4. To Understand basic commands like:- date, cal, echo, bc, ls, who, whoami, hostname, uname, tty, aliase

- **date -** The date command in Linux is a utility used to display and, with appropriate permissions, set the system's date and time.
  Common Format Specifiers:
  %Y: Year (e.g., 2025)
  %m: Month (e.g., 09)
  %d: Day of month (e.g., 27)
  %H: Hour (24-hour format, e.g., 20)
  %M: Minute (e.g., 51)
  %S: Second (e.g., 30)
  %A: Full weekday name (e.g., Saturday)
  %B: Full month name (e.g., September)
  %D: Date in mm/dd/yy format (e.g., 09/27/25)
  %T: Time in HH:MM:SS format (e.g., 20:51:30)

```
root@UbuntuVM:/home# date
Sat Sep 27 07:38:10 PM IST 2025
root@UbuntuVM:/home# date +%d
27
root@UbuntuVM:/home# date +%a
Sat
root@UbuntuVM:/home# date +%t

root@UbuntuVM:/home# date +%r
07:39:02 PM
root@UbuntuVM:/home# date +m
m
root@UbuntuVM:/home# date +%m
09
root@UbuntuVM:/home# date +%D
09/27/25
```

- **cal -** The cal command in Linux is a utility used to display a calendar in the terminal. It is a standard program available on Unix and Unix-like operating systems, including Linux.

  Specify the first day of the week:

  - **cal -m**: Sets Monday as the first day of the week (ISO 8601 standard).
  - **cal -s**: Sets Sunday as the first day of the week (customary North American numbering).

```
root@UbuntuVM:/home# cal -1
   September 2025
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

```
root@UbuntuVM:/home# cal 2005
                                2005
      January               February              March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1            1  2  3  4  5            1  2  3  4  5
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   6  7  8  9 10 11 12
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  13 14 15 16 17 18 19
16 17 18 19 20 21 22  20 21 22 23 24 25 26  20 21 22 23 24 25 26
23 24 25 26 27 28 29  27 28                 27 28 29 30 31
30 31

       April                 May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2   1  2  3  4  5  6  7            1  2  3  4
 3  4  5  6  7  8  9   8  9 10 11 12 13 14   5  6  7  8  9 10 11
10 11 12 13 14 15 16  15 16 17 18 19 20 21  12 13 14 15 16 17 18
17 18 19 20 21 22 23  22 23 24 25 26 27 28  19 20 21 22 23 24 25
24 25 26 27 28 29 30  29 30 31              26 27 28 29 30
        July                August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6               1  2  3
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   4  5  6  7  8  9 10
10 11 12 13 14 15 16  14 15 16 17 18 19 20  11 12 13 14 15 16 17
17 18 19 20 21 22 23  21 22 23 24 25 26 27  18 19 20 21 22 23 24
24 25 26 27 28 29 30  28 29 30 31           25 26 27 28 29 30
31

      October              November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1            1  2  3  4  5               1  2  3
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  11 12 13 14 15 16 17
16 17 18 19 20 21 22  20 21 22 23 24 25 26  18 19 20 21 22 23 24
23 24 25 26 27 28 29  27 28 29 30           25 26 27 28 29 30 31
30 31
root@UbuntuVM:/home# cal 9 2025
   September 2025
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

- **echo -** The echo command in Linux is a fundamental utility used to display lines of text or strings passed as arguments to the standard output, typically the terminal. It is a built-in command in most shells, including Bash, and is widely used in shell scripting for various purposes.
  Key functionalities of the echo command:
  - **Displaying text:** The most common use is to simply print a string to the terminal.

```
echo "Hello, World!"
```

- **Displaying variable values:** It can be used to output the value of environment variables or user-defined variables.

```
echo "Current user's home directory: $HOME"
```

- **Redirection to files:** The output of echo can be redirected to a file using > (to overwrite) or >> (to append).

```
echo "This text will be written to a file." > my_file.txt
echo "This text will be appended." >> my_file.txt
```

```
root@UbuntuVM:/home# echo hello world
hello world
root@UbuntuVM:/home# echo hello                world
hello world
root@UbuntuVM:/home# echo "hello               world"
hello               world
```

- **bc -** bc in Linux stands for "Basic Calculator," but it is an arbitrary-precision calculator language that offers more advanced features than a simple calculator. It can be used for various mathematical calculations directly from the command line or within shell scripts.

```
root@UbuntuVM:/home# bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software
 Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
5 + 67
72
4 * 7 ; 40432/200
28
202
```

Key Features and Usage:

- **Interactive Mode:** Typing bc in the terminal and pressing Enter starts an interactive session where you can input expressions and get results immediately.

```
$ bc
1+1
2
5*8^3
2560
quit
```

- **Command-line Calculation:** You can pipe expressions to bc for non-interactive calculations.

```
$ echo "scale=2; 10/3" | bc
3.33
```

- **ls -** The ls command is one of the most used commands in the Linux terminal to display the files and directories or path in the terminal. So, using the ls command is a basic skill for navigating the Linux file system, handling files, and managing directories.

Syntax of `ls` command in Linux

```
ls [option] [file/directory]
```

Commonly Used Options in `ls` Command

Options	Description
-l	known as a long format that displays detailed information about files and directories.
-a	Represent all files Include hidden files and directories in the listing.
-t	Sort files and directories by their last modification time, displaying the most recently modified ones first.
-r	known as reverse order which is used to reverse the default order of listing.
-S	Sort files and directories by their sizes, listing the largest ones first.
-R	List files and directories recursively, including subdirectories.
-i	known as inode which displays the index number (inode) of each file and directory.
-g	known as group which displays the group ownership of files and directories instead of the owner.
-h	Print file sizes in human-readable format (e.g., 1K, 234M, 2G).
-d	List directories themselves, rather than their contents.

Practical Examples of the ls command:

1. Open Last Edited File Using `ls -t `

This options lists all the files in the folder which is sorted by time which means that newest file comes first. When we use the head option in it they simply picks the first file from the list (which is your most recently changed file).

```
maverick@maverick-Inspiron-5548:~$ vi first.txt
maverick@maverick-Inspiron-5548:~$ vi second.txt
maverick@maverick-Inspiron-5548:~$ ls -t | head -1
second.txt
maverick@maverick-Inspiron-5548:~$
```

2. Display One File Per Line Using `ls -1 `

If you want to list all files and folders in your current directory with each name on a new line, you can use the ls -1 command

```
maverick@maverick-Inspiron-5548:~$ ls -1
1.c
a.out
ass8_1.c
binary.txt
cfile.c
c++file.cpp
cfile.o
cfile.so
client.c
Desktop
Documents
Downloads
end.txt
Exam
examples.desktop
FALCONN-1.2
fifo1.c
fifo2.c
first.txt
glove.cc
google-chrome-stable_current_amd64.deb
kv
libCfile.so
```

3. Display All Information About Files/Directories Using `ls -l`

The ls -l command in Linux is used to list the detailed information about the files and directories in the current folder.

```
maverick@maverick-Inspiron-5548:~$ ls -l
total 44892
-rw-rw-r--  1 maverick maverick     1176 Feb 16 00:19 1.c
-rwxrwxr-x  1 maverick maverick     9008 May 10 22:54 a.out
-rw-rw-r--  1 maverick maverick      484 Mar 29 22:18 ass8_1.c
-rw-rw-r--  1 maverick maverick    19920 Feb 16 00:20 binary.txt
-rw-rw-r--  1 maverick maverick       67 May 31 13:16 cfile.c
-rw-rw-r--  1 maverick maverick      187 May 31 13:21 c++file.cpp
-rw-rw-r--  1 maverick maverick     1552 May 31 13:37 cfile.o
-rwxrwxr-x  1 maverick maverick     8120 May 31 13:37 cfile.so
-rw-rw-r--  1 maverick maverick     1017 Feb 17 04:43 client.c
drwxr-xr-x  2 maverick maverick     4096 May 27 22:28 Desktop
drwxr-xr-x  2 maverick maverick     4096 Apr  2 04:11 Documents
drwxr-xr-x  2 maverick maverick     4096 May 31 13:12 Downloads
-rw-rw-r--  1 maverick maverick       54 Mar 29 22:23 end.txt
drwxrwxr-x 11 maverick maverick     4096 Nov 18  2016 Exam
-rw-r--r--  1 maverick maverick     8980 Nov  6  2016 examples.desktop
drwxr-xr-x  6 maverick maverick     4096 Nov 18  2016 FALCONN-1.2
-rw-rw-r--  1 maverick maverick      513 May 10 22:47 fifo1.c
-rw-rw-r--  1 maverick maverick      496 May 10 22:47 fifo2.c
-rw-rw-r--  1 maverick maverick      152 Jun  3 16:43 first.txt
-rw-r--r--  1 maverick maverick    10856 Nov 18  2016 glove.cc
-rw-rw-r--  1 maverick maverick 45750028 Nov  1  2016 google-chrome-stable_curr
nt_amd64.deb
```

- **who -** The "who" command in Linux is a command-line utility used to display information about users currently logged into the system.

  When executed without any options, the "who" command typically outputs:

  - **Username:** The login name of the user.
  - **Terminal:** The terminal or console through which the user is logged in (e.g., tty1, pts/0).
  - **Login Time:** The date and time when the user logged in.
  - **Remote Hostname/IP:** The hostname or IP address from which the user connected, if applicable.

```
root@UbuntuVM:/home# who
lakshay  seat0        2025-09-27 23:00 (login screen)
lakshay  tty2         2025-09-27 23:00 (tty2)
lakshay  pts/1        2025-09-27 23:01
root@UbuntuVM:/home# who -u
lakshay  seat0        2025-09-27 23:00   ?          1780 (login screen)
lakshay  tty2         2025-09-27 23:00 00:02        1780 (tty2)
lakshay  pts/1        2025-09-27 23:01   .          2901
root@UbuntuVM:/home# who -hu
who: invalid option -- 'h'
Try 'who --help' for more information.
root@UbuntuVM:/home# who -Hu
NAME     LINE         TIME             IDLE          PID COMMENT
lakshay  seat0        2025-09-27 23:00   ?          1780 (login screen)
lakshay  tty2         2025-09-27 23:00 00:03        1780 (tty2)
lakshay  pts/1        2025-09-27 23:01   .          2901
```

- **whoami -** The whoami command in Linux is a simple utility that displays the username of the current effective user. When executed in a terminal, it outputs the username associated with the user session that invoked the command**.**

```
root@UbuntuVM:/home# who am i
lakshay  pts/1        2025-09-27 23:01
```

Purpose and Use Cases:

- **Identifying Current User:** It's primarily used to quickly determine which user account is currently active in the shell. This is particularly useful when working on shared systems or after switching users with commands like su or sudo.
- **Scripting:** whoami can be incorporated into scripts to check the current user's identity before executing certain commands or performing actions that require specific user permissions. For example, a script might use whoami to verify if it's being run as root before attempting to modify system files.
- **Debugging:** It can assist in debugging issues where a script or process might be running under an unexpected user account.

- **hostname -** The hostname command in Linux is used to display or set the system's hostname, domain name, or IP address.

Syntax of the `hostname` command in Linux

```
hostname -[option] [file]
```

The hostname command supports various **options** to **retrieve specific information or perform actions:**

-a, --alias: Displays the alias name of the host (if set).

```
root@jayesh-VirtualBox:~# hostname -a

root@jayesh-VirtualBox:~#
```

-A, --all-fqdns: Displays all Fully Qualified Domain Names (FQDNs) of the host.

```
root@jayesh-VirtualBox:~# hostname -A
jayesh-VirtualBox
```

-b, --boot: Always sets a hostname (useful during boot).

```
root@jayesh-VirtualBox:~# hostname -b
jayesh-VirtualBox
```

-d, --domain: Displays the DNS domain name.

```
root@jayesh-VirtualBox:~# hostname -d
root@jayesh-VirtualBox:~#
```

-f, --fqdn, --long: Displays the FQDN of the host.

```
root@jayesh-VirtualBox:~# hostname -f
jayesh-VirtualBox
```

-i, --ip-address: Displays the IP address(es) of the host.

```
root@jayesh-VirtualBox:~# hostname -i
127.0.1.1
```

-I, --all-ip-addresses: Displays all configured IP addresses on all network interfaces.

```
root@jayesh-VirtualBox:~# hostname -I
10.143.90.2
```

-s, --short: Displays the short hostname (without the domain name).

```
root@jayesh-VirtualBox:~# hostname -s
jayesh-VirtualBox
```

- **uname -** The uname command in Linux is a command-line utility used to display system information. The name "uname" is an abbreviation for "UNIX name."
  When executed without any options, uname displays the kernel name, which is typically "Linux."
  Various options can be used with uname to retrieve specific system details:

```
root@UbuntuVM:/home# uname -a
Linux UbuntuVM 6.14.0-29-generic #29~24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Aug 14 16:52:50 UT
C 2 x86_64 x86_64 x86_64 GNU/Linux
root@UbuntuVM:/home# uname -s
Linux
root@UbuntuVM:/home# uname -n
UbuntuVM
root@UbuntuVM:/home# uname -r
6.14.0-29-generic
root@UbuntuVM:/home# uname -v
#29~24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Aug 14 16:52:50 UTC 2
root@UbuntuVM:/home# uname -m
x86_64
root@UbuntuVM:/home# uname -p
x86_64
root@UbuntuVM:/home# uname -i
x86_64
root@UbuntuVM:/home# uname -o
GNU/Linux
root@UbuntuVM:/home#
```

-a or --all: Prints all available system information in a specific order (kernel name, network node hostname, kernel release, kernel version, machine hardware name, processor type, hardware platform, and operating system).
-s or --kernel-name: Displays the kernel name.
-n or --nodename: Shows the network node hostname.
-r or --kernel-release: Prints the kernel release version.
-v or --kernel-version: Displays the kernel version.

-m or --machine: Shows the machine hardware name (e.g., x86_64).
-p or --processor: Prints the processor type.
-i or --hardware-platform: Displays the hardware platform.
-o or --operating-system: Shows the operating system name.

- **tty -** The tty command in Linux is used to print the file name of the terminal connected to standard input. This is particularly useful for identifying the specific terminal device you are currently interacting with.

```
root@UbuntuVM:/home# tty
/dev/pts/1
root@UbuntuVM:/home# tty -s
root@UbuntuVM:/home# tty -silent
tty: invalid option -- 'i'
Try 'tty --help' for more information.
root@UbuntuVM:/home# tty --help
Usage: tty [OPTION]...
Print the file name of the terminal connected to standard input.

  -s, --silent, --quiet   print nothing, only return an exit status
      --help        display this help and exit
      --version     output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/tty>
or available locally via: info '(coreutils) tty invocation'
root@UbuntuVM:/home# tty --version
tty (GNU coreutils) 9.4
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by David MacKenzie.
```

Purpose and Use Cases:

- Identifying the current terminal: It helps you understand which terminal device file is associated with your current session.
- Scripting: In scripts, tty can be used to check if the script is being run interactively in a terminal or non-interactively (e.g., as part of a cron job or piped input). This allows scripts to behave differently based on the execution environment.
- Troubleshooting: It can assist in debugging issues related to terminal input/output or understanding where a process is attached.

Options:

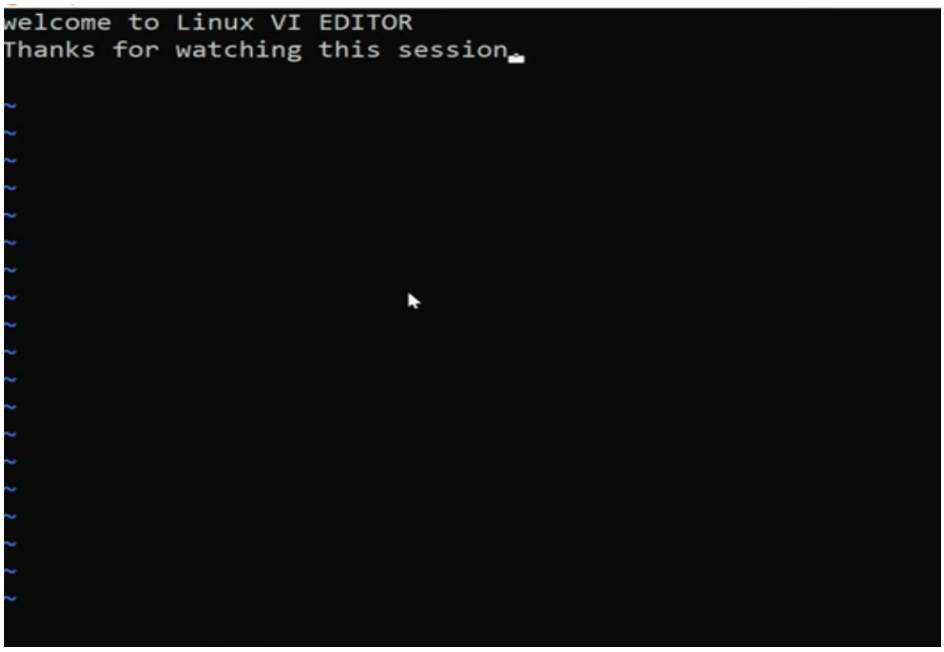
- -s or --silent: This option suppresses output and only returns an exit status. The exit status is 0 if standard input is a terminal, 1 if not, and 2 for incorrect arguments.
- --help: Displays help information and exits.
- --version: Outputs version information and exits.

5. To Understand vi basics, Three modes of vi Editor, how to write, save, execute a shell script in vi Editor

The vi editor is a powerful, modal text editor commonly found in Unix-like operating systems. Understanding its modes is crucial for effective use.

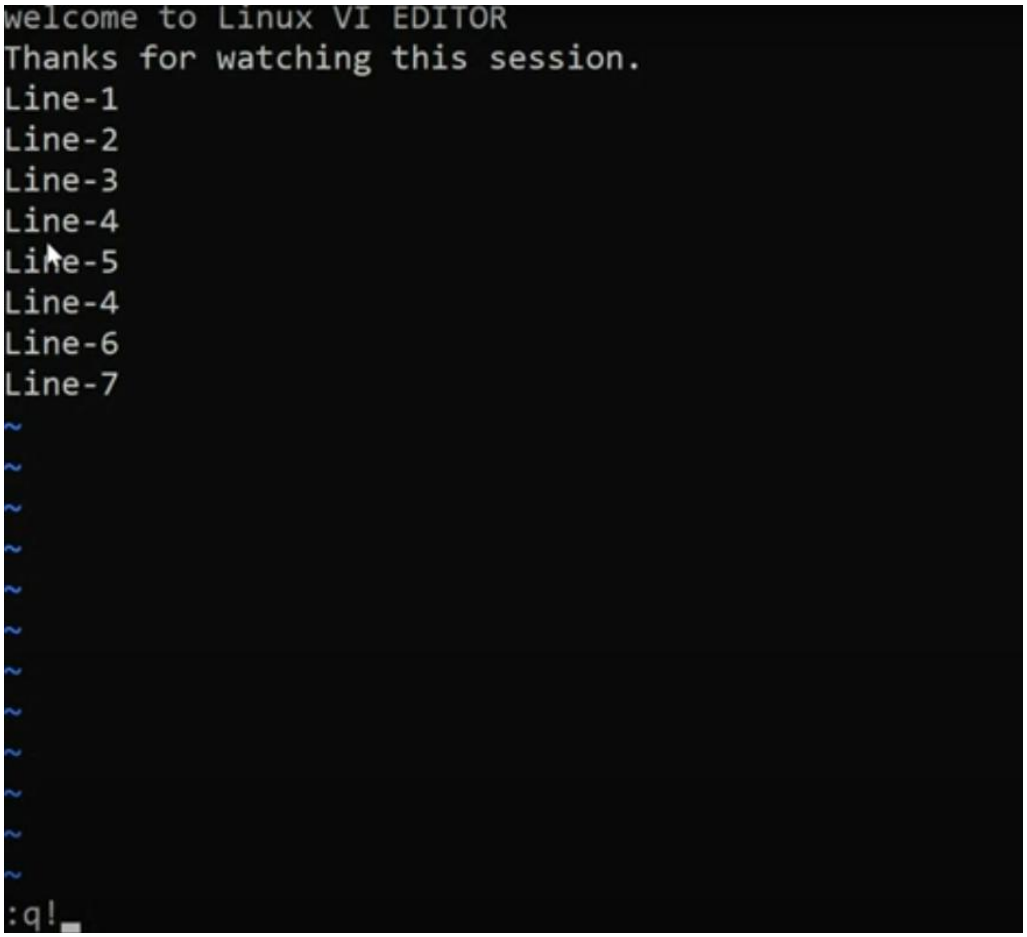
Three Modes of vi Editor:

- **Command Mode (Normal Mode):** This is the default mode when vi starts. In this mode, keystrokes are interpreted as commands for navigation, deletion, copying, pasting, and other editing operations. You can switch to Command Mode from Insert Mode by pressing the Esc key.



- **Insert Mode:** In this mode, characters typed are inserted directly into the file as text. You enter Insert Mode from Command Mode by pressing keys like i (insert at cursor), a (append after cursor), o (open new line below), or O (open new line above).

- **Last Line Mode (Ex Mode):** This mode is accessed from Command Mode by pressing the colon : key. It allows you to execute more complex commands, such as saving files, quitting, searching, and performing global substitutions.



<u>Writing, Saving, and Executing a Shell Script in vi Editor:</u>

1. Open or Create the Script.

   ```
   $ vi 01_basic.sh
   ```

   This command opens myscript.sh for editing. If the file does not exist, vi will create it.

2. Write the Script:
   - Press i to enter Insert Mode.

- Type your shell script content. For example:

```
#!/bin/bash

echo "Hey Buddy!"
~
~
~
~
~
~
~
~
~
~
~
```

3. Press Esc to return to Command Mode.

```
#!/bin/bash

echo "Hey Buddy!"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

```
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ vi 01_basic.sh
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$
```

4. Save the Script:
   - In Command Mode, press : to enter Last Line Mode.
   - Type w and press Enter to save the changes to the file.
   - Alternatively, type :wq to save and quit vi, or :x to save and quit only if changes were made.


```
[paul@redhat01 myscripts]$ cat 01_basic.sh
#!/bin/bash

echo "Hey Buddy!"
```

5. Make the Script Executable:
   - Exit vi if you haven't already.
   - In your shell, write ls –ltr to check the permission of the script:

```
[paul@redhat01 myscripts]$ ls -ltr
total 4
-rw-rw-r--. 1 paul paul 31 Jul 26 06:33 01_basic.sh
```

   - Add Executable permissions if not already present using the following command:

```
[paul@redhat01 myscripts]$ chmod +x 01_basic.sh
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ ls -ltr
total 4
-rwxrwxr-x. 1 paul paul 31 Jul 26 06:33 01_basic.sh
```

6. Execute the Script:
   - In your shell, run the script:

```
[paul@redhat01 myscripts]$ bash 01_basic.sh
Hey Buddy!
```

Or

```
[paul@redhat01 myscripts]$ ./01_basic.sh
Hey Buddy!
```