LINUX COMMANDS

- Prashant Karne

what is Linux?
what is kernel?
what is an operating system? what is shell?
what is command line interface?
what is graphical user interface?

Change Directory Command- Absolute and Relative Paths

cd command ---> To change directory

Absolute Path:

We start the path from the root directory "/". An absolute path is defined as specifying the location of a file or directory from the root directory /. In other words, we can say that an absolute path is a complete path from the start of the actual file system from / directory.

```
[param@mac ~]$ cd /home/param/notes/java
[param@mac java]$
```

Relative Path (Short Path):

We do not start the path from the root directory "/". The relative path is defined as the path related to the present working directory . It starts at your current directory and **never starts with a** '/'.

```
[param@mac notes]$ cd java/
[param@mac java]$
```

`cd.` or **`cd ./`** ---> Current Directory path
`cd ..` or **`cd ../`** ---> To go to previous directory
`cd ../../` ---> To go to previous 2 directories
`cd ../../../` ---> To go to previous 3 directories

Format of Linux Commands:

```
$ Command -Options and Arguments
```

Example:

```
[root@123.293.12.1 ~]# ls -l /var
```

Here,

ls	=	command
-l	=	**option/s**
/var	=	arguments

All about ls Command

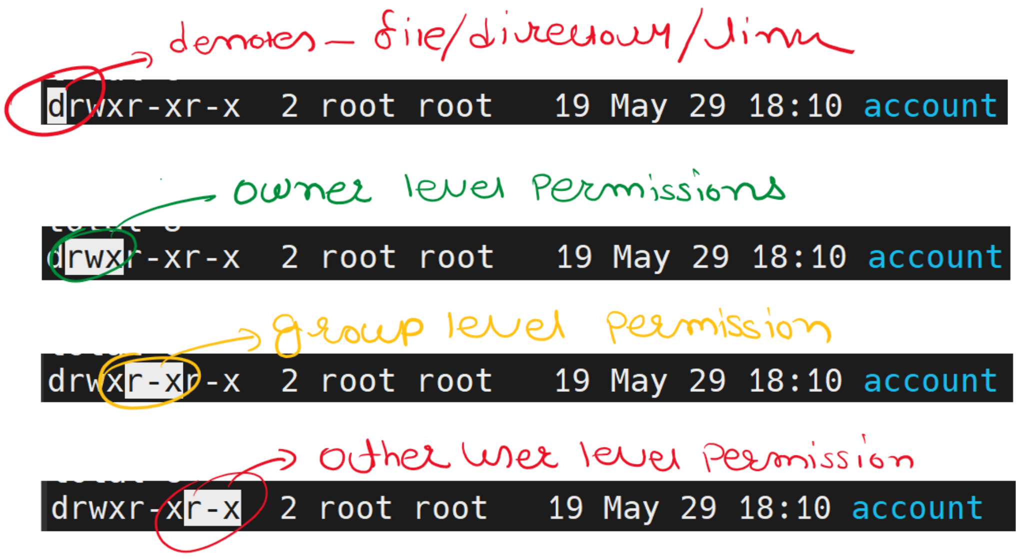
`ls` **command**- is a Linux shell command that lists directory contents of files and directories. It provides valuable information about files, directories, and their attributes.

Syntax:

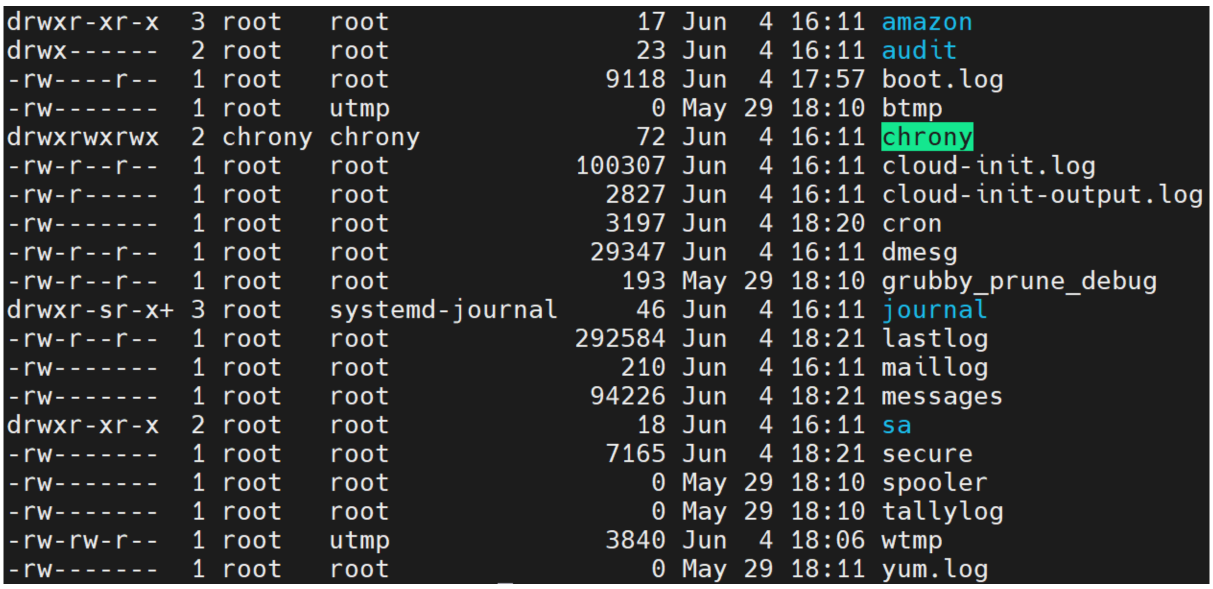
```
ls [option] [file/directory]
```

'ls' will display the contents of the current directory. By default, 'ls' lists files and directories in alphabetical order.

Options	Description
-l	known as a long format that displays detailed information about files and directories.
-a	Represent all files Include hidden files and directories in the listing.
-t	Sort files and directories by their last modification time, displaying the most recently modified ones first.
-r	known as reverse order which is used to reverse the default order of listing.
-h	Print file sizes in human-readable format (e.g., 1K, 234M, 2G).



```
drwxr-xr-x  2 root root    19 May 29 18:10 account
drwxr-xr-x  2 root root    19 May 29 18:10 account
drwxr-xr-x  2 root root    19 May 29 18:10 account
drwxr-xr-x  2 root root    19 May 29 18:10 account
```



```
drwxr-xr-x  3 root   root                17 Jun  4 16:11 amazon
drwx------  2 root   root                23 Jun  4 16:11 audit
-rw----r--  1 root   root              9118 Jun  4 17:57 boot.log
-rw-------  1 root   utmp                 0 May 29 18:10 btmp
drwxrwxrwx  2 chrony chrony              72 Jun  4 16:11 chrony
-rw-r--r--  1 root   root            100307 Jun  4 16:11 cloud-init.log
-rw-r-----  1 root   root              2827 Jun  4 16:11 cloud-init-output.log
-rw-------  1 root   root              3197 Jun  4 18:20 cron
-rw-r--r--  1 root   root             29347 Jun  4 16:11 dmesg
-rw-r--r--  1 root   root               193 May 29 18:10 grubby_prune_debug
drwxr-sr-x+ 3 root   systemd-journal     46 Jun  4 16:11 journal
-rw-r--r--  1 root   root            292584 Jun  4 18:21 lastlog
-rw-------  1 root   root               210 Jun  4 16:11 maillog
-rw-------  1 root   root             94226 Jun  4 18:21 messages
drwxr-xr-x  2 root   root                18 Jun  4 16:11 sa
-rw-------  1 root   root              7165 Jun  4 18:21 secure
-rw-------  1 root   root                 0 May 29 18:10 spooler
-rw-------  1 root   root                 0 May 29 18:10 tallylog
-rw-rw-r--  1 root   utmp              3840 Jun  4 18:06 wtmp
-rw-------  1 root   root                 0 May 29 18:11 yum.log
```

Here,

- **-** : normal file
- **d** : directory
- **l** : link file
- **Field 1** – File Permissions: Next characters specify the files permission. Every 3 characters specify read, write, execute permissions for user(root), group and other users respectively in order. Taking the above example, `-rw-r--r-` indicates read-write permission for user(root), read permission for group, and read permission for others respectively. If all three permissions are given to user(root), group and others, the format looks like `-rwxrwxrwx`
- **Field 2** – Number of links: Second field specifies the number of links for that file. In this example, 1 indicates only one link to this file.
- **Field 3** – Owner: Third field specifies owner of the file. In this example, this file is owned by username 'maverick'.
- **Field 4** – Group: Fourth field specifies the group of the file. In this example, this file belongs to" root's group.
- **Field 5** – Size: Fifth field specifies the size of file in bytes. In above example, see boot.log file indicates the file size in '9118' bytes. You can use -h to get output in human readable form.
- **Field 6** – Last modified date and time: Sixth field specifies the date and time of the last modification of the file. See in above screenshot.
- **Field 7** – File name: The last field is the name of the file.

File Creation and Editors in Linux

touch Command
vi or **vim** editor
nano editor
cat command
echo command

Let's see these commands one by one-

touch command:

You can user touch command to create single or multiple empty files:

Command

```
$ touch <file_name>
$ touch file1 file2 file2
```

vi or vim editor to create or edit existing file:

Its main function is to edit files. The default editor that comes with the Linux/UNIX operating system is called vi (visual editor). Using vi editor, we can edit an existing file or create a new file from scratch. we can also use this editor to just read a text file. The advanced version of the vi editor is the vim editor, its same as vi editor.

Command: to create a new file or edit existing file in vi editor:

```
$ vi filename
```

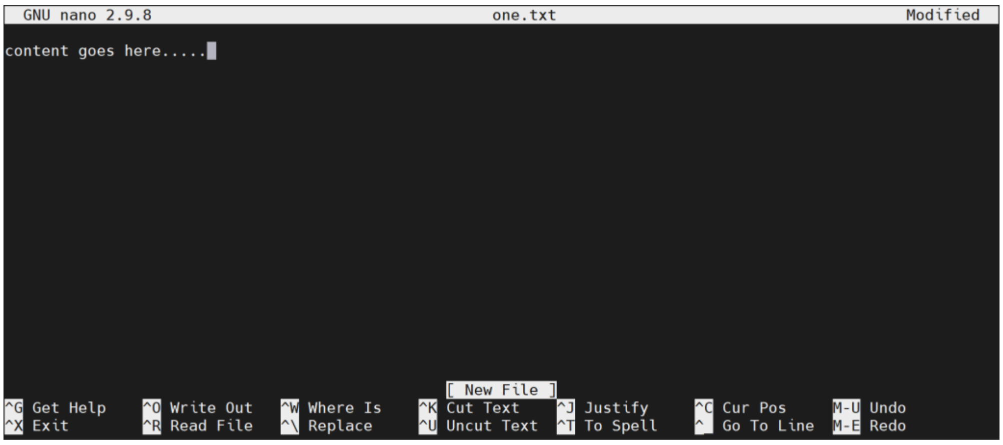
Vi Command Mode: Press ESC to go to command mode and then enter below commands:

	`i`	---->	Insert mode, to edit the file
	`:w`	--->	To write into file (save the file)
	`:q`	--->	To quite (exit) from the file.
	`:wq`	--->	To save and quit/exit the file.
	`:w!`	--->	To forcefully write/save file.
	`:q!`	--->	To quite/exit file forcefully.
	`:wq!`	--->	To save and quite file forcefully.
IMP:	`dd`	--->	To delete the current line in file.
	`cc`	--->	To cut the line
	`p`	--->	To paste the line
	`uu`	--->	Undo changes
	`:r <file.txt>`	--->	Read data from file 'file.txt'
	`/string`	--->	To search particular string in file
IMP:	`:s/string/replace`	--->	To search string and replace in current line only
IMP:	`:%s/string/replace`	--->	To search and replace all occurrences of string in the file.

`nano` Command to create or edit an existing file:

Command:

```
$ nano filename.txt
```



```
  GNU nano 2.9.8                         one.txt                         Modified
content goes here.....





                                       [ New File ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

`cat` command:

used for create file and read file and add content in existing file.

Most universal command/tool for creating files on Linux systems. We cannot edit a file using the cat command. Even though we cannot use cat for file editing we can write content into cat file:

Command:

`cat >> file1.txt`	|	**`cat > file1.txt`**
`Content goes here…`	|	*`content goes here…`*

CTR+C CTRL+C

Press **CTRL+C** or **CTRL+D** to exit out of the command, check the file with cat file1.txt command

IMP: Difference between **>** & **>>**

So, **“>”** is the output redirection operator **used for overwriting files that already exist** in the directory. While, the **“>>”** is an output operator as well, but it **appends the data of an existing file**.

`$ cat file1.txt`	--->	To read the content of the file

Other options with CAT command:

`$ cat file1 file2`	--->	To view the Content of Multiple Files
`$ cat -n file1`	--->	To view content of the file with line number
`$ cat file1.txt file2.txt >> merged_file3.txt`	--->	To copy the content of files into another file.
`$ cat file_name1 >> file_name2`	--->	To append the content of one file into another.
`$ tac file1.txt`	--->	To display the content of the file in reverse order
`$ cat -E file1.txt`	--->	To highlight the end of every line with ($)

---

echo command to create file

echo command is used to print a message but we can use the same command to print the content inside file and if file is not present then echo command will create the file:

Command:

`echo “content goes here” >> file1.txt`	--->	To append content a file
`echo “content goes here” > file1.txt`	--->	To overwrite content in file

File Management in Linux

In Linux, most of the operations are performed on files. And to handle these files Linux has directories also known as folders which are maintained in a tree-like structure.

We have already seen how to list and create file now we will see some of other basic file management commands in Linux, these commands will help you to perform various operations on files.

Copy a File:

cp command: We can use cp command to copy the files between directories. It will create the new file in destination with the same name with same content as that of the file ‘filename’.

Command syntax:

```
$ cp source/filename destination-path/
```

Example:

```
$ cp /var/log/boot.log /mnt
```

So, in above example you can see that we are copying boot.log file from source '/var/log/' to destination '/mnt'

Move a File:

mv command- To move a file from source to destination. It will remove the file from the source folder and would be creating a file with the same name and content in the destination folder.

Command syntax:

```
$ mv source/filename destination/
```

Example: `mv /var/log/boot.log /mnt`

Rename a file:

We can rename the file with mv command. It will rename the filename to new_filename or in other words, it will remove the filename file and would be creating a new file with the new_filename with the same content and name as that of the filename file.

Command syntax:

```
$ mv filename new_filename
```

Example: `mv sys.log boot.log`

Deleting a file:

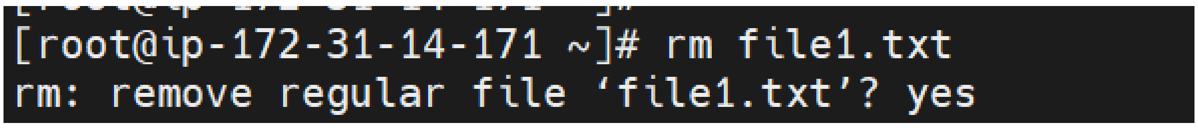
We can use **rm** command to delete files in Linux.

Command syntax:

```
$ rm filename
```

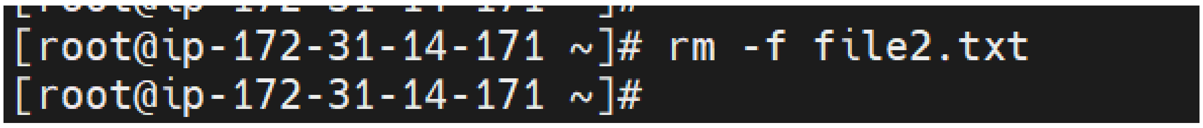
Let's see some examples and options with rm command:

`$ rm filename` ---> to remove a file, this command will prompt for the confirmation



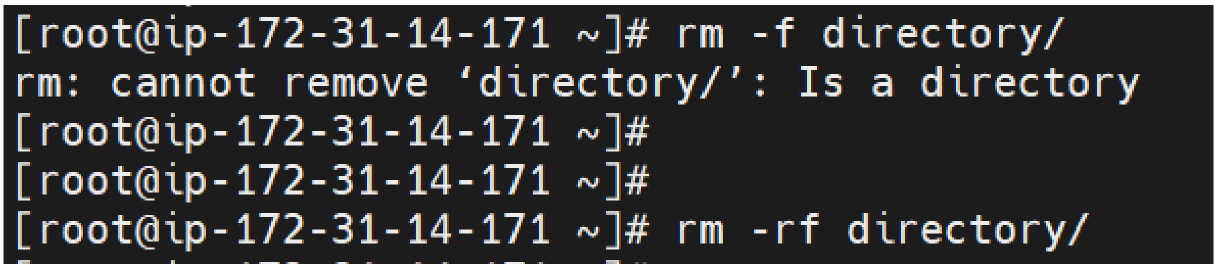
```
[root@ip-172-31-14-171 ~]# rm file1.txt
rm: remove regular file 'file1.txt'? yes
```

`$ rm -f filename` ---> to forcefully delete the file, no confirmation is required.



```
[root@ip-172-31-14-171 ~]# rm -f file2.txt
[root@ip-172-31-14-171 ~]#
```

`$ rm -rf directory` ---> to remove the directory (r = recursively)



```
[root@ip-172-31-14-171 ~]# rm -f directory/
rm: cannot remove 'directory/': Is a directory
[root@ip-172-31-14-171 ~]#
[root@ip-172-31-14-171 ~]#
[root@ip-172-31-14-171 ~]# rm -rf directory/
```

Reading a file:

We have already seen cat command to display the content of the file, lets see another way how you can read the file with the help of less command. Less command is a Linux utility that can be used to read the contents of a text file one page (one screen) at a time. It has faster access because if a file is large, it doesn't access the complete file, but accesses it page by page.
For example, if it's a large file and you are reading it using any text editor, then the complete file will be loaded to the main memory. The less command doesn't load the entire file but loads it part by part which makes it faster.

Command syntax:

```
$ less filename
```

Press 'q' to exit out to terminal.

Directory Management

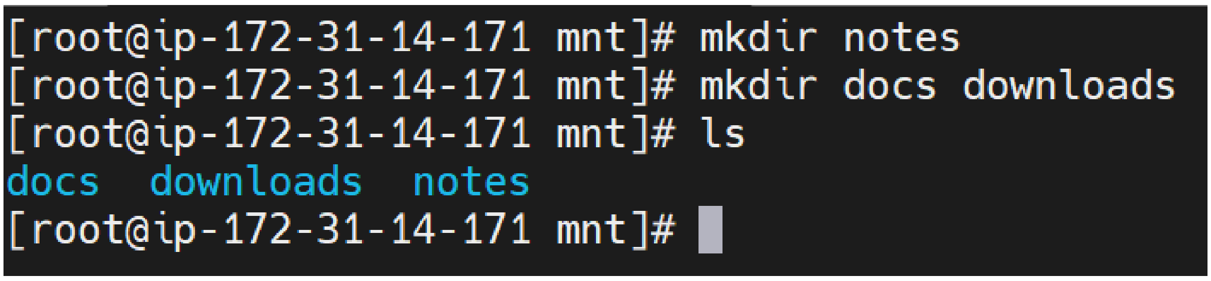
mkdir command: To create a new directory in Linux.

Command syntax:

```
$ mkdir directory_name
```

Example:

To create multiple directories in one command, leave space between directory names.



```
[root@ip-172-31-14-171 mnt]# mkdir notes
[root@ip-172-31-14-171 mnt]# mkdir docs downloads
[root@ip-172-31-14-171 mnt]# ls
docs  downloads  notes
[root@ip-172-31-14-171 mnt]# 
```

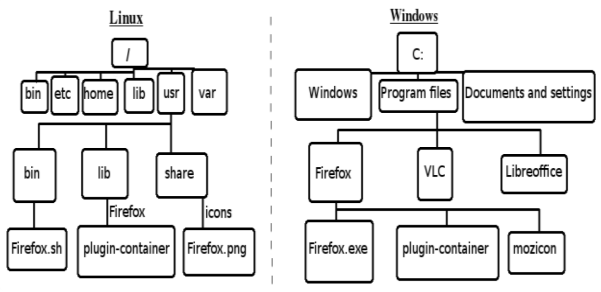
Command to create directory under directory:

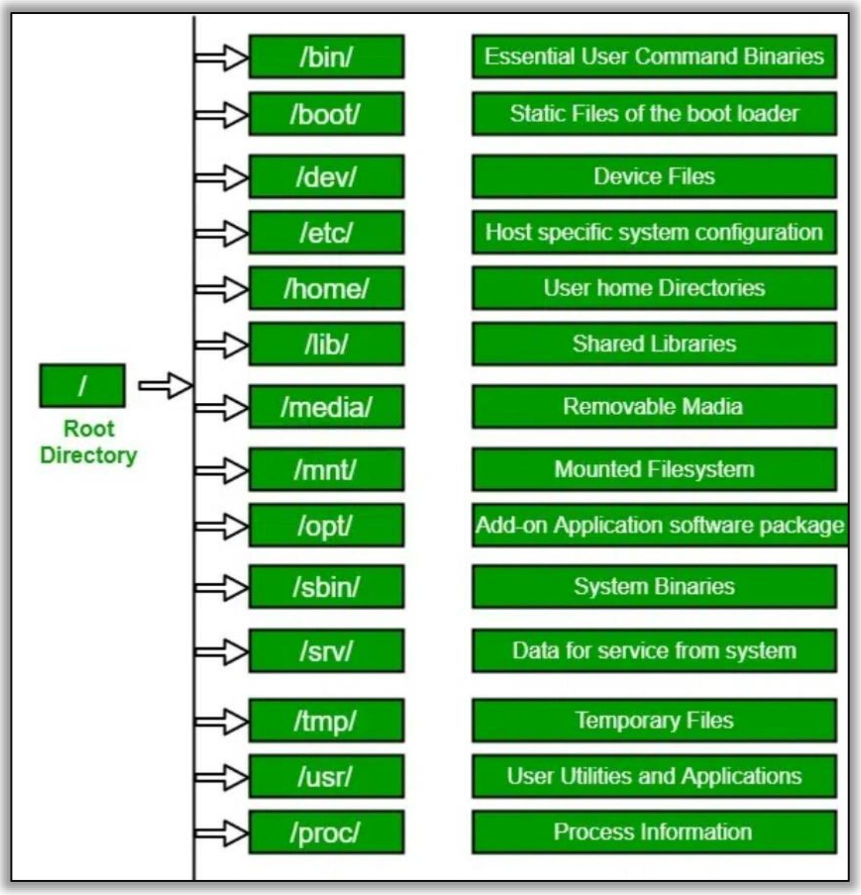
```
$ mkdir -p one/two/three/four
```

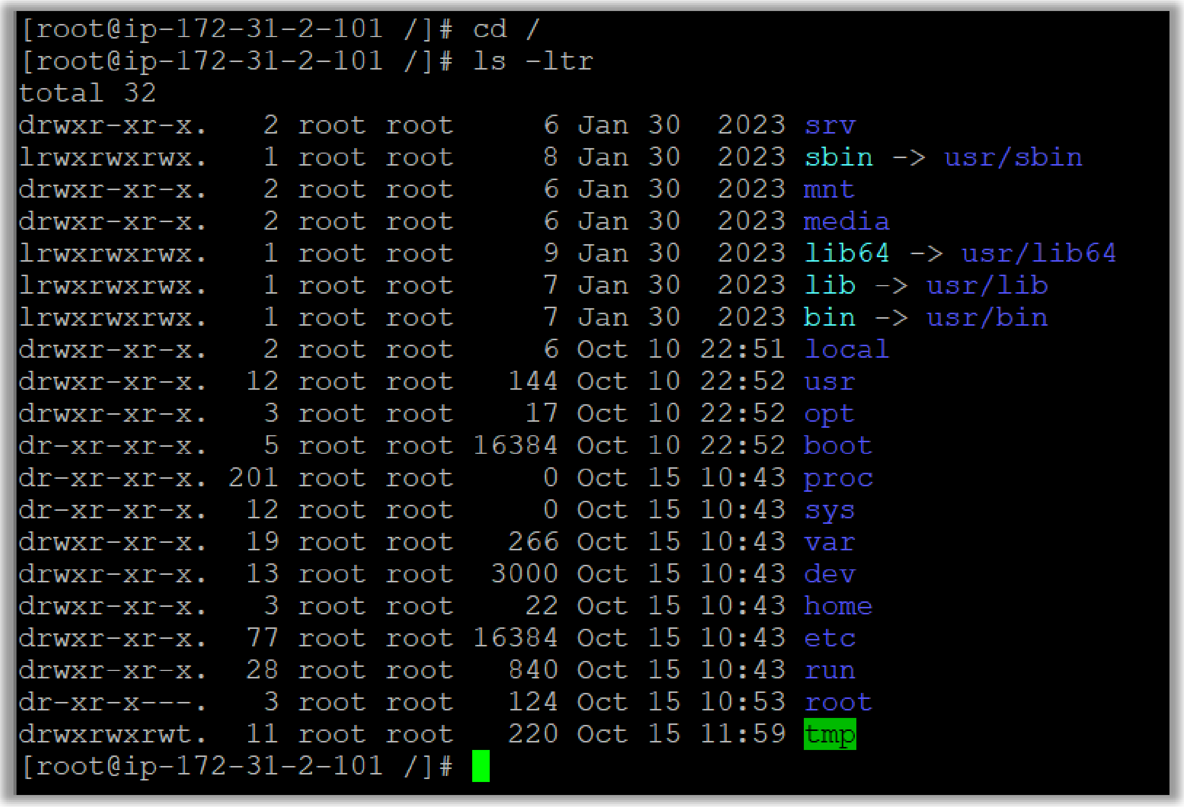
-p - parent directory

Like files, we can use rm and mv commands to remove directory and rename or move directory.

Directory Structure in Linux







```
[root@ip-172-31-2-101 /]# cd /
[root@ip-172-31-2-101 /]# ls -ltr
total 32
drwxr-xr-x.   2 root root     6 Jan 30  2023 srv
lrwxrwxrwx.   1 root root     8 Jan 30  2023 sbin -> usr/sbin
drwxr-xr-x.   2 root root     6 Jan 30  2023 mnt
drwxr-xr-x.   2 root root     6 Jan 30  2023 media
lrwxrwxrwx.   1 root root     9 Jan 30  2023 lib64 -> usr/lib64
lrwxrwxrwx.   1 root root     7 Jan 30  2023 lib -> usr/lib
lrwxrwxrwx.   1 root root     7 Jan 30  2023 bin -> usr/bin
drwxr-xr-x.   2 root root     6 Oct 10 22:51 local
drwxr-xr-x.  12 root root   144 Oct 10 22:52 usr
drwxr-xr-x.   3 root root    17 Oct 10 22:52 opt
dr-xr-xr-x.   5 root root 16384 Oct 10 22:52 boot
dr-xr-xr-x. 201 root root     0 Oct 15 10:43 proc
dr-xr-xr-x.  12 root root     0 Oct 15 10:43 sys
drwxr-xr-x.  19 root root   266 Oct 15 10:43 var
drwxr-xr-x.  13 root root  3000 Oct 15 10:43 dev
drwxr-xr-x.   3 root root    22 Oct 15 10:43 home
drwxr-xr-x.  77 root root 16384 Oct 15 10:43 etc
drwxr-xr-x.  28 root root   840 Oct 15 10:43 run
dr-xr-x---.   3 root root   124 Oct 15 10:53 root
drwxrwxrwt.  11 root root   220 Oct 15 11:59 tmp
[root@ip-172-31-2-101 /]# 
```

Directory	Description
/	This is the entry point of all directories and is described as a forward slash, which is actually the home of the Operating System. Everything is in it. Not every user has read and write privileges to this directory; only the administrators or allowed users of the operating systems can have access to such privileges.
/bin	**Contains commands used by all the users**, this is the directory that has all the binary files of some important programs on the operating system. This directory also holds the data about the most used commands related to making (mkdir), moving (mv), copying (cp), listing (ls), and removing (rm) a directory or file. According to the Linux File System Standards, this directory cannot have subdirectories.
/boot	**Contains bootable file for the Linux,** This is the directory that handles the ignition of the Linux Operating System. First of all, you don't need to modify anything in this directory, otherwise, you can't alter anything in it unless you have administrator's rights. You should stay away from doing anything in this directory, or else it will be a huge mess to set it up again.
/dev	Essential device files like Terminal device, USB Device or a Hard Drive or any other device that is attached to the system.
/etc	This may seem a little bit funny to you, but **this directory is for those types of configuration files and folders in which the system does not know where to put them**. So, it is an "etcetera" directory for the Linux Operating system. This directory mostly contains the static program local files that affect all users. Since this directory mostly contains files related to the configuration, it is better to call it "Everything to Configure".
/home	**Home Directory for other users, this is the directory where most of the user's personal data is placed**. A user spends most of his time here because - Downloads, Documents, Desktop, and all other basic required and much-known directories are in this **"/home"** directory. All the dot configuration files of a user are also in here.
/lib	These are the folders where libraries are stored. Libraries are some files that are needed by any application to perform several tasks or functions. For example, these libraries may be needed by the binary files in the /bin directory.
/media	This is the directory where all the external connected storage devices are mounted automatically. We do not need to do anything in this directory because it is managed by the Operating System itself, but if we want to mount storage devices manually, we have the /mnt directory for that purpose.
/mnt	This is the directory where you can find the other mounted drives. For example, a USB drive, an External Hard Drive, or a Floppy Disk Drive. This is **not used nowadays because the devices are automatically mounted to the /media directory,** but **this is where we can mount our storage devices manually.**
/opt	This is the optional folder. It is the directory where manually installed software by vendors is placed.
/proc	This is the directory with the **pseudo files**. The **pseudo files contain information about the processes**.
/root	Just like /home directory, /root is the house of the Administrator a.k.a. super user. Since this is the superuser's directory, it is better not to touch it unless you have complete knowledge of what you are doing.
/run	This directory is used **to store temporary data of processes running on the Operating System**.
/sbin	It contains commands only used by root user.
/snap	The is the directory with the snap packages stored in it.
/srv	This directory **stores the data of the services running on the system**. For example, it holds the data if a server is running on the Operating System.

/sys	This directory is always created during boot time, so it is a virtual directory like /dev, and it is the directory when you want to communicate to the Kernel. It also holds information related to the connected devices.
/tmp	This is a temporary directory and holds the temporary files of the applications running on the system.
/usr	This directory contains the applications installed and used by the user. It is also known as the "UNIX System Resources". It also has its own /bin, /sbin, and /lib directory, which is different from the superuser's /bin, /sbin, and /lib directories.
/var	This is a variable directory that contains the files and folders whose size is expected to increase with the time and the system's usage. Mainly for Logs directory.

Users and groups in Linux System Administration

Users:

Users are accounts that can be used to login into a system.
Each user is identified by a unique identification number or **UID** by the system.
All the information of users in a system are stored in **/etc/passwd** file.
The hashed passwords for users are stored in **/etc/shadow** file.

Users can be divided into two categories on the basis of the level of access:

1. **Superuser/root/administrator**	:	Access to all the files on the system.
2. **Normal users**	:	Limited access.

When a new user is created, by default system takes following actions:

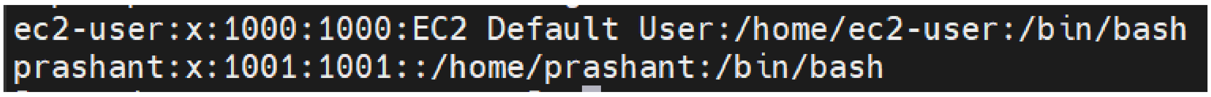
- ✓ Assigns UID to the user.
- ✓ Creates a home directory /home/.
- ✓ Sets the default shell of the user to be /bin/sh.
- ✓ Creates a private user group, named after the username itself.
- ✓ Contents of /etc/skel are copied to the home directory of the new user.
- ✓ .bashrc, .bash_profile and .bash_logout are copied to the home directory of new user.
- ✓ These files provide environment variables for this user's session.

Description of contents of /etc/passwd File
This file is readable by any user but only root as read and write permissions for it. This file consists of the following colon separated information about users in a system:

1. Username field
2. Password field
3. An `x` in this field denotes that the encrypted password is stored in the /etc/shadow file.
4. The user ID number (UID)
5. User's group ID number (GID)
6. Additional information field such as the full name of the user or comment (GECOS)
7. Absolute path of user's home directory
8. Login shell of the user

```
[username]:[password]:[UID]:[GID]:[GECOS]:[home_dir]:[shell_path]
```

Example:



```
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
prashant:x:1001:1001::/home/prashant:/bin/bash
```

Commands:

`users` ---> To print current logged in user on system
`useradd username` ---> To add a new user account
`add user username` ---> This command will call useradd program only ***IMP**
`passwd username` ---> To change the password of the user
`userdel username` ---> To delete the user
`userdel -f username` ---> **-f** forcefully, delete the user
`userdel -r username` ---> **-r** Files in the user's home directory will be removed along with the home directory itself
`cat /etc/passwd` ---> to check all the users in machine

Groups

Each group in a Linux system is uniquely identified by *a group identification number* or *GID*.
All the information listing groups in a system are stored in **/etc/group** file.
The hashed passwords for groups are stored in **/etc/gshadow** file.
Every user has a primary user group and zero or more supplementary groups.

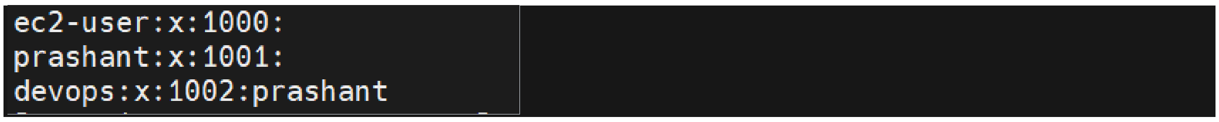
Description of contents of /etc/group **File:**
This file is readable by any user but only root as read and write permissions for it.
This file consists of the following colon separated information about groups in a system:

1. Group name field
2. Password field (If this field is empty, no password is needed.)
3. Group Identification number or GID
4. Comma separated list of usernames of users that belong to the group.

Syntax:

```
[group_name]:[group_password]:[GID]:[users]
```

Example:



```
ec2-user:x:1000:
prashant:x:1001:
devops:x:1002:prashant
```

`groups` ---> To check user is part of which group
`groupadd groupname` ---> To create new group
`usermod -aG groupname username` ---> To add user into a group (-a → append)
`groupdel groupname` ---> To delete the group
`cat /etc/group` ---> To check all the groups in machine
`deluser <username> <groupname>` ---> To remove user from group

vi /etc/group ---> Edit the /etc/group file to remove user from group

chown <username> <filename> ---> To change the ownership of the file to user

chown :<groupname> <filename> ---> change ownership of file to group

More Linux Commands:

$ uname ---> To print system information

- $ uname -s ---> To print the kernel name
- $ uname -n ---> node name, to print the network node hostname
- $ uname -r ---> kernel-release, to print the kernel release
- $ uname -v ---> kernel-version, to print the kernel version

$ man ---> Manual for the command

man [command] ---> To display the manual/description of the command.

$ whoami ---> Print the user's name associated with the current effective user ID.

or

$ id -un ---> To see who you are logged in as, same as whoami

$ hostname ---> To display the system's DNS name or hostname

$ hostname -i ---> To Display all local IP addresses of the host

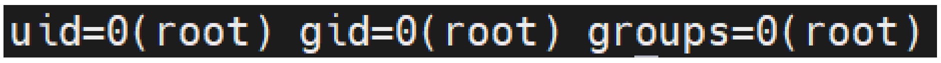
$ sudo ---> To allow a permitted users to execute a command as the superuser

$ uptime ---> To show how long the system has been running

$ date ---> To show the current date and time

$ cal ---> to show this month's calendar

$ id <username/groupname/id> ---> To check the details of user/group/id



```
uid=0(root) gid=0(root) groups=0(root)
```

Here,

- gid ---> Real group
- groups ---> Effective group (user can be part of multiple groups)

$ id -u ---> To print ID of user

$ id -g ---> To print ID of group

$ id -G ---> To print ID of effective group

$ groups ---> It will print groups related to current user

$ su <username> ---> switch user

$ exit ---> to exit out from current user

$ echo *"message"* ---> To print an output wherever we want

wc command – word count:

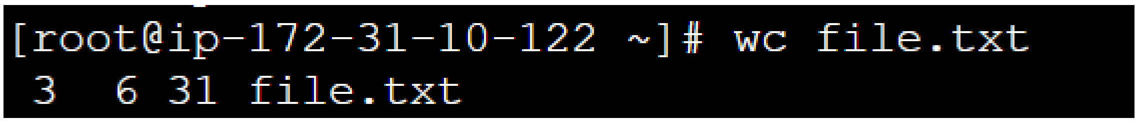
as name suggest we can use WC command for word count. Also helps to count number of lines and size in bytes.

Syntax:

```
$wc filename
```

the command gives output in

eg:



```
[root@ip-172-31-10-122 ~]# wc file.txt
 3  6 31 file.txt
```

Here,
First value indicates number of lines
Second indicate number of words in the file
And third value indicates size in bytes.

Head command- Used t print top N amount of data from the file. by default, head command prints the first 10 lines of the file but we can use -n option to print number of lines from the file.

Command:

```
$ head filename
or
$ head -n 5 filename1 filename2
```

The first command will display first 10 lines whereas second commands will display 5 lines from two files, you can display one or multiple files as well.

Tail command – like head command, tail will also display the content of the file but shows last lines of the file:

Command:

```
$ tail filename
or
$ tail -n 5 filename1 filename2
```

The first command will display last 10 lines whereas second commands will display last 5 lines from two files, you can display one or multiple files as well.

---

grep command:

Used to search pattern in a file or in files. `grep` is widely used by programmers, system administrators, and users alike for its efficiency and versatility in handling text data.

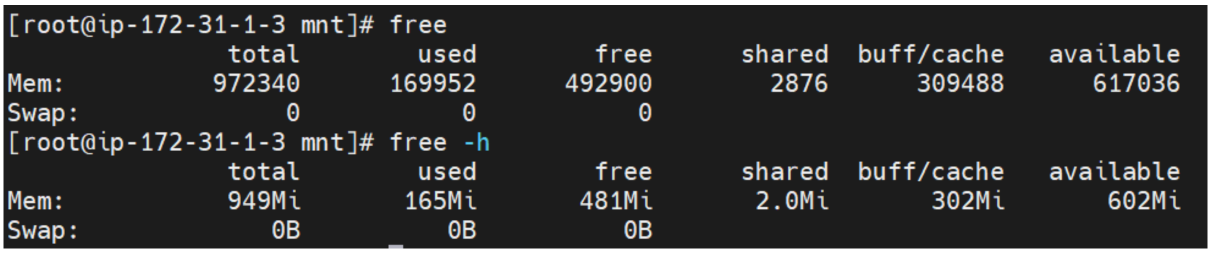
Command:

```
$ grep pattern filename
$ grep -i pattern filename
```

Option **`'-i'`** is used to search pattern regardless of case (case insensitive)

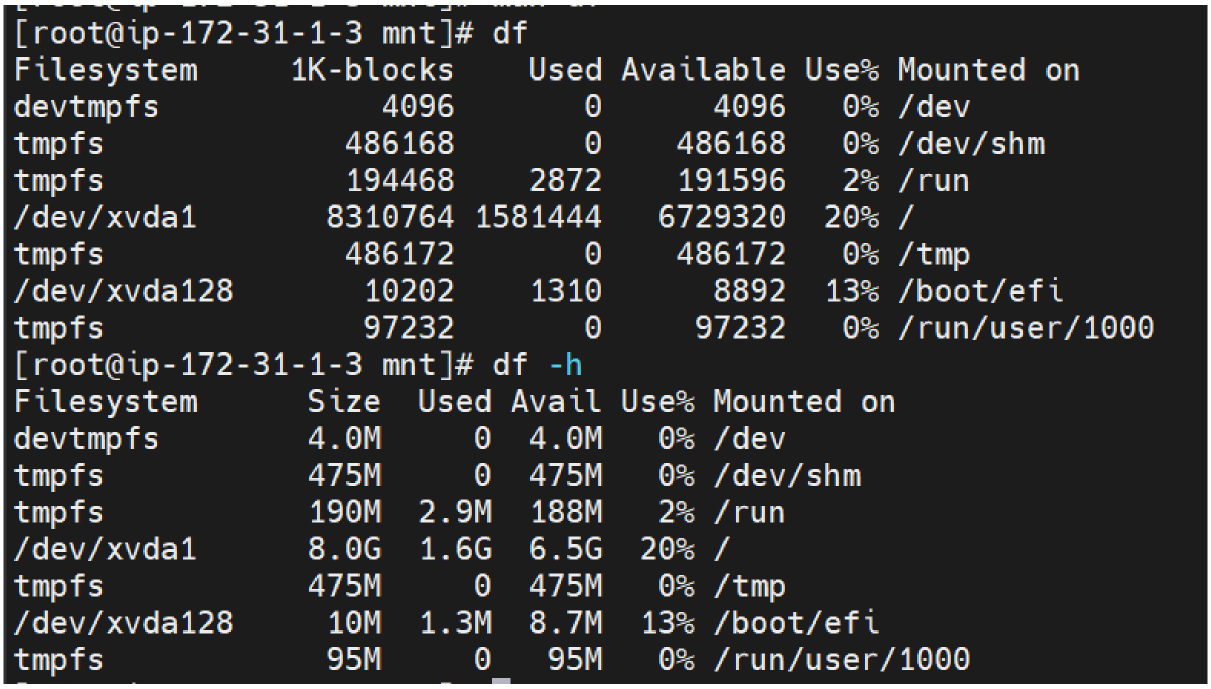
Storage/Volume Related commands:

`free` command: Display amount of free and used **memory (RAM)** in the system



```
[root@ip-172-31-1-3 mnt]# free
              total        used        free      shared  buff/cache   available
Mem:         972340      169952      492900        2876      309488      617036
Swap:             0           0           0
[root@ip-172-31-1-3 mnt]# free -h
              total        used        free      shared  buff/cache   available
Mem:          949Mi       165Mi       481Mi       2.0Mi       302Mi       602Mi
Swap:            0B          0B          0B
```

`df` command: Disk Free command is used to show the **disk usage (HDD)** & information.



```
[root@ip-172-31-1-3 mnt]# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs            4096       0      4096   0% /dev
tmpfs             486168       0    486168   0% /dev/shm
tmpfs             194468    2872    191596   2% /run
/dev/xvda1       8310764 1581444   6729320  20% /
tmpfs             486172       0    486172   0% /tmp
/dev/xvda128       10202    1310      8892  13% /boot/efi
tmpfs              97232       0     97232   0% /run/user/1000
[root@ip-172-31-1-3 mnt]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.0M     0  4.0M   0% /dev
tmpfs           475M     0  475M   0% /dev/shm
tmpfs           190M  2.9M  188M   2% /run
/dev/xvda1      8.0G  1.6G  6.5G  20% /
tmpfs           475M     0  475M   0% /tmp
/dev/xvda128     10M  1.3M  8.7M  13% /boot/efi
tmpfs            95M     0   95M   0% /run/user/1000
```

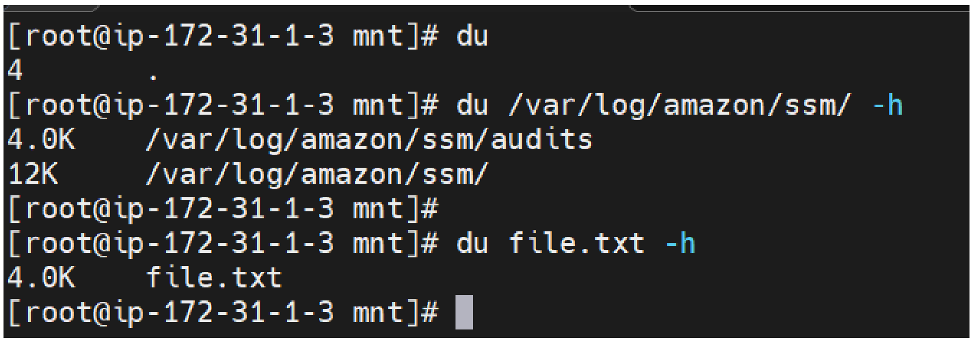
`du` Command:

To retrieve the information about disk space usage information for a specified directory.

Command:

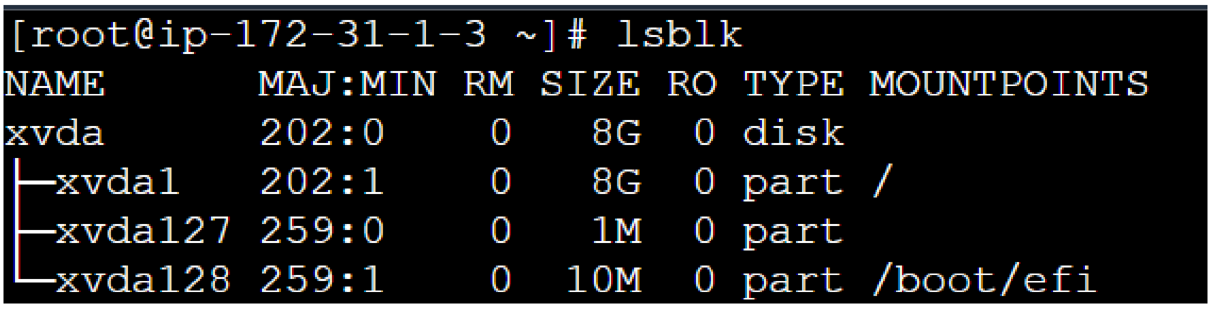
```
du    --->  To retrieve disk utilization of current directory
du /file or directory path    --->  To retrieve du of specific directory or file
```

User option -h to get values in human readable format.



```
[root@ip-172-31-1-3 mnt]# du
4       .
[root@ip-172-31-1-3 mnt]# du /var/log/amazon/ssm/ -h
4.0K    /var/log/amazon/ssm/audits
12K     /var/log/amazon/ssm/
[root@ip-172-31-1-3 mnt]#
[root@ip-172-31-1-3 mnt]# du file.txt -h
4.0K    file.txt
[root@ip-172-31-1-3 mnt]#
```

`lsblk` **command**: to list all block devices connected to Linux machine, it also provides detailed information about block devices such as hard drives, solid-state drives, and other storage-related devices.



```
[root@ip-172-31-1-3 ~]# lsblk
NAME         MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
xvda         202:0    0   8G  0 disk
├─xvda1      202:1    0   8G  0 part /
├─xvda127    259:0    0   1M  0 part
└─xvda128    259:1    0  10M  0 part /boot/efi
```

Linux package management:

yum (Yellowdog Updater, Modified)

YUM is a powerful package management tool that simplifies the process of installing, updating, and managing software on Red Hat-based Linux distributions. It is a command-line package-management utility for RPM-compatible Linux operating systems.

Command:

```
$ yum command package/packages
```

commands can be- `install, remove, update, upgrade`

systemctl & service Command

With this tool, you can start, stop, restart, try-restart, reload services or get the current status of the service:

```
$ systemctl status service_name      $ service service_name status
$ systemctl start service_name       $ service service_name start
$ systemctl stop service_name        $ service service_name stop
$ systemctl restart service_name     $ service service_name restart
$ systemctl enable service_name                     -
```

Compress & Decompress commands in Linux

`gzip` command –
gzip, short for GNU Zip, is a command-line compression tool commonly found on Linux systems. The gzip command in Linux/Unix is used to compress/decompress data. To compress a file. Reduce the size of the file by applying compression. Each single file is compressed into a single file. The compressed file consists of a GNU zip header and deflated data. gzip compresses the file, adds a ".gz" suffix, and deletes the original file.

Command:

```
$ gzip file_name              --->   to compress
$ gzip -d filename            --->   to uncompress
$ gzip file1 file2 file3      --->   Compress multiple files into multiple archives
$ gzip -d file1 file2 file3 ---> decompress multiple files into multiple archives
$ gzip -r directory_name ---> Compress multiple files under a directory in one single command
$ gzip -dr directory_name ---> decompress multiple files under a directory from one single archive
```

`tar` command-
The Linux 'tar' stands for tape archive, which is used to create Archive and extract the Archive files. tar command in Linux is one of the important commands that provides archiving functionality in Linux. We can use the Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

Command:

```
$ tar cvf file.tar filenames
```

This command will create an uncompressed tar Archive using option -cvf,
Here,

c	-->	Create
v	-->	View output
f	-->	filename of the archive with .tar extension

```
$ tar xvf file.tar
```

This command will extract files from .tar package

x	-->	Extracts files from an archive.

```
$ tar cvzf file.tar filenames
```

To compress the files with gzip and create archive,
Here,

c	-->	Create
v	-->	View output
z	-->	Uses gzip compression.
f	-->	filename of the archive with .tar extension

```
$ tar xvzf file.tar
```

To extract files from the archive

`zip/unzip` command:

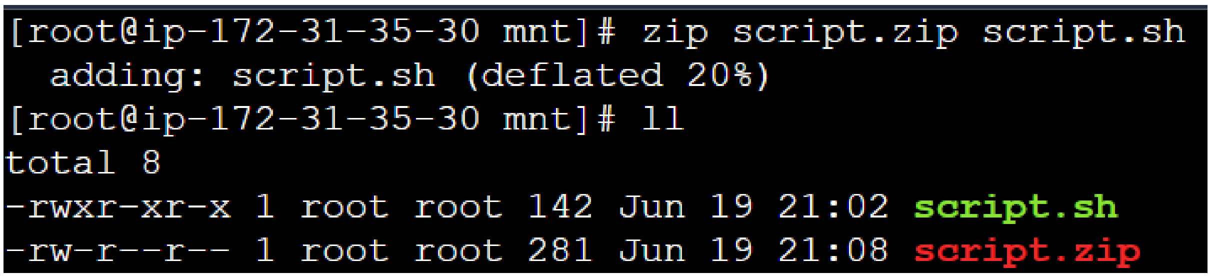
ZIP is a compression and file packaging utility for Linux Each file is stored in a single .zip {.zip-filename} file with the extension .zip.

- Zip is used to compress files to reduce file size and is also used as a file package utility. Zip is available in many operating systems like Unix, Linux, windows, etc.
- If you have limited bandwidth between two servers and want to transfer the files faster, then zip the files and transfer them.
- The zip program puts one or more compressed files into a single zip archive, along with information about the files (name, path, date, time of last modification, protection, and check information to verify file integrity). An entire directory structure can be packed into a zip archive with a single command.

Command:

```
$ zip file_name.zip file_name
```

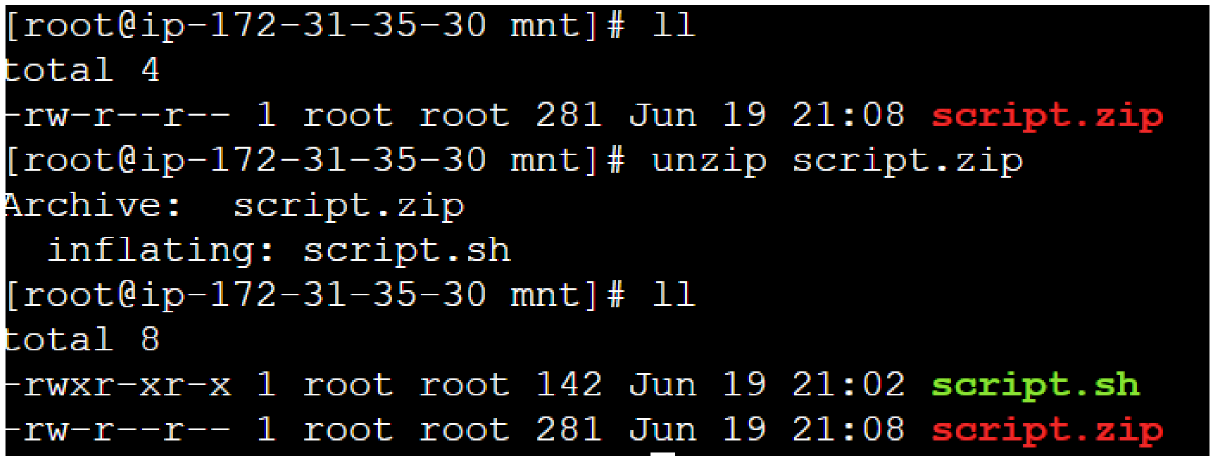
This command will compress and archive the file into a single .zip package-



```
[root@ip-172-31-35-30 mnt]# zip script.zip script.sh
  adding: script.sh (deflated 20%)
[root@ip-172-31-35-30 mnt]# ll
total 8
-rwxr-xr-x 1 root root 142 Jun 19 21:02 script.sh
-rw-r--r-- 1 root root 281 Jun 19 21:08 script.zip
```

```
$ unzip file_name.zip
```

To extract file from .zip



```
[root@ip-172-31-35-30 mnt]# ll
total 4
-rw-r--r-- 1 root root 281 Jun 19 21:08 script.zip
[root@ip-172-31-35-30 mnt]# unzip script.zip
Archive:  script.zip
  inflating: script.sh
[root@ip-172-31-35-30 mnt]# ll
total 8
-rwxr-xr-x 1 root root 142 Jun 19 21:02 script.sh
-rw-r--r-- 1 root root 281 Jun 19 21:08 script.zip
```

`find` command:

The find command in Linux is a dynamic utility designed for comprehensive file and directory searches within a hierarchical structure. Its adaptability allows users to **search by name, size, modification time, or content**, providing a flexible and potent solution. As a pivotal component of the Linux command-line toolkit, the find command caters to the nuanced needs of users, ensuring precision in file exploration and retrieval. Discover the diverse functionalities of the find command and enhance your file management efficiency on the Linux platform.

Syntax of command:

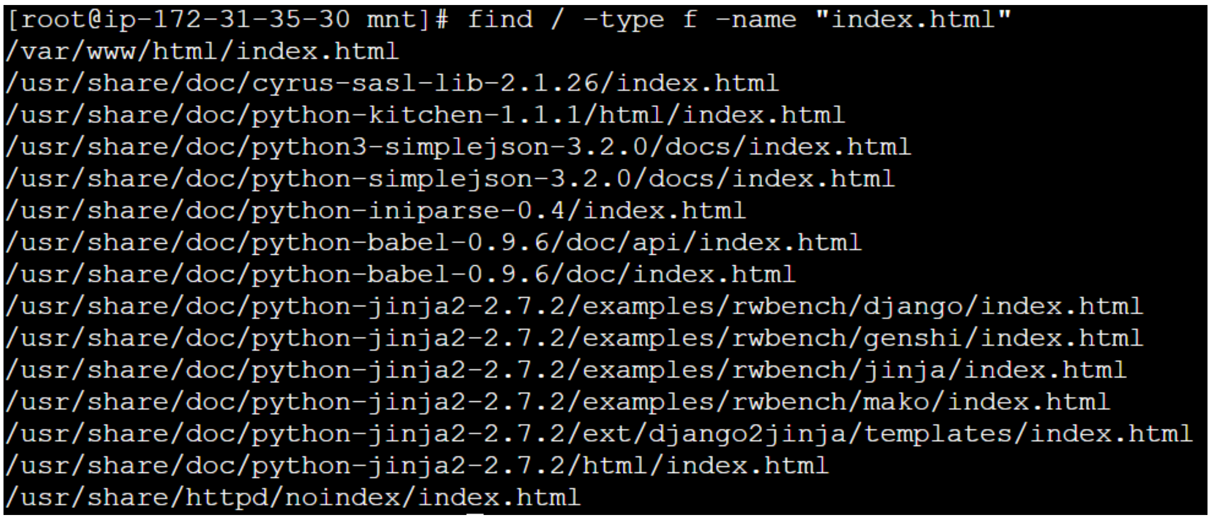
```
$ find [path] [options] [expression]
```
here,

- `path`: Starting directory for the search.
    - Example: find /path/to/search
- `options`: Additional settings or conditions for the search.
    - Example: find /path/to/search -type f -name "*.txt"
- `expression`: Criteria for filtering and locating files.
    - Example: find /path/to/search -type d -name "docs"

Command:
```
$ find /path/to/search -options criteria
```
Replace “/path/to/search" with the directory where you want to start the search and customize the options and criteria based on your requirements.

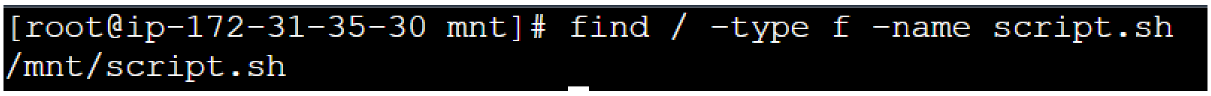
Example:

In below example we tried to find ‘index.html’ file type (-type f) under root directory (/) and you can see all the result where we find index.html files.



```
[root@ip-172-31-35-30 mnt]# find / -type f -name "index.html"
/var/www/html/index.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/index.html
/usr/share/doc/python-kitchen-1.1.1/html/index.html
/usr/share/doc/python3-simplejson-3.2.0/docs/index.html
/usr/share/doc/python-simplejson-3.2.0/docs/index.html
/usr/share/doc/python-iniparse-0.4/index.html
/usr/share/doc/python-babel-0.9.6/doc/api/index.html
/usr/share/doc/python-babel-0.9.6/doc/index.html
/usr/share/doc/python-jinja2-2.7.2/examples/rwbench/django/index.html
/usr/share/doc/python-jinja2-2.7.2/examples/rwbench/genshi/index.html
/usr/share/doc/python-jinja2-2.7.2/examples/rwbench/jinja/index.html
/usr/share/doc/python-jinja2-2.7.2/examples/rwbench/mako/index.html
/usr/share/doc/python-jinja2-2.7.2/ext/django2jinja/templates/index.html
/usr/share/doc/python-jinja2-2.7.2/html/index.html
/usr/share/httpd/noindex/index.html
```

Let’s see another example, here we have script.sh file under /mnt directory:



```
[root@ip-172-31-35-30 mnt]# find / -type f -name script.sh
/mnt/script.sh
```

Processes Related Commands

In Linux, a process is a running instance of a program. When you execute a program, it becomes a process, an independent, executing entity with its own memory space. Each process is assigned a unique identifier, the Process ID (PID). Processes are fundamental to the functioning of the operating system and play a crucial role in multitasking, allowing the computer to execute multiple tasks concurrently.

We can use multiple commands to list the running processes in Linux like **ps, top, htop,** and commands in Linux. We can also have a combination of commands to list the running processes in Linux.

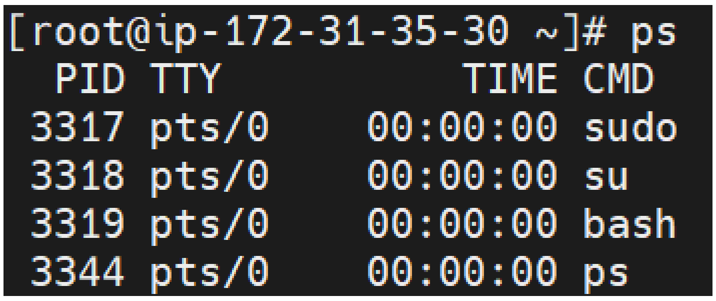
Let’s see commands one by one.

`ps` command:

The ps command, which stands for "**process status**," is like a computer tool that helps you see what's happening inside your Linux.

Command:

```
$ ps
```



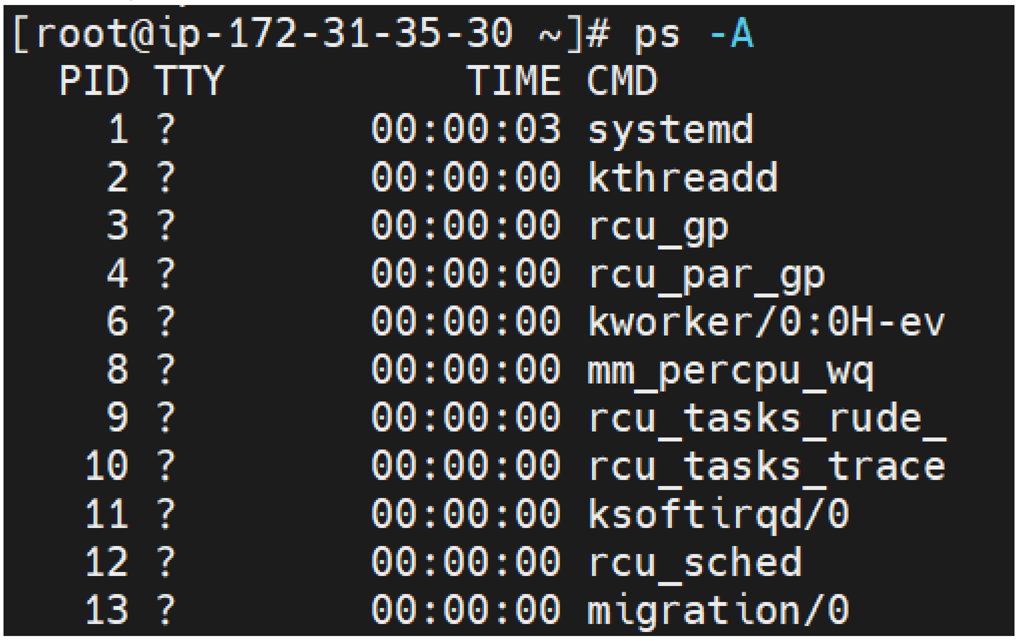
```
[root@ip-172-31-35-30 ~]# ps
  PID TTY          TIME CMD
 3317 pts/0    00:00:00 sudo
 3318 pts/0    00:00:00 su
 3319 pts/0    00:00:00 bash
 3344 pts/0    00:00:00 ps
```

Here,

PID – the unique process ID
TTY – terminal type that the user is logged into
TIME – amount of CPU in minutes and seconds that the process has been running
CMD – name of the command that launched the process.

```
$ ps -A
```

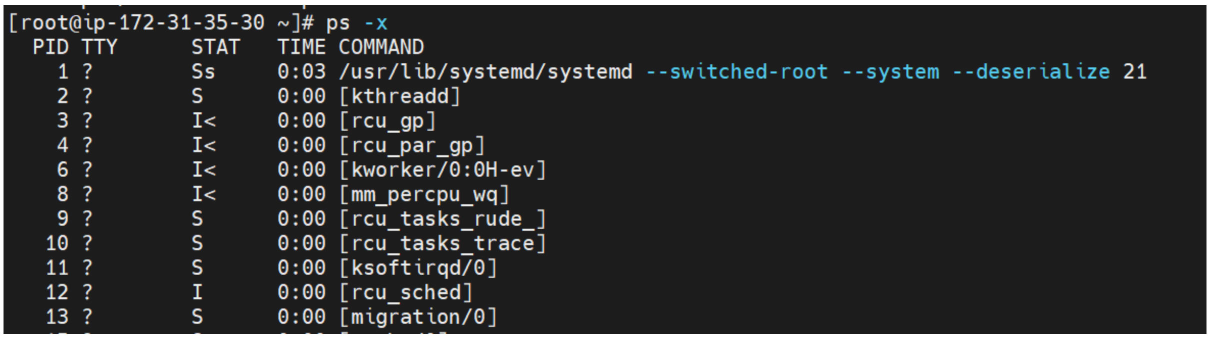
To view All Running Processes in Linux



```
[root@ip-172-31-35-30 ~]# ps -A
  PID TTY          TIME CMD
    1 ?        00:00:03 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 rcu_gp
    4 ?        00:00:00 rcu_par_gp
    6 ?        00:00:00 kworker/0:0H-ev
    8 ?        00:00:00 mm_percpu_wq
    9 ?        00:00:00 rcu_tasks_rude_
   10 ?        00:00:00 rcu_tasks_trace
   11 ?        00:00:00 ksoftirqd/0
   12 ?        00:00:00 rcu_sched
   13 ?        00:00:00 migration/0
```

```
$ ps -x
```

To view processes owned by you



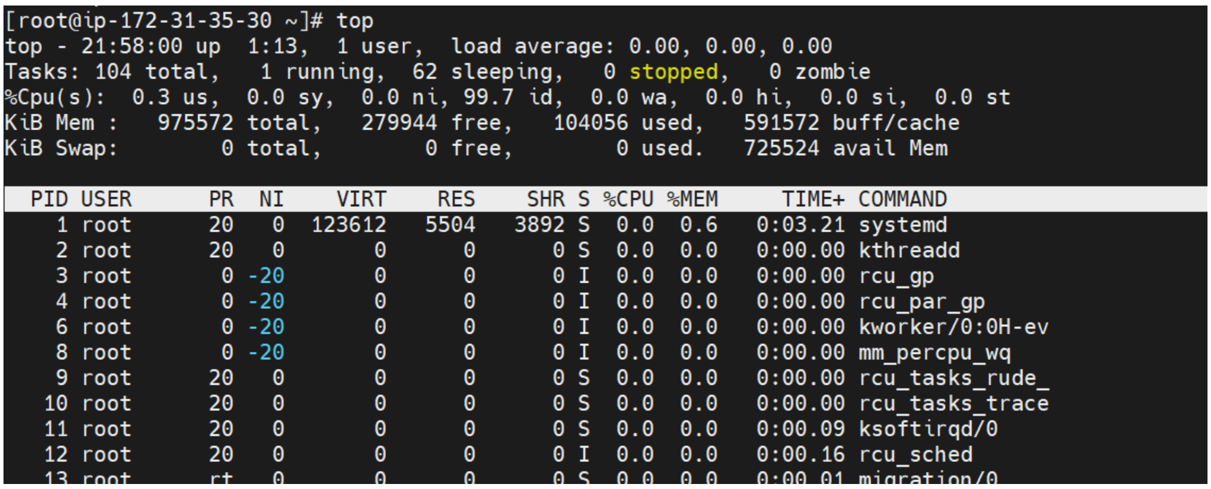
```
[root@ip-172-31-35-30 ~]# ps -x
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:03 /usr/lib/systemd/systemd --switched-root --system --deserialize 21
    2 ?        S      0:00 [kthreadd]
    3 ?        I<     0:00 [rcu_gp]
    4 ?        I<     0:00 [rcu_par_gp]
    6 ?        I<     0:00 [kworker/0:0H-ev]
    8 ?        I<     0:00 [mm_percpu_wq]
    9 ?        S      0:00 [rcu_tasks_rude_]
   10 ?        S      0:00 [rcu_tasks_trace]
   11 ?        S      0:00 [ksoftirqd/0]
   12 ?        I      0:00 [rcu_sched]
   13 ?        S      0:00 [migration/0]
```

top **command**:

the `top` command is a dynamic and interactive tool that provides real-time information about system processes. It offers a comprehensive view of running processes, system resource utilization, and other critical system metrics.

Command:

```
$ top
```



```
[root@ip-172-31-35-30 ~]# top
top - 21:58:00 up  1:13,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 104 total,   1 running,  62 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   975572 total,   279944 free,   104056 used,   591572 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   725524 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
    1 root      20   0  123612   5504   3892 S  0.0  0.6   0:03.21 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 rcu_par_gp
    6 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 kworker/0:0H-ev
    8 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_tasks_rude_
   10 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_tasks_trace
   11 root      20   0       0      0      0 S  0.0  0.0   0:00.09 ksoftirqd/0
   12 root      20   0       0      0      0 I  0.0  0.0   0:00.16 rcu_sched
   13 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 migration/0
```

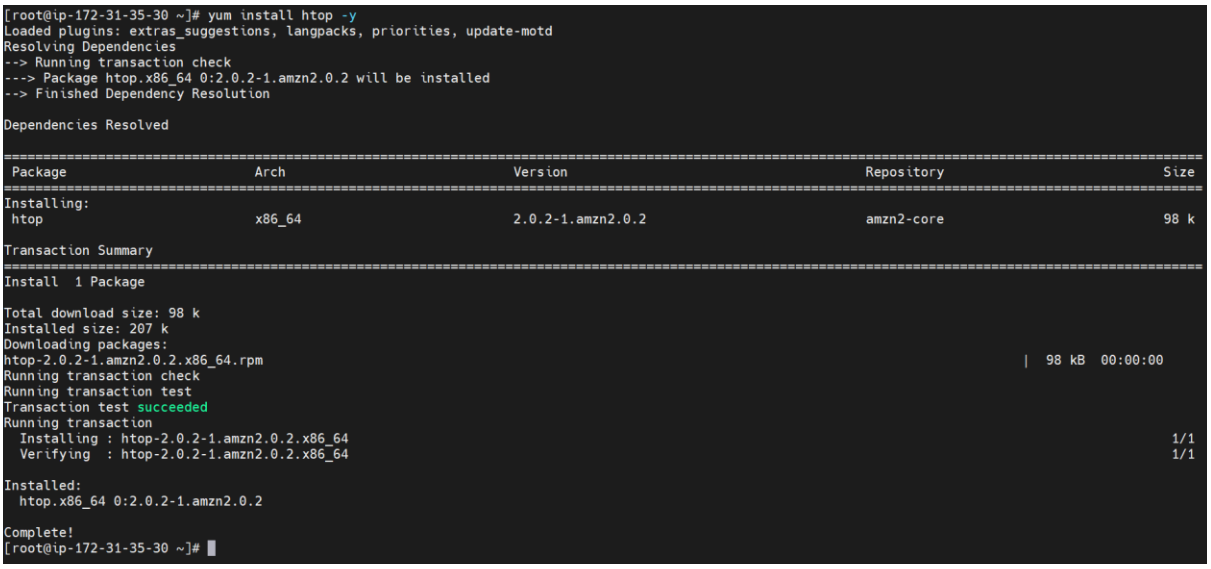
Here,

- **PID**: Process ID
- **USER**: Owner of the process
- **PR**: Priority
- **NI**: Nice value
- **VIRT**: Virtual memory usage
- **RES**: Resident set size (non-swapped physical memory used)
- **SHR**: Shared memory
- **S**: Process status (S: Sleeping, R: Running, I: Idle)
- **%CPU**: Percentage of CPU usage
- **%MEM**: Percentage of memory usage
- **TIME+:** Total CPU time
- **COMMAND**: Command or process name

htop **command**

`htop` is an interactive process viewer for Linux that provides a visually appealing and feature-rich alternative to the traditional `top` command. It allows users to monitor and manage system processes in real-time with an easy-to-use interface.
Before using `htop`, ensure it is installed on your system. Use the package manager yum to install the htop first, use command-

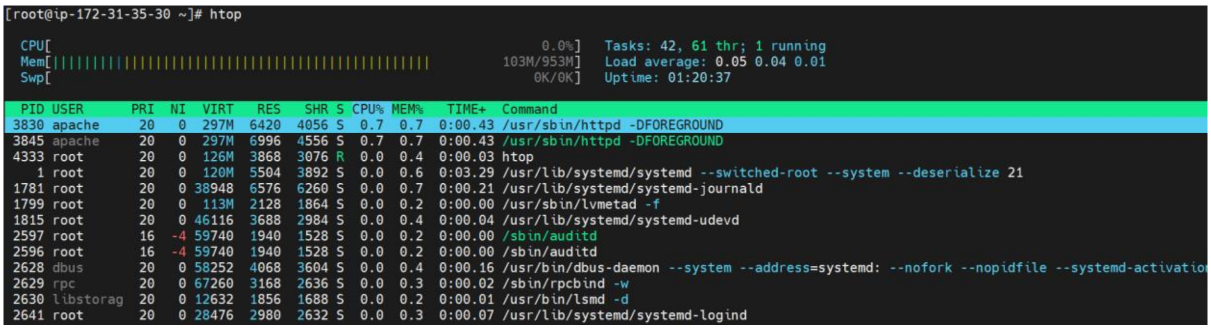
```
$ yum install htop -y
```



```
[root@ip-172-31-35-30 ~]# yum install htop -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package htop.x86_64 0:2.0.2-1.amzn2.0.2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================================================================
 Package                        Arch                             Version                                  Repository                Size
==========================================================================================================================================
Installing:
 htop                           x86_64                           2.0.2-1.amzn2.0.2                        amzn2-core                98 k

Transaction Summary
==========================================================================================================================================
Install  1 Package

Total download size: 98 k
Installed size: 207 k
Downloading packages:
htop-2.0.2-1.amzn2.0.2.x86_64.rpm                                                                                  |  98 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : htop-2.0.2-1.amzn2.0.2.x86_64                                                                                      1/1
  Verifying  : htop-2.0.2-1.amzn2.0.2.x86_64                                                                                      1/1

Installed:
  htop.x86_64 0:2.0.2-1.amzn2.0.2

Complete!
[root@ip-172-31-35-30 ~]#
```

Command:

```
$ htop
```



```
[root@ip-172-31-35-30 ~]# htop

  CPU[                                                     0.0%]   Tasks: 42, 61 thr; 1 running
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||  103M/953M]   Load average: 0.05 0.04 0.01
  Swp[                                                   0K/0K]   Uptime: 01:20:37

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 3830 apache     20   0  297M  6420  4056 S  0.7  0.7  0:00.43 /usr/sbin/httpd -DFOREGROUND
 3845 apache     20   0  297M  6996  4556 S  0.7  0.7  0:00.43 /usr/sbin/httpd -DFOREGROUND
 4333 root       20   0  126M  3868  3076 R  0.0  0.4  0:00.03 htop
    1 root       20   0  120M  5504  3892 S  0.0  0.6  0:03.29 /usr/lib/systemd/systemd --switched-root --system --deserialize 21
 1781 root       20   0 38948  6576  6260 S  0.0  0.7  0:00.21 /usr/lib/systemd/systemd-journald
 1799 root       20   0  113M  2128  1864 S  0.0  0.2  0:00.00 /usr/sbin/lvmetad -f
 1815 root       20   0 46116  3688  2984 S  0.0  0.4  0:00.04 /usr/lib/systemd/systemd-udevd
 2597 root       16  -4 59740  1940  1528 S  0.0  0.2  0:00.00 /sbin/auditd
 2596 root       16  -4 59740  1940  1528 S  0.0  0.2  0:00.00 /sbin/auditd
 2628 dbus       20   0 58252  4068  3604 S  0.0  0.4  0:00.16 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activatio
 2629 rpc        20   0 67260  3168  2636 S  0.0  0.3  0:00.02 /sbin/rpcbind -w
 2630 libstorag  20   0 12632  1856  1688 S  0.0  0.2  0:00.01 /usr/bin/lsmd -d
 2641 root       20   0 28476  2980  2632 S  0.0  0.3  0:00.07 /usr/lib/systemd/systemd-logind
```

Here,

- **PID**: Process ID
- **USER**: Owner of the process
- **PRI**: Priority
- **NI**: Nice value
- **VIRT**: Virtual memory usage
- **RES**: Resident set size (non-swapped physical memory used)
- **SHR**: Shared memory
- **S**: Process status (S: Sleeping, R: Running, I: Idle)
- **CPU%:** Percentage of CPU usage
- **MEM%:** Percentage of memory usage
- **TIME+:** Total CPU time
- **Command**: Command or process name

`kill` **Command**:

kill command in Linux, is a built-in command which is used to terminate processes manually. kill command sends a signal to a process that terminates the process.
We have to pass signal -9 to terminate the process

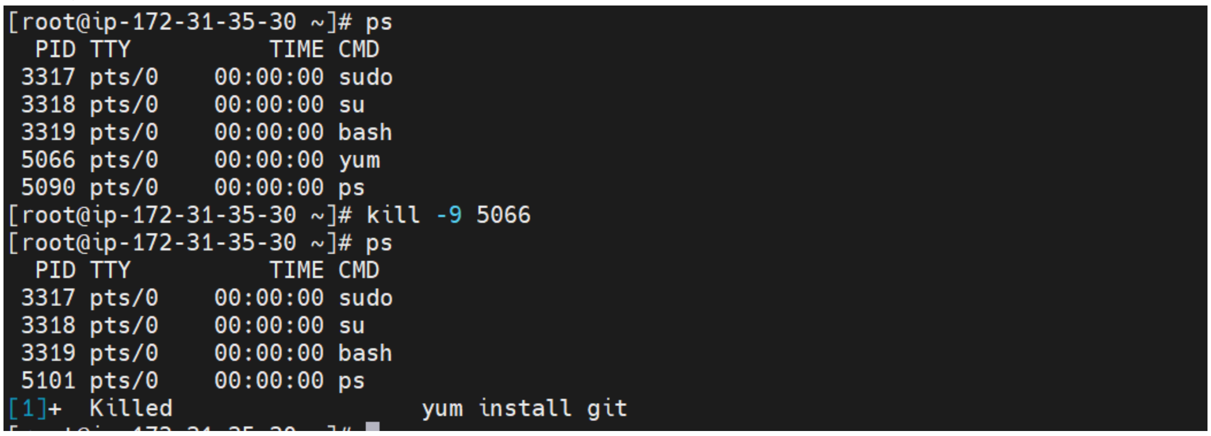
Command:

```
$ kill signal PID
```

Here,

- **PID** = The `kill` command requires the process ID (PID) of the process we want to terminate.
- **[signal]** = We have to specify the signal and if we don't specify the signal, the default signal `TERM` is sent to terminate the process. Use signal 9 to terminate process.

Example:



```
[root@ip-172-31-35-30 ~]# ps
  PID TTY          TIME CMD
 3317 pts/0    00:00:00 sudo
 3318 pts/0    00:00:00 su
 3319 pts/0    00:00:00 bash
 5066 pts/0    00:00:00 yum
 5090 pts/0    00:00:00 ps
[root@ip-172-31-35-30 ~]# kill -9 5066
[root@ip-172-31-35-30 ~]# ps
  PID TTY          TIME CMD
 3317 pts/0    00:00:00 sudo
 3318 pts/0    00:00:00 su
 3319 pts/0    00:00:00 bash
 5101 pts/0    00:00:00 ps
[1]+  Killed                  yum install git
```

wget and curl commands:

`wget` **command:**

Wget is the non-interactive network downloader which is used to download files from the server even when the user has not logged on to the system and it can work in the background without hindering the current process. Wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies. non-interactive, meaning that it can work in the background, while the user is not logged on. This allows you to start a retrieval and disconnect from the system, letting wget finish the work. By contrast, most of the Web browsers require constant user's presence, which can be a great hindrance when transferring a lot of data.

Command:

```
$ wget URL
```

`Curl` **command:**

Used to download the resources from URL

Command:

```
$ curl URL
```