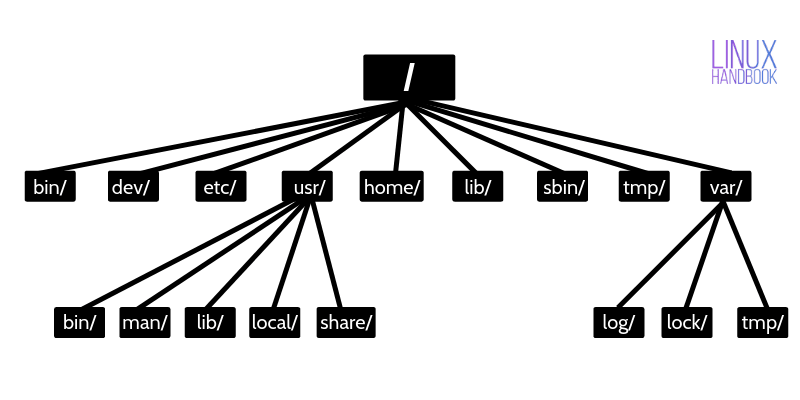
**Managing Files**

## **Filesystem**



## **Absolute paths vs relative paths**

* An obsolute path always begin with a “/”. This indicates that the path starts at the root directory . It’s a fully qualified name.

Example : cd /var/log/samba

* A relative path does not begin with “/”/ It identifies a location relative to your current position.

Example : cd log

cd samba

## **PWD [present working directory]**

* $pwd

## **Change Directory [cd]**

SYNTAX: $ cd [directory]

$ cd or $ cd ~ #Return to users current directory

$ cd - #Switch back to previous directory

$ cd .. #Move to the Parent directory of current directory

$ cd ../ or cd .. #one level back>

$ cd ../../ #two level back>

$ cd ../../ #three level back>

If the directory you want to change to has spaces in its name, you should either surround the path with quotes or use the backslash (\) character to escape the space:

$ cd 'Dir name with space'

$ cd Dir\ name\ with\ space

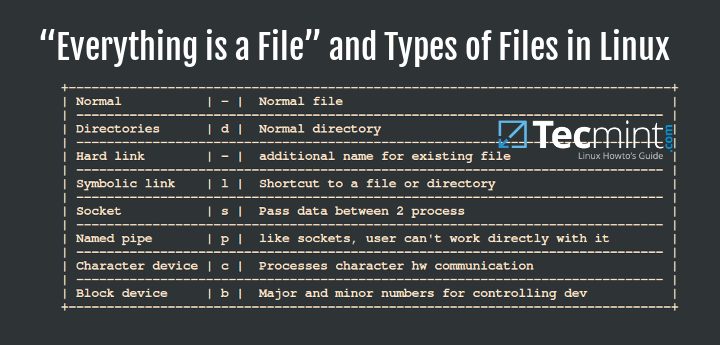
## **Listing Files/Directories**

SYNTAX : $ ls [options] [file|dir]

**-rwxr-xr-x. 1 venkat venkat 100 Sep 19 20:39 hello.sh**

|  |  |
| --- | --- |
| ls | list files & directories in bare format |
| [ls -l](https://www.rapidtables.com/code/linux/ls/ls-l.html) or ll | list with long format - show permissions |
| [ls -a](https://www.rapidtables.com/code/linux/ls/ls-a.html) | list all files including hidden file starting with '.' |
| ls -ld \* | list directories themsleves |
| ls -F | List & will add the **‘/’** Character at the end each directory. |
| ls -i | list file's inode index number |
| ls -I | Capital I [Ignore [Do not list the pattern ] ls -I rhca\* |
| [ls -la](https://www.rapidtables.com/code/linux/ls/ls-l.html) | list long format including hidden files |
| [ls -lh](https://www.rapidtables.com/code/linux/ls/ls-l.html) | list long format with readable file size |
| [ls -ls](https://www.rapidtables.com/code/linux/ls/ls-l.html) | list with long format with file size |
| [ls -r](https://www.rapidtables.com/code/linux/ls/ls-r.html#reverse) | list in reverse order |
| [ls -R](https://www.rapidtables.com/code/linux/ls/ls-r.html#recursive) | list recursively directory tree |
| [ls -s](https://www.rapidtables.com/code/linux/ls/ls-s.html#size) | list file size |
| [ls -S](https://www.rapidtables.com/code/linux/ls/ls-s.html#sort-size) | sort by file size |
| [ls -t](https://www.rapidtables.com/code/linux/ls/ls-t.html) | sort by time & date |
| ls -X | sort by extension name |
| ls -n | display **UID** and **GID** of files and directories |
|  |  |

## Linux File Types



$ ls -l | grep ^- Lists regular files

$ ls -l grep ^d Lists directories

Block files : These are device files that provide buffered access to system hardware components. They provide a method of communication with device drivers through the file system. One important aspect about block files is that they can transfer a large block of data and information at a given time.

Usaully it will be in /dev

$ ls -l /dev | grep ^b

Character files : These are also device files that provide unbuffered serial access to system hardware components. They work by providing a way of communication with devices by transferring data one character at a time.

Usaully it will be in /dev

$ ls -l /dev | grep ^c

Pipes or Named pipes : These are files that allow inter-process communication by connecting the output of one process to the input of another.

A named pipe is actually a file that is used by two process to communicate with each and it acts as a Linux pipe

$ ls -l /dev | grep ^p

Socket files : These are files that provide a means of inter-process communication, but they can transfer data and information between process running on different environments.

This means that sockets provide data and information transfer between process running on different machines on a network.

$ ls -l /dev | grep ^s

## **What is Inode Number ?**

* A Linux consider everything a file including hardware devices, printers, directories, and processes. Any text, music, video, image or any multimedia file is considered a regular file. All the regular file contains metadata about the file which describe everything about the file like, type, Inode etc.
* An inode is an entry in Inode table, containing information (the metadata) about a regular file and directory.
* Inode number is also known as index number. An inode is a unique number assigned to files and directories while it is created. The inode number will be unique to entire filesystem.
* Each inode has an inode number, which is unique within a file system. The same inode number might appear in more than one file system

**Copy file**: cp allocates a free inode number and placing a new entry in inode table.

**Move or Rename a file**: if destination is same filesystem as the source, Has no impact on inode number, it only changes the time stamps in inode table.

**Delete a file**: Deleting a file in Linux decrements the link count and freeing the inode number to be reused.

In the case of inodes are full. You need to remove unused files from the filesystem to make Inode free. There is no option to increase/decrease inodes on disk. It only created during the creation of filesystem on any disk.

Inode number also called index number, it consists following attributes of any file:

* File types ( executable, block special etc )
* Permissions ( read, write etc )
* UID ( Owner )
* GID ( Group )
* FileSize
* Time stamps including last access, last modification and last inode number change.
* File deletion time
* Number of links ( soft/hard )
* Location of ile on harddisk.
* Some other metadata about file

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

**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.

$ df -i

$ df -i /dev/sda1

**Inode Number**

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

$ ls -li <filename>

$ stat <filename>

## **Soft link vs Hard link**

Every file on the Linux filesystem starts with a single hard link. The *link* is between the filename and the actual data stored on the filesystem.

After a new hard link is created , there is no way to tell which off the existing hard links is original.

Haard link count will get increased.

Minimum hardlink count for directory is 2 & for file 1

|  |  |
| --- | --- |
| Soft Link {l} | Hard Link {-} |
| $ln -s <source\_file> <link\_fle>  link file should be already exist in same directory | $ln <source\_file> <link\_file> |
| A symbolic or soft link is an actual link to the original file.  It does contain the content just only the link | hard link is a mirror copy of the original file. |
| Inode number & permission is different | Inode number & permission is same.  permissions will be updated if we change the permissions of source file |
| If you delete the original file, the soft link has no value, because it points to a non-existent file. This situation is referred to as a *dangling soft link* | If you delete the original file, the hard link can still has the data of the original file. Because hard link acts as a mirror copy of the original file. |
| Span Across the file system, | Only within filesystem |
| Both files & directories | Only files |
| Usage  If one file system hass no memory but need to create 1gb, you can create in another filesystem & make softlink  #mkdir script/hello.sh  #ln -s script/hello.sh welcome  #./welcome | Backup |

File Management

## File Creatiom [Touch]

* The touch command is a standard command used in UNIX/Linux operating system which is used to create, change and modify timestamps of a file.
* It is used to create a file without any content.
* $ touch <filename> $ touch file{1..3}
* $ touch -a <filename>

We can change the access time of a file using -a option. By default it will take the current system time and update the atime field

* $ touch -m \*.o

You can change the modification time of a file using -m option

* $ touch -t [[CC]YY]MMDDhhmm[.SS]
* $ touch -am -t 203801181205.09 tgs.txt

Instead of taking the current time-stamp, you can explicitly specify the time file

CC : 1st 2 digits of the yr

YY: last 2 digits of the yr

MM: month DD: date hh: hours mm:min SS:seconds

* $ touch <file1> -r <file2>

Copy the Time-stamp from Another File using -r

## CAT Command

cat command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files

SYNTAX : $ cat [options] [file1] [file2]……

1. Display the contents of file

$ cat <filename1>

1. Display the contents of multiple file

$ cat <filename1> <filename2> <filename3>

1. Create a file with cat command

$ cat > filename

Awaits input from user, type desired text and press CTRL+D

1. Use cat command with more & less options

If file having large number of content that won’t fit in output terminal and screen scrolls up very fast, we can use parameters more and less with cat command

$ cat song.txt | more

$ cat view.text | less

1. Display Line numbers in file

$ cat -n <filename>

1. Display $ at the End of file

You can see with -e option that ‘$‘ is shows at the end of line and also in space showing ‘$‘ if there is any gap between paragraphs. This options is useful to squeeze multiple lines in a single line

$ cat -e filename

1. Display Tab separated lines in file

we could see TAB space is filled up with ‘^I‘ character

$ cat -T filename

1. Use standard output with redirection operators

$ cat file1 file2 file3 file4 > newfile

1. Apending Standard output with redirection operator

$ cat file1 >> newfile

1. Sorting contents of multiple files in a single file

$ cat file1 file2 file2 file3 | sort > newfile

1. Cat command can suppress repeated empty lines in output.

$ cat -s filename

1. Cat command can display content in reverse order using tac command

$ tac filename

## Directory Creation

The mkdir command in Linux/Unix allows users to create or make new directories. mkdir stands for “make directory.”

SYNTAX: $ mkdir [options] [directoryname]….

How to create the directories ?

$ mkdir dir1 dir2 dir3

How to make Parent directories ?

$ mkdir -p Linux/dir1/dir2/dir3

$ mkdir -p Linux/dir Linux dir1 Linux/dir2

## Copy Command [cp]

SYNTAX : $ cp [OPTIONS] source Destination

* Copy a file to another file

Two file names : If the command contains two file names, then it copy the contents of 1st file to the 2nd file. If the 2nd file doesn’t exist, then first it creates one and content is copied to it. But if it existed then it is simply overwritten without any warning

$ cp /dev/etc/file2 /Desktop/newfile

* Copy File(s) to another directory or folder

$ cp file1 file2 file……n directory

* Copy a recursively all content from one directory to another

$ cp -r directory dest\_dir

* Do not overwrite the existing file while copying

$ cp -n file1 file2

* Creating backup of existing destination file while copying (–backup)

$ cp --backup=simple -v /home/linuxtechi/distributions.txt

* Preserve mode, ownership and timestamps while copying (-p)

$ cp -p file1 file2

Options

* -f force
* -i interactive
* -v verbose
* -r recursive
* -n do not overwrite existing file
* - p preserve permsion timestamps
* -u update

copy only when the SOURCE file is newer than the destination file or when the destination file is missing

## Remove Command [rm]

SYNTAX: $ rm [options] [filename/directory]

$ rm -rvf file1

$ rm -r emptydirectory

To remove a file whose name starts with a '-', for example '-foo',

use one of these commands:

$ rm -- -foo

## Move Command [mv]

* mv [Option] source destination
* -n never overwrite any existing file
* -f always overwrite existing file force

## Text Editor [VIM]

* Command mode Navigation | undo | redo | cut | paste
* Insert mode
* Visual mode [select mode]

Inserting Text

i Insert text before cursor position

o open new line below current line

O open new line above the current line

A Append text at the end of the current line.

a append test after cursor position

Navigation

$ Move to the end of current line

0 [zero] Move to the beginning of current line

Undo || Redo

u undo the last command

CNTRL + r Redo

. [dot] repeat the last command

Deleting text

x delete the character at current cursor position

dw delete word or part of word to the right of cursor

dd delete current line

D delete current line starting from cursor position

Copy & Paste

yw copy the current word

yy copy the current line

p paste the data below current line

P paste the data above current line.

Save | Quit

:w write [save] change into file without quiting

:w! force save

:wq save and quit

:wq! Force save & quit

:q quit without save

:q! force quit

:w <filename> save current file under dif file name

Running the command

:r! command

:set number

:set nonumber

Search | Replace

/<string> search forward for string

?<string> search string backword

:%s/old/new search and replace 1st occurrence in each line

:%s/old/new/g search and replace all occurrence