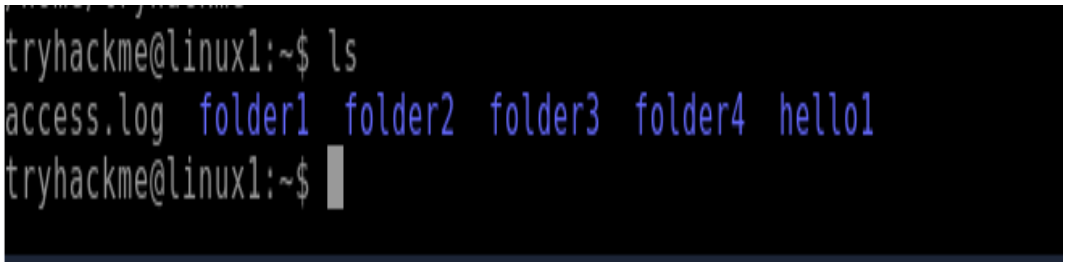
Linux Lab -2

Basic Linux Command

1. **Ls command :-**

   Displays information about files in the current directory.

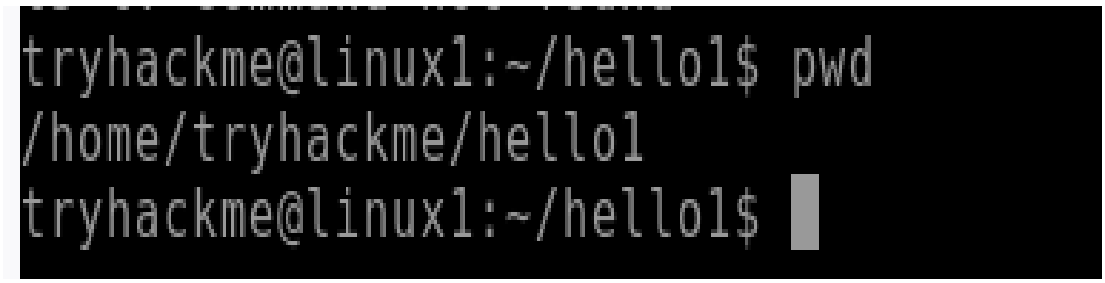
   Syntax :- **ls**



2. **Pwd command :-**

   Displays the current working directory.

   Syntax :- **pwd**



3. **Mkdir command :-**

   This mkdir command allows you to create fresh directories in the terminal itself.

   Syntax :- **mkdir <directory name>**

```
tryhackme@linux1:~$ mkdir hello1
tryhackme@linux1:~$
```

4. **Cd command :-**

   The cd command is used to navigate between directories.

   Syntax :- **cd <directory name>**

```
tryhackme@linux1:~$ mkdir hello1
tryhackme@linux1:~$ cd hello1
tryhackme@linux1:~/hello1$
```

5. **Rmdir command :-**

   The rmdir command is used to delete permanently an empty directory.

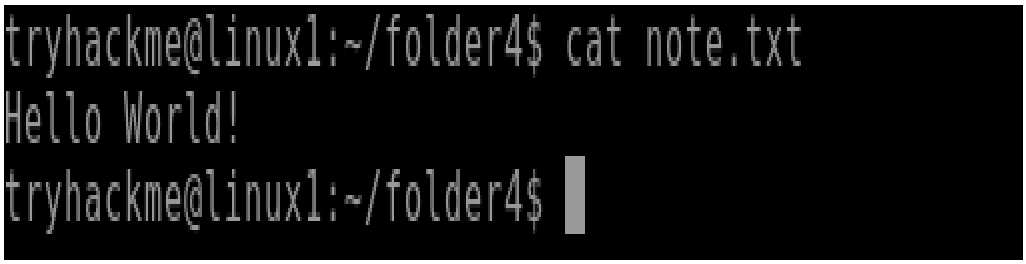
   Syntax :- **rmdir <directory name>**

```
tryhackme@linux1:~$ mkdir ps1
tryhackme@linux1:~$ ls
access.log  folder1  folder2  folder3  folder4  hello1  ps1
tryhackme@linux1:~$ rmdir ps1
tryhackme@linux1:~$ ls
access.log  folder1  folder2  folder3  folder4  hello1
tryhackme@linux1:~$
```

6. **Cat command :-**

Display file contents on terminal

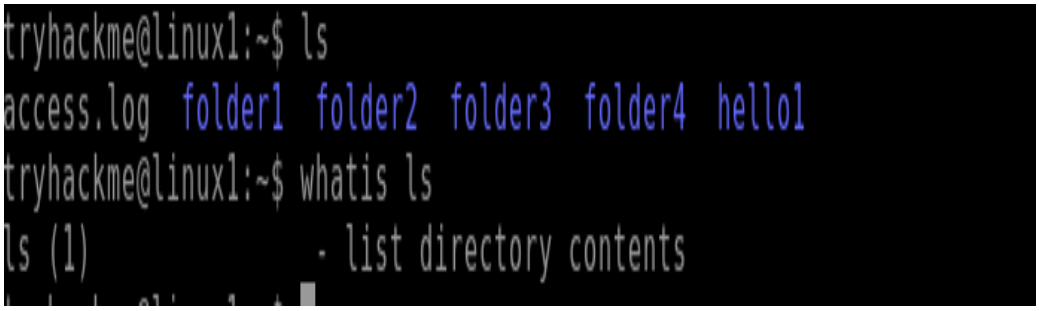
Syntax :- **cat <file name>**



7. Whatis command :-

whatis command in Linux is used to get a one-line manual page description.

Syntax :- **whatis [option] [command_name]**



8. Find command :-

The find command in Linux is a dynamic utility designed for comprehensive file and directory searches within a hierarchical structure.

Syntax :- **find [path] [options] [expression]**

```
tryhackme@linux1:~$ find
.
./hello1
./folder1
./.profile
./.bashrc
./folder4
./folder4/note.txt
./.bash_history
./.cache
./.cache/motd.legal-displayed
./access.log
./folder2
./.Xauthority
./folder3
./.bash_logout
tryhackme@linux1:~$
```

9. Echo command :-

echo command in Linux is specially used to print something in the terminal

Syntax :- **echo <Text>**

```
tryhackme@linux1:~$ echo "Hello Friend!"
Hello Friend!
tryhackme@linux1:~$
```

10. Cal command :-

The cal command is not the most famous command in the terminal but it functions to view the calendar for a particular month in the terminal. Let's see how this works.

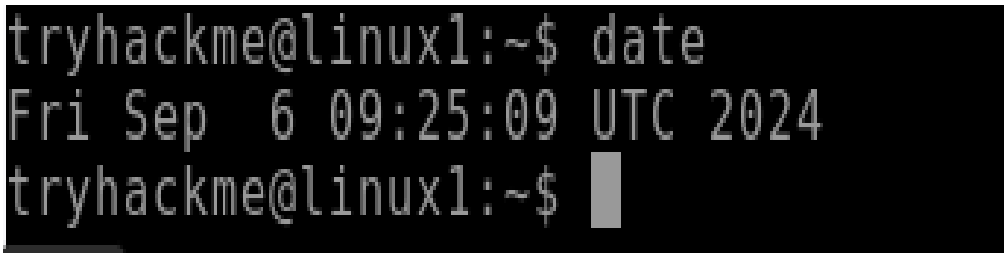
Syntax :- **cal <month> <Year>**

```
tryhackme@linux1:~$ cal 4 1997
     April 1997
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

tryhackme@linux1:~$
```

11. Date command :-

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

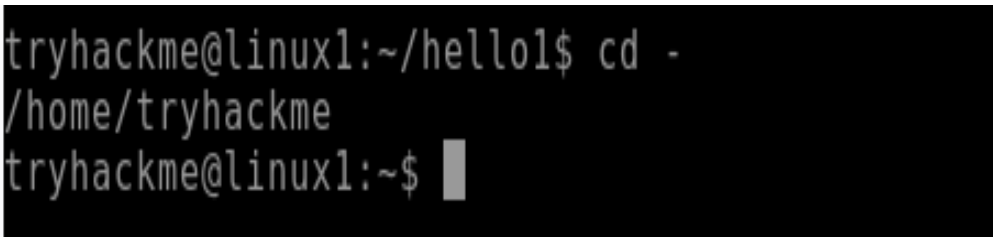
Syntax :- **date**



12. Cd - command :-

Cd - command is used to go back to previous directory in terminal of the system.

Syntax :- **cd –**

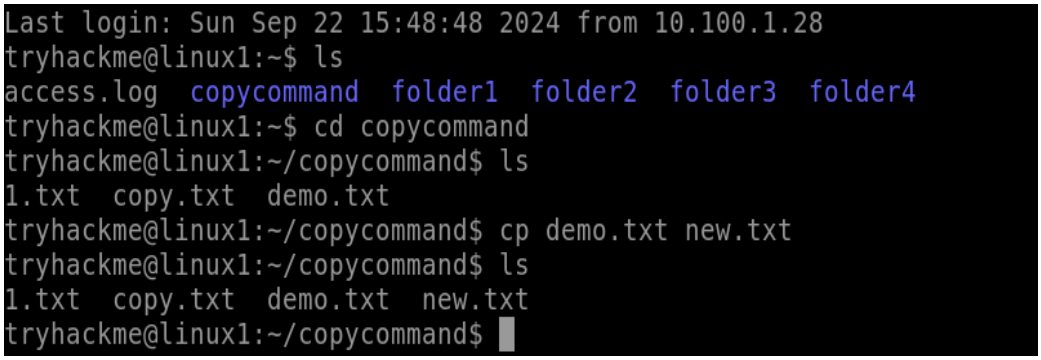


13. Cp command :-

The cp command copies files or directories from a source to a destination. It can handle single or multiple files and directories, and it can also overwrite existing files if specified.

Syntax: **cp [options] source destination**

Example:

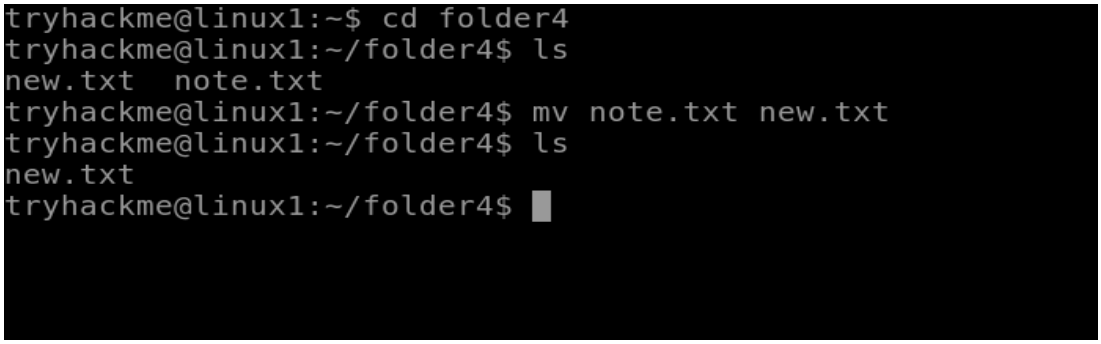


14. mv

The mv command in Linux is used to move or rename files and directories.

Syntax: **mv [options] source destination**

Example:



15. head

The head command outputs the first part of files or input data. It is commonly used to preview the beginning of a file or stream.

Syntax: **head [options] [file...]**

Example:

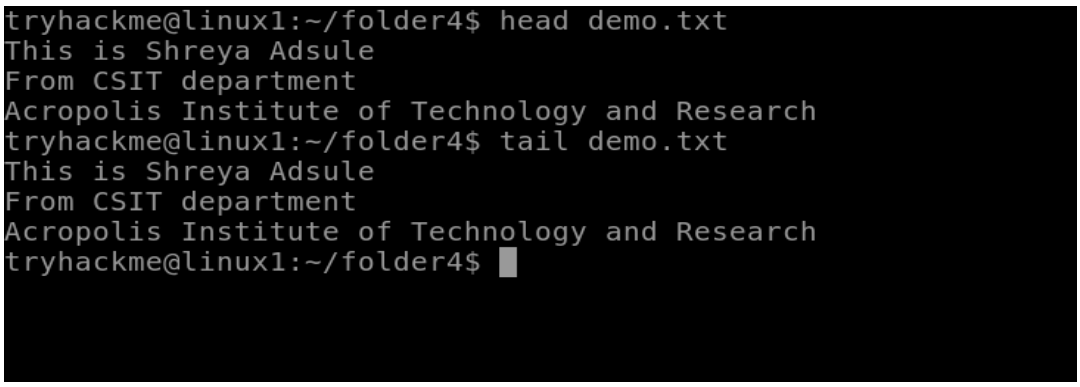
```
tryhackme@linux1:~/folder4$ head demo.txt
This is Shreya Adsule
From CSIT department
Acropolis Institute of Technology and Research
tryhackme@linux1:~/folder4$ tail demo.txt
This is Shreya Adsule
From CSIT department
Acropolis Institute of Technology and Research
tryhackme@linux1:~/folder4$
```

16. tail

The tail command outputs the last part of files or input data. It is often used to view the most recent entries in a log file or to monitor the end of a file for changes.

Syntax: **tail [options] [file...]**

Example :



17. sudo

The sudo command grants elevated privileges to run commands that require root or administrative permissions. It's typically used to perform system administration tasks.

Syntax: **sudo [options] command**

Example:

```
vboxuser@UBUNTU:~$ sudo -h
sudo - execute a command as another user

usage: sudo -h | -K | -k | -V
usage: sudo -v [-ABkNnS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-ABkNnS] [-g group] [-h host] [-p prompt] [-U user]
            [-u user] [command [arg ...]]
usage: sudo [-ABbEHkNnPS] [-r role] [-t type] [-C num] [-D directory]
            [-g group] [-h host] [-p prompt] [-R directory] [-T timeout]
            [-u user] [VAR=value] [-i | -s] [command [arg ...]]
usage: sudo -e [-ABkNnS] [-r role] [-t type] [-C num] [-D directory]
            [-g group] [-h host] [-p prompt] [-R directory] [-T timeout]
            [-u user] file ...

Options:
  -A, --askpass                 use a helper program for password prompting
  -b, --background              run command in the background
  -B, --bell                    ring bell when prompting
  -C, --close-from=num          close all file descriptors >= num
  -D, --chdir=directory         change the working directory before running
                                command
  -E, --preserve-env            preserve user environment when running command
```

18. ifconfig

The ifconfig (interface configuration) command is used to display or configure a network interface.

Syntax: **ifconfig [interface] [options]**

Example:

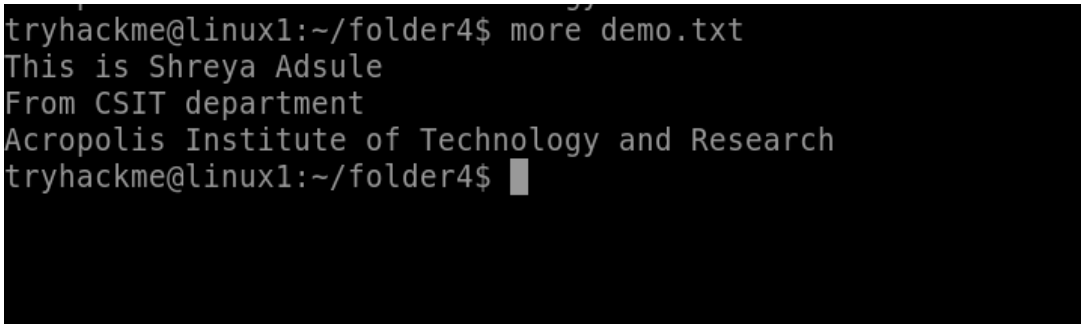
```
vboxuser@UBUNTU:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:feb7:96d  prefixlen 64  scopeid 0x20<link>
        inet6 fd00::a00:27ff:feb7:96d  prefixlen 64  scopeid 0x0<global>
        inet6 fd00::56cd:195c:2024:fba6  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:b7:09:6d  txqueuelen 1000  (Ethernet)
        RX packets 843  bytes 390380 (390.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1089  bytes 124455 (124.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 516  bytes 47295 (47.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 516  bytes 47295 (47.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vboxuser@UBUNTU:~$
```

19. more

The more command displays the contents of a file, pausing after each screen of text. It is useful for viewing long files that don't fit on a single screen.

Syntax: **more [options] [file]**

Example:

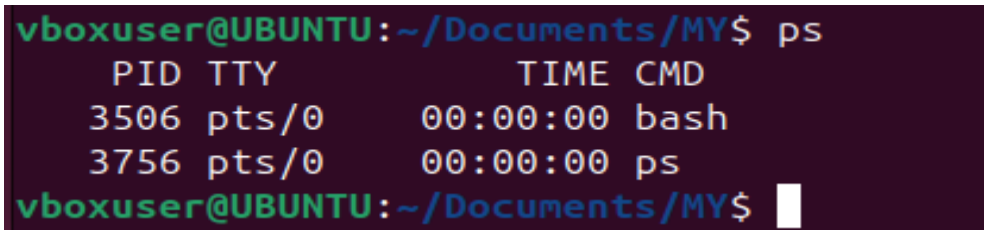


20. ps

The ps command provides a snapshot of current processes, showing details like process IDs (PIDs), terminal associated with the process, CPU and memory usage, and the command that started the process.

Syntax: p**s [options]**

Example:



21. cat >

Using cat > filename, you can start typing text directly into a new file. This command redirects the terminal input into the specified file until you signal that you're done.

Syntax: **cat > filename**

Example:

```
tryhackme@linux1:~/folder4$ cat > demo.txt
Hi me Student
tryhackme@linux1:~/folder4$ cat demo.txt
Hi me Student
tryhackme@linux1:~/folder4$
```

22. cat >>

Using cat >> filename, you can add new content to the end of a specified file. This command allows you to continue writing to the file without overwriting its current contents.

Syntax: **cat >> filename**

```
tryhackme@linux1:~/folder4$ ls
demo.txt  new.txt
tryhackme@linux1:~/folder4$ cat demo.txt
Hi me Student
tryhackme@linux1:~/folder4$ cat >> demo.txt
Hi, This is Shreya Adsule
tryhackme@linux1:~/folder4$ cat demo.txt
Hi me Student
Hi, This is Shreya Adsule
tryhackme@linux1:~/folder4$
```

23. ls -l

The ls -l command lists files and directories in a long format, showing detailed attributes for each item, including permissions, number of links, owner, group, size, and modification date.

Syntax: **ls -l [directory]**

Example:

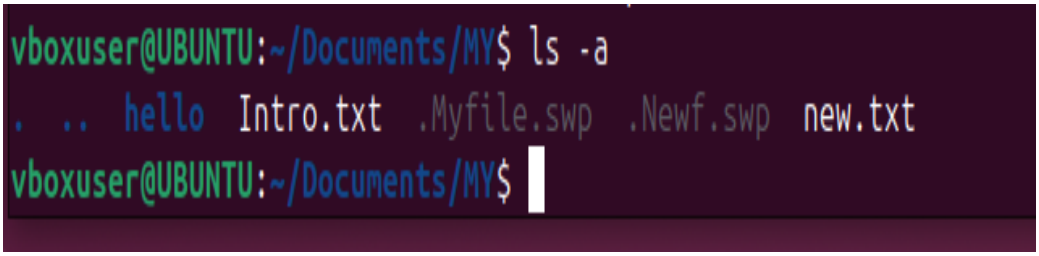
```
vboxuser@UBUNTU:~/Documents/MY$ ls -l
total 12
drwxrwxr-x 2 vboxuser vboxuser 4096 Sep 19 19:07 hello
-rw-rw-r-- 1 vboxuser vboxuser   28 Sep 20 09:09 Intro.txt
-rw-rw-r-- 1 vboxuser vboxuser   22 Sep 19 19:03 new.txt
vboxuser@UBUNTU:~/Documents/MY$
```

24. ls -a

The ls -a command displays all entries in a directory, including those that begin with a dot (.), which are considered hidden files in Unix-like systems.

Syntax: **ls -a [directory]**

Example:



25. touch

The touch command creates a new, empty file if the specified file does not exist. If the file already exists, it updates the access and modification timestamps to the current time without modifying the file's content.

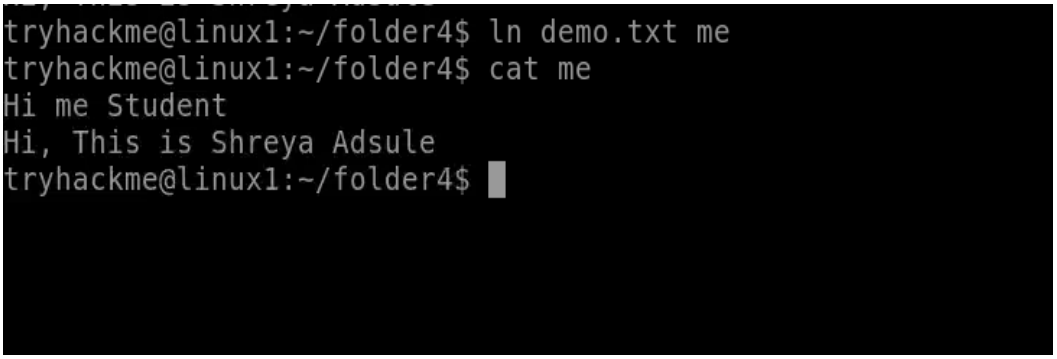
Syntax: **touch [options] filename**

26. ln

The ln command in Linux is used to create links between files. There are two types of links: hard links and symbolic (soft) links.

Syntax: **ln [options] target [link_name]**

Example:

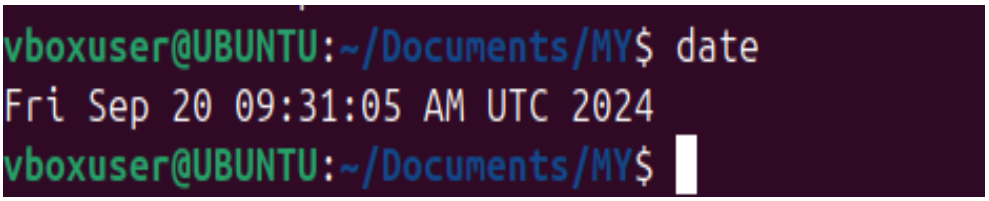


27. date

The date command in Linux is used to display or set the system date and time.

Syntax: **date [options] [+format]**

Example:

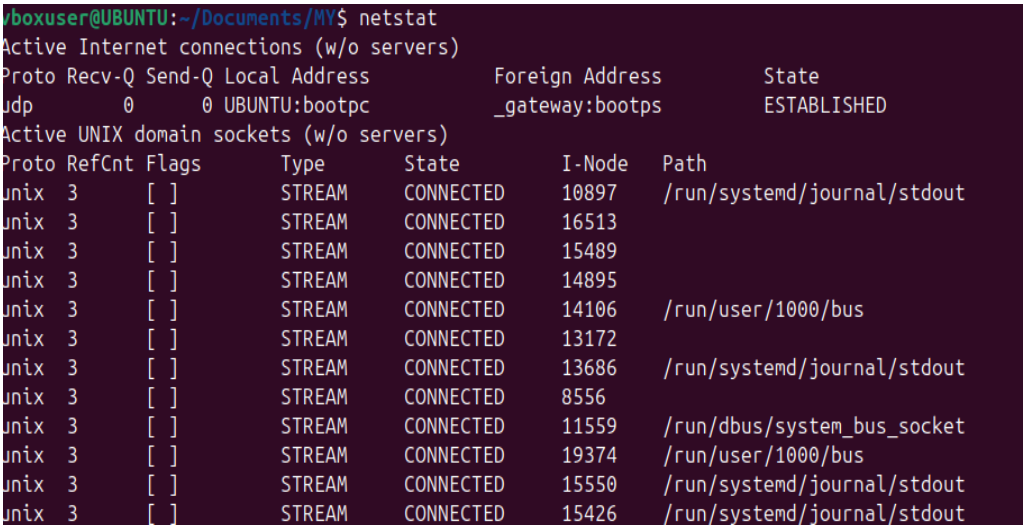


28. netstat

The netstat command provides information about active network connections and network interface statistics, helping users monitor and troubleshoot network issues.

Syntax: **netstat [options]**

Example:



29. gid

In Linux, GID stands for Group Identifier. It is a numeric value used to identify a specific group on the system. Each user in Linux can belong to one or more groups, and each group is assigned a unique GID.

Syntax: **id username**

Example:

```
vboxuser@UBUNTU:~/Documents/MY$ id
uid=1000(vboxuser) gid=1000(vboxuser) groups=1000(vboxuser),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),100(users),114(lpadmin)
vboxuser@UBUNTU:~/Documents/MY$
```

30. chmod

The chmod command allows users to specify who can read, write, or execute a file. Permissions can be set for three categories: the file owner, the group, and others.

Syntax: **chmod [options] mode file**

Example:

```
vboxuser@UBUNTU:~/Documents/MY$ chmod u+rw Intro.txt
vboxuser@UBUNTU:~/Documents/MY$
```

31. man

The man command is a built-in command that allows users to access the manual documentation for commands, functions, system calls, and other components in Linux.

Syntax: **man [options] command**

Example :

```
vboxuser@UBUNTU:~/Documents/MY$ man chmod
```

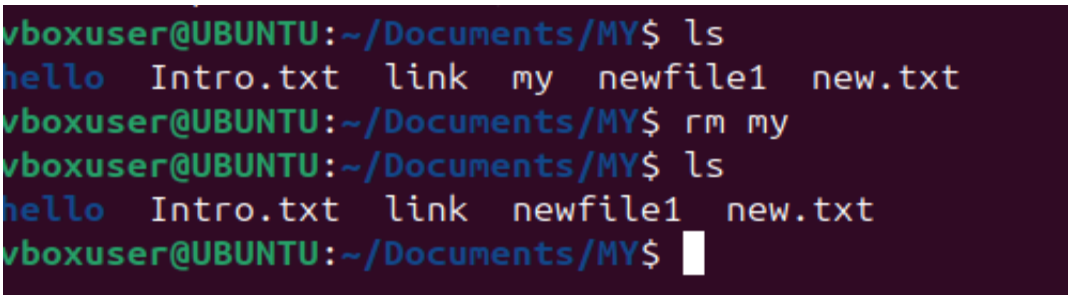
```
CHMOD(1)                          User Commands                          CHMOD(1)

NAME
       chmod - change file mode bits

SYNOPSIS
       chmod [OPTION]... MODE[,MODE]... FILE...
       chmod [OPTION]... OCTAL-MODE FILE...
       chmod [OPTION]... --reference=RFILE FILE...

DESCRIPTION
       This  manual  page documents the GNU version of chmod.  chmod changes the file mode bits of each given file according to mode, which can be either a symbolic representation
       of changes to make, or an octal number representing the bit pattern for the new mode bits.

       The format of a symbolic mode is [ugoa...][[-+=][perms...]...], where perms is either zero or more letters from the set rwxXst, or a single letter from the set ugo.  Multi-
       ple symbolic modes can be given, separated by commas.

       A combination of the letters ugoa controls which users' access to the file will be changed: the user who owns it (u), other users in the file's group (g), other  users  not
       in the file's group (o), or all users (a).  If none of these are given, the effect is as if (a) were given, but bits that are set in the umask are not affected.

       The  operator  +  causes the selected file mode bits to be added to the existing file mode bits of each file; - causes them to be removed; and = causes them to be added and
       causes unmentioned bits to be removed except that a directory's unmentioned set user and group ID bits are not affected.

       The letters rwxXst select file mode bits for the affected users: read (r), write (w), execute (or search for directories) (x), execute/search only if the file is  a  direc-
       tory  or  already  has  execute  permission for some user (X), set user or group ID on execution (s), restricted deletion flag or sticky bit (t).  Instead of one or more of
       these letters, you can specify exactly one of the letters ugo: the permissions granted to the user who owns the file (u), the permissions granted to  other  users  who  are
       members of the file's group (g), and the permissions granted to users that are in neither of the two preceding categories (o).

       A  numeric  mode  is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1.  Omitted digits are assumed to be leading zeros.  The first
       digit selects the set user ID (4) and set group ID (2) and restricted deletion or sticky (1) attributes.  The second digit selects permissions for the  user  who  owns  the
       file:  read  (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in
       the file's group, with the same values.

       chmod never changes the permissions of symbolic links; the chmod system call cannot change their permissions.  This is not a problem since the permissions of symbolic links
       are never used.  However, for each symbolic link listed on the command line, chmod changes the permissions of the pointed-to file.  In  contrast,  chmod  ignores  symbolic
       links encountered during recursive directory traversals.
```

32. rm

The rm command allows users to delete files and directories from the filesystem. It is a powerful command that permanently removes files without placing them in a recycle bin or trash.

Syntax: **rm [options] file**

Example:

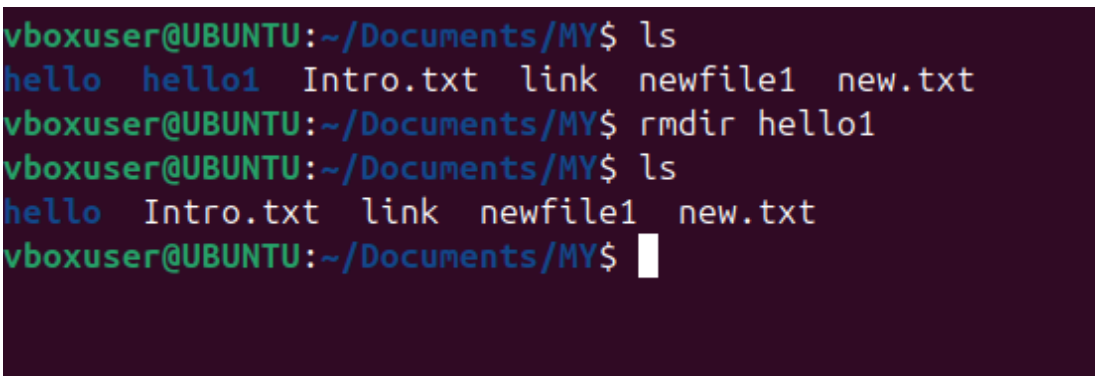


33. rmdir

The rmdir command allows users to delete directories, but it can only remove those that are empty. If the directory contains files or other directories, the command will fail.

Syntax: **rmdir [options] directory**

Example:



34. less

The less command provides a convenient way to scroll through text files, allowing both forward and backward navigation.

Syntax: **less [options] file**

Example:

35. grep

The grep command searches through the input (files or standard input) and prints lines that match a specified pattern. It's commonly used for text processing and searching logs.

Syntax: **grep [options] pattern [file...]**

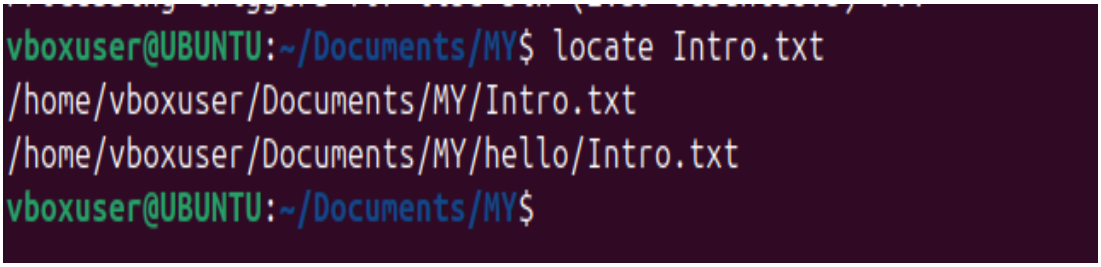
Example:

36. locate

The locate command searches for files and directories in a database that contains the paths of all files on the system. This database is typically updated daily by a background service (updatedb), allowing for fast searches.

Syntax: **locate [options] pattern**

Example:



37. sort

The sort command arranges the lines of a file or input in a specified order (ascending or descending). By default, it sorts in ascending order based on the ASCII values of characters.

Syntax**:** **sort [options] [file...]**

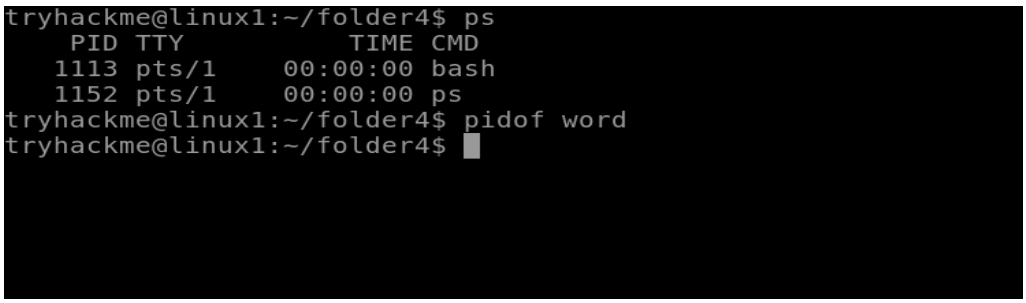
Example:

38. pid

The PID (process identification number) is a serial number (starting from 1) given by the operating system to uniquely identify the process. Every process started either by the operating system or by the user gets a PID in order of their invocation by the kernel.

Syntax: **pidof <exact_process_name>**

Example:



39. whoami

The command allows Linux users to see the currently logged-in user. The output displays the username of the effective user in the current shell. Additionally, whoami is useful in bash scripting to show who runs the script

Syntax : **whoami [OPTION]**

Example :

```
tryhackme@linux1:~$ whoami
tryhackme
tryhackme@linux1:~$
```

40. kill

kill command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually

Syntax : **kill [signal] PID**

Example :

```
tryhackme@linux1:~$ ps
    PID TTY          TIME CMD
   1113 pts/1    00:00:00 bash
   1192 pts/1    00:00:00 ps
tryhackme@linux1:~$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
tryhackme@linux1:~$ kill 1192
```

41. kill all

The killall command in Linux is a utility that terminates running processes based on their name. It can be useful when you need to kill multiple instances of a process or when you don't know the process ID (PID).

Syntax : **killall** [**-**] [**-signal**]

Example :

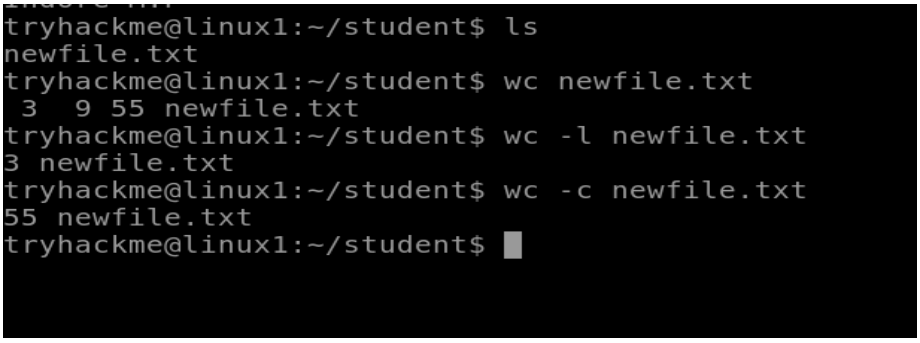
```
ubuntu@ubuntu1804:~$
ubuntu@ubuntu1804:~$ sudo killall yes
[1]   Terminated              yes > /dev/null
[2]   Terminated              sudo yes > /dev/null
[3]   Terminated              sudo yes > /dev/null
[5]-  Terminated              sudo yes > /dev/null
[6]+  Terminated              sudo yes > /dev/null
ubuntu@ubuntu1804:~$
```

42. wc

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

Syntax : **wc [option]... [file]...**

Example :



43. su

The su command in Linux switches users or executes commands as a different user. It's useful for administrative tasks that require elevated privileges.

Syntax : **su [options] [username]**

Example :