man and help command

- ***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]...

man and help command

***1. No Option**: It displays the whole manual of the command.*

$ man [COMMAND NAME]

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

$ man [SECTION-NUM] [COMMAND NAME]

***3. -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.*

$ man -f [COMMAND NAME]

***4. -w option**: This option returns the location in which the manual page of a given command is present.*

$ man -w [COMMAND NAME]

man and help command

- ***help****: help command which as its name says help you to learn about any built-in command. help command displays information about shell built-in commands. Here's the syntax for it:*

$help [-dms] [pattern ...]

- The pattern specified in the syntax above refers to the command about which you would like to know and if it is matched with any shell built-in command then **help** give details about it and if it is not matched then **help** prints the list of help topics for your convenience. And the d, m and s here are options that you can use with the **help** command.
- ***-d option :*** *It is used when you just want to get an overview about any shell built-in command i.e it only gives short description.*
- ***-m option :*** *It displays usage in pseudo-manpage format.*
- ***-s option*** *: It just displays only a short usage synopsis for each topic matching.*

printf command

- *"printf" command in Linux is used to create formatted output. It displays the given string, number or any other format specifier on the terminal window. It works the same way as "printf" works in programming languages like C.*
- ***printf** can have format specifiers, escape sequences or ordinary characters.*

Syntax:

$printf [-v var] format [arguments]

Example

$printf " %s\n" "Hello, World!"

$printf "%d \n %s \n %f" 100 ghghdj 100

Passwd, who, uname command

- ***passwd*** *command in Linux is used to change the user account passwords.*

$ passwd

- *The Linux* ***"who"*** *command lets you display the users currently logged in to your UNIX or Linux operating system.*

$ who

- ***uname*** *is a command-line utility that prints basic information about the operating system name*

$uname

uname command

The options are as follows:

- `-s`, (`--kernel-name`) - Prints the kernel name.
- `-n`, (`--nodename`) - Prints the system's node name (hostname). This is the name the system uses when communicating over the network. When used with the `-n` option, `uname` produces the same output as the `hostname` command.
- `-r`, (`--kernel-release`) - Prints the kernel release.
- `-v`, (`--kernel-version`) - Prints the kernel version.
- `-m`, (`--machine`) - Prints the name of the machine's hardware name.
- `-p`, (`--processor`) - Prints the architecture of the processor.
- `-i`, (`--hardware-platform`) - Prints the hardware platform.
- `-o`, (`--operating-system`) - Print the name of the operating system. On Linux systems that is "GNU/Linux"
- `-a`, (`--all`) - When the `-a` option is used, `uname` behaves the same as if the `-snrvmo` options have been given.

tty command

Linux operating system represents everything in a file system, the hardware devices that we attach are also represented as a file. The terminal is also represented as a file. There a command exists called ***tty*** *which displays information related to* ***terminal****.* ***The tty command of terminal basically prints the file name of the terminal connected to standard input.***

$ tty

Tty, stty command

Linux operating system represents everything in a file system, the hardware devices that we attach are also represented as a file. The terminal is also represented as a file. There a command exists called ***tty*** *which displays information related to* ***terminal****.* ***The tty command of terminal basically prints the file name of the terminal connected to standard input.***

$ tty

stty *command in Linux is used to change and print terminal line settings. Basically, this command shows or changes terminal characteristics.*

$stty

comm command

- ***comm*** *compare two sorted files line by line and write to standard output; the lines that are* ***common*** *and the lines that are* ***unique****.*

- *Suppose you have two lists of people and you are asked to find out the names available in one and not in the other, or even those common to both. comm is the command that will help you to achieve this. It requires two sorted files which it compares line by line.*

$comm [OPTION]... FILE1 FILE2

cmp command

- ***cmp*** *command in Linux/UNIX is used to compare the two files byte by byte and helps you to find out whether the two files are identical or not.*
- *When* ***cmp*** *is used for comparison between two files, it reports the location of the first mismatch to the screen if difference is found and if no difference is found i.e the files compared are identical. cmp displays no message and simply returns the prompt if the the files compared are identical.*

$cmp file1.txt file2.txt

diff command

- ***diff*** *stands for difference. This command is used to* ***display the differences in the files by comparing the files line by line****. Unlike its fellow members, cmp and comm, it tells us which lines in one file have is to be changed to make the two files identical.*

- *The important thing to remember is that diff uses certain special symbols and instructions that are required to make two files identical. It tells you the instructions on how to change the first file to make it match the second file.*

$diff file1.txt file2.txt

zip command

- ***zip*** *is a compression and file packaging utility for Linux.*

- *Syntax :*

zip [options] zipfile files_list

$zip –r myfile.zip filename.txt

- ***unzip*** *is a compression and file packaging utility for Linux*

$unzip myfile.zip

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.*
- *Creating an uncompressed tar Archive using option -cvf :*

$ tar cvf file.tar *.txt

- *Extracting files from Archive using option -xvf : This command extracts files from Archives.*

$ tar xvf file.tar

File permissions and changing the access rights.

Linux, running on shared computers use settings called permissions to determine who can access and modify the files and directories stored in their file systems. Every file in Unix has the following attributes −

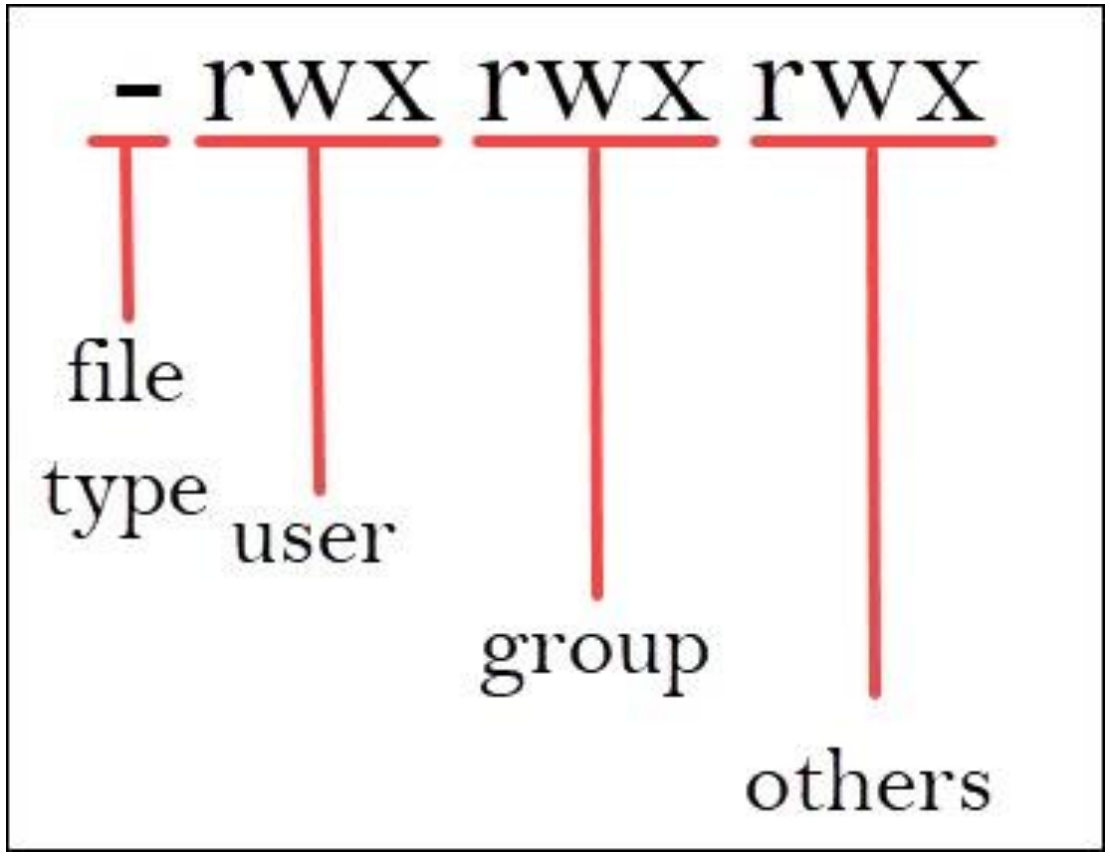
- ***Owner permissions*** *− The owner's permissions determine what actions the owner of the file can perform on the file.*
- ***Group permissions*** *− The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.*
- ***Other (world) permissions*** *− The permissions for others indicate what action all other users can perform on the file.*

-rw-r--r-- 1 user1 group1 62 Jan 15 16:10 myfile.txt

drwxr-xr-x 2 user1 group1 2048 Jan 15 17:10 Example

In the output example above, the first character in each line indicates whether the listed object is a file or a directory. Directories are indicated by a (d); the absence of a d at the beginning of the first line indicates that myfile.txt is a regular file.

File permissions and changing the access rights.



File permissions and changing the access rights.

The letters rwx represent different permission levels:

Permission	Files	Directories
`r`	can read the file	can `ls` the directory
`w`	can write the file	can modify the directory's contents
`x`	can execute the file	can `cd` to the directory

- ***Owner or user permissions:*** *After the directory (d) slot, the first set of three characters indicate permission settings for the owner (also known as the user).*
- ***Group permissions:*** *The second rwx set indicates the group permissions. In the fourth column of the example above, group1 is the group name.*
- ***Other permissions****: The final rwx set is for "other" (sometimes referred to as "world"). This is anyone outside the group.*

Change file permissions

To change file and directory permissions, use the command ***chmod (change mode).*** *The owner of a file can change the permissions for user (u), group (g), or others (o) by adding (+) or subtracting (-) the read, write, and execute permissions. There are two basic ways of using* ***chmod*** *to change file permissions:*

1. ***The symbolic (relative) method***
2. ***The absolute method***

Symbolic method:

The first and probably easiest way is the relative (or symbolic) method, which lets you specify permissions with single letter abbreviations. A ***chmod*** *command using this method consists of at least three parts from the following lists:* ***chmod o-r filea***

Access class	Operator	Access Type
u (user)	+ (add access)	r (read)
g (group)	- (remove access)	w (write)
o (other)	= (set exact access)	x (execute)
a (all: u, g, and o)		

Change file permissions

Absolute form:

The other way to use the **chmod** command is the absolute form, in which you specify a set of three numbers that together determine all the access classes and types. Rather than being able to change only particular attributes, you must specify the entire state of the file's permissions. The three numbers are specified in the order: user (or owner), group, and other. Each number is the sum of values that specify read, write, and execute access:

Permission	Number
Read (r)	4
Write (w)	2
Execute (x)	1

Change file permissions

- For file myfile, to grant read, write, and execute permissions to yourself (4+2+1=7), read and execute permissions to users in your group (4+0+1=5), and only execute permission to others (0+0+1=1), you would use:

chmod 751 myfile

- To grant read, write, and execute permissions on the current directory to yourself only, you would use:

chmod 700

Change file permissions

Some other examples are:

777	*anyone can do anything (read, write, or execute)*
755	*you can do anything; others can only read and execute*
711	*you can do anything; others can only execute*
644	*you can read and write; others can only read*

Linux, running on shared high-performance computers use settings called permissions to determine who can access and modify the files and directories stored in their file systems. Each file and directory in a file system is assigned "owner" and "group" attributes.

Chown file permissions

Use the following procedure to change the ownership of a file.

1. *Become superuser or assume an equivalent role.*

2. *Change the owner of a file by using the chown command.*

 # chown new-owner filename

new-owner	Specifies the user name or UID of the new owner of the file or directory.
filename	Specifies the file or directory.

find Command

- *The Linux **find** command is very powerful. It can search the entire filesystem to find files and directories according to the search criteria you specify. Besides using the find command to locate files, you can also use it to execute other Linux commands (grep, mv, rm, etc.) on the files and directories you find, which makes find extremely powerful.*

find ~ -name readme.txt

find ~ -name *.txt

- *Locating Files by Size Another possible search is to search for files by size.*

find ~ -name *.txt -size +100M

find ~ -name *.txt -size -100M

- *Locating Files by Access Time*

find $HOME -atime +30

find $HOME -iname '*.ogg' -atime +30

expr – Computational and String handling

*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.*

You have to provide the space between the values and the operands. Otherwise the expr command may throw error or print them as a string.

$expr expression

1. **Using expr for basic arithmetic operations :**

$expr 12 + 8

2. **For String operations**

expr length hello

3. **Matching number of characters in two strings**

$ expr hello : hell

expr – Computational and String handling

3. Finding substring of a string

$ expr substr hello 2 3

Arithmetic Operator Examples:

Example:

1. Sum of numbers

$ expr 5 + 3

2. Difference between two numbers

$ expr 10 - 6

3. Multiplying numbers

*$ expr 7 * 9*

*Here the * is shell builtin operator, that is why it needs to escaped with backslash.*

4. Dividing numbers

$ expr 6 / 4

The division operator returns only the arithmetic quotient.

5. Remainder or modulus

$ expr 6 % 4

expr – Computational and String handling

Arithmetic Operator Examples:

Example:

1. Sum of numbers

```
$ expr 5 + 3
```

2. Difference between two numbers

```
$ expr 10 - 6
```

3. Multiplying numbers

```
$ expr 7 \* 9
```

*Here the * is shell builtin operator, that is why it needs to escaped with backslash.*

4. Dividing numbers

```
$ expr 6 / 4
```

The division operator returns only the arithmetic quotient.

5. Remainder or modulus

```
$ expr 6 % 4
```

expr – Computational and String handling

Comparison or Relational Operator Examples:

You can use the following comparison operators with the expr command:

- ***Val1 < Val2 :*** *Returns 1 if val1 is less than val2. otherwise zero.*
- ***Val1 <= Val2 :*** *Returns 1 if val1 is less than or equal to val2. otherwise zero.*
- ***Val1 > Val2 :*** *Returns 1 if val1 is greater than val2. otherwise zero.*
- ***Val1 >= Val2 :*** *Returns 1 if val1 is greater than or equal to val2. otherwise zero.*
- ***Val1 = Val2 :*** *Returns 1 if val1 is equal to val2. otherwise zero.*
- ***Val1 != Val2 :*** *Returns 1 if val1 is equal to val2. otherwise zero.*
- ***val1 | val2 :*** *Returns val1 if val1 is neither null nor zero. Otherwise val2.*
- ***val1 & val2 :*** *Returns val1 if both val1 and val2 is neither null nor zero. Otherwise 0.*

Note: You have to escape most of the operators with backslash as they are shell built in.

expr – Computational and String handling

```
$ expr 1 \< 2
1
$ expr 1 \<= 1
1
$ expr 2 \> 5
0
$ expr 2 \>= 5
0
$ expr 7 = 7
1
$ expr 9 != 18
1
$ expr 2 \| 5
2
$ expr 0 \| 5
5
$ expr 2 \& 5
2
$ expr 6 \& 3
6
$ expr 6 \& 0
0
$ expr 0 \& 3
0
```

expr – Computational and String handling

1. ***Length of string:*** *The length function is used to find the number of characters in a string.*

$ expr length linux

5

$expr length linux\ system

12

$expr length "linux system"

If you have spaces in your string escape them with backslash or quote them with double quotes.

2. ***Find Substring:*** *You can extract a portion of the string by using the substr function. The syntax of substr function is*

substr string position length

Here position is the character position in the string. length is the number of chracters to extract from the main string.

$ expr substr unixserver 5 6

server

expr – Computational and String handling

3. ***Index of the substring:*** *You can find the position of a string in the main string using the index function. The syntax of index function is shown below:*

index string chars

If the chars string is found in the main string, then the index function returns the position of the chars. Otherwise it returns 0. See the following examples:

$ expr index linux nux

3

$expr index linux win

0

expr – Computational and String handling

4. ***Matching a regexp:*** *The match function is used to find anchored pattern match of regexp in the string. The syntax of match function is shown below:*

match string pattern

The match function returns the number of characters in the pattern is a match is found. Otherwise, it returns 0

```
$ expr match linuxserver lin
3
$ expr match linuxserver server
0
```