# ls command

# lsis a Linux shell command that lists directory contents of files and directories.

# syntax

$ ls [*options*] [*file*|*dir*]

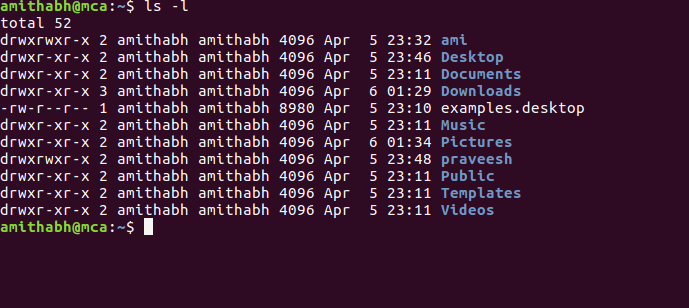
## *ls command options*

## ls -l command

ls -l option flag lists with long listing format.

**Syntax**

$ ls -l [*options*] [*file*|*dir*]

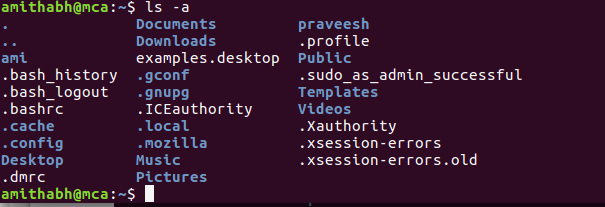


# ls -a command

ls -a option flag lists all files including hidden files starting with '.'

### Syntax

$ ls -a [*options*] [*file*|*dir*]

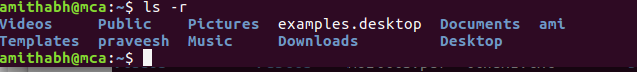


**ls -r command**

ls -r option flag lists files/directories in reverse order.

**Syntax**

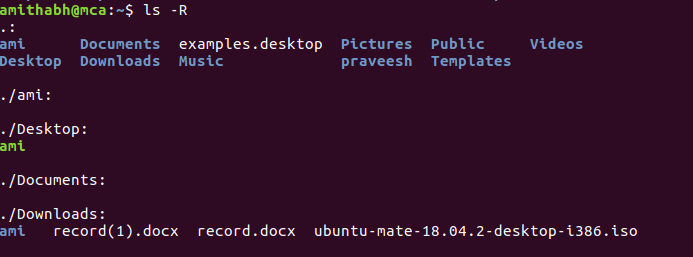
$ ls -r [*options*] [*file*|*dir*]



## ls -R Command

ls -R option flag lists directory tree recursively.**Syntax**

$ ls -R [*options*] [*file*|*dir*]

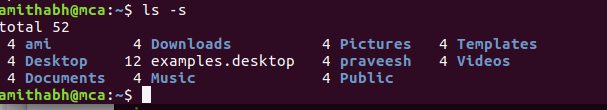


## ls -s command

ls -s option flag lists file size.

### Syntax

$ ls -s [*options*] [*file*|*dir*]

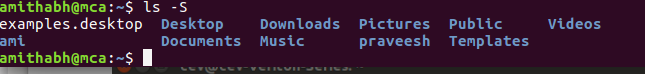


## ls -S command

ls -S option flag sorts files/directories list by file size.

### Syntax

$ ls -S [*options*] [*file*|*dir*]

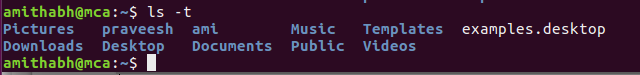


# ls -t command

ls -t option flag sorts files/directories list by time/date.

### Syntax

$ ls -t [*options*] [*file*|*dir*]



# mkdir command

mkdircommand 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 recieve a ‘permission denied’ error.

## Syntax

mkdir [OPTION]... DIRECTORY...

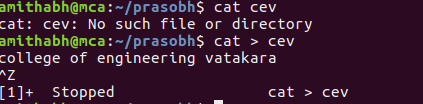


# Cat command

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

**Syntax**$

cat >newfile



# mv command

mv stands for move. mv is used to move one or more files or directories from one place to another in file system like UNIX. It has two distinct functions:  
(i) It rename a file or folder.  
(ii) It moves group of files to different directory.  
No additional space is consumed on a disk during renaming. This command normally works silently means no prompt for confirmation.

Syntax:

mv [Option] source destination

# cp command

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.

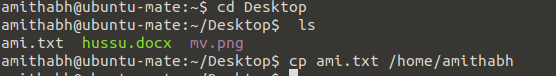
Syntax:

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.



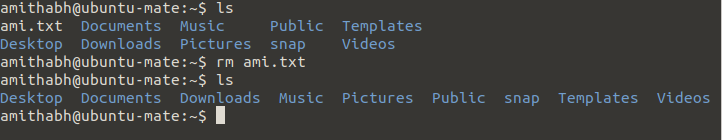
# rm command

rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the filesystem, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories.

This command normally works silently and you should be very careful while running rm command because once you delete the files then you are not able to recover the contents of files and directories.

Syntax:

rm [OPTION]... FILE...



# chmod command

In Unix-like operating systems, the chmod command is used to change the access mode of a file.  
The name is an abbreviation of change mode.

**Syntax**

chmod [reference][operator][mode] file...

# 

# pwd command

pwd stands for Print Working Directory. It prints the path of the working directory, starting from the root.

pwd is shell built-in command(pwd) or an actual binary(/bin/pwd).  
$PWD is an [environment variable](https://www.geeksforgeeks.org/environment-variables-in-linux-unix/) which stores the path of the current directory.

**Syntax**

pwd -L: Prints the symbolic path.  
pwd -P: Prints the actual path.



# chown command

Different users in the operating system have ownership and permission to ensure that the files are secure and put restrictions on who can modify the contents of the files. In Linux there are different users who use the system:

* Each user has some properties associated with them, such as a user ID and a home directory. We can add users into a group to make the process of managing users easier.
* A group can have zero or more users. A specified user can be associated with a “default group”. It can also be a member of other groups on the system as well.

Ownership and Permissions: To protect and secure files and directory in Linux we use permissions to control what a user can do with a file or directory. Linux uses three types of permissions:

* Read: This permission allows the user to read files and in directories, it lets the user read directories and subdirectories stores in it.
* Write: This permission allows a user to modify and delete a file. Also it allows a user to modify its contents (create, delete and rename files in it) for the directories. Unless the execute permission is not given to directories changes does do affect them.
* Execute: The write permission on a file allows it to get executed. For example, if we have a file named php.sh so unless we don’t give it execute permission it won’t run.

Types of file Permissions:

* User: These type of file permission affect the owner of the file.
* Group: These type of file permission affect the group which owns the file. Instead of the group permissions, the user permissions will apply if the owner user is in this group.
* Other: These type of file permission affect all other users on the system.

Note: To view the permissions we use:

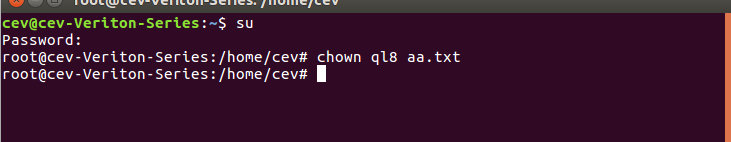
ls -l

chown command is used to change the file Owner or group. Whenever you want to change ownership you can use chown command.

Syntax

chown [OPTION]… [OWNER][:[GROUP]] FILE…

chown [OPTION]… –reference=RFILE FILE…

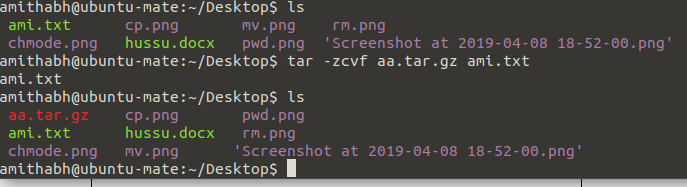


# tar command

The Linux ‘tar’ stands for tape archive, is used to create Archive and extract the Archive files. tar command in Linux is one of the important command which provides archiving functionality in Linux. We can use Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

Syntax:

tar [options] [archive-file] [file or directory to be archiv



# grep command

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for globally search for regular expression and print out).

Syntax:

grep [options] pattern [files]

# grep

# history command

**history**command is used to view the previously executed command. This feature was not available in the Bourne shell. Bash and Korn support this feature in which every command executed is treated as the event and is associated with an event number using which they can be recalled and changed if required. These commands are saved in a history file. In Bash shell **history** command shows the whole list of the command.

Syntax:

$ history

# find command

The **find** command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the ‘-exec’ other UNIX commands can be executed on files or folders found.

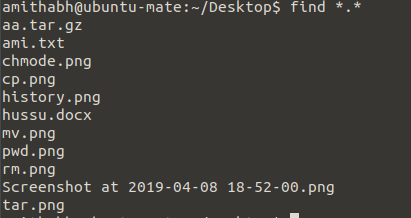
Syntax :

$ find [where to start searching from]

[expression determines what to find] [-options] [what to find]

Options :

* -exec CMD: The file being searched which meets the above criteria and returns 0 for as its exit status for sucessful command execution.
* -ok CMD : It works same as -exec except the user is prompted first.
* -inum N : Search for files with inode number ‘N’.
* -links N : Search for files with ‘N’ links.
* -name demo : Search for files that are specified by ‘demo’.
* -newer file : Search for files that were modified/created after ‘file’.
* -perm octal : Search for the file if permission is ‘octal’.
* -print : Display the path name of the files found by using the rest of the criteria.
* -empty : Search for empty files and directories.
* -size +N/-N : Search for files of ‘N’ blocks; ‘N’ followed by ‘c’can be used to measure size in characters; ‘+N’ means size > ‘N’ blocks and ‘-N’ means size <'N' blocks.
* -user name : Search for files owned by user name or ID ‘name’.
* \(expr \) : True if ‘expr’ is true; used for grouping criteria combined with OR or AND.
* ! expr : True if ‘expr’ is false.

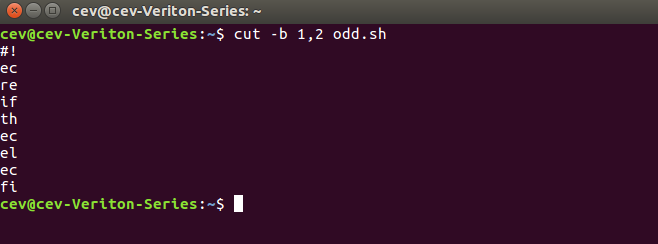


# cut command

The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field. Basically the cut command slices a line and extracts the text. It is necessary to specify option with command otherwise it gives error. If more than one file name is provided then data from each file is not precedes by its file name.

Syntax:

cut OPTION... [FILE]...

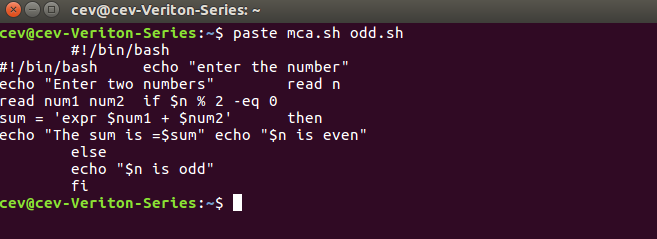


# Paste command

Paste command is one of the useful commands in Unix or Linux operating system. It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output. When no file is specified, or put dash (“-“) instead of file name, paste reads from standard input and gives output as it is until a interrupt command [Ctrl-c] is given.

Syntax:

paste [OPTION]... [FILES]...

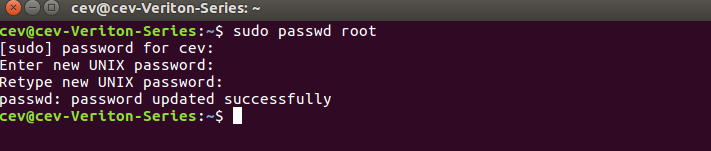


# passwd command

passwd command in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.

Syntax:

passwd [options] [username]

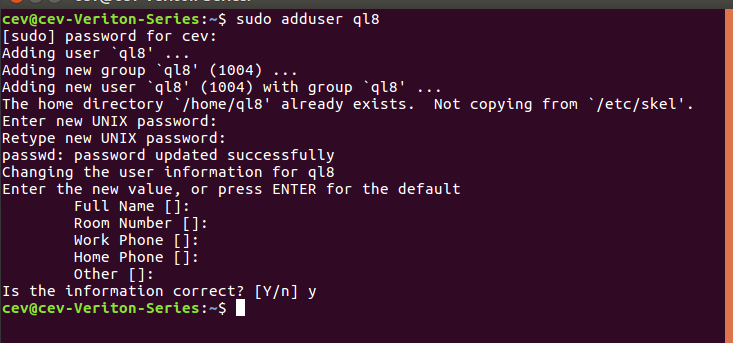


# groupadd command

[Groups](https://www.geeksforgeeks.org/groups-in-linux-system-administration/) in Linux refer to the user groups. In Linux, there can be many users of a single system, (normal user can take uid from 1000 to 60000, and one root user (uid 0) and 999 system users (uid 1 to 999)). In a scenario where there are many users, there might be some privileges that some users have and some don’t, and it becomes difficult to manage all the permissions at the individual user level. So using groups, we can group together a number of users, and set privileges and permissions for the entire group. groupadd command is used to create a new user group.

Syntax:

groupadd [option] group\_name

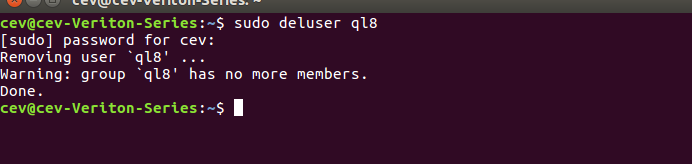


# groupdel command

groupdel command is used to delete a existing group. It will delete all entry that refers to the group, modifies the system account files, and it is handled by superuser or root user.

Syntax:

groupdel [options] GROUP



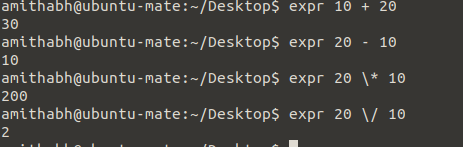
# expr command

The expr command in Unix evaluates a given expression and displays its corresponding output. It is used for:

* Basic operations like addition, subtraction, multiplication, division, and modulus on integers.
* Evaluating regular expressions, string operations like substring, length of strings etc.

Syntax:

$expr expression



**ENVIRONMENTAL VARIABLES**

echo "Display : $DISPLAY"

echo "User : $USER"

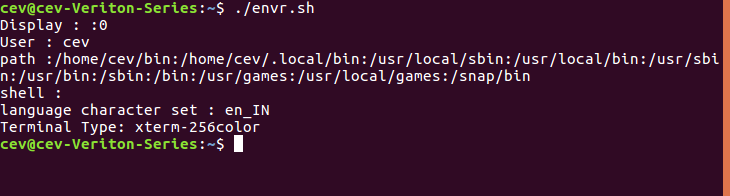
echo "path :$PATH"

echo "shell :$shell"

echo "language character set : $LANG"

echo "Terminal Type: $TERM"

**OUTPUT**



**SHELL PROFEAMMING**

**1. Read the name and print it**

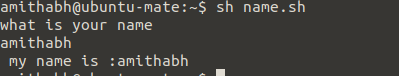
#!/bin/bash

echo "what is your name"

read name

echo " my name is :"$name

**OUTPUT**



**2.Read two numbers and find the sum**

#!/bin/bash

echo -n "Enter the first number : "

read num1

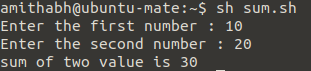
echo -n "Enter the second number : "

read num2

sum=`expr $num1 + $num2`

echo "sum of two value is $sum"

**OUTPUT**



1. **Find the greatest of 3 numbers.**

echo "Enter Num1"

read num1

echo "Enter Num2"

read num2

echo "Enter Num3"

read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]

then

echo "greatest number is:" $num1

elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]

then

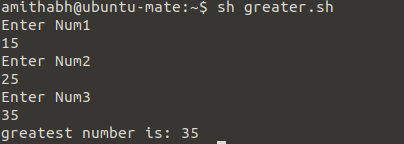
echo "greatest number is:" $num2

else

echo "greatest number is:" $num3

fi

**OUTPUT**



**4.Find the area of a circle.**

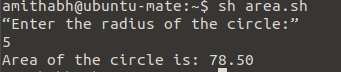
echo “Enter the radius of the circle:”

read radius

area=$(echo "3.14\*($radius\*$radius)" | bc)

echo "Area of the circle is:" $area

**OUTPUT**



**5.Check whether a given number is odd or even**

echo -n "Enter numnber : "

read n

rem=$(( $n % 2 ))

if [ $rem -eq 0 ]

then

echo "$n is even number"

else

echo "$n is odd number"

fi

**OUTPUT**



**6.Find the sum of n numbers using for loop**

echo "Enter Size(N)"

read n

sum=0

echo "Enter Numbers"

for i in $(seq 1 $n)

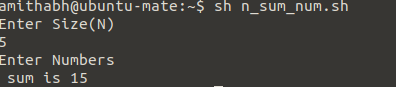
do

sum=$(( $sum + $i))

done

echo " sum is $sum"

**OUTPUT**

****

**7.Find the sum of n numbers using while loop**

#!/bin/bash

echo "enter the number"

read n

i=1

sum=0

a=$n

while [ $i -le $n ]

do

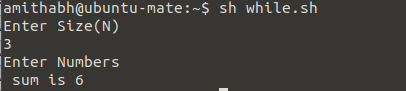
sum=$(( $sum + $i))

i=$(( $i + 1))

done

echo "the sum of $n is $sum"

**OUTPUT**



**8.Find the numbers between 1 to 10**

echo "finding even numbers range in 1 to 10"

for n in $(seq 1 10)

do

rem=$(( $n % 2 ))

if [ $rem -eq 0 ]

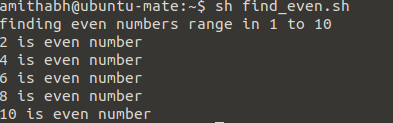
then

echo "$n is even number"

fi

done

**OUTPUT**



**Scheduling using cron and at**

**Linux Crontab Format**

|  |
| --- |
| MIN HOUR DOM MON DOW CMD |

**Crontab Fields and Allowed Ranges (Linux Crontab Syntax)**

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **Allowed Value** |
| MIN | Minute field | 0 to 59 |
| HOUR | Hour field | 0 to 23 |
| DOM | Day of Month | 1-31 |
| MON | Month field | 1-12 |
| DOW | Day Of Week | 0-6 |
| CMD | Command | Any command to be executed. |

**1. Scheduling a Job For a Specific Time**

The basic usage of cron is to execute a job in a specific time as shown below. This will execute the Full backup shell script (full-backup) on 10th June 08:30 AM.

The time field uses 24 hours format. So, for 8 AM use 8, and for 8 PM use 20.

|  |
| --- |
| 30 08 10 06 \* /home/maverick/full-backup |

30 – 30th Minute  
08 – 08 AM  
10 – 10th Day  
06 – 6th Month (June)  
\* – Every day of the week

**2.To schedule a job for more than one time (e.g. Twice a Day)**

The following script take a incremental backup twice a day every day.

This example executes the specified incremental backup shell script (incremental-backup) at 11:00 and 16:00 on every day. The comma separated value in a field specifies that the command needs to be executed in all the mentioned time.

|  |
| --- |
| 00 11, 16 \* \* \* /home/maverick/bin/incremental-backup |

00 – 0th Minute (Top of the hour)  
11, 16 – 11 AM and 4 PM  
\* – Every day  
\* – Every month  
\* – Every day of the week

1. **To schedule a job for certain range of time (e.g. Only on Weekdays)**

If you wanted a job to be scheduled for every hour with in a specific range of time then use the following.

Cron Job everyday during working hours :  
This example checks the status of the database everyday (including weekends) during the working hours 9 a.m – 6 p.m

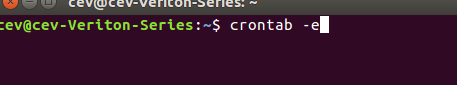
|  |
| --- |
| 00 09-18 \* \* \* /home/maverick/bin/check-db-status |

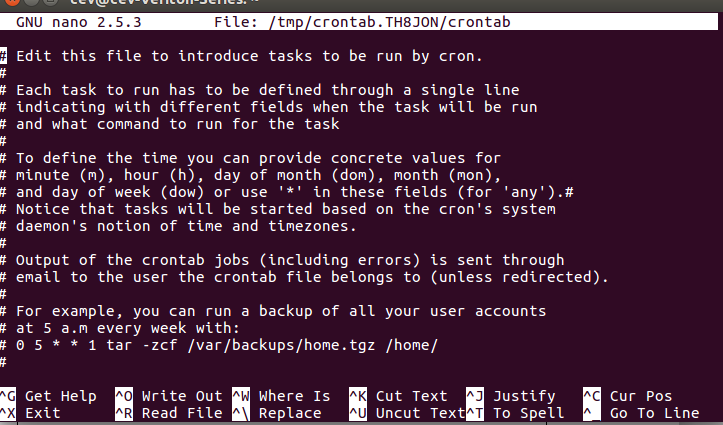
00 – 0th Minute (Top of the hour)  
09-18 – 9 am, 10 am, 11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm  
\* – Every day  
\* – Every month  
\* – Every day of the week

Cron Job every weekday during working hours :  
This example checks the status of the database every weekday (i.e excluding Sat and Sun) during the working hours 9 a.m – 6 p.m.

|  |
| --- |
| 00 09-18 \* \* 1-5 /home/maverick/bin/check-db-status |

00 – 0th Minute (Top of the hour)  
09-18 – 9 am, 10 am, 11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm  
\* – Every day  
\* – Every month  
1-5 -Mon, Tue, Wed, Thu and Fri (Every Weekday)





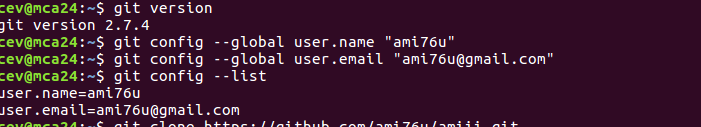
**GIT**

Git is an **Open Source Distributed Version Control System**. Now that’s a lot of words to define Git.

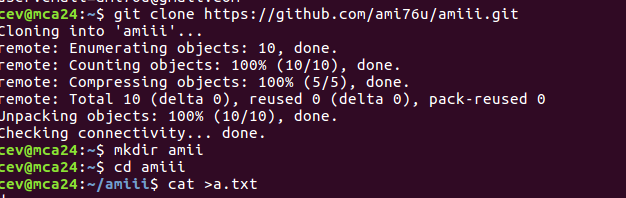
Let me break it down and explain the wording:

* **Control System:** This basically means that Git is a content tracker. So Git can be used to store content — it is mostly used to store code due to the other features it provides.
* **Version Control System**: The code which is stored in Git keeps changing as more code is added. Also, many developers can add code in parallel. So Version Control System helps in handling this by maintaining a history of what changes have happened. Also, Git provides features like branches and merges, which I will be covering later.
* **Distributed Version Control System**: Git has a remote repository which is stored in a server and a local repository which is stored in the computer of each developer. This means that the code is not just stored in a central server, but the full copy of the code is present in all the developers’ computers. Git is a Distributed Version Control System since the code is present in every developer’s computer. I will explain the concept of remote and local repositories later in this article.

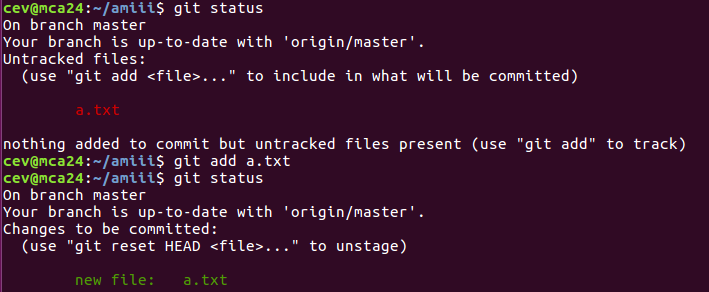
**Git Config**



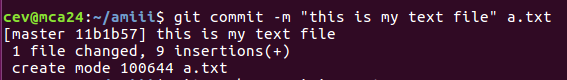
**Git Clone**



**Git Status**



**Git Commit**



**Git Push**

