**Define the basic commands of the prompt to:**

***Create and delete directories (cd, mkdir, rm, rmdir, cp, mv)***

1. **CD** (Change *Directory*) is used to display the name of the current directory and allows you to change directories. **CD** is an internal **cmd.exe** command that has the same functionality as **CHDIR**.
2. The **cp** ("copy") command copies files and directories.

**Options**

After the command name, the file to be copied and the path and name of the new file will be indicated. It does not allow copying a file over itself, although it does allow overwriting an existing file.

If the name of two files is specified, the cp command copies the first to the second. If the -to option is added if the last argument indicated is a directory, cp copies all the indicated files to the directory.

* **-t**: Copy all the files indicated in the directory given as the last argument.

1. The **mkdir command**is one of the most used commands in the Linux Terminal or Console and **its function is to create directories or a set of directories**in the file system.

The syntax for the mkdir command is as follows:

*mkdir DirectoryName*

1. **rm command**

**rm** removes each specified file argument (if it is a valid path name). If you specify an invalid file name or path, rm displays an error message and goes to the next file. If you specify a file for which you do not have write permission, rm will ask you for a confirmation to see if you really want to delete it.

1. To delete a **rmdir** tmp directory

The rmdir command will remove the tmp directory if the directory is empty.

To delete a directory tree: rm -ir tmp

This command will repeatedly delete the contents of all subdirectories in the tmp directory, asking for confirmation to delete each file, and then delete the tmp directory itself.

1. Mv command syntax

The **mv command** is used to move or rename files and directories in your operating system file system. Additionally, it presents us with an option with which we can also back up files. Join us to learn how to use this command.

Depending on the objective, its syntax may have variations. Let us see the cases below:

*Rename a file named source to destination:*

[code] mv [options] [-T] source destination [/ code]

*Move the source files to a directory named destination:*

[code] mv [options] source [source2…] destination [/ code]

***Navigate (cd, pwd)***

**Pwd linux command**

The literal description of the linux pwd command is print working directory or print working directory. In simple terms it shows you the path or directory in which you are when you are in a linux terminal.

The way to use this command is as follows:

pwd [options] ...

This command does not have many parameters and does not require extensive explanations since its use is very simple. It has 2 useful parameters and another 2 that are the generic ones in all the commands for help and the version.

**-L, –logical** uses the PWD from the environment, even if it contains symbolic links  
**-P, –physical** excludes all symbolic links  
**-help** shows help for the command  
**–version** shows version information for this command

**CD** (Change *Directory*) is used to display the name of the current directory and allows you to change directories. **CD** is an internal **cmd.exe** command that has the same functionality as **CHDIR**.

***Compare (diff, cat)***

The diff command compares the differences between files line by line. It is a very useful command when we want to check the differences between files, probably between files and configuration files.

* recursive diff (compares subdirectories it finds)
* diff in contextual mode (-c)
* diff in unified mode (-u)

***Find files, folders and inside files (grep, cat, echo)***

You can use the -r option with grep to recursively search all files in a directory and its subdirectories for a specified pattern.

$ **grep** -r pattern / directory / to / search

If you do not specify a directory, grep will only search your current working directory.

Using the **cat command**, you can quickly create a file and add text to it. To do that, use the ">" operator to redirect the text in the file.

*cat> filename.txt*

This is how the file is created and you can start filling it with text. To add multiple lines of text, just hit **Enter** at the end of each line.

**Create and edit text files (nano, vim, touch, cat + ">")**

*NANO*

The [commands](https://www.sololinux.es/comandos-avanzados-en-linux-unix/) , or rather, the most common [**nano editor**](https://www.sololinux.es/los-mejores-editores-de-texto-en-consola-terminal/) shortcuts are the following:

* **Ctrl + g** ———- Open nano help.
* **Ctrl + x** ———- Exit nano.
* **Ctrl + or** ———- Save the current file.
* **Ctrl + r** ———- Insert another file in the current one.
* **Ctrl + w** ———- Find a text in the file you have open.
* **Ctrl + and** ———- Return to the previous page.
* **Ctrl + v** ———- Jump to the next page.
* **Ctrl + k** ———- Cut the selected line or region and save it in the cutbuffer.
* **Ctrl + u** ———- Paste what is saved in the cutbuffer on the current line.
* **Ctrl + l** ———- Reload the current screen.
* **Ctrl + j** ———- Justify the current paragraph.
* **Ctrl + m** ———- Insert a carriage return at the cursor position.
* **Ctrl + \_** ———- Jump to a specific line number.
* **M + g** ———- Go to a specific line number.
* **M + i** ———- Auto indent in addition to enabling and disabling.
* **M + x** ———- Enable / disable help mode.
* **M + p** ———- Enable / disable peak mode.
* **M + m** ———- Enable / disable mouse support.
* **M + r** ———- Replace text.
* **M + e** ———- Enable / disable regular expressions.
* **M + b** ———- Enable / disable file backup.
* **M + s** ———- Enable / disable smooth scrolling.
* **M + h** ———- Enable / disable the 'smart home' key.
* **M + and** ———- Enable / disable syntax coloring.
* **M + p** ———- Enable / disable show targets.

*VIM*

**Vim Essential Guide**

Now let's see the most useful and necessary commands so that our first steps with Vim are more bearable. **Remember that you have to write these commands in command mode and not in insert mode.**

For some of these commands you need to press the Enter key to actually run. In this case, the commands will appear at the bottom of the terminal, so that we can see what is going to be executed.

**Save and close.**

* ***: w* -** Allows you to save the file.
* ***: q*** - Exit Vim. If the file has been modified but has not been saved, it will warn us and we will not be able to exit Vim using this command.
* ***: q!***- Exit Vim, discarding possible unsaved changes that have been made to the file.
* ***: wq***- Save the file and then exit Vim.

**Undo and redo**

* ***u*** - Undo action.
* ***Ctrl + r*** - Redo an action.

**Move through the file.**

In addition to using the cursors to move around the file, we can move faster with some commands:

* ***gg*** - Go to the beginning of the file.
* ***Shift + g*** - Go to the last line of the file.
* ***Num + G*** - Go to a certain line. For example, 14G would bring the cursor to line 14.
* : set number - Makes the editor display the number of lines.
* ***$*** - Go to the end of the line.
* ***0*** - Go to the beginning of the line.

**Delete lines.**

* ***dd*** - The command allows you to delete the current line, where the cursor is.
* ***d + num*** - This command allows you to delete a number of lines. For example, d3 will delete three lines.

**Look for.**

Vim has some very powerful tools to find text in files. The most useful commands are:

* ***/ + text*** - Pressing «/» opens the search function. Then we can write the text we want to search for. The editor will highlight all occurrences of this text. We press enter and it will take us to the next appearance of the search, with respect to the position of our cursor.
* ***n*** and***N*** - Once we have accepted a search, the command***n*** takes us to the next occurrence of the search string. The***N*** command will take us to the previous one.

**Other aids**

With these commands we assure you that you will be able to defend yourself by Vim with some agility, so that you can carry out the most common editing tasks. Also, from Vim you can do the following:

* ***: h - Opens*** the main Vim help. This will make our terminal window split into two editors. At this point our cursor will be in the text of the Vim help file. We can read the file and use the search functions discussed above. When we want to exit we have to do it like any other file, using the corresponding Vim command, for example***: q*** .

*TOUCH*

The Linux touch command is mainly used to create empty files and change timestamps of files or folders. File timestamp information consists of three attributes: access time, modification time, and change time.

The syntax of the touch command is:

*touch [options] [filename]*

The following section lists the uses of the Linux touch command by including the various options available.

CAT + “>”

Using the cat command, you can quickly create a file and add text to it. To do that, use the ">" operator to redirect the text in the file.

*cat> filename.txt*

This is how the file is created and you can start filling it with text. To add multiple lines of text, just hit Enter at the end of each line. When we are done, press CTRL + D to exit the file.

**Get the state of the computer (history, top, whoami)**

The Gnu / Linux terminal stores its history in a file. This is called **' .bash\_history '** and is stored in the home directory, from where anyone can edit it. Since the terminal history file is stored in the user's directory, each will have a file.

Any user of the system can see the history of another with a simple command. So, for example, if we wanted to look at the command line history, we would just type something like the following in the terminal (Ctrl + Alt + T):

Cat bash history command

*cat /home/usuario/.bash\_history*

Users will also be able to see the history of the current user with which we log in to the terminal by executing:

*history*

Since the history is just a file, we will be able to search inside as in a normal text file using the grep utility. For example, to find instances of ' touch ' you would use something like the following:

*cat grep touch*

*cat /home/user/.bash\_history | grep 'touch'*

We open a console and simply execute the command:

***top***

An interface will appear in text mode that will be updated every 3 seconds. It shows a summary of the state of our system and the list of processes that are running.