



Top 10 Linux Commands for Newbies

Linux is a free and open-source operating system that has gained widespread acceptance over the years. It is extensively used in a variety of domains, including servers, embedded systems, and personal computers. Learning Linux can be intimidating for newbies, but it's actually not that difficult. The usage of commands is one of the most important things to master in Linux. With this article 'Top 10 Linux Commands for newbies', get to explore the world of Linux Commands, that every beginner Linux user should know.

Table of Contents

- List of Top 100 LINUX Commands..... 1
- 1. ls..... 1
- 2. cd..... 2
- 3. pwd..... 2
- 4. touch..... 2
- 5. cat..... 3
- 6. cp..... 3
- 7. mv..... 3
- 8. mkdir..... 4
- 9. rm..... 4
- 10. man..... 4

List of Top 10 LINUX Commands for Newbies

Following, I will give a list of **10 top Linux commands for newbies** with their description, syntax & useful options. You can also learn more about these commands with necessary examples & practical usages by visiting the corresponding attached link that is appended after each command.

1. ls

The **ls** command '**Lists**' the contents, both files and subdirectories of the current directory by default. It is one of the most used commands, as one can view the contents of a directory without exiting the terminal and perform their desired tasks on the specific contents.

Syntax

```
ls [OPTION]... [FILE]...
```

Useful Options

- **-a** → Doesn't ignore the hidden files (files named with **.(dot)** at the beginning).
- **-h** → Prints sizes in human-readable forms.
- **-l** → Lists in a long form.
- **-S** → Sorts according to file size, largest first.
- **-d** → Only lists the current directory, not its contents.

Practical Example

To see how the **ls** command works simply type the **ls** command in your terminal as follows:

```

ls
munny@ubuntu:~$ ls
A1 Desktop iso new.txt Shared_Folder sh_files snap test text_files
munny@ubuntu:~$
  
```

As shown in the image above, employing the command without any additional arguments displays all the files and directories within the current directory.

To learn more read, [The “ls” Command in Linux \[7+ Practical Examples\]](#)

2. pwd

The **pwd** command stands for **print working directory**. It displays the **Absolute path** of the **current** directory, in a simple context, prints the name of the **current/working** directory all the way beginning from the root(/) directory. So it shows where the **Terminal** currently is in detail and will help you when you are lost inside some unknown directory.

Syntax

```
pwd [OPTION]...
```

Useful Options

- **-L, --logical** → Even as it carries symlinks, PWD utilizes from the environment.
- **-P, --physical** → Avoids the symlinks.

When no option is mentioned, it is assumed that **option -P** is being used.

Practical Example

To print the name of the working directory using the **pwd** command, simply type the command on your terminal:

```

pwd
munny@ubuntu:~$ pwd
/home/munny
munny@ubuntu:~$
  
```

The image shows the directory I was working on which is **/home/munny**.

Typically, your terminal prompt already displays the entire directory path. However, if it's not visible, this command is a swift way to view your current directory. Additionally, when crafting scripts, this command proves valuable by helping identify the directory where the script is saved.

To learn more read, [The “pwd” Command in Linux \[4 Practical Examples\]](#)

3. cd

The word **cd** stands for **change directory**. This command is used for changing the current directory of the user. It will take the user from the current directory (current location) to a specified directory.

Syntax

```
cd [OPTION]... [DIRECTORY]
```

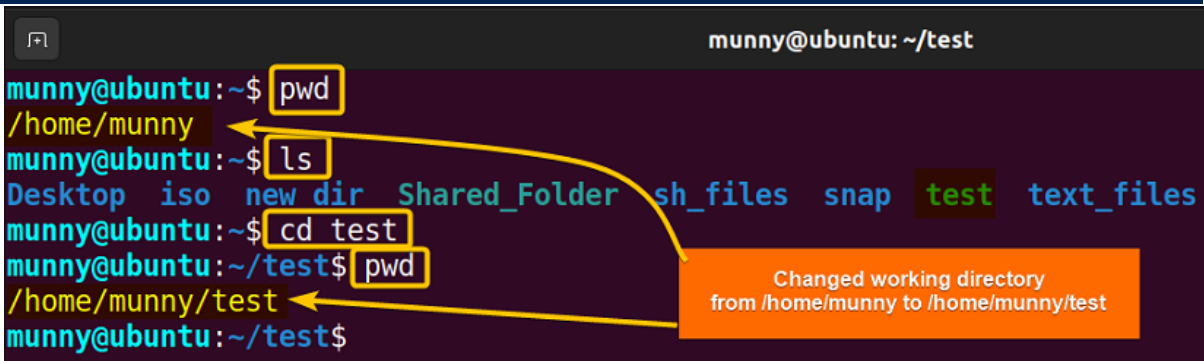
Useful Options

- **cd ~[username]** → Changes the directory to the home directory of the specified user.
- **cd ..** → Changes directory one directory up the current directory.
- **cd -** → Changes the directory to the previously changed directory.

Practical Example

To see practically how the **cd** command works, let's change the current working directory to another directory. For that just type the command followed by the directory name:

```
cd test
```



```
munny@ubuntu: ~$ pwd
/home/munny
munny@ubuntu: ~$ ls
Desktop iso new_dir Shared_Folder sh_files snap test text_files
munny@ubuntu: ~$ cd test
munny@ubuntu: ~/test$ pwd
/home/munny/test
munny@ubuntu: ~/test$
```

Changed working directory from /home/munny to /home/munny/test

See from the image, first, I typed the **pwd** command to show my current working directory. After that, I displayed the files & directory lists of that directory, just to show you in which directory I will move (here I selected the **test** directory). Then I shifted to that directory & used the **pwd** command again to show you the changed directory name.

To learn more read, [The “cd” Command in Linux \[6 Practical Examples\]](#)

4. touch

The **touch** command allows us to update a file's access or modification time. However, if the file doesn't exist you can **create** that file. This ability to create files makes the **touch** command one of the most useful commands.

Syntax

```
touch [OPTION]... FILE...
```

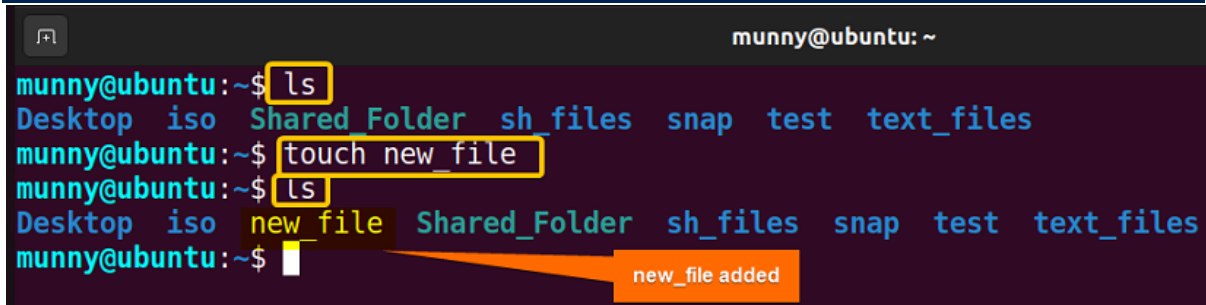
Useful Options

- **-a** → changes only the access time.
- **-m** → Changes only the modification time.

Practical Example

To create a new file use the touch command followed by the file name which will create a file in the current directory. Now to create a file named **new_file**, run the command:

```
touch new_file
```



A terminal window titled 'munny@ubuntu: ~' showing a sequence of commands and their outputs. The first command is 'ls', which lists files: Desktop, iso, Shared Folder, sh_files, snap, test, text_files. The second command is 'touch new_file'. The third command is 'ls', which now lists: Desktop, iso, new_file, Shared Folder, sh_files, snap, test, text_files. An orange callout box points to the 'new_file' entry in the second 'ls' output, with the text 'new_file added'.

The output image shows how a new file named **new_file** is created and added to the list.

To learn more read, [The “touch” Command in Linux \[8 Practical Examples\]](#)

5. cat

The **cat** command prints the contents of the file specified. Generally, **cat** (concatenates) reads the contents of the files fed to its arguments and prints them serially on the **terminal**.

Syntax

```
cat [OPTION]... [FILE]...
```

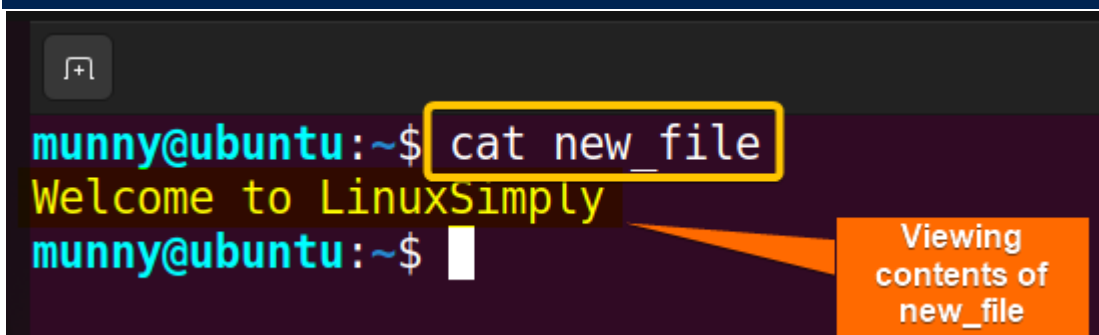
Useful Options

- **-E, --show-ends** → Display \$ at the end of each line.
- **-n, --number** → Displays line numbers when utilized.
- **-s, --squeeze-blank** → suppress repeated empty output lines.

Practical Example

To print the contents of any file just type the cat command followed by the file name. For example, to see the contents of **new_file**:

```
cat new_file
```



A terminal window titled 'munny@ubuntu: ~' showing the command 'cat new_file' being executed. The output is 'Welcome to LinuxSimply'. An orange callout box points to the output text with the text 'Viewing contents of new_file'.

See from the above snapshot, the **cat** command is displaying the contents of the **new_file**.

To learn more read, [The “cat” Command in Linux \[10 Practical Examples\]](#)

6. cp

The **cp** command resembles the word ‘copy’. As the name suggests, it copies things from one place to another place. The command can copy one or multiple files to the specified destination directory. If the directory doesn’t exist it just renames the files. It can also be used to copy directories and their contents.

Syntax

The syntax for Copying Files

```
cp [OPTION]... [-T] SOURCE DESTINATION
```

The syntax for Copying Files to a Directory

```
cp [OPTION]... SOURCE... DIRECTORY
```

The syntax for Copying Directory

```
cp [OPTION]... SOURCE DIRECTORY DESTINATION DIRECTORY
```

Useful Options

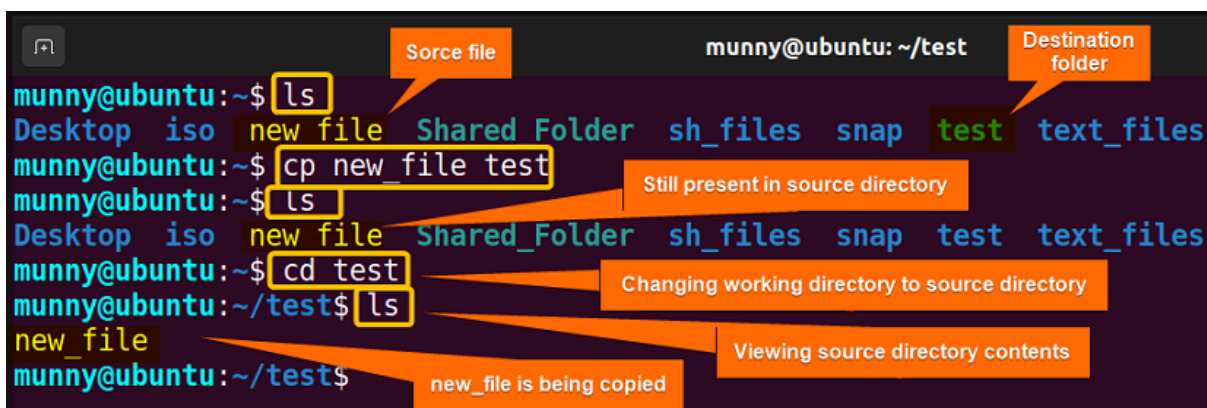
- **-i, --interactive** → Displays interactive prompt before completing the modification.
- **-R, -r, --recursive** → Copies the directory as well as its contents recursively.
- **-v, --verbose** → Prints message of what is being performed.

Practical Example

Type the **cp** command with the source file name (in this case, **new_file**) & the destination directory name (here, **test**). Remember to add space between these two. See the command below:

```
cp new_file test
```

Note: Make sure to add the relative path for the source file and destination folder if they are not present in the current working directory.



```
munny@ubuntu: ~/test
munny@ubuntu:~$ ls
Desktop iso new_file Shared Folder sh_files snap test text_files
munny@ubuntu:~$ cp new_file test
munny@ubuntu:~$ ls
Desktop iso new_file Shared Folder sh_files snap test text_files
munny@ubuntu:~$ cd test
munny@ubuntu:~/test$ ls
new_file
munny@ubuntu:~/test$
```

Annotations in the image:

- Source file: points to `new_file` in the first `ls` command.
- Destination folder: points to `test` in the first `ls` command.
- Still present in source directory: points to `new_file` in the second `ls` command.
- Changing working directory to source directory: points to `cd test`.
- Viewing source directory contents: points to `ls` in the `~/test` directory.
- `new_file is being copied`: points to the `cp` command.

Here, the **cp** command copies the **new_file** from its source folder to the new destination directory. I executed the **ls** before after executing the **cp** command to show you the changes made by the command and used the **cd** command to enter the destination directory to display the copied file there.

To learn more read, [The “cp” Command in Linux \[6 Practical Examples\]](#)

7. mv

The **mv** command is a widely used **file/folder** management command that allows changing the location of a file or folder. Moreover, when changing the location you can rename the file as well just by changing the file name.

Syntax

```
mv [OPTION]... SOURCE... DESTINATION
```

Useful Options

- **-n, --no-clobber** → Does not overwrite an existing file.
- **-i, --interactive** → Prompts before overwriting.
- **-f, --force** → Does not prompt before overwriting.
- **-v, --verbose** → Explains what is being done.
- **-u, --update** → Moves only when the source file is newer than the Destination file.

Practical Example

This command works almost the same way as the **cp** command. Just that it moves the file/folder to a new place instead of copying it. Same as the **cp** command add the source file name and then the destination folder name. Such as:

```
mv new_file test
```

The screenshot shows a terminal window with the following sequence of commands and outputs:

```
munny@ubuntu:~$ ls
Desktop iso new_file Shared Folder sh_files snap test text_files
munny@ubuntu:~$ mv new_file test
munny@ubuntu:~$ ls
Desktop iso Shared Folder sh_files snap test text_files
munny@ubuntu:~$ cd test
munny@ubuntu:~/test$ ls
new_file
munny@ubuntu:~/test$
```

Annotations in the image:

- Source file**: Points to `new_file` in the first `ls` command.
- Destination folder**: Points to `test` in the `mv` command.
- Source file is not present in list anymore**: Points to the absence of `new_file` in the second `ls` command.
- Changing working directory to the destination folder**: Points to the `cd test` command.
- new_file has been moved**: Points to `new_file` in the final `ls` command.
- Viewing destination folder contents**: Points to the `ls` command in the `~/test` directory.

See from the image, the **mv** command moves the `new_file` from its source folder to the new destination directory. I executed the `ls` before and after executing the **mv** command to show you the changes made by the command. As you can see after running the **mv** command the source file is not present in its previous location anymore. Later used the `cd` command to enter the destination directory to display the moved file there.

To learn more read, [The “mv” Command in Linux \[8 Practical Examples\]](#)

8. mkdir

The command **mkdir** is the abbreviation for **make directory**. As the name suggests, the **mkdir** command is used to create one or more directories.

Syntax

```
mkdir [OPTION]... DIRECTORY...
```

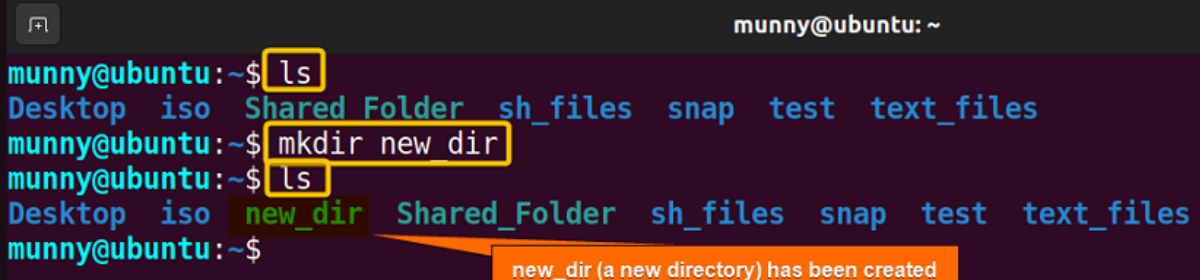
Useful Options

- **-p - -parents** → Creates the necessary parent directories if required.
- **-v, --verbose** → Prints message of what is being performed.

Practical Example

It's so straightforward to use the `mkdir` command, just type the desired directory name after the command. For example, to create a directory named `new_dir` type:

```
mkdir new_dir
```



```
munny@ubuntu:~$ ls
Desktop iso Shared Folder sh_files snap test text_files
munny@ubuntu:~$ mkdir new_dir
munny@ubuntu:~$ ls
Desktop iso new_dir Shared Folder sh_files snap test text_files
munny@ubuntu:~$
```

new_dir (a new directory) has been created

Upon execution, the command adds the new directory `new_dir` to the list, as you can see from the output of the `ls` command.

To learn more read, [The “mkdir” Command in Linux \[6+ Practical Examples\]](#)

9. rm

The `rm` command is the abbreviation for **remove**. As the name suggests, it removes files and the removal is **permanent**, so be cautious while using it. The command can also be used to remove directories and their contents permanently.

Syntax

```
rm [OPTION]... [FILE]...
```

Useful Options

- **-i** → Displays interactive prompt before completing the deletion each time.
- **-I** → Only shows prompt while deleting 3 or more files or deleting recursively.
- **-d, --dir** → Removes the empty directories.
- **-R, -r, --recursive** → Removes any directory as well as its contents recursively.
- **-v, --verbose** → Prints message of what is being performed.

Practical Example

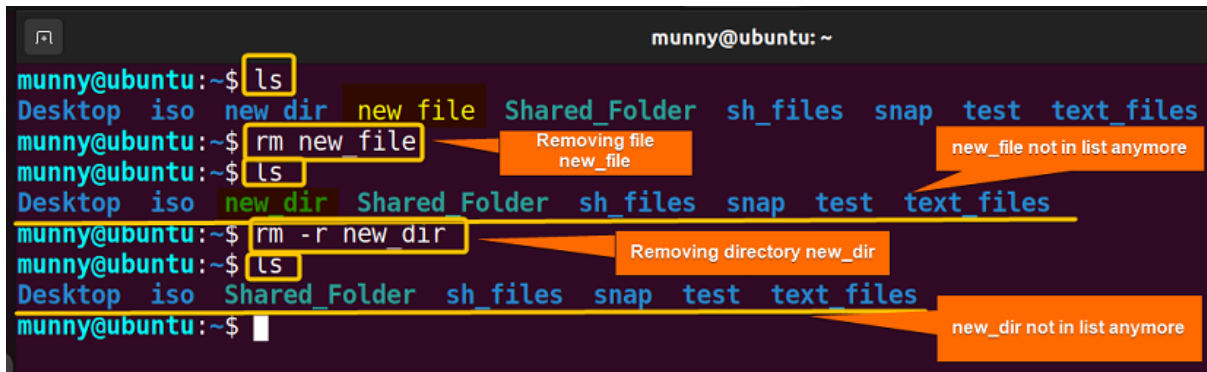
By adding the file name simply after the `rm` command, it removes it from the system permanently. Let's delete previously created `new_file`:

```
rm new_file
```

To delete a directory, you have to add the **--recursive** or **-r** option to it. Without the **-r** argument, the `rm` command won't delete directories. Now, to delete the `new_dir` directory, use the command:

```
rm -r new_dir
```

```
munny@ubuntu: ~
munny@ubuntu:~$ ls
Desktop iso new_dir new_file Shared_Folder sh_files snap test text_files
munny@ubuntu:~$ rm new_file
munny@ubuntu:~$ ls
Desktop iso new_dir Shared_Folder sh_files snap test text_files
munny@ubuntu:~$ rm -r new_dir
munny@ubuntu:~$ ls
Desktop iso Shared_Folder sh_files snap test text_files
munny@ubuntu:~$
```

The image shows a terminal window with a dark background. The user 'munny' is at the 'ubuntu' machine in the home directory. The terminal shows a sequence of commands: 'ls' to list files, 'rm new_file' to remove a file, 'ls' to verify, 'rm -r new_dir' to remove a directory, and 'ls' to verify again. Annotations in orange boxes point to the commands: 'Removing file new_file' points to 'rm new_file', 'new_file not in list anymore' points to the second 'ls' output, 'Removing directory new_dir' points to 'rm -r new_dir', and 'new_dir not in list anymore' points to the final 'ls' output.

The **rm** command removes the file **new_file**, as well as this command with command option **-r** removes the directory **new_dir**, as you can see from the above image. I used the **ls** command after executing each of the commands so that you get a clearer overview of how the command works.

To learn more read, [The “rm” Command in Linux \[7 Practical Examples\]](#)

10. man

The **man** command in **Linux** stands for **manual**. Upon execution, it will display a **manual** page or **documentation** of a specified **Linux** command. It displays information like the **synopsis**, **description**, **options**, **exit status**, **authors**, **copyright**, etc.

Syntax

```
man [OPTION]... [Command_NAME]...
```

Useful Options

- **SECTION COMMAND** → Shows the specific SECTION of a COMMAND.
- **-k KEYWORD** → Search for the Keyword in the whole manual page and show all the matches.
- **-f KEYWORD** → Looks for a short description of any Keyword or Command.
- **-d, --default** → Resets the man command behavior to default.
- **-i, --ignore-case** → Ignore case sensitivity of the command.
- **-I, --match-case** → Looking inside the man page with case sensitivity.
- **-a, --all** → Shows all manual pages that match the specific keyword or command.

Practical Example

To see the manual page of the **man** command just type the following command:

```
man man
```



```
munny@ubuntu: ~
MAN(1) Manual pager utils MAN(1)
NAME
  man - an interface to the system reference manuals
SYNOPSIS
  man [man options] [[section] page ...] ...
  man -k [apropos options] regexp ...
  man -K [man options] [section] term ...
  man -f [whatis options] page ...
  man -l [man options] file ...
  man -w|-W [man options] page ...
DESCRIPTION
  man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the
Manual page man(1) line 1 (press h for help or q to quit)
```

The command output displays the manual page for the **man** command, providing detailed information about its usage and options.

And, to view the manual page of any other command type the command name as an argument after the **man** command. For instance, to view the **man page** of the **pwd** command, type:

```
man pwd
```

```
munny@ubuntu: ~
PWD(1) User Commands
NAME
  pwd - print name of current/working directory
SYNOPSIS
  pwd [OPTION]...
DESCRIPTION
  Print the full filename of the current working directory.
  -L, --logical
    use PWD from environment, even if it contains symlinks
  -P, --physical
    avoid all symlinks
  --help
    display this help and exit
Manual page pwd(1) line 1 (press h for help or q to quit)
```

From the manpage of the **pwd** command, you can see the command **name**, **description**, and **synopsis**, along with all the **flags** or **options** of the command with their short description.

To learn more read, [The “man” Command in Linux \[6 Practical Examples\]](#)

Conclusion

To sum up, With these **top 10 Linux commands**, you're well on your way to becoming a command-line ninja. As you become more comfortable, explore additional commands to further enhance your Linux skills. Hope it helps new users to surf through the CLI world!

To learn more Linux commands, go through [100 top Linux commands](#).

