Experiment No.2

Aim

Study of a terminal based text editor such as Vim or Emacs. Basic Linux commands, familiarity with following commands/operations expected

1. man
2. ls, echo, read
3. more, less, cat,
4. cd, mkdir, pwd, find
5. mv, cp, rm ,tar
6. wc, cut, paste
7. head, tail, grep, expr 8 chmod, chown
9. Redirections & Piping
10. useradd, usermod, userdel, passwd
11. df,top, ps
12. ssh, scp, ssh-keygen, ssh-copy-id

Result

Text Editor

A text editor is program that allows you to open, view, and edit plain text files. Unlike word processors, text editors do not add formatting to text, instead focusing on editing functions for plain text. Text editors are used by a wide variety of people, for a wide variety of purposes

Vim is a highly configurable text editor built to enable efficient text editing. Vim is acronym for Vi IMproved. It is an improved version of the vi editor distributed with most UNIX systems .Vim is often called a "programmer's editor" .It is free and open source text editor written by Bram Moolenaar. It was first released in 1991 for UNIX variants and its main goal was to provide enhancement to the Vi editor, which was released way back in 1976.

Some common Unix Text editors are : vim, nano, GUI editors , emacs, mousepad , xedit , Pico, Emacs , Xemacs – a GUI-based version of Emacs., vi – a mostly terminal-based text editor

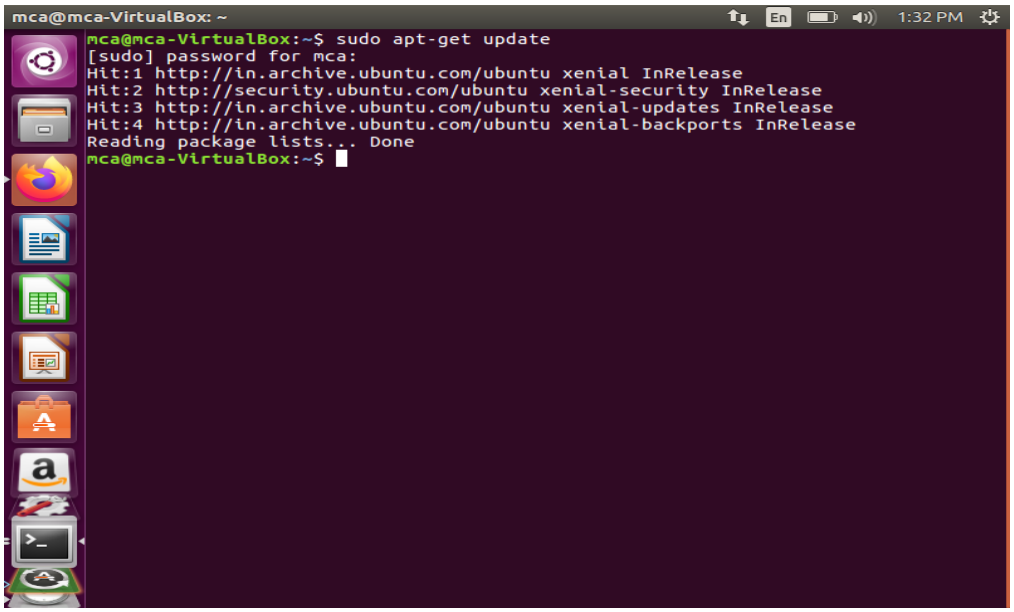
Some of the important features of Vim −

- Its memory footprint is very low

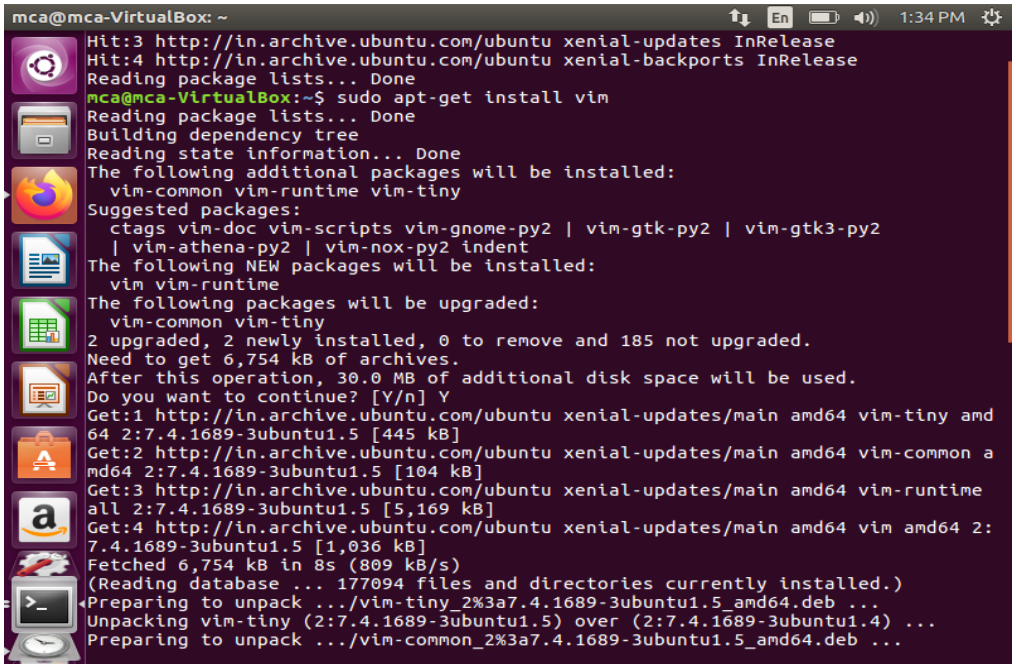
- It is command centric. You can perform complex text related task with few commands
- It is highly configurable and uses simple text file to store its configuration
- There are many plug-in available for Vim. Its functionality can be extended in great manner using these plug-in
- It supports multiple windows. Using this feature screen can be split into multiple windows
- Same as multiple windows, it also supports multiple buffers
- It supports multiple tabs which allows to work on multiple files

Vim Installation

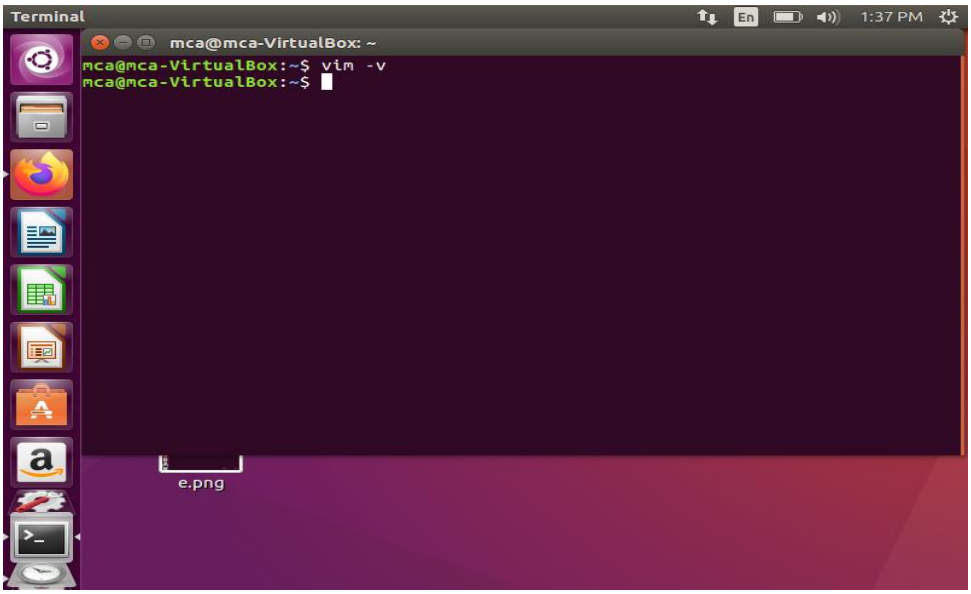
➢ Open terminal and type the command *sudo apt-get update*

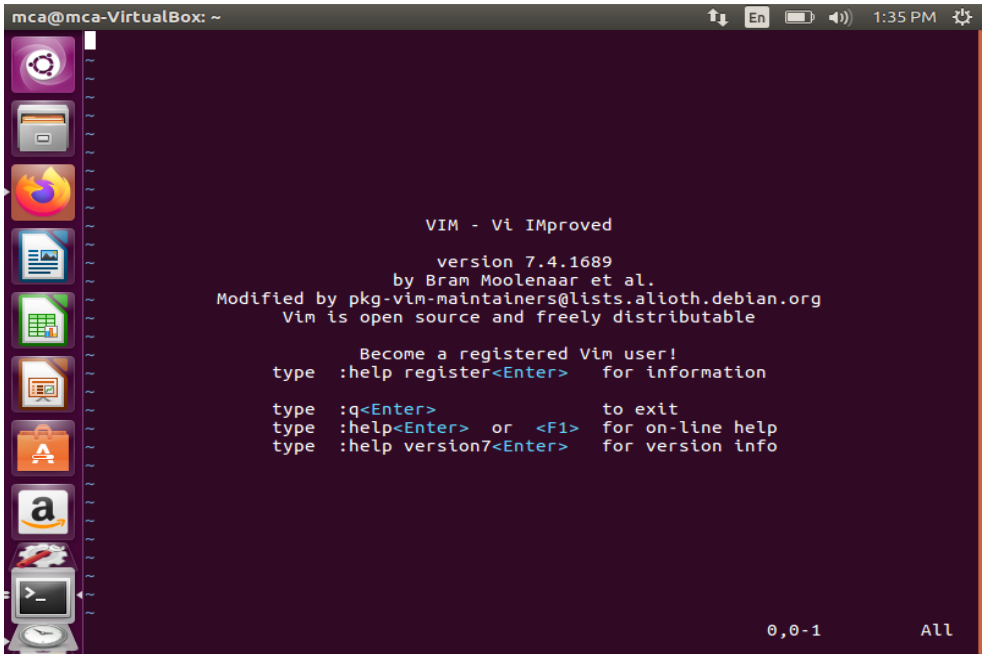


➢ Install vim using the command *sudo apt-get install vim*



➢ Then type *vim –v*





Modes in VIM editor

There are three modes in vim editor.

- Command Mode
- Insert Mode
- Last Line Mode

Command mode

When we first start editing a file using the Vim editor, the editor will be opened in a command mode.We can issue many commands that allow us to insert, append, delete text or search and navigate within our file.When using command mode we cannot insert text immediately.We first need to issue an insert(i), append(a), or open(o) command to insert the text in the file.

Insert mode

When we issue an insert,append,or open command, we will be in insert mode.Once in an insert mode we can type text into our file or navigate within the file.We can toggle between the command mode and the insert mode by pressing the ESC key.

Last Line Mode

The last line mode normally is used to perform operations like quitting the Vim session or savig a file.To go to the last line mode we first need to be in the command mode.From command mode we can go to last line mode by pressing the colon(:) key.After pressing this key,we will see a colon character at the beginning of the last line of our editor window with a

cursor blinking near it. This indicates that the editor is ready for accepting a ‘last line command’.

It is possible to toggle back to the command mode from the last line mode by pressing the ESC key twice or by pressing the backspace key until the initial ‘ : ‘ character is gone along with all the characters that we had typed or by simply pressing the ENTER key.

Vim Commands

1.<u>Basic Vim commands</u> : The most simple commands allow you to open and close documents as well as saving them.

- :help [keyword] - Performs a search of help documentation for whatever keyword you enter
- :e [file] - Opens a file, where [file] is the name of the file you want opened
- :w - Saves the file you are working on
- :w [filename] - Allows you to save your file with the name you've defined
- :wq - Save your file and close Vim
- :q! - Quit without first saving the file you were working on

2. <u>Vim commands for movement</u> : In movement commands, putting a number in front of the command can increase the number of times a task is completed.

- h - Moves the cursor to the left
- l - Moves the cursor to the right
- j - Moves the cursor down one line
- k - Moves the cursor up one line
- H - Puts the cursor at the top of the screen
- M - Puts the cursor in the middle of the screen
- L - Puts the cursor at the bottom of the screen
- w - Puts the cursor at the start of the next word
- b - Puts the cursor at the start of the previous word
- e - Puts the cursor at the end of a word
- 0 - Places the cursor at the beginning of a line
- $ - Places the cursor at the end of a line
-) - Takes you to the start of the next sentence
- (- Takes you to the start of the previous sentence
- } - Takes you to the start of the next paragraph or block of text
- { - Takes you to the start of the previous paragraph or block of text
- Ctrl + f - Takes you one page forward
- Ctrl + b - Takes you one page back
- gg - Places the cursor at the start of the file
- G - Places the cursor at the end of the file

- # - Where # is the number of a line, this command takes you to the line specified

3.<u>Vim commands for editing</u> : The command for copying a word is yw, which stands for yank word, and the command for pasting whatever has been copied is p, meaning put.

- yy - Copies a line
- yw - Copies a word
- y$ - Copies from where your cursor is to the end of a line
- v - Highlight one character at a time using arrow buttons or the h, k, j, l buttons
- V - Highlights one line, and movement keys can allow you to highlight additional lines
- p - Paste whatever has been copied to the unnamed register
- d - Deletes highlighted text
- dd - Deletes a line of text
- dw - Deletes a word
- D - Deletes everything from where your cursor is to the end of the line
- d0 - Deletes everything from where your cursor is to the beginning of the line
- dgg - Deletes everything from where your cursor is to the beginning of the file
- dG - Deletes everything from where your cursor is to the end of the file
- x - Deletes a single character
- u - Undo the last operation; u# allows you to undo multiple actions
- Ctrl + r - Redo the last undo
- . - Repeats the last action

4. <u>Vim commands for searching text</u> : Vim allows you to search your text and find and replace text within your document.

- /[keyword] - Searches for text in the document where keyword is whatever keyword, phrase or string of characters you're looking for
- ?[keyword] - Searches previous text for your keyword, phrase or character string
- n - Searches your text again in whatever direction your last search was
- N - Searches your text again in the opposite direction
- :%s/[pattern]/[replacement]/g - This replaces all occurrences of a pattern without confirming each one
- :%s/[pattern]/[replacement]/gc - Replaces all occurrences of a pattern and confirms each one

5. <u>Vim commands for working with multiple files</u> : We can also edit more than one text file at a time. Vim gives you the ability to either split your screen to show more than one file at a time or you can switch back and forth between documents.

- :bn - Switch to next buffer
- :bp - Switch to previous buffer
- :bd - Close a buffer
- :sp [filename] - Opens a new file and splits your screen horizontally to show more than one buffer
- :vsp [filename] - Opens a new file and splits your screen vertically to show more than one buffer
- :ls - Lists all open buffers
- Ctrl + ws - Split windows horizontally
- Ctrl + wv - Split windows vertically
- Ctrl + ww - Switch between windows
- Ctrl + wq - Quit a window
- Ctrl + wh - Moves your cursor to the window to the left
- Ctrl + wl - Moves your cursor to the window to the right
- Ctrl + wj - Moves your cursor to the window below the one you're in
- Ctrl + wk - Moves your cursor to the window above the one you're in

6. Marking text (visual mode) : Visual mode allows you to select a block of text in Vim. Once a block of text is selected you can use visual commands to perform actions on the selected text such as deleting it, copying it, etc.

- v - starts visual mode, you can then select a range of text, and run a command
- V - starts linewise visual mode (selects entire lines)
- Ctrl + v - starts visual block mode (selects columns)
- ab - a block with ()
- aB - a block with {}
- ib - inner block with ()
- iB - inner block with {}
- aw - mark a word
- Esc - exit visual mode

Once you've selected a particular range of text, you can then run a command on that text such as the following:

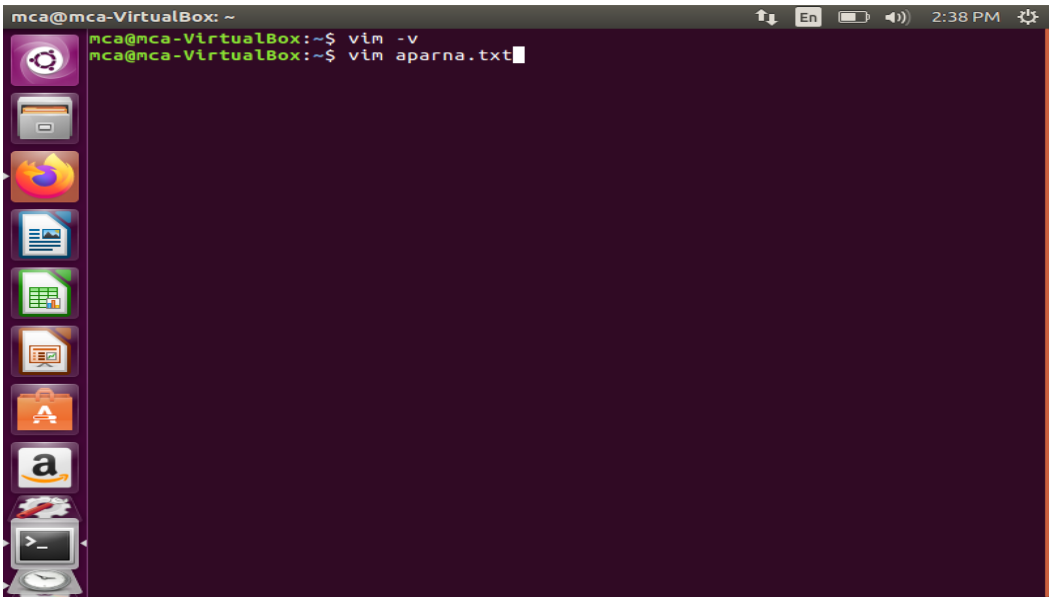
- d - delete marked text
- y - yank (copy) marked text
- shift text right
- < - shift text left
- ~ - swap case (upper or lower)

7. Tab pages : Just like any browser, you can also use tabs within Vim. This makes it incredibly easy to switch between multiple files while you're making some code changes instead of working in one single file, closing it, and opening a new one. Below are some useful Vim commands for using tab pages:

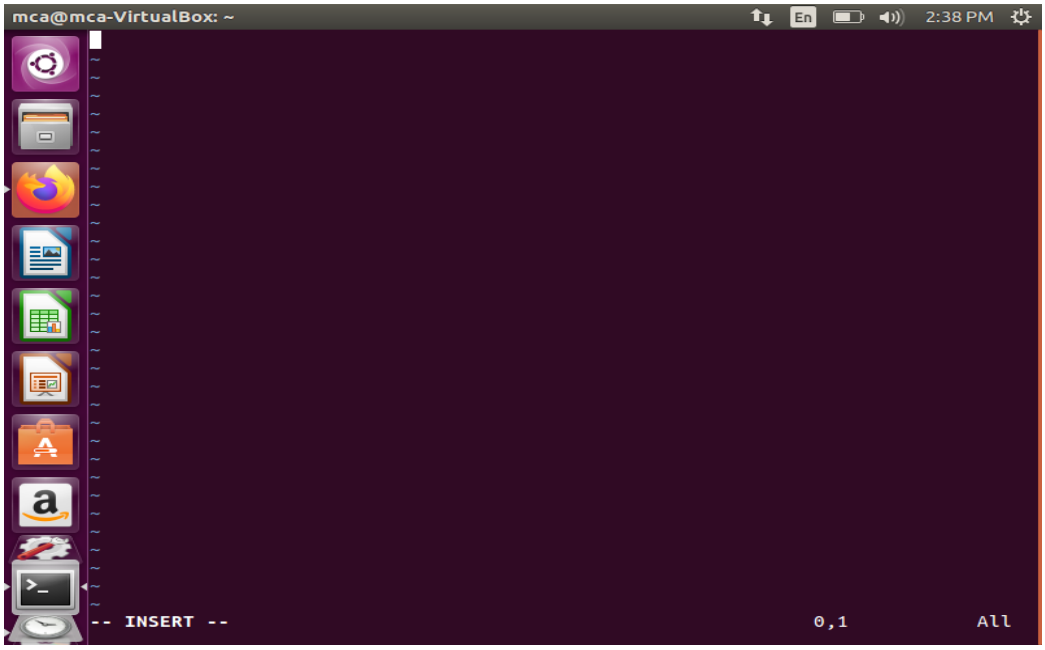
- :tabedit file - opens a new tab and will take you to edit "file"
- gt - move to the next tab
- gT - move to the previous tab
- #gt - move to a specific tab number (e.g. 2gt takes you to the second tab)
- :tabs - list all open tabs
- :tabclose - close a single tab

Simple Vim workflow example

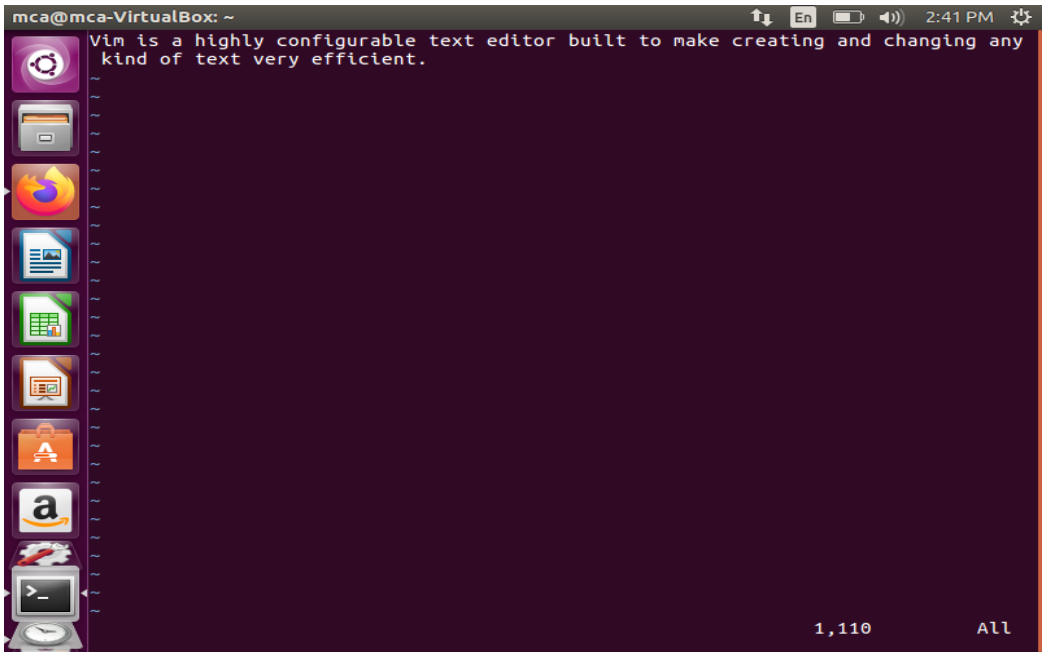
➢ Open a new or existing file with *vim filename*.



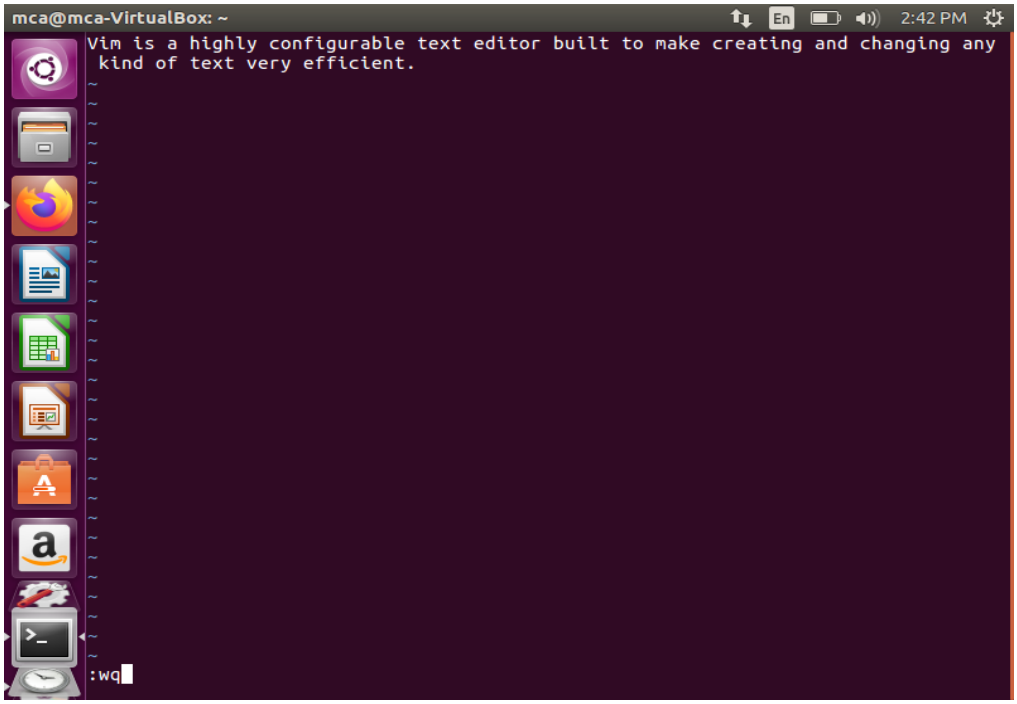
➢ Type i to switch into insert mode so that you can start editing the file. Enter or modify the text with your file.

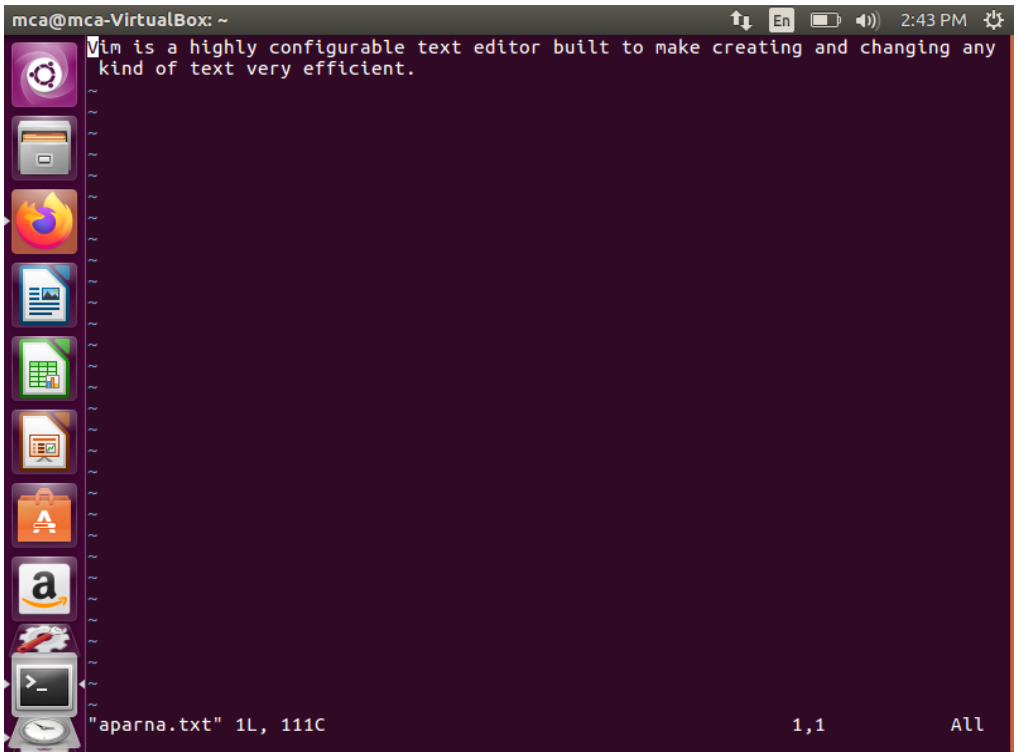


➢ Once you're done, press the escape key Esc to get out of insert mode and back to command mode.



➢ Type :wq to save and exit your file.





BASIC LINUX COMMANDS

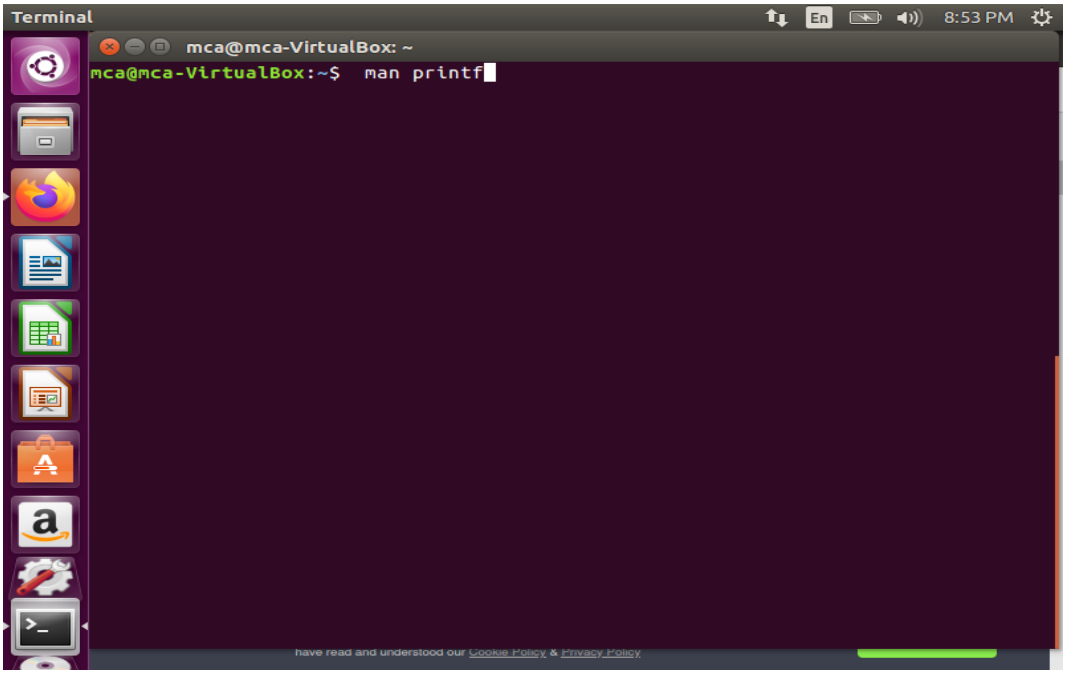
1. man

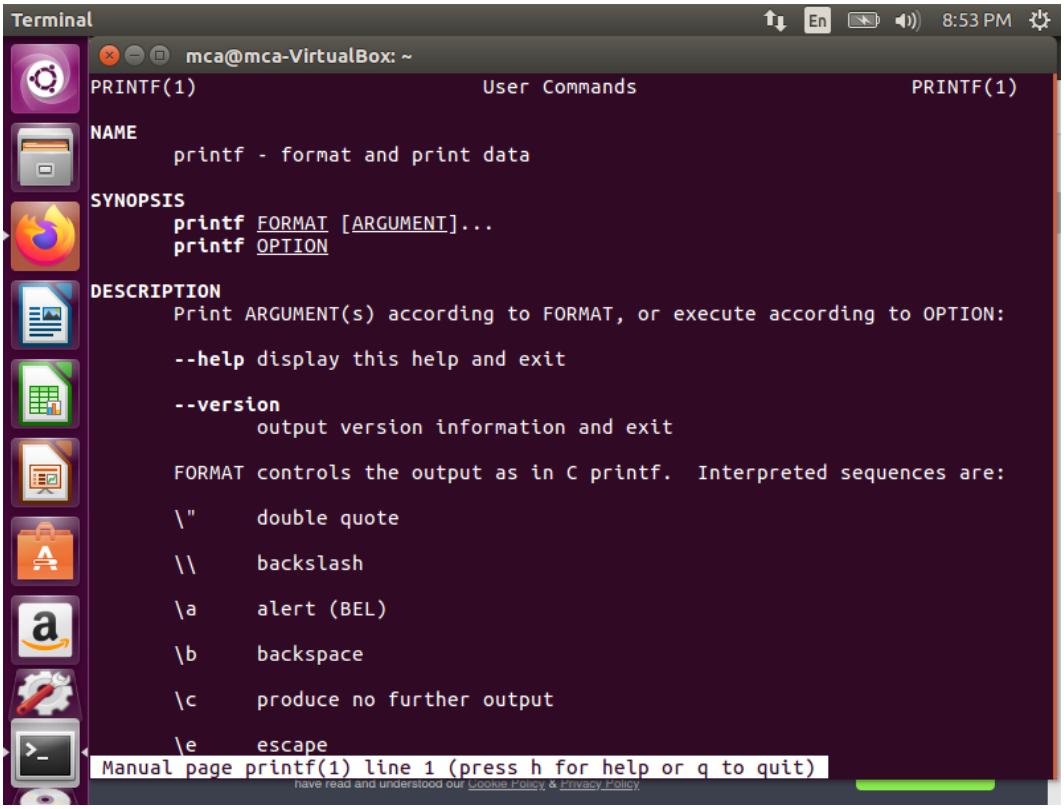
man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes *NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS* and *SEE ALSO.*

Syntax : *$ man [OPTION]... [COMMAND NAME]...*

- No Option: It displays the whole manual of the command.

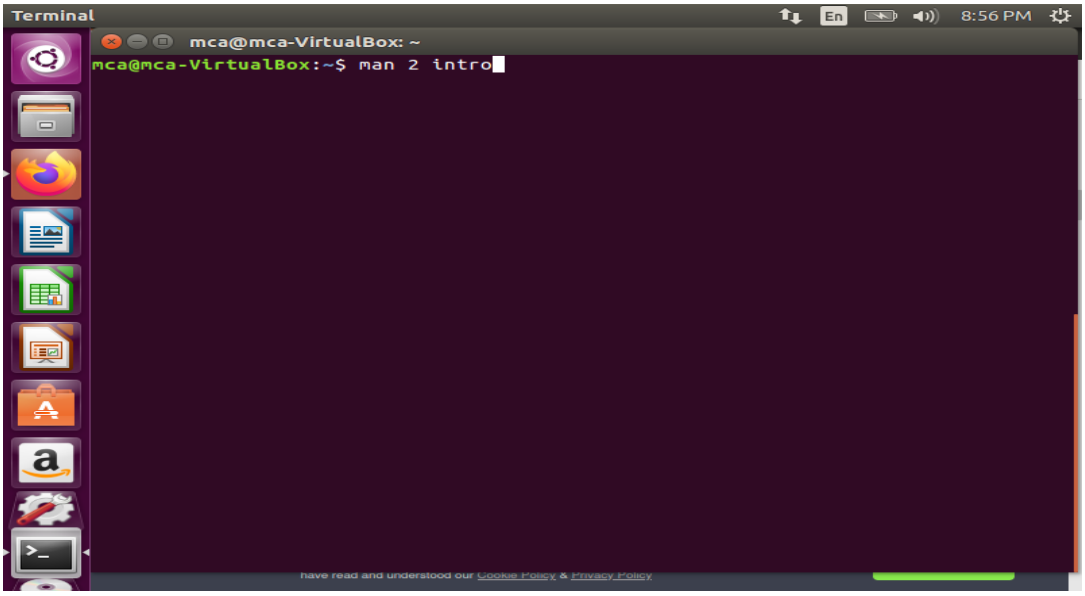
Syntax : *$ man [COMMAND NAME]*

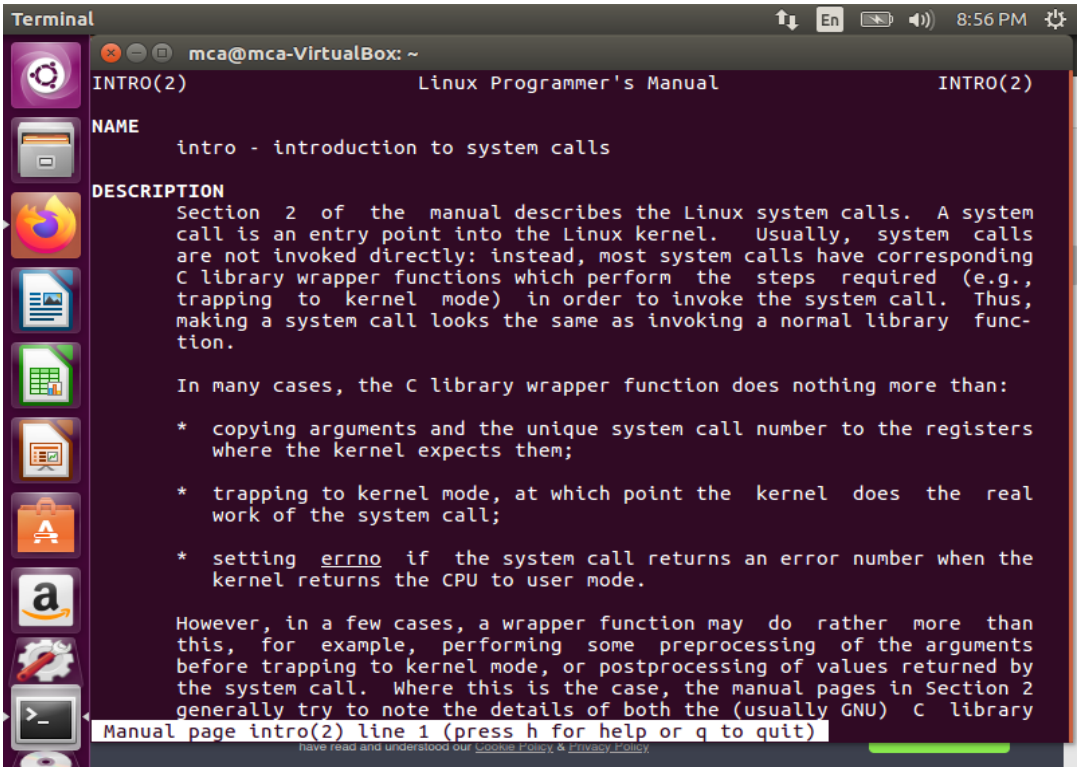




- Section-num: Since a manual is divided into multiple sections so this option is used to display only a specific section of a manual.

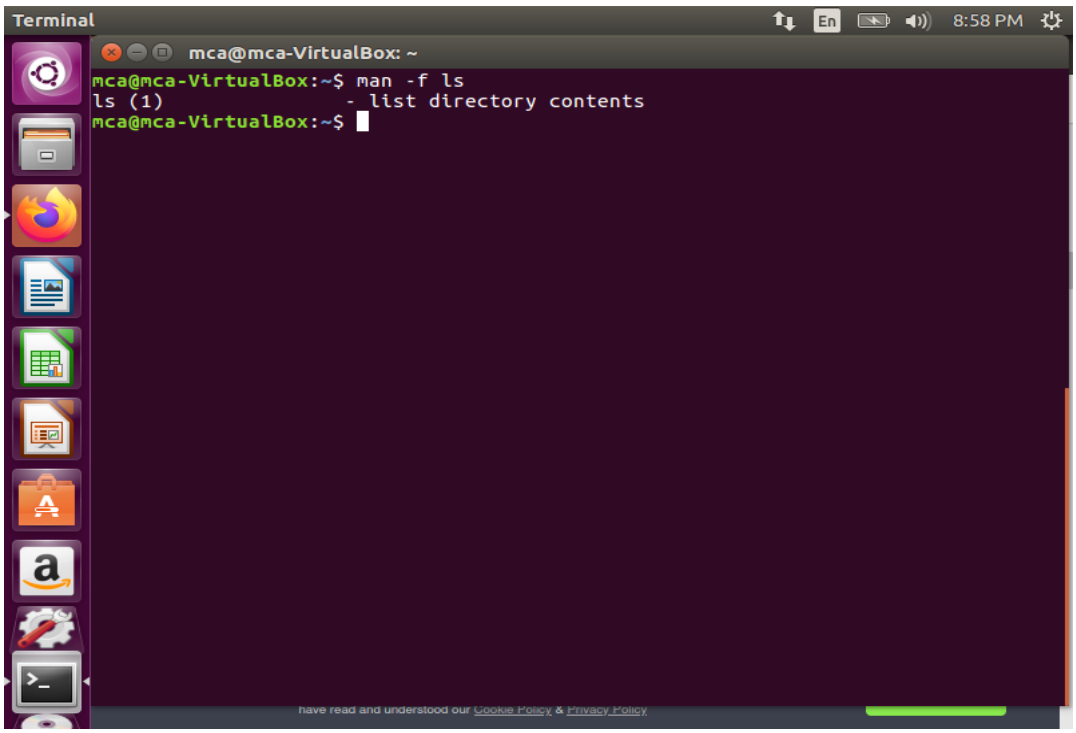
Syntax : *$ man [SECTION-NUM] [COMMAND NAME]*





- -f option: One may not be able to remember the sections in which a command is present. So this option gives the section in which the given command is present.

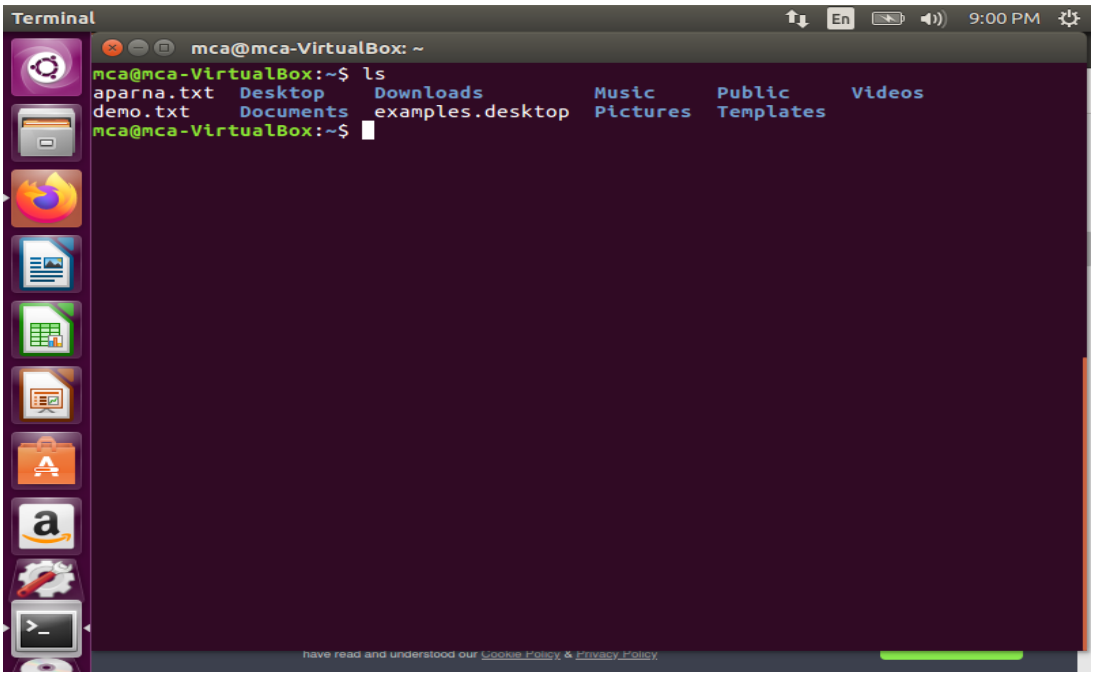
Syntax: *$ man -f [COMMAND NAME]*



2. ls

ls is a Linux shell command that lists directory contents of files and directories.

Syntax: ***$ ls***



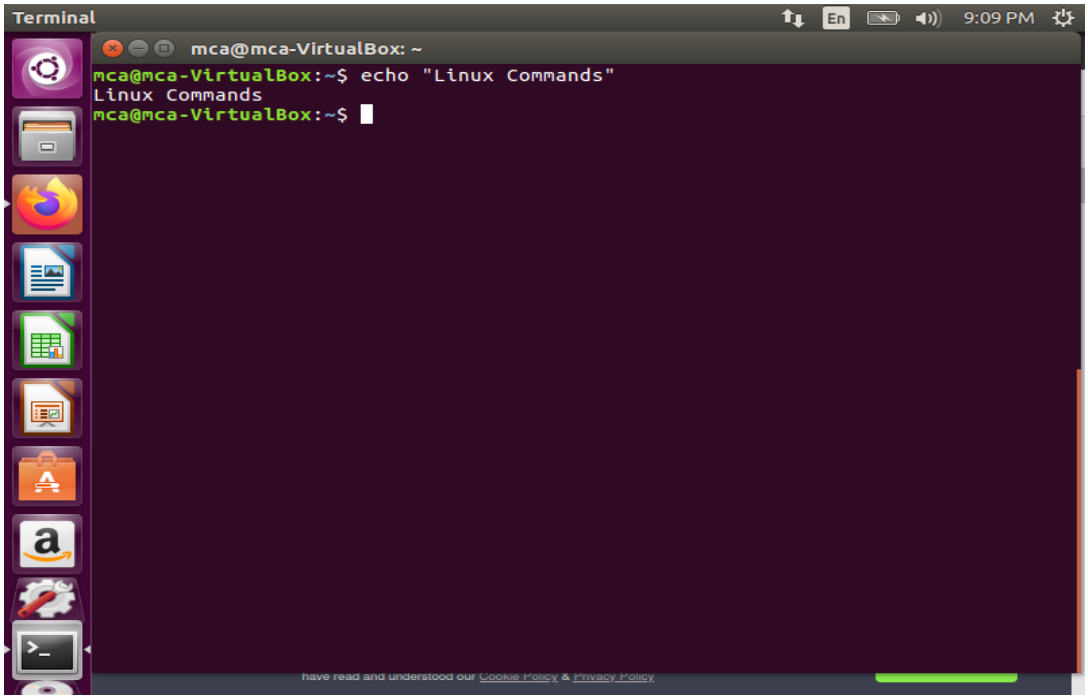
3. echo

echo command in linux is used to display line of text/string that are passed as an argument . This is a built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.

Syntax : *echo [option] [string]*

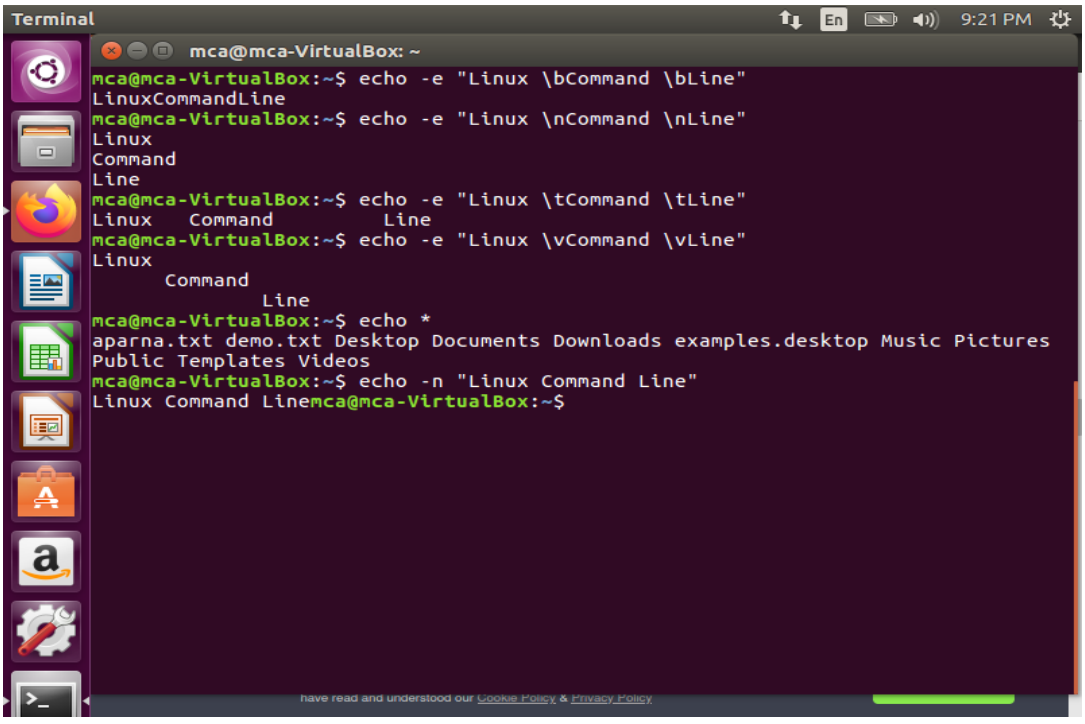
- For displaying a text/string :

Syntax : *echo [string]*



Options of echo command

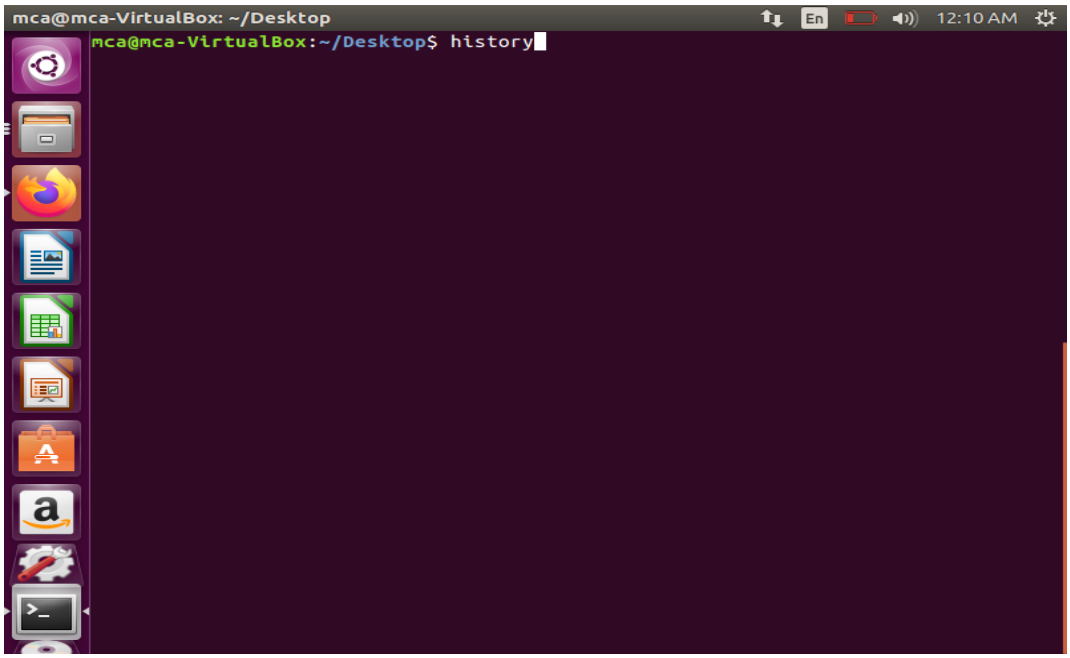
- \b : it removes all the spaces in between the text
- \n : this option creates new line from where it is used.
- \t : this option is used to create horizontal tab spaces.
- \v : this option is used to create vertical tab spaces.
- echo * : this command will print all files/folders, similar to ls command .
- -n : this option is used to omit echoing trailing newline .

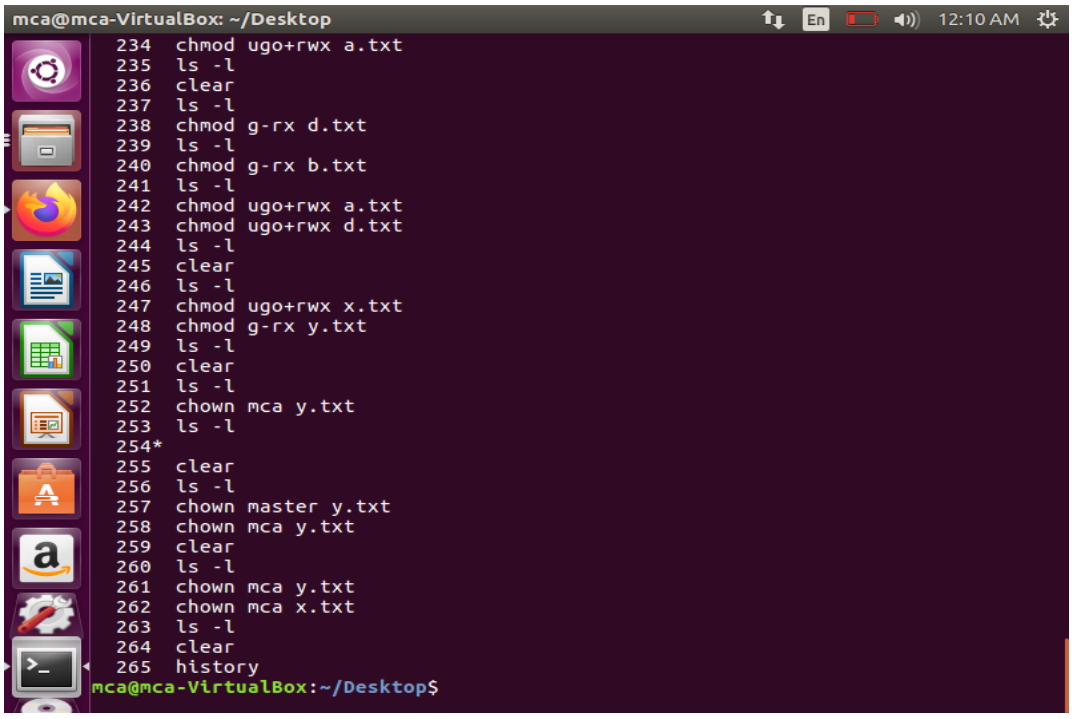


4. history

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*





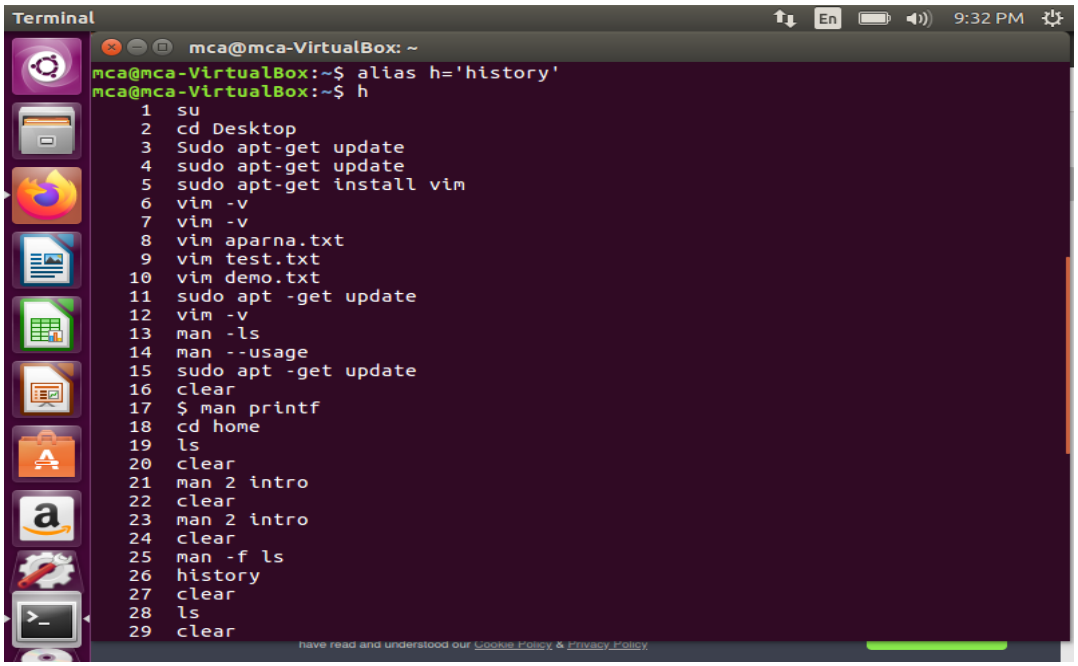
5. alias

alias command instructs the shell to replace one string with another string while executing the commands**.**

Syntax: *alias [-p] [name[=value] ...]*

For creating an alias :

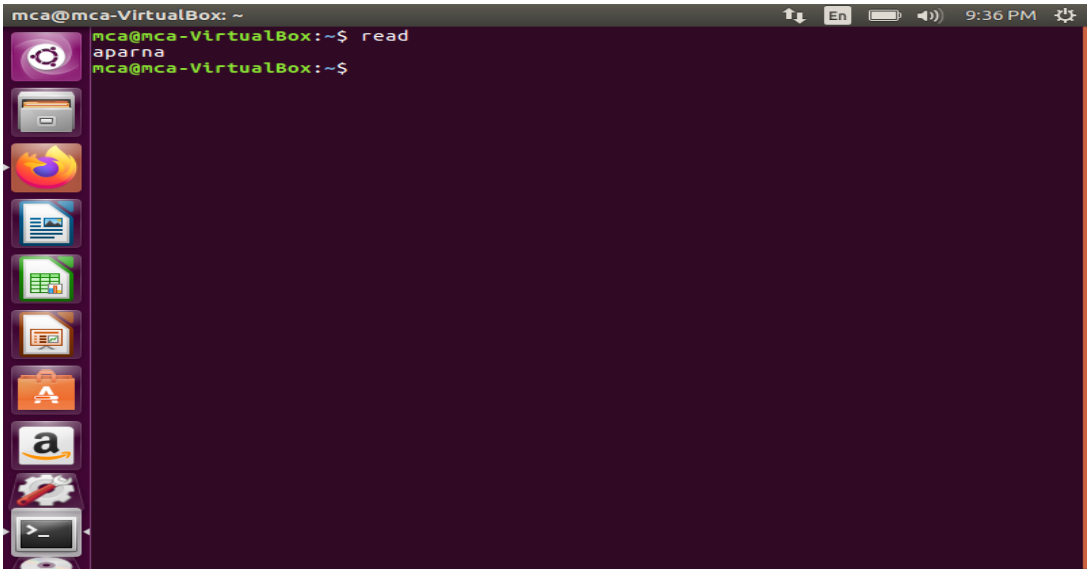
Syntax: *alias name="value"*

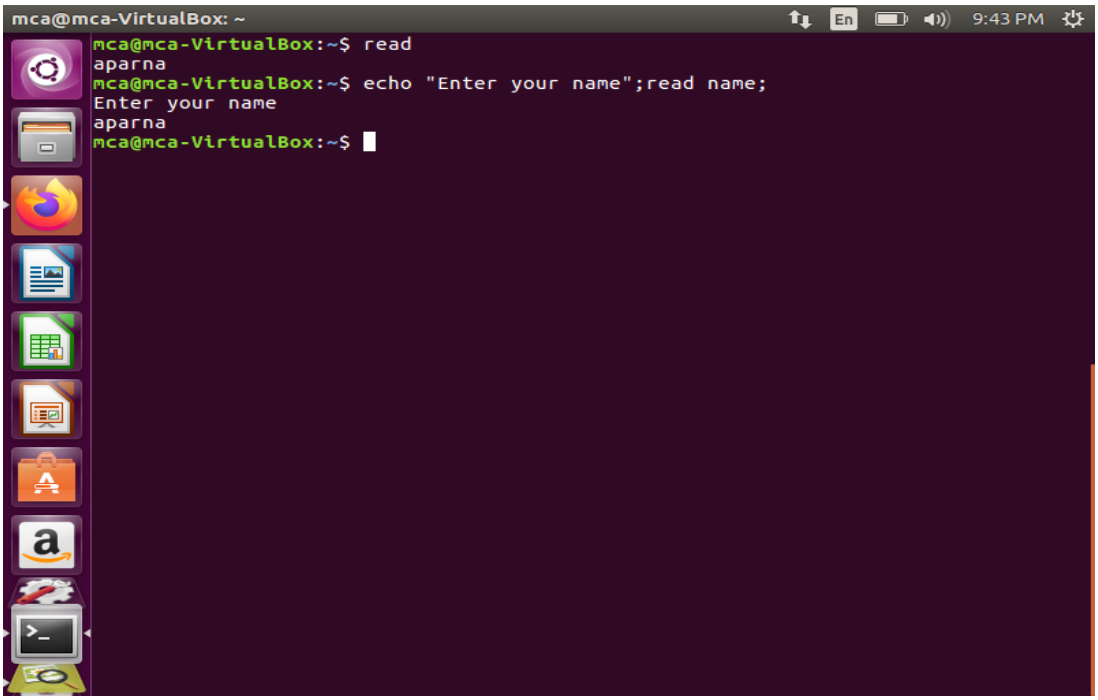


6. read

read command in Linux system is used to read from a file descriptor. Basically, this command read up the total number of bytes from the specified file descriptor into the buffer. If the number or count is zero then this command may detect the errors. But on success, it returns the number of bytes read. Zero indicates the end of the file. If some errors found then it returns -1.

Syntax: *read*





7. more

more command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large. The more command also allows the user do scroll up and down through the page. When the output is large, we can use more command to see output one by one.

Syntax: *more [-options] [-num] [+/pattern] [+linenum] [file_name]*

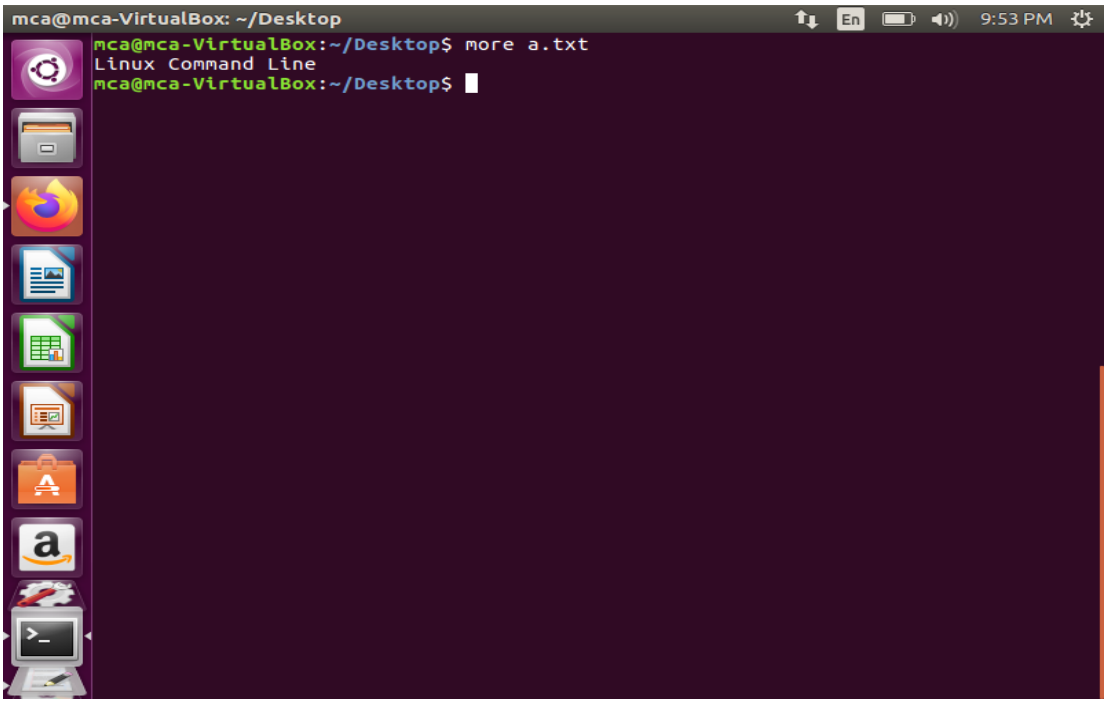
- *[-options]: any option that you want to use in order to change the way the file is displayed. Choose any one from the followings: (-d,-f, -p, -u)*
- *[-num]: type the number of lines that you want to display per screen.*
- *[+/pattern]: replace the pattern with any string that you want to find in the text file.*
- *[+linenum]: use the line number from where you want to start displaying the text content.*
- *[file_name]: name of the file containing the text that you want to display on the screen.*

Options:

-d : Use this command in order to help the user to navigate.

-f : This option does not wrap the long lines and displays them as such.

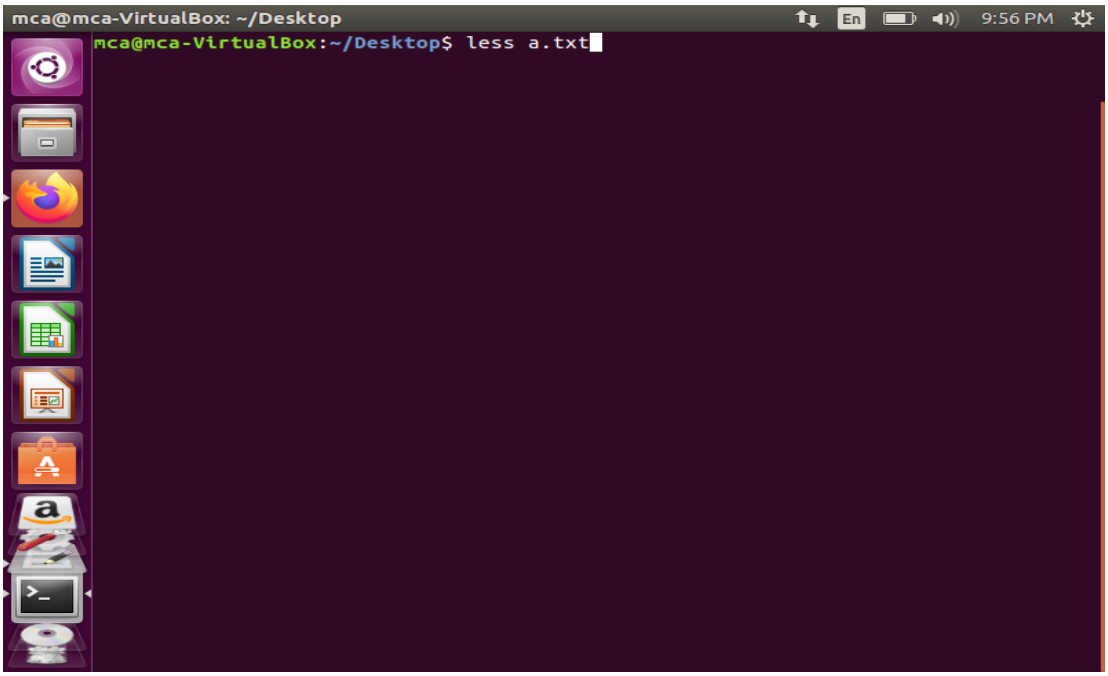
-p : This option clears the screen and then displays the text

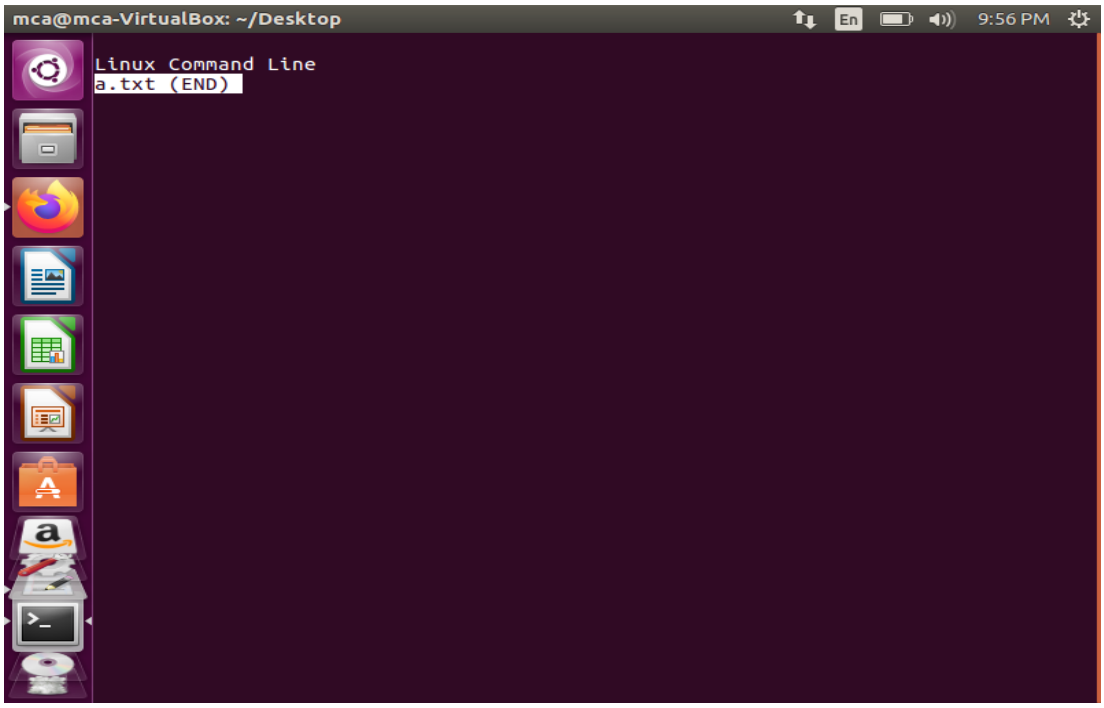


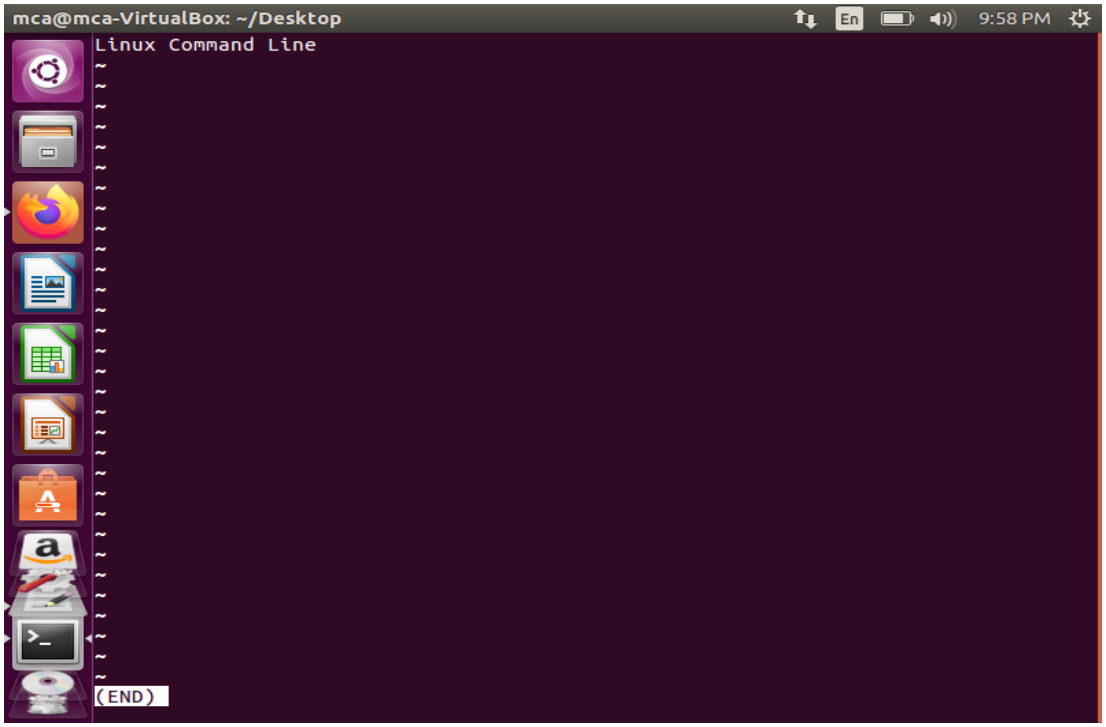
8. less

Less command is a Linux utility that can be used to read the contents of a text file one page(one screen) at a time. It has faster access because if file is large it doesn't access the complete file, but accesses it page by page.

Syntax : *less filename*







9. cat

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.

- To view a single file

Command: *$ cat filename*

- To view multiple files

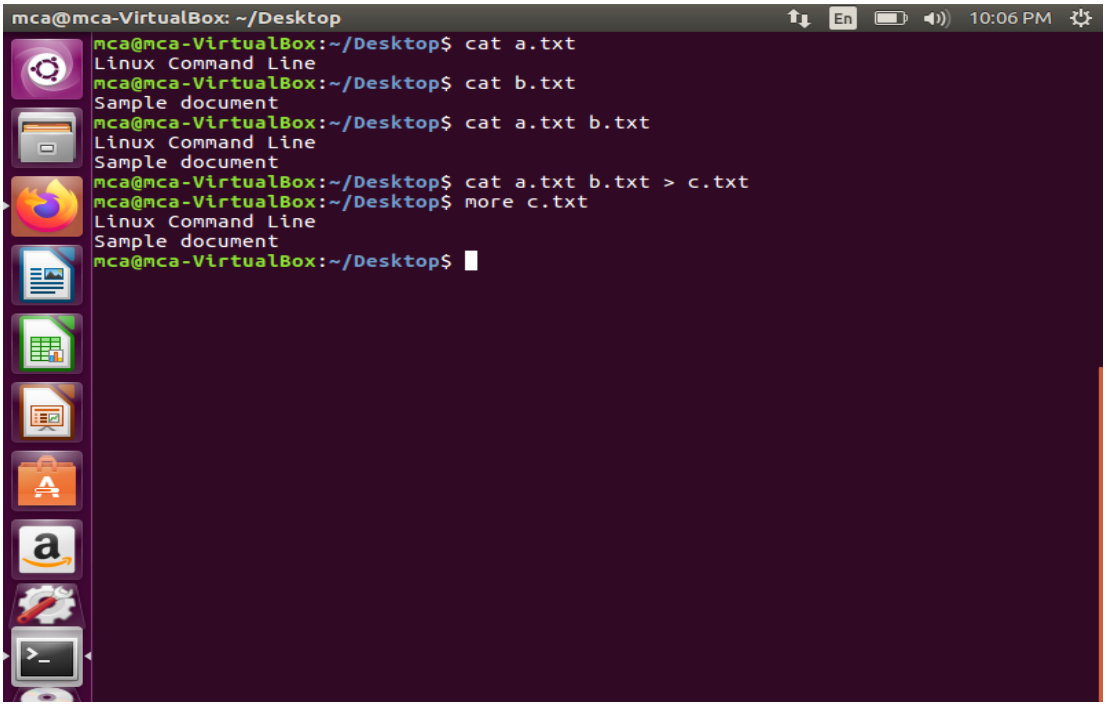
Command: *$cat file1 file2*

- Create a file

Command: *$ cat >newfile*

- Copy the contents of one file to another file.

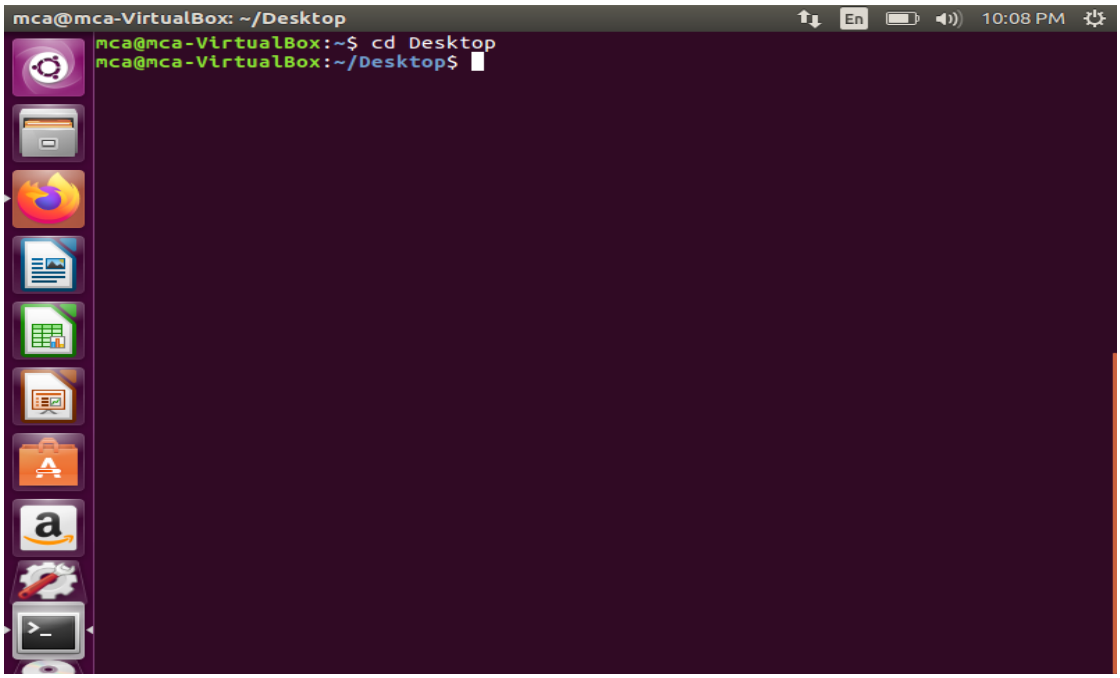
Command: *$cat [filename-whose-contents-is-to-be-copied] > [destination-filename]*



10. cd

cd command in linux known as change directory command. It is used to change current working directory.

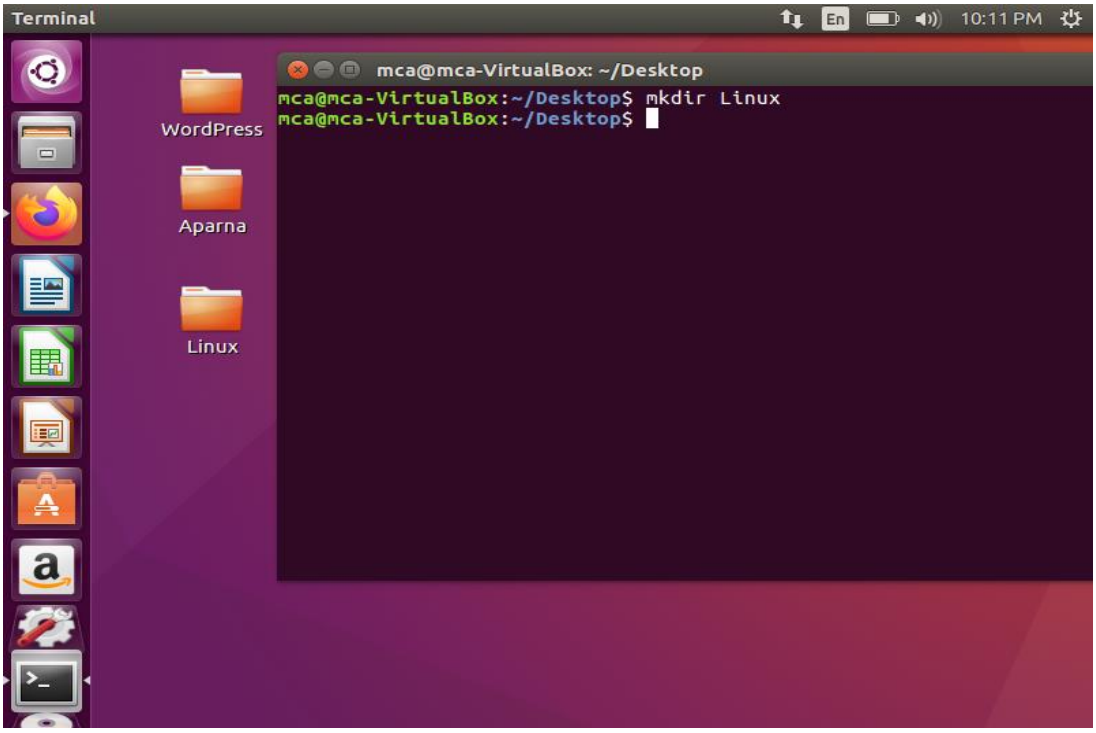
Syntax: *$ cd [directory]*

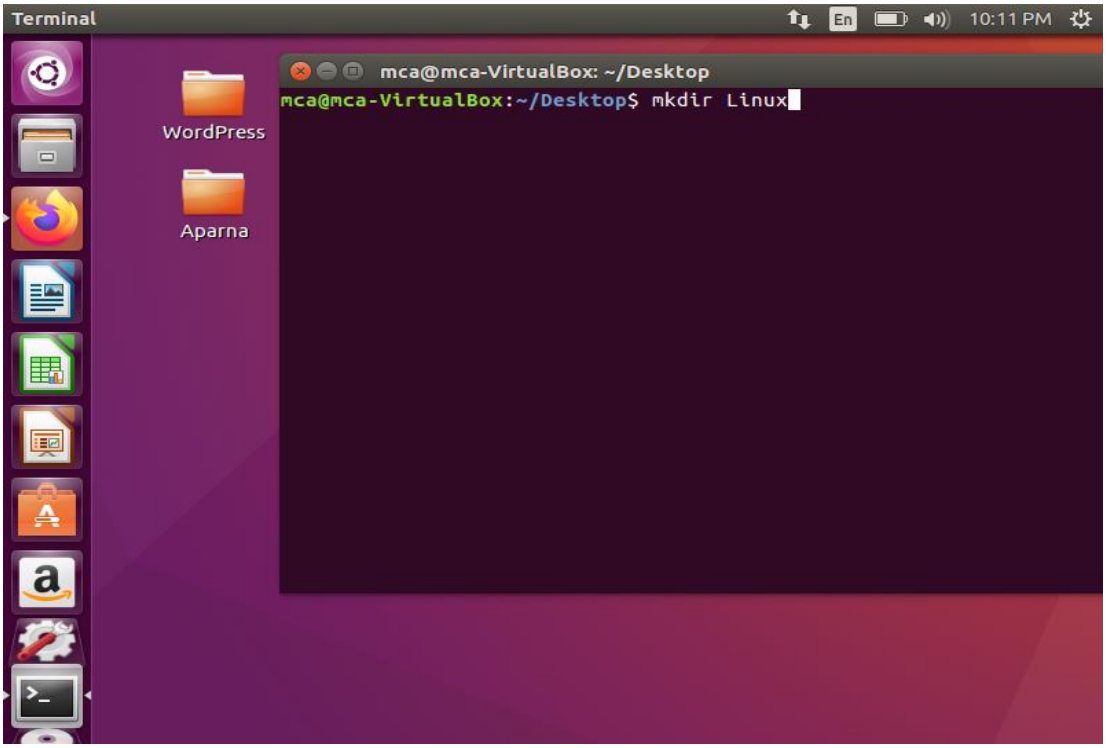


11. mkdir

mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems).

Syntax: *mkdir [options...] [directories ...]*

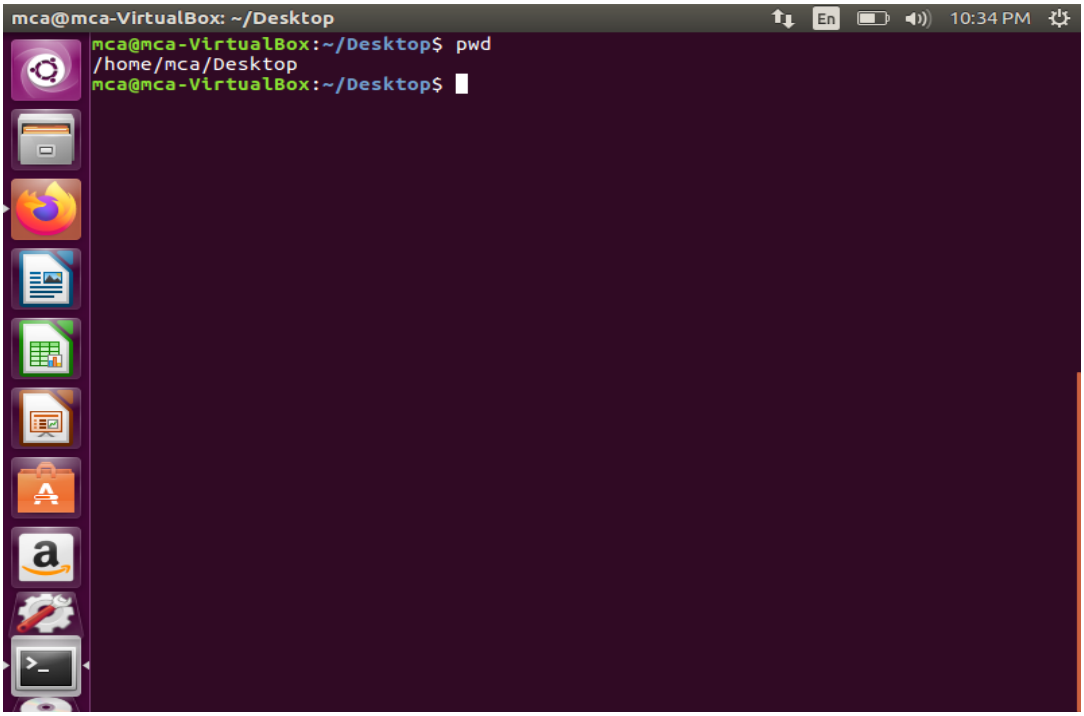




12. pwd

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 which stores the path of the current directory.

Syntax: *pwd*

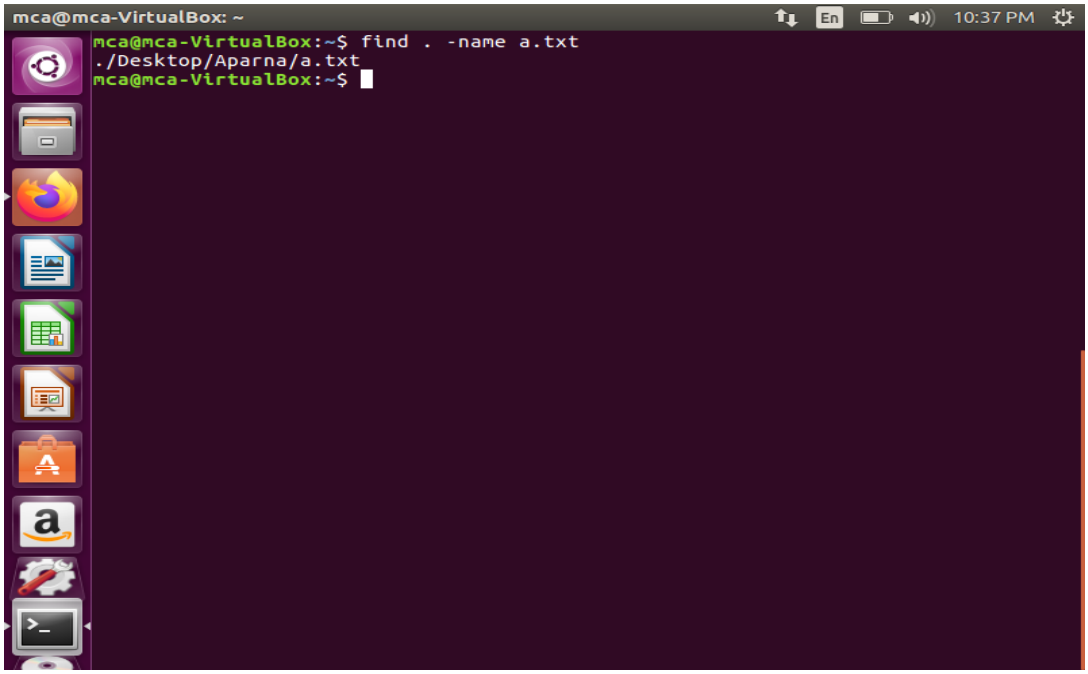


13. find

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.

Syntax : *$ find [where to start searching from]*

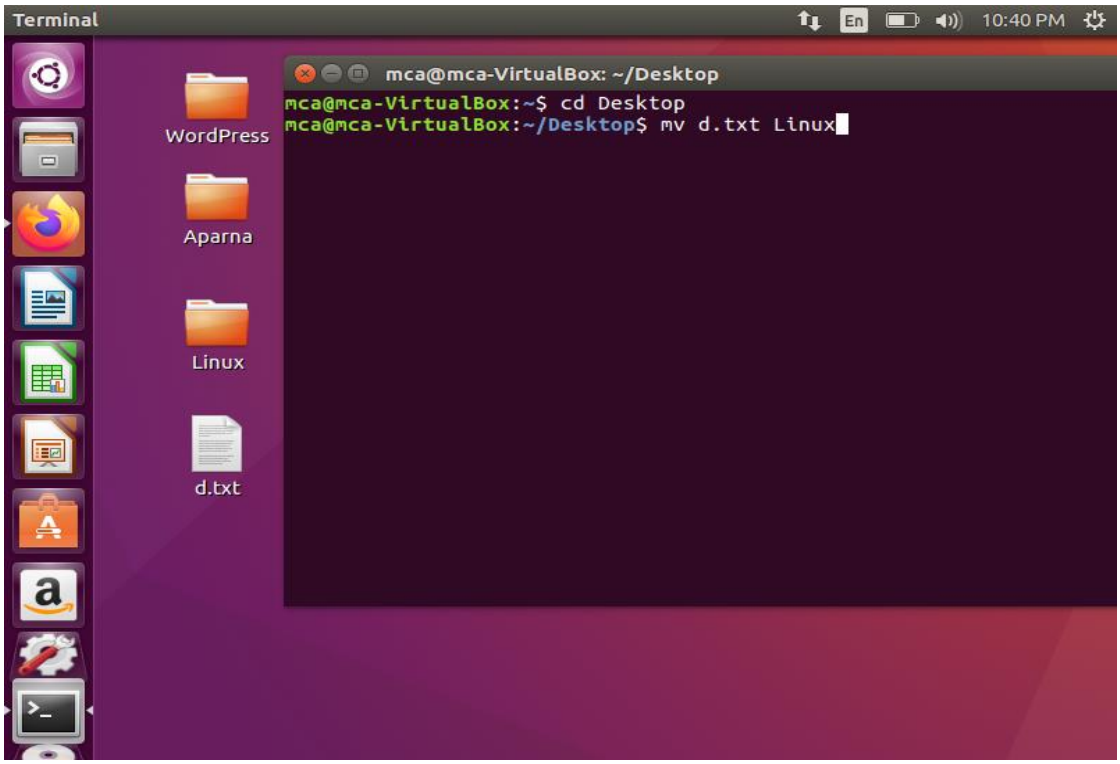
[expression determines what to find] [-options] [what to find]

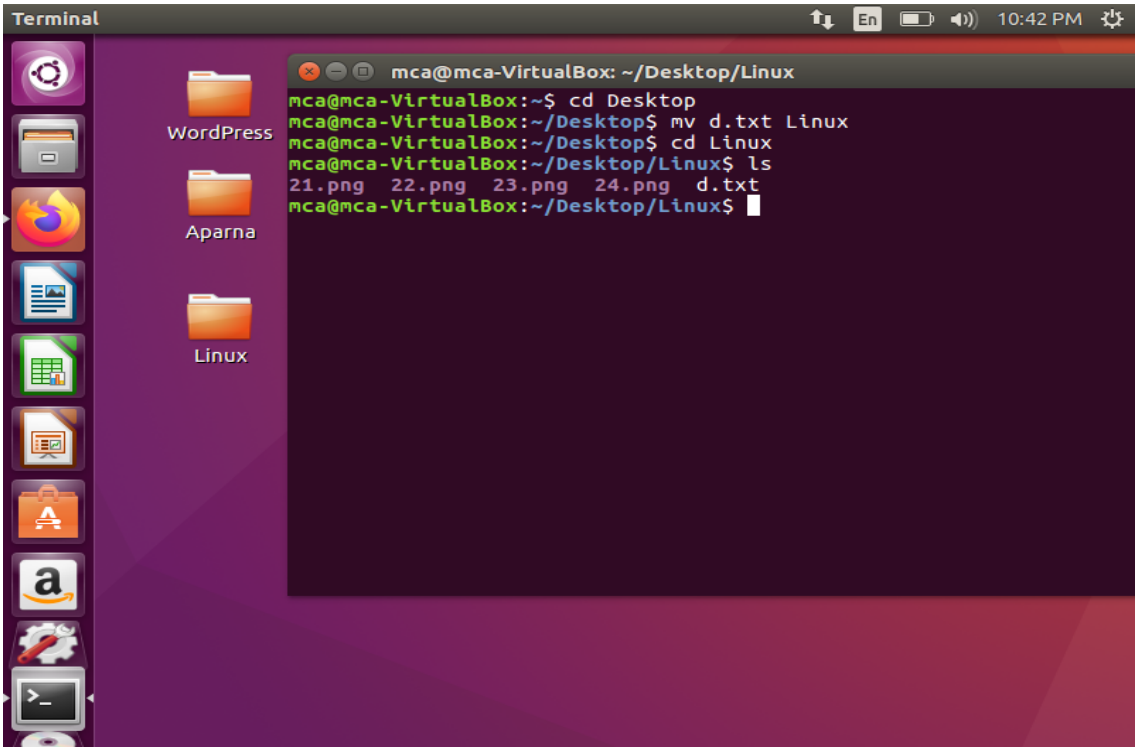


14. mv

mv stands for move. mv is used to move one or more files or directories from one place to another in a file system like UNIX.

Syntax: *mv [Option] source destination*

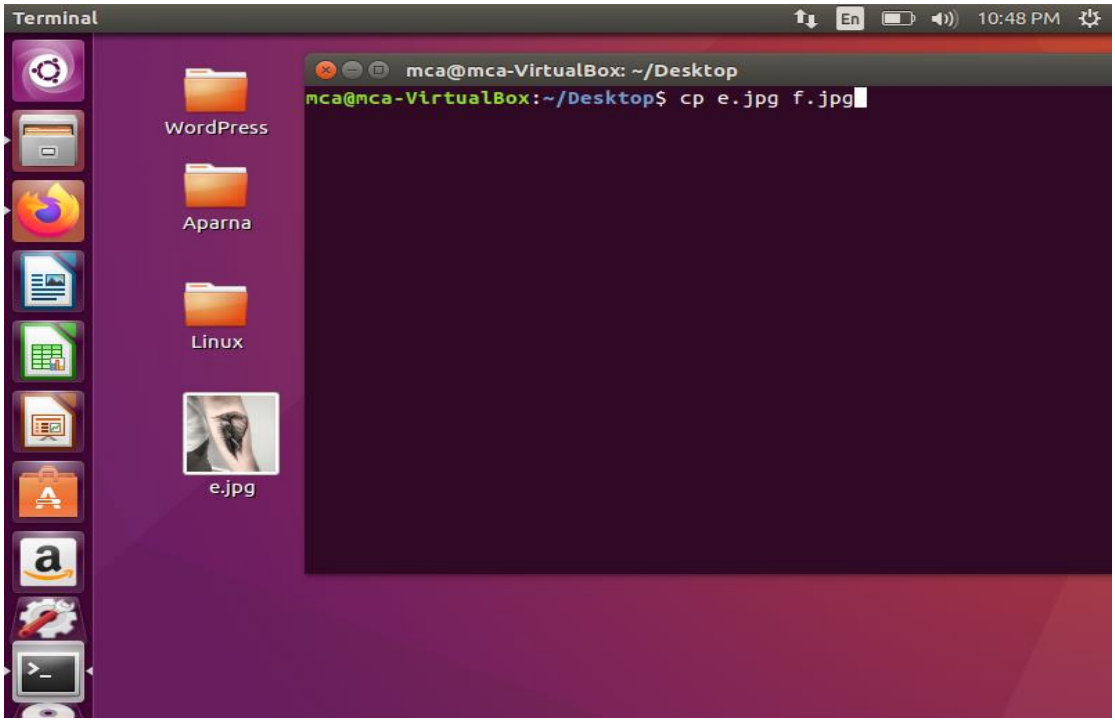


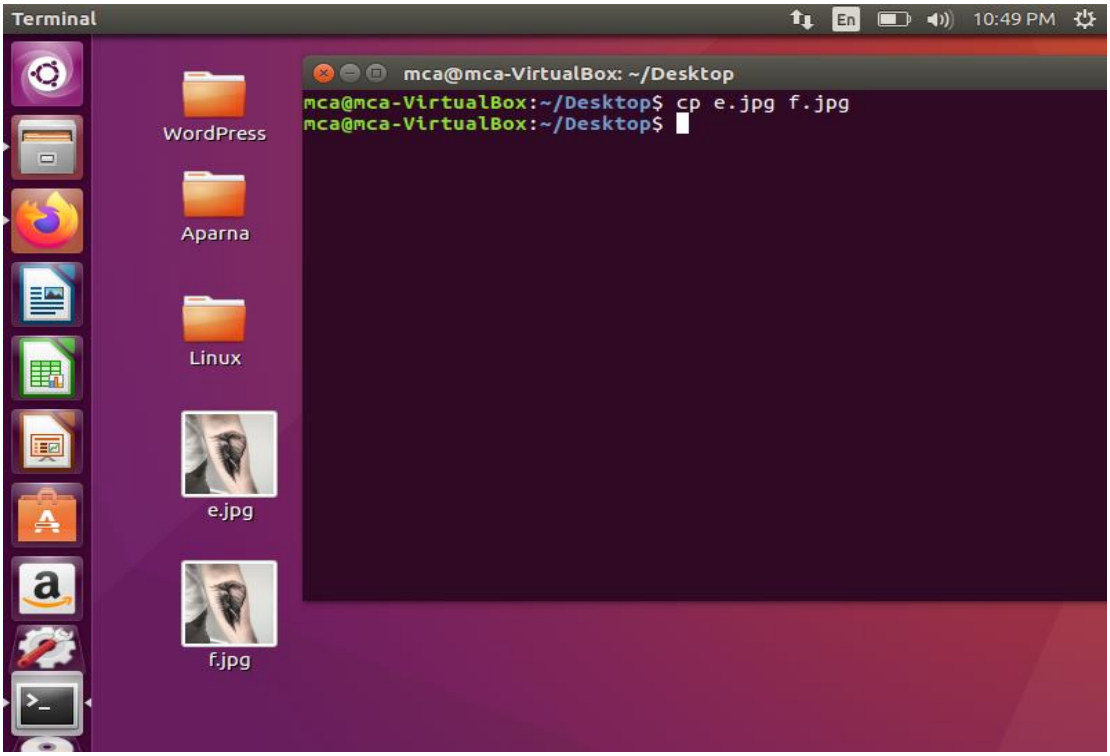


15. cp

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*

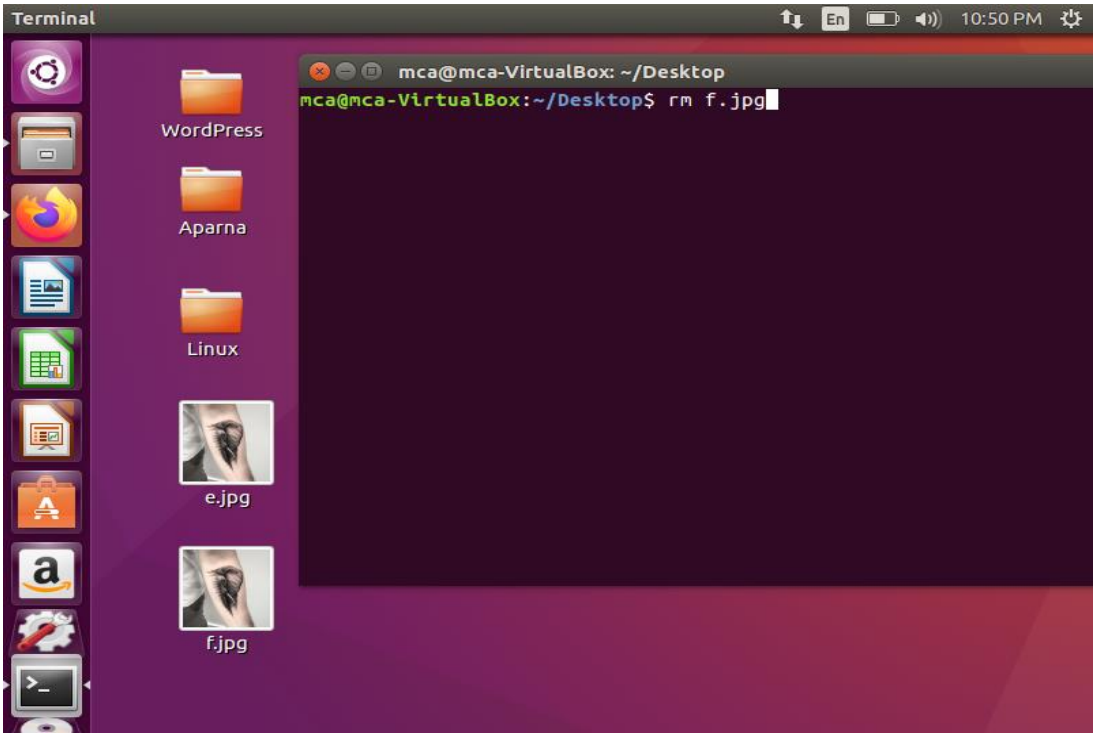


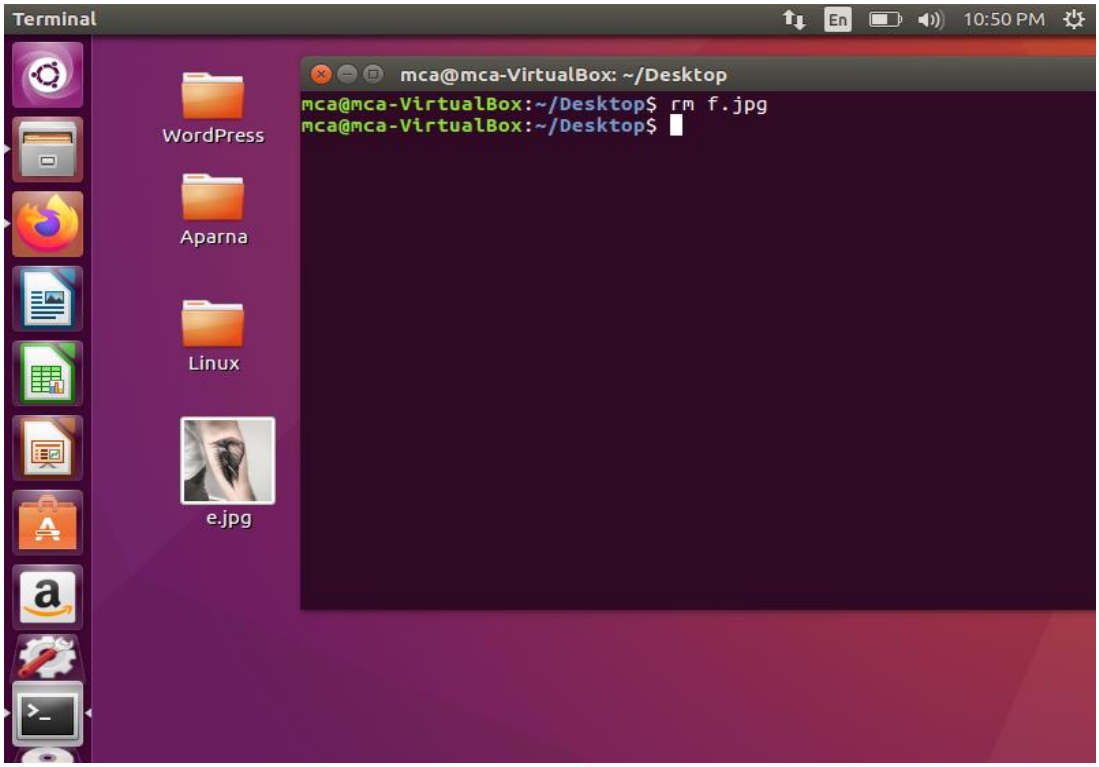


16. rm

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.

Syntax: *rm [OPTION]... FILE...*

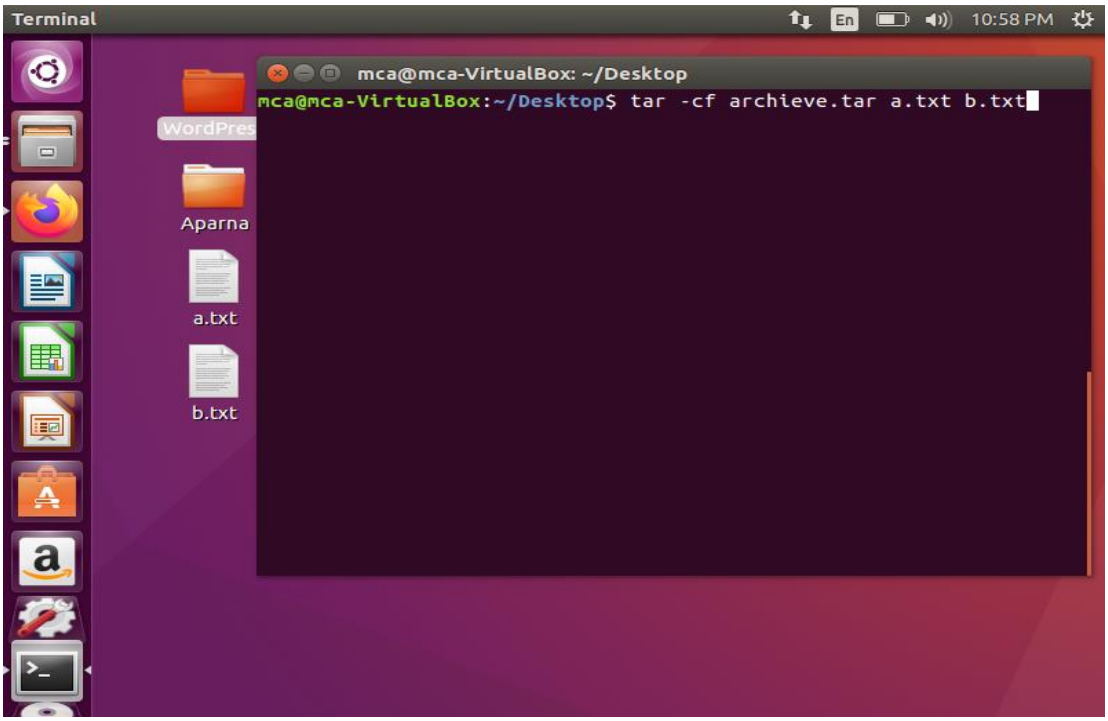


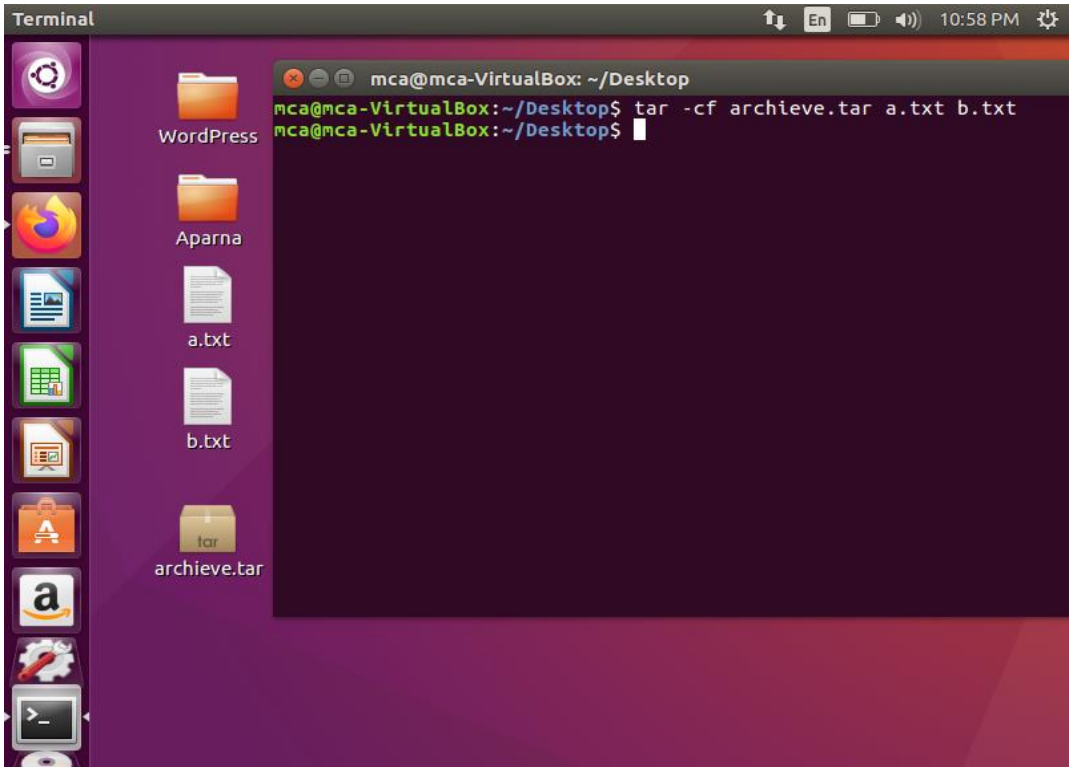


17. tar

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 archived]*

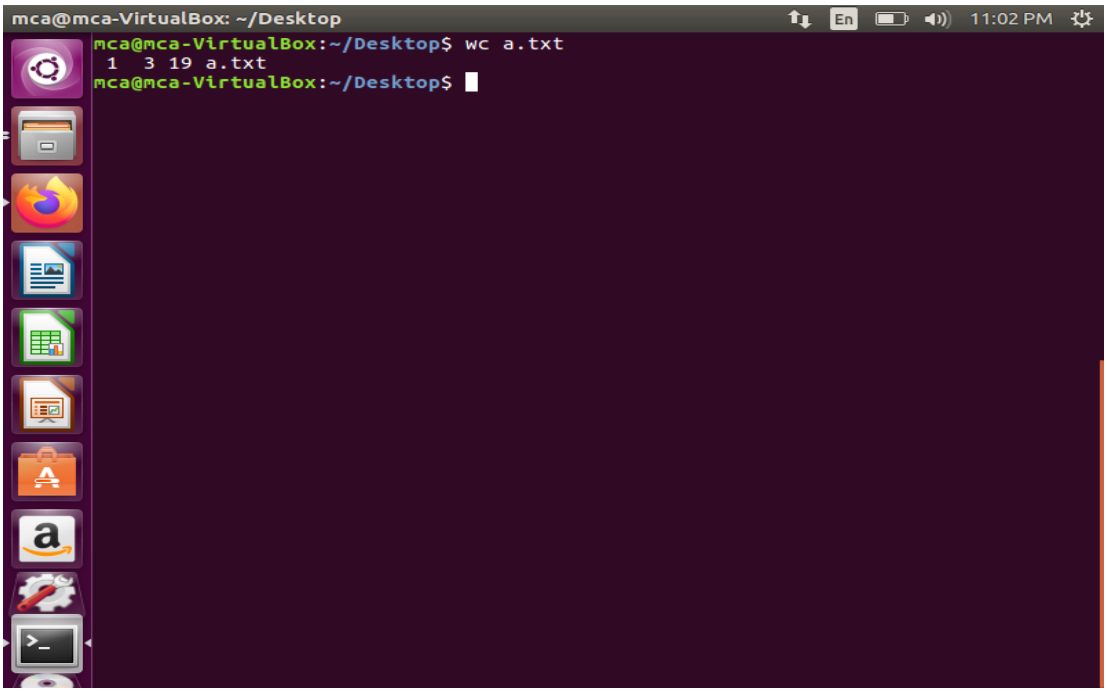




18. wc

wc stands for word count. As the name implies, it is mainly used for counting purpose.It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.

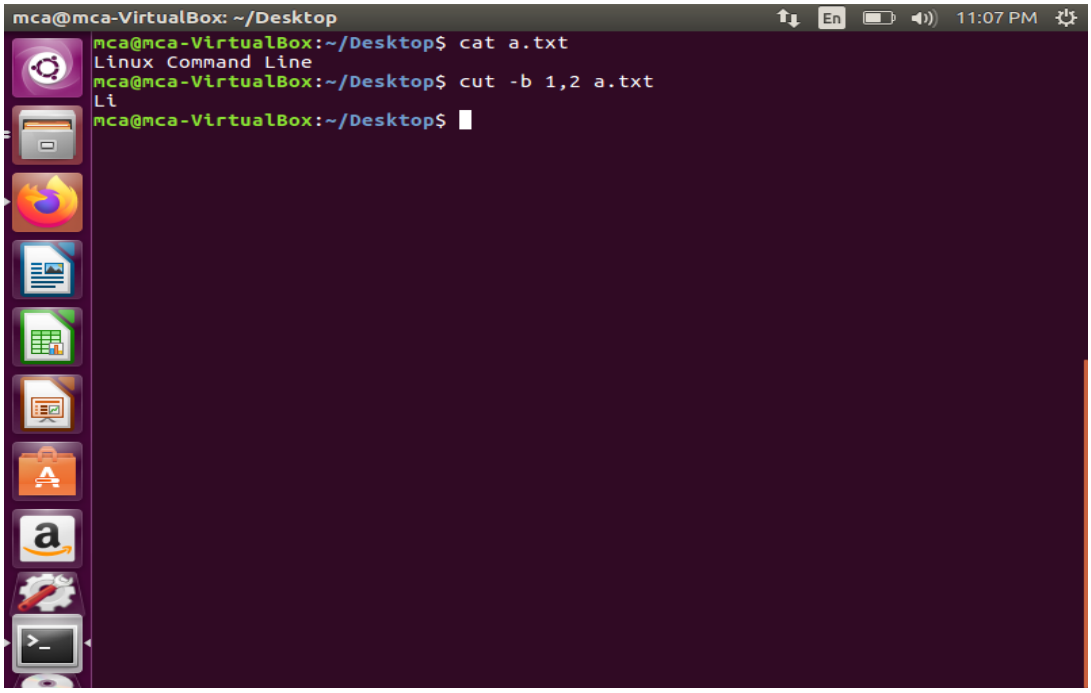
Syntax: *wc [OPTION]... [FILE]...*



19. cut

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.

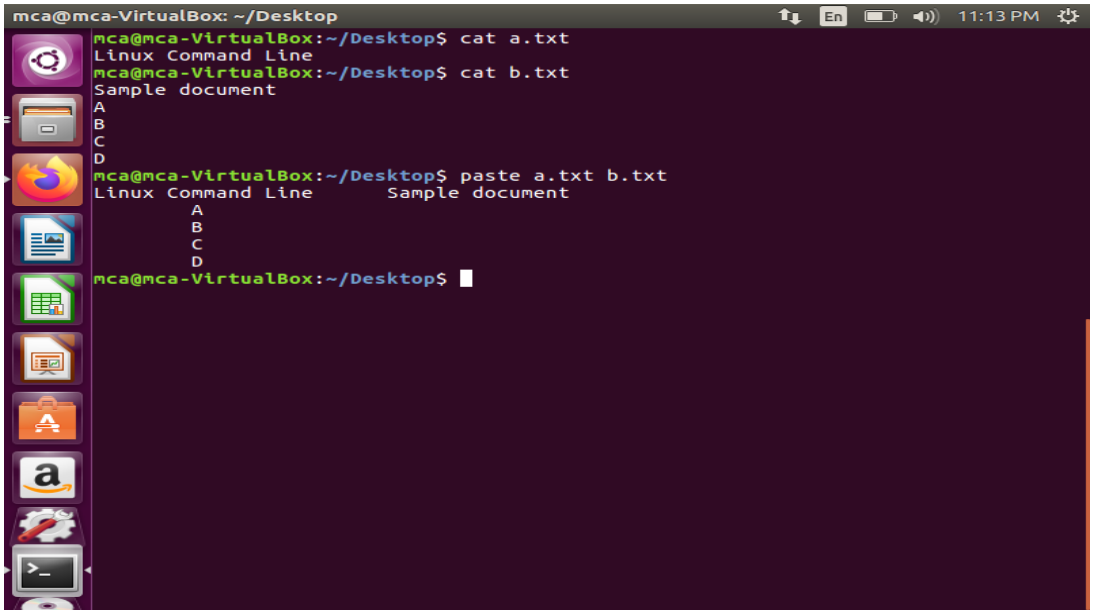
Syntax: *cut OPTION... [FILE]...*



20. paste

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.

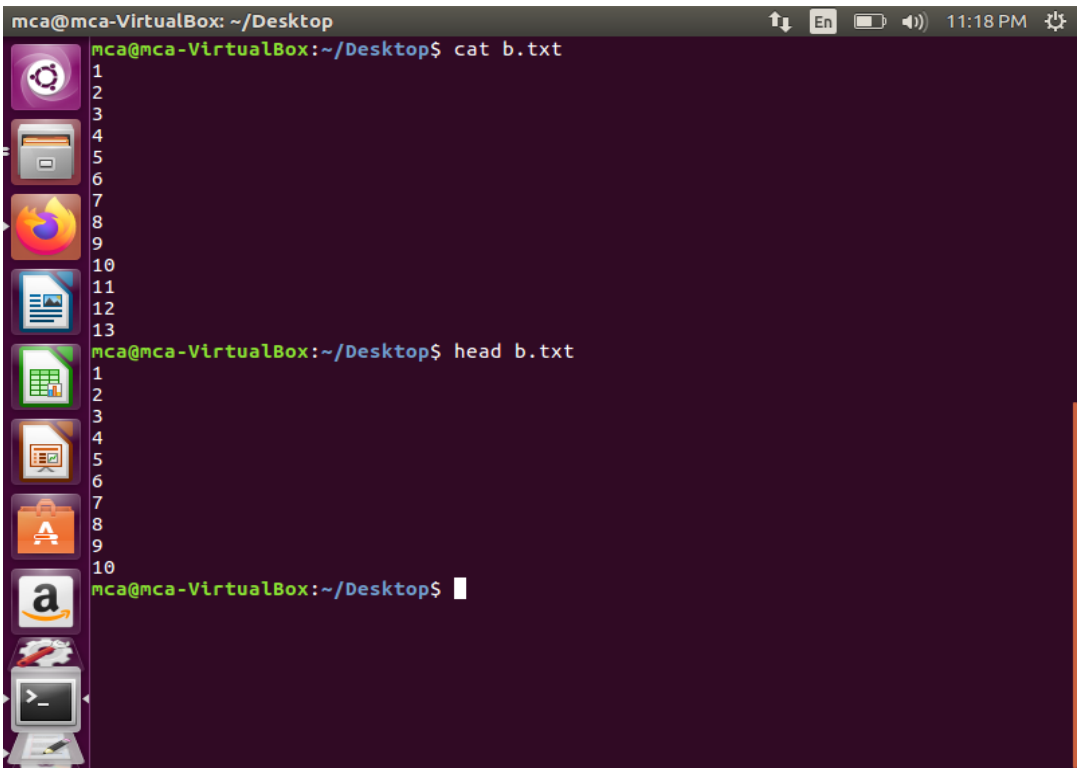
Syntax: *paste [OPTION]... [FILES]...*



21. head

The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

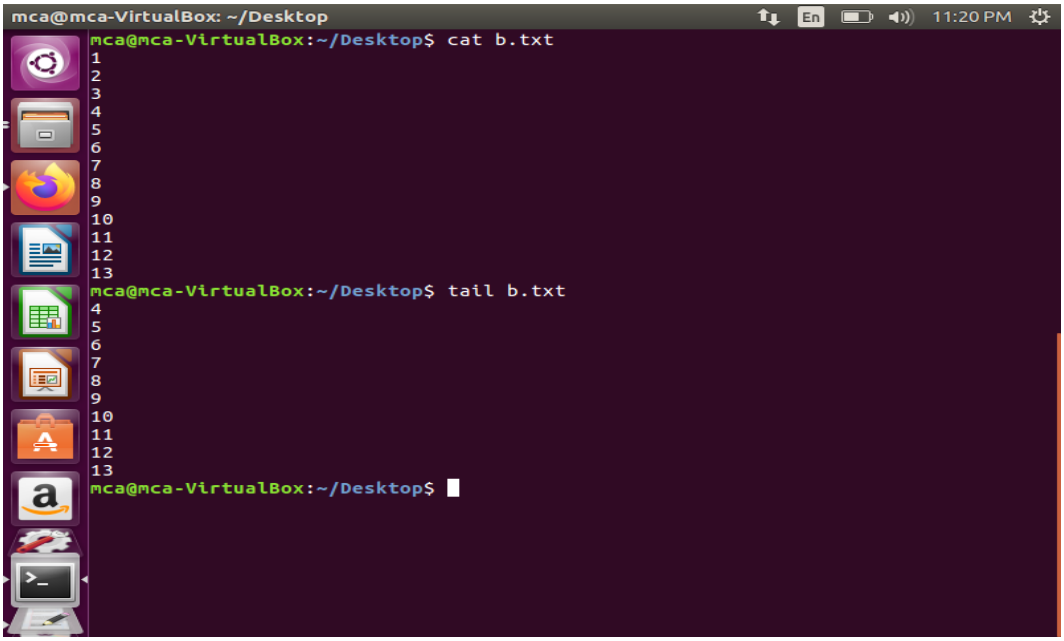
Syntax: *head [OPTION]... [FILE]...*



22. tail

The tail command, as the name implies, print the last N number of data of the given input. By default it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is precedes by its file name.

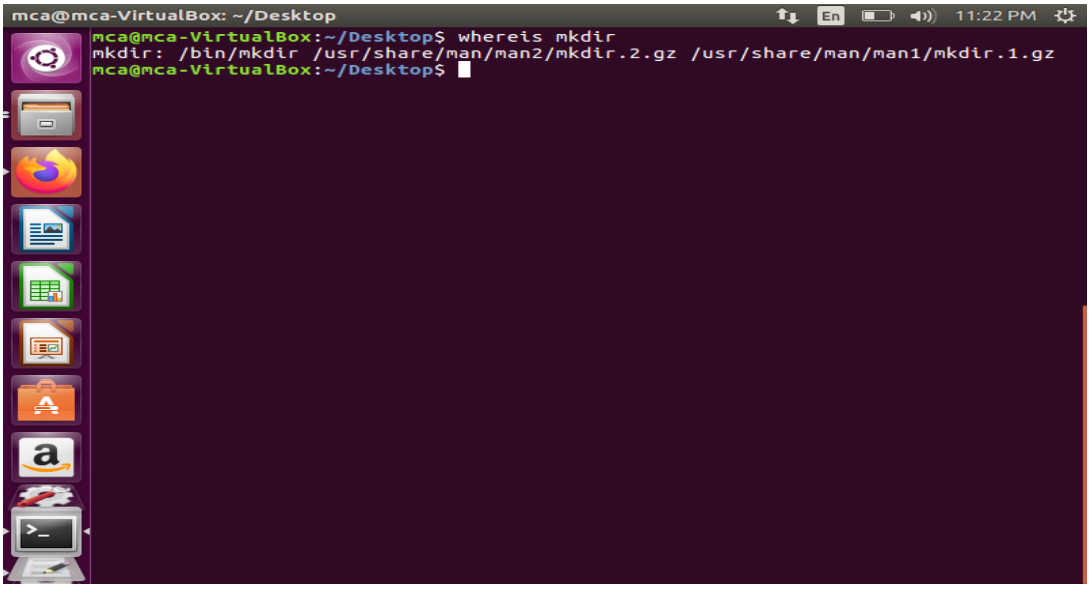
Syntax: *tail [OPTION]... [FILE]...*



23. whereis

whereis command is used to find the location of source/binary file of a command and manuals sections for a specified file in Linux system.

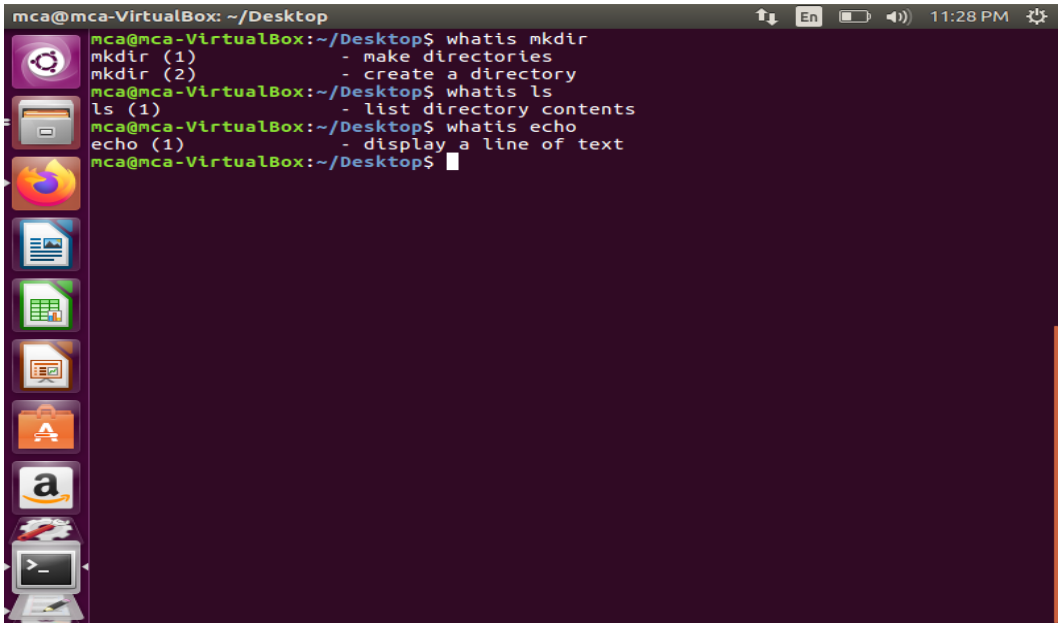
Syntax: *whereis [options] filename...*



24. whatis

whatis command in Linux is used to get a one-line manual page descriptions. In Linux, each manual page has some sort of description within it. So this command search for the manual pages names and show the manual page description of the specified filename or argument.

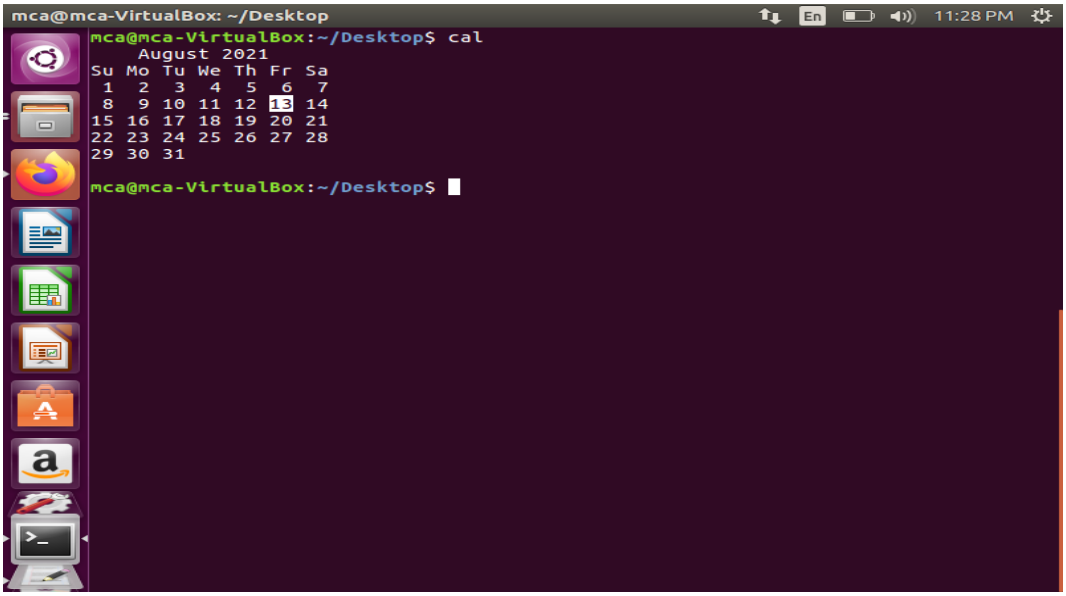
Syntax: *whatis [-dlv?V] [-r|-w] [-s list] [-m system[, ...]] [-M path] [-L locale] [-C file] name ...*



25. cal

cal command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.

Syntax: *cal [[month] year]*

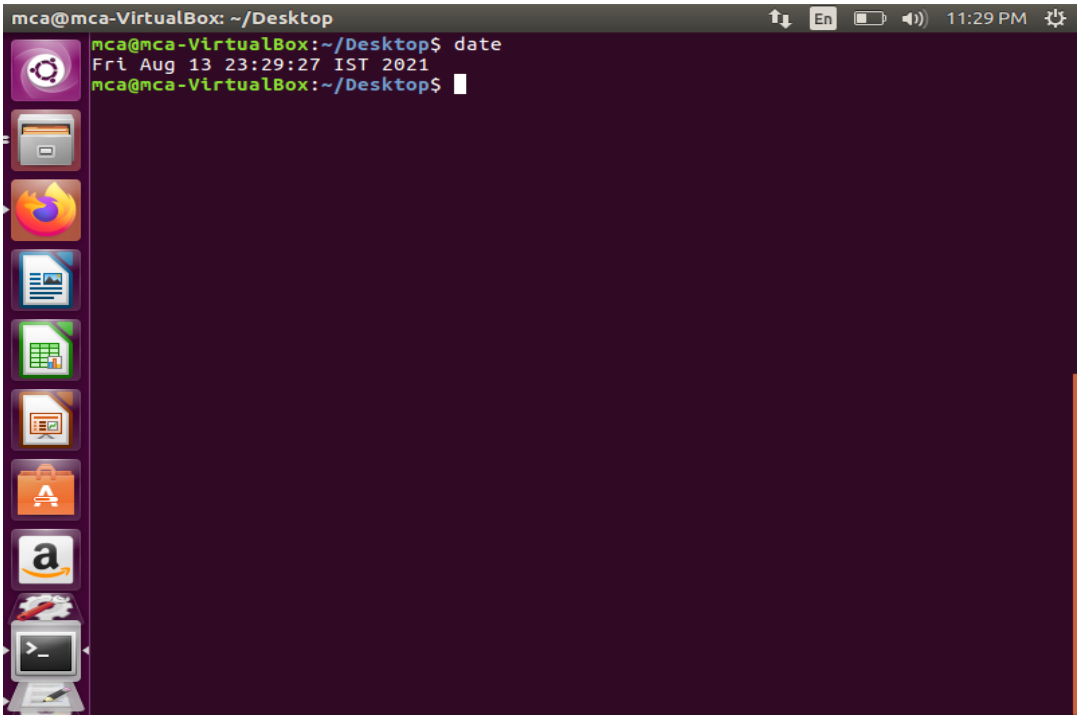


26. date

date command is used to display the system date and time. date command is also used to set date and time of the system.

Syntax: *date [OPTION]... [+FORMAT]*

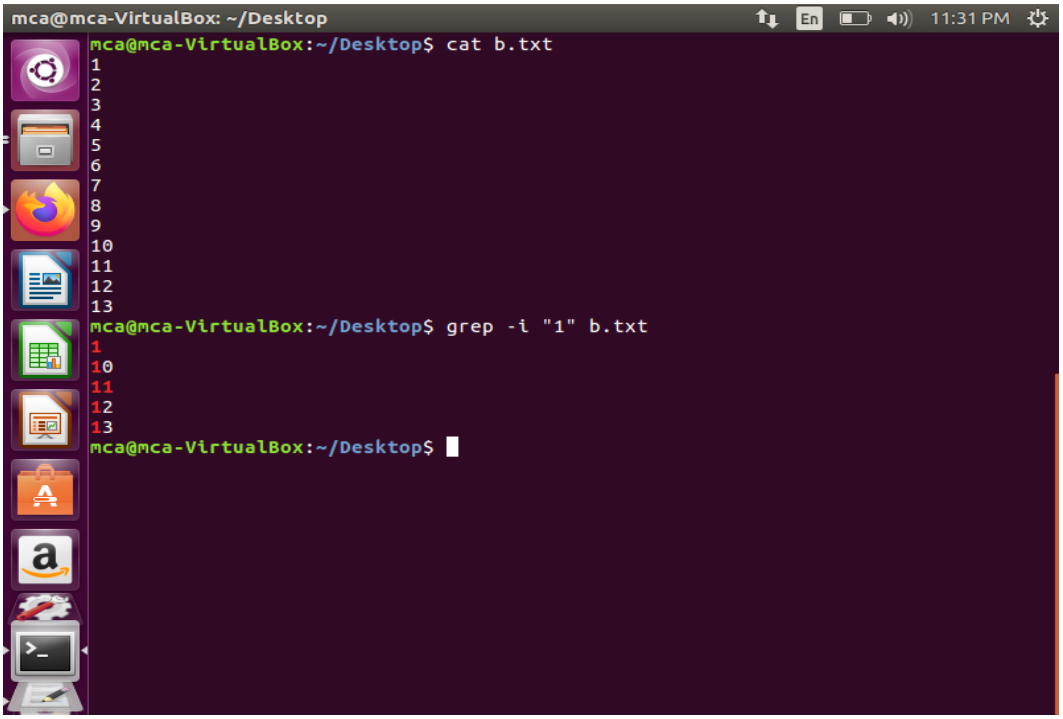
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]



27. grep

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

Syntax: *grep [options] pattern [files]*

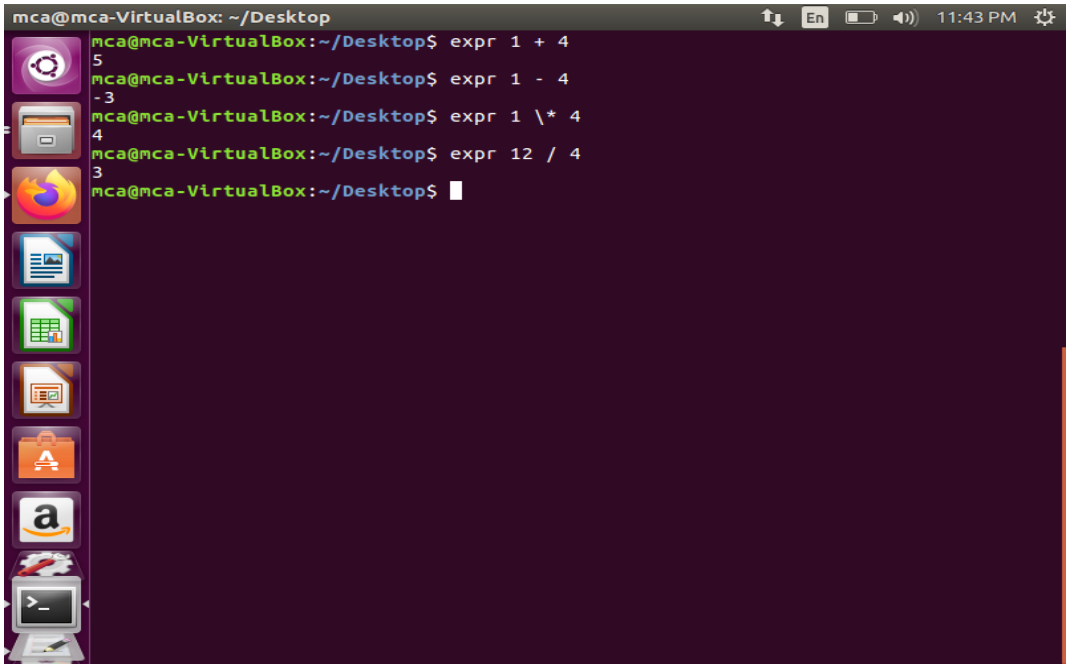


28. expr

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*



29. chmod

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.

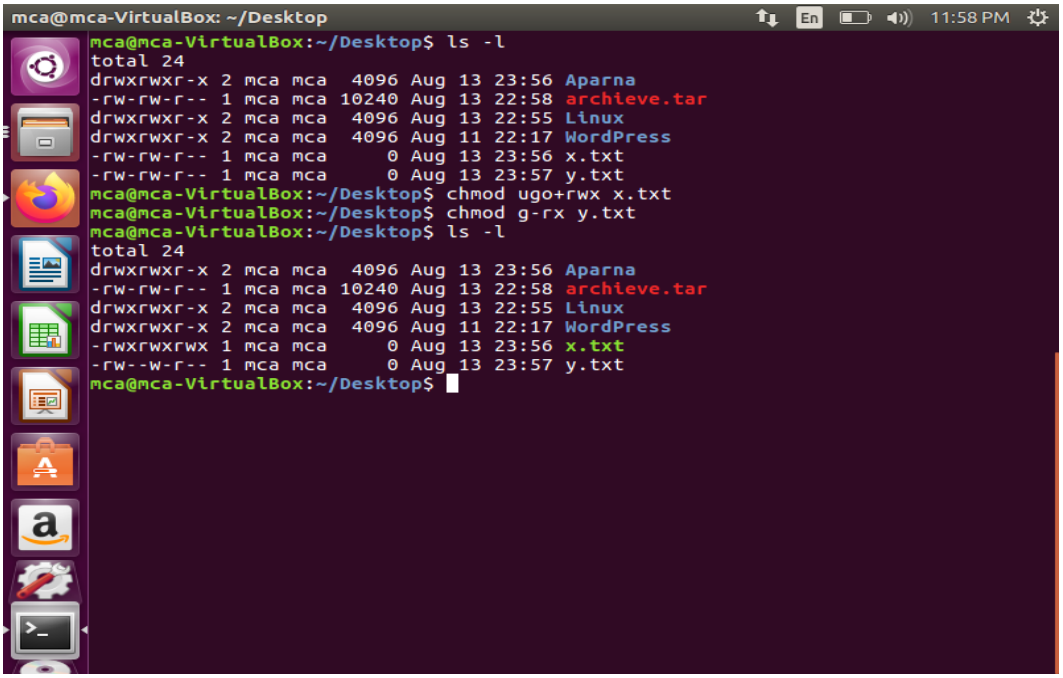
Syntax : *chmod [reference][operator][mode] file...*

The references are used to distinguish the users to whom the permissions apply i.e. they are list of letters that specifies whom to give permissions.

Reference Class	Description
u	owner file's owner
g	group users who are members of the file's group
o	others users who are neither the file's owner nor members of the file's group
a	All three of the above, same as ugo

The operator is used to specify how the modes of a file should be adjusted. The following operators are accepted:

Operator	Description
+	Adds the specified modes to the specified classes
-	Removes the specified modes from the specified classes
=	The modes specified are to be made the exact modes for the specified classes



30. chown

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…