Mastering Linux Manual Pages

Introduction

I hope you have installed Linux distribution to your system. Now, after you have installed the required distribution, then the first thing that you should play around is the terminal. The shortcut to access terminal is Ctrl+Alt+T.
It will be highly ambitious to comprehend and know all the commands used in the terminal in a single document. Instead of trying to accomplish this insurmountable task, we can comprehensively understand the built-in support system in the Linux terminal: **the Linux Man Pages**, which is short for Linux manual pages.

Linux Man Pages

To access man page of any command, we should write the following command: man name of the command

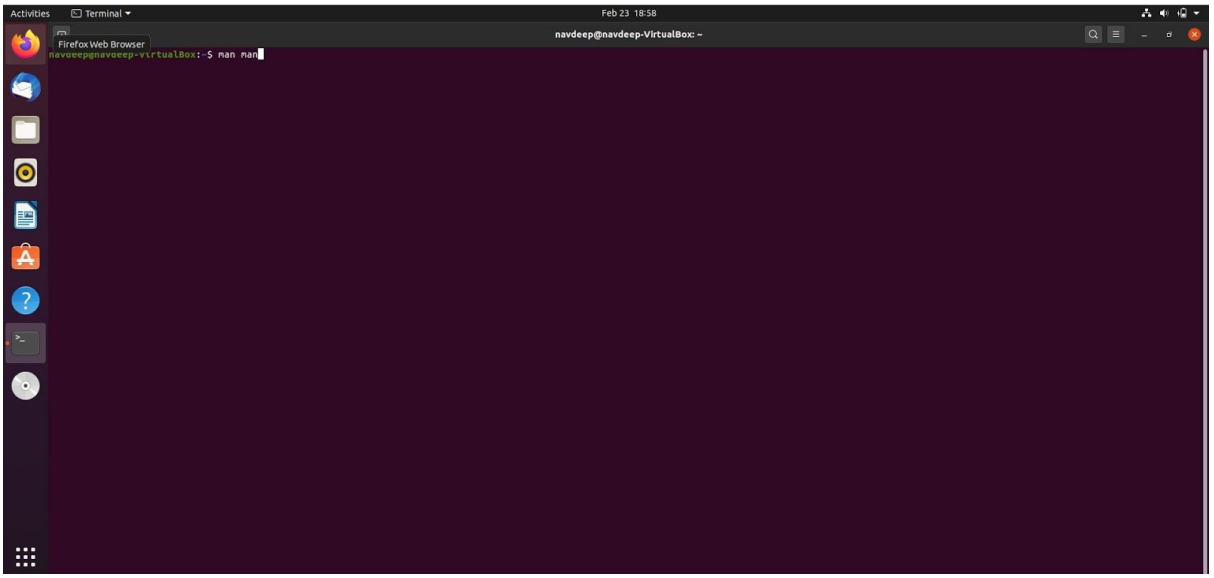
Examples:

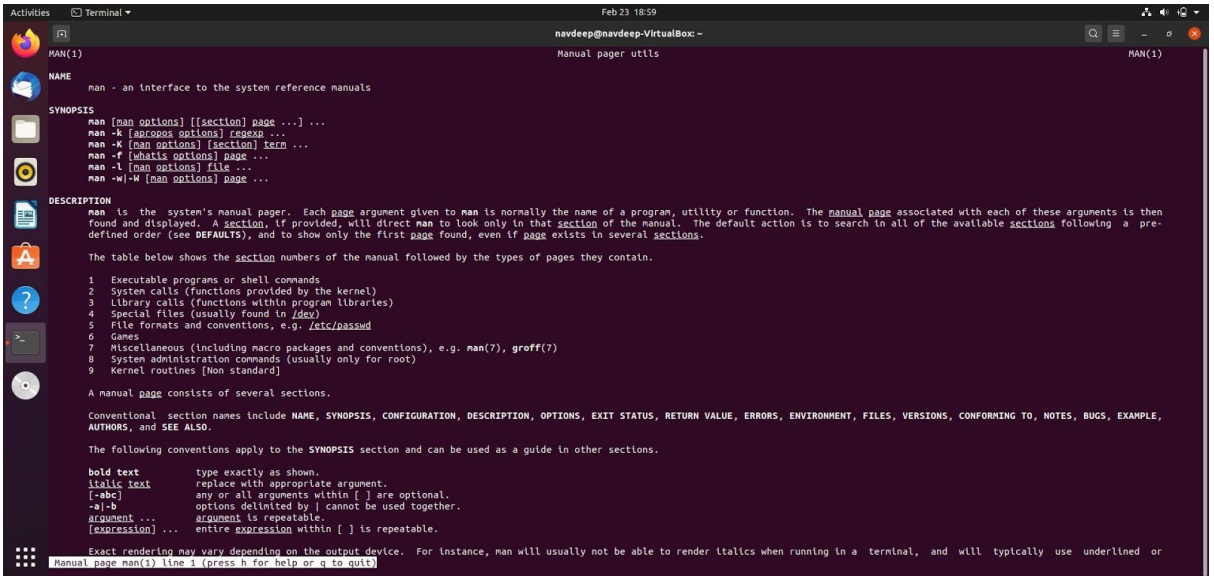
1. man ls
2. man man
3. man cat

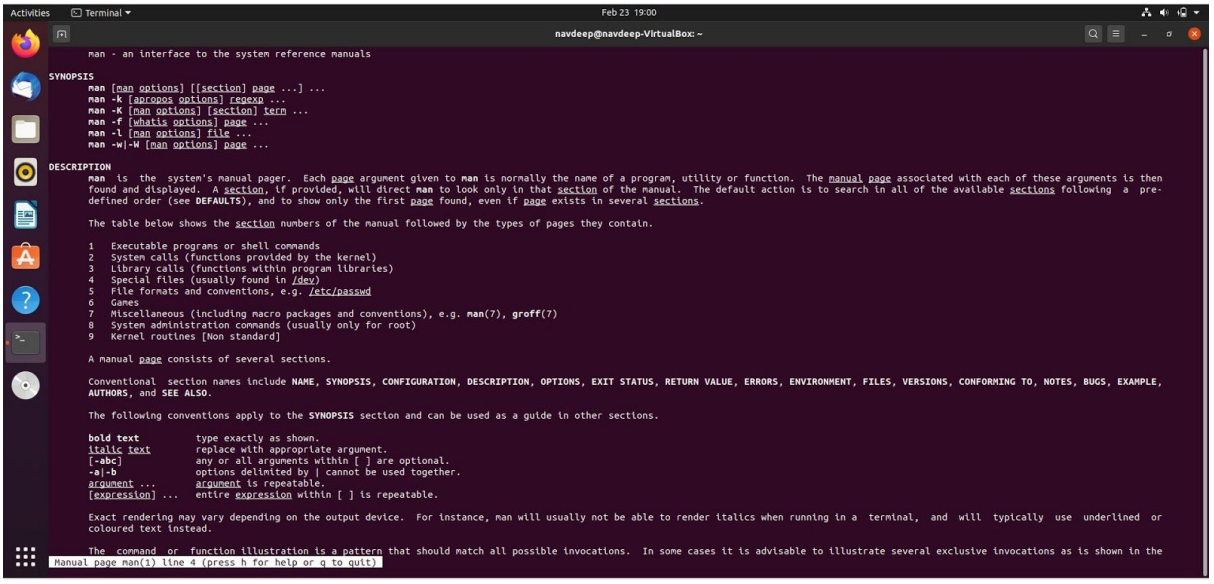
To explore more, let's start digging deep into **man** itself. Type the following command into terminal:

```
>>> man man
```

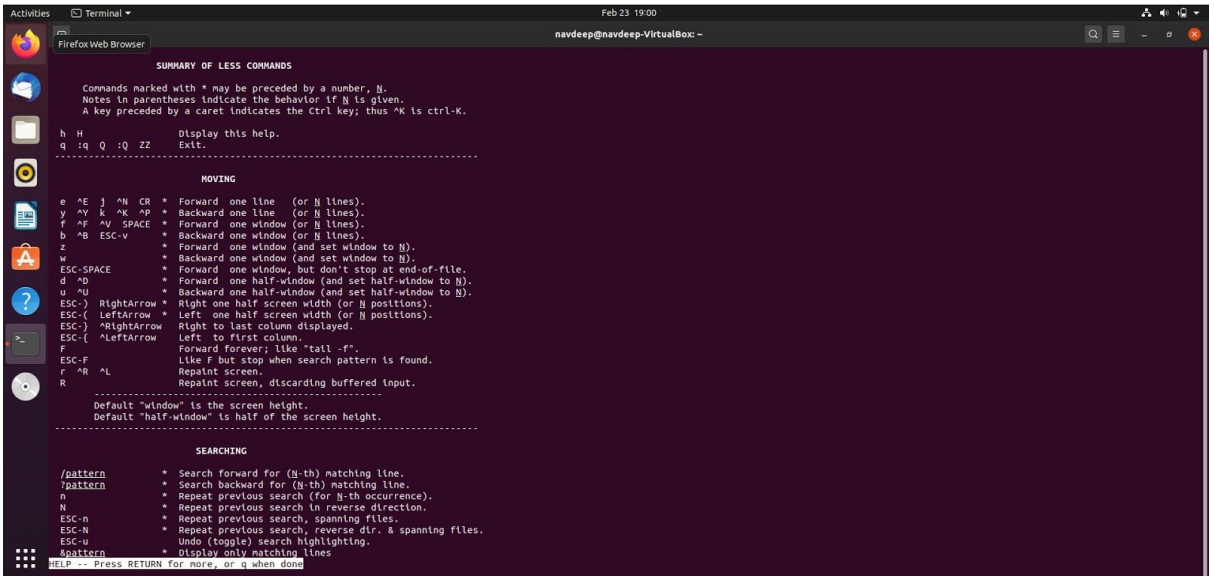
- The following command will open up the man page for the **man** command.


Activities
Terminal
Feb 23 18:58
Firefox Web Browser
navdeep@navdeep-VirtualBox: ~
navdeep@navdeep-VirtualBox:~$ man man


Activities
Terminal
Feb 23 18:59
navdeep@navdeep-VirtualBox: ~
MAN(1) Manual pager utils MAN(1)
NAME
man - an interface to the system reference manuals
SYNOPSIS
man [man options] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [man options] [section] term ...
man -f [whatis options] page ...
man -l [man options] file ...
man -w|-W [man options] page ...
DESCRIPTION
man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sections.
The table below shows the section numbers of the manual followed by the types of pages they contain.
1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions, e.g. /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]
A manual page consists of several sections.
Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES, VERSIONS, CONFORMING TO, NOTES, BUGS, EXAMPLE, AUTHORS, and SEE ALSO.
The following conventions apply to the SYNOPSIS section and can be used as a guide in other sections.
bold text type exactly as shown.
italic text replace with appropriate argument.
[-abc] any or all arguments within [ ] are optional.
-a|-b options delimited by | cannot be used together.
argument ... argument is repeatable.
[expression] ... entire expression within [ ] is repeatable.
Exact rendering may vary depending on the output device. For instance, man will usually not be able to render italics when running in a terminal, and will typically use underlined or
Manual page man(1) line 1 (press h for help or q to quit)

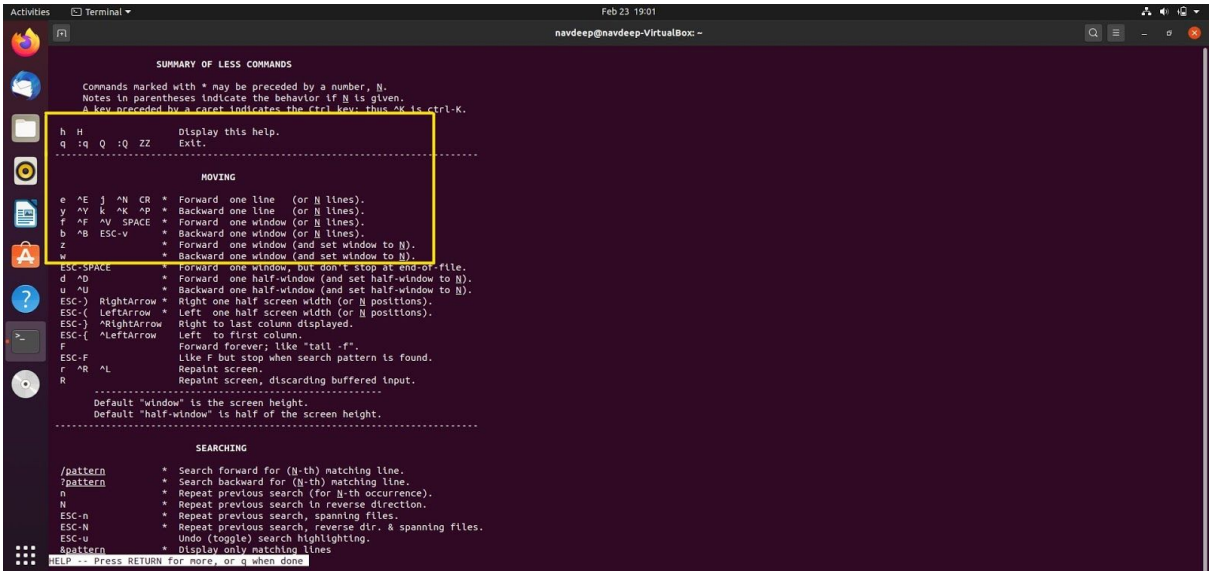


- The only command that needs to be remembered is h. h stands for help. You can type h for help and support and you can type q to go back. The following image shows the interface that will come on typing h.



- The following image highlights the information that you need to maneuver on the man page. Following are the commands highlighted in the image:
  - j You will go forward one line
  - k You will go backward one line
  - ^F You will go forward one window
  - ^B You will go backward one window

- g You will go to the first line in file
- G You will go to the last line in file



Searching

The following command will be used to search a particular pattern in the man page.

```
>>> /pattern
```

The following command will highlight all the occurrences of pattern in the man page.

```
navdeep@navdeep-VirtualBox: ~
 ESC-F                Like F but stop when search pattern is found.
 r  ^R  ^L            Repaint screen.
 R                    Repaint screen, discarding buffered input.
       ---------------------------------------------------
       Default "window" is the screen height.
       Default "half-window" is half of the screen height.
---------------------------------------------------------------------------

                          SEARCHING

 /pattern           *  Search forward for (N-th) matching line.
 ?pattern           *  Search backward for (N-th) matching line.
 n                  *  Repeat previous search (for N-th occurrence).
 N                  *  Repeat previous search in reverse direction.
 ESC-n              *  Repeat previous search, spanning files.
 ESC-N              *  Repeat previous search, reverse dir. & spanning files.
 ESC-u                 Undo (toggle) search highlighting.
 &pattern           *  Display only matching lines
       ---------------------------------------------------
       A search pattern may begin with one or more of:
       ^N or !  Search for NON-matching lines.
       ^E or *  Search multiple files (pass thru END OF FILE).
       ^F or @  Start search at FIRST file (for /) or last file (for ?).
       ^K       Highlight matches, but don't move (KEEP position).
       ^R       Don't use REGULAR EXPRESSIONS.
---------------------------------------------------------------------------

                           JUMPING

 g  <  ESC-<        *  Go to first line in file (or line N).
 G  >  ESC->        *  Go to last line in file (or line N).
 p  %               *  Go to beginning of file (or N percent into file).
 t                  *  Go to the (N-th) next tag.
 T                  *  Go to the (N-th) previous tag.
 {  (  [            *  Find close bracket } ) ].
 }  )  ]            *  Find open bracket { ( [.
 ESC-^F <c1> <c2>   *  Find close bracket <c2>.
 ESC-^B <c1> <c2>   *  Find open bracket <c1>
       ---------------------------------------------------
       Each "find close bracket" command goes forward to the close bracket
         matching the (N-th) open bracket in the top line.
       Each "find open bracket" command goes backward to the open bracket
         matching the (N-th) close bracket in the bottom line.

 m<letter>             Mark the current top line with <letter>.
HELP -- Press RETURN for more, or q when done
```

Man Page Conventions

Following are few conventions to be followed in the man page.

bold text	If syntax is in bold, then it needs to be typed exactly as shown.
italic text	This means it needs to be replaced by appropriate argument
[-abc]	Argument with square brackets are optional
-a | -b	Options which are separated by |, cannot be used together
argument ...	This means that argument is repeatable

Man Page Sections

Linux terminal commands are divided into certain sections in the Linux manual. The generic command for opening a man page for a particular command is:

man [SECTION] Command_name

Examples:

1. man unlink

```
UNLINK(1)                                                                User Commands

NAME
       unlink - call the unlink function to remove the specified file

SYNOPSIS
       unlink FILE
       unlink OPTION

DESCRIPTION
       Call the unlink function to remove the specified FILE.

       --help display this help and exit

       --version
              output version information and exit

AUTHOR
       Written by Michael Stone.
```

2. man 2 unlink: This command opens the man page for unlink in second section

```
navdeep@navdeep-VirtualBox: ~
UNLINK(2)                                                     Linux Programmer's Manual

NAME
       unlink, unlinkat - delete a name and possibly the file it refers to

SYNOPSIS
       #include <unistd.h>

       int unlink(const char *pathname);

       #include <fcntl.h>           /* Definition of AT_* constants */
       #include <unistd.h>

       int unlinkat(int dirfd, const char *pathname, int flags);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       unlinkat():
           Since glibc 2.10:
               _POSIX_C_SOURCE >= 200809L
           Before glibc 2.10:
               _ATFILE_SOURCE
```

Following are all the sections:

```
Sections of the manual pages
    The manual Sections are traditionally defined as follows:

    1 User commands (Programs)
              Those commands that can be executed by the user from within a shell.

    2 System calls
              Those functions which wrap operations performed by the kernel.

    3 Library calls
              All library functions excluding the system call wrappers (Most of the libc functions).

    4 Special files (devices)
              Files found in /dev which allow to access to devices through the kernel.

    5 File formats and configuration files
              Describes various human-readable file formats and configuration files.

    6 Games   Games and funny little programs available on the system.

    7 Overview, conventions, and miscellaneous
              Overviews or descriptions of various topics, conventions and protocols, character set standards, the standard filesystem layout, and miscellaneous other things.

    8 System management commands
              Commands like mount(8), many of which only root can execute.
```

Exercise 1

Before we go ahead with understanding Linux manual pages, we want to solve this exercise and you can go ahead and explore the internet for the same. You can see an image, which elaborately explains all the sections of the manual pages. If it is given that this image has been taken from a page called Man-Pages of section 7, which describes the conventions of writing Linux manual pages, then can you tell us the commands that we wrote to arrive at what is shown in image.

Getting Help with Shell Builtins

There are few commands that don't have a dedicated man-page. Mostly these commands are shell builtins. We can think of shell builtins as internal commands and are contained within the shell itself. We don't have to run a separate file on disk to run them. Since it is not a part of a separate program, whenever we have to run a shell, it locates the commands in itself and runs them.

Let's look into the man page for shell. Since, the most commonly used shell is bash, hence we will type `$ man bash`. After opening the man page, we will do a forward search for the shell builtin commands by typing: `/Shell Builtin Commands`. We can keep moving on the search results till we stumble upon the part of man page which contains all the shell builtin commands.

Note:- n is used to jump forward from one search result of patterns to another (here "Shell Builtin Commands"). Similarly, we use **N** to jump backwards.

```
navdeep@navdeep-VirtualBox: ~
SHELL BUILTIN COMMANDS
       Unless  otherwise  noted,  each  builtin  command  documented in this section
as accepting options preceded by - accepts -- to signify the end of the options.  The
 :, true, false, and test/[
       builtins do not accept options and do not treat -- specially.  The exit, logou
t, return, break, continue, let, and shift builtins accept and process arguments begi
nning with  -  without  re-
       quiring  --.  Other builtins that accept arguments but are not specified as ac
cepting options interpret arguments beginning with - as invalid options and require -
- to prevent this interpre-
       tation.
       : [arguments]
              No effect; the command does nothing beyond expanding arguments and perf
orming any specified redirections.  The return status is zero.

        .  filename [arguments]
       source filename [arguments]
              Read and execute commands from filename in the current shell environmen
t and return the exit status of the last command executed from filename.  If filename
 does not contain a  slash,
              filenames  in PATH are used to find the directory containing filename.
```

Now, the above image shows the part of bash man page which talks about all the shell builtin commands. As we scroll down on this man page, we will be able to get a list of all the shell builtin commands. Interestingly, help is a shell builtin command. Let's start with the **help** command.

```
      help [-dms] [pattern]
             Display  helpful information about builtin commands.  If pattern is specified, help gives
detailed help on all commands matching pattern; otherwise help for all the builtins and shell
             control structures is printed.
             -d     Display a short description of each pattern
             -m     Display the description of each pattern in a manpage-like format
             -s     Display only a short usage synopsis for each pattern

             The return status is 0 unless no command matches pattern.
```

Now, we use **help** command for reading documentation of those commands (shell builtin commands) which don't have a dedicated man page. So, let's try the following command: `$ help help`.

```
navdeep@navdeep-VirtualBox:~$ help help
help: help [-dms] [pattern ...]
    Display information about builtin commands.

    Displays brief summaries of builtin commands.  If PATTERN is
    specified, gives detailed help on all commands matching PATTERN,
    otherwise the list of help topics is printed.

    Options:
      -d        output short description for each topic
      -m        display usage in pseudo-manpage format
      -s        output only a short usage synopsis for each topic matching
                PATTERN

    Arguments:
      PATTERN   Pattern specifying a help topic

    Exit Status:
    Returns success unless PATTERN is not found or an invalid option is given.
navdeep@navdeep-VirtualBox:~$ man help
No manual entry for help
navdeep@navdeep-VirtualBox:~$
```

The next shell builtin command that we will see is type command. **Type** command is used to determine the type of the command. Let's type `$ type help`:

```
navdeep@navdeep-VirtualBox: ~
navdeep@navdeep-VirtualBox:~$ type help
help is a shell builtin
navdeep@navdeep-VirtualBox:~$
```

Determining Type of Command

Let's go deeper into the **type** command. Let's type the following command: `$ help type`

```
navdeep@navdeep-VirtualBox:~$ help type
type: type [-afptP] name [name ...]
    Display information about command type.

    For each NAME, indicate how it would be interpreted if used as a
    command name.

    Options:
      -a        display all locations containing an executable named NAME;
                includes aliases, builtins, and functions, if and only if
                the `-p' option is not also used
      -f        suppress shell function lookup
      -P        force a PATH search for each NAME, even if it is an alias,
                builtin, or function, and returns the name of the disk file
                that would be executed
      -p        returns either the name of the disk file that would be executed,
                or nothing if `type -t NAME' would not return `file'
      -t        output a single word which is one of `alias', `keyword',
                `function', `builtin', `file' or `', if NAME is an alias,
                shell reserved word, shell function, shell builtin, disk file,
                or not found, respectively

    Arguments:
      NAME      Command name to be interpreted.

    Exit Status:
    Returns success if all of the NAMEs are found; fails if any are not found.
navdeep@navdeep-VirtualBox:~$
```

Let's look at one more example of shell builtin command. Let's go ahead and type: $ man while

```
navdeep@navdeep-VirtualBox: ~
navdeep@navdeep-VirtualBox:~$ man while
No manual entry for while
navdeep@navdeep-VirtualBox:~$
```

Let's figure out the type of this command. When we type: $ type while, it says that "while is a shell keyword", which is another way of saying while is a shell builtin command. Let us go ahead and figure out its documentation with the help command.

```
navdeep@navdeep-VirtualBox: ~
navdeep@navdeep-VirtualBox:~$ man while
No manual entry for while
navdeep@navdeep-VirtualBox:~$ type while
while is a shell keyword
navdeep@navdeep-VirtualBox:~$ help while
while: while COMMANDS; do COMMANDS; done
    Execute commands as long as a test succeeds.

    Expand and execute COMMANDS as long as the final command in the
    `while' COMMANDS has an exit status of zero.

    Exit Status:
    Returns the status of the last command executed.
navdeep@navdeep-VirtualBox:~$
```

So, that's how we can get help for the shell builtin commands. Moreover, most commonly used commands have help documentation which can be accessed using `--help` command. For example: if we type `$ man --help`, we will get following results:

```
navdeep@navdeep-VirtualBox: ~
                            foo-bar'

 Controlling formatted output:
  -P, --pager=PAGER          use program PAGER to display output
  -r, --prompt=STRING        provide the `less' pager with a prompt

  -7, --ascii                display ASCII translation of certain latin1 chars
  -E, --encoding=ENCODING    use selected output encoding
      --no-hyphenation, --nh turn off hyphenation
      --no-justification,                            --nj   turn off justification
  -p, --preprocessor=STRING  STRING indicates which preprocessors to run:
                             e - [n]eqn, p - pic, t - tbl,
g - grap, r - refer, v - vgrind

  -t, --troff                use groff to format pages
  -T, --troff-device[=DEVICE]   use groff with selected device

  -H, --html[=BROWSER]       use www-browser or BROWSER to display HTML output
  -X, --gxditview[=RESOLUTION]   use groff and display through gxditview
                             (X11):
                             -X = -TX75, -X100 = -TX100, -X100-12 = -TX100-12
  -Z, --ditroff              use groff and force it to produce ditroff

  -?, --help                 give this help list
      --usage                give a short usage message
  -V, --version              print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.

Report bugs to cjwatson@debian.org.
navdeep@navdeep-VirtualBox:~$
```

Exercise 2

What command will be used to read about man-page on sync system call (Please note that it is not the sync command, it is sync system call). What command will be used to read sync's local man page that is kept in /usr/local/share/man?

Exercise 3

Explore and find commands to do the following:

1. Create a file in vim editor
2. Write your name in the created file
3. Save and close the vim editor using commands

Exercise 4

Explore the man page to find out the usage of the following commands:

1. ls
2. touch
3. rm
4. cp
5. mv
6. cat

Exercise 5

Explore the man page resources and the internet to find the main differences between man and info? What are the main advantages of each? Differentiate between the output of the following commands: "man info" and "info info".

Conclusion

This document helps you understand the basic commands available on the Linux terminal. We need to constantly explore and keep digging the man pages to explore other commands. The key takeaway from this document is how to get help on the Linux terminal.