**File Permissions**

Linux is a clone of UNIX, the **multi-user operating system**which can be accessed by many users simultaneously. Linux can also be used in mainframes and servers without any modifications. But this raises security concerns as an unsolicited or **malign user** can **corrupt, change or remove crucial data**. For effective security, Linux divides authorization into 2 levels.

1. Ownership
2. Permission

In this Linux file commands tutorial, you will learn-

* [Ownership of Linux files](https://www.guru99.com/file-permissions.html#1)
* [Permissions](https://www.guru99.com/file-permissions.html#2)
* [Changing file/directory permissions with 'chmod' command](https://www.guru99.com/file-permissions.html#3)
* [Absolute(Numeric) Mode](https://www.guru99.com/file-permissions.html#4)
* [Symbolic Mode](https://www.guru99.com/file-permissions.html#5)
* [Changing Ownership and Group](https://www.guru99.com/file-permissions.html#6)
* [Summary](https://www.guru99.com/file-permissions.html#7)

## Ownership of Linux files

Every file and directory on your Unix/Linux system is assigned 3 types of owner, given below.

### User

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

### Group

A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

### Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

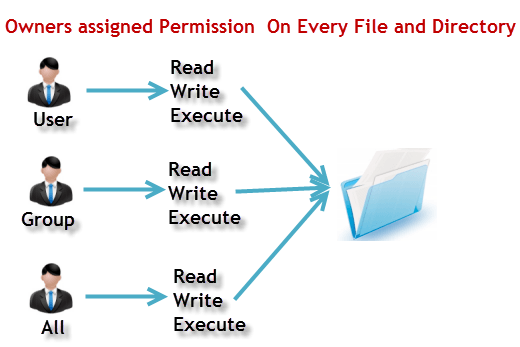
Now, the big question arises how does **Linux distinguish**between these three user types so that a user 'A' cannot affect a file which contains some other user 'B's' vital information/data. It is like you do not want your colleague, who works on your Linux computer, to view your images. This is where **Permissions** set in, and they define **user behavior**.

Let us understand the **Permission system** on Linux.

## Permissions

Every file and directory in your UNIX/Linux system has following 3 permissions defined for all the 3 owners discussed above.

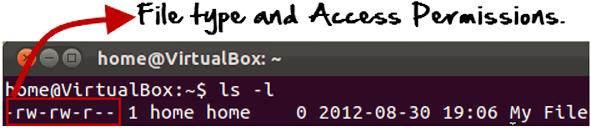
* **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
* **Write:**The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.
* **Execute:**In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.

[](https://www.guru99.com/images/PermissionsConcept.png)

**Let's see this in action**

**ls - l** on terminal gives

ls - l

[](https://www.guru99.com/images/Permis_system.png)

Here, we have highlighted **'-rw-rw-r--'**and this weird looking code is the one that tells us about the permissions given to the owner, user group and the world.

Here, the first '**-**' implies that we have selected a file.p>

[](https://www.guru99.com/images/its_a_file.png)

Else, if it were a directory, **d**would have been shown.

[File Permissions in Linux/Unix](https://www.guru99.com/images/Directory.png)

The characters are pretty easy to remember.

**r** = read permission  
**w** = write permission  
**x** = execute permission  
**-** = no permission

Let us look at it this way.

The first part of the code is **'rw-'**. This suggests that the owner 'Home' can:

[](https://www.guru99.com/images/no_execute.png)

* Read the file
* Write or edit the file
* He cannot execute the file since the execute bit is set to '-'.

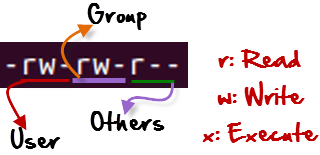
By design, many Linux distributions like Fedora, CentOS, Ubuntu, etc. will add users to a group of the same group name as the user name. Thus, a user 'tom' is added to a group named 'tom'.

The second part is **'rw-'.** It for the user group 'Home' and group-members can:

* Read the file
* Write or edit the file

The third part is for the world which means any user. It says **'r--'.** This means the user can only:

* Read the file

[](https://www.guru99.com/images/permission(1).png)

## Changing file/directory permissions with 'chmod' command

Say you do not want your colleague to see your personal images. This can be achieved by changing file permissions.

We can use the '**chmod'** command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world. **Syntax:**

chmod permissions filename

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

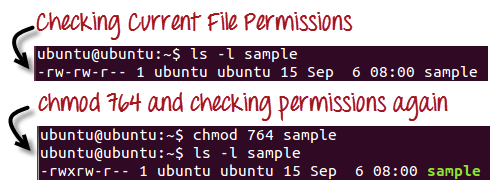
## Absolute(Numeric) Mode

In this mode, file **permissions are not represented as characters but a three-digit octal number**.

The table below gives numbers for all for permissions types.

|  |  |  |
| --- | --- | --- |
| **Number** | **Permission Type** | **Symbol** |
| 0 | No Permission | --- |
| 1 | Execute | --x |
| 2 | Write | -w- |
| 3 | Execute + Write | -wx |
| 4 | Read | r-- |
| 5 | Read + Execute | r-x |
| 6 | Read +Write | rw- |
| 7 | Read + Write +Execute | rwx |

Let's see the chmod command in action.

[](https://www.guru99.com/images/chmod_new(1).png)

In the above-given terminal window, we have changed the permissions of the file 'sample to '764'.

'764' absolute code says the following:

* Owner can read, write and execute
* Usergroup can read and write
* World can only read

**This is shown as '-rwxrw-r-**

This is how you can change the permissions on file by assigning an absolute number.

## Symbolic Mode

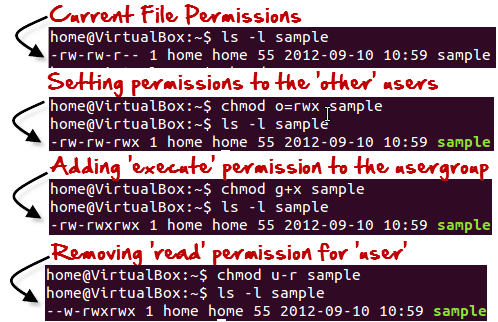
In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **+** | Adds a permission to a file or directory |
| **-** | Removes the permission |
| **=** | Sets the permission and overrides the permissions set earlier. |

The various owners are represented as -

|  |  |
| --- | --- |
| **User Denotations** | |
| u | user/owner |
| g | group |
| o | other |
| a | all |

We will not be using permissions in numbers like 755 but characters like rwx. Let's look into an example

[](https://www.guru99.com/images/Symbolic_Mode(1).png)

## Changing Ownership and Group

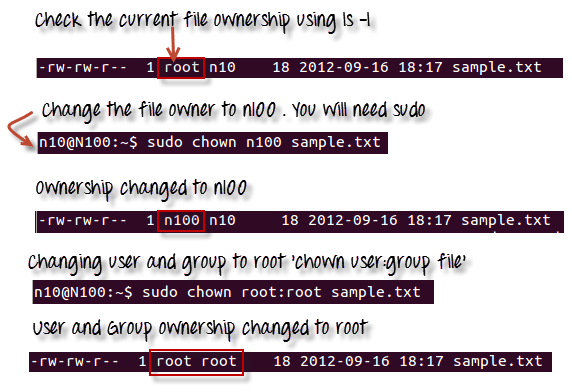
For changing the ownership of a file/directory, you can use the following command:

chown user

In case you want to change the user as well as group for a file or directory use the command

chown user:group filename

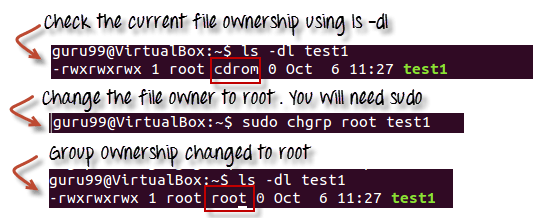
Let's see this in action

[](https://www.guru99.com/images/chown_comm(1).png)

In case you want to change group-owner only, use the command

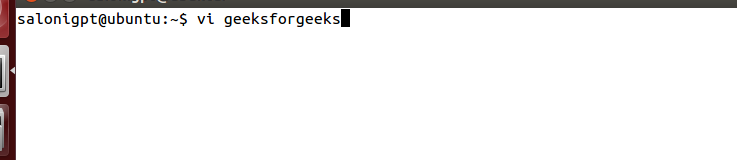
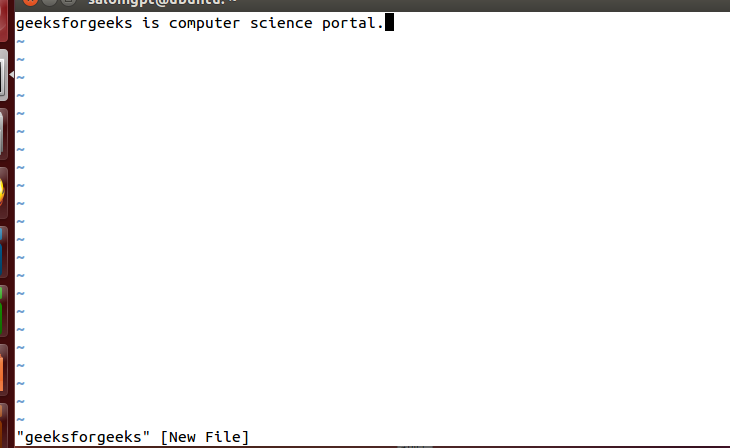
chgrp group\_name filename

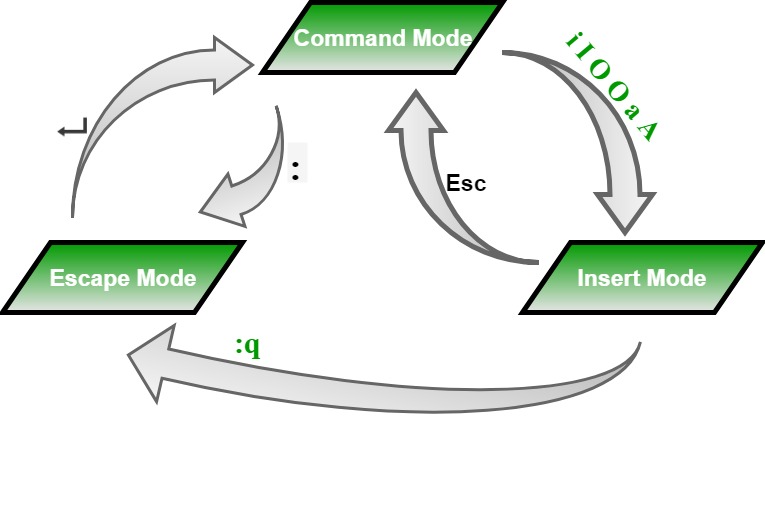
'**chgrp'** stands for change group.

[](https://www.guru99.com/images/chgrp.png)

**VI Editor in Linux**

vi filename

**Input:**  
  
**Output:** 

**Modes of Operation in vi editor**There are three modes of operation in vi:  


* **Command Mode:** When vi starts up, it is in Command Mode. This mode is where vi interprets any characters we type as commands and thus does not display them in the window. This mode allows us to move through a file, and to delete, copy, or paste a piece of text.  
  To enter into Command Mode from any other mode, it requires pressing the **[Esc]** key. If we press [Esc] when we are already in Command Mode, then vi will beep or flash the screen.
* **Insert mode:**This mode enables you to insert text into the file. Everything that’s typed in this mode is interpreted as input and finally, it is put in the file. The vi always starts in command mode. To enter text, you must be in insert mode. To come in insert mode you simply type i. To get out of insert mode, press the Esc key, which will put you back into command mode.
* **Last Line Mode(Escape Mode):** Line Mode is invoked by typing a colon [:], while vi is in Command Mode. The cursor will jump to the last line of the screen and vi will wait for a command. This mode enables you to perform tasks such as saving files, executing commands.
* ***vi filename:****Creates a new file if it already not exist, otherwise opens existing file.*
* ***vi -R filename****: Opens an existing file in read only mode.*
* ***view filename****: Opens an existing file in read only mode.*

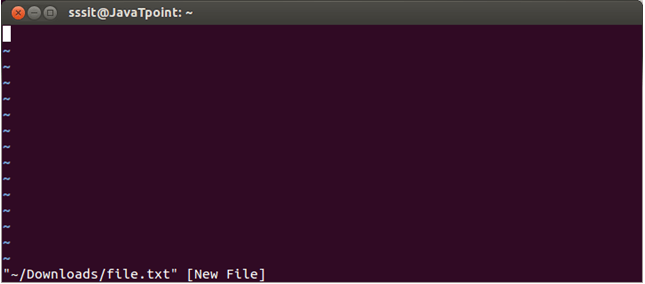
**Control Commands(Scrolling):** There are following useful commands which can used along with **Control Key**:

***Commands and their Description:***

* ***CTRL+d****: Move forward 1/2 screen.*
* ***CTRL+f****: Move forward one full screen.*
* ***CTRL+u****: Move backward 1/2 screen.*
* ***CTRL+b****: Move backward one full screen.*
* ***CTRL+e****: Moves screen up one line.*
* ***CTRL+y****: Moves screen down one line.*
* ***CTRL+u****: Moves screen up 1/2 page.*
* ***CTRL+d****: Moves screen down 1/2 page.*
* ***CTRL+b****: Moves screen up one page.*
* ***CTRL+f****: Moves screen down one page.*
* ***CTRL+I****: Redraws screen.*

**Command mode**

This is what you'll see when you'll press enter after the above command. If you'll start typing, nothing will appear as you are in command mode. By default vi opens in command mode.



Look at the above snapshot, it is blank as it is a new file. To start typing, you have to move to the insert mode. At the end of the terminal window, directory name and file name are displayed.

Insert mode

To move to the insert mode press **i.** Although, there are other commands also to move to insert mode which we'll study in next page.

## Linux Vi Editor3To save and quit

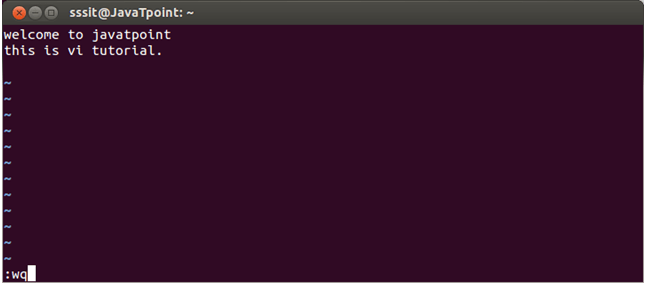
You can save and quit vi editor from command mode. Before writing save or quit command you have to press colon **(:).** Colon allows you to give instructions to vi.

**exit vi table:**

|  |  |
| --- | --- |
| **Commands** | **Action** |
| :wq | Save and quit |
| :w | Save |
| :q | Quit |
| :w fname | Save as fname |
| ZZ | Save and quit |
| :q! | Quit discarding changes made |
| :w! | Save (and write to non-writable file) |

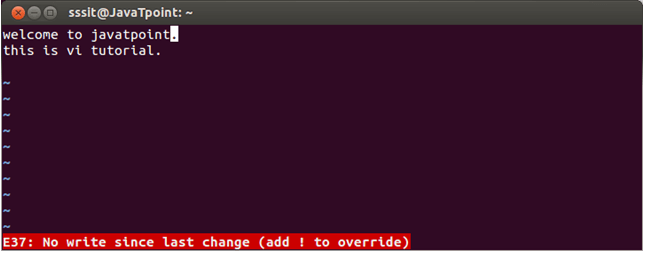
To exit from vi, first ensure that you are in command mode. Now, type :wq and press enter. It will save and quit vi.

Type **:wq** to save and exit the file.



Look at the above snapshot, command :wq will save and quit the vi editor. When you'll type it in command mode, it will automatically come at bottom left corner.

If you want to quit without saving the file, use **:q.** This command will only work when you have not made any changes in the file.



Look at the above snapshot, this file is modified and hence on typing **:q** it displays this message at bottom left corner.

The above file can be saved with the command **:!q.** It discards the changes made in the file and save it.



Look at the above snapshot, we have typed **:!q,** it will save our file by discarding the changes made.

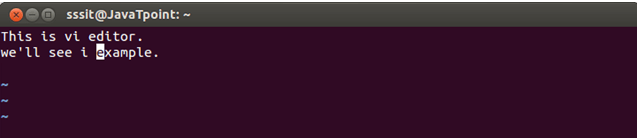
# vi switch from command mode

To start typing in vi editor, you have to be in insert mode. There are different commands in vi editor through which you can start typing at different locations in insert mode. They are defined below.

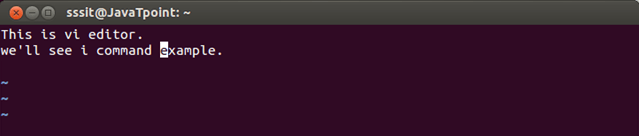
**i**

The **i**command allows us to write **before the current character.**

**Example:**



Look at the above snapshot, our cursor is at **e.** We forgot to write word **command**before word **example.** Hence, here we can use i command.

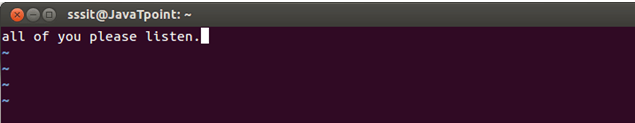


Look at the above snapshot, after pressing **i,** we wrote word command.

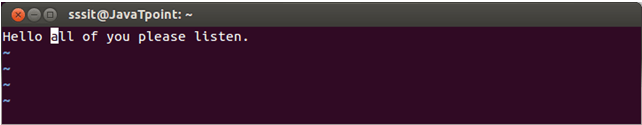
**I**

The command **I**allows us to write at the **start of the current line.**

**Example:**



Look at the above snapshot, we want to write word **Hello**at the starting of the line.

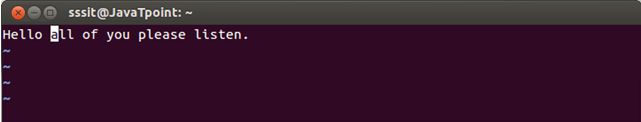


Look at the above snapshot, after pressing I, we have typed **Hello**at the starting of the line.

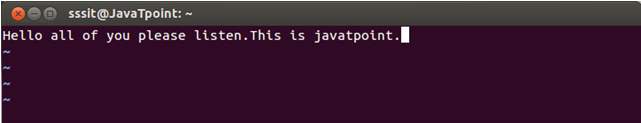
**A**

The A command allows us to start typing at the **end of the current line.**

**Example:**



Look at the above snapshot, we want to move at the end of the current line.

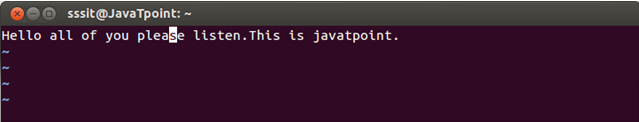


Look at the above snapshot, after pressing A, we have written one more line.

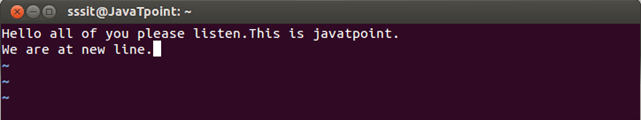
**o**

The **o**command allows us to start typing a **new line after the current line.**

**Example:**



Look at the above snapshot, we are at the middle of the line and want to start a new line.

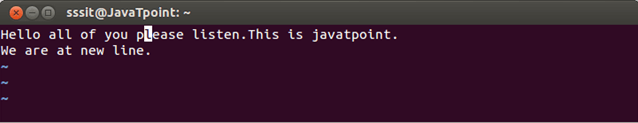


Look at the above snapshot, after pressing o, we arrived at next line and typed there.

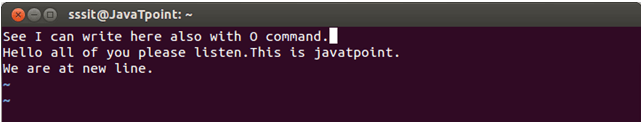
**O**

The **O**command allows us to start typing on a **new line before the current line.**

**Example:**



Look at the above snapshot, we are at the middle of the line and want to write before the current line.



Look at the above snapshot, after pressing O, we have written in a new line before the current line.

**What is the VI editor?**

The VI editor is the most popular and classic text editor in the Linux family. Below, are some reasons which make it a widely used editor –

1) It is available in almost all Linux Distributions

2) It works the same across different platforms and Distributions

3) It is user-friendly. Hence, millions of Linux users love it and use it for their editing needs

Nowadays, there are advanced versions of the vi editor available, and the most popular one is **VIM**which is **V**i **Im**proved. Some of the other ones are Elvis, Nvi, Nano, and Vile. It is wise to learn vi because it is feature-rich and offers endless possibilities to edit a file.

To work on VI editor, you need to understand **its operation modes**. They can be divided into two main parts.

In this tutorial, you will learn more about-

* [vi Command mode](https://www.guru99.com/the-vi-editor.html#2)
* [vi Editor Insert mode](https://www.guru99.com/the-vi-editor.html#3)
* [How to use vi editor](https://www.guru99.com/the-vi-editor.html#4)
* [vi Editing commands](https://www.guru99.com/the-vi-editor.html#5)
* [Moving within a file](https://www.guru99.com/the-vi-editor.html#6)
* [Saving and Closing the file](https://www.guru99.com/the-vi-editor.html#7)
* The vi editor opens in this mode, and it only **understands commands**
* In this mode, you can, **move the cursor and cut, copy, paste the text**
* This mode also saves the changes you have made to the file
* **Commands are case sensitive.** You should use the right letter case.

**vi Editor Insert mode:**

* This mode is for inserting text in the file.
* You can switch to the Insert mode from the command mode **by pressing 'i' on the keyboard**
* Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working.
* To return to the command mode and save the changes you have made you need to press the Esc key

**How to use vi editor**

To launch the VI Editor -Open the Terminal (CLI) and type

vi <filename\_NEW> or <filename\_EXISTING>

And if you specify an existing file, then the editor would open it for you to edit. Else, you can create a new file.

**VI E diting commands**

* i - Insert at cursor (goes into insert mode)
* a - Write after cursor (goes into insert mode)
* A - Write at the end of line (goes into insert mode)
* ESC - Terminate insert mode
* u - Undo last change
* U - Undo all changes to the entire line
* o - Open a new line (goes into insert mode)
* dd - Delete line
* 3dd - Delete 3 lines.
* D - Delete contents of line after the cursor
* C - Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
* dw - Delete word
* 4dw - Delete 4 words
* cw - Change word
* x - Delete character at the cursor
* r - Replace character
* R - Overwrite characters from cursor onward
* s - Substitute one character under cursor continue to insert
* S - Substitute entire line and begin to insert at the beginning of the line
* ~ - Change case of individual character

**Moving within a file**

* k - Move cursor up
* j - Move cursor down
* h - Move cursor left
* l - Move cursor right

You need to be in the command mode to move within a file. The default keys for navigation are mentioned below else; You can **also use the arrow keys on the keyboard**.

**Saving and Closing the file**

* Shift+zz - Save the file and quit
* :w - Save the file but keep it open
* :q - Quit without saving
* :wq - Save the file and quit
* The vi editor is the most popular and commonly used Unix text editor
* It is usually available in all Linux Distributions.
* It works in two modes, Command and Insert
* Command mode takes the user commands, and the Insert mode is for editing text
* You should know the commands to work on your file easily
* Learning to use this editor can benefit you in creating scripts and editing files.

**WILD CARDS**

**Wildcards** (also referred to as meta characters) are symbols or special characters that represent other characters. You can use them with any command such as [ls command](https://www.tecmint.com/tag/linux-ls-command/" \t "_blank) or **rm command** to list or remove files matching a given criteria, receptively.

Read Also: [10 Useful Practical Examples on Chaining Operators in Linux](https://www.tecmint.com/chaining-operators-in-linux-with-practical-examples/)

These wildcards are interpreted by the shell and the results are returned to the command you run. There are three main wildcards in Linux:

* An asterisk (\*) – matches one or more occurrences of any character, including no character.
* Question mark (?) – represents or matches a single occurrence of any character.
* Bracketed characters ([ ]) – matches any occurrence of character enclosed in the square brackets. It is possible to use different types of characters (alphanumeric characters): numbers, letters, other special characters etc.

You need to carefully choose which wildcard to use to match correct filenames: it is also possible to combine all of them in one operation as explained in the examples below.

### How to Match Filenames Using Wildcards in Linux

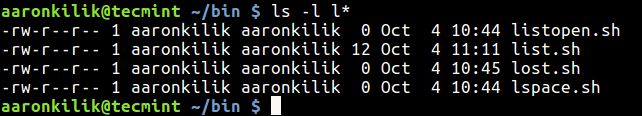
For the purpose of this article, we will use following files to demonstrate each example.

createbackup.sh list.sh lspace.sh speaker.sh

listopen.sh lost.sh rename-files.sh topprocs.sh

**1.** This command matches all files with names starting with l (which is the prefix) and ending with one or more occurrences of any character.

$ ls -l l\*

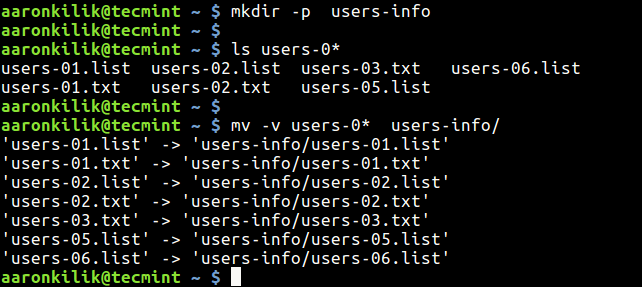
[](https://www.tecmint.com/wp-content/uploads/2017/10/List-Files-with-Character.png)List Files with Character

**2.** This example shows another use of \* to copy all filenames prefixed with users-0 and ending with one or more occurrences of any character.

$ mkdir -p users-info

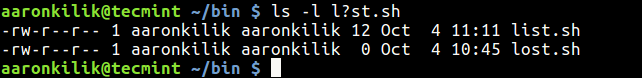
$ ls users-0\*

$ mv -v users-0\* users-info/ # Option **-v** flag enables verbose output

[](https://www.tecmint.com/wp-content/uploads/2017/10/List-and-Copy-All-Files.png)List and Copy All Files

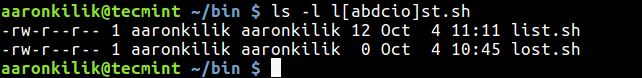
**3.** The following command matches all files with names beginning with l followed by any single character and ending with st.sh (which is the suffix).

$ ls l?st.sh

[](https://www.tecmint.com/wp-content/uploads/2017/10/Match-File-with-Character-Name.png)Match File with Character Name

**4.** The command below matches all files with names starting with l followed by any of the characters in the square bracket but ending with st.sh.

$ ls l[abdcio]st.sh

[](https://www.tecmint.com/wp-content/uploads/2017/10/Matching-Files-with-Names.png)Matching Files with Names

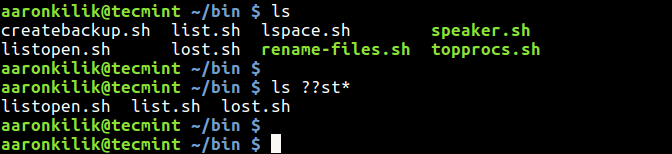
### How to Combine Wildcards to Match Filenames in Linux

You can combine wildcards to build a complex filename matching criteria as described in the following examples.

**5.** This command will match all filenames prefixed with any two characters followed by st but ending with one or more occurrence of any character.

$ ls

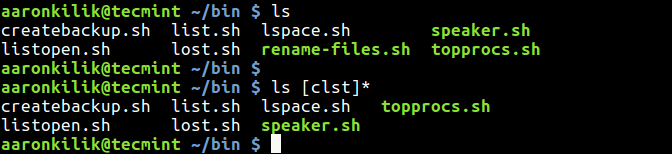
$ ls ??st\*

[](https://www.tecmint.com/wp-content/uploads/2017/10/Find-File-Names-with-Prefix.png)Match File Names with Prefix

**6.** This example matches filenames starting with any of these characters [clst] and ending with one or more occurrence of any character.

$ ls

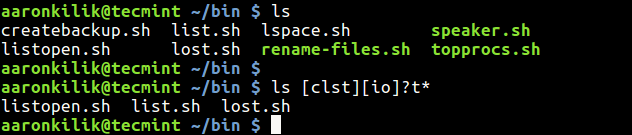
$ ls [clst]\*

[](https://www.tecmint.com/wp-content/uploads/2017/10/Match-Files-with-Characters.png)Match Files with Characters

**7.** In this examples, only filenames starting with any of these characters [clst] followed by one of these [io] and then any single character, followed by a t and lastly, one or more occurrence of any character will be listed.

$ ls

$ ls [clst][io]?t\*

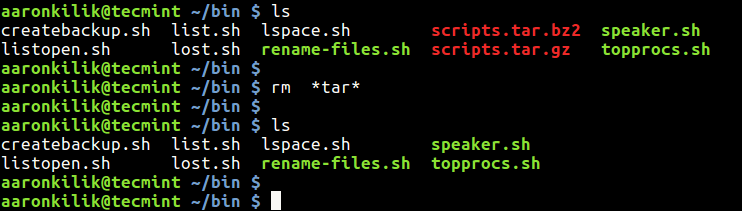
[](https://www.tecmint.com/wp-content/uploads/2017/10/List-Files-with-Multiple-Characters.png)List Files with Multiple Characters

**8.** Here, filenames prefixed with one or more occurrence of any character, followed by the letters tar and ending with one or more occurrence of any character will be removed.

$ ls

$ rm \*tar\*

$ ls

[](https://www.tecmint.com/wp-content/uploads/2017/10/Remove-Files-with-Letters.png)Remove Files with Character Letters

### How to Match Characters Set in Linux

**9.** Now lets look at how to specify a set of characters. Consider the filenames below containing system users information.

$ ls

users-111.list users-1AA.list users-22A.list users-2aB.txt users-2ba.txt

users-111.txt users-1AA.txt users-22A.txt users-2AB.txt users-2bA.txt

users-11A.txt users-1AB.list users-2aA.txt users-2ba.list

users-12A.txt users-1AB.txt users-2AB.list users-2bA.list

This command will match all files whose name starts with users-i, followed by a number, a lower case letter or number, then a number and ends with one or more occurrences of any character.

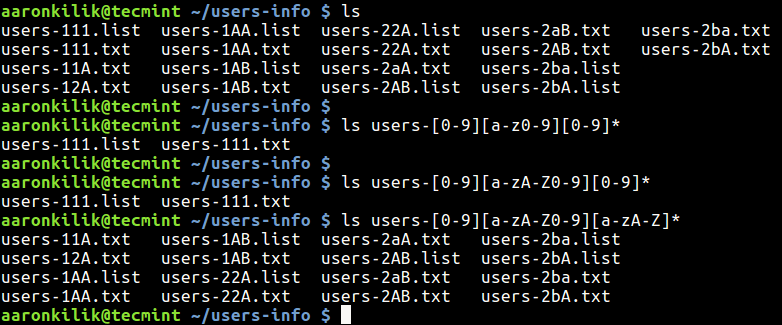
$ ls users-[0-9][a-z0-9][0-9]\*

The next command matches filenames beginning with users-i, followed by a number, a lower or upper case letter or number, then a number and ends with one or more occurrences of any character.

$ ls users-[0-9][a-zA-Z0-9][0-9]\*

This command that follows will match all filenames beginning with users-i, followed by a number, a lower or upper case letter or number, then a lower or upper case letter and ends with one or more occurrences of any character.

$ ls users-[0-9][a-zA-Z0-9][a-zA-Z]\*

[](https://www.tecmint.com/wp-content/uploads/2017/10/Match-Characters-in-Filenames.png)Match Characters in Filenames

### How to Negate a Set of Characters in Linux

**10.** You can as well negate a set of characters using the ! symbol. The following command lists all filenames starting with users-i, followed by a number, any valid file naming character apart from a number, then a lower or upper case letter and ends with one or more occurrences of any character.

$ ls users-[0-9][!0-9][a-zA-Z]\*

That’s all for now! If you have tried out the above examples, you should now have a good understanding of how wildcards work to match filenames in Linux.

You might also like to read these following articles that shows examples of using wildcards in Linux:

1. [How to Extract Tar Files to Specific or Different Directory in Linux](https://www.tecmint.com/extract-tar-files-to-specific-or-different-directory-in-linux/)
2. [3 Ways to Delete All Files in a Directory Except One or Few Files with Extensions](https://www.tecmint.com/delete-all-files-in-directory-except-one-few-file-extensions/)
3. [10 Useful Tips for Writing Effective Bash Scripts in Linux](https://www.tecmint.com/useful-tips-for-writing-bash-scripts-in-linux/)
4. [How to Use Awk and Regular Expressions to Filter Text or String in Files](https://www.tecmint.com/use-linux-awk-command-to-filter-text-string-in-files/)

If you have any thing to share or a question(s) to ask, use the comment form below.

**Link concept in linux**

# Linux File Links

A Linux filesystem has many hard links and symbolic links. A link is a connectivity between the filename and the actual data byte in the disk space. More than one filename can **link** to the same data.

There are two types of links in Linux OS:

1. Hard Links
2. Soft Links

## 1) Hard Links

They are the low-level links. It links more than one filename with the same Inode and it represents the physical location of a file.

When hard link is created for a file, it directly points to the Inode of the original file in the disk space, which means no new Inode is created. Directories are not created using hard links and they can not cross filesystem boundaries. When the source file is removed or moved, then hard links are not affected.

## 2) Soft Links (Symbolic Links)

Soft links are very common. It represents a virtual or abstract location of the file. It is just like the shortcuts created in Windows. A soft link doesn't contain any information or content of the linked file, instead it has a pointer to the location of the linked file. In other words, a new file is created with new Inode, having a pointer to the Inode location of the original file.

It is used to create link between directories and can cross filesystem boundaries. When the source file is removed or moved, then soft links are not updated.

We'll study in deep about both the links how to create it and remove it.

## Linux File Link Topics:

* [Inodes](https://www.javatpoint.com/linux-inodes)
* [Directories](https://www.javatpoint.com/linux-file-links-directories)
* [Hard links](https://www.javatpoint.com/linux-hard-links)
* [Soft links](https://www.javatpoint.com/linux-symbolic-links)
* [To remove a link](https://www.javatpoint.com/linux-removing-links)

# Linux Inodes

An Inode number is a uniquely existing number for all the files in Linux and all Unix type systems.

When a file is created on a system, a file name and Inode number is assigned to it.

Generally, to access a file, a user uses the file name but internally file name is first mapped with respective Inode number stored in a table.

## Inode Contents

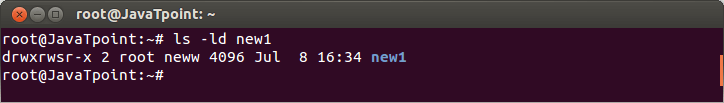
An Inode is a data structure containing metadata about the files.

Following contents are stored in the Inode from a file:

* User ID of file
* Group ID of file
* Device ID
* File size
* Date of creation
* Permission
* Owner of the file
* File protection flag
* Link counter to determine number of hard links

**Example:**

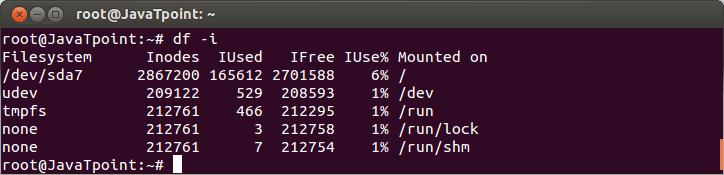
1. ls -ld new1



Look at the above snapshot, it shows some of the Inode contents.

## Inode Table

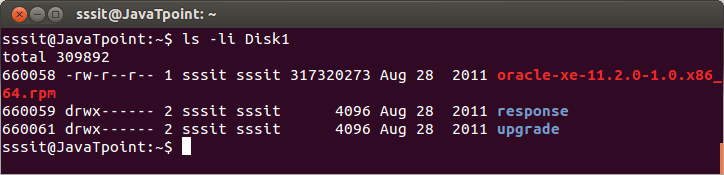
The Inode table contains all the Inodes and is created when file system is created. The **df -i**command can be used to check how many inodes are free and left unused in the filesystem.



Look at the above snapshot, the command **"df -i"** shows the usage of several file systems.

## Inode Number

Each Inode has a unique number and Inode number can be seen with the help of**ls -li**command.



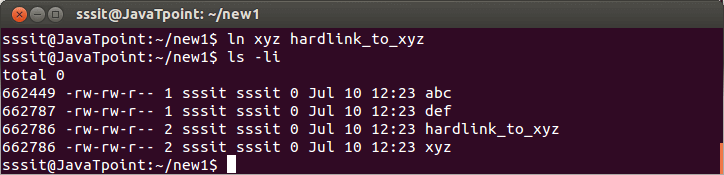
Look at the above snapshot, Directory**Disk1** has the three files and each file has a different Inode number.

**Note:** The Inode doesn't contain file content, instead it has a pointer to that data.

# Hard Links

## Creating Hard Links

Hard links for any file can be created with command **ln**. One extra hard link file will be created in the respective directory.



Look at the above snapshot, we have created a hard link for the file **xyz** in the directory **new1.**

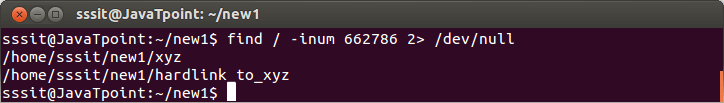
The original file and hard linked file both contain the same Inode number and hence, they have the same permissions and same owners. Content will also be the same for both the files. In short, both the files are equal now, but if original file will be removed then hard link file will not be affected.

## Finding Hard Links

A hard link can be find with **find**command by specifying the Inode number. Inode number is always unique to its partition.

**Example:**

1. find / -inum 662786 2**>** /dev/null



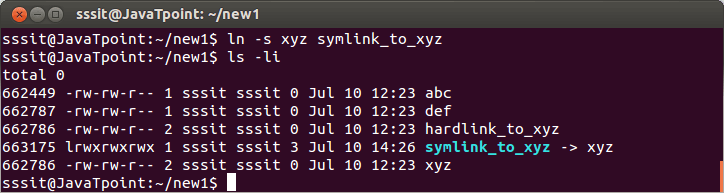
Look at the above snapshot, we have found hard link files with command **"find / -inum 662786 2> /dev/null"** for the Inode number**662786.**

# Symbolic Links

Symbolic links are also called**soft links.** Command **ln -s** is used to create soft link. It doesn't link to Inodes but create a name to mapping. It create its own Inode number.

**Example:**

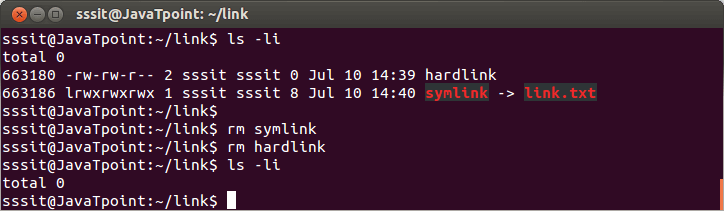
1. ln -s xyz symlink\_to\_xyz



Look at the above snapshot, we have created a symbolic link for file **xyz** with command **"ln -s xyz symlink\_to\_xyz"**. Symbolic link Inode is different from the original file Inode number. Target permissions are applied on the symlink file. Hard links are limited to their own partition, but symbolic links can be linked anywhere.

# Removing Links

With**rm** command links can be removed.



Look at the above snapshot, directory **link** contains both hard link and soft link. With the command **rm** we have removed both the links.

# 

# 