COMMAND	MEANING	EXAMPLE & SYNTAX
`ls (list all directories)`	Lists all the files and directories inside the current directory in which you are.	Syntax: `$ ls`
`ls -R`	Lists all the files and directories inside the current directory as well as all the files and directories of the sub-directories as well.	Syntax: `$ ls -R`
`ls -a`	Lists all the files and directories in the current directory and also lists the hidden files (such as .git files). However, this command does not list the files and directories of the sub-directories.	Syntax: `$ ls -a`
`ls -al`	Lists files and directories of the current directory along with the details like permissions (read, write, execute), owner, file/dir size, etc.	Syntax: `$ ls -al`
`cd`	This command is used to move to the root directory.	Syntax: `$ cd`
`cd ~`	Same function as cd i.e. move to the root/home directory. Please note that there is a space between cd and tilde (~) symbol.	Syntax: `$ cd ~`
`cd ..`	Move to one level up directory.	Syntax: `$ cd ..`
`cd dirName`	Move to a particular directory from the current directory. Note that you can only move down the directory and not to the directories in the above level.	Example: In the command shown on the right, we move from the root directory to Desktop. Syntax: `$ cd Desktop`
`mkdir`	This command creates a directory.	Example: The command shown in the right will create a directory named "exampleDir" in the current directory in which we are. Syntax: `$ mkdir exampleDir`
`cat > fileName`	This command creates a file in the current directory.	Example: The command shown in the right creates a new file in the current directory and the name of the file will be file1 with an extension of '.txt'. Syntax: `$ cat > file1.txt`

COMMAND	MEANING	EXAMPLE & SYNTAX
`cat fileName`	This command displays the content in a file. If a file is not present in the current directory, it gives a message showing no such file exists.	Example: The command shown on the right displays the content of the file file1.txt. “Hello there!” is the content inside it. Syntax: `$ cat file1.txt Hello there!`

`cat f1 f2 > f3`	This command joins the content of two files and stores it in the third file. If the third file does not exist, it is first created and then the joined content is stored.	Example: The command in the right stores the joined content of file1 and file2 in file3. File1 has “Hello there!” and file2 has “What’s up?” in their content. We have displayed the content of file3.
		
		Syntax:
		`$ cat file1.txt  file2.txt > file3.txt`
		`$ cat file3.txt Hello there! What’s up?`

`rmdir dirName`	This command is the remove directory command. It deletes a directory.	Example: The remove directory command for deleting a directory named “exampleDir” is shown on the right. Syntax: `$ rmdir exampleDir`
`mv fileName “new file path”`	This command is the move file command. It moves the file to the new path specified.	Example: The mv command moves the file file1.txt to “Docs” directory. Syntax: `$ mv file1.txt “Docs/”`
`mv fileName newName`	This command changes the name of the file from the old name i.e. the fileName to the newName.	Example: The command in the right changes the

COMMAND	MEANING	EXAMPLE & SYNTAX
		name of the file file1 to file2. Syntax: `$ mv file1.txt file2.txt`
`find <starting position to search> <expression determining what to find> <options> <what to find>`	This command is used for walking a file hierarchy. It is used to find files/directories and perform operations on them. We can search by file, folder, name, creation date, modification date, etc. There are a number of options available. For instance, exec searches the file that meets the criteria and returns 0 as exit status for successful command execution.	Example: The command in the right is for searching a file with the name file1.txt in the Docs directory. Syntax: `$ find ./Docs -name file1.txt`
`grep <options> pattern fileName`	The full form of this command is a global search for regular expression and printout. This command searches a file for a particular pattern of characters and displays all the lines that contain that pattern. The pattern being searched is called a regular expression (regex). There are a lot of <options> available. For instance, c is an option that is used to only count the number of lines in the file that matches the pattern.	Example: The command to count the number of lines that have "abc" in them in the file file1.txt is shown on the right. Syntax: `$ grep  -c "abc" file1.txt`

2. System Information Commands

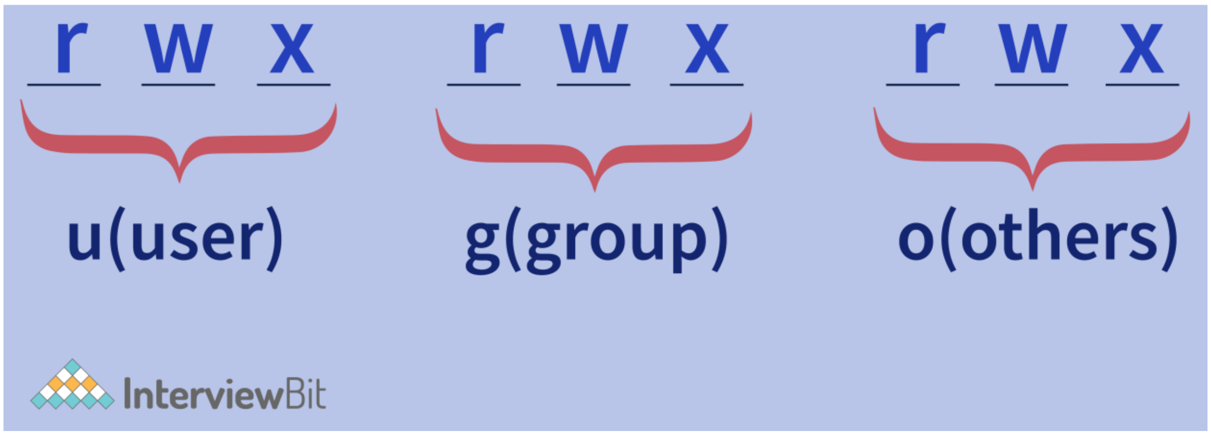
These are some of the general-purpose system information commands that are important to know and easy to remember.

COMMAND	MEANING	EXAMPLE & SYNTAX
`history`	This command displays the list of all the typed commands in the current terminal session.	Syntax: `$ history`
`clear`	Clears the terminal i.e. no previous command will be visible on the screen now.	Syntax: `$ clear`
`hostname`	Shows the name of the system host.	Syntax: `$ hostname`
`hostid`	Displays the id of the host of the system.	Syntax: `$ hostid`
`sudo`	Allows a regular user to run the programs with the security privileges of a superuser or root.	Syntax: `$ sudo`

COMMAND	MEANING	EXAMPLE & SYNTAX
`apt-get`	This command is used to install and add new packages.	Syntax: `$ apt-get`
`date`	This command is used to show the current date and time.	Example: The command and its output are shown on the right. Syntax: `$ date` `Fri Feb 25 14:58:08 IST 2022`
`cal`	Shows the calendar of the current month.	Example: The command cal and its output is shown on the right. Syntax: `$ cal`
`whoami`	This command displays the name with which you are logged in.	Example: The command is typed in and it shows the username with which the user has logged in. Syntax: `$ whoami Guneet Malhotra`
`whereis [options] fileName`	This command is used to find the location of source/binary file of a command and manuals sections for a specified file in Linux System. This command is similar to the find command but this command is faster as it produces more accurate results by taking less time compared to the find command. There are again a number of options available.	Example: The command to locate apropos command in Linux System is given on the right. Syntax: `$ whereis apropos`

3. File Permission Commands

There are 3 types of people who can use a file and each type has 3 types of access to the file. This is shown in the diagram given below:



The diagram shows that there are 3 types of people accessing a file and they are:

1. User (u)
2. Group (g)
3. Others (o)

Also, the access that we want to give to each of them is of three types:

1. Read (r)
2. Write (w)
3. Execute (x)

So, each of them can have 0 or more out of these 3 permissions. Now let us understand the Linux commands that help us give these permissions to the files. One important thing to note here is that before these 9 slots of the user, group and others (read, write and execute permissions), there is also one another slot. This slot is for special files. For instance, if you something as drwxr--r--, here 'd' shows that it is a directory of which you are viewing the permissions. Further, rwx means that the user has all the three permissions where as r-- means that the group has only read permission and the write and execute permissions are not there with the group. The same is the case for others (another r--).

- **The chmod Command:**

Before we jump into the Linux file permission commands and see some examples, it is very important to understand this chmod command in detail first as understanding this command completely will clear the entire concept of file permission commands. The chmod command stands for "change-mode" which means that using this command, we can change the mode in which some user is able to access the file. This command is used to change the file permissions. The syntax can be either using symbols (characters) or numbers. We will see that in detail.

- **Symbolic Method for granting permissions:**

This is the first method of chmod command using which we can give permissions. The basic syntax is as follows:

chmod [ugoa...][-+=]perms...[,....] FILE....

Let us understand this syntax in detail.

The first set means the type of person to give access to. Here:

1. u → Stands for User
2. g → Stands for Group
3. o → Stands for Others
4. a → Stands for All the users i.e. instead of writing ugo, we can just write a.

If the user's flag is not included in the command i.e. we do not mention for which kind of people out of u, g and o, are we changing the permissions for, by default, it takes a i.e. all the users.

The second set is the set of operators. Let us see what they mean.

1. - → removes the mentioned permission
2. + → adds the mentioned permission
3. = → Changes the current permission to the mentioned permission. IF no permission is mentioned after using the = operator, all the permissions from the mentioned class are removed.

The perms stand for permission and ',' is used to separate different permissions. Let us now see the Linux commands using the symbolic notation of chmod.

COMMAND	MEANING	EXAMPLE & SYNTAX
`ls -l fileName`	This command is used to show the file permissions along with the owner and other details of the specified file.	Example: The file permissions along with the owner and other details is shown for the file file1.txt on the right. Syntax: `$ ls -l file1.txt -rw-r--r-- 1 Guneet Malhotra 197121 0 Feb 25 10:51 file1.txt`

COMMAND	MEANING	EXAMPLE & SYNTAX
`r`	This command represents the read permission.	Example: The command shown in the right adds the read permission to the o (other) class for the file file1.txt. Syntax: `$ chmod o+r file1.txt`
`w`	This command represents the write permission.	Example: This commands adds the write permission for a(all) i.e. user, group and others. Syntax: `$ chmod a+w file1.txt`
`x`	This command represents the execute permission.	Example: This command adds the execution permission for the user. Syntax: `$ chmod u+x file1.txt`

- **Numerical Method for granting file permissions**

There are numeric codes for each permission. They are as follows:

1. r (read) = 4
2. w (write) = 2
3. x (execute) = 1
4. No permissions = 0

The permissions number of a specific user class is represented by the sum of the values of all the permissions. For instance, if the user has read and executed permissions, but not the write permission, then the permissions number for the user will be read (4) + execute(1) = 5.

For instance, if we have to write a command to provide read and write permissions to the user, group and others, there can be many ways of doing so. Let us see one symbolic way:

- **Symbolic Way**

```
$ chmod ugo+rw file1.txt
```

We can write this in a numeric way as shown below:

- **Numeric Way**

```
$ chmod 666 file1.txt
```

Explanation: We have already studied that if we do not mention u/g/o then by default the permissions are applied to all. Also, read + write = 4 + 2 = 6. We have written 6 thrice because of applying the permissions to user, group and others. So, read and write permissions are applied to the user, group and others (666) for the file file1.txt.

---

4. Hardware Information Commands

Let us now see, some of the hardware information commands that give us the information about the hardware that we are using.

COMMAND	MEANING	EXAMPLE & SYNTAX
`cpu-info`	This command is used to display the information about your CPU. Note that this command is not available by default. It can be used after installation of the necessary package using **sudo apt install cpuinfo**.	Syntax: `$ cpu-info`
`free -h`	This command is used to display the free and used memory. -h is used for converting the information (to be displayed) to human-readable form.	Syntax: `$ free -h`
`lsusb -tv`	List all the USB connected devices.	Syntax: `$ lsusb -tv`
`cat /proc/meminfo`	Gives the information about memory like total and occupied and so on.	Syntax: `$ cat /proc/meminfo`
`du`	This command stands for disk usage and is used to estimate the space usage for a file or directory.	Example: The following command gives the size in human-readable form for the Desktop folder. Syntax: `$ du -h Desktop`

5. File and Directory Compression Commands

COMMAND	MEANING	EXAMPLE & SYNTAX
`gzip fileName`	This command is used to compress a file with gzip compression.	Example: The command to zip file1 using gzip compression is shown on the right. Syntax: `$ gzip file1`
`gunzip fileName.gz`	This command is used to unzip a file that has gzip compression.	Example: The command to unzip fileDemo.gz file with gz compression is shown on the right. Syntax: `$ gunzip fileDemo.gz`
`tar cf myDir.tar myDir`	This command is used to create an uncompressed tar archive.	Example: The command to create an uncompressed tar archive for the directory demoDir is shown on the right. Syntax: `$ tar cf demoDir.tar demoDir`
`tar cfz myDir.tar myDir`	This command is used to create a tar archive with gzip compression.	Example: The command to create gzip tar archive for the directory demoDir is shown on the right. Syntax: `$ tar cfz demoDir.tar demoDir`
`tar xf file`	This command is used to extract the contents of any type of tar archive.	Example: The command to extract the content of demoFile tar archive is shown on the right. Syntax: `$ tar xf demoFile`

6. Environment Variable Commands

COMMAND	MEANING	EXAMPLE & SYNTAX
`env`	This command displays all the environment variables.	Syntax: `$ env`
`echo $Variable`	This command displays the environment variable.	Example: The command at the right will display the INSTANCE environment variable. Syntax: `$ echo $INSTANCE=`
`unset`	This command removes a variable.	Syntax: `$ unset`

7. User Management Commands

COMMAND	MEANING	EXAMPLE & SYNTAX
`sudo adduser username`	This command is used to add a user.	Syntax: `$ sudo adduser username`
`sudo passwd -l 'username'`	This command is used to change the password of a user.	Example: Command to change the password for user1 is shown Syntax: `$ sudo passwd -l 'user1'`
`sudo userdel -r 'username'`	This command is used to remove a newly created user.	Example: Command to delete the newly created user1 Syntax: `$ sudo userdel -r 'user1'`
`sudo usermod -a -G GROUPNAME USERNAME`	This command is used to add a user to a particular group.	Example: The command to add user2 to group1 is shown. Syntax: `$ sudo usermod -a -G group1 user2`
`Sudo deluser USER GROUPNAME`	This command is used to remove a user from a group.	Example: The command to delete user1 from group1 is shown. Syntax: `$ sudo deluser user1 group1`
`finger`	This command shows the information of all the users logged in.	Syntax: `$ finger`
`finger username`	This command gives information about a particular user.	Example: The command to get information about the user1 is shown on the right. Syntax: `$ finger user1`

8. Networking Commands

COMMAND	MEANING	SYNTAX
`dir`	This command is used to display files in the current directory of a remote computer.	Syntax: `$ dir`
`put file`	This command is used to upload 'file' from local to the remote computer.	Syntax: `$ put file`
`get file`	This file is used to download 'file' from remote to the local computer.	Syntax: `$ get file`
`quit`	This command is used to log out.	Syntax: `$ quit`

9. Process Commands

COMMAND	MEANING	EXAMPLE & SYNTAX
`bg`	This command is used to send a process to the background.	Example: The process with id 1 is sent to the background by providing its id to bg. Syntax: `$ bg %1`
`fg`	This command is used to run a stopped process in the background.	Example: The process with id 1 is brought to the foreground with the help of this command. Syntax: `$ fg %1`
`top`	This command is used to get the details of all active processes.	Syntax: `$ top`
`ps`	This command is used to give the status of running for a user.	Syntax: `$ ps`
`ps PID`	This command gives the status of a particular process.	Example: Displays the status of the process with id 12230. Syntax: `$ ps 12230`
`pidof`	This command is used to give the process ID of a particular process.	Syntax: `$ pidof bash`