**Practical : 01**  **Date:**

**Aim :** Basics concept- Introduction to Linux Operating system, Basic command in Linux and writing shell script in Vi editor.

# Theory:Linux distribution is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software

**Architecture of Linux**

Linux architecture has the following components:

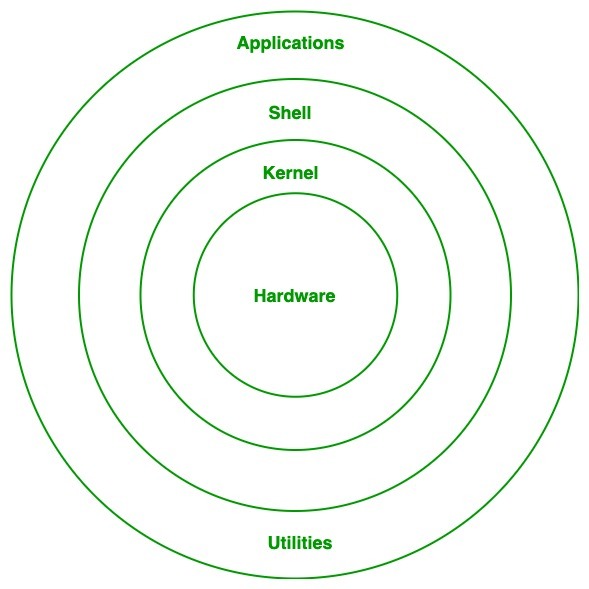


Fig:01 Linux architecture

**Kernel:**Kernel is the core of the Linux based operating system. It virtualizes the common hardware resources of the computer to provide each process with its virtual resources. This makes the process seem as if it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes. Different types of the kernel are:

1. Monolithic Kernel
2. Hybrid kernels
3. Exo kernels
4. Micro kernels
5. **System Library:**This is the special types of functions that are used to implement the functionality of the operating system.
6. **Shell:**It is an interface to the kernel which hides the complexity of the kernel’s functions from the users. It takes commands from the user and executes the kernel’s functions.
7. **Hardware Layer:**This layer consists all peripheral devices like RAM/ HDD/ CPU etc.
8. **System Utility:**It provides the functionalities of an operating system to the user.

**Advantages of Linux :**

The main advantage of Linux, is it is an open-source operating system. This means the source code is easily available for everyone and you are allowed to contribute, modify and distribute the code to anyone without any permissions.

In terms of security, Linux is more secure than any other operating system. It does not mean that Linux is 100 percent secure it has some malware for it but is less vulnerable than any other operating system. So, it does not require any anti-virus software.

The software updates in Linux are easy and frequent.

Various Linux distributions are available so that you can use them according to your requirements or according to your taste.

Linux is freely available to use on the internet.

It has large community support.

**Disadvantages of Linux :**

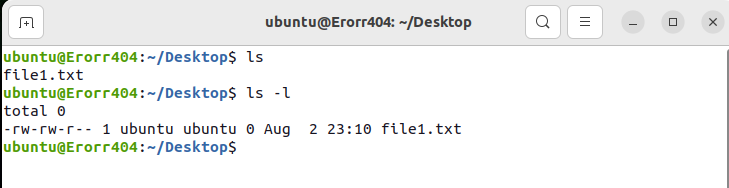
* It is not very user-friendly. So, it may be confusing for beginners.
* It has small peripheral hardware drivers as compared to windows.

**Basic Commands**

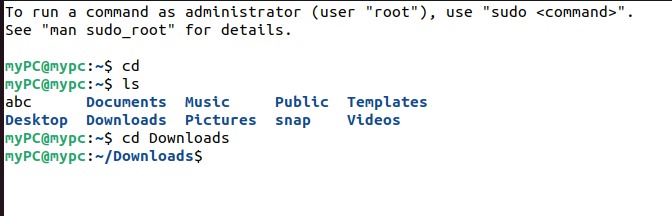
**1. pwd** **:** When you first open the terminal, you are in the home directory of your user. To know which directory you are in, you can use the **“pwd”** command. It gives us the absolute path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash( / ). The user directory is usually something like "/home/username".

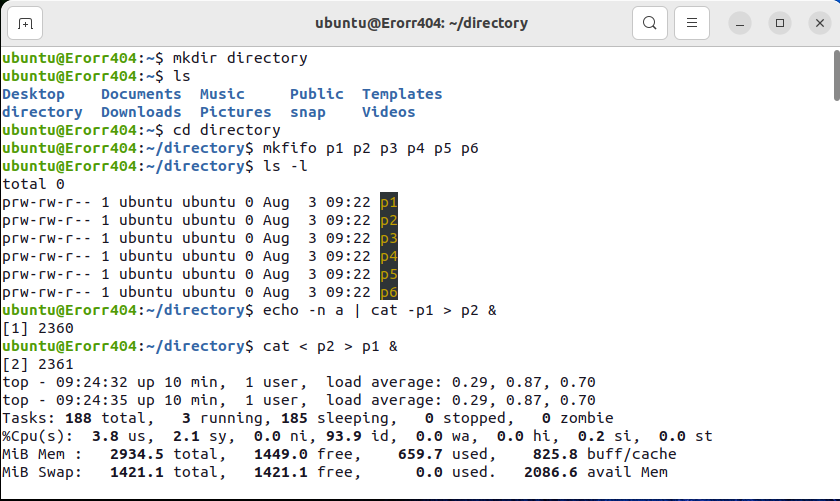


**2. ls :** Use the **"ls"** command to know what files are in the directory you are in. You can see all the hidden files by using the command **“ls -a”**.



**3. cd** **:** Use the **"cd"** command to go to a directory. For example, if you are in the home folder, and you want to go to the downloads folder, then you can type in **“cd Downloads”**. Remember, this command is case sensitive, and you have to type in the name of the folder exactly as it is. But there is a problem with these commands. Imagine you have a folder named “Raspberry Pi”. In this case, when you type in **“cd Raspberry Pi”**, the shell will take the second argument of the command as a different one, so you will get an error saying that the directory does not exist. Here, you can use a backward slash. That is, you can use **“cd Raspberry\ Pi”** in this case. Spaces are denoted like this: If you just type **“cd”** and press enter, it takes you to the home directory. To go back from a folder to the folder before that, you can type “**cd** ..” . The two dots represent back.

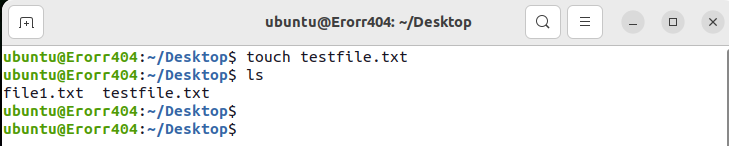


**4. mkdir  :** Use the **mkdir** command when you need to create a folder or a directory. For example, if you want to make a directory called “DIY”, then you can type **“mkdir DIY**”.

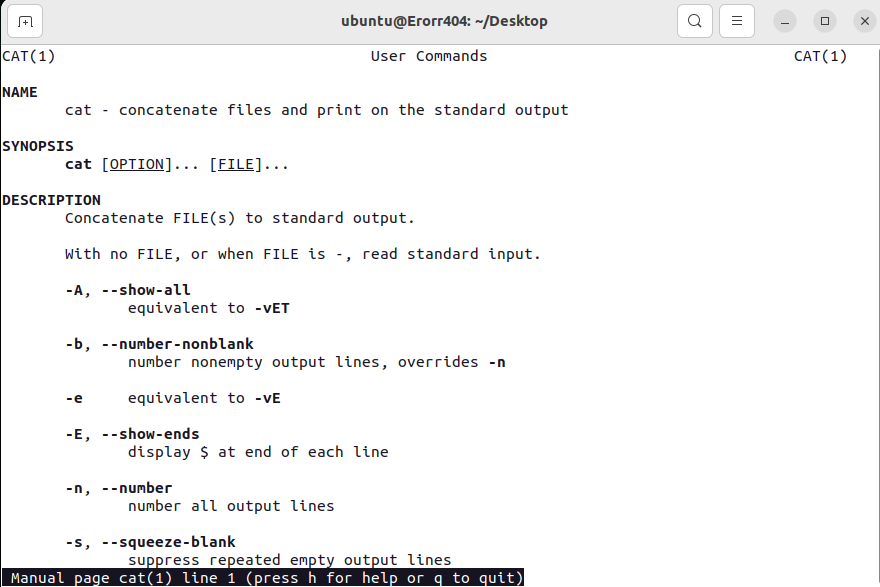
**5. rm** **:** Use the **rm** command to delete files and directories.  Use "**rm -r**" to delete just the directory. It deletes both the folder and the files it contains when using only the **rm** command.



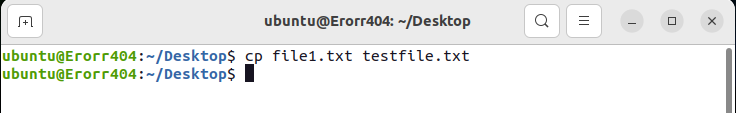
**6. touch** **:** The**touch** command is used to create a file. It can be anything, from an empty txt file to an empty zip file. For example, “**touch new.txt**”.



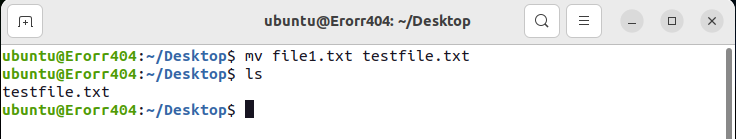
**7. man & --help** **:** To know more about a command and how to use it, use the **man** command. It shows the manual pages of the command. For example, “**man cd**” shows the manual pages of the **cd**command. Typing in the command name and the argument helps it show which ways the command can be used (e.g., **cd –help**).



**8. cp** **:** Use the **cp**command to copy files through the command line. It takes two arguments: The first is the location of the file to be copied, the second is where to copy.

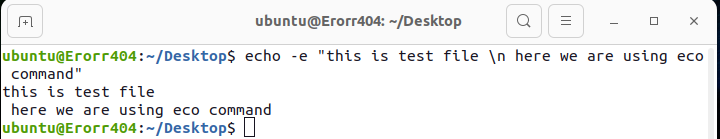


**9. mv** **:** Use the **mv** command to move files through the command line. We can also use the **mv** command to rename a file. For example, if we want to rename the file “**text**” to “**new**”, we can use “**mv text new**”. It takes the two arguments, just like the**cp** command

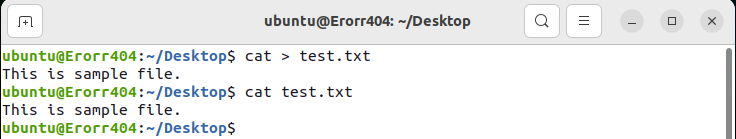


**Intermediate Commands**

**1. echo :** The "**echo**" command helps us move some data, usually text into a file. For example, if you want to create a new text file or add to an already made text file, you just need to type in, “**echo hello, my name is alok >> new.txt**”. You do not need to separate the spaces by using the backward slash here, because we put in two triangular brackets when we finish what we need to write.



**2. cat :** Use the **cat** command to display the contents of a file. It is usually used to easily view programs.

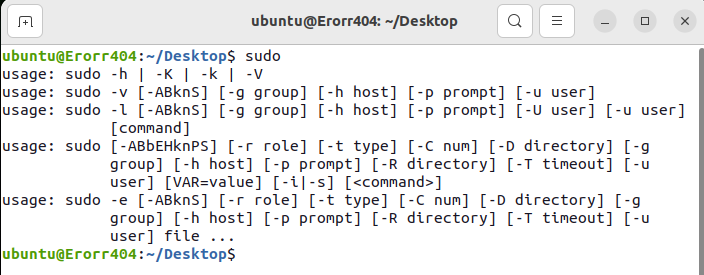


**3. du** **:** Use **du** to know the disk usage of a file in your system. If you want to know the disk usage for a particular folder or file in Linux, you can type in the command **df** and the name of the folder or file. For example, if you want to know the disk space used by the documents folder in Linux, you can use the command “**du Documents**”. You can also use the command “**ls -lah**” to view the file sizes of all the files in a folder.

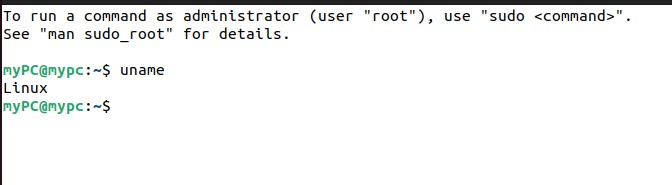


**4. reboot command** **:** It may be used to [halt, power-off or reboot a system](https://www.tecmint.com/shutdown-poweroff-halt-and-reboot-commands-in-linux/" \t "_blank) as follows.

**5.[sudo command](https://www.tecmint.com/sudoers-configurations-for-setting-sudo-in-linux/" \t "_blank) :**  allows a permitted system user to run a command as root or another user, as defined by the security policy such as sudo users



1. **[uname command](https://www.tecmint.com/find-linux-kernel-version-distribution-name-version-number/" \t "_blank)** **:**displays system information such as operating system, network node hostname kernel name, version and release etc.



**Conclusion**: Thus, We have studied the Linux operating systems and the basic command of Linux. And I have implemented some basic command of Linux successfully.

**Practical :** 2                             Date-

**AIM: Process management**

1. Use ps to search for init process by name
2. What is the process id of init process?
3. Use the who am I command to determine the terminal name
4. Using your terminal name from above, use ps to find all process           associates with your terminal.
5. What is the process id  of your shell
6. What is the parent process id of your shell
7. Start two instances of sleep 4432 in the background
8. Locate the process id of all the sleep command
9. Display those two sleep processes on top then quite top
10. Use the standard kill to kill one of the processes

**Theory:**

**Process Management:**

A process (or job) is the fundamental unit of work in an operating system. Process management includes creating and deleting processes and providing mechanisms for processes to communicate and synchronize with each other.

It handles operations by performing tasks like process scheduling and such as resource allocation.

* **ps command:**

**ps** displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use *top*(1) instead.

This version of **ps** accepts several kinds of options:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| 1 | UNIX options, which may be grouped and must be preceded by a dash. |
| 2 | BSD options, which may be grouped and must not be used with a dash. |
| 3 | GNU long options, which are preceded by two dashes. |

options of ps command

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and **ps** implementations that this **ps** is compatible with.

**Syntax: ~$ ps**

It displays the following information in the output.

* **PID**: It shows the unique process ID.
* **TTY**: It shows the terminal type into which the user is logged in.
* **TIME**: It displays the total time that the process has been running.
* **CMD**: It displays the name of the command that launches the process. As we can notice in the output, the second process is started by the ps command itself.
* **Process id**: Each process has a unique identifier. The PID is used to specify the process to the operating system when an application makes a system call to signal, modify, or wait for the process. Additional identifiers associate the process with a process group (typically, a tree of processes forked by a single user command) and login session.

* **whoami command**: It is basically the concatenation of the strings “who”,”am”,”i” as whoami. It displays the username of the current user when this command is invoked. It is similar as running the id command with the options -un.

**Syntax:** ~$ whoami

* **ps command**: Command to check the process status (ps command

Item, Description, PID, Process ID, PPID, Parent process ID

Process id of shell: The parent process ID of your current context is exposed as an environment variable. To see the value, you can echo it out. For example, if you are SSH 'd into a Linux server,

the value of the $PPID environment variable will the process ID of the SSH process. The environment variable only outputs the process ID. Syntax: ~$ ps $ PPID

* **sleep command**: The **sleep** command suspends the calling process of the next command for a specified amount of time. This property is useful when the following command’s execution depends on the successful completion of a previous command.

# Process id of sleep command:

**Syntax:** ~$ sleep process I'd

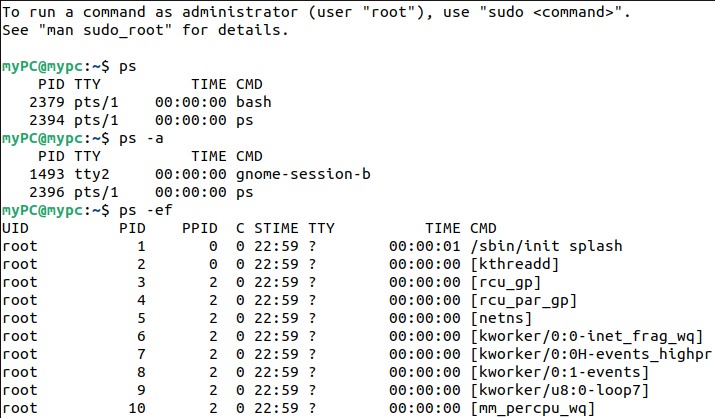
* **top command**: The top command is a well-known and most widely used utility to display dynamic real-time information about running processes in Linux and Unix-like operating systems. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

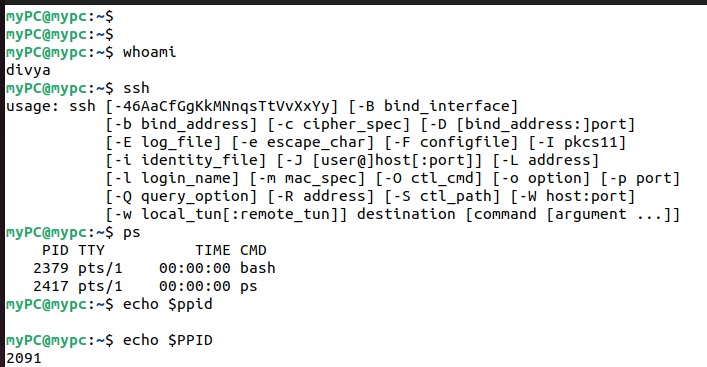
**Syntax**: ~$ top

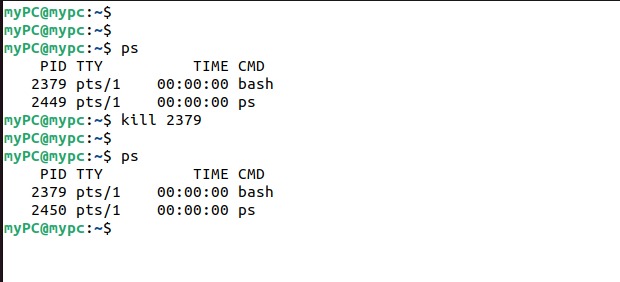
# kill command: The kill command transfers a signal towards a process in which further terminates the process. When the user does not describe any signal that is to be transferred with the kill command then the **TERM** signal (default) is transferred that will terminate the process. If the signal is not described, then it will default to **-15** (**-TERM**).

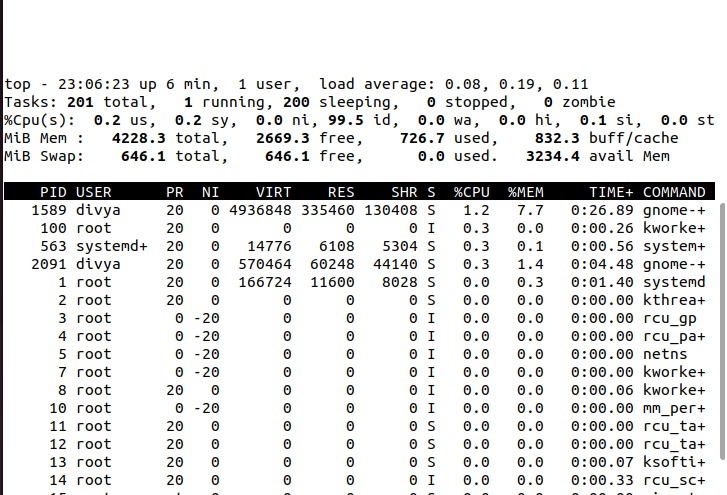
**Syntax:** ~$ kill

# Implementation:









**Conclusion**: Thus, we have successfully understood the process management and implemented all its topics.

# Practical : 03 Date:

**Aim :** Process priorities:

1. Create a new directory and create six pipes in that directory.
2. Bounce a character between two pipes.
3. use top and ps to display.
4. Bounce another character between two other pipes but this same time start the command nice, verify that all cat processes are battling to the CPU.
5. Use ps to verify that the two new cat process have a nice value. Use the o and c.
6. use renice to increase the nice value from 10 to 15. Notice the difference between the usual commands.

# Theory :

Priority value — The priority value is the process’s actual priority which is used by the Linux kernel to schedule a task.

In Linux system priorities are 0 to 139 in which 0 to 99 for real-time and 100 to 139 for users.

Nice value — Nice values are user-space values that we can use to control the priority of a process. The nice value range is -20 to +19 where -20 is highest, 0 default and +19 is lowest.

The relation between nice value and priority is as such - Priority\_value

=Nice\_value+20

All processes have a priority and a nice value. Higher priority processes will get more CPU time than lower priority processes. You can influence this with the nice and renice commands.

**nice and renice command** :  
nice command is used to start a process with specified nice value, which renice command is used to alter priority of running process.

**Usage of nice command :**

programs running on it(processes) are not responding quickly, in that case if you want to kill some of the processes, you need to start a terminal, if you start your bash shell normally, it will also

produce lag but you can avoid this by starting the bash shell with high priority.

For example:

nice -n -5 bash

**Usage of renice command :**  
To alter priority of running process, we use renice command.

renice value PID

value is new priority to be assigned  
PID is PID of process whose priority is to be changed

pipes (mkfi):

a pipe is a connection between two processes, such that the standard output from one process becomes the standard input of the other process. In UNIX Operating System, Pipes are useful for communication between related processes (interprocess communication).

Processes can communicate with each other via pipes. These pipes can be created with the mkfifo command.

Syntax for creating pipes in directory:

~$ mkdir pipes; cd pipes

~$ mkfifo p1 p2 p3….

~$ ls-1 Cat:

To demonstrate the use of the top and renice commands we will make the cat command use the

previously created pipes to generate a full load on the CPU. The cat is copied with a distinct name to the current directory. Syntax: top:

Just running top without options or arguments will display all processes and an overview of

information. The top of the top screen might look something like this.

**nice and renice command** :  
 nice command is used to start a process with specified nice value, which renice command is used to alter priority of running process.

**Usage of nice command :**

programs running on it(processes) are not responding quickly, in that case if you want to kill some of the processes, you need to start a terminal, if you start your bash shell normally, it will also produce lag but you can avoid this by starting the bash shell with high priority.

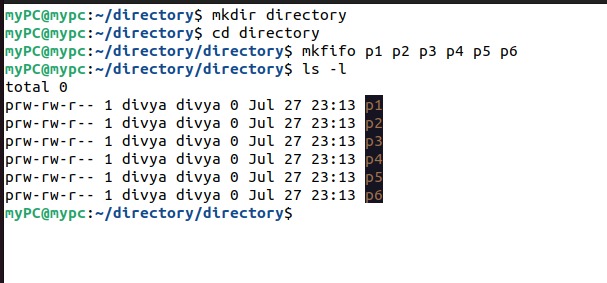
For example:

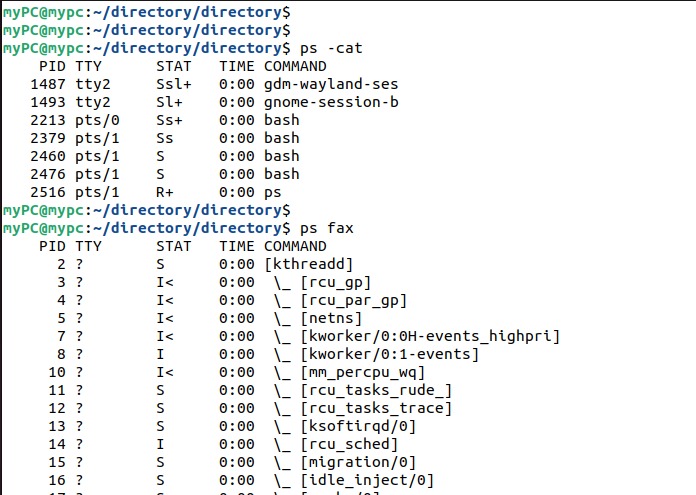
nice -n -5 bash

**Usage of renice command :**  
To alter priority of running process, we use renice command.

renice value PID

value is new priority to be assigned  
PID is PID of process whose priority is to be changed





**Conclusion**: Thus, we have successfully understood the process priorities and implemented all its topics.

**Practical :** 4                            Date-

**AIM:** Disk partitions

1. Use fdisk -l to display existing partitions and sizes.
2. Use df -h to display existing partitions and sizes.
3. Compare the output of fdisk and df.
4. Create a 200MB primary partition on a small disk.
5. Create a 400MB primary partition and two 300MB logical drives on a big disk.
6. Use df - h and fdisk -l to verify your work.
7. Compare the output again of fdisk and df. Do both commands display the new partitions?
8. Create a backup with dd of the mbr that contains your 200MB primary partition.
9. Take a backup of the partition table containing your 400MB primary and 300MB logical drives. Make sure the logical drives are in the backup.
10. Remove all your partitions with fdisk. Then restore your backups.

**Theory:**

Disk partitioning is one step of [disk formatting](https://www.partitionwizard.com/help/what-is-disk-formatting.html" \t "_blank). It is the process of dividing a disk into one or more regions, the so called partitions. If a partition is created, the disk will store the information about thelocation and size of partitions in [partition table](https://www.minitool.com/lib/partition-table.html" \t "_blank) that is usually located in the first sector of a disk.

With the partition table, each partition can appear to the operating system as a logical disk and users can read and write dataon disk. And each partition can be managed separately.

Why we need it?

* To upgrade Hard Disk (to incorporate new Hrd Disk to the system)
* Dual Booting(Multiple operating systems on the same system)
* Efficient disk management
* Ensure backup and security
* Work with different File systems using the same system

**Commands use in disk partition:**

* **fdisk** : The fdisk is a Linux command called fixed disk/format disk and is used with the Linux/Unix-based systems for command-line-based disk manipulation process.

It is used to create and organize space for new partitions or new drives, rearrange old drives, and move and copy information to new disks. In your system, you can create as many partitions as you want, and you can extend the partitions using fdisk commands.

**Syntax-** fdisk [options] device

example: fdisk -l [device...]

**Options for fdisk:**

**-b**, **--sector-size** sectorsize

Specify the sector size of the disk. Valid values are 512,

1024, 2048, and 4096. (Recent kernels know the sector size.

Use this option only on old kernels or to override the

kernel’s ideas.) Since util-linux-2.17, **fdisk** differentiates

between logical and physical sector size. This option changes

both sector sizes to sectorsize.

**-c**, **--compatibility**[=mode]

Specify the compatibility mode, 'dos' or 'nondos'. The

default is non-DOS mode. For backward compatibility, it is

possible to use the option without the mode argument — then

the default is used. Note that the optional mode argument

cannot be separated from the **-c** option by a space, the

correct form is for example **-c**=dos.

**-h**, **--help**

Display a help text and exit.

**-l**, **--list**

List the partition tables for the specified devices and then

exit.

If no devices are given, the devices mentioned in

/proc/partitions (if this file exists) are used. Devices are

always listed in the order in which they are specified on the

command-line, or by the kernel listed in /proc/partitions.

**-x**, **--list-details**

Like **--list**, but provides more details.

**-o**, **--output** list

Specify which output columns to print. Use **--help** to get a

list of all supported columns.

The default list of columns may be extended if list is

specified in the format +list (e.g., **-o +UUID**)

**-t**, **--type** type

Enable support only for disklabels of the specified type, and

disable support for all other types.

* **df command**: The df command (short for disk free), is used to display information related to file systems about total space and available space. df [OPTION]... [FILE]... If no file name is given, it displays the space available on all currently mounted file systems.

Syntax: df [OPTION]... [FILE]..

**Options for df command**

* -a,- -all : It includes all the dummy files also in the output which are actually having zero block sizes.
* -B,- -block-size=S : This is the option we were talking in the above para which is used to scale sizes by SIZE like -BM prints sizes in units of 1,048,576 bytes.
  + -total : It is used to display the grand total for size.
* -h,- -human-readable : It print sizes in human readable format.
* -H,- -si : This option is same as -h but it use powers of 1000 instead of 1024.
* -i,- -inodes : This option is used when you want to display the inode information instead of block usage.
* -k : Its use is like –block-size-1k.
* -l,- -local : This will display the disk usage of only local file systems.
* -P,- -portability : It uses the POSIX output format.
* -t,- -type=TYPE : It will only show the output of file systems having type TYPE.
* -T,- -print-type : This option is used to print file system type shown in the output.
* -x,- -exclude-type=TYPE : It will exclude all the file systems having type TYPE from the output.
* -v : Ignored, included for compatibility reasons.

**Partition a Disk Using fdisk Command:**

Follow the steps below to partition a disk in Linux by using the fdisk command.

Step 1: List Existing Partitions

Sudofdisk -l

Step 2: Select Storage Disk

Select the storage disk you want to create partitions on by running the following command: Sudofdisk/dev/sdb

Step 3: Create a New Partition

1. Run the n command to create a new partition.
2. Select the partition number by typing the default number (2).
3. After that, you are asked for the starting and ending sector of your hard drive. It is best to type the default number in this section (3622912).
4. The last prompt is related to the size of the partition. You can choose to have several sectors or to set the size in megabytes or gigabytes. Type +2GB to set the size of the partition to 2GB.

A message appears confirming that the partition is created.

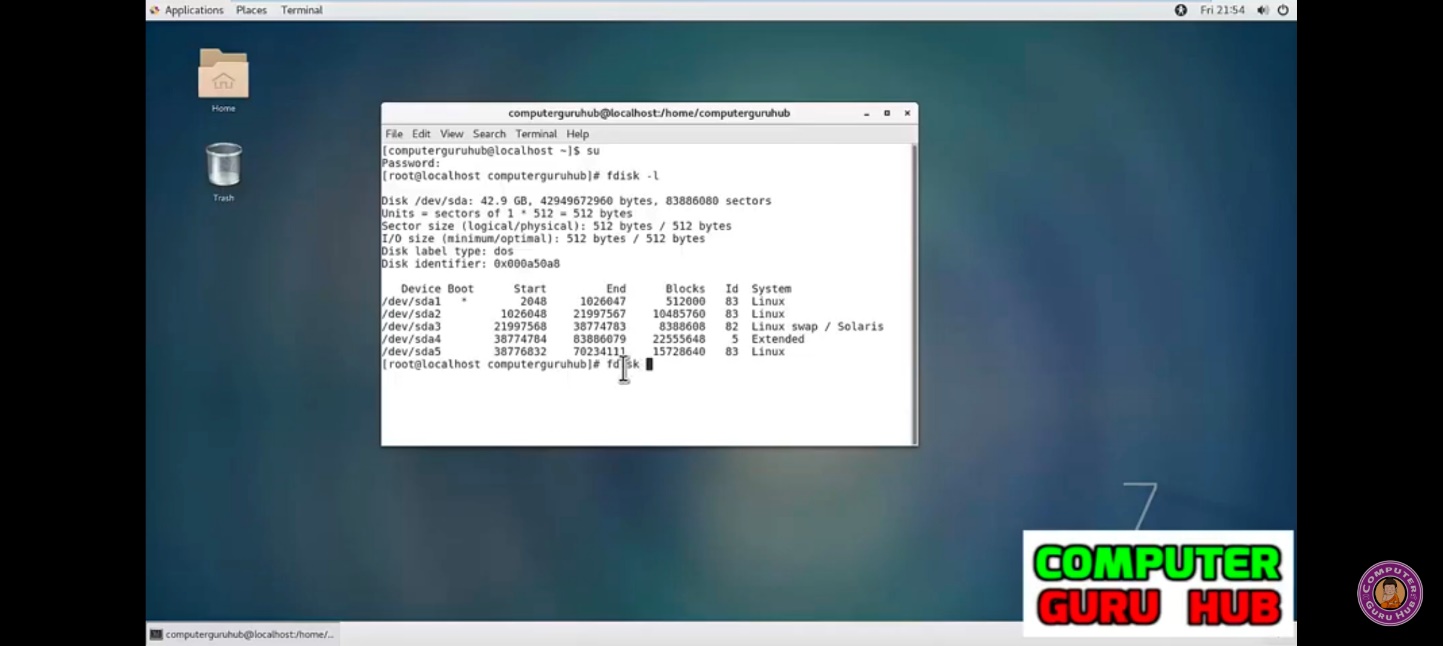
Step 4: Write on Disk

The system created the partition, but the changes are not written on the disk.

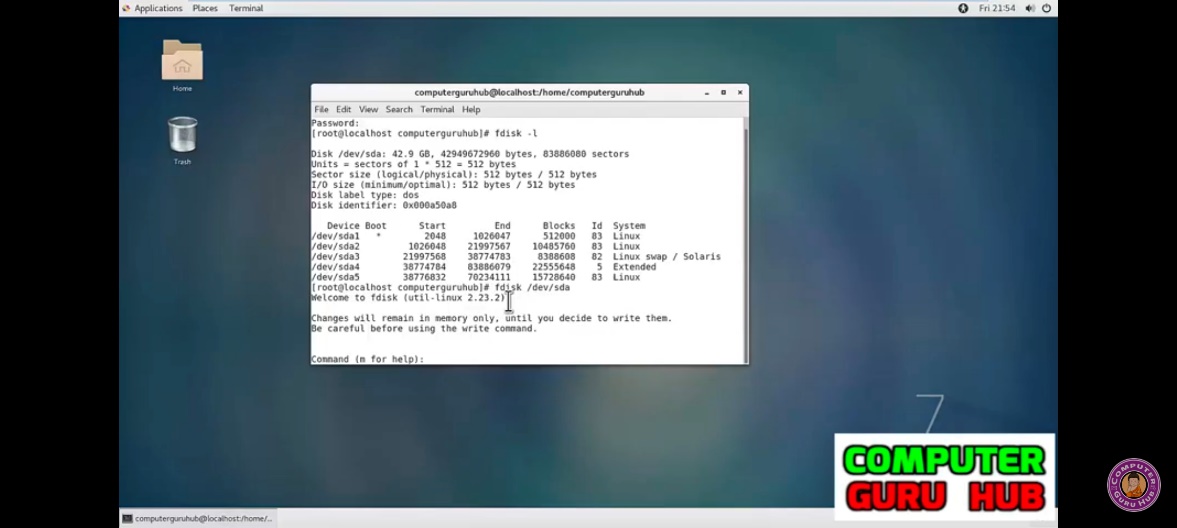
1. To write the changes on disk, run the w command:
2. Verify that the partition is created by running the following command: Sudofdisk -l

# Disk partition

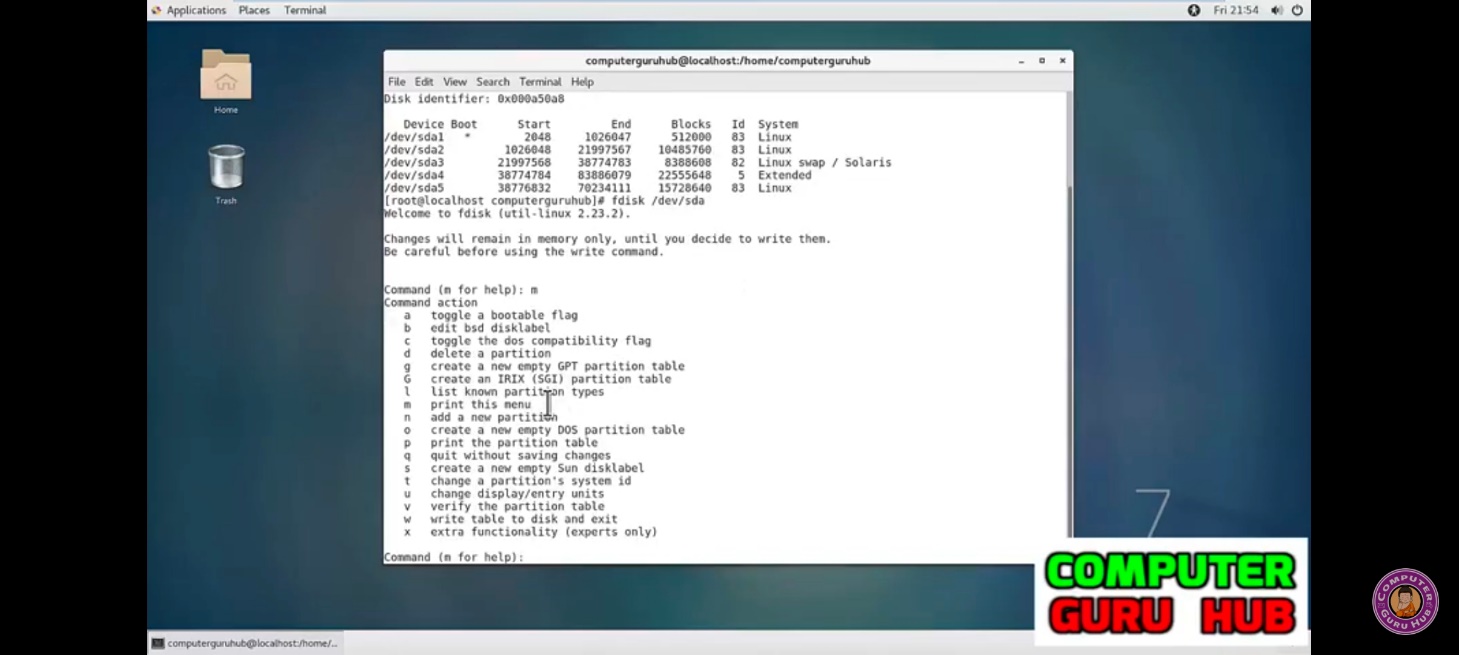
1. Use fdisk -l to display existing partitions and sizes.



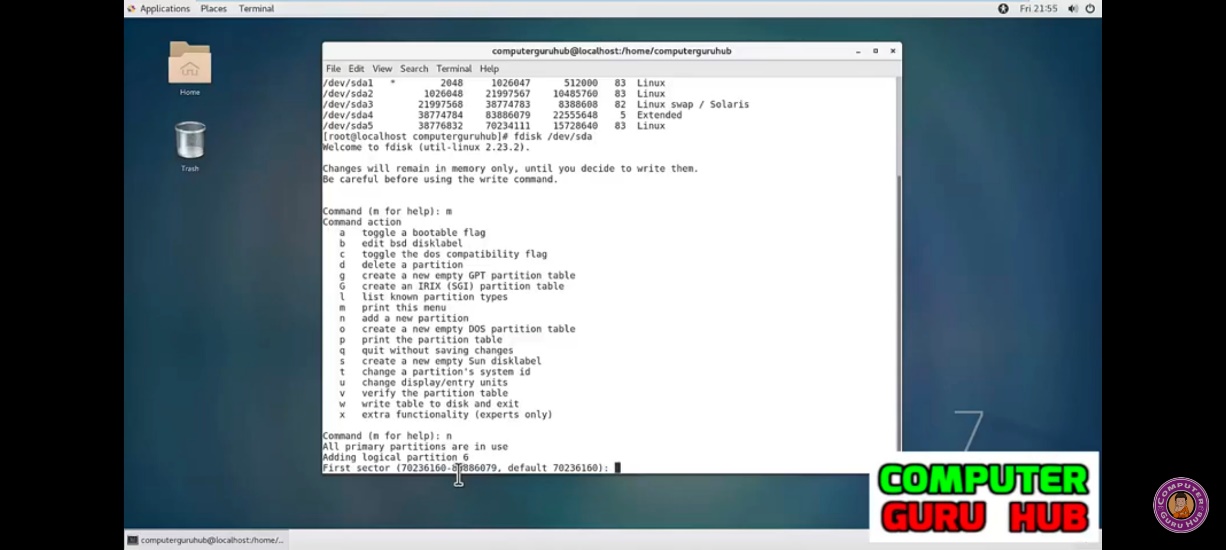
1. Use df -h to display existing partitions and sizes.
2. Compare the output of fdisk and df.

Some partitions will be listed in both outputs (maybe /dev/sda1 or /dev/hda1

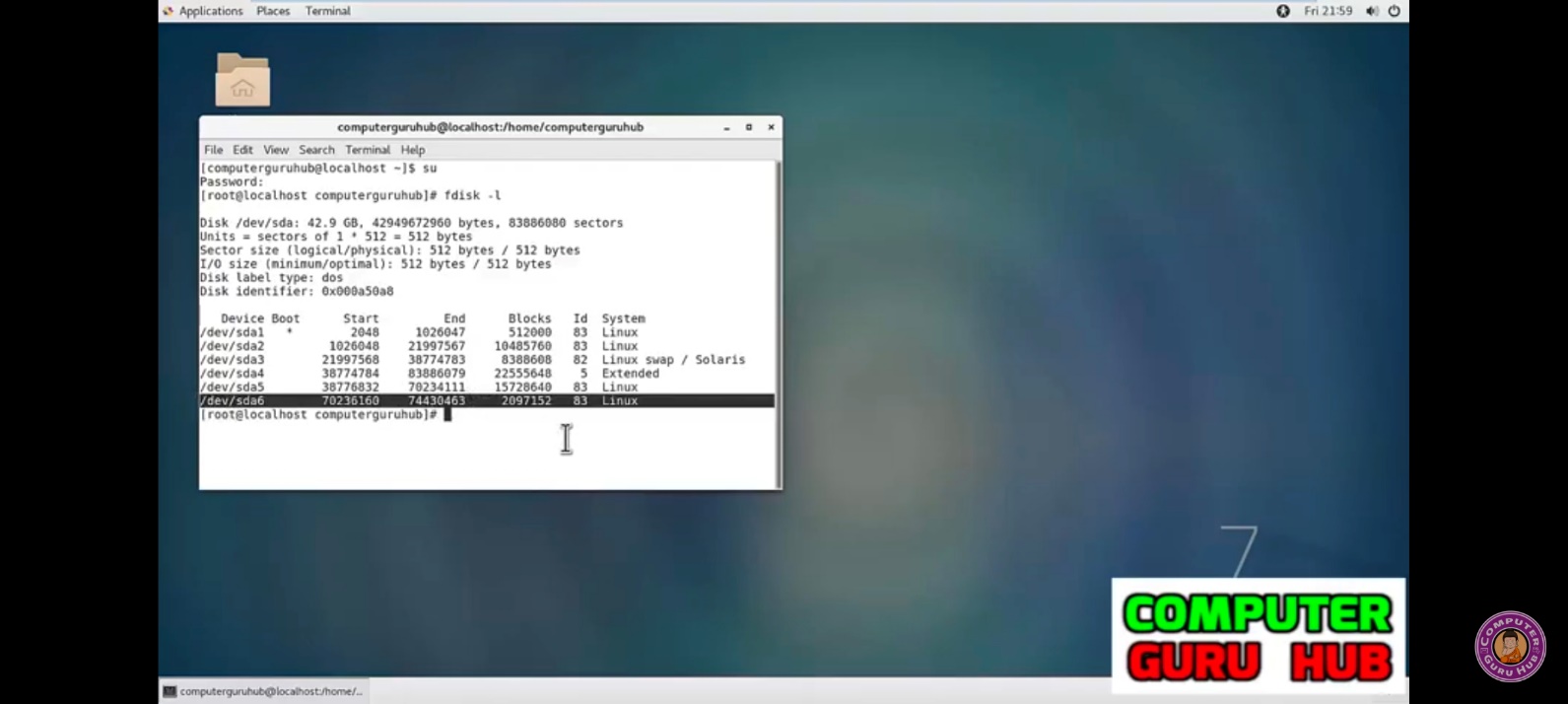
1. Create a 200MB primary partition on a small disk.



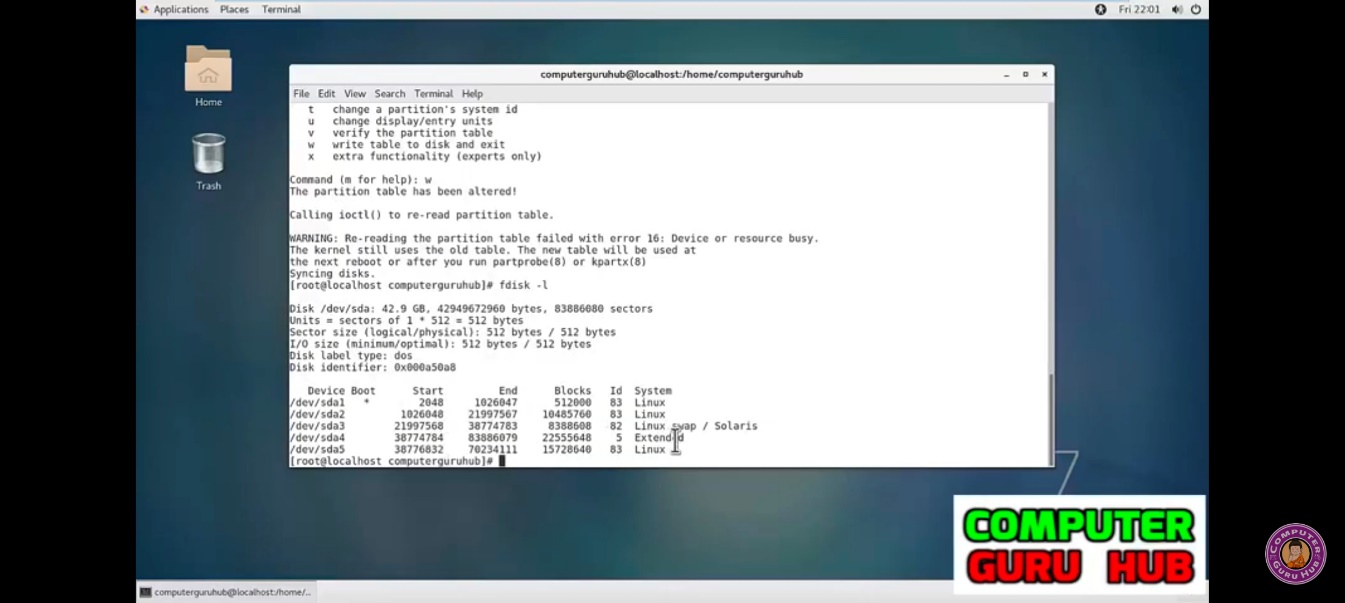
1. Create a 400MB primary partition and two 300MB logical drives on a big disk.



1. Use df -h and fdisk -l to verify your work



1. Remove all your partitions with fdisk. Then restore your backups. To backup /dev/sda partition table, enter:



# Conclusion:Thus, we have successfully understood how to partition a disk and also perform various operations on it.