Teacher: Utkarsh Tripathi
Email: ansutkarsh@gmail.com

Linux

Introduction to Linux

Kernel :

A kernel is the most important part of an operating system - it performs important functions like process management, memory management, filesystem management etc.

Linux is a kernel and not a complete operating system. Linux kernel is combined with GNU system to make a complete operating system. Therefore, linux based operating systems are also called as GNU/Linux systems. GNU is an extensive collection of free softwares like compiler, debugger, C library etc.

Linux distributions :

A Linux distribution(distro) is an operating system based on the Linux kernel and a package management system. A package management system consists of tools that help in installing, upgrading, configuring and removing softwares on the operating system.

List of popular Linux distributions :

- *Fedora*
- *Ubuntu*
- *Debian*
- *Centos*
- *Red Hat Enterprise Linux*

Packaging systems	Distributions	Package manager
Debian style (.deb)	Debian, Ubuntu	APT
Red Hat style (.rpm)	Fedora, CentOS, Red Hat Enterprise Linux	YUM

Shell vs Terminal :

Shell is a program that takes commands from the users and gives them to the operating system for processing. Shell is an example of a CLI (command line interface). Bash is one of the most popular shell programs available on Linux servers.

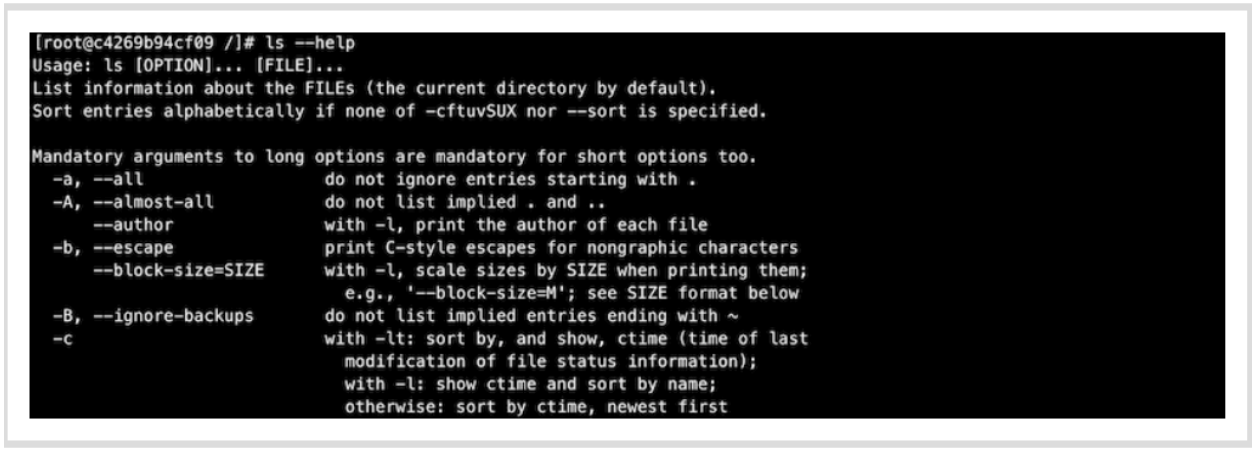
Terminal is a program that opens a window and lets you interact with the shell. Some popular examples of terminals are gnome-terminal, xterm, konsole etc.

Command Line Basics :

A command is a program that tells the operating system to perform specific work. Programs are stored as files in linux. Therefore, a command is also a file which is stored somewhere on the disk.

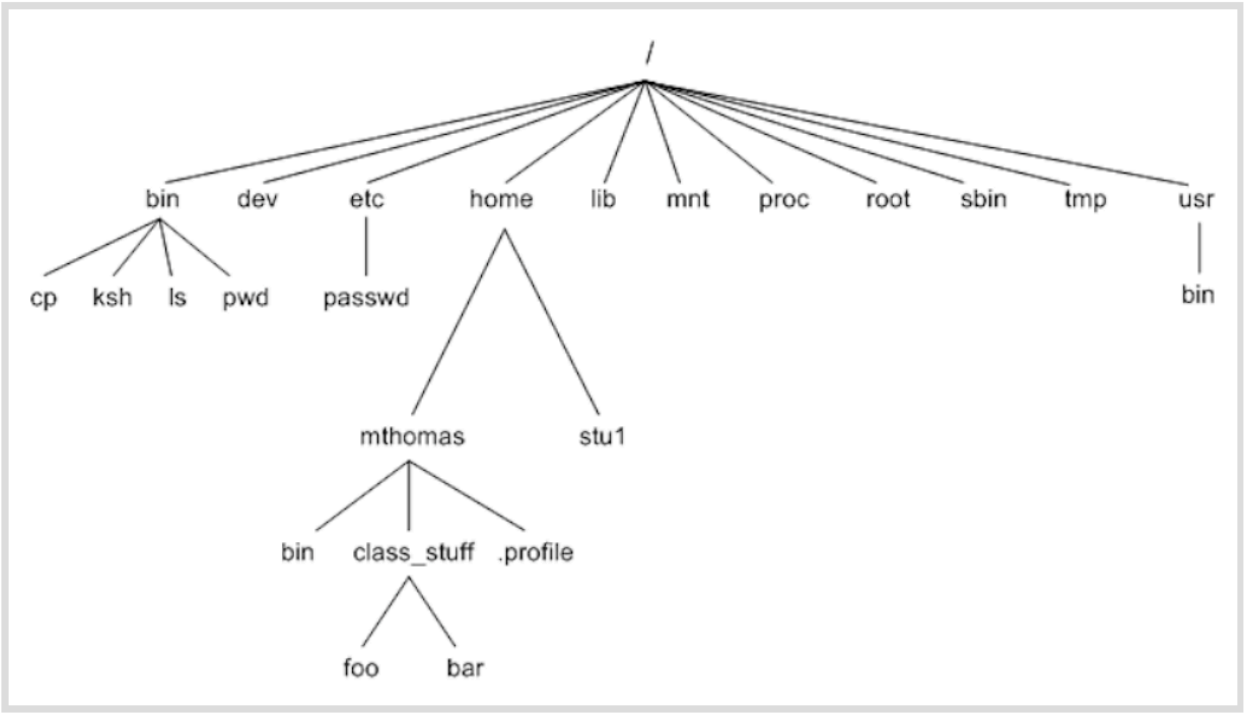
Commands may also take additional arguments as input from the user. These arguments are called command line arguments. Almost every command will have some form of documentation, most commands will have a command-line argument -h or --help that will display a reasonable amount of documentation. But the most popular documentation system in Linux is called man pages - short for manual pages.

EX:- *Using --help to show the documentation for ls command.*



File System Organization :

The linux file system has a hierarchical (or tree-like) structure with its highest level directory called root (denoted by /). Directories present inside the root directory stores file related to the system. These directories in turn can either store system files or application files or user related files.



- ***bin :-*** *The executable program of most commonly used commands reside in bin directory.*
- ***dev :-*** *This directory contains files related to devices on the system.*
- ***etc :-*** *This directory contains all the system configuration files.*
- ***home :-*** *This directory contains user related files and directories.*
- ***lib :-*** *This directory contains all the library files.*
- ***mnt :-*** *This directory contains files related to mounted devices on the system.*
- ***proc :-*** *This directory contains files related to the running processes on the system.*
- ***root :-*** *This directory contains root user related files and directories.*
- ***sbin :-*** *This directory contains programs used for system administration.*
- ***tmp :-*** *This directory is used to store temporary files on the system.*
- ***usr :-*** *This directory is used to store application programs on the system.*

Commands for Navigating the File System :

There are three basic commands which are used frequently to navigate the file system:

- *ls*
- *pwd*
- *cd*

pwd (print working directory) :

At any given moment of time, we will be standing in a certain directory. To get the name of the directory in which we are standing, we can use the pwd command in linux.

```
pwd
/etc/python
```

cd (change directory) :

The cd command can be used to change the working directory. Using the command, you can move from one directory to another.

```
pwd
/
cd /etc
pwd
/etc
```

ls (list files and directories)** :

The ls command is used to list the contents of a directory. It will list down all the files and folders present in the given directory.

If we just type ls in the shell, it will list all the files and directories present in the current directory.

```
ls
bin dev home    lib    media  proc  run_dir  sys  var
boot    etc inject  lib32  mnt    root  sbin     tmp
config  gocode  io  lib64  opt    run   srv      usr
```

We can also provide the directory name as argument to ls command. It will then list all the files and directories inside the given directory.

```
> ls /usr
bin  games    lib    libexec  sbin   src
doc  include  lib32  local    share
```

Commands for Manipulating Files :

There are five basic commands which are used frequently to manipulate files:

- *touch*
- *mkdir*
- *cp*
- *mv*
- *rm*

touch (create new file) :

The touch command can be used to create an empty new file. This command is very useful for many other purposes but we will discuss the simplest use case of creating a new file.

Syntax:-

```
touch <file_name>
```

```
> cd /home/runner/
> ls
_test_runner.py  UntidySardonicVoxel
> touch test_file
> ls
test_file  _test_runner.py  UntidySardonicVoxel
```

mkdir (create new directories) :

The mkdir command is used to create directories.You can use ls command to verify that the new directory is created.

Syntax:-

```
mkdir <directory_name>
```

```
> ls
test_file  _test_runner.py  UntidySardonicVoxel
> mkdir test_dir
> ls
test_dir  test_file  _test_runner.py  UntidySardonicVoxel
```

rm (delete files and directories) :

The rm command can be used to delete files and directories. It is very important to note that this command permanently deletes the files and directories. It's almost impossible to recover these files and directories once you have executed rm command on them successfully.

Syntax:-

```
rm <file_name>
```

```
> ls
test_dir  test_file  _test_runner.py  UntidySardonicVoxel
> rm test_file
> ls
test_dir  _test_runner.py  UntidySardonicVoxel
> rm -r test_dir
> ls
_test_runner.py  UntidySardonicVoxel
```

cp (copy files and directories) :

The cp command is used to copy files and directories from one location to another. Do note that the cp command doesn't do any change to the original files or directories. The original files or directories and their copy both co-exist after running cp command successfully.

Syntax:-

```
cp <source_path> <destination_path>
```

```
› ls
test_directory  _test_runner.py  UntidySardonicVoxel
› cp _test_runner.py test_directory
› ls
test_directory  _test_runner.py  UntidySardonicVoxel
› ls test_directory/
_test_runner.py
```

We can also use the cp command to copy the whole directory from one location to another.

```
› mkdir another_directory
› ls
another_directory  _test_runner.py
test_directory     UntidySardonicVoxel
› cp -r test_directory another_directory
› ls another_directory/
test_directory
```

We used the cp command along with an additional argument '-r' to copy the whole directory.

mv (move files and directories) :

The mv command can either be used to move files or directories from one location to another or it can be used to rename files or directories. Do note that moving files and copying them are very different. When you move the files or directories, the original copy is lost.

Syntax:-

```
mv <source_path> <destination_path>
```

```
› ls
another_directory  _test_runner.py
test_directory     UntidySardonicVoxel
› mv _test_runner.py test_directory
› ls
another_directory  test_directory  UntidySardonicVoxel
› ls test_directory/
_test_runner.py
```

One of the important uses of the mv command is to rename files and directories.

```
› ls
another_directory  test_directory  UntidySardonicVoxel
› cd test_directory
› ls
_test_runner.py
› mv _test_runner.py test.py
› ls
test.py
```

Commands for Viewing Files :

There are five basic commands which are used frequently to view the files:

- *cat*
- *head*
- *tail*
- *more*
- *less*

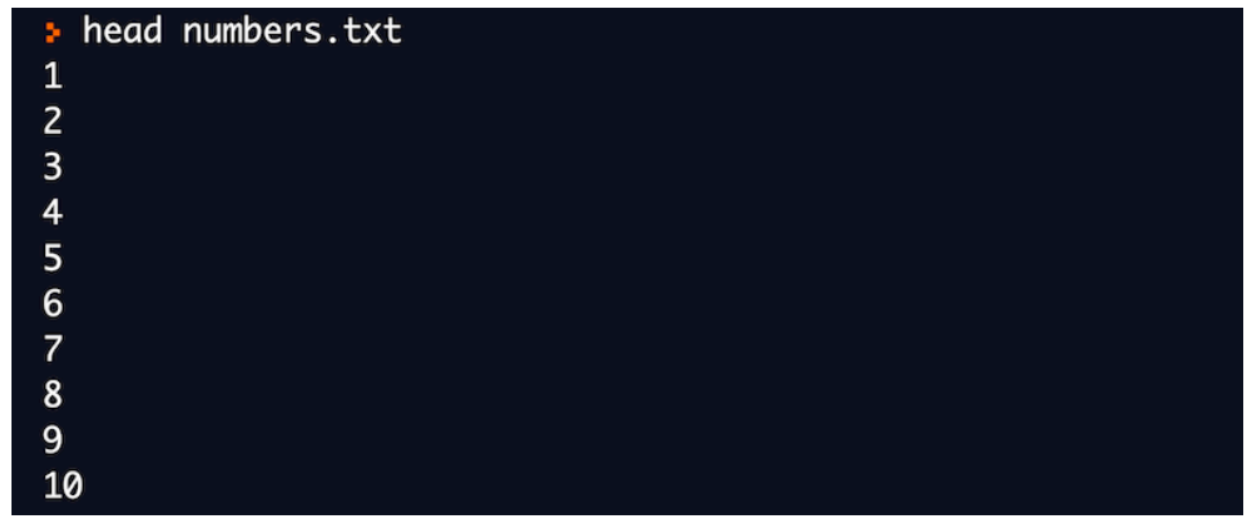
cat :

The most simplest use of cat command is to print the contents of the file on your output screen. This command is very useful and can be used for many other purposes.

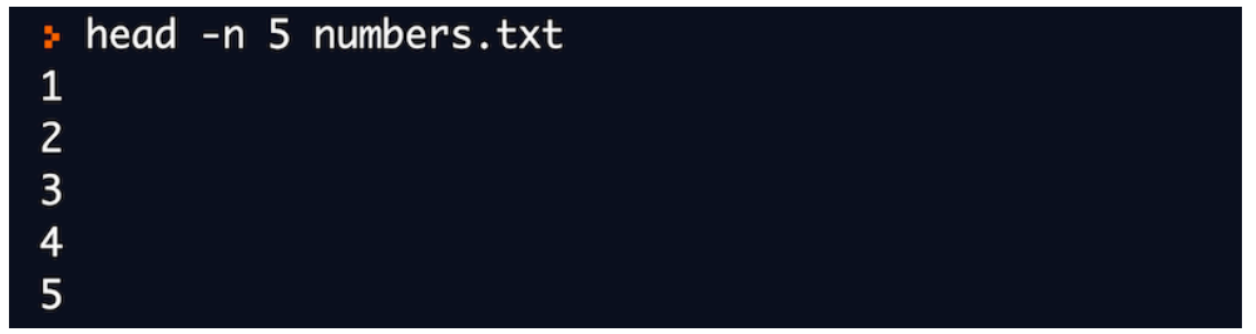
```
› cat numbers.txt
```

head :

The head command displays the first 10 lines of the file by default. We can include additional arguments to display as many lines as we want from the top.



By default, head command will only display the first 10 lines. If we want to specify the number of lines we want to see from start, use the '-n' argument to provide the input.



tail :

The tail command displays the last 10 lines of the file by default. We can include additional arguments to display as many lines as we want from the end of the file.

By default, the tail command will only display the last 10 lines. If we want to specify the number of lines we want to see from the end, use '-n' argument to provide the input.

```
> tail -n 5 numbers.txt
96
97
98
99
100
```

more :

More command displays the contents of a file or a command output, displaying one screen at a time in case the file is large (Eg: log files). It also allows forward navigation and limited backward navigation in the file.

```
> more numbers.txt
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
--More--(20%)
```

More command displays as much as can fit on the current screen and waits for user input to advance. Forward navigation can be done by pressing Enter, which advances the output by one line and Space, which advances the output by one screen.

less :

Less command is an improved version of more. It displays the contents of a file or a command output, one page at a time. It allows backward navigation as well as forward navigation in the file and also has search options. We can use arrow keys for advancing backward or forward by one line. For moving forward by one page, press Space and for moving backward by one page, press b on your keyboard. You can go to the beginning and the end of a file instantly.

Echo Command in Linux :

The echo command is one of the simplest commands that is used in the shell. The echo command prints the given input string on the screen.

```
echo "hello world"
hello world
```

Text Processing Commands :

There are three basic commands which are used frequently to process texts:

- *grep*
- *sed*
- *sort*

grep :

The grep command in its simplest form can be used to search particular words in a text file. It will display all the lines in a file that contains a particular input. The word we want to search is provided as an input to the grep command.

Syntax:-

```
grep <word_to_search> <file_name>
```

```
grep "1" numbers.txt
1
10
```

sed :

The sed command in its simplest form can be used to replace a text in a file.

Syntax:-

```
sed 's/<text_to_replace>/<replacement_text>/' <file_name>
```

Let's try to replace each occurrence of "1" in the file with "3" using sed command.

The content of the file will not change in the above example. To do so, we have to use an extra argument '-i' so that the changes are reflected back in the file.

sort :

The sort command can be used to sort the input provided to it as an argument. By default, it will sort in increasing order.

Now, we will try to sort the file using the sort command. The sort command sorts the content in lexicographical order.

The content of the file will not change in the above example.

I/O Redirection :

Each open file gets assigned a file descriptor. A file descriptor is an unique identifier for open files in the system. There are always three default files open, stdin (the keyboard), stdout (the screen), and stderr (error messages output to the screen). These files can be redirected.

We can use some special operators to redirect the output of the command to files or even to the input of other commands. I/O redirection is a very powerful feature.

In the below example, we have used the '>' operator to redirect the output of ls command to output.txt file.

```
> ls
numbers.txt  _test_runner.py  UntidySardonicVoxel
> ls > output.txt
> cat output.txt
numbers.txt
output.txt
_test_runner.py
UntidySardonicVoxel
```

In the below example, we have redirected the output from echo command to a file.

```
> echo "hello world" > hello_world.txt
> cat hello_world.txt
hello world
```

We can also redirect the output of a command as an input to another command. This is possible with the help of pipes.

In the below example, we have passed the output of cat command as an input to grep command using pipe(|) operator.

```
> cat numbers.txt | grep "3"
3
3
30
```

In the below example, we ha:ve passed the output of sort command as an input to uniq command using pipe(|) operator. The uniq command only prints the unique numbers from the input.

```
> sort numbers.txt | uniq
2
3
30
4
5
6
7
8
9
```

Linux Server Administration :

We can use docker container to ssetup the Linux box.

```
~ docker run -it --name test registry.access.redhat.com/ubi8 bash
[root@f3bc96e5ca7c /]#
```

User/Group Management :

- Users in Linux has an associated user ID called UID attached to them.
- Users also has a home directory and a login shell associated with them.
- A group is a collection of one or more users. A group makes it easier to share permissions among a group of users.
- Each group has a group ID called GID associated with it.

id command :

id command can be used to find the uid and gid associated with an user. It also lists down the groups to which the user belongs to.

The uid and gid associated with the root user is 0.

```
[root@24600d7e1583 /]# id
uid=0(root) gid=0(root) groups=0(root)
```

A good way to find out the current user in Linux is to use the whoami command.

```
[root@24600d7e1583 /]# whoami
root
```

"root" user or superuser is the most privileged user with unrestricted access to all the resources on the system. It has UID 0

Important files associated with users/groups :

/etc/passwd	Stores the user name, the uid, the gid, the home directory, the login shell etc
/etc/shadow	Stores the password associated with the users
/etc/group	Stores information about different groups on the system

```
[root@24600d7e1583 /]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
[root@24600d7e1583 /]# cat /etc/shadow
root:!locked::0:99999:7:::
bin:*:18199:0:99999:7:::
daemon:*:18199:0:99999:7:::
adm:*:18199:0:99999:7:::
lp:*:18199:0:99999:7:::
```

```
[root@24600d7e1583 /]# cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
```

Important commands for managing users :

useradd :

The useradd command adds a new user in Linux.

We will create a new user 'shivam'. We will also verify that the user has been created by tailing the /etc/passwd file. The uid and gid are 1000 for the newly created user. The home directory assigned to the user is /home/shivam and the login shell assigned is /bin/bash. Do note that the user home directory and login shell can be modified later on.

```
[root@24600d7e1583 /]# useradd shivam
[root@24600d7e1583 /]# tail /etc/passwd
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
shivam:x:1000:1000::/home/shivam:/bin/bash
```

If we do not specify any value for attributes like home directory or login shell, default values will be assigned to the user. We can also override these default values when creating a new user.

```
[root@24600d7e1583 /]# useradd amit -s /bin/sh
[root@24600d7e1583 /]# tail /etc/passwd
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
shivam:x:1000:1000::/home/shivam:/bin/bash
amit:x:1001:1001::/home/amit:/bin/sh
```

passwd :

The passwd command is used to create or modify passwords for a user. In the above examples, we have not assigned any password for users 'shivam' or 'amit' while creating them.

"!!" in an account entry in shadow means the account of an user has been created, but not yet given a password.

```
[root@24600d7e1583 /]# tail /etc/shadow
mail:*:18199:0:99999:7:::
operator:*:18199:0:99999:7:::
games:*:18199:0:99999:7:::
ftp:*:18199:0:99999:7:::
nobody:*:18199:0:99999:7:::
dbus:!!:18352::::::
systemd-coredump:!!:18352::::::
systemd-resolve:!!:18352::::::
shivam:!!:18570:0:99999:7:::
```

Let's now try to create a password for user "shivam".

```
[root@24600d7e1583 /]# passwd shivam
Changing password for user shivam.
New password:
/usr/share/cracklib/pw_dict.pwd.gz: No such file or directory
BAD PASSWORD: The password fails the dictionary check - error loading dictionary
Retype new password:
passwd: all authentication tokens updated successfully.
[root@24600d7e1583 /]# tail /etc/shadow
halt:*:18199:0:99999:7:::
mail:*:18199:0:99999:7:::
operator:*:18199:0:99999:7:::
games:*:18199:0:99999:7:::
ftp:*:18199:0:99999:7:::
nobody:*:18199:0:99999:7:::
dbus:!!:18352::::::
systemd-coredump:!!:18352::::::
systemd-resolve:!!:18352::::::
shivam:$6$JCnR6WzIW0ALrpqY$8B4DwC8TnijPb4G4h4tNDIN5L1eFQyFA1pcv/Z9mQuLpq3VSOVFNsUAC
vUBa8AGp2P1YZifWteKPKAwTLVaE81:18570:0:99999:7:::
```

usermod :

The usermod command is used to modify the attributes of an user like the home directory or the shell.

Let's try to modify the login shell of user "amit" to "/bin/bash".

```
[root@24600d7e1583 /]# usermod amit -s /bin/bash
[root@24600d7e1583 /]# tail /etc/passwd
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
shivam:x:1000:1000::/home/shivam:/bin/bash
amit:x:1001:1001::/home/amit:/bin/bash
```

userdel :

The userdel command is used to remove a user on Linux. Once we remove a user, all the information related to that user will be removed.

Let's try to delete the user "amit". After deleting the user, you will not find the entry for that user in "/etc/passwd" or "/etc/shadow" file.

```
[root@24600d7e1583 /]# userdel amit
[root@24600d7e1583 /]# tail /etc/passwd
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
shivam:x:1000:1000::/home/shivam:/bin/bash
```

Important commands for managing groups :

Commands for managing groups are quite similar to the commands used for managing users.

groupadd \<group_name>	Creates a new group
groupmod \<group_name>	Modifies attributes of a group
groupdel \<group_name>	Deletes a group
gpasswd \<group_name>	Modifies password for group

```
[root@24600d7e1583 /]# groupadd sre
[root@24600d7e1583 /]# tail /etc/group
utempter:x:35:
dbus:x:81:
input:x:999:
kvm:x:36:
render:x:998:
systemd-journal:x:190:
systemd-coredump:x:997:
systemd-resolve:x:193:
shivam:x:1000:
sre:x:1001:
```

We will now try to add user "shivam" to the group we have created above.

```
[root@24600d7e1583 /]# usermod -a -G sre shivam
[root@24600d7e1583 /]# groups shivam
shivam : shivam sre
[root@24600d7e1583 /]# tail /etc/group
utempter:x:35:
dbus:x:81:
input:x:999:
kvm:x:36:
render:x:998:
systemd-journal:x:190:
systemd-coredump:x:997:
systemd-resolve:x:193:
shivam:x:1000:
sre:x:1001:shivam
```

Becoming a Superuser :

Before running the below commands, do make sure that you have set up a password for user "shivam" and user "root" using the passwd command described in the above section.

The su command can be used to switch users in Linux. Let's now try to switch to user "shivam".

```
[root@24600d7e1583 /]# su shivam
[shivam@24600d7e1583 /]$ whoami
shivam
```

Let's now try to open the "/etc/shadow" file.

```
[shivam@24600d7e1583 /]$ tail /etc/shadow
tail: cannot open '/etc/shadow' for reading: Permission denied
[shivam@24600d7e1583 /]$
```

The operating system didn't allow the user "shivam" to read the content of the "/etc/shadow" file. This is an important file in Linux which stores the passwords of users. This file can only be accessed by root or users who have the superuser privileges.

The sudo command allows a user to run commands with the security privileges of the root user. *Do remember that the root user has all the privileges on a system. We can also use su command to switch to the root user and open the above file but doing that will require the password of the root user. An alternative way which is preferred on most modern operating systems is to use sudo command for becoming a superuser. Using this way, a user has to enter his/her password and they need to be a part of the sudo group.*

How to provide superpriveleges to other users ?

Let's first switch to the root user using su command. Do note that using the below command will need you to enter the password for the root user.

```
[shivam@24600d7e1583 /]$ su root
Password:
```

In case, you forgot to set a password for the root user, type "exit" and you will be back as the root user. Now, set up a password using the passwd command.

***The file /etc/sudoers holds the names of users permitted to invoke sudo**. In redhat operating systems, this file is not present by default. We will need to install sudo.*

```
[root@cfa888eee361 /]# yum install sudo
```

Try to open the "/etc/sudoers" file on the system. The file has a lot of information. This file stores the rules that users must follow when running the sudo command. For example, root is allowed to run any commands from anywhere.

```
[root@cfa888eee361 /]# grep "root" /etc/sudoers
## the root user, without needing the root password.
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
## cdrom as root
```

One easy way of providing root access to users is to add them to a group which has permissions to run all the commands. "wheel" is a group in redhat Linux with such privileges.

```
[root@cfa888eee361 /]# grep "wheel" /etc/sudoers
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
# %wheel        ALL=(ALL)       NOPASSWD: ALL
```

Let's add the user "shivam" to this group so that it also has sudo privileges.

```
[root@cfa888eee361 /]# usermod -a -G wheel shivam
[root@cfa888eee361 /]# id shivam
uid=1000(shivam) gid=1000(shivam) groups=1000(shivam),10(wheel)
```

Let's now switch back to user "shivam" and try to access the "/etc/shadow" file.

```
[shivam@cfa888eee361 /]$ sudo tail /etc/shadow

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for shivam:
halt:*:18367:0:99999:7:::
mail:*:18367:0:99999:7:::
operator:*:18367:0:99999:7:::
games:*:18367:0:99999:7:::
ftp:*:18367:0:99999:7:::
```

We need to use sudo before running the command since it can only be accessed with the sudo privileges. We have already given sudo privileges to user "shivam" by adding him to the group "wheel".

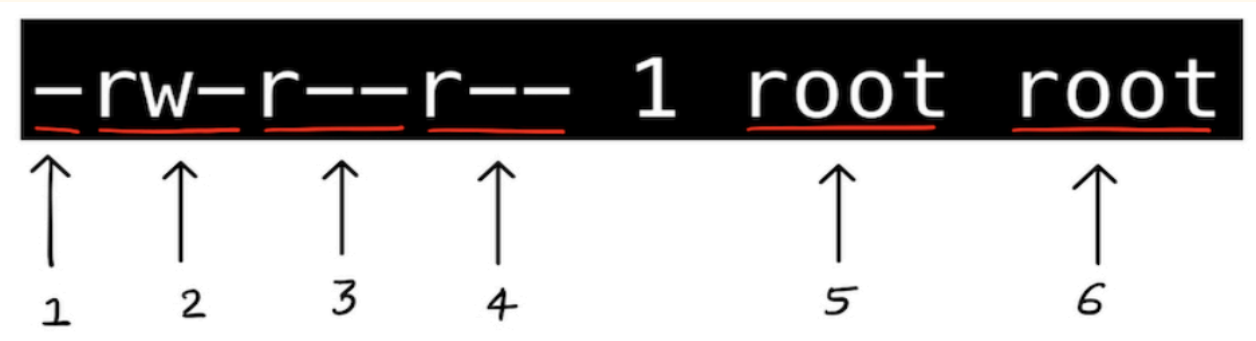
File Permissions :

On a Linux operating system, each file and directory is assigned access permissions for the owner of the file, the members of a group of related users and everybody else. This is to make sure that one user is not allowed to access the files and resources of another user.

To see the permissions of a file, we can use the ls command. Let's look at the permissions of /etc/passwd file.

```
[root@4b388f51f39a /]# ls -l /etc/passwd
-rw-r--r-- 1 root root 746 Nov  8 12:53 /etc/passwd
```

Let's go over some of the important fields in the output that are related to file permissions.



1. -	Type of the file. • - denotes regular file • d denotes a directory
2. rw-	Read(r), write(w) and execute(x) permissions of the owner. root is the owner in the above example. Root can read the file and write to it. This file is not executable.
3. r--	Read, write and execute permissions of the group owner of the file.The root group is the group owner of the file. The users in this group can only read the file
4. r--	Read, write and execute permissions of all other users. All other users can only read this file.
5. root	Name of the owner of the file
6. root	Name of the group owner of the file

Chmod command :

The chmod command is used to modify files and directories permissions in Linux.

The chmod command accepts permissions in as a numerical argument. We can think of permission as a series of bits with 1 representing True or allowed and 0 representing False or not allowed.

Permission	rwx	Binary	Decimal
Read, write and execute	rwx	111	7
Read and write	rw-	110	6
Read and execute	r-x	101	5
Read only	r--	100	4
Write and execute	-wx	011	3
Write only	-w-	010	2
Execute only	--x	001	1
None	---	000	0

We will now create a new file and check the permission of the file.

```
[root@4b388f51f39a /]# touch test_file
[root@4b388f51f39a /]# ls -l test_file
-rw-r--r-- 1 root root 0 Nov  8 18:18 test_file
```

The group owner doesn't have the permission to write to this file. Let's give the group owner or root the permission to write to it using chmod command.

```
[root@4b388f51f39a /]# chmod 664 test_file
[root@4b388f51f39a /]# ls -l test_file
-rw-rw-r-- 1 root root 0 Nov  8 18:18 test_file
```

Chmod command can be also used to change the permissions of a directory in the similar way.

Chown command :

The chown command is used to change the owner of files or directories in Linux.

Command syntax: *chown \<new_owner> \<file_name>*

```
[root@4b388f51f39a /]# ls -l test_file
-rw-rw-r-- 1 root root 0 Nov  8 18:18 test_file
[root@4b388f51f39a /]# chown shivam test_file
[root@4b388f51f39a /]# ls -l test_file
-rw-rw-r-- 1 shivam root 0 Nov  8 18:18 test_file
```

In case, we do not have sudo privileges, we need to use sudo command*. Let's switch to user 'shivam' and try changing the owner. We have also changed the owner of the file to root before running the below command.*

```
[shivam@4b388f51f39a /]$ ls -l test_file
-rw-rw-r-- 1 root root 0 Nov  8 18:18 test_file
[shivam@4b388f51f39a /]$ chown shivam test_file
chown: changing ownership of 'test_file': Operation not permitted
[shivam@4b388f51f39a /]$ sudo chown shivam test_file
[sudo] password for shivam:
[shivam@4b388f51f39a /]$ ls -l test_file
-rw-rw-r-- 1 shivam root 0 Nov  8 18:18 test_file
```

Chown command can also be used to change the owner of a directory in the similar way.

Chgrp command :

The chgrp command can be used to change the group ownership of files or directories in Linux. The syntax is very similar to that of chown command.

```
[shivam@4b388f51f39a /]$ ls -l test_file
-rw-rw-r-- 1 shivam root 0 Nov  8 18:18 test_file
[shivam@4b388f51f39a /]$ sudo chgrp shivam test_file
[shivam@4b388f51f39a /]$ ls -l test_file
-rw-rw-r-- 1 shivam shivam 0 Nov  8 18:18 test_file
```

Chgrp command can also be used to change the owner of a directory in the similar way.

SSH Command :

The ssh command is used for logging into the remote systems, transfer files between systems and for executing commands on a remote machine. SSH stands for secure shell and is used to provide an encrypted secured connection between two hosts over an insecure network like the internet.

Passwordless Authentication Using SSH :

Using this method, we can ssh into hosts without entering the password. This method is also useful when we want some scripts to perform ssh-related tasks.

Passwordless authentication requires the use of a public and private key pair. As the name implies, the public key can be shared with anyone but the private key should be kept private.

Steps for setting up a passwordless authentication with a remote host:

Generating public-private key pair :

Install openssh package which contains all the commands related to ssh.

```
[shivam@bbe5a6d63498 /]$ sudo yum install openssh
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Last metadata expiration check: 0:00:53 ago on Tue Nov 10 18:46:37 2020.
Dependencies resolved.
================================================================================
 Package              Architecture          Version                  Repository                Size
================================================================================
Installing:
 openssh              x86_64                8.0p1-5.el8              ubi-8-baseos              520 k
```

Generate a key pair using the ssh-keygen command. One can choose the default values for all prompts.

```
[shivam@bbe5a6d63498 /]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/shivam/.ssh/id_rsa):
Created directory '/home/shivam/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/shivam/.ssh/id_rsa.
Your public key has been saved in /home/shivam/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:WAcNquuV9A6wIIZduGWuXPekYb88RKApThdhJFmOa6w shivam@bbe5a6d63498
The key's randomart image is:
+---[RSA 3072]----+
|  .+=.  oo       |
|  .*. .. ..      |
|  o =+... .      |
|.ooB+. o..       |
|o+Bo= *.S        |
|.=.+ B O.        |
|E o o =.+        |
|   . . +..       |
|    .   +.       |
+----[SHA256]-----+
```

After running the ssh-keygen command successfully, we should see two keys present in the \~/.ssh directory. Id_rsa is the private key and id_rsa.pub is the public key. Do note that the private key can only be read and modified by you.

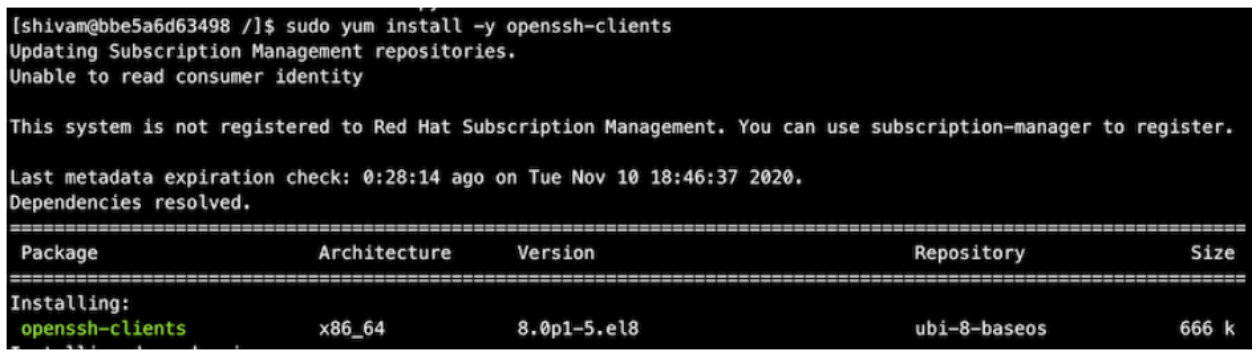
```
[shivam@bbe5a6d63498 /]$ ls -l ~/.ssh/
total 8
-rw------- 1 shivam shivam 2655 Nov 10 18:49 id_rsa
-rw-r--r-- 1 shivam shivam  573 Nov 10 18:49 id_rsa.pub
```

Transferring the public key to the remote host :

There are multiple ways to transfer the public key to the remote server. We will look at one of the most common ways of doing it using the ssh-copy-id command.

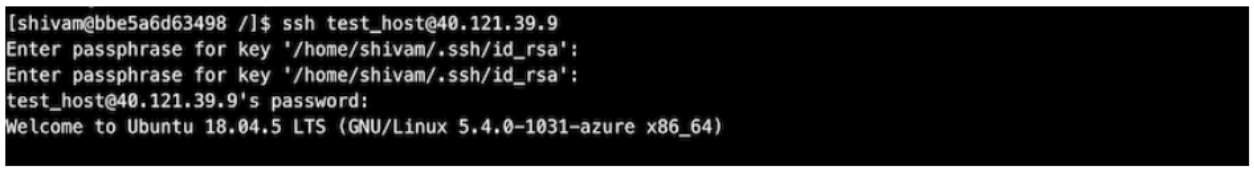
```
[shivam@bbe5a6d63498 /]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCvRwgY73KotZr8HEwej/aMUHlG479ScThbeUpzy863/+OEjUcmjnxjVbHWwg5ZV2ureuXGoaOT
jkZtWbFiKUC+WUJHZSjgcyh2wCgN/iyCiMhZUUXRsRfOTJ1gWRaruPP+1YJLMdxocTbfx1lsTNCOtJnGVaLrQTT+jeNp67jECm2g620acaEtWWT3
TU5IJ8+lM021OWtC9tOFYpgVxGlg3rzG1ZMhlcnZ1JPALdffCsqZRMJ48zkuK3QbaIwdKhi967w9B6MeQNZkZ6AFetoyWzTLYGeAq5N0yHWoHe9b
qNx7R4VY/JBDHvNPDniQ7eh0PZAh2InhCq3gk1ug1L6nWKi2DW2lXDghLBjGbRKhVruzHKbmRMdsay6FMK8Jw+fNRxef0jD6M0Xgh6YYbEac4GVL
l6HWVarp/0zRRJm+ylLxvu/xd9XGNr82H/QPurd7CFjNhs+DgwIcAPFhheLiZh53NCXxlAcu2/un9xvuSBtqIG42QayGs4Hjw9EZtkE= shivam@
bbe5a6d63498
```

Install the openssh-clients package to use ssh-copy-id command.



Use the ssh-copy-id command to copy your public key to the remote host.

Now, ssh into the remote host using the password authentication.



Our public key should be there in \~/.ssh/authorized_keys now.

\~/.ssh/authorized_key contains a list of public keys. The users associated with these public keys have the ssh access into the remote host.

How to run commands on a remote host ?

General syntax: *ssh \<user>@\<hostname/hostip> \<command>*

```
~ ssh shivam@40.71.100.197 'ps aux | head'
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2  78108  9368 ?        Ss   Oct09   1:28 /sbin/init
root         2  0.0  0.0      0     0 ?        S    Oct09   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   Oct09   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   Oct09   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   Oct09   0:00 [kworker/0:0H-kb]
root         9  0.0  0.0      0     0 ?        I<   Oct09   0:00 [mm_percpu_wq]
root        10  0.0  0.0      0     0 ?        S    Oct09   0:40 [ksoftirqd/0]
root        11  0.0  0.0      0     0 ?        I    Oct09  10:26 [rcu_sched]
root        12  0.0  0.0      0     0 ?        S    Oct09   0:10 [migration/0]
```

How to transfer files from one host to another host ?

General syntax: *scp \<source> \<destination>*

```
~ scp test_file shivam@40.71.100.197:/home/shivam
test_file

~ ssh shivam@40.71.100.197 'ls /home/shivam'
test_file
```

Package Management :

Package management is the process of installing and managing software on the system. We can install the packages which we require from the Linux package distributor. Different distributors use different packaging systems.

Packaging systems	Distributions
Debian style (.deb)	Debian, Ubuntu
Red Hat style (.rpm)	Fedora, CentOS, Red Hat Enterprise Linux

Popular Packaging Systems in Linux :

Command	Description
yum install \<package_name>	Installs a package on your system
yum update \<package_name>	Updates a package to it's latest available version
yum remove \<package_name>	Removes a package from your system
yum search \<keyword>	Searches for a particular keyword

DNF is the successor to YUM which is now used in Fedora for installing and managing packages. DNF may replace YUM in the future on all RPM based Linux distributions.

```
[shivam@4b388f51f39a log]$ sudo yum install httpd
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to regi
ster.

Last metadata expiration check: 3:34:26 ago on Sun Nov  8 18:31:38 2020.
Dependencies resolved.
================================================================================================================
 Package                Arch       Version                                      Repository             Size
================================================================================================================
Installing:
 httpd                  x86_64     2.4.37-30.module+el8.3.0+7001+0766b9e7       ubi-8-appstream       1.4 M
Installing dependencies:
 apr                    x86_64     1.6.3-11.el8                                 ubi-8-appstream       125 k
 apr-util               x86_64     1.6.1-6.el8                                  ubi-8-appstream       105 k
 httpd-filesystem       noarch     2.4.37-30.module+el8.3.0+7001+0766b9e7       ubi-8-appstream        37 k
 httpd-tools            x86_64     2.4.37-30.module+el8.3.0+7001+0766b9e7       ubi-8-appstream       104 k
 mailcap                noarch     2.1.48-3.el8                                 ubi-8-baseos           39 k
 mod_http2              x86_64     1.15.7-2.module+el8.3.0+7670+8bf57d29        ubi-8-appstream       154 k
 redhat-logos-httpd     noarch     81.1-1.el8                                   ubi-8-baseos           26 k
```

After httpd is installed, we will use the yum remove command to remove httpd package.

```
[shivam@4b388f51f39a log]$ sudo yum remove httpd
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to regi
ster.

Dependencies resolved.
================================================================================================================
 Package                Arch       Version                                      Repository             Size
================================================================================================================
Removing:
 httpd                  x86_64     2.4.37-30.module+el8.3.0+7001+0766b9e7       @ubi-8-appstream      4.3 M
```

Process Management :

ps (process status) :

The ps command is used to know the information of a process or list of processes.

```
[shivam@4b388f51f39a /]$ ps
  PID TTY          TIME CMD
   98 pts/0    00:00:00 bash
  127 pts/0    00:00:00 ps
```

*If you get an error "ps command not found" while running ps command, do install **procps** package.*

ps without any arguments is not very useful. Let's try to list all the processes on the system by using the below command.

- ***ps:** This basic command displays a snapshot of active processes running in the current terminal session.*
- ***ps aux:** This is one of the most commonly used variations. It displays a detailed list of all processes running on the system, including user, CPU usage, memory usage, and other information.*

  ```
  [shivam@4b388f51f39a /]$ ps aux
  USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
  root           1  0.0  0.1  19316  3584 pts/0    Ss   12:52   0:00 bash
  root          97  0.0  0.2 109012  5632 pts/0    S    18:34   0:00 su shivam
  shivam        98  0.0  0.1  19320  3748 pts/0    S    18:34   0:00 bash
  shivam       146  0.0  0.1  51816  3776 pts/0    R+   19:42   0:00 ps aux
  ```

  The columns typically include:

  - ***USER:** The owner of the process.*
  - ***PID:** Process ID, a unique identifier for each process.*
  - ***%CPU:** The percentage of CPU usage.*
  - ***%MEM:** The percentage of RAM usage.*
  - ***VSZ:** Virtual memory size in kilobytes.*
  - ***RSS:** Resident set size, the portion of a process's memory held in RAM.*

    - ***TTY:*** *Terminal type associated with the process.*
    - ***STAT:*** *Process status.*
    - ***START:*** *Start time of the process.*
    - ***TIME:*** *Total accumulated CPU time*

- ***ps -e or ps -ef:*** *Lists information about all processes on the system. The -e option is equivalent to aux, displaying additional details.*
- ***ps -p <PID>:*** *Displays information about a specific process identified by its PID.*

```
[shivam@4b388f51f39a /]$ ps -p 98
  PID TTY          TIME CMD
   98 pts/0    00:00:00 bash
```

- ***ps -u <username>:*** *Displays information about processes owned by a specific user.*
- ***ps -ax:*** *Lists all processes on the system, similar to ps aux.*

top :

The top command is used to show information about Linux processes running on the system in real time. It also shows a summary of the system information.

```
top - 19:28:23 up 1 day, 21:40,  0 users,  load average: 0.00, 0.04, 0.01
Tasks:   4 total,   1 running,   3 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.1 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
MiB Mem :   1998.8 total,    290.5 free,    252.4 used,   1455.9 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1591.6 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
    1 root      20   0   19316   3584   3036 S   0.0   0.2   0:00.13
   97 root      20   0  109012   5632   4772 S   0.0   0.3   0:00.00
   98 shivam    20   0   19320   3744   3188 S   0.0   0.2   0:00.04
  133 shivam    20   0   56264   4116   3508 R   0.0   0.2   0:00.01
```

- ***PID (Process ID):*** *A unique identifier for each process.*
- ***USER:*** *The user who owns the process.*
- ***PR (Priority):*** *The scheduling priority of the process.*
- ***NI (Nice Value):*** *The "niceness" of the process, which influences its priority.*
- ***VIRT (Virtual Memory):*** *The total virtual memory used by the process.*
- ***RES (Resident Set Size)****: The portion of the process's memory held in RAM.*

- ***SHR (Shared Memory):*** *The amount of shared memory used by the process.*
- ***%CPU:*** *The percentage of CPU usage by the process.*
- ***%MEM:*** *The percentage of RAM usage by the process.*
- ***TIME+:*** *The total accumulated CPU time used by the process.*

Memory Management :

free :

The free command is used to display the memory usage of the system. The command displays the total free and used space available in the RAM along with space occupied by the caches/buffers.

```
[shivam@4b388f51f39a /]$ free
              total        used        free      shared  buff/cache   available
Mem:        2046748      258104      296608         856     1492036     1630060
Swap:       1048572           0     1048572
```

free command by default shows the memory usage in kilobytes. We can use an additional argument to get the data in human-readable format.

```
[shivam@4b388f51f39a /]$ free -h
              total        used        free      shared  buff/cache   available
Mem:          2.0Gi       251Mi       289Mi       0.0Ki       1.4Gi       1.6Gi
Swap:         1.0Gi          0B       1.0Gi
```

vmstat :

The vmstat command can be used to display the memory usage along with additional information about io and cpu usage.

```
[shivam@4b388f51f39a /]$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 5  0      0 296832  54264 1438124    0    0     1     2   25   52  0  0 100  0  0
```

Checking Disk Space :

df (disk free) :

The df command is used to display the free and available space for each mounted file system.

```
[shivam@4b388f51f39a /]$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
overlay         61255492 2230168  55884000   4% /
tmpfs              65536       0     65536   0% /dev
tmpfs            1023372       0   1023372   0% /sys/fs/cgroup
/dev/sda1       61255492 2230168  55884000   4% /etc/hosts
shm                65536       0     65536   0% /dev/shm
tmpfs            1023372       0   1023372   0% /proc/acpi
tmpfs            1023372       0   1023372   0% /sys/firmware
```

- ***df -h :-*** *Used to display sizes in a human-readable format (e.g., KB, MB, GB).*
- ***df --exclude=filesystem_name :-*** *The --exclude option allows you to exclude specific file systems from the output.*
- ***df -t nfs :-*** *To display information about remote file systems mounted using NFS or other network file systems, use the -t option.*

du (disk usage) :

The du command is used to display disk usage of files and directories on the system.

```
[shivam@4b388f51f39a etc]$ du -h | head
12K     ./libnl
4.0K    ./sasl2
4.0K    ./binfmt.d
4.0K    ./dbus-1/session.d
12K     ./dbus-1/system.d
28K     ./dbus-1
4.0K    ./terminfo
4.0K    ./crypto-policies/local.d
4.0K    ./crypto-policies/policies/modules
8.0K    ./crypto-policies/policies
```

Managing System Services :

Service units end with .service file extension. Systemctl command can be used to start/stop/restart the services managed by systemd.

Command	Description
systemctl start name.service	Starts a service
systemctl stop name.service	Stops a service
systemctl restart name.service	Restarts a service
systemctl status name.service	Check the status of a service
systemctl reload name.service	Reload the configuration of a service

Logs :

In this section, we will talk about some important files and directories which can be very useful for viewing system logs and applications logs in Linux. These logs can be very useful when you are troubleshooting on the system.

File/Directory	Description
/var/log/*	Stores logs related to daemon processes along with system logs. Some important log files present in /var/log directory are: • /var/log/messages - contains logs related to system errors, system booting and shutdown, system configuration changes etc. • /var/log/authlog - contains system authorization related logs • /var/log/lastlog - contains the recent login information of all users