EXPIRIMENT-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

VIM (VI IMPROVED)

Vim is a text editor originally written by Bram Moolenar. The editor is a clone of Vi, a Unix text editor written by Sun Microsystems cofounder Bill Joy while he was a graduate student at UC Berkeley in the late 1970s. Vi originally appeared as part of the Berkeley Software Distribution of Unix, or BSD.

Unlike a word processor, Vim edits files in plain text. It is mostly used for writing programs.

Like its predecessor, Vi, Vim is characterized by its modal user interface. Users move around and select text in the "command mode," while editing is done in "insert mode." Vim proponents say that this method is very efficient because the commands are mostly on the home row of the keyboard.

Vim allows for a high degree of customization. Users can define macros to personalize their key mappings as well as automate editing tasks. It also supports syntax highlighting for most programming languages, including C, Python and HTML.

Vim is a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems.

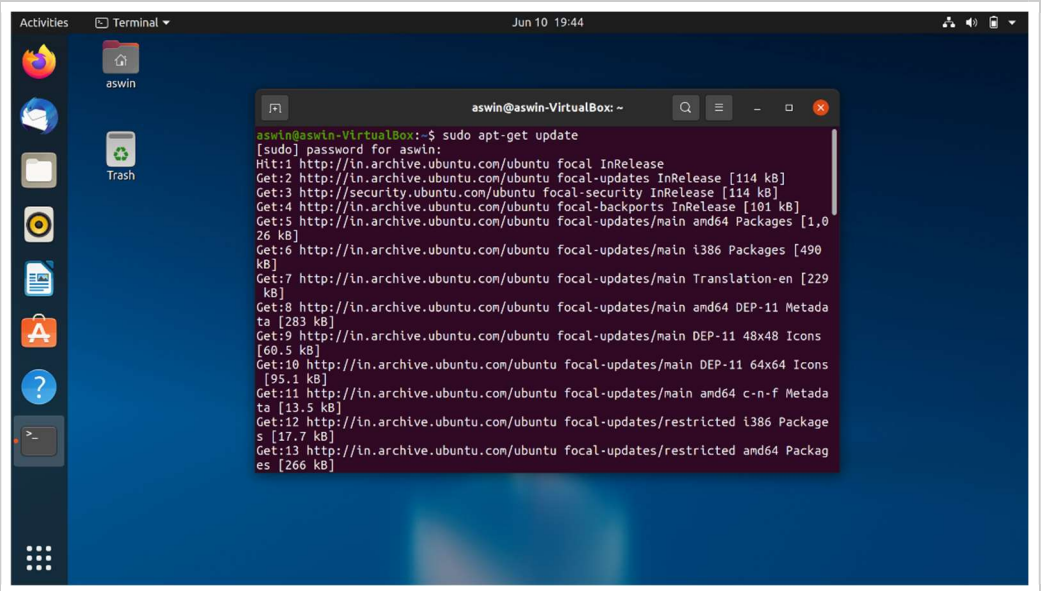
Vim is often called a "programmer's editor," and so useful for programming that many consider it an entire IDE. It's not just for programmers, though. Vim is perfect for all kinds of text editing, from composing email to editing configuration files.

Despite what the above comic suggests, Vim can be configured to work in a very simple (Notepad-like) way, called evim or Easy Vim.

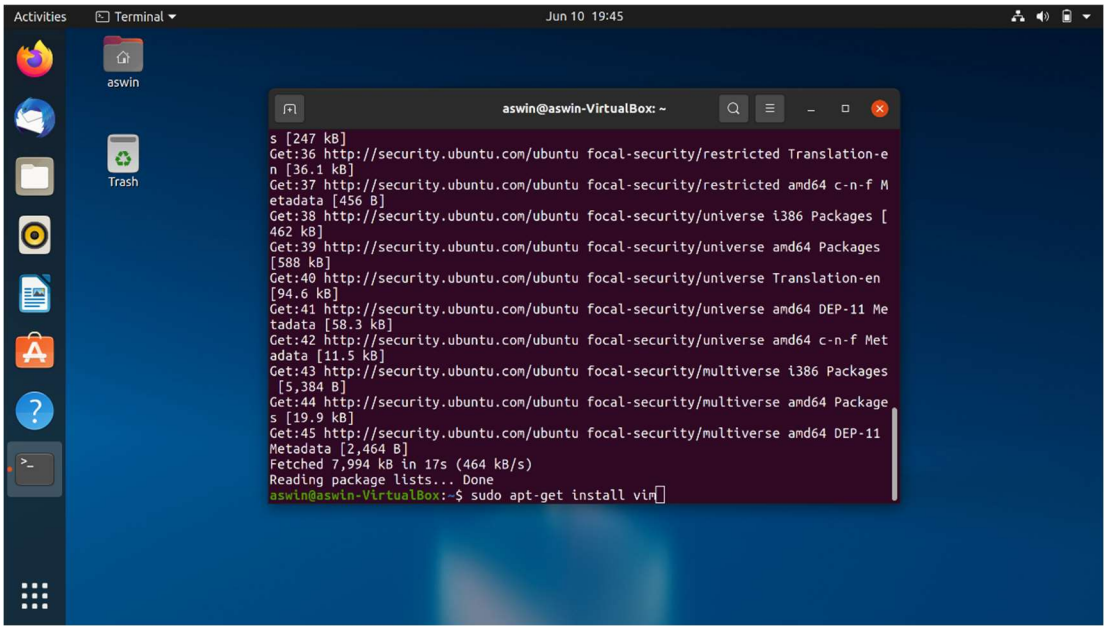
Ubuntu Linux install vim using apt

1. Open terminal application. You can also press **CTRL+ALT+T** keyboard shortcut.

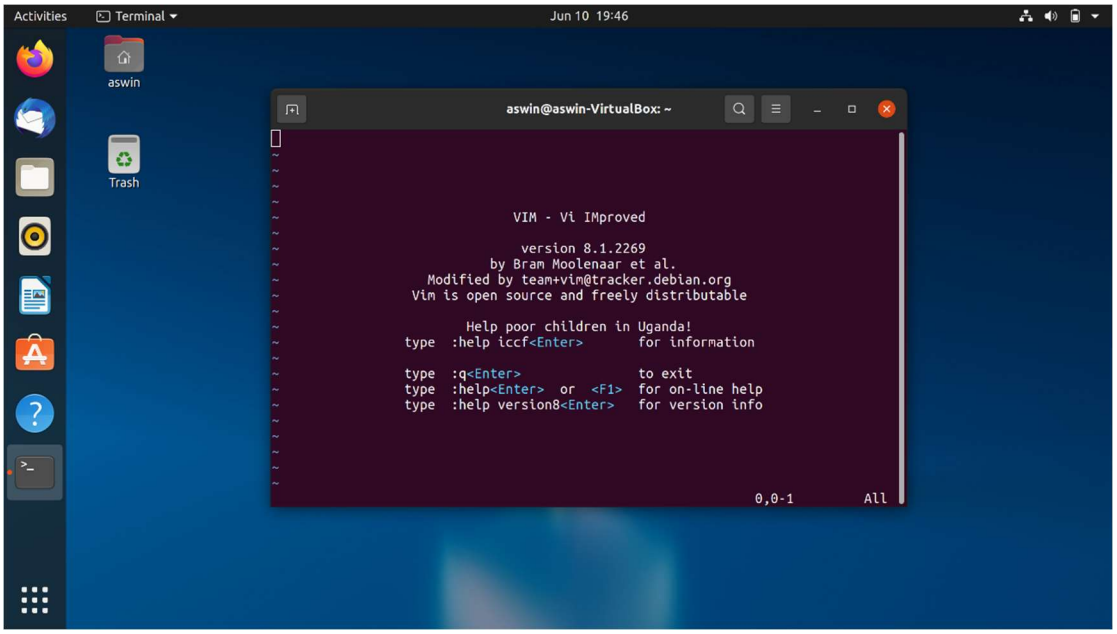
2. Update package database by typing the **sudo apt-get update** command.



3. Install vim on Ubuntu Linux, type: **sudo apt-get install vim** command.



4. Verify vim installation by typing the **vim -v** command.



Vim text editor is successfully installed.

VIM Modes

vim has three modes:

- Command Mode
- Input Mode
- Last Line Mode

1. Command Mode

   When vi starts up, it is in Command Mode. This mode is where vi interprets any characters we type as commands and thus does not display them in the window. This mode allows us to move through a file, and to delete, copy, or paste a piece of text. To enter into Command Mode from any other mode, it requires pressing the **Esc** key. If we press **Esc** when we are already in Command Mode, then vi will beep or flash the screen.

2. Insert Mode

   This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and finally, it is put in the file. The vi always starts in command mode. To enter text, you must be in insert mode. To come in insert mode you simply type i. To get out of insert mode, press the Esc key, which will put you back into command mode.

3. Last Line Mode

   Line Mode is invoked by typing a colon [:], while vi is in Command Mode. The cursor will jump to the last line of the screen and vi will wait for a command. This mode enables you to perform tasks such as saving files, executing commands.

Vim commands

The following is a list of frequently used commands and what they do. Many of the commands can be made to repeat by adding a number to the command. This is not an exhaustive list because more advanced commands, such as how to use multiple buffers, are not included. However, just about all of the basic commands for opening, editing and saving documents are included as well as commands that enable you to find and replace text and work with multiple documents.

1. Basic Vim Commands

`: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. Vim commands for movement

`h`	Moves the cursor to the left.
`i`	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. commands for editing

`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. commands for searching text

`?[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. commands for working with multiple files

`: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

`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

`: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.

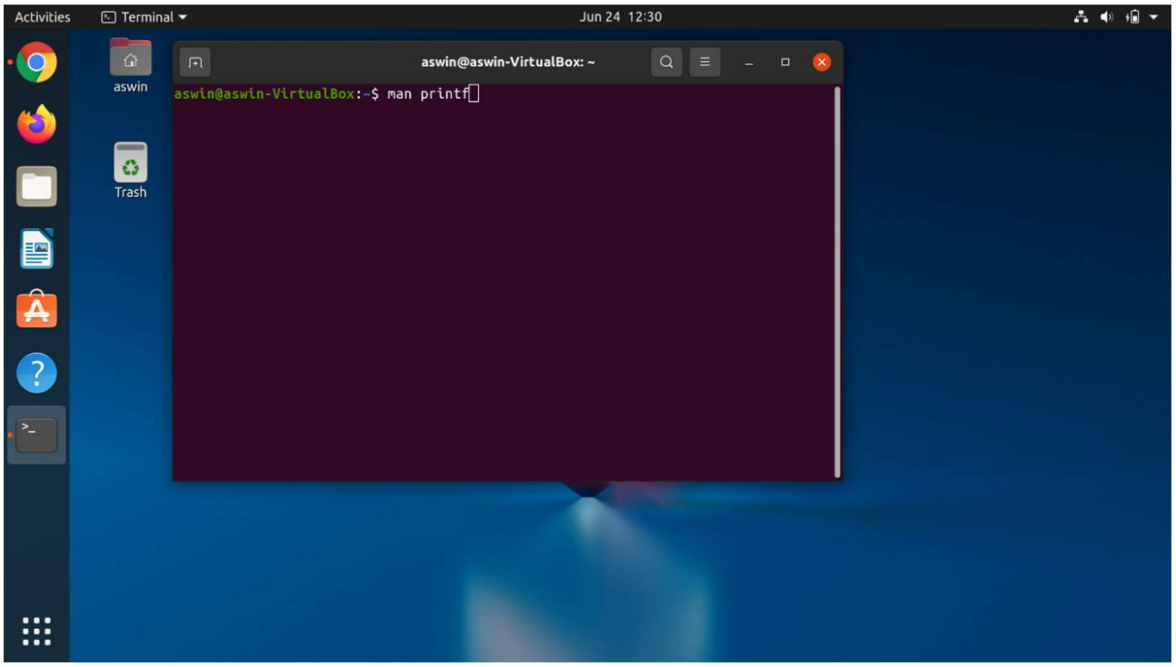
BASIC LINUX COMMANDS

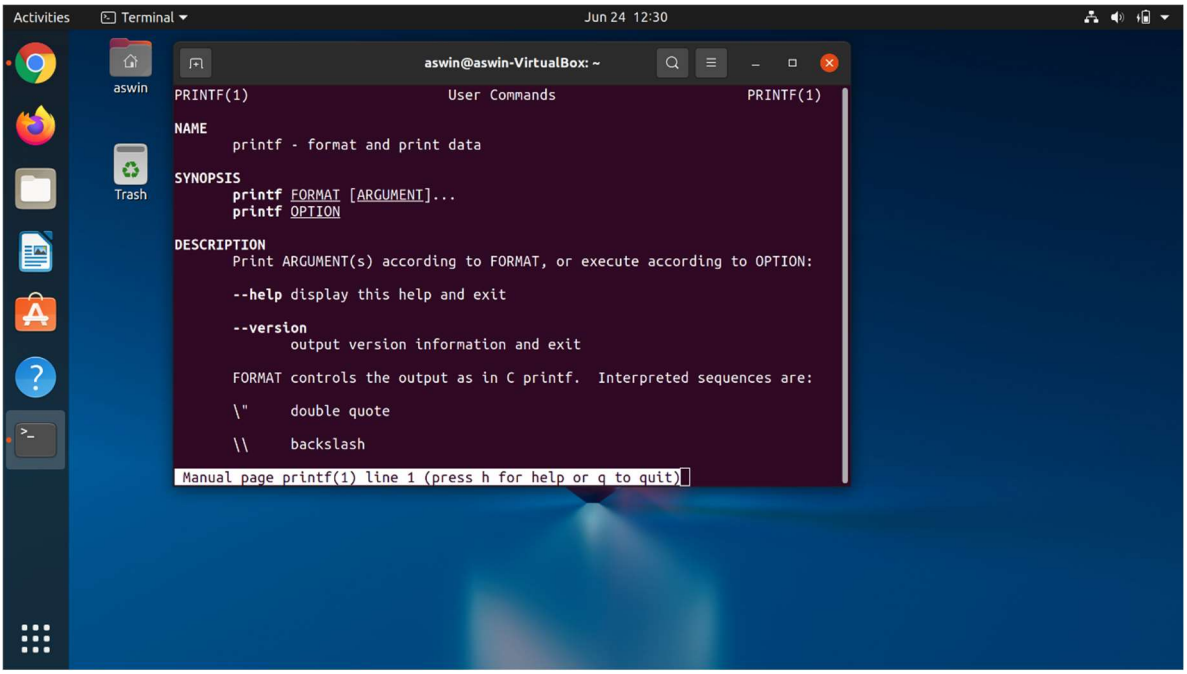
1. Man

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]...
```



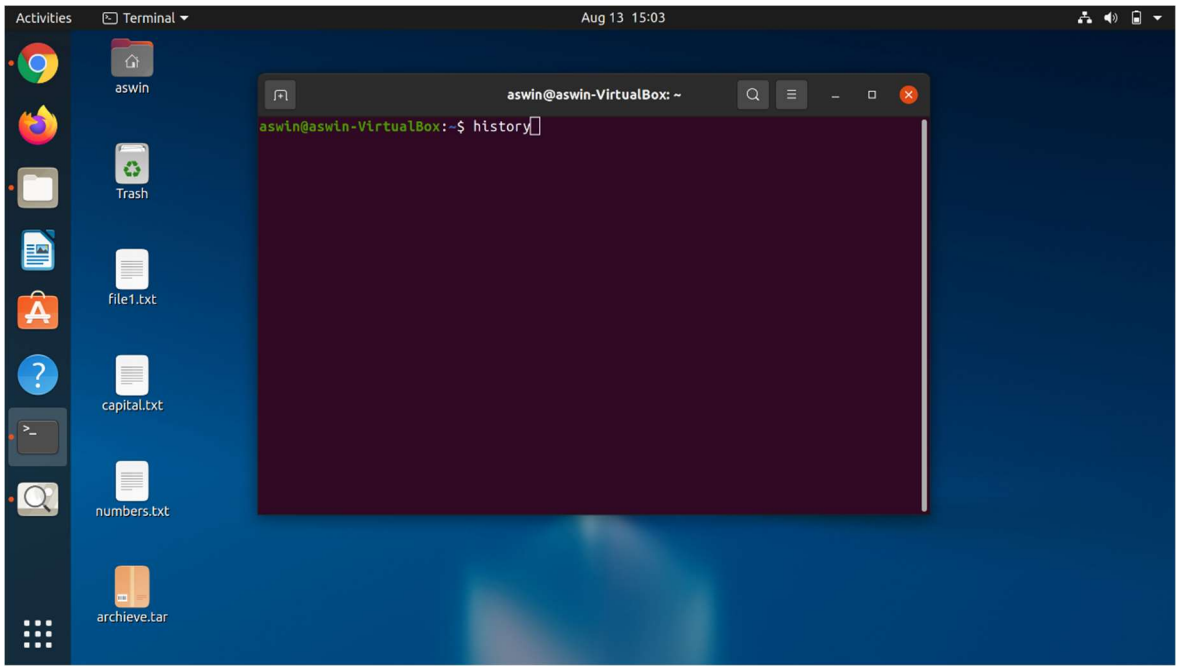


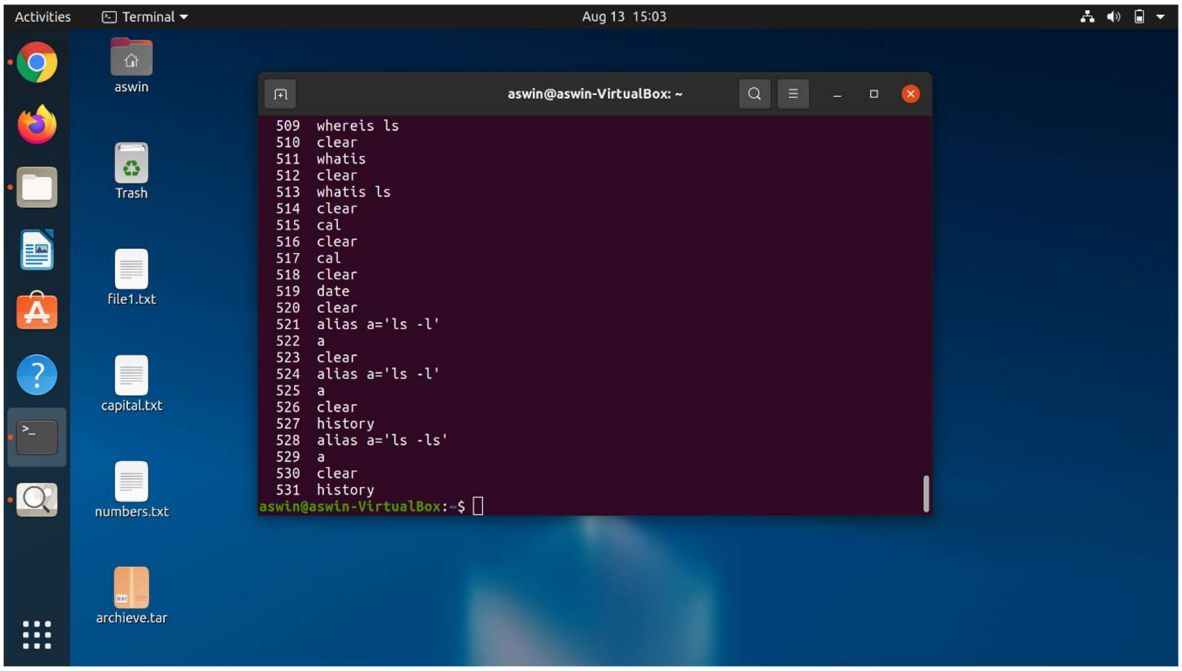
2. history

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
```



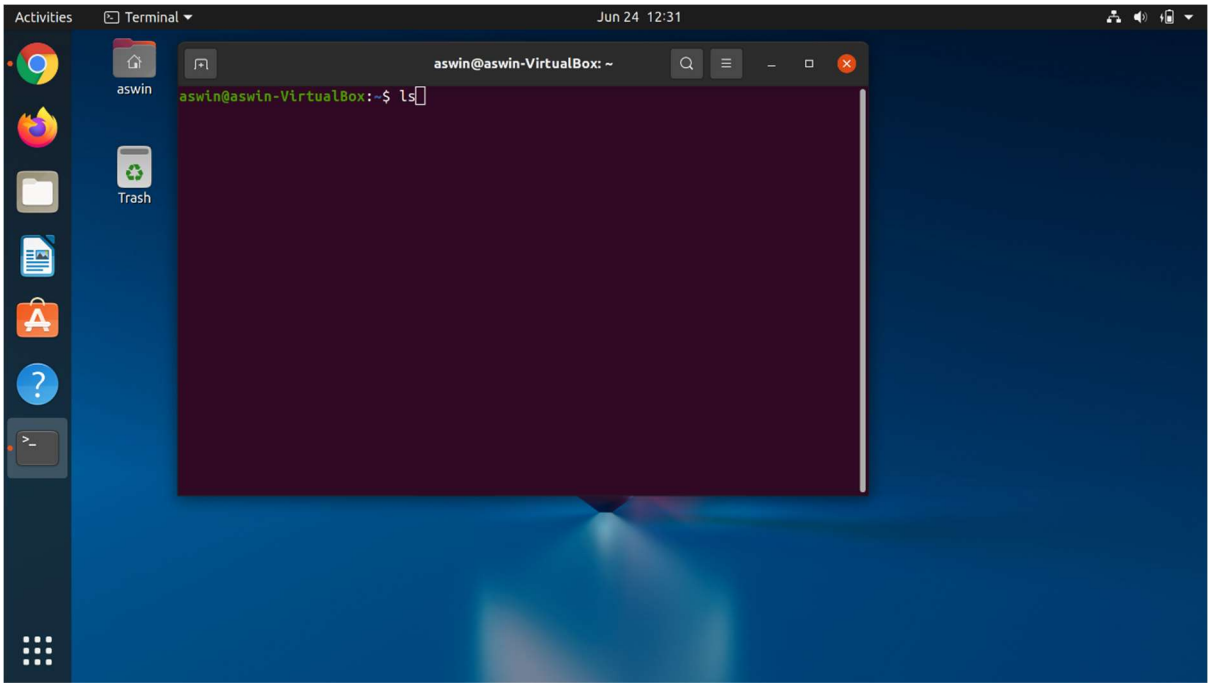


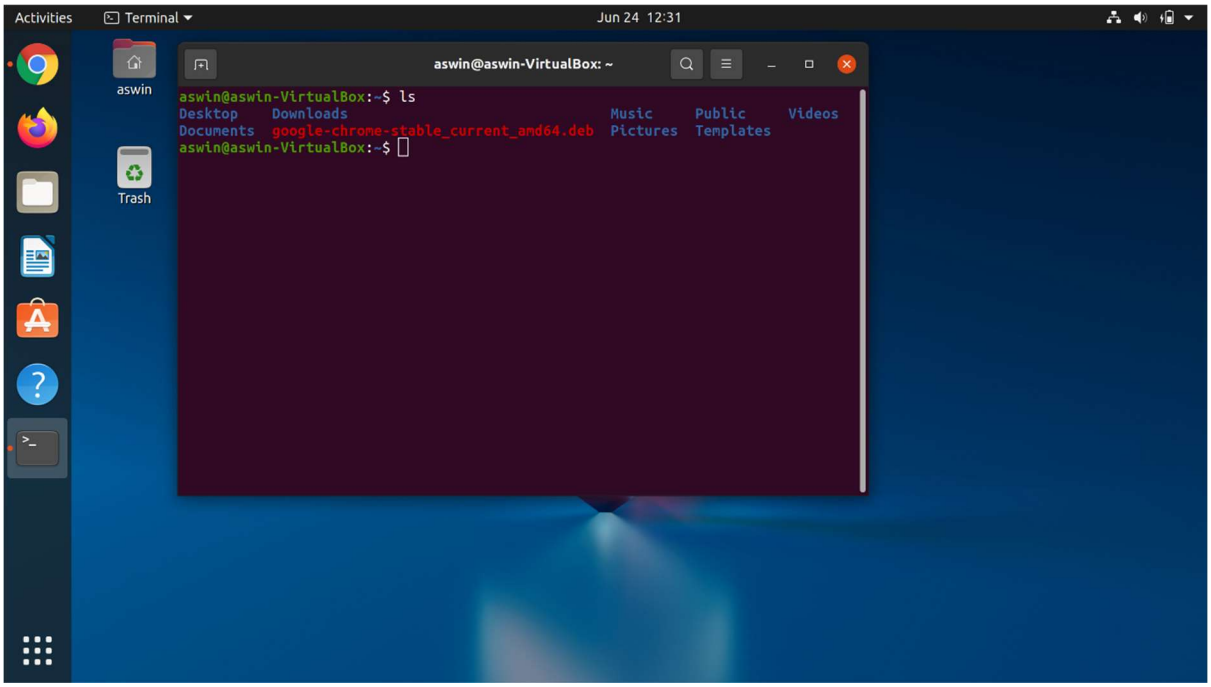
3. ls

The **ls** is the list command in Linux. It will show the full list or content of your directory. Just type *ls* and press the enter key. The whole content will be shown.

Syntax:

```
$ ls
```



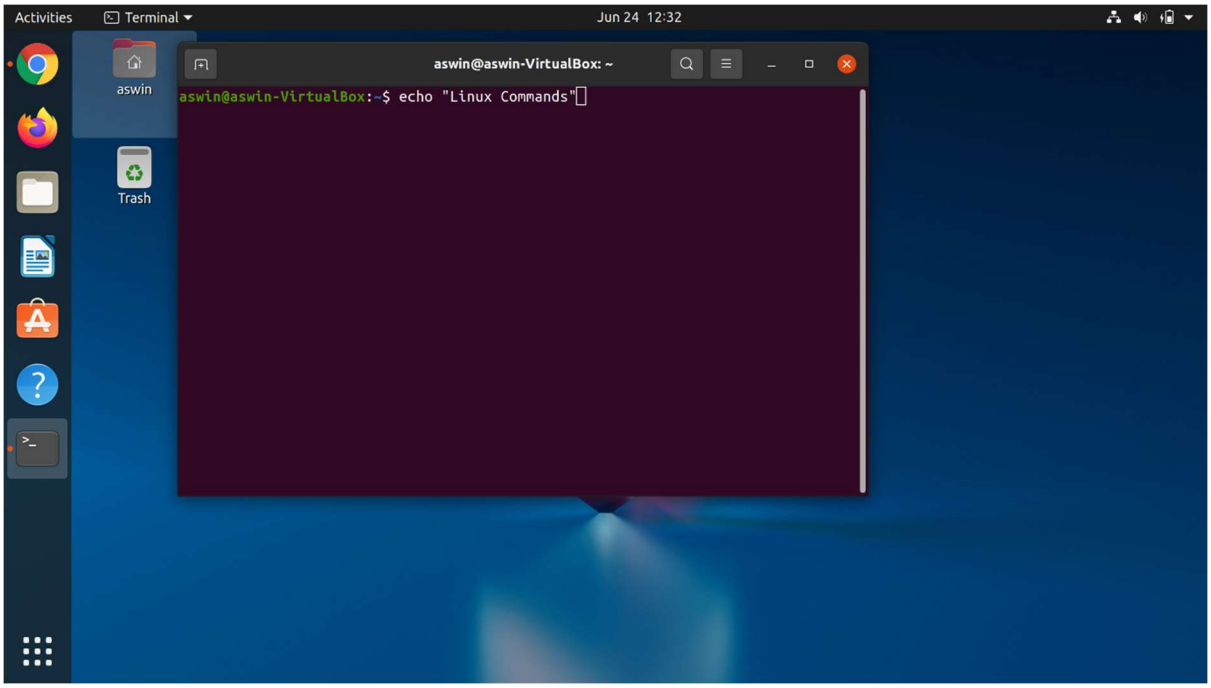


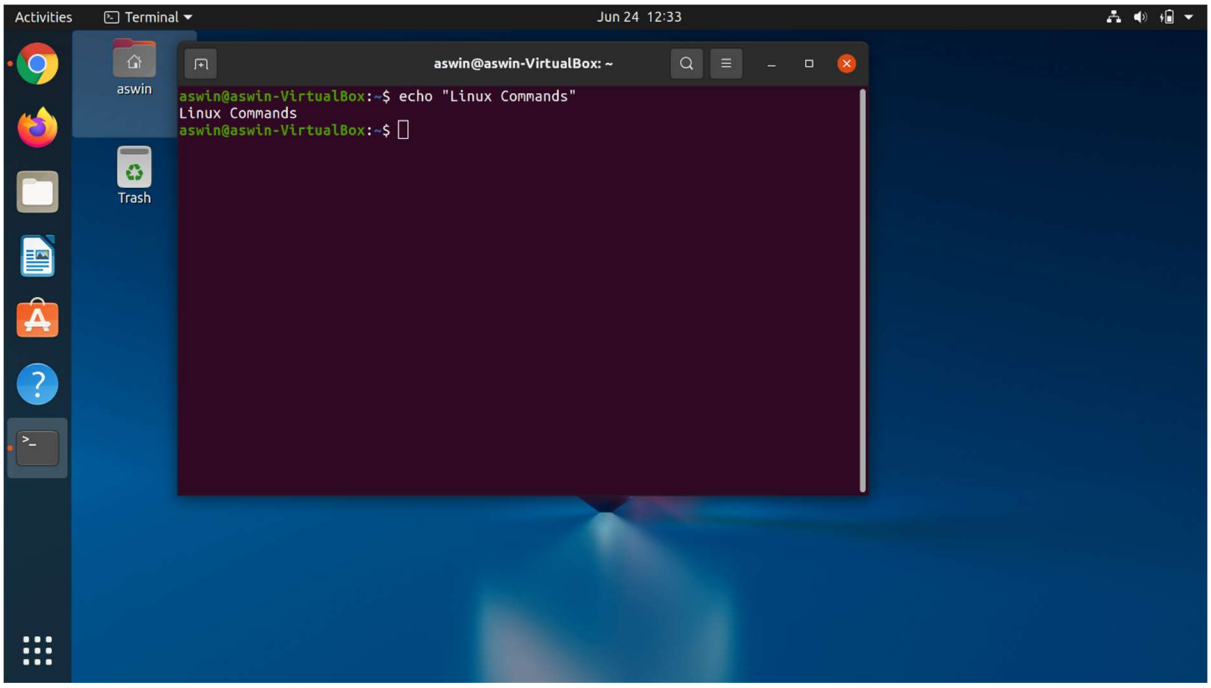
4. echo

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]
```



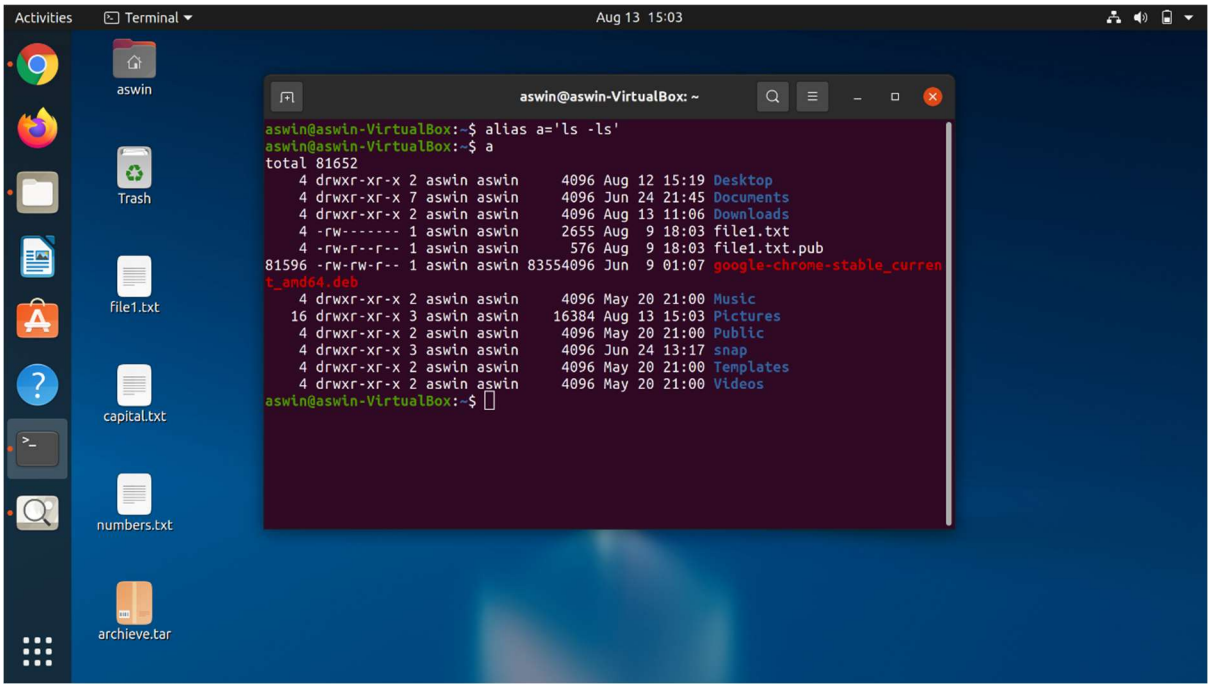


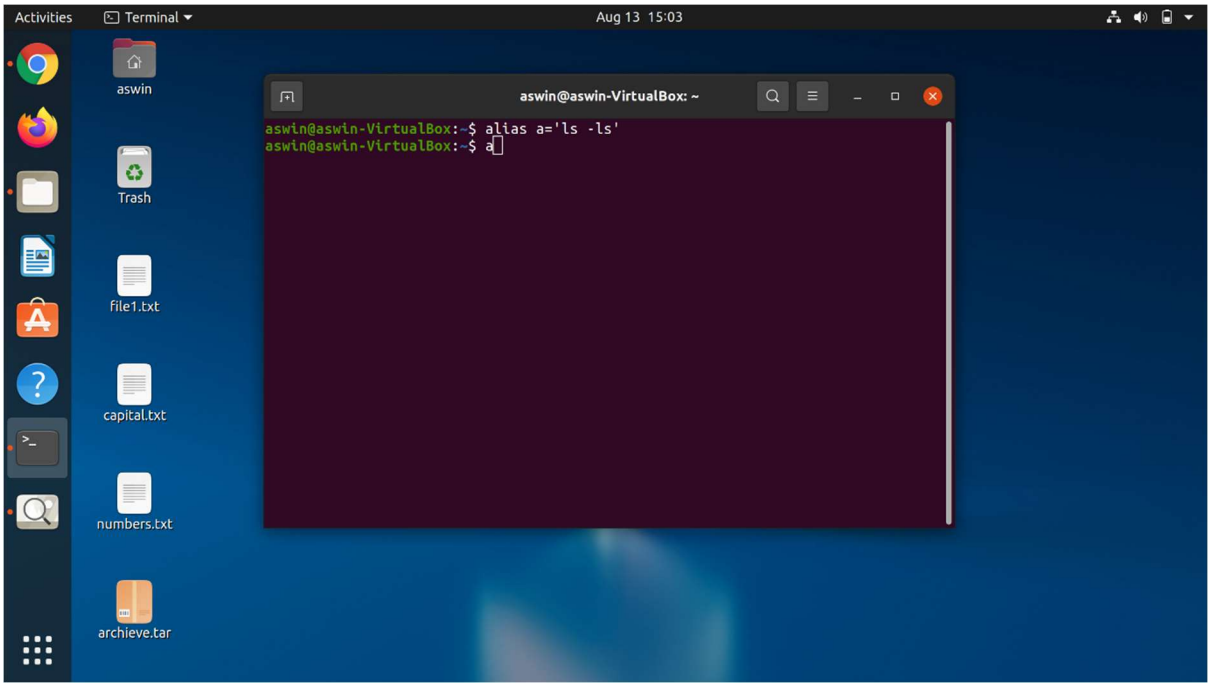
5. alias

alias command instructs the shell to replace one string with another string while executing the commands.
When we often have to use a single big command multiple times, in those cases, we create something called as alias for that command. Alias is like a shortcut command which will have same functionality as if we are writing the whole command.

Syntax:

```
alias [-p] [name[=value] ... ]
```



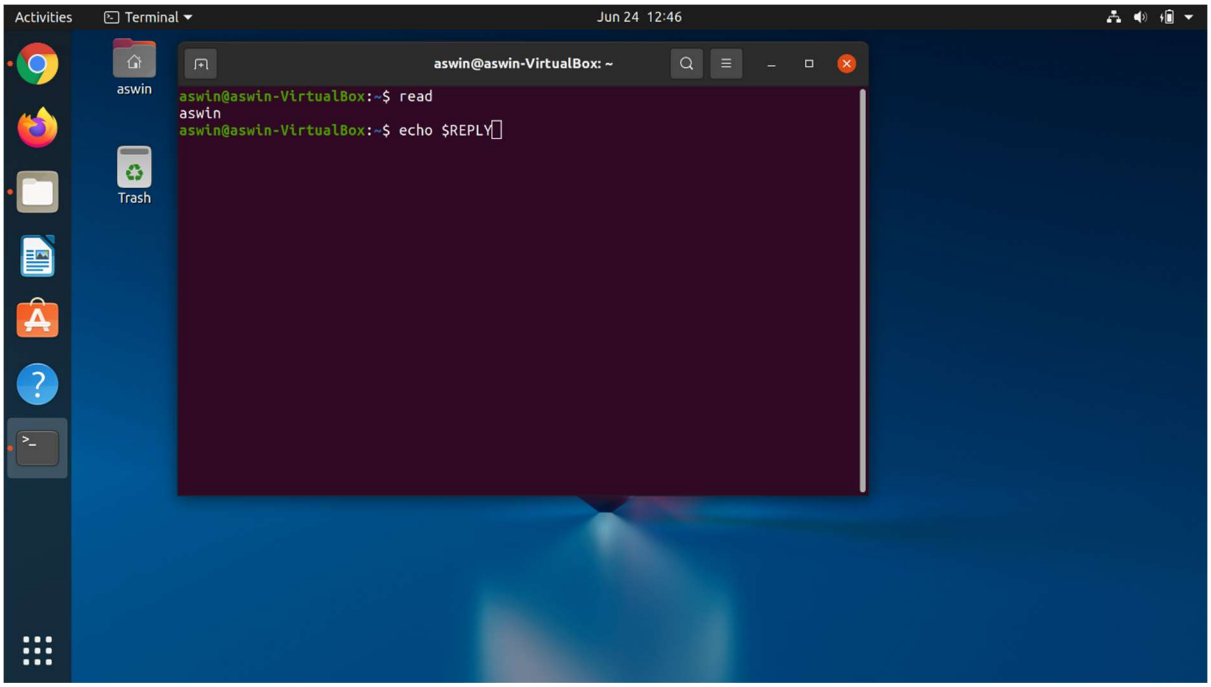


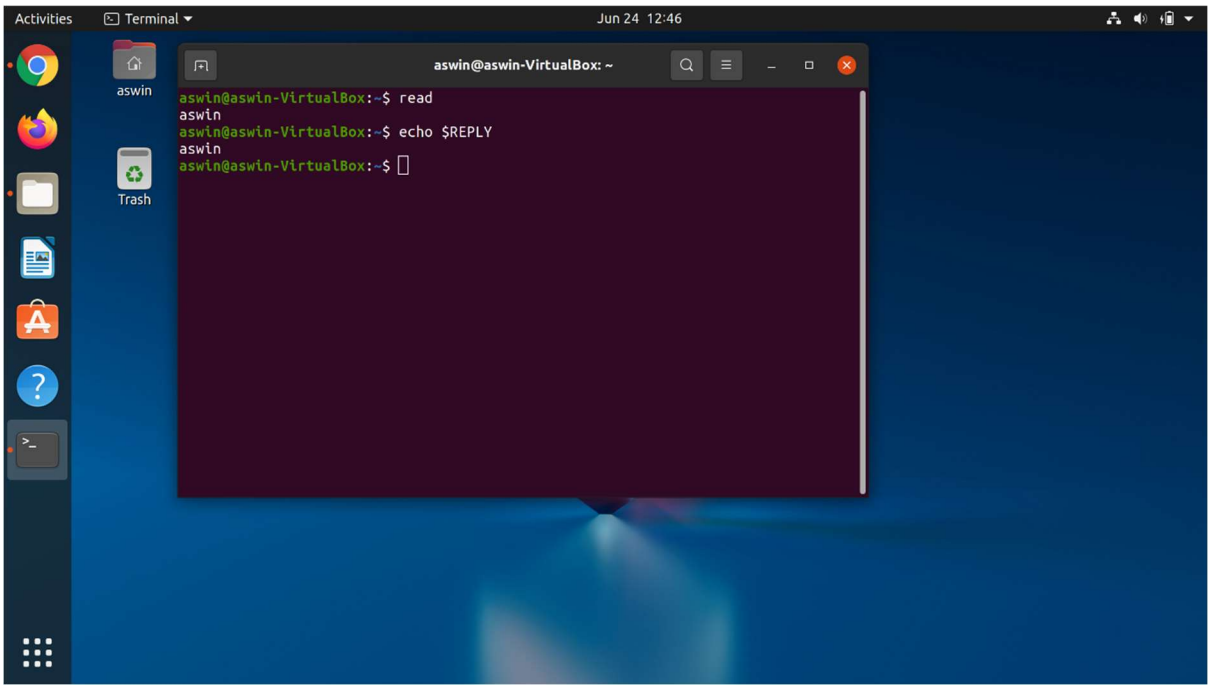
6. read

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
```



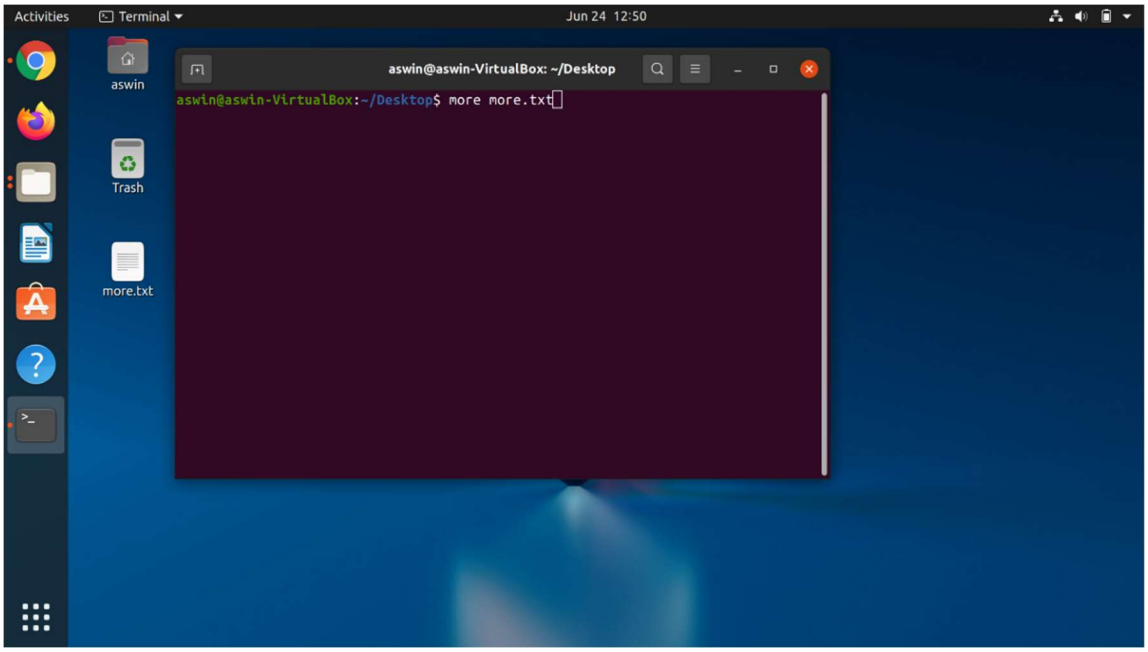


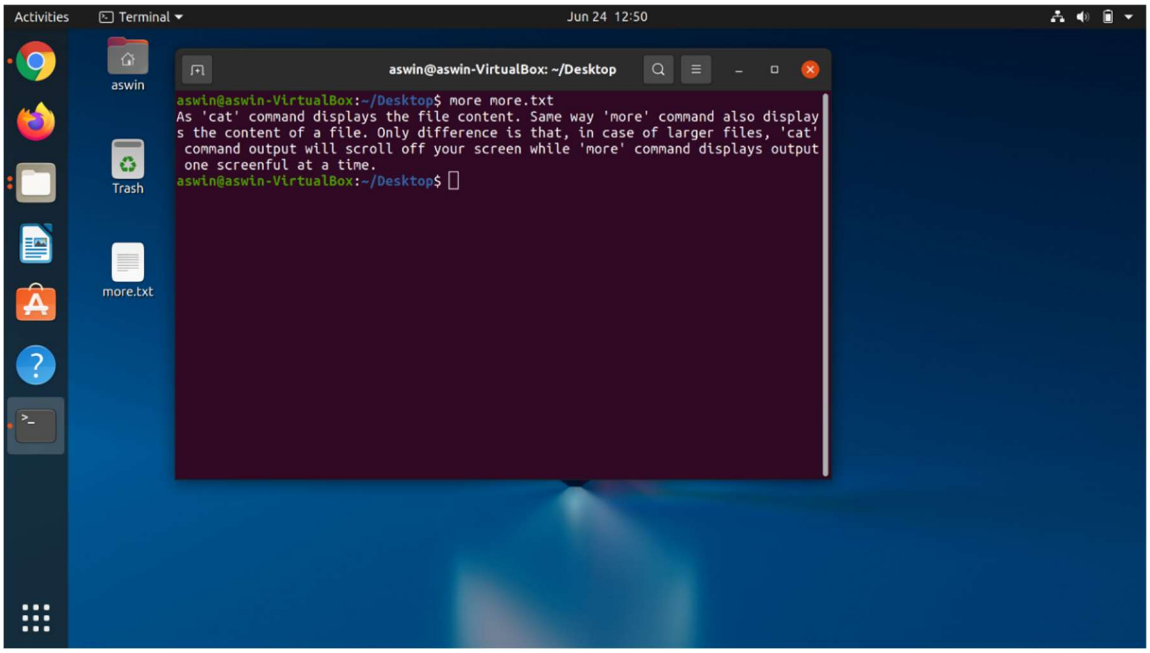
7. more

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 (For example log files). The more command also allows the user do scroll up and down through the page. The syntax along with options and command is as follows. Another application of more is to use it with some other command after a pipe. When the output is large, we can use more command to see output one by one.

Syntax:

```
more [-options] [-num] [+/pattern] [+linenum] [file_name]
```



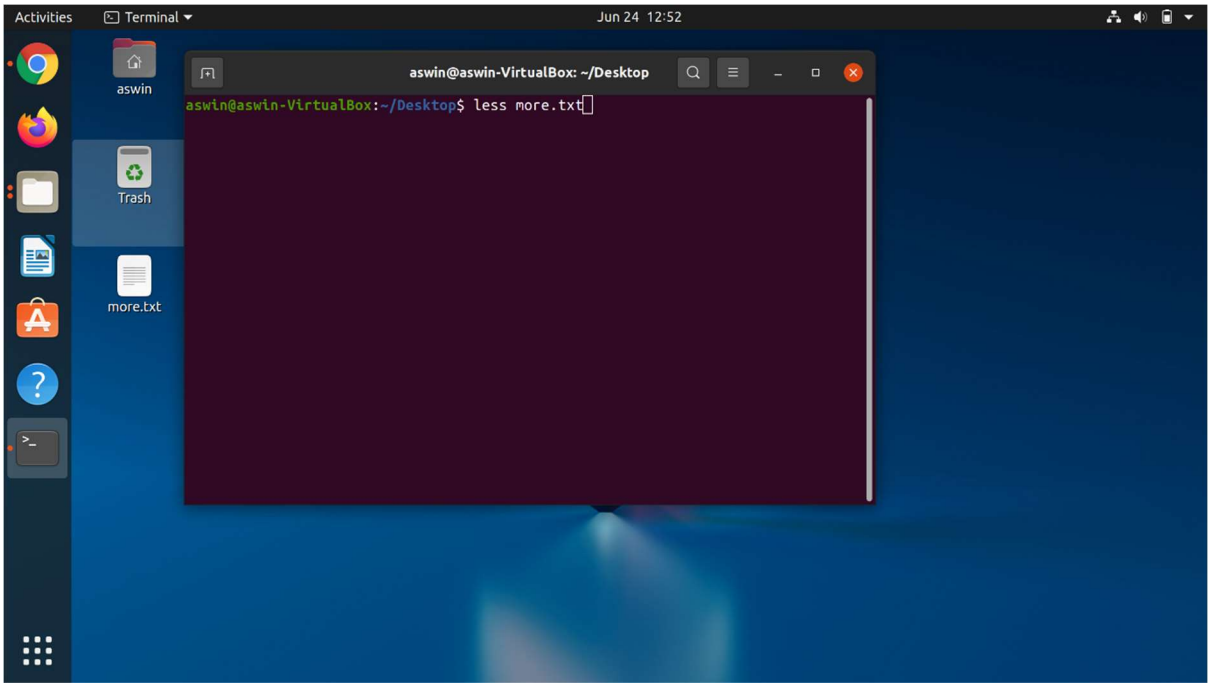


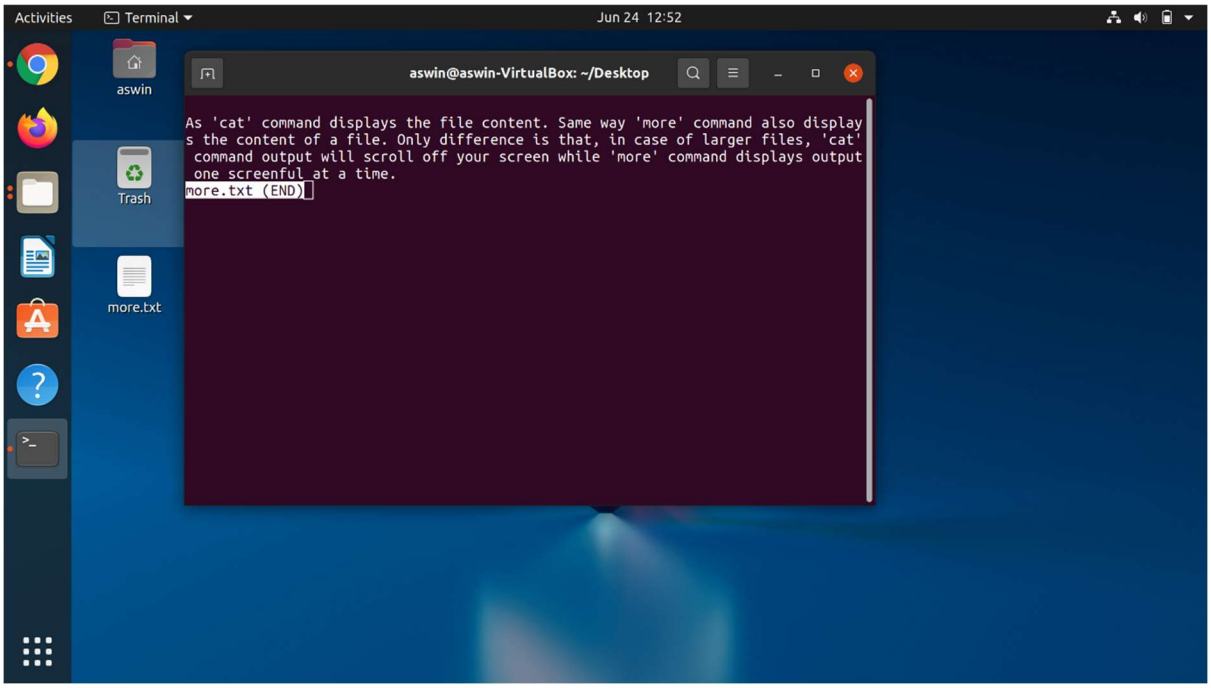
8. less

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
```



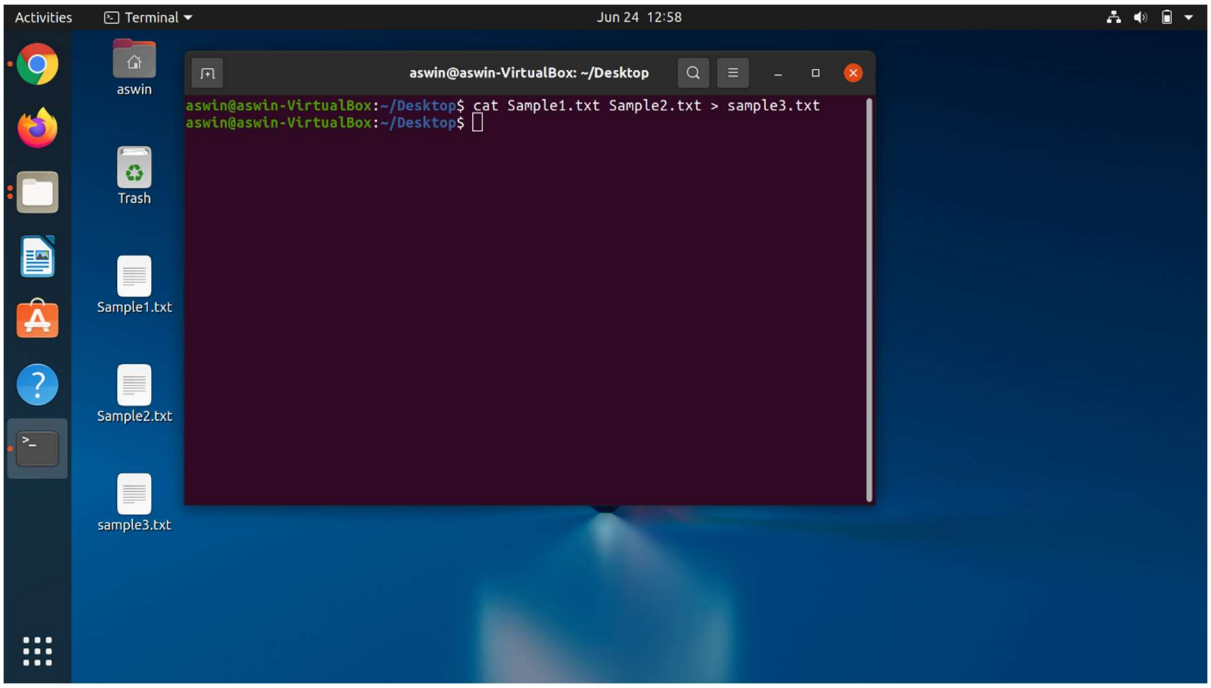


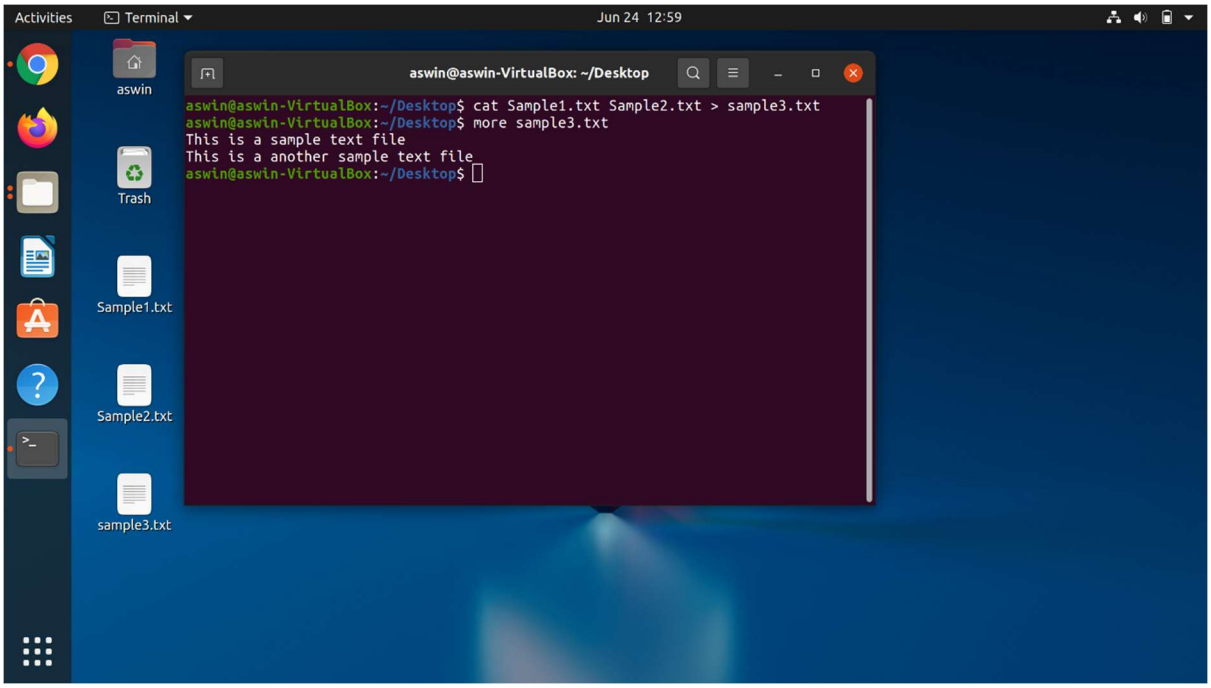
9. cat

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.

Syntax:

```
cat [OPTION] [FILE]...
```



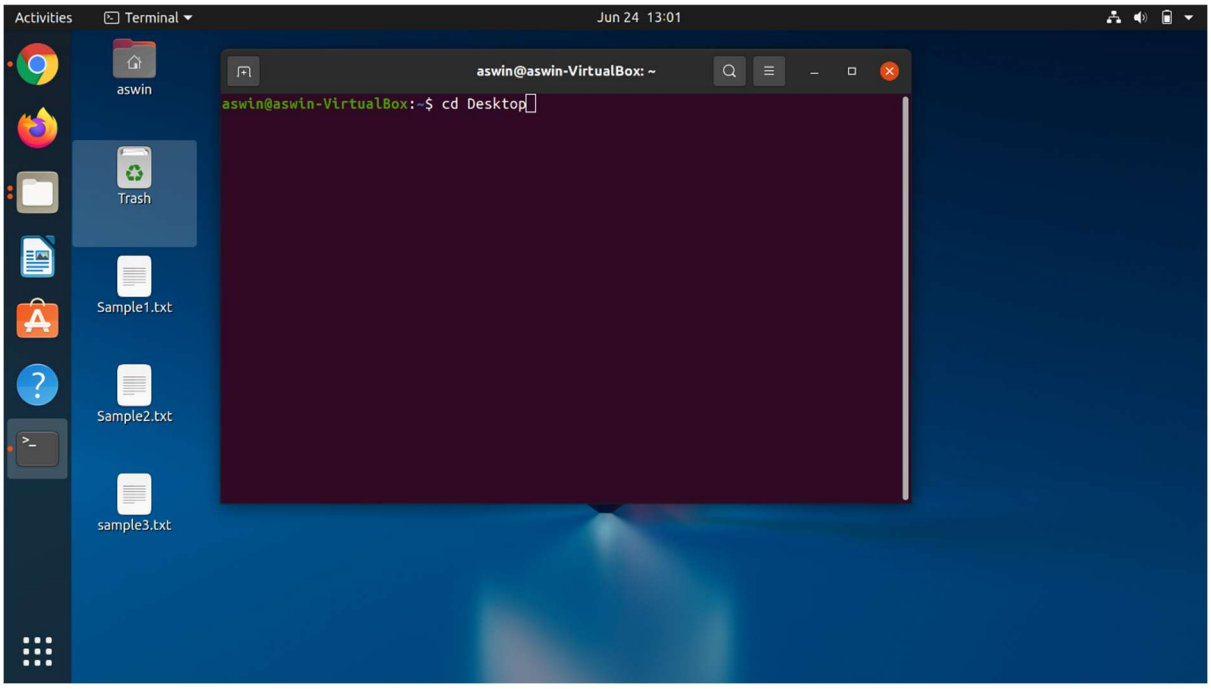


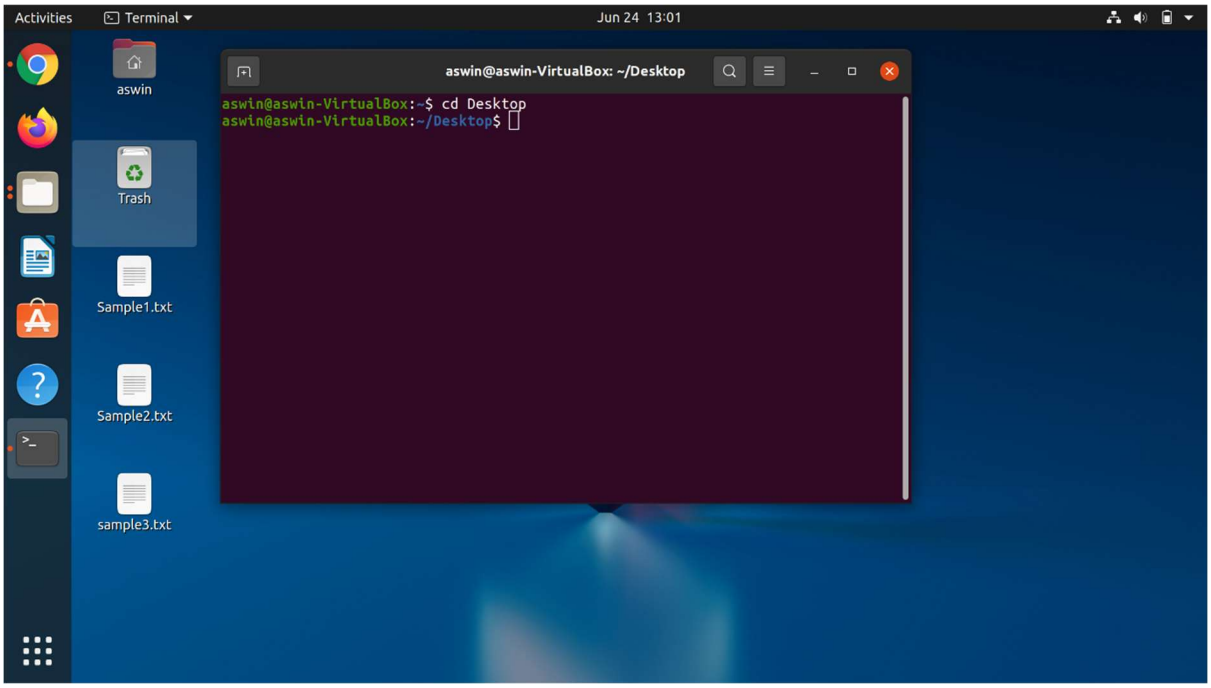
10. cd

cd command in Linux known as change directory command. It is used to change current working directory. In the above example, we have checked number of directories in our home directory and moved inside the Documents directory by using cd Documents command.

Syntax:

```
$ cd [directory]
```



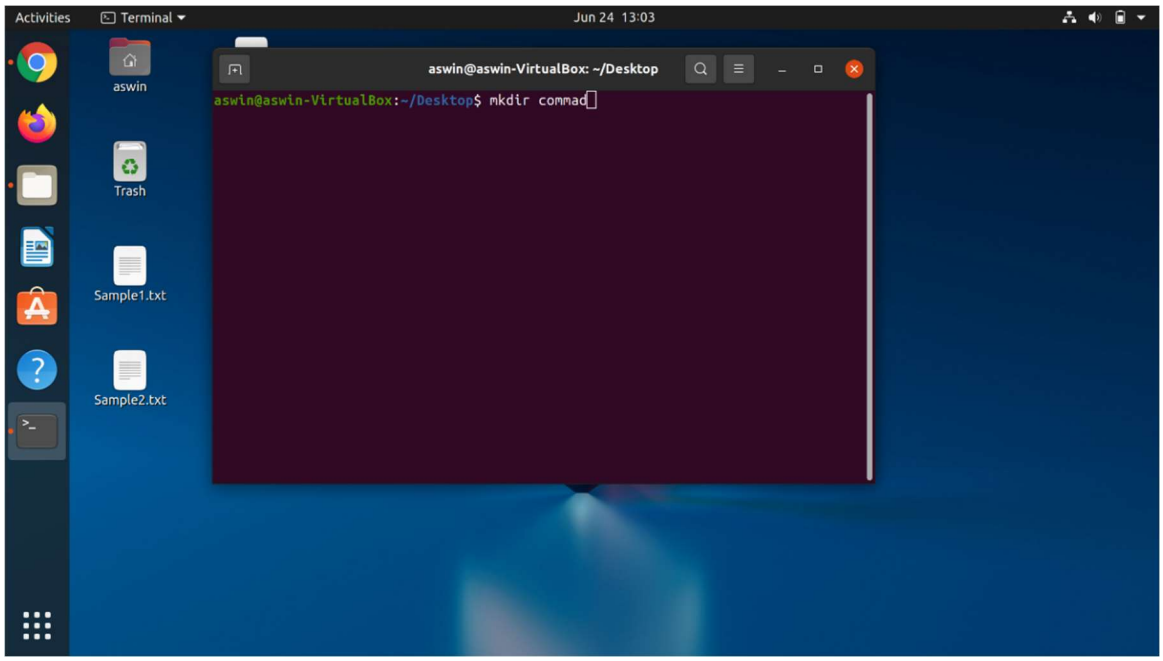


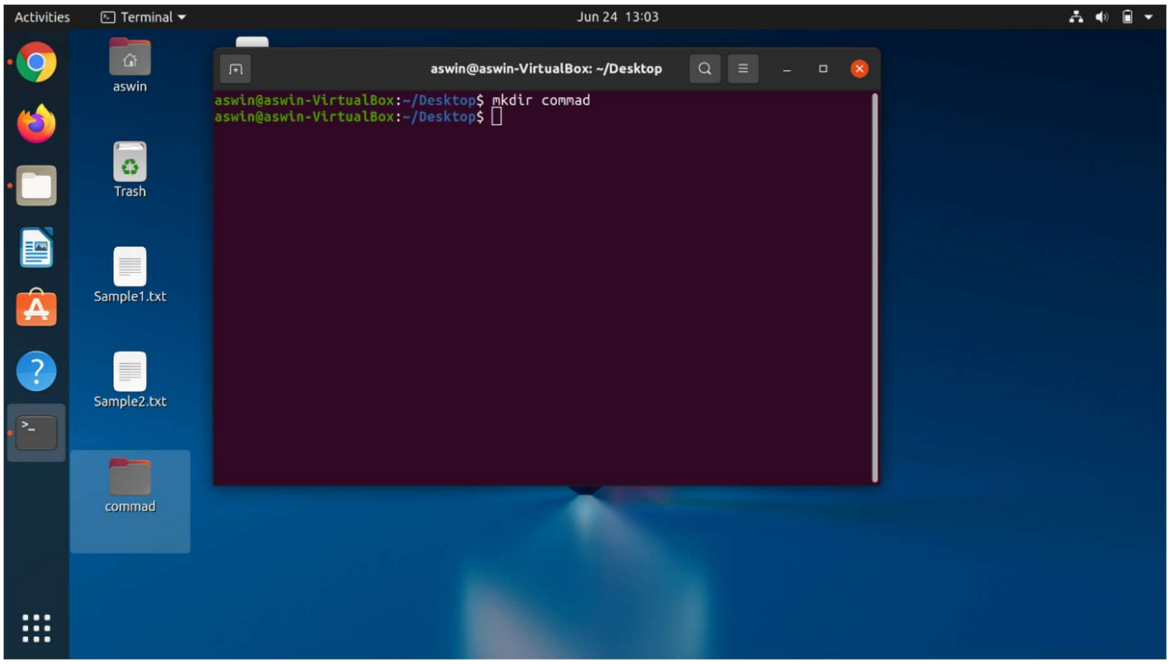
11. mkdir

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 ...]
```



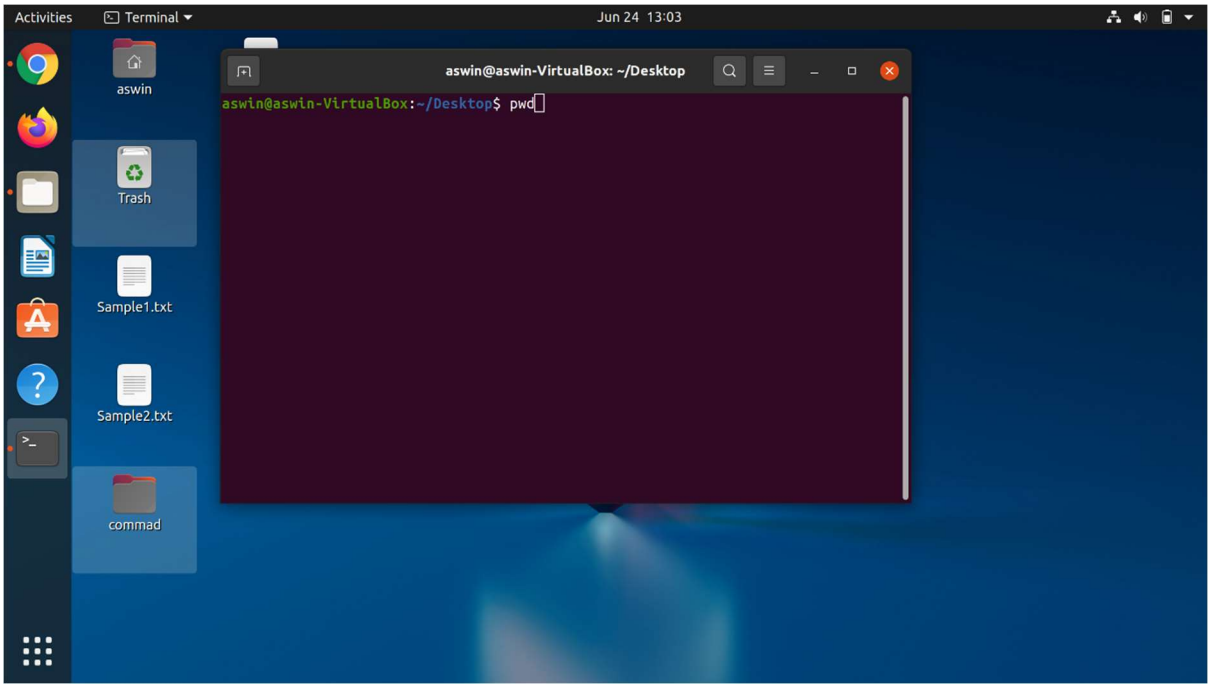


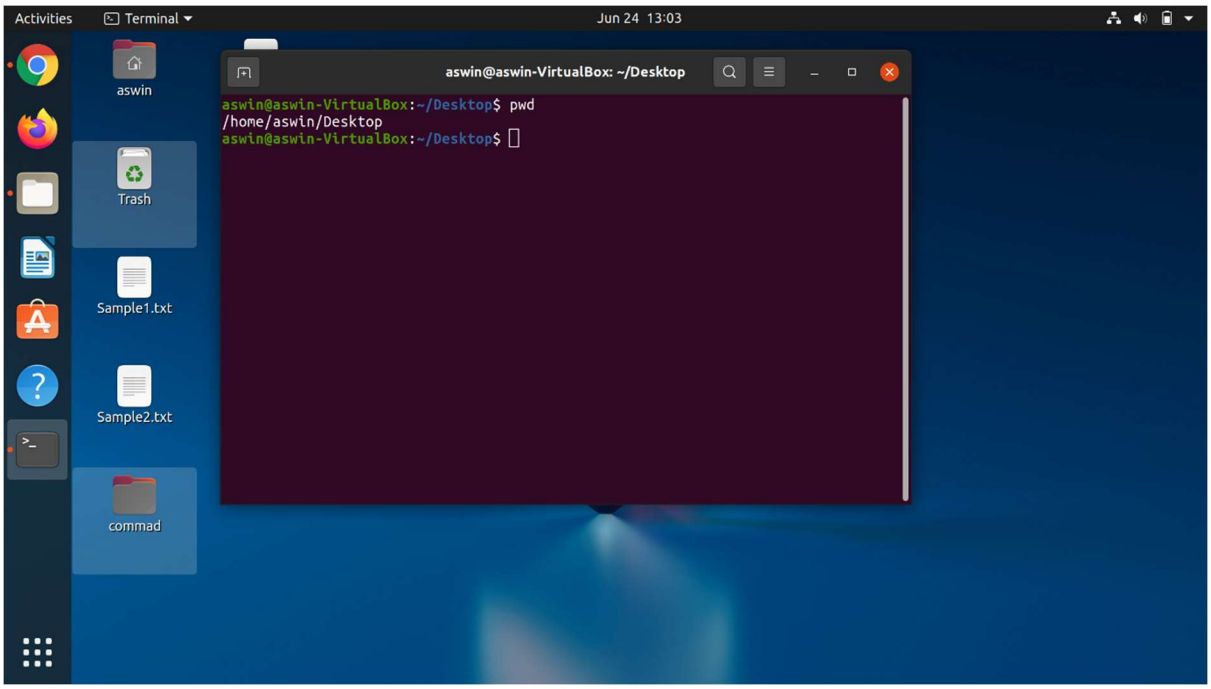
12. pwd

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
```



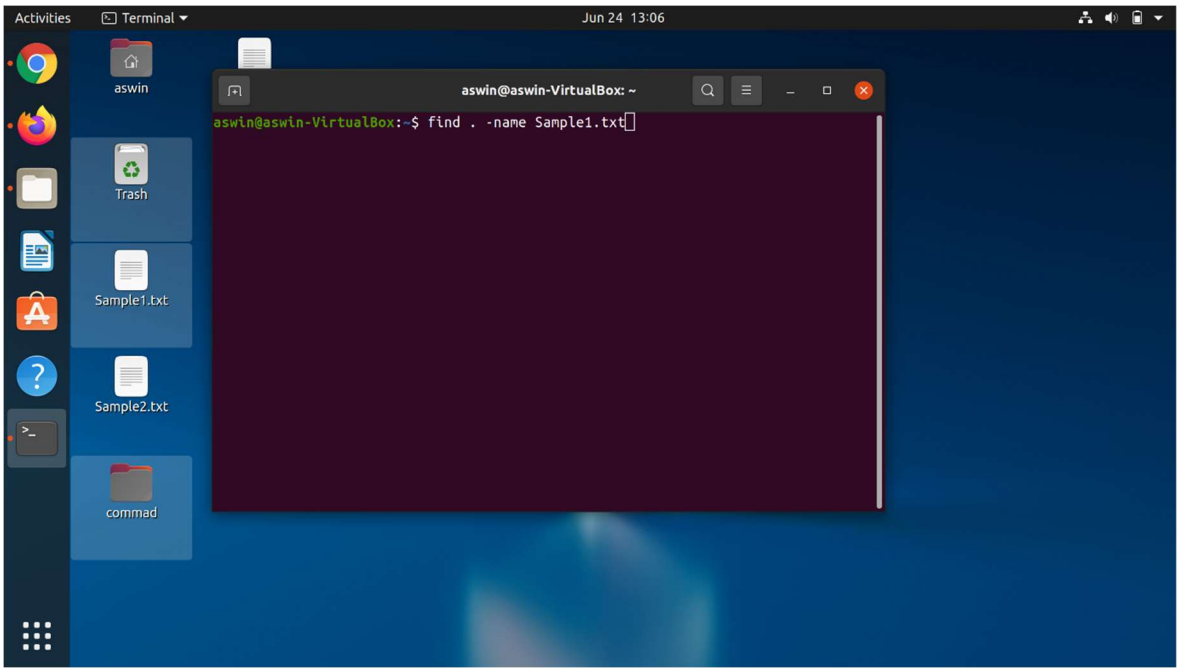


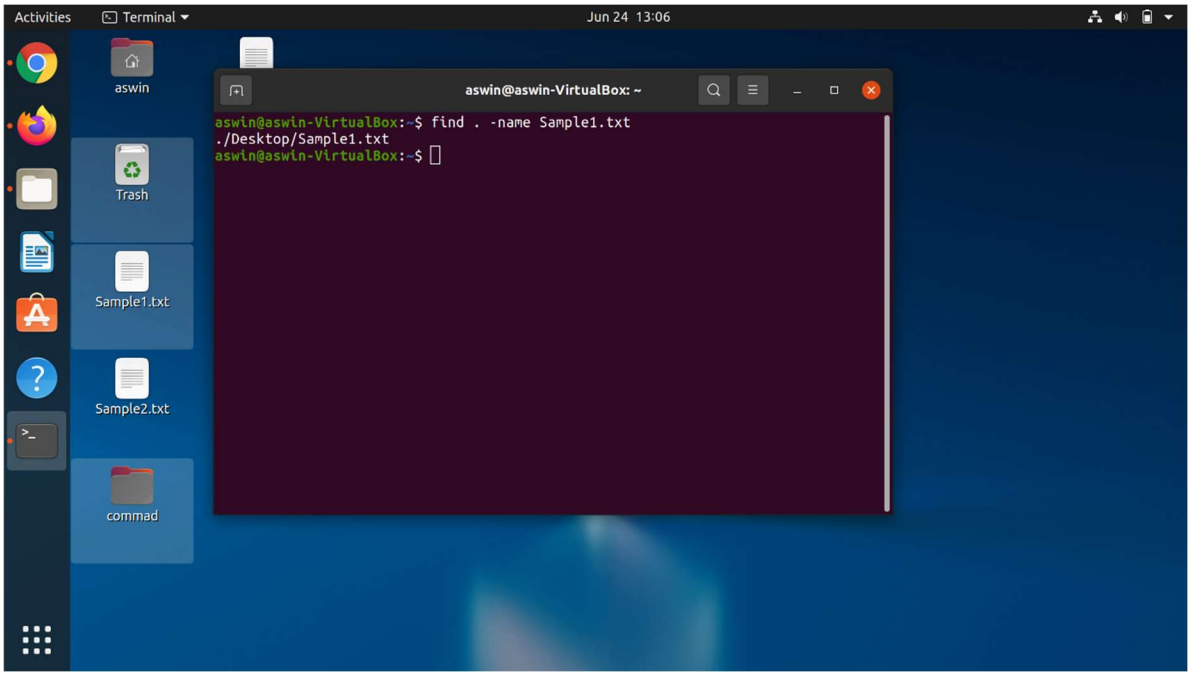
13. find

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. 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]
```



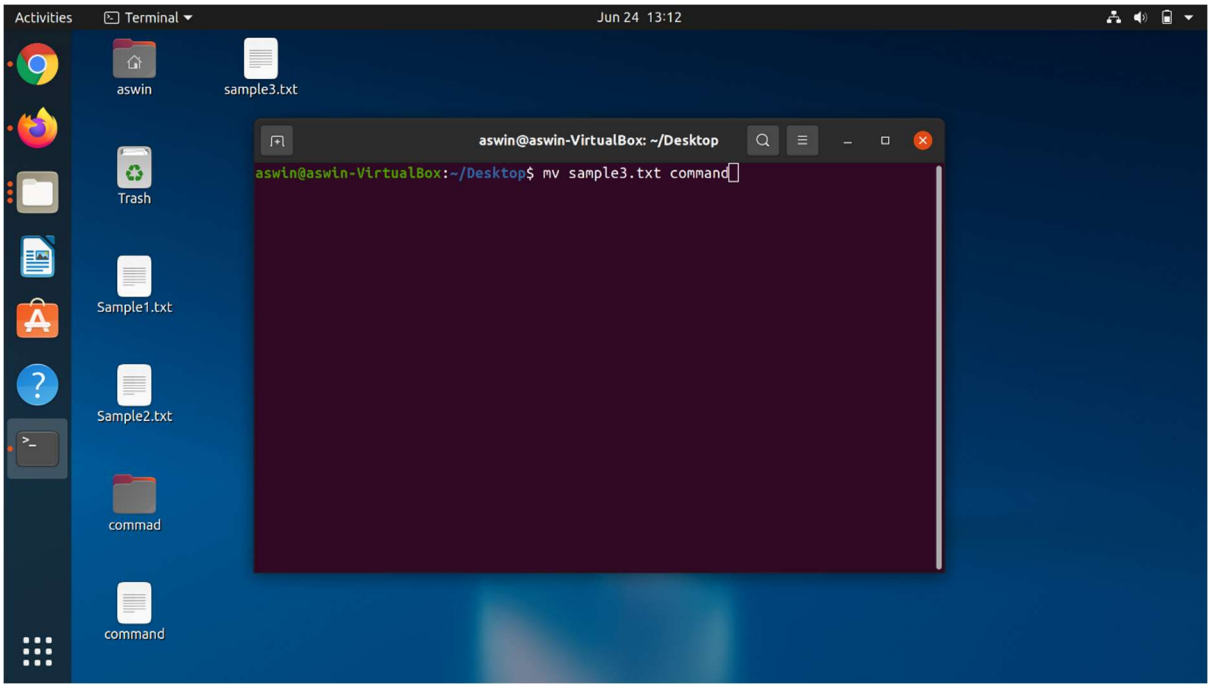


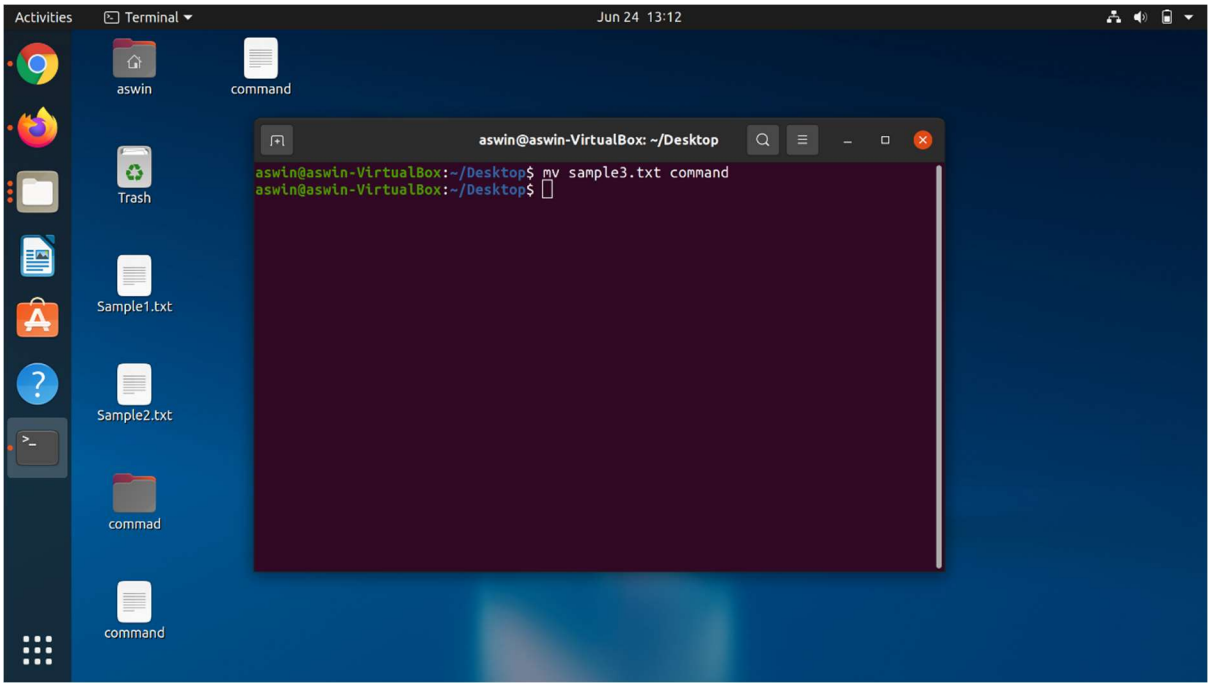
14. mv

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
```



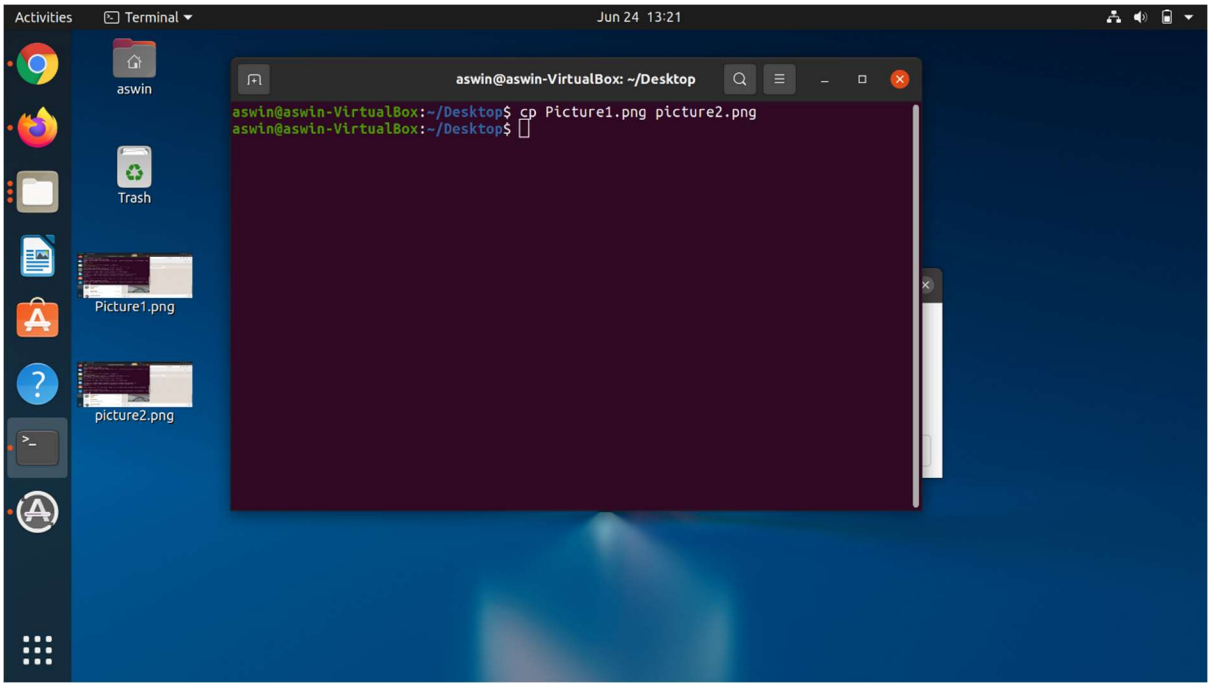


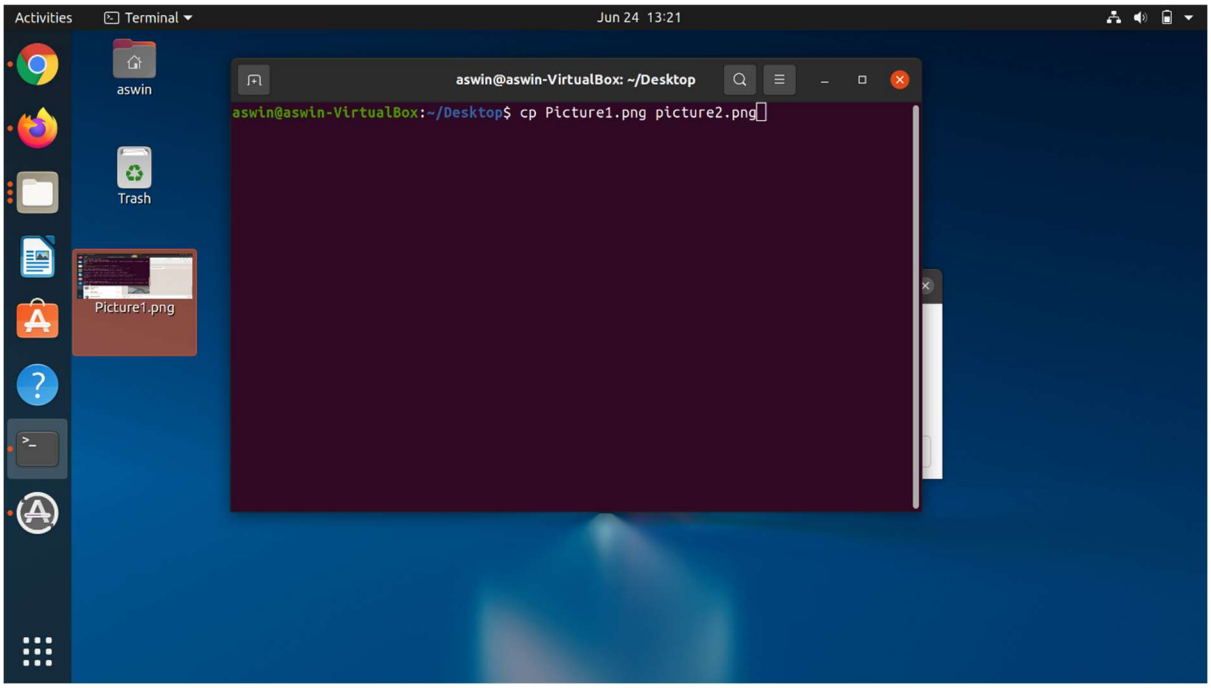
15. cp

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 requires at least two filenames in its arguments.

Syntax:

```
cp [OPTION] Source Destination
```



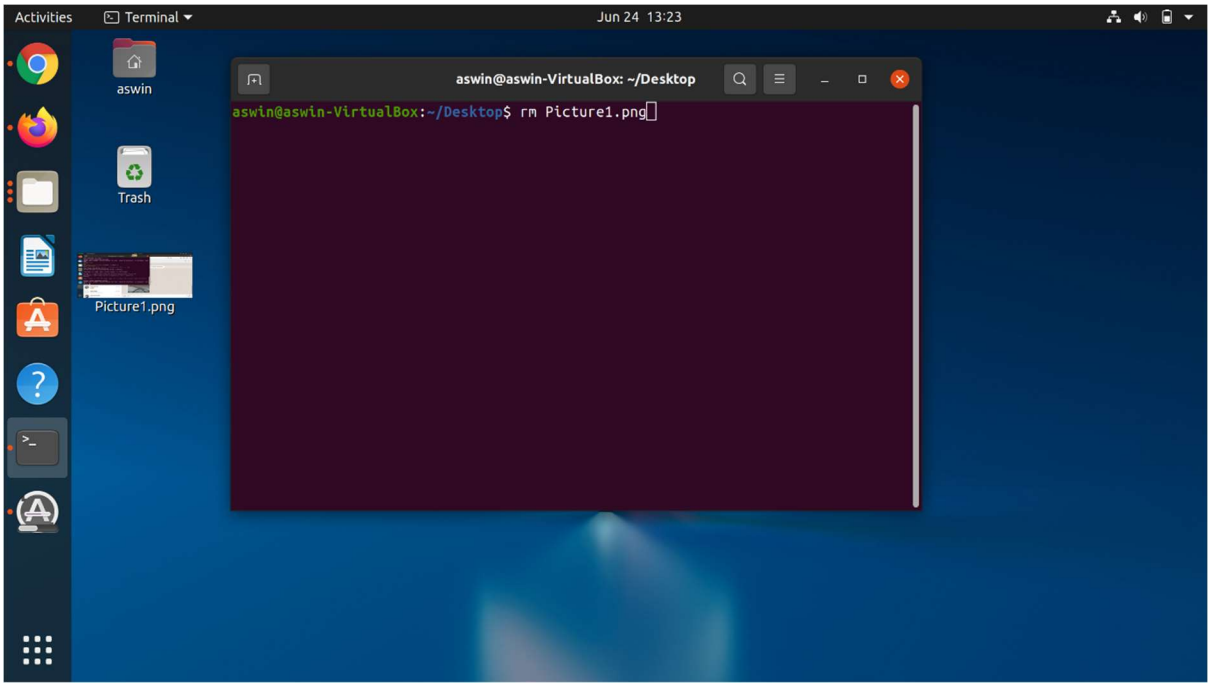


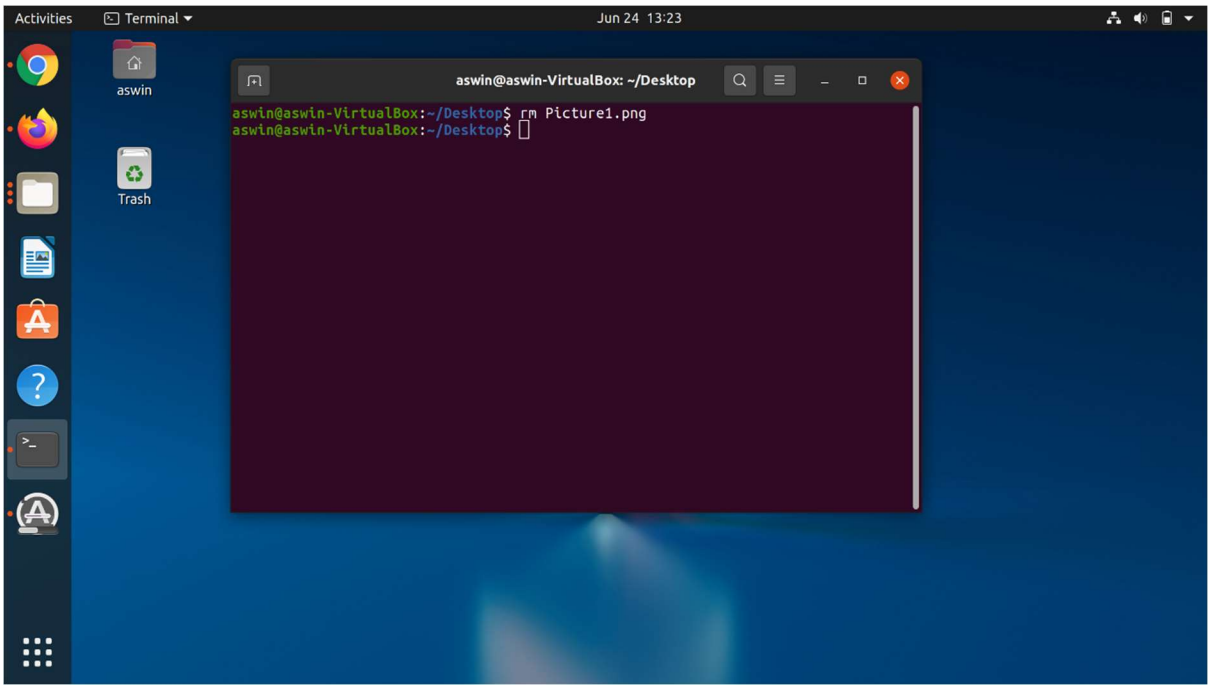
16. rm

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. 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.

Syntax:

```
rm [OPTION]... FILE...
```





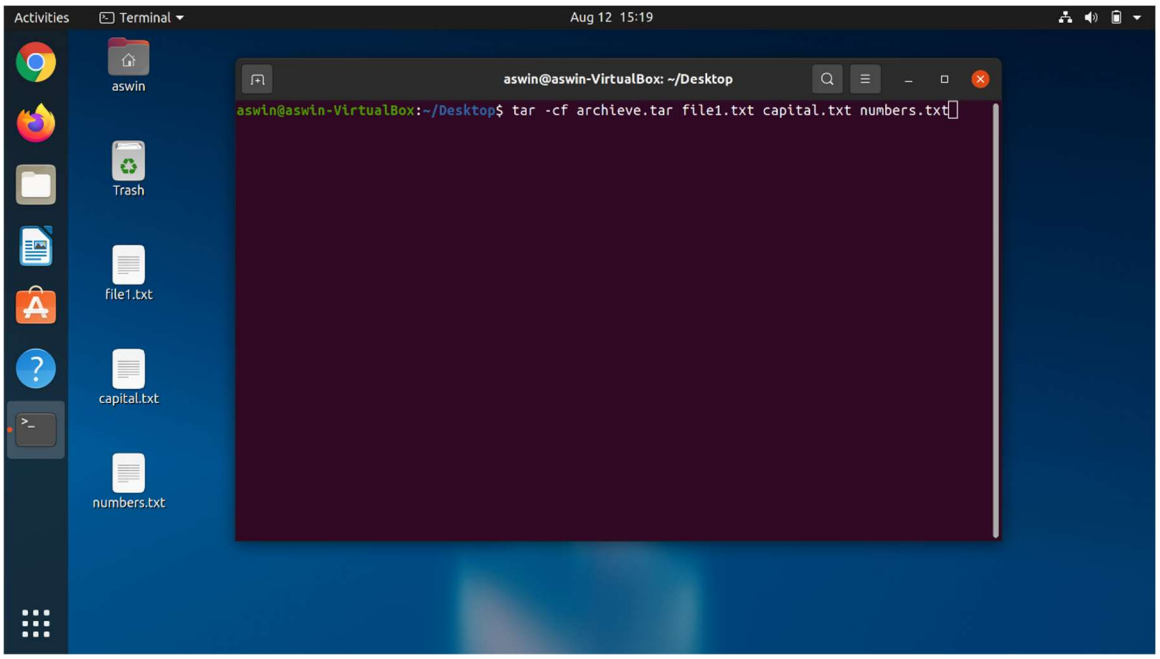
17. tar

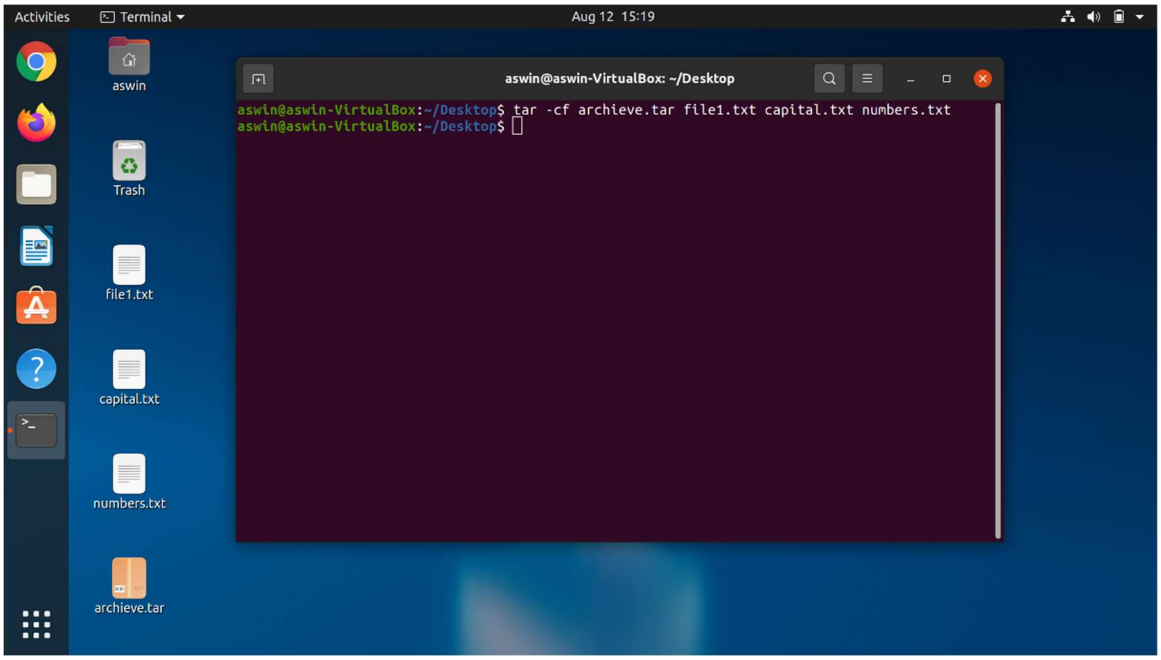
The Linux ‘tar’ stands for tape archive, is used to create Archive and extract the Archive files. tar command in Linux is one of the important commands which provides archiving functionality in Linux. We can use Linux tar command to create compressed or uncompressed Archive files and also maintain and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

Syntax:

```
tar [options] [archive-file] [file or directory to be archived]
```

For example, to create an archive named archive.tar from the files named file1, file2, file3, you would run the following command:



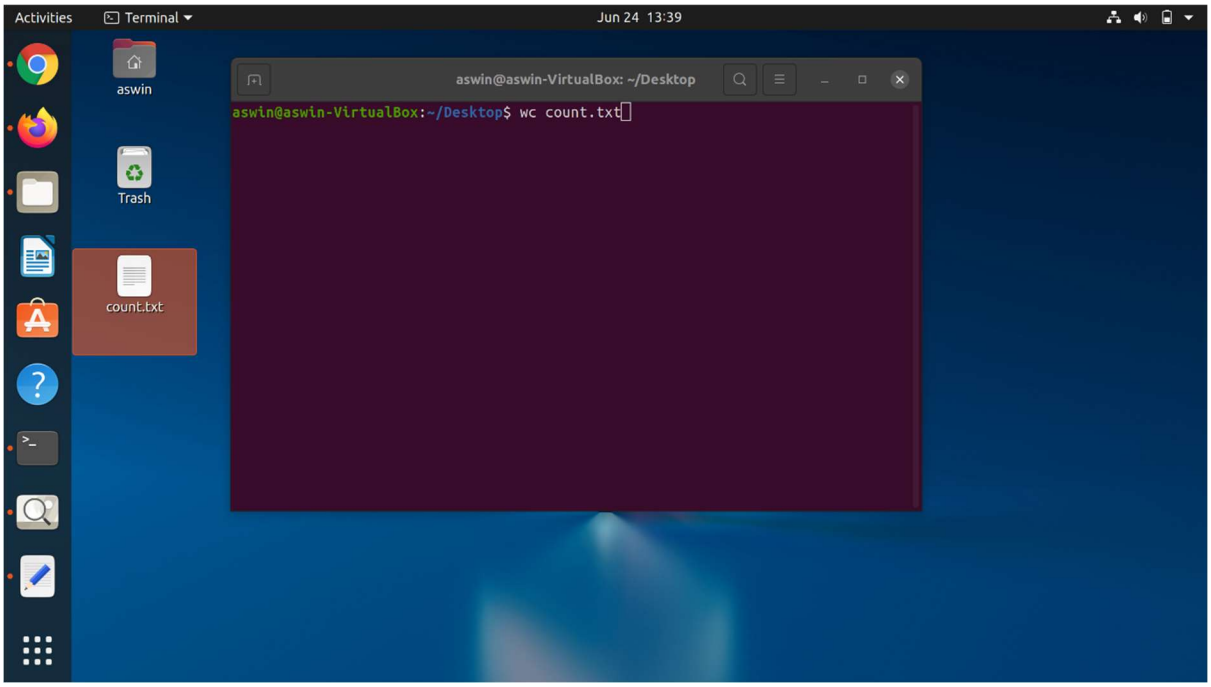


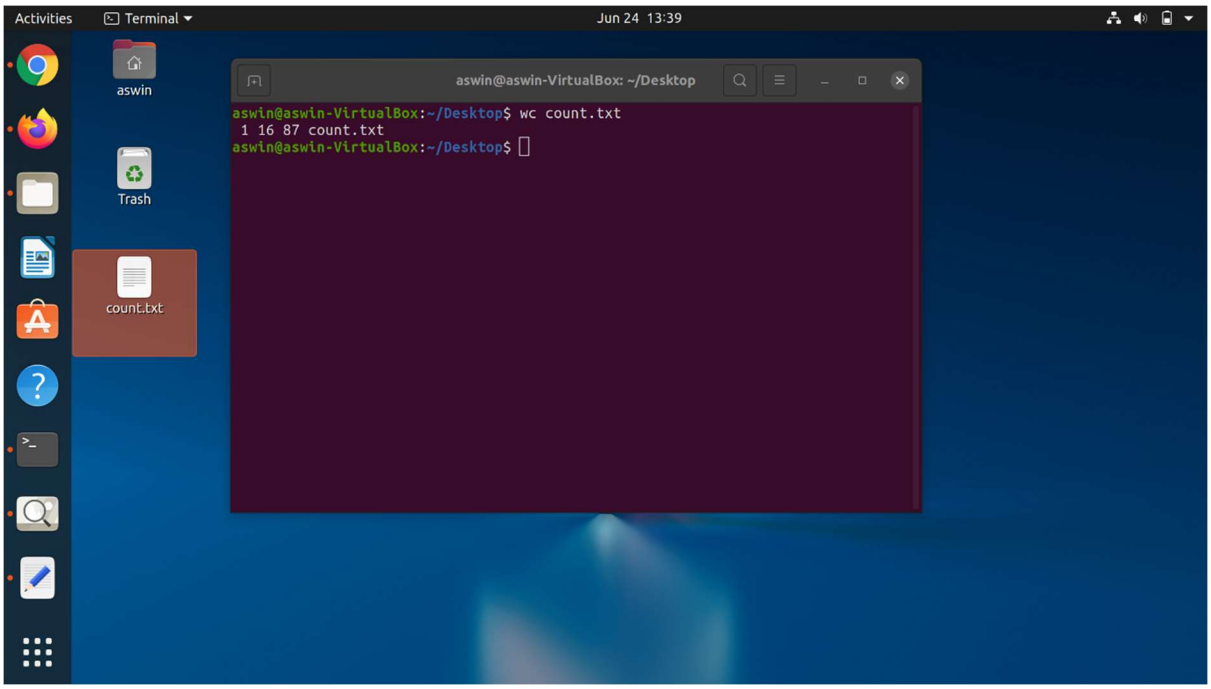
18. wc

wc stands for word count. As the name implies, it is mainly used for counting purpose.

Syntax:

```
wc [OPTION]... [FILE]...
```



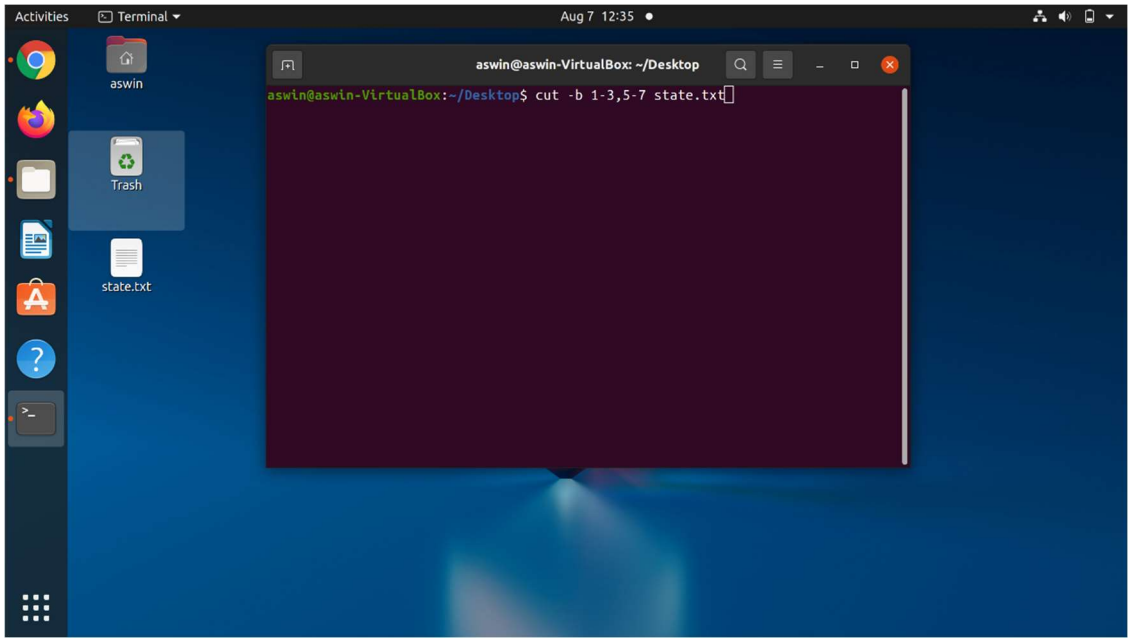


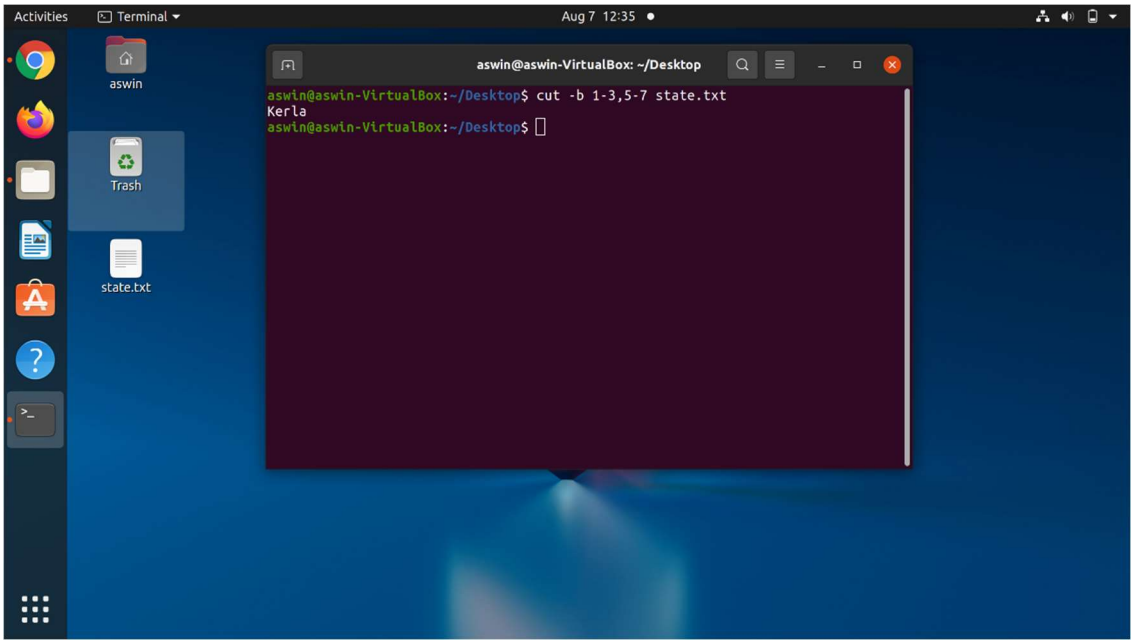
19. cut

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. 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]...
```



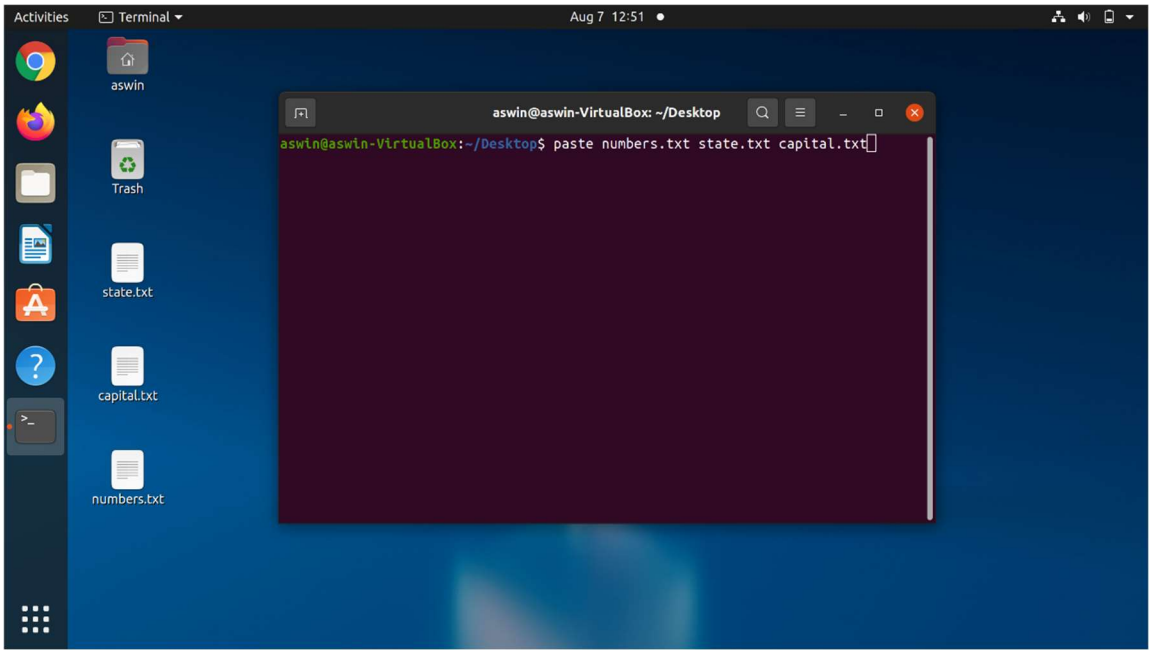


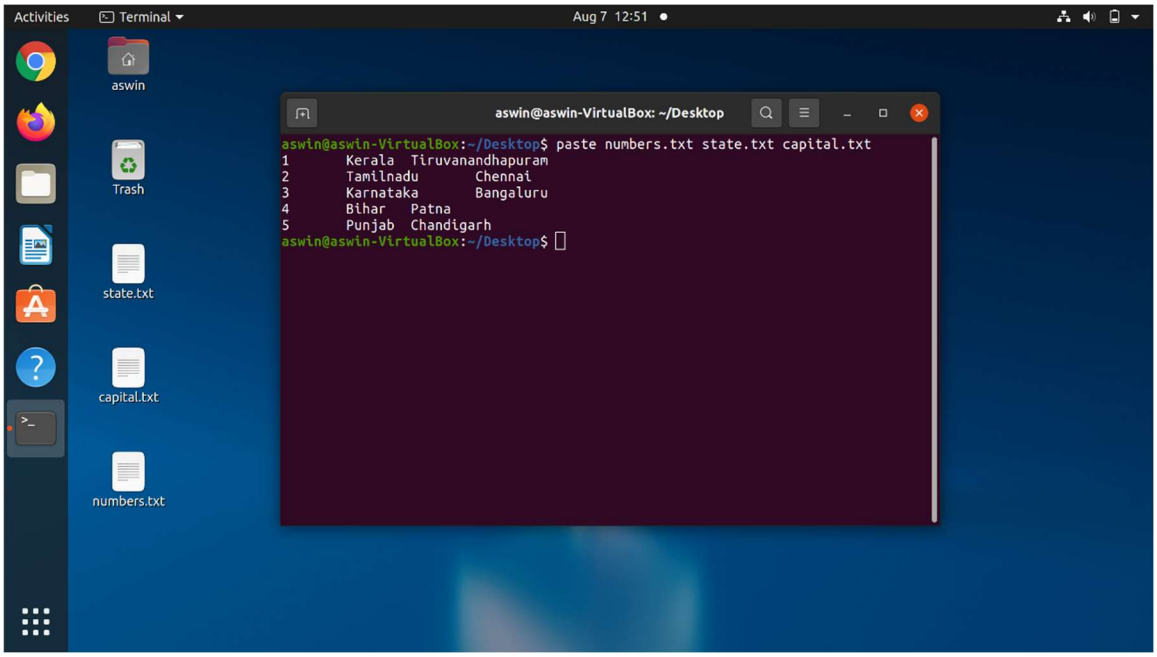
20. paste

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. 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]...
```



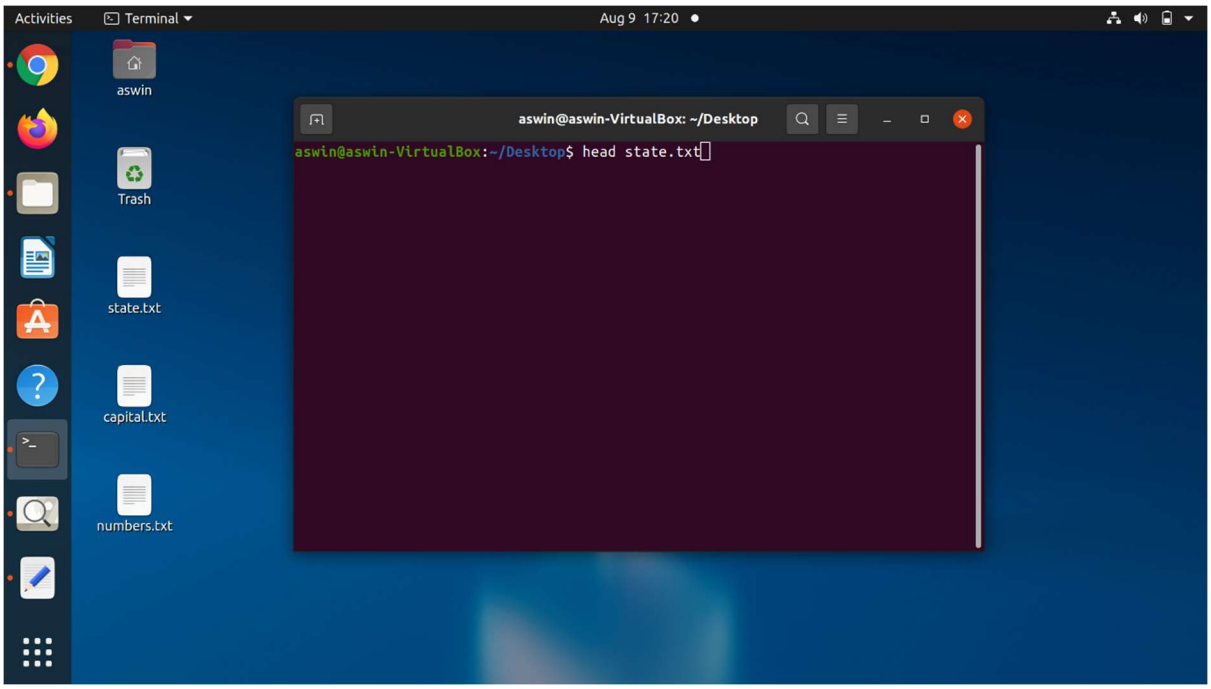


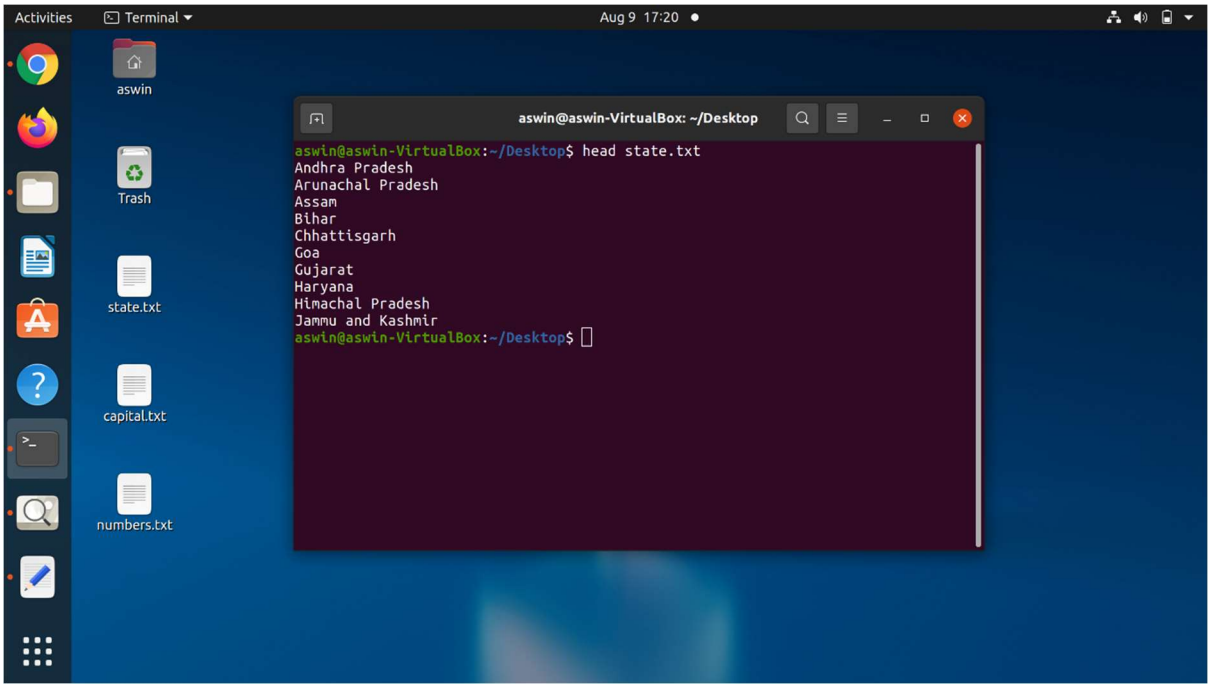
21. head

It is the complementary of Tail command. The head command, as the name implies, print the top N amount 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]... [FILES]...
```



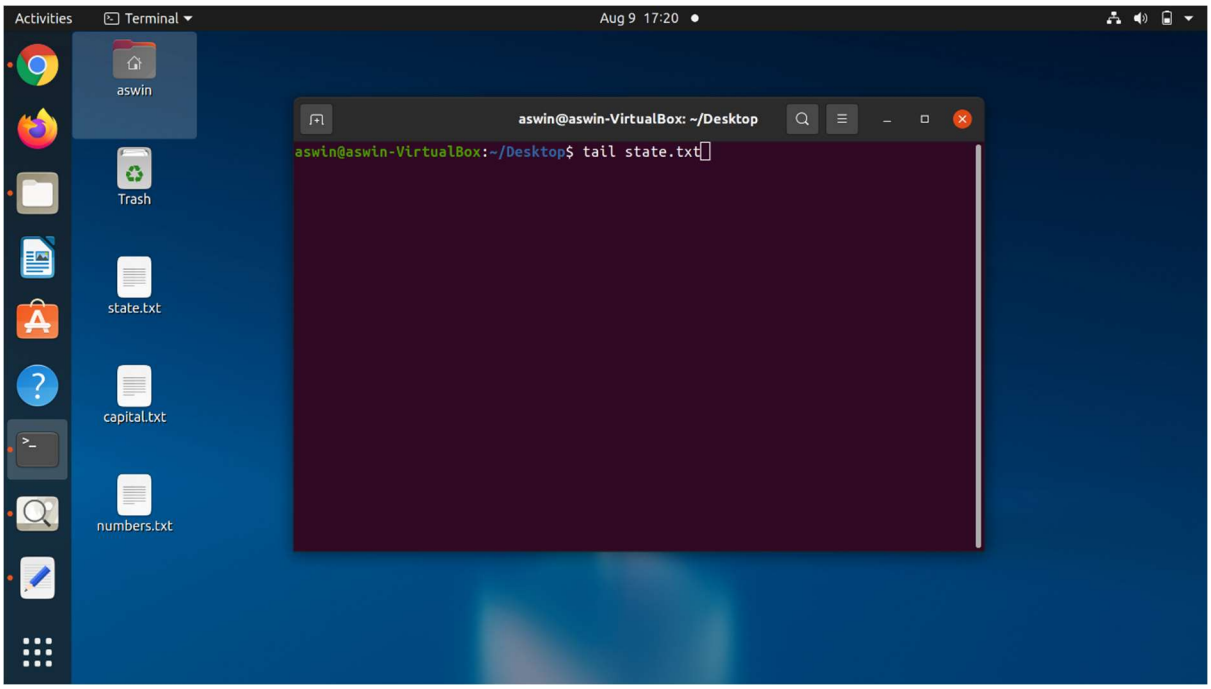


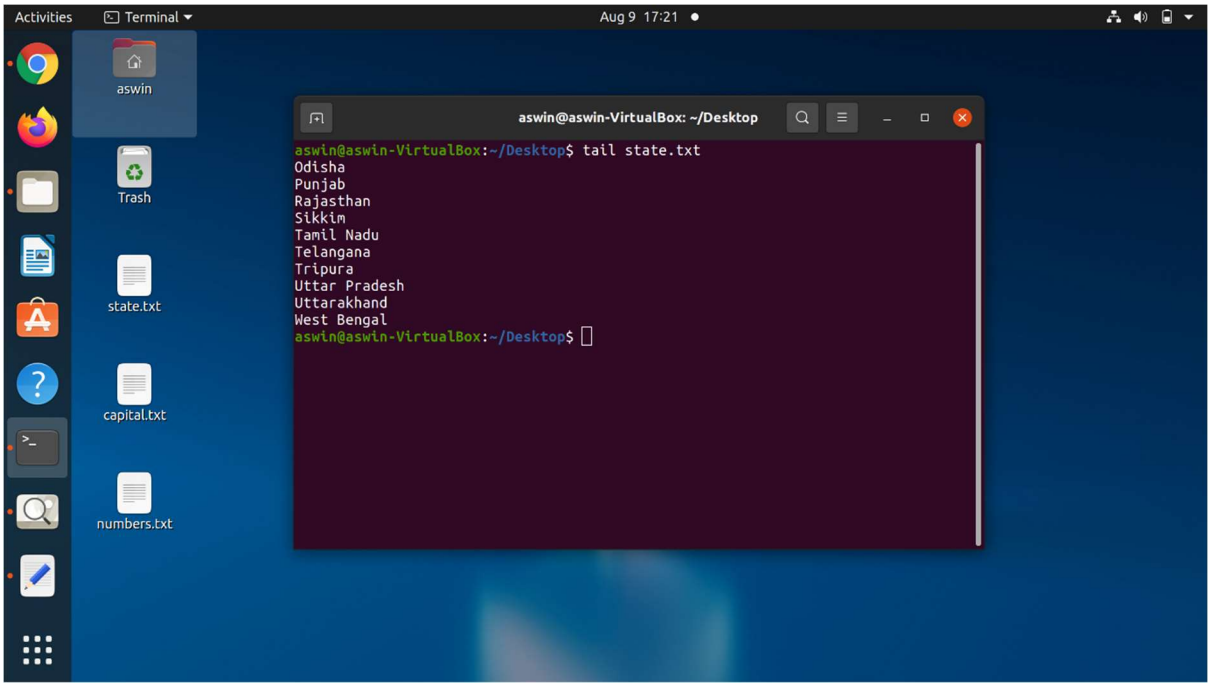
22. tail

It is the complementary of head command. The tail command, as the name implies, print the last N amount 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]... [FILES]...
```



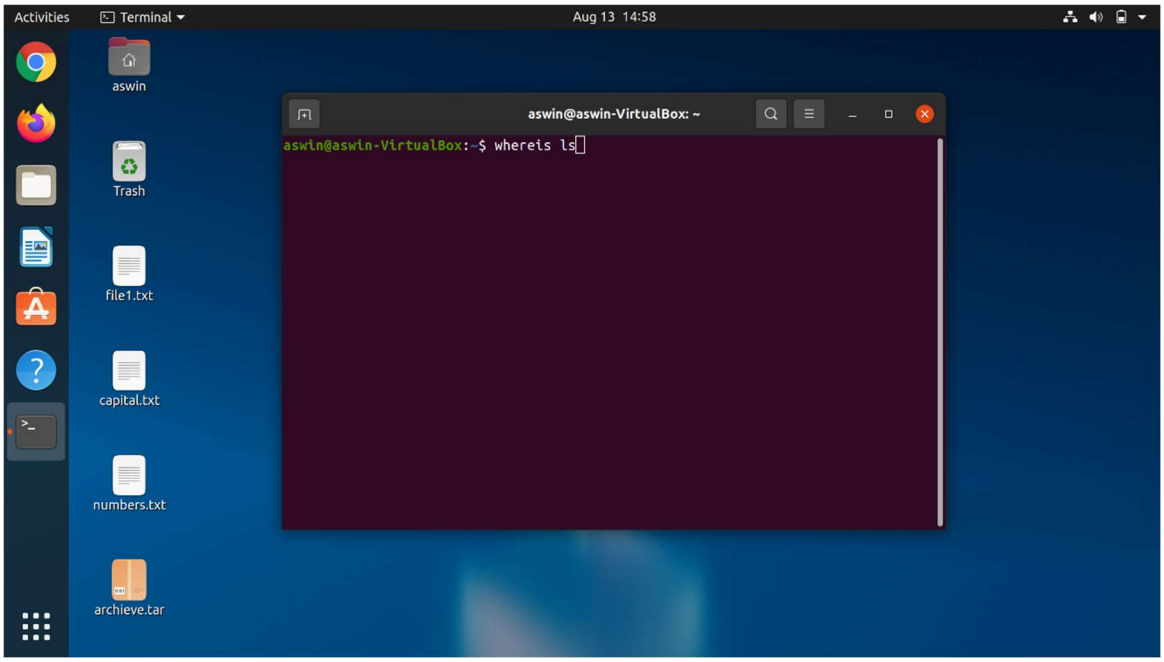


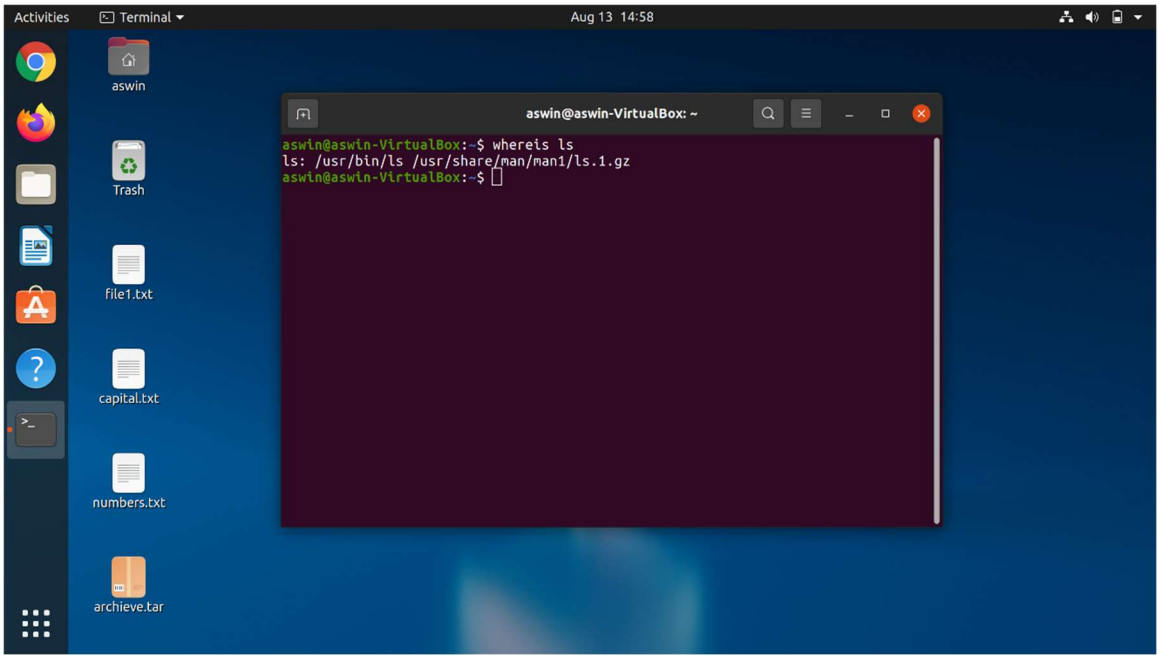
23. whereis

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. If we compare *whereis* command with *find* command they will appear similar to each other as both can be used for the same purposes but *whereis* command produces the result more accurately by consuming less time comparatively. *whereis* doesn't require any root privilege to execute in any *RHEL/CentOS 7*.

Syntax:

```
whereis [options] filename...
```



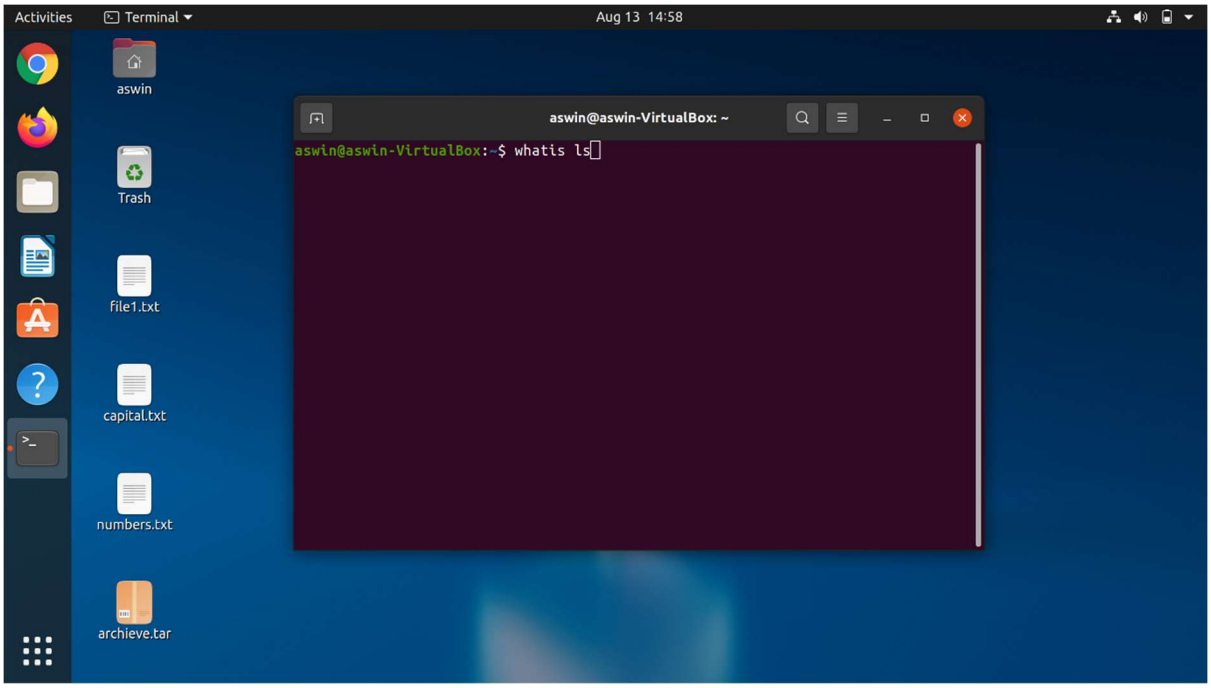


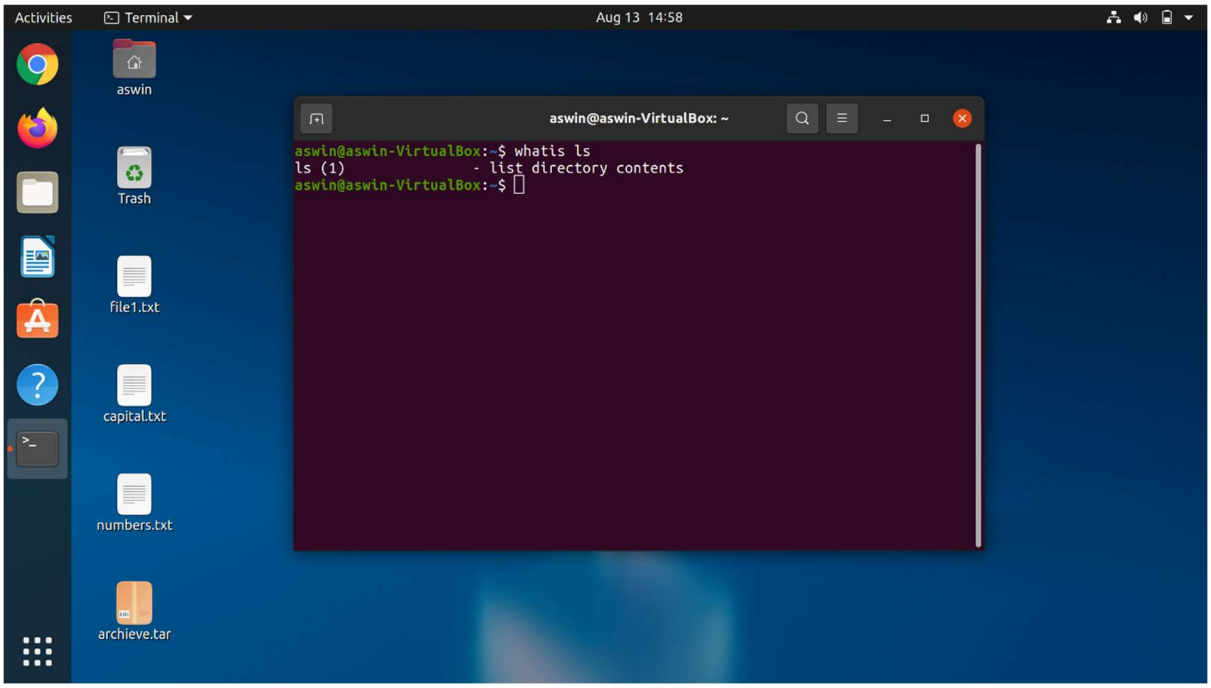
24. whatis

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 …
```



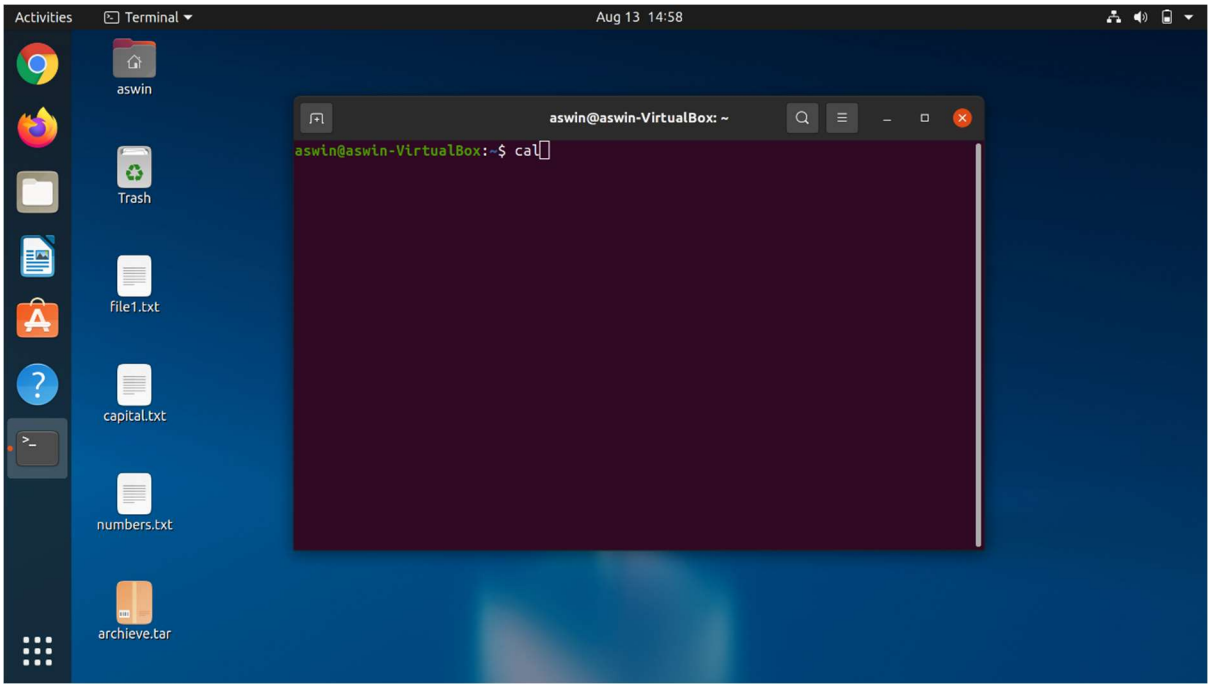


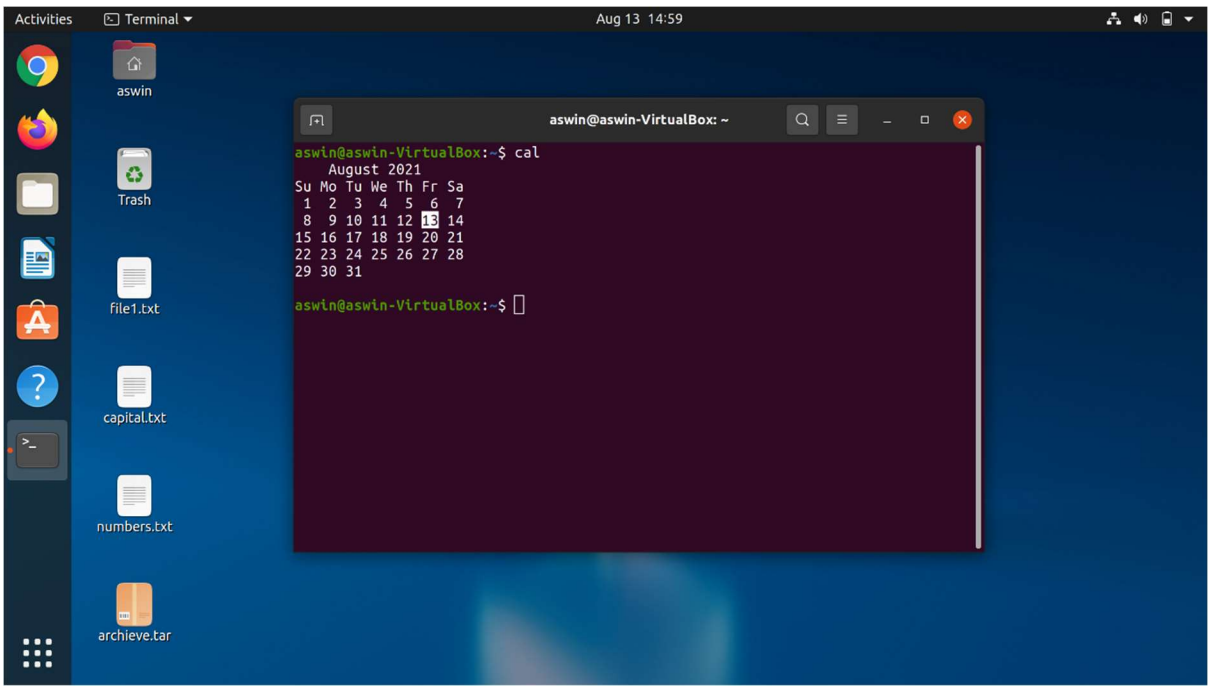
25. cal

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]
```



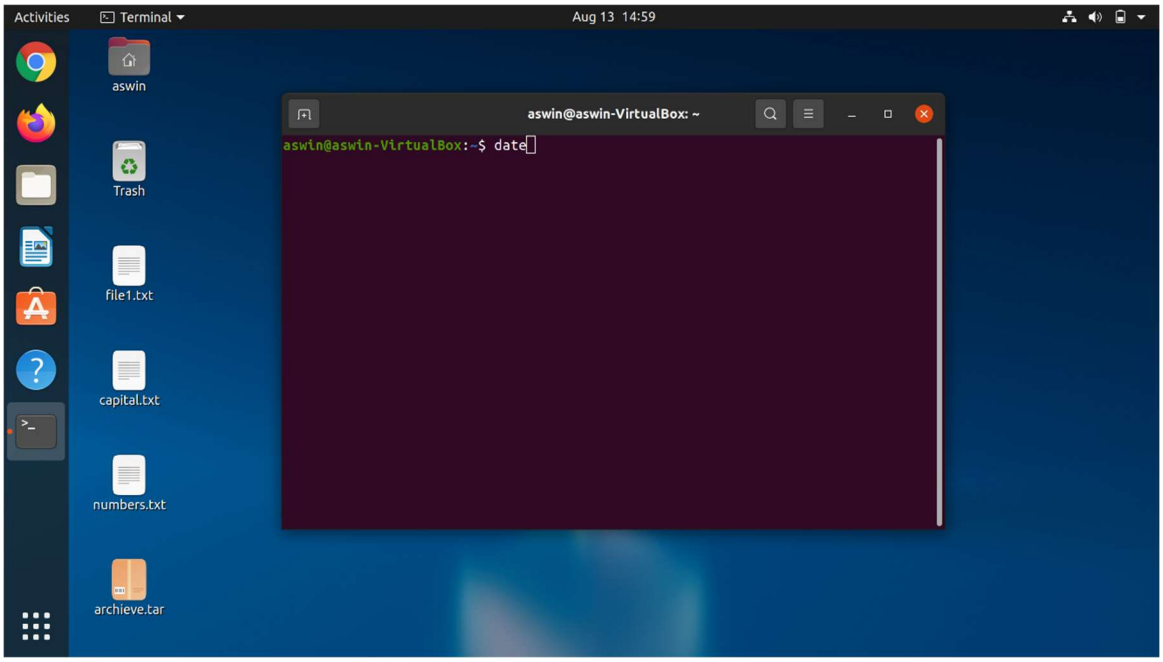


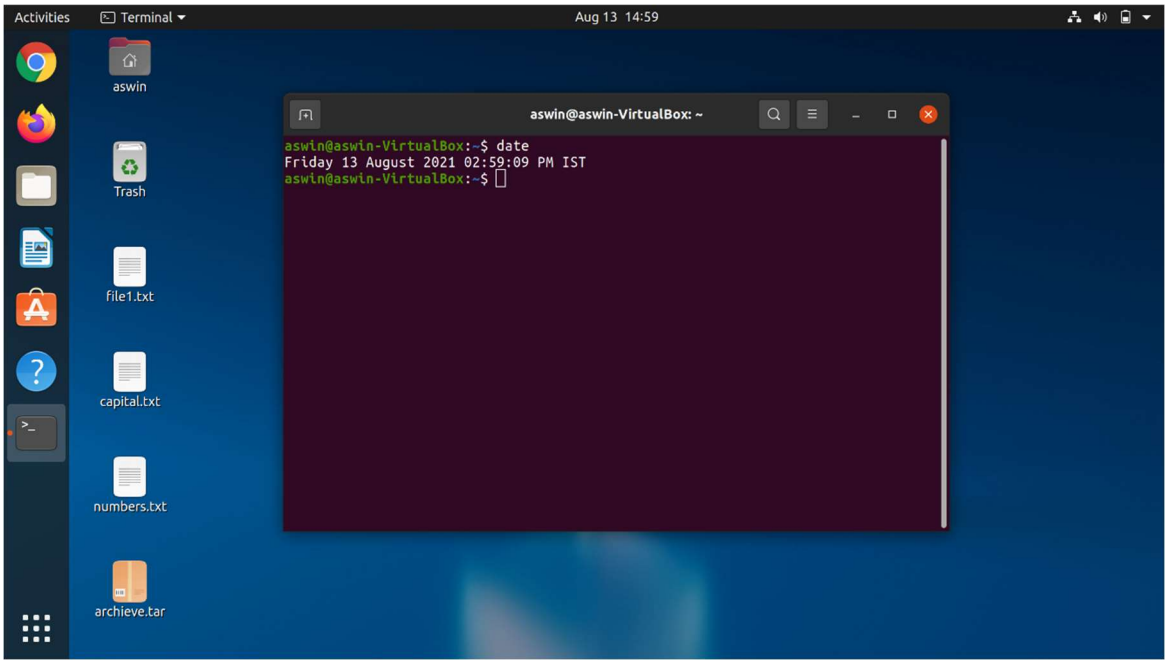
26. date

date command is used to display the system date and time. date command is also used to set date and time of the system. By default the date command displays the date in the time zone on which unix/linux operating system is configured.You must be the super-user (root) to change the date and time.

Syntax:

```
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```



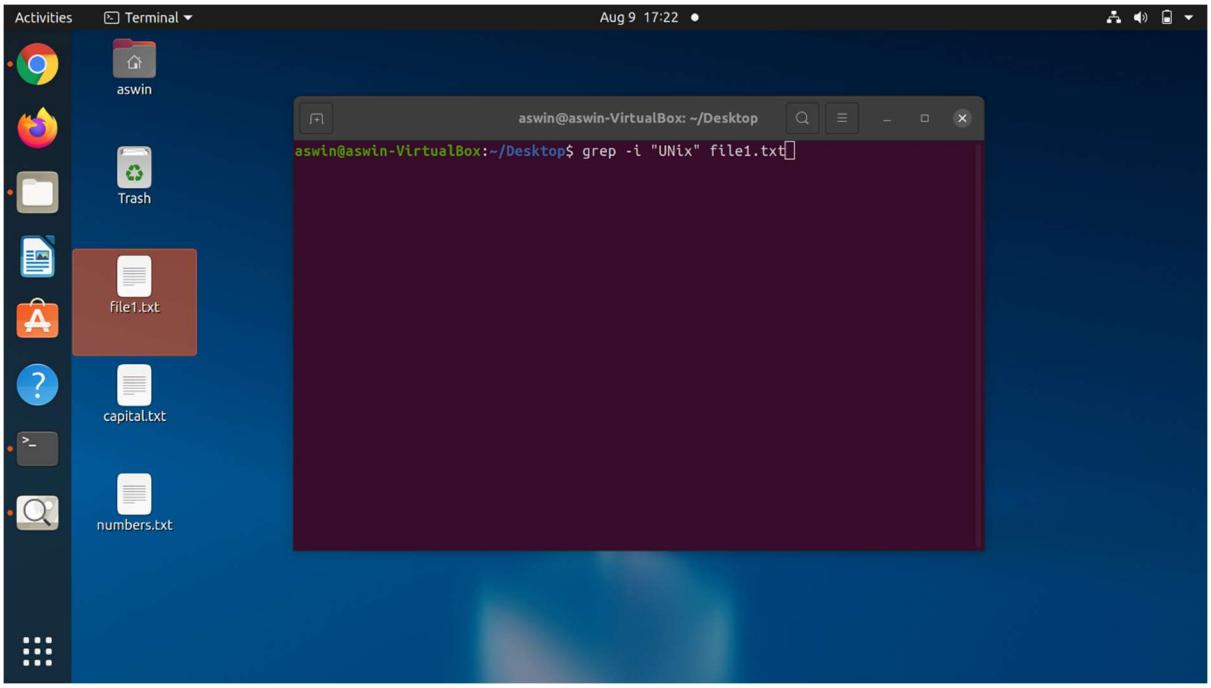


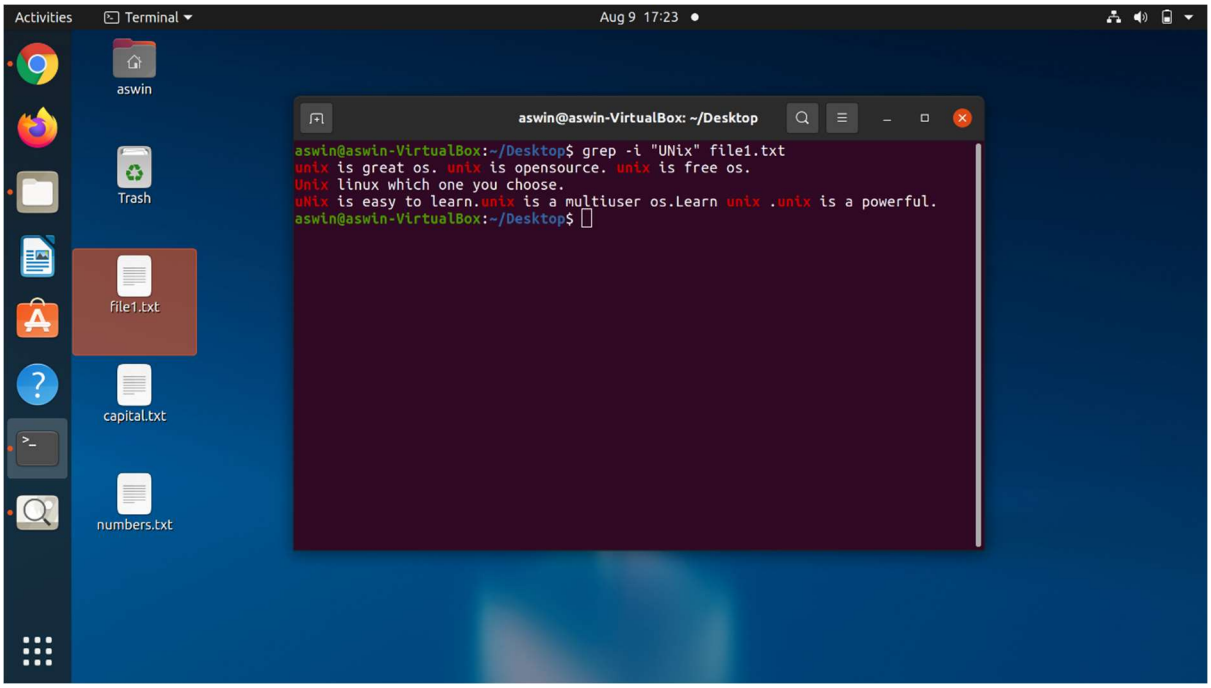
27. grep

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 (grep stands for globally search for regular expression and print out).

Syntax:

```
grep [options] pattern [files]
```





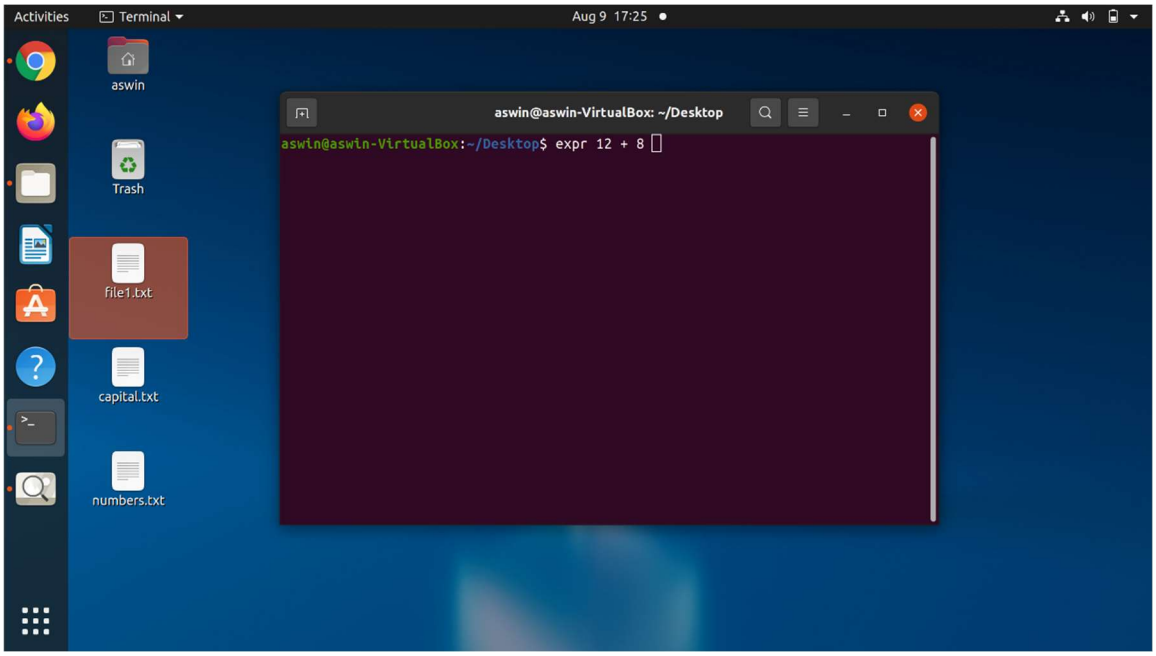
28. expr

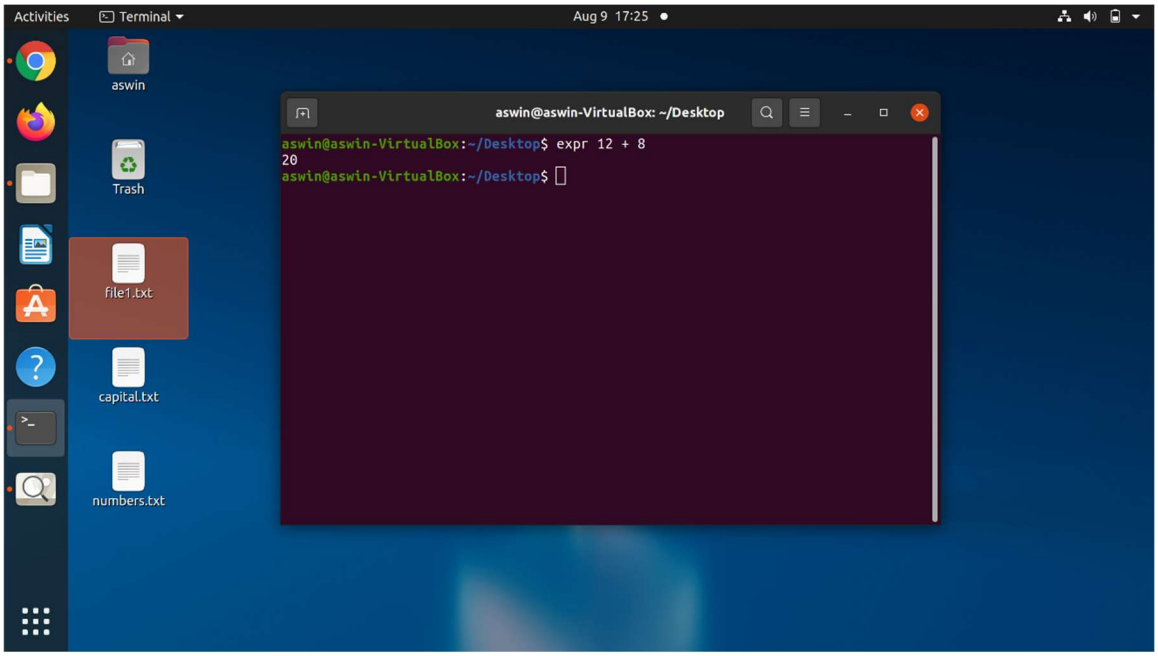
The **expr** command in Unix evaluates a given expression and displays its corresponding output. It is used for:

- 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. The references are represented by one or more of the following letters:

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	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…
```