**Basic Linux Commands:**

**Listing Files**

To list the files and directories stored in the current directory, use the following command −

$ls

The command **ls** supports the **-l** option which would help you to get more information about the listed files −

$ls -l

To list the invisible files, specify the **-a** option to **ls** −

$ ls -a

**Creating Files**

You can use the **vi** editor to create ordinary files on any Linux system.

$ vi filename

press the key **i** to come into the edit mode. Once you are in the edit mode, you can start writing your content in the file.

***Simple Commands of the vi Editor***

|  |  |
| --- | --- |
| **ESC** | Changes to last line mode. |
| i | Changes to insert mode (characters appear at the current cursor position). |
| a | Changes to insert mode (characters appear *after* the current cursor position). |
| A | Changes to insert mode (characters are added at the end of the line). |
| R | Changes to command mode (overwrites the old text). |
| r | Changes to insert mode and overwrites *each* character. |
| s | Changes to insert mode (the character where the cursor is positioned is replaced by the next entry you make). |
| C | Changes to insert mode (the rest of the line is replaced by the new text). |
| o | Changes to insert mode (a new line is inserted *following* the current one). |
| O | Changes to insert mode (a new line is inserted *preceding* the current one). |
| x | Deletes the current character. |
| dd | Deletes the current line. |
| dw | Deletes up to the end of the current word. |
| cw | Changes to insert mode (the rest of the current word is overwritten by the next entries you make). |
| u | Undoes the last command. |
| J | Joins the following line with the current one. |
| . | Repeats the last command. |
| : | Changes to last line mode. |

## Editing Files

You can edit an existing file using the **vi** editor.

$ vi filename

Once the file is opened, enter into edit mode by pressing the key **i** and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key **Esc**. After this, you can use the following keys to move inside a file −

* **l** key to move to the right side.
* **h** key to move to the left side.
* **k** key to move upside in the file.
* **j** key to move downside in the file.

So using the above keys, you can position your cursor wherever you want to edit. Once you are positioned, then you can use the **i** key to come in the edit mode. Once you are done with the editing in your file, press **Esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

## Display Content of a File

You can use the **cat** command to see the content of a file. Following is a simple example to see the content of the above created file −

$ cat filename

This is unix file....I created it for the first time.....

I'm going to save this content in this file.

$

You can display the line numbers by using the **-b** option along with the **cat**command as follows −

$ cat -b filename

1 This is unix file....I created it for the first time.....

2 I'm going to save this content in this file.

$

## Counting Words in a File

You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file. Following is a simple example to see the information about the file created above −

$ wc filename

2 19 103 filename

$

Here is the detail of all the four columns −

* **First Column** − Represents the total number of lines in the file.
* **Second Column** − Represents the total number of words in the file.
* **Third Column** − Represents the total number of bytes in the file. This is the actual size of the file.
* **Fourth Column** − Represents the file name.

You can give multiple files and get information about those files at a time. Following is simple syntax −

$ wc filename1 filename2 filename3

## Copying Files

To make a copy of a file use the **cp** command. The basic syntax of the command is −

$ cp source\_file destination\_file

Following is the example to create a copy of the existing file **filename**.

$ cp filename copyfile

$

You will now find one more file **copyfile** in your current directory. This file will exactly be the same as the original file **filename**.

## Renaming Files

To change the name of a file, use the **mv** command. Following is the basic syntax −

$ mv old\_file new\_file

The following program will rename the existing file **filename** to **newfile**.

$ mv filename newfile

$

The **mv** command will move the existing file completely into the new file. In this case, you will find only **newfile** in your current directory.

## Deleting Files

To delete an existing file, use the **rm** command. Following is the basic syntax −

$ rm filename

**Caution** − A file may contain useful information. It is always recommended to be careful while using this **Delete** command. It is better to use the **-i** option along with **rm** command.

Following is the example which shows how to completely remove the existing file **filename**.

$ rm filename

$

You can remove multiple files at a time with the command given below −

$ rm filename1 filename2 filename3

$

**File Commands:**

**ls** [option(s)] [file(s)]

If you run **ls** without any additional parameters, the program will list the contents of the current directory in short form.

-l

detailed list

-a

displays hidden files

**cp** [option(s)] sourcefile targetfile

Copies source file to target file.

-i

Waits for confirmation, if necessary, before an existing target file is overwritten

-r

Copies recursively (includes subdirectories)

**mv** [option(s)] source file target file

Copies source file to target file then deletes the original source file.

-b

Creates a backup copy of the source file before moving

-i

Waits for confirmation, if necessary, before an existing target file is overwritten

**rm** [option(s)] file(s)

Removes the specified files from the file system. Directories are not removed by **rm** unless the option -r is used.

-r

Deletes any existing subdirectories

-i

Waits for confirmation before deleting each file.

**ln** [option(s)] source file target file

Creates an internal [*link*](https://www-uxsup.csx.cam.ac.uk/pub/doc/suse/suse9.0/userguide-9.0/go01.html#link) from the source file to the target file, under a different name. Normally, such a link points directly to the source file on one and the same file system. However, if **ln** is executed with the -s option, it creates a symbolic link that only points to the directory where the source file is located, thus enabling linking across file systems.

-s

Creates a symbolic link

**cd** [options(s)] [directory]

Changes the current directory. **cd** without any parameters changes to the user's home directory.

**mkdir** [option(s)] directory name

Creates a new directory.

**rmdir** [option(s)] directory name

Deletes the specified directory, provided it is already empty.

**chown** [option(s)] username.group file(s)

Transfers the ownership of a file to the user with the specified user name.

-R

Changes files and directories in all subdirectories.

**chgrp** [option(s)] groupname file(s)

Transfers the group ownership of a given file to the group with the specified group name. The file owner can only change group ownership if a member of both the existing and the new group.

**chmod** [options] mode file(s)

Changes the access permissions.

The mode parameter has three parts: group, access, and access type. group accepts the following characters:

u

user

g

group

o

others

For access, access is granted by the + symbol and denied by the - symbol.

The access type is controlled by the following options:

r

read

w

write

x

eXecute — executing files or changing to the directory.

s

Set uid bit — the application or program is started as if it were started by the owner of the file.

**gzip** [parameters] file(s)

This program compresses the contents of files, using complex mathematical algorithms. Files compressed in this way are given the extension .gz and need to be uncompressed before they can be used. To compress several files or even entire directories, use the **tar** command.

-d

decompresses the packed gzip files so they return to their original size and can be processed normally (like the command **gunzip**).

**tar** options archive file(s)

The **tar** puts one file or (usually) several files into an archive. Compression is optional.

**tar** is a quite complex command with a number of options available. The most frequently used options are:

-f

Writes the output to a file and not to the screen as is usually the case

-c

Creates a new tar archive

-r

Adds files to an existing archive

-t

Outputs the contents of an archive

-u

Adds files, but only if they are newer than the files already contained in the archive

-x

Unpacks files from an archive (*extraction*)

-z

Packs the resulting archive with **gzip**

-j

Compresses the resulting archive with **bzip2**

-v

Lists files processed

The archive files created by **tar** end with .tar. If the tar archive was also compressed using **gzip**, the ending is .tgz or .tar.gz. If it was compressed using **bzip2**, .tar.bz2.

Application examples can be found in Section [“Archives and Data Compression”](https://www-uxsup.csx.cam.ac.uk/pub/doc/suse/suse9.0/userguide-9.0/ch24s02.html#sec:bash.tar).

**locate** pattern(s)

The locate command can find in which directory a specified file is located. If desired, use [*wild cards*](https://www-uxsup.csx.cam.ac.uk/pub/doc/suse/suse9.0/userguide-9.0/go01.html#wildcard) to specify file names. The program is very speedy, as it uses a database specifically created for the purpose (rather than searching through the entire file system). This very fact, however, also results in a major drawback: locate is unable to find any files created after the latest update of its database.

The database can be generated by root with **updatedb**.

**updatedb** [options(s)]

This command performs an update of the database used by **locate**. To include files in all existing directories, run the program as root. It also makes sense to place it in the background by appending an ampersand (&), so you can immediately continue working on the same command line (**updatedb &**).

**find** [option(s)]

The **find** command allows you to search for a file in a given directory. The first argument specifies the directory in which to start the search. The option **-name** must be followed by a search string, which may also include [*wild cards*](https://www-uxsup.csx.cam.ac.uk/pub/doc/suse/suse9.0/userguide-9.0/go01.html#wildcard). Unlike **locate**, which uses a database, **find** scans the actual directory.

#### Commands to Access File Contents

**cat** [option(s)] file(s)

The **cat** command displays the contents of a file, printing the entire contents to the screen without interruption.

-n

Numbers the output on the left margin

**less** [option(s)] file(s)

This command can be used to browse the contents of the specified file. Scroll half a screen page up or down with **PgUp** and **PgDn** or a full screen page down with **Space**. Jump to the beginning or end of a file using **Home** and **End**. Press **Q** to exit the program.