execute a prerecorded list of instructions (a program).

**What is Software:**

Written coded commands or set of instructions that tell a computer what tasks to perform or Computer instructions or data. Anything that can be stored electronically is called as software. The storage devices and display devices are hardware.

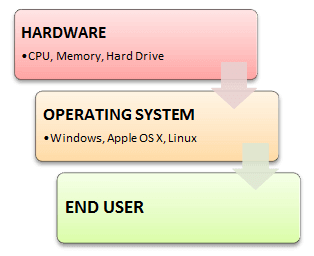
## Software is often divided into two categories:

* **Systems software:** Includes the operating system and all the utilities that enable the computer to function.
* **Applications software:** Includes programs that do real work for users. For example, word processors, spreadsheets, and database management systems fall under the category of applications software.

**What is an Operating System?**

**I**n **Operating System (OS)** is software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs. Applications like Browsers, MS Office, Notepad, Games, SAP Applications, Different databases etc., need some environment to run and perform its tasks.

The OS helps you to communicate with the computer without knowing how to speak the computer’s language. It is not possible for the user to use any computer or mobile device without having an operating system.

Introduction to Operating System

**Types of Operating systems:**

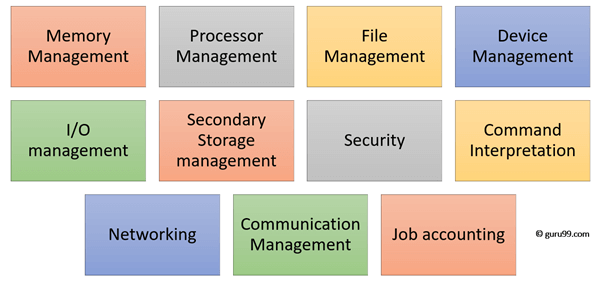
Following are the popular types of OS (Operating System):

* Batch Operating System
* Multitasking/Time Sharing OS
* Multiprocessing OS
* Real Time OS
* Distributed OS
* Network OS
* Mobile OS

**Functions of Operating System:**

Some typical operating system functions may include managing memory, files, processes, I/O system & devices, security, etc.

Below are the main functions of Operating System:

Functions of Operating System

In an operating [system software](https://www.guru99.com/difference-system-software-application-software.html) performs each of the function:

**Process management:** Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

**Memory management:** Memory management module performs the task of allocation and de-allocation of memory space to programs in need of this resources.

**File management:** It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.

**Device Management:** Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices.

**I/O System Management:** One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.

**Secondary-Storage Management:** Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.

**Security:** Security module protects the [data and information](https://www.guru99.com/difference-information-data.html%20) of a computer system against malware threat and authorized access.

**Command interpretation:** This module is interpreting commands given by the and acting system resources to process that commands.

**Networking:** A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.

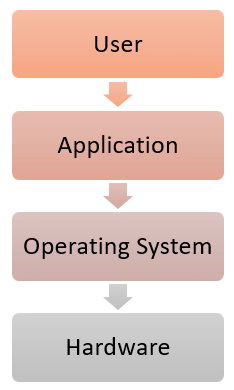
**Job accounting:** Keeping track of time & resource used by various job and users.

**Communication management:** Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

**Features of Operating System (OS):**

Here is a list important features of OS:

* Protected and supervisor mode
* Allows disk access and file systems Device drivers Networking Security
* Program Execution
* Memory management, Virtual Memory Multitasking
* Handling I/O operations
* Manipulation of the file system
* Error Detection and handling
* Resource allocation
* Information and Resource Protection



**Operating Systems available in market:**

* Windows
* Android
* iOS
* Mac OS
* Linux (RedHat, SUSE, Cent OS…)
* Chrome OS
* Windows Phone OS

SAP Application can be installed on any operating system, But most of the SAP Application servers runs on LINUX and UNIX flavors for security reasons.

**Difference between Linux and Windows:**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Linux** | **Windows** |
| Case Sensitivity | The file system in Linux is very case-sensitive. | The file system in Windows is not case-sensitive. |
| Cost Incurred | Linux is free to use for everyone. | Windows do not come free for any user. |
| Open Source | It is open source. | It is not open source. |
| Type of Kernel Used | Linux utilizes the monolithic kernel. | Windows uses the micro-kernel. |
| Path Separator | The path separator that Linux uses is Slash. | The path separator that Windows uses is backward slash. |
| Efficiency | In the case of operations, Linux is way more efficient than Windows. | For operations, Windows are comparatively way less efficient than Linux. |
| Security | It is more secure than Windows OS. | It provides much less security to its users than Linux. |
| Uses in Hacking | People generally use Linux for the systems that are hacking-based. | Windows is not a very efficient OS for hacking purposes as compared to Linux. |

**Hardware and OS Components:**

**Computer Hardware:**

Hardware is the physical medium built with electronic technology lies between input and Output.

As an example objects that you can actually touch, like disks, disk drives, display screens, keyboards, printers, boards, and chips.

In contrast, software is untouchable. Software exists as ideas, concepts, and symbols, but it has no substance.

The distinction between software and hardware is sometimes confusing because they are so integrally linked.

**Computer Hardware devices:**

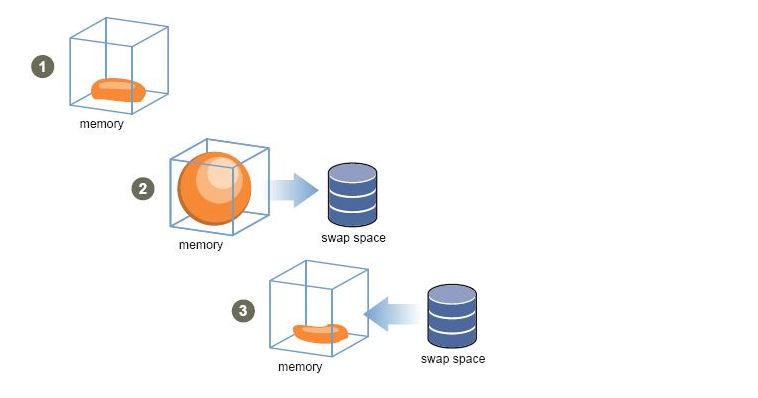
**Random Access Memory (RAM)** – The temporary storage device and part of CPU which is used for program execution and short-term data storage, so the computer does not have to take the time to access the hard drive to find the file(s) it requires. More RAM will normally contribute to a faster PC. RAM is almost always removable as it sits in slots in the motherboard, attached with small clips. The RAM slots are normally located next to the CPU socket.

**Motherboard** – It is the main component inside of a computer designed using electronic technology with all circuits internally within it. It holds the processor, memory, and any add-in boards. It's inside the case and is the component that all of your peripherals plug into. It is also called the "mainboard," or, "mobo," for short.

**Processor** - The exact term processor is a sub-system of a data processing system which processes received information after it has been encoded into data by the input sub- system. These data are then processed by the processing sub-system before being sent to the output sub-system where they are decoded back into information. However, it is usually termed as the microprocessor, the brains of the modern-day computers.

**SWAP (Virtual Memory):**

The second type of memory in modern Linux systems is swap space. The primary function of swap space is to substitute disk space for RAM memory when real RAM fills up and more space is needed.

****

Swap memory or swap space is the on-disk component of the virtual memory system. It is pre-configured as a swap partition or a swap file when Linux is first installed, but can also be added later.

Disk access is very slow compared to memory access. The system runs more slowly if there is excessive swapping, or thrashing, when the system is unable to find enough free memory. In this situation the only solution is to add more RAM.

**Virtual memory:**

Virtual memory is a combination of RAM and disk space that running processes can use. Swap space is the portion of virtual memory that is on the hard disk, used when RAM is full.

**What is the difference between RSS and VSS?**

**RSS** is the Resident Set Size and is used to show how much memory is allocated to that process and is in RAM. It does not include memory that is swapped out. It does include memory from shared libraries as long as the pages from those libraries are actually in memory. It does include all stack and heap memory.

**VSZ** is the Virtual Memory Size. It includes all memory that the process can access, including memory that is swapped out, memory that is allocated, but not used, and memory that is from shared libraries.

**What is CPU?**

The Central Processing Unit (CPU) is the primary component of a computer that acts as its “control center.” The CPU, also referred to as the “central” or “main” processor, is a complex set of electronic circuitries that runs the machine’s operating system and apps. The CPU interprets, processes and executes instructions, most often from the hardware and software programs running on the device.  
  
The CPU performs arithmetic, logic, and other operations to transform data input into more usable information output.

**What is HDD?**

A hard drive is a non-volatile hardware component on a computer that acts as the storage for all digital content. It holds program files, documents, pictures, videos, music, and more.

The non-volatile nature of hard drives means they don’t lose data, even if power is lost. Due to this, they help computers store files and other data for a long time – as long as they don’t get damaged or corrupted.

**Types of Hard Drives**

Currently, there are two common types of hard drive:

* Hard Disk Drive (HDD)
* Solid State Drive (SSD)

To administrate SAP application and databases we must familiar on below Linux basic commands (these commands

Will include with OS perforce and Network config/tuning)

**LINUX/UNIX Commands**

**All those commands work on Linux, macOS, WSL, and anywhere you have a UNIX environment.**

**Shortcuts**

1. **ctrl+c** Halts the current command

2. **ctrl+z** Stops the current command, resume with fg in the foreground or bg in the background

3. **ctrl+d** Logout the current session, similar to exit

4. **ctrl+w** Erases one word in the current line

5. **ctrl+u** Erases the whole line

6. **ctrl+r** Type to bring up a recent command

7. **!!** Repeats the last command

8. **exit** Logout the current session

9. **history :** to get old executed commands

10. **ctrl+l :** clear the screen

11**. !<last starting char> :** will repeats the last executed command

12. **ctrl+a :** curser will go to starting of command

13: **ctrl+e :** curser will go to end of command

**14. Alt+b/f :** move curser one word back and forward

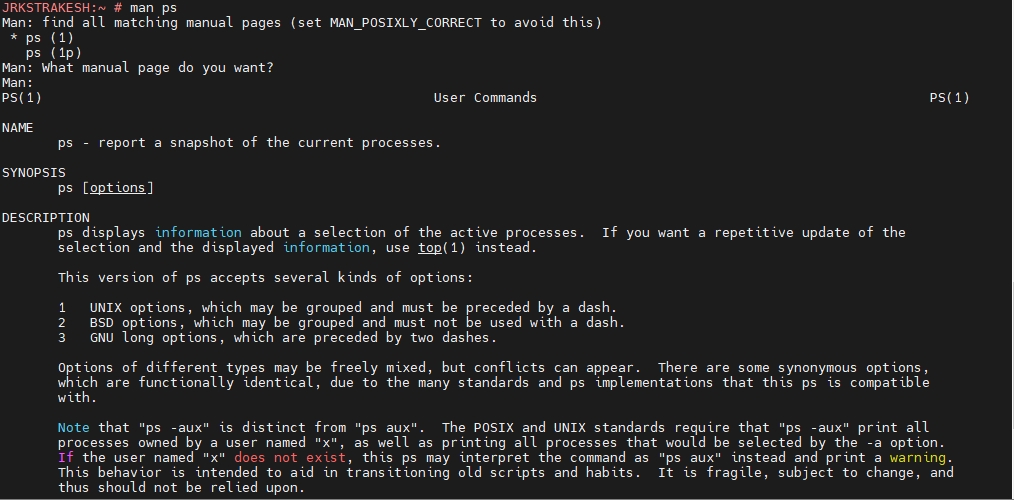
1. **Basic Linux commands**

**1.1 man**

**Syntax :** man <command>

Every time I don't know how to use a command, I type man to get the manual:

Exp: man ps (ps is linux command)

****

* 1. **alias**

alias command will used to create short cut commands

**example :**

ls –l command will display the long listing files for this we can create short cut

command like below

#alias ll=’ls –l’

#alias

**1.3 ls**

Ls command we will used to check the list of files/directories and sub directories including

File or dir/link type, permissions, ownership(owner,group), number of sub directories , last modification,

Name. Below are the 10 examples for ls command

Syntax : ls <option>

1. **ls** command with no option list files and directories in a bare format where we won’t be able to view details like file types, size, modified date and time, permission and links, etc**.**
2. **Open Last Edited File Using ls –t**
3. **Open long listing files and directories ls –l**

**d. Display File Size in Human Readable Format Using ls -lh**

ls -lh (h stands for human readable form) : To display file size in easy to read format. i.e i.e M for MB, K for KB, G for GB.

**e. Order Files Based on Last Modified Time Using ls -lt**

ls -lt : To sort the file names displayed in the order of last modification time.You will be finding it

handy to use it in combination with -l option.

**h. Order Files Based on Last Modified Time (In Reverse Order) Using ls -ltr**

$ ls -ltr : To sort the file names in the last modification time in reverse order. This will be showing the last edited file in the last line which will be handy when the listing goes beyond a page.

1. **Display Hidden Files Using ls -a**

$ ls -a : To show all the hidden files in the directory, use ‘-a option’. Hidden files in Unix starts with ‘.’ in its file name.It will show all the files including the ‘.’ (current directory) and ‘..’ (parent directory).

**$ ls -A : To show the hidden files, but not the ‘.’ (current directory) and ‘..’ (parent directory).**

**1.4. uname :**

The command ‘ uname ‘ displays the information about the system.

**Syntax:**

**uname [OPTION]**

-a, --all print all information, in the following order,

except omit -p and -i if unknown:

-s, --kernel-name print the kernel name

-n, --nodename print the network node hostname

-r, --kernel-release print the kernel release

-v, --kernel-version print the kernel version

-m, --machine print the machine hardware name

-p, --processor print the processor type (non-portable)

-i, --hardware-platform print the hardware platform (non-portable)

-o, --operating-system print the operating system

**1.5. cat :** reads data from the file and gives their content as output

**Syntax:** cat <option> <filename>

**Examples:**

**a) To view a single file**   
**Command:** 

**$cat filename**

Output   
 It will show content of given filename

**b) To view multiple files**   
**Command:** 

**$cat file1 file2**

Output

This will show the content of file1 and file2.

**c) To view contents of a file preceding with line numbers.**   
**Command:** 

**$cat -n filename**

Output 

It will show content with line number

example:-cat -n geeks.txt

**d) Create a file**  
**Command:** $ cat > newfile

Output

Will create a file named new file

**e) Copy the contents of one file to another file.**   
**Command:** 

$cat [filename-whose-contents-is-to-be-copied] > [destination-filename]

Output

The content will be copied in destination file

**f) Cat command can suppress repeated empty lines in output**   
**Command:** 

$cat -s geeks.txt

Output

Will suppress repeated empty lines in output

**g) Cat command can append the contents of one file to the end of another file.**   
**Command:** 

$cat file1 >> file2

Output

Will append the contents of one file to the end of another file

**h) Cat command can display content in reverse order using tac command.**   
**Command:** 

$tac filename

Output

Will display content in reverse order

**i) Cat command can highlight the end of line.**   
**Command:** 

$cat -E "filename"

Output

Will highlight the end of line

**1.5 More**

The **more** command reads files and displays the text one screen at a time. The command pauses after each screen and prints the word More at the bottom of the screen. If you then press a carriage return, the **more** command displays an additional line. If you press the space bar, the **more** command displays another full screen of text.

**Syntax:**

***more [-options] [-num] [+/pattern] [+linenum] [file\_name]***

* ***[-options]****: any option that you want to use in order to change the way the file is displayed. Choose any one from the followings: (-d, -l, -f, -p, -c, -s, -u)*
* ***[-num]****: type the number of lines that you want to display per screen.*
* ***[+/pattern]****: replace the pattern with any string that you want to find in the text file.*
* ***[+linenum]****: use the line number from where you want to start displaying the text content.*
* ***[file\_name]****: name of the file containing the text that you want to display on the screen.*

***While viewing the text file use these controls:***

*Enter key: to scroll down line by line.  
Space bar: To go to the next page.  
b key: To go to back one page.*

*Q key: to quit*

**1.6 less**

The 'less' command is same as 'more' command but include some more features.

It automatically adjust with the width and height of the teminal window, while 'more' command cuts the content as the width of the terminal window get shorter.

**Syntax:**

1. less **<file** name**>**

**Example:**e

less /var/log/messages

Look at the above snapshot, we have given the command **"less /var/log/udev"**. In the left corner, name of the file is displayed.

A particular string is searched by typing forward slash (/) and then followed by the string name, at place of file name in the lower left corner of the terminal box.

**Syntax:**

1. /string name

**Example:**

1. /add

Look at the above snapshot, we want to find string 'add' hence we have typed **"/add"** at the lower left corner.

**Output:** Below snapshot, shows the output of command **"/add"**. The string 'add' will be automatically highlighted.

## less Command Navigation Keys

The 'less' command have navigation keys similar to the 'vim' editor. Let's have a look on some of the navigation keys and other operations of less command.

### 1. Search Navigation

search navigtion keys will help you in forward and backward search.

**Forward search**

* / : search for a pattern for the next occurrence
* n : search for next match
* N : search for previous match

**Backward search**

* ? : search for a pattern for the next occurrence
* n : for next match in backward direction
* N : for previous match in forward direction

### 2. Screen Navigation

* Ctrl + f : forwards one window
* Ctrl + d : forwards half window
* Ctrl + b : backwards one window
* Ctrl + u : backwards half window

### 3. Line Navigation

to move forward or backward line by line

* j : forward by one line
* k : backward by one line

### 4. other Navigations

* G : used to go to end of the file
* g : used to go to start of the file
* q or ZZ : to exit

### 5. Count Keys

* 10j : 10 lines forward
* 10k : 10 lines backward

**1.7 Head and Tail**

Many types of commands are available in bash to show the content of a file. Most commonly used commands are ‘cat’, ‘more’, ‘less’, ‘head’ and ‘tail ‘commands. To read the entire file, ‘cat’, ‘more’, and ‘less ‘commands are used. But when the specific part of the file is required to read then ‘head ‘and ‘tail ‘commands are used to do that task.

‘head ‘command is used to read the file from the beginning and the ‘tail ‘command is used to read the file from the ending. How you can use ‘head ‘and ‘tail ‘commands with different options to read the particular portion of a file

|  |  |
| --- | --- |
| -n or –lines | It is used to print the first n number of lines. |
| -c or –bytes | It is used to print the first n number of characters or bytes. |
| -q or –quiet | It is used to print the content of one or more files without mentioning the filename in the output. |
| -v or –verbose | It is used to print the content of one or more files by mentioning the filename in the output. |

**1.8. mkdir**

To create directory and subdirectory

**Syntax :** mkdir <dir name>

Mkdir –p <dir>/<subdir>/<subdir>/… (to create directory tree recursively)

**Linux Dir structure:**

* **/bin** – binary or executable programs.
* **/etc** – system configuration files.
* **/home** – home directory. It is the default current directory.
* **/opt**– optional or third-party software.
* **/tmp** – temporary space, typically cleared on reboot.
* **/usr** – User related programs.
* **/var** – log files.
* **/boot-** It contains all the boot-related information files and folders such as conf, grub..
* **/dev –** It is the location of the device files such as dev/sda1, dev/sda2, etc.
* **/lib –** It contains kernel modules and a shared library.
* **/lost+found –** It is used to find recovered bits of corrupted files.
* **/proc**– It is a virtual and pseudo-file system to contains info about the running processes with a specific process ID or PID.
* **/sbin –** binary executable programs for an administrator.

**1.9 touch**

To create file

Syntax: touch <filename>

**1.10 cd (change directory)**

Cd command will used to switch from one location to different location in terms of

### [Absolute and Relative Pathnames](https://www.geeksforgeeks.org/absolute-relative-pathnames-unix/" \l ":~:text=An%20absolute%20path%20is%20defined%20as%20specifying%20the%20location%20of,actual%20file%20system%20from%20%2F%20directory.&text=Relative%20path%20is%20defined%20as,present%20working%20directly(pwd).)

**..** Denotes parent directory

**.** Denotes current directory

**~** Denotes current login user home directory

### Syntax : cd <option/ Absolute/Relative Pathnames>

**1.11 w/who/whoami**

**who command** shows a list of logged-in users, and the system’s current run level.

**w command** displays user information like user id and activities on the system. It also gives the knowledge of the system’s running time along with the system load average and is used to determine when the system has booted last time

**whoami command**is used to print the user’s name from which they are currently logged-in.

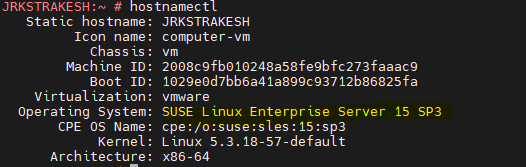
**1.12 know OS and version**

**RedHat :** cat /etc/redhat-release

**SUSE:** cat /etc/os-release

Use below command:

$hostnamectl



**1.13 pwd**

This command we used to know our present working directory

**Syntax : pwd**

**1.14 grep**

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression

**Syntax:** 

**grep [options] pattern [files]**

**a). Case insensitive search:**The -i option enables to search for a string case insensitively in the given file. It matches the words like “UNIX”, “Unix”, “Unix”. 

**$grep -i "UNix" geekfile.txt**

**Output:** 

Unix is great os. Unix was developed in Bell labs.

Unix Linux which one you choose.

uNix is easy to learn. Unix is a multiuser os. Learn Unix .Unix is a powerful.

**b). Display the file names that matches the pattern:**We can just display the files that contains the given string/pattern. 

**$grep -l "unix" \***

**or**

**$grep -l "unix" f1.txt f2.txt f3.xt f4.txt**

**Output:** 

geekfile.txt

**c). Checking for the whole words in a file :**By default, grep matches the given string/pattern even if it is found as a substring in a file. The -w option to grep makes it match only the whole words. 

**$ grep -w "unix" geekfile.txt**

**Output:** 

Unix is great os. Unix was developed in Bell labs.

uNix is easy to learn. Unix is a multiuser os. Learn Unix .Unix is a powerful.

**d). Matching the lines that end with a string:**The $ regular expression pattern specifies the end of a line. This can be used in grep to match the lines which end with the given string or pattern. 

**$ grep "os$" geekfile.txt**

**1.15 | (pipe)**

Pipes help combine two or more commands and are used as input/output concepts in a command.

## Syntax

Command 1 | command 2 | command 3 | ……

**Examples:**

$ Cat file1.txt | sort

$ Cat file2.txt | sort | uniq

$ cat file2.txt | sort  | uniq > list4.txt

$ Cat file2.txt | head -4

$ cat sample2.txt | head -7 | tail -5

$ ls | wc –l

$ Cat file3.txt | grep “yasin” | tee file4.txt | wc –l

**1.16 locate**

The **locate** command is a Unix utility used for quickly finding files and directories. The command is a more convenient and efficient alternative to the [**find** command](https://phoenixnap.com/kb/guide-linux-find-command), which is more aggressive and takes longer to complete the search.

Opposite to **find**, the **locate** command doesn't search the entire filesystem, but looks through a regularly updated file database in the system. Thus, the search completes much faster.

Syntax:

updatedb

locate <option> <filename>

**1.17 find**

**find command** can be used search file/dir in a variety of conditions like you can find files by **permissions**, **users**, **groups**, **file types**, **date**, **size**, and other possible criteria.

**Examples:**

**a). Find Files Using Name in Current Directory**

Find all the files whose name is tecmint.txt in a current working directory.

# find . -name tecmint.txt

**b). Find Files Under Home Directory**

Find all the files under /home directory with the name tecmint.txt.

# find /home -name tecmint.txt

**c). Find Files Using Name and Ignoring Case**

Find all the files whose name is tecmint.txt and contains both capital and small letters in /home directory.

# find /home -iname tecmint.txt

**d). Find Directories Using Name**

Find all directories whose name is Tecmint in / directory.

# find / -type d -name Tecmint

**e). Find PHP Files Using Name**

Find all php files whose name is tecmint.php in a current working directory.

# find . -type f -name tecmint.php

**f). Find all PHP Files in the Directory**

Find all php files in a directory.

# find . -type f -name "\*.php"

**Part II – Find Files Based on their Permissions**

**a). Find Files With 777 Permissions**

Find all the files whose permissions are 777.

# find . -type f -perm 0777 -print

**b). Find Files Without 777 Permissions**

Find all the files without permission 777.

# find / -type f ! -perm 777

**c). Find SGID Files with 644 Permissions**

Find all the SGID bit files whose permissions are set to 644.

# find / -perm 2644

**d). Find Sticky Bit Files with 551 Permissions**

Find all the Sticky Bit set files whose permission is 551.

# find / -perm 1551

**e). Find Files with 777 Permissions and Chmod to 644**

Find all 777 permission files and use the chmod command to set permissions to 644.

# find / -type f -perm 0777 -print -exec chmod 644 {} \;

**f). Find Directories with 777 Permissions and Chmod to 755**

# find / -type d -perm 777 -print -exec chmod 755 {} \;

**g). Find and remove single File**

To find a single file called tecmint.txt and remove it.

# find . -type f -name "tecmint.txt" -exec rm -f {} \;

**h). Find all Empty Files**

To find all empty files under a certain path.

# find /tmp -type f -empty

**i). Find all Empty Directories**

To file all empty directories under a certain path.

# find /tmp -type d -empty

**Part III – Search Files Based On Owners and Groups**

**a). Find Single File Based on User**

To find all or single files called tecmint.txt under / root directory of owner root.

# find / -user root -name tecmint.txt

**b). Find all Files Based on User**

To find all files that belong to user Tecmint under /home directory.

# find /home -user tecmint

**c). Find all Files Based on Group**

To find all files that belong to the group Developer under /home directory.

# find /home -group developer

**d). Find Particular Files of User**

To find all .txt files of user Tecmint under /home directory.

# find /home -user tecmint -iname "\*.txt"

**Part IV – Find Files and Directories Based on Date and Time**

**a). Find Last 50 Days Modified Files**

To find all the files which are modified 50 days back.

# find / -mtime 50

**b). Find Last 50 Days Accessed Files**

To find all the files which are accessed 50 days back.

# find / -atime 50

**c). Find Last 50-100 Days Modified Files**

To find all the files which are modified more than 50 days back and less than 100 days.

# find / -mtime +50 –mtime -100

**d). Find Changed Files in Last 1 Hour**

To find all the files which are changed in the last 1 hour.

# find / -cmin -60

**e). Find Modified Files in Last 1 Hour**

To find all the files which are modified in the last 1 hour.

# find / -mmin -60

**f). Find Accessed Files in Last 1 Hour**

To find all the files which are accessed in the last 1 hour.

# find / -amin -60

**Part V – Find Files and Directories Based on Size**

**a). Find 50MB Files**

To find all 50MB files, use.

# find / -size 50M

**b). Find Size between 50MB – 100MB**

To find all the files which are greater than 50MB and less than 100MB.

# find / -size +50M -size -100M

**c). Find and Delete 100MB Files**

To find all 100MB files and delete them using one single command.

# find / -type f -size +100M -exec rm -f {} \;

**1.18 uniq, diff, sort**

Ordering and manipulating data in Linux-based text files can be carried out using the **sort** and **uniq** utilities.

**Sort:** command orders a list of items both alphabetically and numerically

**Uniq:** command removes adjacent duplicate lines in a list.

**diff** is a command-line utility that allows you to compare two files line by line. It can also compare the contents of directories. The diff command is most commonly used to create a patch containing the differences between one or more files that can be applied using the patch command.

Example: diff file1 file2

**1.19: wc**

wc command **calculates a file's word, line, character, or byte count**. Far from just being a utility for word processing,

exp: wc –l/-c/-w/-s

**1.20 : du**

The **du** (disk usage) command measures the disk space occupied by files or directories. By default, it measures the current directory and all its subdirectories, printing totals in blocks for each, with a grand total at the bottom.

$du file1 file2

$du –sh \*

**1.21: ln**

**ln command** is a Unix command for linking files or directories to each other. Essentially, it creates new files with the names you specify, and refer them to already existing files or directories. When you run any Unix command against a symlink, it is first resolved (the original file it points to is confirmed) and the Unix command works with that file to produce desired

## Two types of linking files and directories

There are two common approaches to link a file or directory in Unix: soft linking and hard linking. Soft links are also called symlinks (symbolic links).

### What is a soft link?

**Soft link** (also referred to as **symlink** – short for symbolic link) is a special type of file in Unix, which references another file or directory. Symlink contains the name for another file and contains no actual data. To most commands, symlinks look like a regular file, but all the operations (like reading from a file) are referred to the file the symlink points to.

When you remove a soft link, you simply remove one of the pointers to the real file. When you remove the original file a soft link points to, your data is lost. Even though your soft link will still exist, it will be pointing to the non-existent file and will therefore be useless (it will probably have to be removed as well).

### What is a hard link?

**Hard link** is a pointer to physical data. Effectively, all standard files are hard links, because they ultimately create an association between a file name and a physical data which corresponds to each file.

In Unix, you can create as many hard links to a file as you like, and there is even a special counter for such references. When you’re using the long format of an ls command, you can see this counter.

When you remove a hard link, you decrease this link counter for a data on your storage. If you remove the original file, the data will not be lost as long as there’s at least one hard link pointing to it.

$ln <option> <source file> <target file>

**1.22 df**

The **df** command (short for disk free), is used to display information related to file systems about total space and available space.

Example: df –h/ df –th/df .

Make sure all mount points are maintained in file /etc/fstab

**1.23 su**

**Su** command is used to switch user from one user to other user in same terminal in Linux

**1.24 uptime**

Uptime will used to check since how long vm is up and running or last reboot taken

**1.25 history**

History command will used to get old executed commands from terminal

**1.26 fg and bg**

* [**ctrl-z**](https://afni.nimh.nih.gov/pub/dist/edu/data/CD.expanded/AFNI_data6/unix_tutorial/misc/unix_commands.html#u-mcc-ctrl-z)**:** keystroke: suspend the foreground process
* [**bg**](https://afni.nimh.nih.gov/pub/dist/edu/data/CD.expanded/AFNI_data6/unix_tutorial/misc/unix_commands.html#u-mc-bg)**:** put the recently suspended process in the background
* [**fg**](https://afni.nimh.nih.gov/pub/dist/edu/data/CD.expanded/AFNI_data6/unix_tutorial/misc/unix_commands.html#u-mc-fg)**:** put the recently suspended process in the foreground
* [**&**](https://afni.nimh.nih.gov/pub/dist/edu/data/CD.expanded/AFNI_data6/unix_tutorial/misc/unix_commands.html#u-mcc-amp)**:** run a program in the background to begin with
* [**jobs**](https://afni.nimh.nih.gov/pub/dist/edu/data/CD.expanded/AFNI_data6/unix_tutorial/misc/unix_commands.html#u-mc-jobs)**:** list child processes under terminal shell

**1.27 contab**

Crontab stands for “cron table”. It allows to use job scheduler, which is known as cron to execute tasks at OS level

**Why use Cronjobs?**

Here are the reasons for using Cronjobs in Linux:

* Helps OS to take a scheduled backup of log files or database.
* Delete old log files
* Archive and purge database tables
* Send out any notification email such as Newsletters, Password expiration email
* Regular clean-up of cached data
* Crontab is an ideal option to automate Unix jobs.
* It is used to automate system maintenance
* crontab [ -u user ] file
* crontab [ -u user ] { -l | -r | -e }
* crontab [ -u user ] [ -i ] { -e | -l | -r }
* crontab [ -u user ] [ -l | -r | -e ] [-i] [-s]
* \* \* \* \* \* Command\_to\_execute
* | | | | |
* | | | | Day of the Week ( 0 - 6 ) ( Sunday = 0 )
* | | | |
* | | | Month ( 1 - 12 )
* | | |
* | | Day of Month ( 1 - 31 )
* | |
* | Hour ( 0 - 23 )
* |
* Min ( 0 - 59 )

## Crontab Options

To Install or update job in crontab, use -e option:

$ crontab -e

To List Crontab entries, use -l option:

$ crontab -l

To Deinstall job from crontab, use -r option:

$ crontab -r

To Confirm Deinstall of job from crontab, use -i option:

$ crontab -i -r

To edit other user crontab, user -u option and specify username:

$ crontab -u username -e

To List other user crontab entries:

$ crontab -u username -l

## EXAMPLES

To run /usr/bin/sample.sh at 12.59 every day and supress the output

59 12 \* \* \* simon /usr/bin/sample.sh > /dev/null 2>&1

To run sample.sh everyday at 9pm (21:00)

0 21 \* \* \* sample.sh 1>/dev/null 2>&1

To run sample.sh every Tuesday to Saturday at 1am (01:00)

0 1 \* \* 2-7 sample.sh 1>/dev/null 2>&1

To run sample.sh at 07:30, 09:30 13:30 and 15:30

30 07,09,13,15 \* \* \* sample.sh

To run sample.sh at 07:30, 09:30 13:30 and 15:30

30 07,09,13,15 \* \* \* sample.sh

To run sample.sh at 2am daily.

0 2 \* \* \* sample.sh ( This is widely used in cases of backup of files/databases etc on daily basis. )

To run sample.sh twice a day. say 5am and 5pm

0 5,17 \* \* \* sample.sh

To run sample.sh every minutes.

\* \* \* \* \* sample.sh

**1.28 fdisk**

**fdisk** also known as format disk is a dialog-driven command in Linux used for creating and manipulating disk partition table. It is used for the view, create, delete, change, resize, copy and move partitions on a hard drive using the dialog-driven interface.   
fdisk allows you to create a maximum of four primary partitions and the number of logical partition depends on the size of the hard disk you are using. It allows the user:

* To Create space for new partitions.
* Organizing space for new drives.
* Re-organizing old drives.
* Copying or Moving data to new disks(partitions).

**Syntax:**

fdisk [options] device

or

fdisk -l [device...]

**1.29: mount and umount**

The **mount** command [mounts](https://www.computerhope.com/jargon/m/mount.htm) a [storage device](https://www.computerhope.com/jargon/s/stordevi.htm) or [filesystem](https://www.computerhope.com/jargon/f/filesyst.htm), making it accessible and attaching it to an existing [directory](https://www.computerhope.com/jargon/d/director.htm) structure.

The **umount** command "unmounts" a mounted filesystem, informing the system to complete any pending [read](https://www.computerhope.com/jargon/r/read.htm) or [write](https://www.computerhope.com/jargon/w/write.htm) operations, and safely detaching it.

Syntax : mount [OPTION...] DEVICE\_NAME DIRECTORY

Example :

1. mount /dev/sdb1 /mnt/media

2. mount -t TYPE DEVICE\_NAME DIRECTORY

3. umount <mountpoint>

**Note:**

If the /etc/fstab contains information about the given file system, the mount command uses the value for the other parameter and the mount options specified in the fstab file.

The /etc/fstab file contains a list of entries in the following form:

/etc/fstab

[File System] [Mount Point] [File System Type] [Options] [Dump] [Pass]

**1.32 vi**

The default editor that comes with the UNIX/LINUX operating system is called vi (visual editor). The UNIX vi editor is a full screen editor and has two modes of operation:

**Command mode** commands which cause action to be taken on the file, and

**Insert mode** in which entered text is inserted into the file.

In the command mode, every character typed is a command that does something to the text file being edited; a character typed in the command mode may even cause the vi editor to enter the insert mode. In the insert mode, every character typed is added to the text in the file; pressing the <Esc> (Escape) key turns off the Insert mode.

**To Start vi**

To use vi on a file, type in vi filename. If the file named filename exists, then the first page (or screen) of the file will be displayed; if the file does not exist, then an empty file and screen are created into which you may enter text.

vi filename edit filename starting at line 1

vi -r filename recover filename that was being edited when system crashed

**To Exit vi**

Usually the new or modified file is saved when you leave vi. However, it is also possible to quit vi without saving the file.

Note: The cursor moves to bottom of screen whenever a colon (:) is typed. This type of command is completed by hitting the <Return> (or <Enter>) key.

**:x<Return>** quit vi, writing out modified file to file named in original invocation

**:wq<Return>** quit vi, writing out modified file to file named in original invocation

**:q<Return>** quit (or exit) vi

**:q!<Return>** quit vi even though latest changes have not been saved for this vi call

**Moving the Cursor:**

* + **j or <Return>** move cursor down one line
  + **k [or up-arrow]** move cursor up one line
  + **h or <Backspace>**move cursor left one character
  + **l or <Space>**move cursor right one character
  + **0 (zero)** move cursor to start of current line (the one with the cursor)
  + **$** move cursor to end of current line
  + **w** move cursor to beginning of next word
  + **b** move cursor back to beginning of preceding word
  + **:0<Return> or 1G** move cursor to first line in file
  + **:n<Return> or nG** move cursor to line n
  + **:$<Return> or G** move cursor to last line in file

**Inserting or Adding Text**

The following commands allow you to insert and add text. Each of these commands puts the vi editor into insert mode; thus, the <Esc> key must be pressed to terminate the entry of text and to put the vi editor back into command mode.

**i** insert text before cursor, until <Esc> hit

**I** insert text at beginning of current line, until <Esc> hit

**a** append text after cursor, until <Esc> hit

**A** append text to end of current line, until <Esc> hit

**o** open and put text in a new line below current line, until <Esc> hit

**O** open and put text in a new line above current line, until <Esc> hit

**Changing Text**

The following commands allow you to modify text.

**r** replace single character under cursor (no <Esc> needed)

**R** replace characters, starting with current cursor position, until <Esc> hit

**cw** change the current word with new text,

starting with the character under cursor, until <Esc> hit

**cNw** change N words beginning with character under cursor, until <Esc> hit;

**C** change (replace) the characters in the current line, until <Esc> hit

**cc** change (replace) the entire current line, stopping when <Esc> is hit

**Ncc or cNc** change (replace) the next N lines, starting with the current line,

**Deleting Text**

The following commands allow you to delete text.

**x** delete single character under cursor

**Nx** delete N characters, starting with character under cursor

**dw** delete the single word beginning with character under cursor

**dNw** delete N words beginning with character under cursor;

**D** delete the remainder of the line, starting with current cursor position

**dd** delete entire current line

**Ndd or dNd** delete N lines, beginning with the current line;

**Cutting and Pasting Text**

The following commands allow you to copy and paste text.

**yy** copy (yank, cut) the current line into the buffer

**Nyy or yNy** copy (yank, cut) the next N lines, including the current line, into the buffer

**p** put (paste) the line(s) in the buffer into the text after the current line

**Searching Text**

A common occurrence in text editing is to replace one word or phase by another. To locate instances of particular sets of characters (or strings), use the following commands.

**/string** search forward for occurrence of string in text

**?string** search backward for occurrence of string in text

**n** move to next occurrence of search string

**N** move to next occurrence of search string in opposite direction

**Determining Line Numbers**

Being able to determine the line number of the current line or the total number of lines in the file being edited is sometimes useful.

**:.=** returns line number of current line at bottom of screen

**:=** returns the total number of lines at bottom of screen

**^g** provides the current line number, along with the total number of lines,

in the file at the bottom of the screen

**Saving and Reading Files**

These commands permit you to input and output files other than the named file with which you are currently working.

**:r filename<Return>** read file named filename and insert after current line

(the line with cursor)

**:w<Return>** write current contents to file named in original vi call

**:w newfile<Return>** write current contents to a new file named newfile

**:12,35w smallfile<Return>** write the contents of the lines numbered 12 through 35 to a new file named smallfile

**:w! prevfile<Return**> write current contents over a pre-existing file named prevfile

**\*\*\*\*\*\*\*\*\*IMP\*\*\*\*\*\*\***

**Replace a word (IMP):**

**:**<startLine,endLine> s/<oldString>/<newString>/g

**Example:**

**Commands Action**

**:1,$ s/readable/changed/** Replace forward with backward from first line to the last line

**:3,6 s/letters/neww/g** Replace forward with backward from third line to the ninth line

**2 Data copy and move and backup and extract and archive**

**2.1 cp**

cp command copies files (or, optionally, directories). The copy is completely independent of the original. You can either copy one file to another, or copy arbitrarily many files to a destination directory.

**Examples**

**a). Copy file in current directory itself**.

$ cp viewers\_list.txt users\_list.txt

**b). Copy a file in ‘backup’ directory**

$ cp viewers\_list.txt backup/

**c). Copy in ‘backup’ directory with different name**

$ cp viewers\_list.txt backup/viewers\_list.bak

**d). Copy multiple files to a specified directory, in this case 'news'**

$ cp current\_news.txt headlines.txt cover\_story.txt news/

**e). Copy multiple files using wild card. It copies all files with extension .txt to 'newsportal' directory.**

$ cp \*.txt newsportal/

7. To create a backup of the destination file, if that already exists, use -b or --backup option.

$ cp -b viewers\_list.txt users\_list.txt

$ ls -lt

**f). To create backup with specific suffix, use -S or --sufix= option.**

$ cp -S .bak viewers\_list.txt users\_list.txt

$ cp -v -b --suffix=.bak mount.htm mount\_old

'mount.htm' -> 'mount\_old' (backup: 'mount\_old.bak')

**g). To copy only when source file is newer than the destination file we can use -u option.**

$ cp -u headlines.txt current\ news/

$ cp --update headlines.txt current\ news/

**2.2 mv**

The mv command is one of the basic Linux commands that is used to move the files and directories from one location to another. It is also used to rename the files and directories. The mv command is available on all Linux distributions by default.

$ mv <sourcefile/dir> <targetfile/dir>

**2.3 scp**

**SCP** **(Secure Copy Protocol)**is a network protocol used to securely copy files/folders between Linux ([Unix](https://phoenixnap.com/glossary/what-is-unix)) systems on a network. To transmit, use the scp command line utility, a safer variant of the [cp (copy) command](https://phoenixnap.com/kb/how-to-copy-files-directories-linux).SCP protects your data while copying across an [SSH (Secure Shell)](https://phoenixnap.com/kb/what-is-ssh) connection by encrypting the files and the [passwords](https://phoenixnap.com/blog/strong-great-password-ideas). Therefore, even if the traffic is intercepted, the information is still encrypted.

**Example:**

scp -rpc source\_file\_name username@destination\_host:destination\_folder

**2.4 rsync**

**Rsync** (**Remote Sync**) is the most commonly used command for [copying and synchronizing files and directories remotely](https://www.tecmint.com/sync-new-changed-modified-files-rsync-linux/) as well as **locally** in **Linux**/**Unix** systems.

With the help of the **rsync** command, you can copy and synchronize your data remotely and locally across directories, disks, and networks, perform data backups, and [mirror between two Linux machines](https://www.tecmint.com/clone-centos-server/).

**Examples:**

rsync -zvh backup.tar.gz /tmp/backups/

rsync -avzh /root/rpmpkgs [root@192.168.0.141:/root/](mailto:root@192.168.0.141:/root/)

**2.5 rm/rmdir**

**rm and rmdir commands**

rm and rmdir are the most widely used command to remove files and directories in Linux-based operating systems. If you are a new user in Linux then you should be very careful while running rm and rmdir command because you cannot recover the files and directory after deleting them. we will explain how to use rm and rmdir commands to remove files and directories in linux.

**rm Command**

rm also stands for remove is a command line tool used to remove objects such as, files, directories and symbolic links from the file system. You can also remove special files such as, pipes, sockets and device nodes with the rm command.

**Syntax:**

rm [-f] [-i] [-I] [-r] [-d] [--no-preserve-root] [--one-file-system] [--preserve-root] [FILENAME]

Where:

-f : This will overrides minor protection and removes the file forcefully without prompt.

-i : This will ask the user for confirmation before removing each file.

-I : This will prompt you once before removing more than three files.

-r : This will delete all the files and sub-directories recursively

-d : This will used to remove empty directory.

Examples:

a). To remove file interactively, use option -i with rm command. This will prompt you before deleting a file.

$rm -i file1.txt

b).If you want to remove more than three files and prompt only once before removing them use -I option with the rm command:

$rm -I file1.txt fil2.txt file3.txt file4.txt

c).To remove multiple files with verbose output run the following command:

$rm -v file1.txt fil2.txt file3.txt file4.txt

d).To remove an empty directory run the following command:

$rm -d dir1

e).Remove Directory with rmdir Command

$rmdir dir1

f).To remove multiple directories run the following command:

$rmdir dir1 dir2 dir3

**2.6 Zip/unzip**

To create a zip file, you will need to provide the name for the zipped archive and the files that need to be included in the zip.

a). $ zip options zip\_file file1 file2...

b). $ zip test.zip testfile.txt

c). $ zip files.zip testfile1.txt testfile2.txt testfile3.txt testfile4.txt

d). $ zip -u files1.zip testfile4.txt

e). $ zip -d zip\_file filename

f). $ zip -r zip\_file directoryname

g) $ unzip test.zip

h).$ unzip zip\_file -d /path/to/directory

**2.7 rar/unrar**

RAR files are in compressed archive format, if you have downloaded rar files from the Internet, you need to unpack or unrar them (extract rar files).

a). To open/extract a RAR file in the current working directory, just use the following command with unrar e option.

# unrar tecmint.rar

b). To open/extract a RAR file in a specific path or destination directory, just use the unrar e option, it will extract all the files in the specified destination directory.

# unrar e tecmint.rar /home/

**3. User management, user groups and file & directory permissions**

**useradd, userdel, usermod, passwd, id**

User management includes everything from creating a user to deleting a user on your system. User management can be done in three ways on a Linux system.

**Command line tools** include commands like **useradd, userdel, passwd**, etc. These are mostly used by the server administrators.

The local user database in Linux is **/etc/passwd** , **/etc/shadow** directory.

Creating user at linux :

* **useradd**

With useradd commands you can add a user.

useradd -m -d /home/<userName> -c "<userName>" <userName>

useradd -m -d /home/xyz -c "xyz" xyz  

* **userdel**

To delete a user account userdel command is used.

userdel -r <userName>

## usermod

The command usermod is used to modify the properties of an existing user.

usermod -c <'newName'> <oldName>

usermod -s <newShell> <userName>

**3.1 chsh**

Users can change their login shell with chsh command.

Both the command**chsh** and**chsh -s** will work to change the shell.

**3.2 Id <username> to get user details**

**3.3 groupadd,groupmod, groupdel**

/etc/group is the database in linux to store group info

#groupadd –g 5000 rhcegroup

#groupmod –n [New Group Name] [Old Group Name]

#groupmod –g [New Group GID] [Group Name]

#groupdel [Group Name]

**3.4 chown**

The **chown** command changes user ownership of a file, directory, or link in Linux. Every file is associated with an owning user or group. It is critical to configure file and folder permissions properly.

chown NewUser:NewGroup FILE

chown linuxuser:group3 sample3

How to Recursively Change File Ownership

The chown command allows changing the ownership of all files and subdirectories within a specified directory. Add the -R option to the command to do so:

chown -R NewUser:NewGroup DirNameOrPath

In the following example, we will recursively change the owner and the group for all files and directories in Dir1.

chown -R linuxuser:group3 Dir1

chgrp <groupname> file or dir

**3.5 chmod**

To change directory permissions in Linux, use the following:

chmod +rwx filename to add permissions

chmod -rwx directoryname to remove permissions.

chmod +x filename to allow executable permissions.

chmod -wx filename to take out write and executable permissions.

Note that “r” is for read, “w” is for write, and “x” is for execute.

This only changes the permissions for the owner of the file.

Where to input chmod commands in terminal

Diagram of the components in chmod

What are the three permission groups?

There are three options for permission groups available to you in Linux. These are

owners: these permissions will only apply to owners and will not affect other groups.

groups: you can assign a group of users specific permissions, which will only impact users within the group.

all users: these permissions will apply to all users, and as a result, they present the greatest security risk and should be assigned with caution.

What are the three kinds of file permissions in Linux?

There are three kinds of file permissions in Linux:

Read (r): Allows a user or group to view a file.

Write (w): Permits the user to write or modify a file or directory.

Execute (x): A user or grup with execute permissions can execute a file or view a directory.

More ways to manage permissions

Here's a more comprehensive list of ways you can manage file permissions, groups, and ownership beyond the basic commands listed at the top of this guide.

How to Change Directory Permissions in Linux for the Group Owners and Others

The command for changing directory permissions for group owners is similar, but add a “g” for group or “o” for users:

chmod g+w filename

chmod g-wx filename

chmod o+w filename

chmod o-rwx foldername

To change directory permissions for everyone, use “u” for users, “g” for group, “o” for others, and “ugo” or “a” (for all).

chmod ugo+rwx foldername to give read, write, and execute to everyone.

chmod a=r foldername to give only read permission for everyone.

Terminal view of changing permissions for groups

How to Change Groups of Files and Directories in Linux

By issuing these commands, you can change groups of files and directories in Linux.

chgrp groupname filename

chgrp groupname foldername

Note that the group must exit before you can assign groups to files and directories.

Changing groups of files in the terminal

Changing ownership in Linux

Another helpful command is changing ownerships of files and directories in Linux:

chown name filename

chown name foldername

Viewing chown name filename

These commands will give ownership to someone, but all sub files and directories still belong to the original owner.

You can also combine the group and ownership command by using:

chown -R name:filename /home/name/directoryname

chown -R to combine group and ownership command

Changing Linux permissions in numeric code

You may need to know how to change permissions in numeric code in Linux, so to do this you use numbers instead of “r”, “w”, or “x”.

0 = No Permission

1 = Execute

2 = Write

4 = Read

Basically, you add up the numbers depending on the level of permission you want to give.

chmod 777 command using numeric code

Permission numbers are:

0 = ---

1 = --x

2 = -w-

3 = -wx

4 = r--

5 = r-x

6 = rw-

7 = rwx

For example:

chmod 777 foldername will give read, write, and execute permissions for everyone.

chmod 700 foldername will give read, write, and execute permissions for the user only.

chmod 327 foldername will give write and execute (3) permission for the user, w (2) for the group, and read, write, and execute for the users.

**Special permission explained**

Special permissions make up a fourth access level in addition to user, group, and other. Special permissions allow for additional privileges over the standard permission sets (as the name suggests). There is a special permission option for each access level discussed previously. Let's take a look at each one individually, beginning with Set UID:

user + s (pecial)

Commonly noted as SUID, the special permission for the user access level has a single function: A file with SUID always executes as the user who owns the file, regardless of the user passing the command. If the file owner doesn't have execute permissions, then use an uppercase S here.

Now, to see this in a practical light, let's look at the /usr/bin/passwd command. This command, by default, has the SUID permission set:

[tcarrigan@server ~]$ ls -l /usr/bin/passwd

-rwsr-xr-x. 1 root root 33544 Dec 13 2019 /usr/bin/passwd

Note the s where x would usually indicate execute permissions for the user.

group + s (pecial)

Commonly noted as SGID, this special permission has a couple of functions:

If set on a file, it allows the file to be executed as the group that owns the file (similar to SUID)

If set on a directory, any files created in the directory will have their group ownership set to that of the directory owner

[tcarrigan@server article\_submissions]$ ls -l

total 0

drwxrws---. 2 tcarrigan tcarrigan 69 Apr 7 11:31 my\_articles

Great Linux resources

Advanced Linux commands cheat sheet

Download RHEL 9 at no charge through the Red Hat Developer program

A guide to installing applications on Linux

Linux system administration skills assessment

Cheat sheet: Old Linux commands and their modern replacements

This permission set is noted by a lowercase s where the x would normally indicate execute privileges for the group. It is also especially useful for directories that are often used in collaborative efforts between members of a group. Any member of the group can access any new file. This applies to the execution of files, as well. SGID is very powerful when utilized properly.

As noted previously for SUID, if the owning group does not have execute permissions, then an uppercase S is used.

other + t (sticky)

The last special permission has been dubbed the "sticky bit." This permission does not affect individual files. However, at the directory level, it restricts file deletion. Only the owner (and root) of a file can remove the file within that directory. A common example of this is the /tmp directory:

[tcarrigan@server article\_submissions]$ ls -ld /tmp/

drwxrwxrwt. 15 root root 4096 Sep 22 15:28 /tmp/

The permission set is noted by the lowercase t, where the x would normally indicate the execute privilege.

Setting special permissions

To set special permissions on a file or directory, you can utilize either of the two methods outlined for standard permissions above: Symbolic or numerical.

Let's assume that we want to set SGID on the directory community\_content.

To do this using the symbolic method, we do the following:

[tcarrigan@server article\_submissions]$ chmod g+s community\_content/

Using the numerical method, we need to pass a fourth, preceding digit in our chmod command. The digit used is calculated similarly to the standard permission digits:

Start at 0

SUID = 4

SGID = 2

Sticky = 1

The syntax is:

[tcarrigan@server ~]$ chmod X### file | directory

Where X is the special permissions digit.

Here is the command to set SGID on community\_content using the numerical method:

[tcarrigan@server article\_submissions]$ chmod 2770 community\_content/

**3.6 SUDO**

sudo (**S**uper **U**ser **DO**) command in Linux is generally used as a prefix of some command that only superuser are allowed to run. If you prefix **“sudo”** with any command, it will run that command with elevated privileges or in other words allow a user with proper permissions to execute a command as another user, such as the superuser. This is the equivalent of “run as administrator” option in Windows. The option of sudo lets us have multiple administrators.   
These users who can use the **sudo** command need to have an entry in the **sudoers** file located at **“/etc/sudoers”**. Remember that to edit or view the sudoers file you have to use sudo command. To edit the sudoers file it is recommended to use “visudo” command.   
By default, sudo requires that users authenticate themselves with a password which is the user’s password, not the root password itself.

**sudoers file:**

**cat /etc/sudoers**

**4 Performance tuning commands:**

**4.1 free**

The Linux free command outputs a summary of RAM usage, including total, used, free, shared, and available memory and swap space. The command helps [monitor resource usage](https://phoenixnap.com/kb/top-command-in-linux) and allows an admin to determine if there's enough room for running new programs.

Example:

$free –h

$free –s 2

$free –g/m/k/b

$ cat /proc/meminfo

**4.2 lscpu**

The command-line utility “lscpu” in Linux is used to get CPU information of the system. The “lscpu” command fetches the CPU architecture information from the “sysfs” and /proc/cpuinfo files and displays it in a terminal

$lscpu

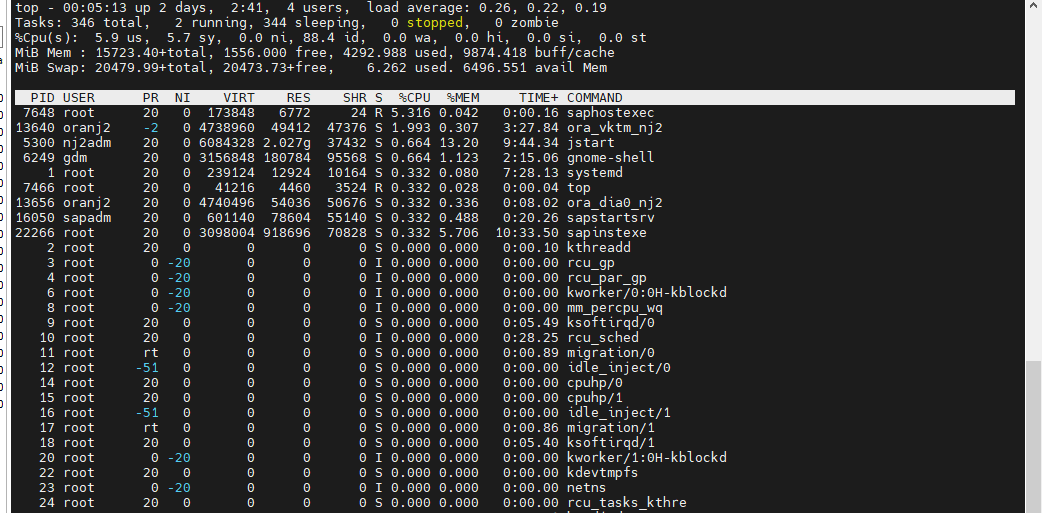
$cat /proc/cpuinfo

**4.3 lsblk**

lsblk lists information about all available or the specified block devices. The lsblk command reads the sysfs filesystem and udev db to gather information. If the udev db is not available or lsblk is compiled without udev support, then it tries to read LABELs, UUIDs and filesystem types from the block device

**4.4 top**

The **top** command (table of processes) displays the processor activity of your Linux box and also displays tasks managed by the kernel in real-time. It also shows information about CPU and memory utilization of a [list of running processes](https://www.tecmint.com/find-linux-processes-memory-ram-cpu-usage/).



**Examples:**

a). List All Running Linux Processes

To list all running Linux Processes, simply type top on the command line to get the information of running tasks, memory, cpu, and swap. Press ‘q‘ to quit window.

# top

b). Sort Linux Processes by PID

c). Sort Linux Processes by Memory and CPU Usage

d). Sort Linux Processes by Running Time

e). Display Linux Processes By Specific User

To display all user-specific running processes information, use the -u option will list specific User process details.

# top -u tecmint

f). Highlight Running Process in Top

Press ‘z‘ option will display the running process in color which may help you to identify the running process easily.

g). List Absolute Path of Running Linux Processes

Press ‘c‘ option in running top command will display the absolute path of the running process.

h). Set Refresh Interval for Top Command

By default screen refresh interval is set to 3.0 seconds, the same can be changed by pressing the ‘d‘ option in running the top command to set desired interval time.

i). Kill Running Linux Process Using Top Command

You can kill a process after finding the PID of the process by pressing the ‘k‘ option in running the top command without closing the top window as shown below.

j). Sort Linux Processes by CPU Utilisation

To sort all running processes by CPU utilization, simply press Shift+P key.

k). Save Top Command Results in File

To save the running top command results output to a file /root/.toprc use the following command.

# top -n 1 -b > top-output.txt

l). List Linux Idle Processes

Press 'i' to get the list of idle/sleeping processes.

m). Getting Top Command Help

Press the ‘h‘ option to obtain the top command help.

**4.5 ps**

In Linux, a process is any executing (running) instance of a program. Whenever you run a program, it creates a new process. Linux is a multi-tasking and multi-user operating system that allows running several programs simultaneously. So, there are multiple Linux processes running at the same time. There are different tools to view and track processes in the Linux system. This article will introduce the **ps** command in Linux which displays a list of currently running processes and their PIDs. ps is the short form for **Process Status**.

**Examples:**

a). If you run the ps command without any arguments, it displays processes for the current shell.

$ ps

Print All Processes in Different Formats

b). Display every active process on a Linux system in generic (Unix/Linux) format.

$ ps -A

OR

$ ps -e

c). Display all processes in BSD format.

$ ps -au

OR

$ ps -axu

d). To perform a full-format listing, add the -f or -F flag.

$ ps -ef

OR

$ ps -eF

**Display User Running Processes**

e). You can select all processes owned by you (runner of the ps command, root in this case), type:

$ ps -x

f). To select process by PPID, type.

$ ps -f --ppid 1154

g). Check the execution time of a process.

$ ps -eo comm,etime,user | grep httpd

The output below shows the HTTPD service has been running for 1 hour, 48 minutes, and 17 seconds.

**Troubleshoot Linux System Performance**

If your system isn’t working as it should be, for instance, if it’s unusually slow, you can perform some system troubleshooting as follows.

h). Find top running processes by highest memory and CPU usage in Linux.

$ ps -eo pid,ppid,rss,cmd,%mem,%cpu --sort=-%mem | head

OR

$ ps -eo pid,ppid,rss,cmd,%mem,%cpu --sort=-%cpu | head

i). To kill Linux processes/unresponsive applications or any process that is consuming high CPU time.

First, find the PID of the unresponsive process or application.

$ ps -A | grep -i stress

Then use the kill command to terminate it immediately.

$ kill -9 2583 2584

j). Finding a Zombie Process

ps aux | egrep "Z|defunct"

kill <pid> to kill the zombie process

**4.6 performance monitoring tools**

### 4.6.1 Virtual Memory Statistics (vmstat)

The Virtual Memory Statistics tool, vmstat, provides instant reports on your system's processes, memory, paging, block input/output, interrupts, and CPU activity. Vmstat lets you set a sampling interval so that you can observe system activity in near-real time.

vmstat is provided by the procps-ng package. For detailed information about using vmstat

**4.6.2 System Activity Reporter (sar)**

The System Activity Reporter, sar, collects and reports information about system activity that has occurred so far on the current day. The default output displays the current day's CPU usage at 10 minute intervals from the beginning of the day (00:00:00 according to your system clock).

You can also use the -i option to set the interval time in seconds, for example, sar -i 60 tells sar to check CPU usage every minute.

sar is a useful alternative to manually creating periodic reports on system activity with top. It is provided by the sysstat package. For detailed information about using sar

Exp:

# sar -r ALL -f sa20230822

**4.6.3  iostat**

The **iostat** tool, provided by the sysstat package, monitors and reports on system input/output device loading to help administrators make decisions about how to balance input/output load between physical disks. The **iostat** tool reports on processor or device utilization since **iostat** was last run, or since boot. You can focus the output of these reports on specific devices by using the parameters defined in the iostat(1) manual page. For detailed information on the await value and what can cause its values to be high

**4.6.4 mpstat**

The mpstat command writes to standard output activities for each available processor, processor 0 being the firstone. Global average activities among all processors are also reported. The mpstat command can be used both on SMP and UP machines, but in the latter, only global average activities will be printed. If no activity has been selected, then the default report is the CPU utilization report.

mpstat 2 5

Display five reports of global statistics among all processors at two second intervals.

mpstat -P ALL 2 5

Display five reports of statistics for all processors at two second intervals.

**5. Network commands and network tuning**

**5.1 ip/ifconfig**

The ip**command** is a Linux net-tool for system and network administrators. IP stands for Internet Protocol and as the name suggests, the tool is used for configuring network interfaces.Older Linux distributions used the [ifconfig command](https://phoenixnap.com/kb/centos-ifconfig" \t "_blank), which operates similarly. However, ifconfig has a limited range of capabilities compared to the ip command.

**Examples: ip, ip a show, ip a, ip route list, ip addr show**

**5.2. hostname**

The Linux hostname command is used to view or change a system’s domain and hostname. It can also check a computer’s IP address.

**Syntax: hostname <option>**

Options:

-a, --alias: Displays the alias name of the host.

-A, --all-fqdns: Displays every FQDN (Fully Qualified Domain Name) of the computer.

-b, --boot: Always set a hostname.

-d, --domain: Display DNS domain name.

-f, --fqdn, --long: Display the FQDN.

-F, --file: Check a file to recover and display the hostname.

-h, --help: Print the help message as the output.

-i, --ip-address: Display the computer’s IP address.

-I, --all-ip-addresses: Display all of the computer’s network addresses.

-s, --short: Display the short version of the hostname.

-v, --verbose: Expand all output to verbose.

-y, --yp, --nis: Display the NIS domain name.

**5.3 hostnamectl**

The hostname is an identity of the system and is used by the networks to search the system.

The “hostnamectl” is a Linux command that is used to set the hostname in the terminal without even opening and editing in the etc/hostname file of a system.Using the “hostnamectl” command, the user can edit the static, pretty, and transient hostname as well.

**$ hostnamectl set-hostname <hostname> –static**

**5.4 /etc/hosts**

As your machine gets started, it will need to know the mapping of some hostnames to IP addresses before DNS can be referenced. This mapping is kept in the /etc/hosts file. In the absence of a name server, any network program on your system consults this file to determine the IP address that corresponds to a host name.

Following is a sample /etc/hosts file:

|  |
| --- |
| IPAddress Hostname Alias  127.0.0.1 localhost deep.openna.com  208.164.186.1 deep deep.openna.com  208.164.186.2 mail mail.openna.com  208.164.186.3 web web.openna.com |

**6.5 wget**

we are going to review the wget utility which retrieves files from World Wide Web (WWW) using widely used protocols like HTTP, HTTPS, FTP, and FTPS.

Wget is a free command-line utility and network file downloader, which comes with many features that make file downloads easy, including:

$\* wget [option] [URL]

# wget http://ftp.gnu.org/gnu/wget/wget2-2.0.0.tar.gz

a). Wget Download File with Different Name

Using -O (uppercase) option, downloads files with the different file names. Here we have given the wget.zip file name as shown below.

# wget -O wget.zip http://ftp.gnu.org/gnu/wget/wget2-2.0.0.tar.gz

b). Wget Download Multiple Files with HTTP and FTP Protocol

Here we see how to download multiple files using HTTP and FTP protocol with the wget command at once.

$ wget http://ftp.gnu.org/gnu/wget/wget2-2.0.0.tar.gz ftp://ftp.gnu.org/gnu/wget/wget2-2.0.0.tar.gz.sig

c). Wget Download Files in Background

With -b option you can send download in the background immediately after download start and logs are written in wget.log file.

$ wget -b wget.log https://releases.ubuntu.com/20.04.3/ubuntu-20.04.3-desktop-amd64.iso

$ wget -c --limit-rate=100k -b wget.log https://releases.ubuntu.com/20.04.3/ubuntu-20.04.3-desktop-amd64.iso

d).\* Wget Download Password Protected Files via FTP and HTTP

To download a file from a password-protected FTP server, you can use the options --ftp-user=username and --ftp-password=password as shown.

$ wget --ftp-user=narad --ftp-password=password ftp://ftp.example.com/filename.tar.gz

To download a file from a password-protected HTTP server, you can use the options --http-user=username and --http-password=password as shown.

$ wget --http-user=narad --http-password=password http://http.example.com/filename.tar.gz

e)\*. Wget Ignore SSL Certificate Check

To ignore SSL certificate check while downloading files over HTTPS, you can use the --no-check-certificate option:

$ wget --no-check-certificate https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/linux-5.1.1.tar.gz

**5.6 curl**

A curl is a command-line tool that helps transfer the data from the server to the client and vice-versa. It uses the following protocols, i.e., HTTP, SMTP, FTP, and POP3. etc. Curl is known to transfer multiple files at a time.

To save the data in a specific file, pass the '-o' option followed by directory, file name, and URL as follows:

curl -o <directory>/<filename> <URL>

Consider the below command:

curl -o /home/javatpoint/Documents/linux.html https://www.javatpoint.com/linux-tutorial

curl -C <URL>

curl -O <URL1> <URL2>

curl -I www.javatpoint.com

curl --speed-limit 300 --speed-time 10 https://example.com

curl --ssl-allow-beast https://example.com

curl --ssl-auto-client-cert https://example.com

curl --ssl-no-revoke https://example.com

curl --ssl-reqd ftp://example.com

curl --stderr output.txt https://example.com

curl --url https://example.com

**5.7 ss/netstat**

The **ss** command is used to dump socket statistics on Linux systems. It serves as a replacement for the netstat command and is often used for troubleshooting network problems.

what is **socket**?

A **socke**t identifies both the remote host and the port that it connects to so that data can be sent between the systems. Sockets are similar to pipes except that pipes only facilitate connections between processes on the same system where sockets work on the same or different systems. Unlike pipes, sockets also provide bidirectional communication.

**syntax**

**ss <options>**

**Or list the options individually:**

**ss <option 1> <option 2> <option 3>**

examples:

1.List All Connections

List all listening and non-listening connections with:

$ss -a

2.Terminal output of the command ss -a command

List Listening Sockets

To display only listening sockets, which are omitted by default, use:

$ss -l

3.Terminal output of the command ss -l

List TCP Connections

To list TCP connections, add the -t option to the ss command:

$ss -t

4.Terminal output of the command ss -t

List All TCP Connections

Combine the options -a and -t with the ss command to output a list of all the TCP connections:

$ss -at

5.List All Listening TCP Connections

Combine the options -l and -t with the ss command to list all listening TCP connections:

$ss -lt

6.Terminal output of the command

$ss -lt

**5.8 /etc/services**

The /etc/services file is used by applications to translate human readable service names into port numbers when connecting to a machine across a network. The file will typically include the service name, port/protocol, any aliases, and comments.

**5.9 telnet, nc**

Linux, the **telnet** command is used to create a remote connection with a system over a TCP/IP network. It allows us to administrate other systems by the terminal. We can run a program to conduct administration.

Example : telent <des tip/hostname> <port>

$ telnet 38.76.11.19 22

Alternately we can use **nc**

nc -zv kafka02 6667

**other network commands**

nslookup <IP/Hostname>

tracert -d <target IP> (windows)

traceroute <target IP>

ping <target IP>

**5.10 niping (sap specific)**

**6.Environment variable**

**6.1 export/env/setenv**

Linux Set Environment Variable

The environment variables are dynamic values that are stored within a system and used by applications launched in shells or sub-shells. These variables have a name and their respected value. The environment variable customizes the system performance and the behavior of an application.

The environment is the track for a computer application to interact with the system. The environment variable can have information about the default applications of the system, the system locale, the path of the executable file and the keyboard layout setting, and more. The environment variable makes an app available as per the system.

Common Environment Variables

Some standard environment variables are as follows:

PATH

This variable contains a list of directories in which our system looks for files. It separates directories by a (:) colon.

Play Video

USER

This variable holds the username.

HOME

This variable holds the default path to the user's home directory.

EDITOR

This variable contains the path to the specified editor.

UID

This variable contains the path to the user's unique id.

TERM

This variable contains the path to the default terminal emulator.

SHELL

This variable contains the path to the default shell that is being used by the user.

ENV

This variable displays all the environment variable.

How to set Environment Variable in Linux?

There are multiple commands in Linux that allow us to set and create the environment variable.

Use the export command to set a new environment variable.

To create a new variable, use the export command followed by a variable name and its value.

Syntax:

export NAME=VALUE

To create a new variable say new\_variable, execute the command as follows:

export new\_variable=10

The echo command is used to display the variable:

echo $new\_variable

To display the value of the variable, use the $ symbol before the variable name:

echo $new\_variable

Consider the below output:

Linux Set Environment Variable

To set Java Environment Variable, execute the command as follows:

export PATH=$PATH:/home/jdk1.8/bin/

We can also create a user to define a variable by directly declaring it on the terminal.

If we want to create a variable new\_variable2, we can create it as follows:

new\_variable2=100

Consider the below output:

Linux Set Environment Variable

Accessing the value of Environment Variable

To access the value of a variable, execute the echo command as follows:

echo $variable name

Note: The variables are case sensitive; we cannot use any variable name 'new\_variable' as 'New\_variable.'

The env command

The env command is used to display all the available variables in the system.

env

Output:

Removing an Environment Variable

By removing an environment variable we can remove all existing component of particular variable.

To remove an environment variable, execute the unset command followed by variable name:

unset variable name

The above command will delete the specified variable and its components from the system.

To remove a variable new\_variable from the system, execute the command as follows:

unset new\_variable

Consider the below output:

**7. Running scripts or executables in Linux**

In Linux we can execute third party software’s or scripts Linux by placing ./ on command propt

* Open the terminal. Go to the directory where you want to create your script.
* Create a file with **.sh** extension.
* Write the script in the file using an editor.
* Make the script executable with command **chmod +x** <**fileName**>.
* Run the script using ./<**fileName**>.

**Linux PKG installation:**

Rpm (any Linux/unix Mechine)

YUM ((any Linux/unix Mechine except SUSE)

Zypper (Suse Linux)