1. All file names are case sensitive. So filename vivek.txt Vivek.txt VIVEK.txt all are three different files.
2. You can use upper and lowercase letters, numbers, “.” (dot), and “\_” (underscore) symbols.

[](https://www.cyberciti.biz/faq/category/file-system/)

1. You can use other special characters such as blank space, but they are hard to use and it is better to avoid them.
2. In short, filenames may contain any character except / (root directory), which is reserved as the separator between files and directories in a pathname. You cannot use the null character.
3. No need to use . (dot) in a filename. Some time dot improves readability of filenames. And you can use dot based filename extension to identify file. For example:
   * .sh = Shell file
   * .tar.gz = Compressed archive
4. Most modern Linux and UNIX limit filename to 255 characters (255 bytes). However, some older version of UNIX system limits filenames to 14 characters only.
5. A filename must be unique inside its directory. For example, inside /home/vivek directory you cannot create a demo.txt file and demo.txt directory name. However, other directory may have files with the same names. For example, you can create demo.txt directory in /tmp.

## **Linux / UNIX: Reserved Characters And Words**

Avoid using the following characters from appearing in file names:

1. /
2. >
3. <
4. |
5. :
6. &

Please note that Linux and UNIX allows white spaces, <, >, |, \, :, (, ), &, ;, as well as wildcards such as ? and \*, to be quoted or [escaped](https://bash.cyberciti.biz/guide/Quoting) using \ symbol.

**8.3. absolute and relative paths**

You should be aware of **absolute and relative paths** in the file tree. When you type a path

starting with a **slash (/)**, then the **root** of the file tree is assumed. If you don't start your path

with a slash, then the current directory is the assumed starting point.

The screenshot below first shows the current directory **/home/paul**. From within this

directory, you have to type **cd /home** instead of **cd home** to go to the **/home** directory.

paul@debian8$ **pwd**

/home/paul

paul@debian8$ **cd home**

bash: cd: home: No such file or directory

paul@debian8$ **cd /home**

paul@debian8$ **pwd**

/home

When inside **/home**, you have to type **cd paul** instead of **cd /paul** to enter the subdirectory

**paul** of the current directory **/home**.

paul@debian8$ **pwd**

/home

paul@debian8$ **cd /paul**

bash: cd: /paul: No such file or directory

paul@debian8$ **cd paul**

paul@debian8$ **pwd**

/home/paul

In case your current directory is the **root directory /**, then both **cd /home** and **cd home** will

get you in the **/home** directory.

paul@debian8$ **pwd**

/

paul@debian8$ **cd home**

paul@debian8$ **pwd**

/home

paul@debian8$ **cd /**

paul@debian8$ **cd /home**

paul@debian8$ **pwd**

/home

This was the last screenshot with **pwd** statements. From now on, the current directory will

often be displayed in the prompt. Later in this book we will explain how the shell variable

**$PS1** can be configured to show this.

working with directories

{cd ~/virat}

77

**8.4. path completion**

The **tab key** can help you in typing a path without errors. Typing **cd /et** followed by the **tab**

**key** will expand the command line to **cd /etc/**. When typing **cd /Et** followed by the **tab key**,

nothing will happen because you typed the wrong **path** (upper case E).

You will need fewer key strokes when using the **tab key**, and you will be sure your typed

**path** is correct!

## **What is a path?**

A path is a unique location to a file or a folder in a file system of an OS. A path to a file is a combination of / and alpha-numeric characters.

## **What is an absolute path?**

An absolute path is defined as the specifying the location of a file or directory from the root directory(/). In other words we can say absolute path is a complete path from start of actual filesystem from / directory.

#### **Some examples of absolute path:**

**/var/**[**ftp**](http://www.linuxnix.com/ftps-server-configuration/)**/pub**

[**/etc**](http://www.linuxnix.com/linux-directory-structure-explainedetc-folder/)**/samba.smb.conf**

**/boot/grub/grub.conf**

If you see all these paths started from / directory which is a root directory for every Linux/Unix machines.

## **What is the relative path?**

Relative path is defined as path related to the present working directory(pwd). Suppose I am located in /var/log and I want to change directory to /var/log/kernel. I can use relative path concept to change directory to kernel

changing directory to /var/log/kernel by using relative path concept.

**pwd/var/logcd kernel**

**Note:** If you observe there is no / before kernel which indicates itâs a relative directory to present working directory.

Changing directory to /var/log/kernel using absolute path concept.

**cd /var/log/kernel**

**Note:** We can use an absolute path from any location where as if you want to use relative path we should be present in a directory where we are going to specify relative to that present working directory.

**8.5. ls**

You can list the contents of a directory with **ls**.

paul@debian8:~$ **ls**

allfiles.txt dmesg.txt services stuff summer.txt

paul@debian8:~$

**8.5.1. ls -a**

A frequently used option with ls is **-a** to show all files. Showing all files means including

the **hidden files**. When a file name on a Linux file system starts with a dot, it is considered

a **hidden file** and it doesn't show up in regular file listings.

paul@debian8:~$ **ls**

allfiles.txt dmesg.txt services stuff summer.txt

paul@debian8:~$ **ls -a**

. allfiles.txt .bash\_profile dmesg.txt .lesshst stuff

.. .bash\_history .bashrc services .ssh summer.txt

paul@debian8:~$

**8.5.2. ls -l**

Many times you will be using options with **ls** to display the contents of the directory in

different formats or to display different parts of the directory. Typing just **ls** gives you a

list of files in the directory. Typing **ls -l** (that is a letter L, not the number 1) gives you a

long listing.

paul@debian8:~$ **ls -l**

total 17296

-rw-r--r-- 1 paul paul 17584442 Sep 17 00:03 allfiles.txt

-rw-r--r-- 1 paul paul 96650 Sep 17 00:03 dmesg.txt

-rw-r--r-- 1 paul paul 19558 Sep 17 00:04 services

drwxr-xr-x 2 paul paul 4096 Sep 17 00:04 stuff

-rw-r--r-- 1 paul paul 0 Sep 17 00:04 summer.txt

working with directories

78

**8.5.3. ls -lh**

Another frequently used ls option is **-h**. It shows the numbers (file sizes) in a more human

readable format. Also shown below is some variation in the way you can give the options

to **ls**. We will explain the details of the output later in this book.

*Note that we use the letter L as an option in this screenshot, not the number 1.*

paul@debian8:~$ **ls -l -h**

total 17M

-rw-r--r-- 1 paul paul 17M Sep 17 00:03 allfiles.txt

-rw-r--r-- 1 paul paul 95K Sep 17 00:03 dmesg.txt

-rw-r--r-- 1 paul paul 20K Sep 17 00:04 services

drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff

-rw-r--r-- 1 paul paul 0 Sep 17 00:04 summer.txt

paul@debian8:~$ **ls -lh**

total 17M

-rw-r--r-- 1 paul paul 17M Sep 17 00:03 allfiles.txt

-rw-r--r-- 1 paul paul 95K Sep 17 00:03 dmesg.txt

-rw-r--r-- 1 paul paul 20K Sep 17 00:04 services

drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff

-rw-r--r-- 1 paul paul 0 Sep 17 00:04 summer.txt

paul@debian8:~$ **ls -hl**

total 17M

-rw-r--r-- 1 paul paul 17M Sep 17 00:03 allfiles.txt

-rw-r--r-- 1 paul paul 95K Sep 17 00:03 dmesg.txt

-rw-r--r-- 1 paul paul 20K Sep 17 00:04 services

drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff

-rw-r--r-- 1 paul paul 0 Sep 17 00:04 summer.txt

paul@debian8:~$ **ls -h -l**

total 17M

-rw-r--r-- 1 paul paul 17M Sep 17 00:03 allfiles.txt

-rw-r--r-- 1 paul paul 95K Sep 17 00:03 dmesg.txt

-rw-r--r-- 1 paul paul 20K Sep 17 00:04 services

drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff

-rw-r--r-- 1 paul paul 0 Sep 17 00:04 summer.txt

paul@debian8:~$

**9.5. rm**

**9.5.1. remove forever**

When you no longer need a file, use **rm** to remove it. Unlike some graphical user interfaces,

the command line in general does not have a **waste bin** or **trash can** to recover files. When

you use **rm** to remove a file, the file is gone. Therefore, be careful when removing files!

paul@debian7:~$ **ls**

BigBattle.txt file33 file42 SinkoDeMayo

paul@debian7:~$ **rm BigBattle.txt**

paul@debian7:~$ **ls**

file33 file42 SinkoDeMayo

paul@debian7:~$

**9.5.2. rm -i**

To prevent yourself from accidentally removing a file, you can type **rm -i**.

paul@debian7:~$ **ls**

file33 file42 SinkoDeMayo

paul@debian7:~$ **rm -i file33**

rm: remove regular empty file `file33'? yes

paul@debian7:~$ **rm -i SinkoDeMayo**

rm: remove regular empty file `SinkoDeMayo'? n

paul@debian7:~$ **ls**

file42 SinkoDeMayo

paul@debian7:~$

**9.5.3. rm -rf**

By default, **rm -r** will not remove non-empty directories. However **rm** accepts several

options that will allow you to remove any directory. The **rm -rf** statement is famous because

it will erase anything (providing that you have the permissions to do so). When you are

logged on as root, be very careful with **rm -rf** (the **f** means **force** and the **r** means **recursive**)

since being root implies that permissions don't apply to you. You can literally erase your

entire file system by accident.

paul@debian7:~$ **mkdir test**

paul@debian7:~$ **rm test**

rm: cannot remove `test': Is a directory

paul@debian7:~$ **rm -rf test**

paul@debian7:~$ **ls test**

ls: cannot access test: No such file or directory

paul@debian7:~$

**9.6. cp**

**9.6.1. copy one file**

To copy a file, use **cp** with a source and a target argument.

paul@debian7:~$ **ls**

file42 SinkoDeMayo

paul@debian7:~$ **cp file42 file42.copy**

paul@debian7:~$ **ls**

file42 file42.copy SinkoDeMayo

**9.6.2. copy to another directory**

If the target is a directory, then the source files are copied to that target directory.

paul@debian7:~$ **mkdir dir42**

paul@debian7:~$ **cp SinkoDeMayo dir42**

paul@debian7:~$ **ls dir42/**

SinkoDeMayo

**9.6.3. cp -r**

To copy complete directories, use **cp -r** (the **-r** option forces **recursive** copying of all files

in all subdirectories).

paul@debian7:~$ **ls**

dir42 file42 file42.copy SinkoDeMayo

paul@debian7:~$ **cp -r dir42/ dir33**

paul@debian7:~$ **ls**

dir33 dir42 file42 file42.copy SinkoDeMayo

paul@debian7:~$ **ls dir33/**

SinkoDeMayo

**9.6.4. copy multiple files to directory**

You can also use cp to copy multiple files into a directory. In this case, the last argument

(a.k.a. the target) must be a directory.

paul@debian7:~$ **cp file42 file42.copy SinkoDeMayo dir42/**

paul@debian7:~$ **ls dir42/**

file42 file42.copy SinkoDeMayo

**9.6.5. cp -i**

To prevent **cp** from overwriting existing files, use the **-i** (for interactive) option.

paul@debian7:~$ **cp SinkoDeMayo file42**

paul@debian7:~$ **cp SinkoDeMayo file42**

paul@debian7:~$ **cp -i SinkoDeMayo file42**

cp: overwrite `file42'? **n**

paul@debian7:~$

working with files

89

**9.7. mv**

**9.7.1. rename files with mv**

Use **mv** to rename a file or to move the file to another directory.

paul@debian7:~$ **ls**

dir33 dir42 file42 file42.copy SinkoDeMayo

paul@debian7:~$ **mv file42 file33**

paul@debian7:~$ **ls**

dir33 dir42 file33 file42.copy SinkoDeMayo

paul@debian7:~$

When you need to rename only one file then **mv** is the preferred command to use.

**9.7.2. rename directories with mv**

The same **mv** command can be used to rename directories.

paul@debian7:~$ **ls -l**

total 8

drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir33

drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir42

-rw-r--r-- 1 paul paul 0 Oct 15 09:38 file33

-rw-r--r-- 1 paul paul 0 Oct 15 09:16 file42.copy

-rw-r--r-- 1 paul paul 0 May 5 2005 SinkoDeMayo

paul@debian7:~$ **mv dir33 backup**

paul@debian7:~$ **ls -l**

total 8

drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 backup

drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir42

-rw-r--r-- 1 paul paul 0 Oct 15 09:38 file33

-rw-r--r-- 1 paul paul 0 Oct 15 09:16 file42.copy

-rw-r--r-- 1 paul paul 0 May 5 2005 SinkoDeMayo

paul@debian7:~$

**9.7.3. mv -i**

The **mv** also has a **-i** switch similar to **cp** and **rm**.

this screenshot shows that **mv -i** will ask permission to overwrite an existing file.

paul@debian7:~$ **mv -i file33 SinkoDeMayo**

mv: overwrite `SinkoDeMayo'? **no**

paul@debian7:~$

### Introduction

This guide will show you how to copy files and folders from one place to another using the most popular graphical file managers and also by using the Linux command line.

Most people will be used to using graphical tools to copy files from their disks. If you are used to using Windows then you will be aware of a tool called Windows Explorer which makes it very easy.

Windows Explorer is a tool known as a file manager and Linux has a number of different file managers.

The one that appears on your system largely depends on the version of Linux you are using and to a certain degree [the desktop environment](https://www.lifewire.com/linux-desktop-environment-explained-4121640) you are using.

The most common file managers are as follows:

* Nautilus (aka files)
* Dolphin
* Thunar
* PCManFM
* Caja

If you are running [Ubuntu](https://www.lifewire.com/beginners-guide-to-ubuntu-2205722), [Linux Mint](https://www.lifewire.com/create-uefi-bootable-linux-mint-usb-2202084), [Zorin](https://www.lifewire.com/way-to-create-zorinos-usb-drive-2202071), [Fedora](https://www.lifewire.com/create-bootable-fedora-usb-drive-2202081) or [openSUSE](https://www.lifewire.com/create-bootable-opensuse-usb-drive-2202082) then it is likely that your file manager is called Nautilus.

### How To Use Nautilus To Copy Files And Folders

Nautilus will be available via the menu within Linux Mint and Zorin or it will appear in the Unity Launcher within Ubuntu or via the dashboard view within any distribution using GNOME such as Fedora or openSUSE.

To copy a file navigate through the file system by double clicking on the folders until you get to the file you wish to copy.

You can use the standard keyboard commands to copy files. For instance clicking on a file and pressing CTRL and C together takes a copy of a file. Pressing CTRL and V pastes the file in the location you choose to copy the file to.

If you paste a file into the same folder then it will have the same name as the original except it will have the word (copy) at the end of it.

You can also copy a file by right-clicking on the file and choose the "copy" menu item. You can then choose the folder you wish to paste it in, right click and choose "paste".

Another way of copying a file is to right-click on the file and choose the "copy to" option. A new window will appear. Find the folder you wish to copy the file to and click the "select" button.

You can copy multiple files by holding down the CTRL key whilst selecting each file. Any of the previous methods such as choosing CTRL C or selecting "copy" or "copy to" from the context menu will work for all selected files.

The copy command works on files and folders.