**Displaying the file contents on the terminal:**

Cat(concatenate) command is very frequently used in Linux. It reads data from the file and gives their content as output. It helps us to create, view, concatenate files. So let us see some frequently used cat commands.

**1) To view a single file**

$cat filename

**2) To view multiple files**   
**Command:** 

$cat file1 file2

**3) To view contents of a file preceding with line numbers.**   
**Command:** 

$cat -n filename

**4) Create a file**  
**Command:** 

$ cat > newfile

**5) Copy the contents of one file to another file.**   
**Command:** 

$cat [filename-whose-contents-is-to-be-copied] > [destination-filename]

**6) Cat command can suppress repeated empty lines in output**   
**Command:** 

$cat -s coep.txt

**7) Cat command can append the contents of one file to the end of another file.**   
**Command:** 

$cat file1 >> file2

**8) Cat command can display content in reverse order using tac command.**   
**Command:** 

$tac filename

**9) Cat command can highlight the end of line.**   
**Command:** 

$cat -E "filename"

**10) If you want to use the -v, -E and -T option together, then instead of writing -vET in the command, you can just use the -A command line option.**   
Command 

$cat -A "filename"

**11) Cat command to open dashed files.**   
**Command:** 

$cat -- "-dashfile"

**12) Cat command if the file has a lot of content and can’t fit in the terminal.**   
**Command:**

$cat "filename" | more

**13) Cat command to merge the contents of multiple files.**   
**Command:** 

$cat "filename1" "filename2" "filename3" > "merged\_filename"

**14) Cat command to display the content of all text files in the folder.**   
**Command:** 

$cat \*.txt

**15) Cat command to write in an already existing file.**

**Command :**

$cat >> coep.txt

The newly added text.

**2). File and Directory Manipulation Commands:**

[**mkdir**](https://www.geeksforgeeks.org/mkdir-command-in-linux-with-examples/) : Used to create a directory if not already exist. It accepts the directory name as an input parameter.

**mkdir** command in Linux allows the user to create directories (also referred to as folders in some operating systems ). This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have enough permissions to create a directory in the parent directory, or he/she may receive a ‘permission denied’ error.

**Syntax:**

mkdir [options...] [directories ...]

* **–version:** It displays the version number, some information regarding the license and exits.   
  **Syntax:**

mkdir --version

* **–help**: It displays the help related information and exits.   
  **Syntax:**

mkdir --help

* **-v or –verbose**: It displays a message for every directory created.   
  **Syntax:**

mkdir -v [directories]

* **-p**: A flag which enables the command to create parent directories as necessary. If the directories exist, no error is specified.

**Syntax:**

mkdir -p [directories]

* Suppose you execute the following command –

mkdir -p first/second/third

* If the first and second directories do not exist, due to the -p option, mkdir will create these directories for us. If we do not specify the -p option, and request the creation of directories, where parent directory doesn’t exist,
* **-m**: This option is used to set the file modes, i.e. permissions, etc. for the created directories. The syntax of the mode is the same as the **chmod** command.

**Syntax:**

mkdir -m a=rwx [directories]

The above syntax specifies that the directories created give access to all the users to read from, write to and execute the contents of the created directories. You can use ‘a=r’ to only allow all the users to read from the directories and so on.

[**cp**](https://www.geeksforgeeks.org/cp-command-linux-examples/) : This command will copy the files and directories from the source path to the destination path. It can copy a file/directory with the new name to the destination path. It accepts the source file/directory and destination file/directory.

**cp** stands for **copy**. This command is used to copy files or group of files or directory. It creates an exact image of a file on a disk with different file name. cp command require at least two filenames in its arguments.

**Syntax:**

**cp [OPTION] Source Destination**

**cp [OPTION] Source Directory**

**cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory**

First and second syntax is used to copy Source file to Destination file or Directory.

Third syntax is used to copy multiple Sources(files) to Directory.

**cp command works on three principal modes of operation and these operations depend upon number and type of arguments passed in cp command :**

1. **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. So be careful when you choose destination file name.
2. **p Src\_file Dest\_file**
3. Suppose there is a directory named college having a text file **a.txt**.  
   **Example:**
4. **$ ls**
5. a.txt
6. **$ cp a.txt b.txt**
7. **$ ls**
8. a.txt b.txt

**One or more arguments :** If the command has one or more arguments, specifying file names and following those arguments, an argument specifying directory name then this command copies each source file to the destination directory with the same name, created if not existed but if already existed then it will be overwritten, so be careful !!.

**cp Src\_file1 Src\_file2 Src\_file3 Dest\_directory**

Suppose there is a directory named *COEPPUNE* having a text file **a.txt, b.txt** and a directory name **new** in which we are going to copy all files.

**$ ls**

a.txt b.txt new

Initially new is empty

**$ ls new**

**$ cp a.txt b.txt new**

**$ ls new**

a.txt b.txt

**Note:** For this case last argument must be a **directory** name. For the above command to work, Dest\_directory must exist because **cp** command

 won’t create it.

 **Two directory names :** If the command contains two directory names, **cp** copies all files of the source directory to the destination directory, creating any files or directories needed. This mode of operation requires an additional option, typically **R**, to indicate the recursive copying of directories.

**cp -R Src\_directory Dest\_directory**

1. In the above command, **cp** behavior depend upon whether *Dest\_directory* is exist or not. If the *Dest\_directory* doesn’t exist, cp creates it and copies content of *Src\_directory* recursively as it is. But if Dest\_directory exists then copy of *Src\_directory* becomes sub-directory under *Dest\_directory*.

**Options:**

There are many options of **cp** command, here we will discuss some of the useful options:  
Suppose a directory named **college** contains two files having some content named as **a.txt** and **b.txt**. This scenario is useful in understanding the following options.

**$ ls college**

a.txt b.txt

**$ cat a.txt**

GFG

**$ cat b.txt**

College of Engineering Pune

**1. -i(interactive): i** stands for Interactive copying. With this option system first warns the user before overwriting the destination file. **cp** prompts for a response, if you press **y** then it overwrites the file and with any other option leave it uncopied.

**$ cp -i a.txt b.txt**

**cp: overwrite 'b.txt'? y**

**$ cat b.txt**

GFG

**2. -b(backup):** With this option **cp** command creates the backup of the destination file in the same folder with the different name and in different format.

**$ ls**

a.txt b.txt

**$ cp -b a.txt b.txt**

**$ ls**

a.txt b.txt b.txt~

**3. -f(force):** If the system is unable to open destination file for writing operation because the user doesn’t have writing permission for this file then by using **-f** option with **cp** command, destination file is deleted first and then copying of content is done from source to destination file.

**$ ls -l b.txt**

-r-xr-xr-x+ 1 User User 3 Nov 24 08:45 b.txt

User, group and others doesn't have writing permission.

Without -f option, command not executed

**$ cp a.txt b.txt**

**cp: cannot create regular file 'b.txt': Permission denied**

With -f option, command executed successfully

**$ cp -f a.txt b.txt**

**Copying using \* wildcard:** The star wildcard represents anything i.e. all files and directories. Suppose we have many text document in a directory and wants to copy it another directory, it takes lots of time if we copy files 1 by 1 or command becomes too long if specify all these file names as the argument, but by using \* wildcard it becomes simple.

Initially Folder1 is empty

**$ ls**

a.txt b.txt c.txt d.txt e.txt Folder1

**$ cp \*.txt Folder1**

**$ ls Folder1**

a.txt b.txt c.txt d.txt e.txt