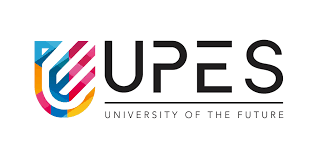
****

**Name : Mridul Singh**

**SAP ID : 500094181**

**Batch : B1 (H) DevOps**

**Roll no: R2142210501**

# Subject :

# Development Automation and Linux

**Lab File**

**Development Automation and Linux Lab**

**Experiment-1**

A shell is a special user program that provides an interface to the user to use operating system services. Shell accepts human-readable commands from the user and converts them into something which the kernel can understand. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.

* **Man**

The **man** command provides reference information on topics, such as commands, subroutines, and files. The **man** command provides one-line descriptions of commands specified by name. The **man** command also provides information on all commands whose descriptions contain a set of user-specified keywords.

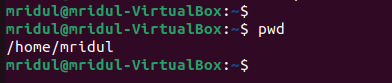
* **LS**

It is used to list information about files and directories within the file system. The ls utility is a part of the GNU core utilities package which is installed on all Linux distributions.



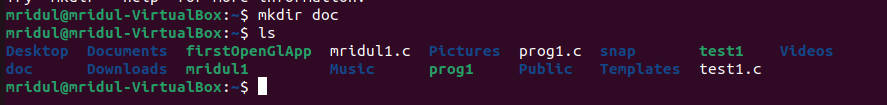
* **Pwd**

The **pwd** command writes to standard output the full path name of your current directory (from the root directory). All directories are separated by a / (slash). The root directory is represented by the first /, and the last directory named is your current directory.

****

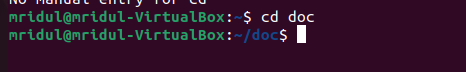
* **Mkdir**

Creates a directory or subdirectory. Command extensions, which are enabled by default, allow you to use a single mkdir command to create intermediate directories in a specified path.

****

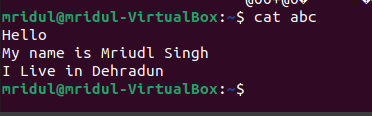
* **Cd**

Displays the name of the current directory or changes the current directory.

****

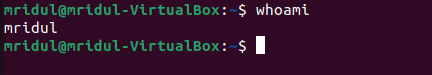
* **Cat**

The **cat** command reads each File parameter in sequence and writes it to standard output. If you do not specify a file name, the **cat** command reads from standard input. You can also specify a file name of **-** (dash) for standard input.



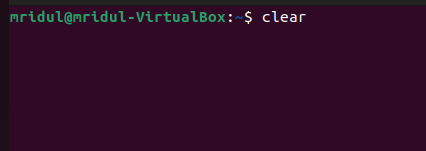
* **Whoami**

The whoami command allows Linux users to see the currently logged-in user.



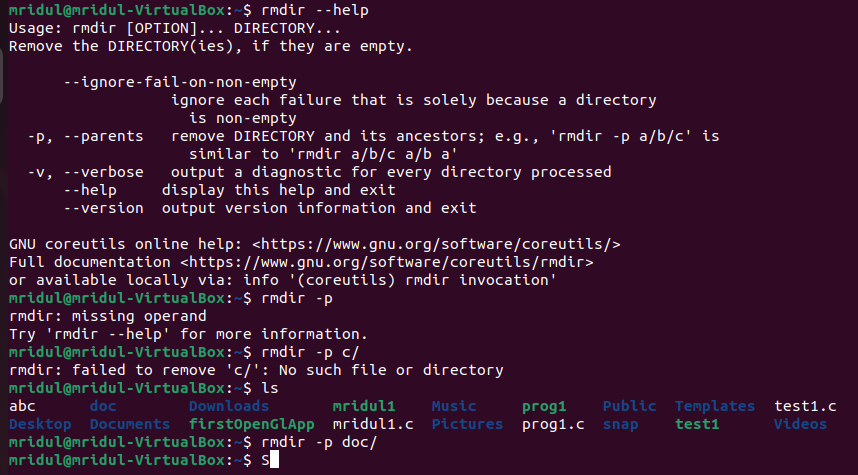
* Clear

clears the console if the console allows it.



* **Rmdir**

Removes empty directories

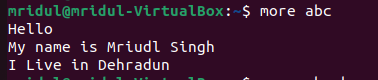


* **Touch**

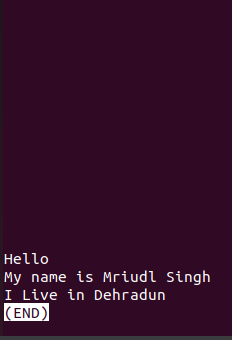
Change the timestamps

****

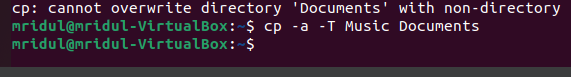
* **More**

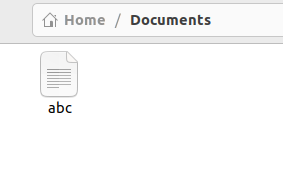
****

* **Less**

****

* **CP**

****

****

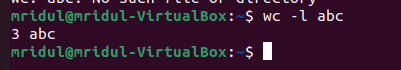
Say you want to copy a file to a new location making sure the original copy still exists in the main location. For that we do ctrl+c this is what cp does in Linux.

Now say you want to copy a file to a new location making sure the original copy is deleted from the main location. In that case you will use ctrl+x and this what mv does for you in Linux.

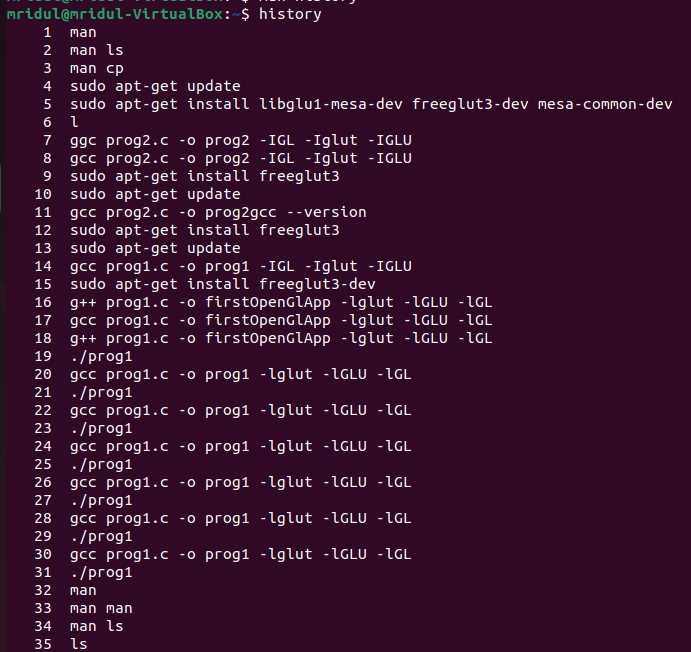
* **MV**

****

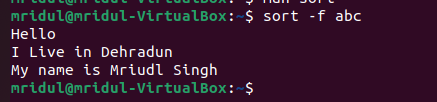
* **WC**

****

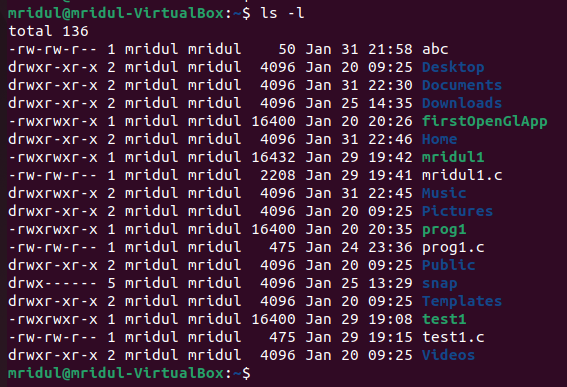
* **History**

****

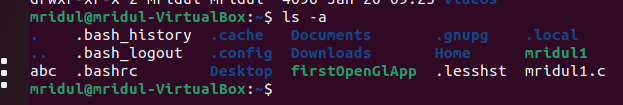
* **Sort**

****

* **ls -l**

****

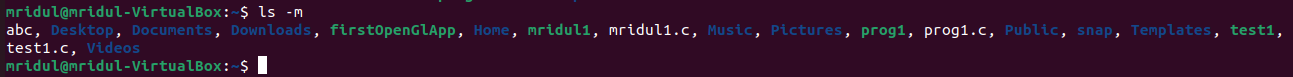
* **ls -a**

****

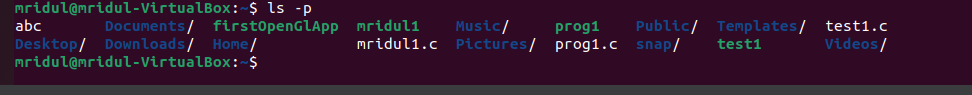
* **ls -c**

****

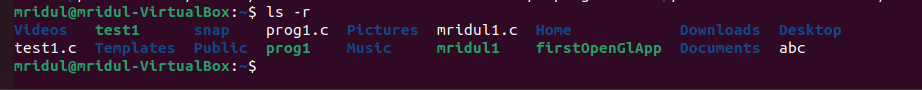
* **ls -m**

****

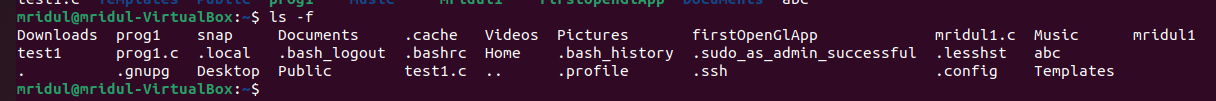
* **ls -p**

****

* **ls -r**

****

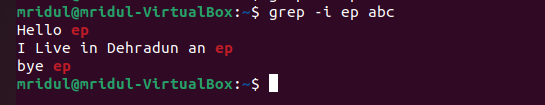
* **ls -f**

****

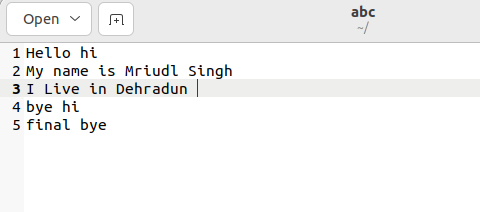
**Grep command in Unix/Linux**

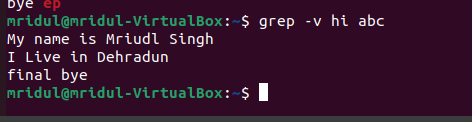
The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).

i) find all the patterns that start with “ep” in a given file



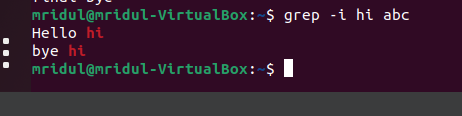
ii) Print those lines of a given text file which do not contain the word “hi”.



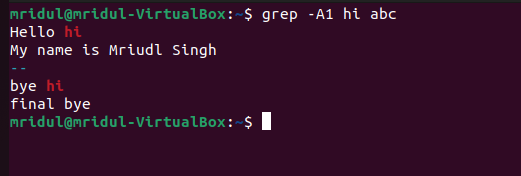


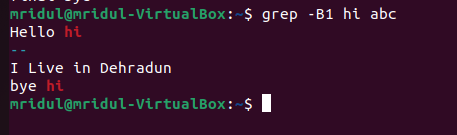
iii) Explain the usage of options -i, -A, -B, -C in grep command with examples

-A n : Prints searched line and nlines after the result.

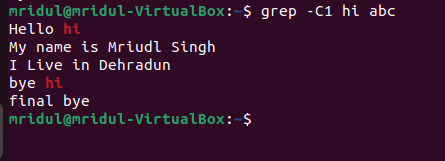


-B n : Prints searched line and n line before the result.





-C n : Prints searched line and n lines after before the result.

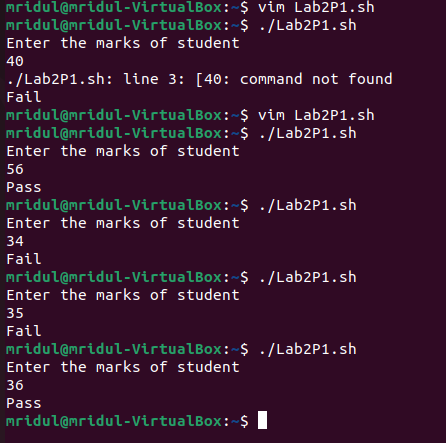


**Development Automation and Linux Lab**

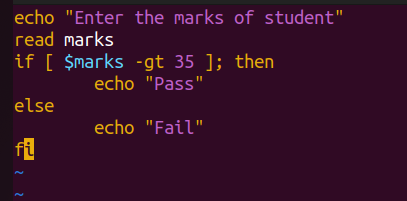
**Experiment No. 2**

1. Write a script to enter marks by user in the terminal. If the entered mark is greater than 35 then output pass, otherwise output fail.

**Output:**

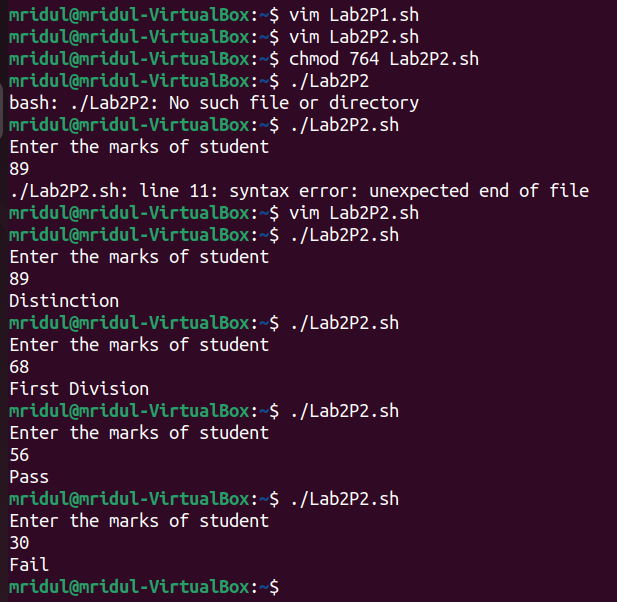


**Code:**

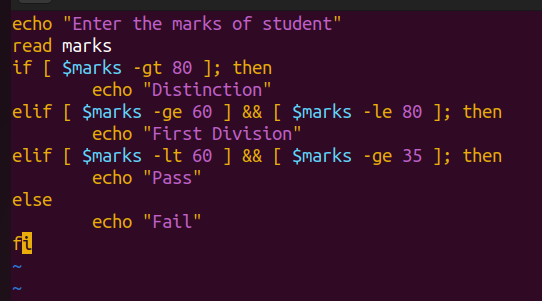


2. Write a script to enter marks by user in the terminal. If marks are greater than 80output distinction, if marks between 60 and 80 then output first division. If marks are less than 60 and greater than or equal to 35 then output pass, otherwise output fail.

**OUTPUT:**

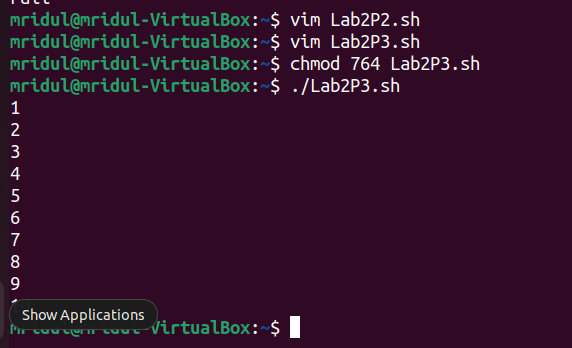


**CODE:**

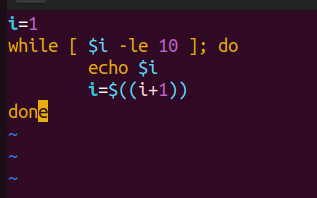


3. Write a script to print numbers from 1 to 10 using while loop.

**OUTPUT:**

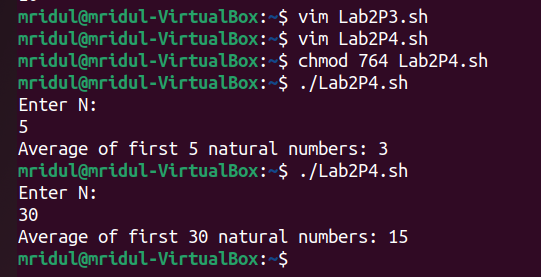


**CODE:**

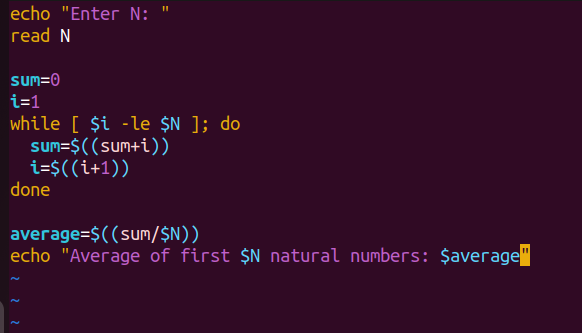


4. Write a script to find the average of first N natural numbers, where N is entered by the user at the terminal.

**OUTPUT:**

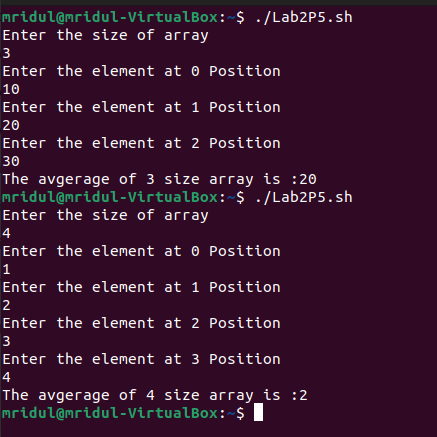


**CODE:**

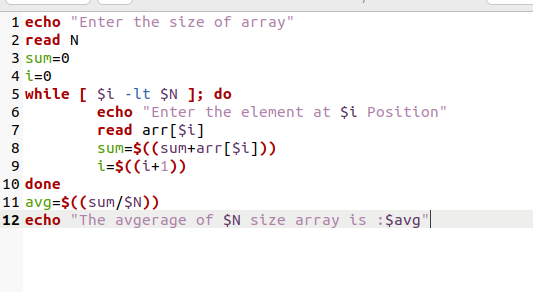


5. Write a script to find the average of N numbers of an array (of size N). The values of the array are entered by the user at the terminal. (Use while loop)

**OUTPUT**



**CODE**

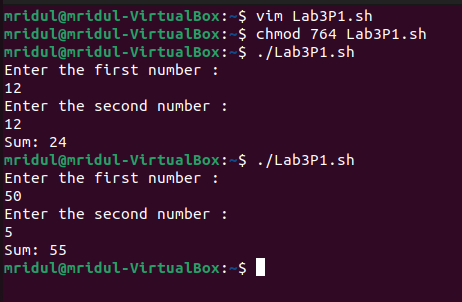


**Development Automation and Linux**

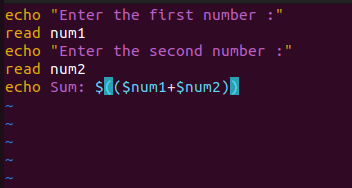
**LAB 3**

1. Write a shell script to fetch two command line arguments (Integers) and then add them and display the result.

Output:

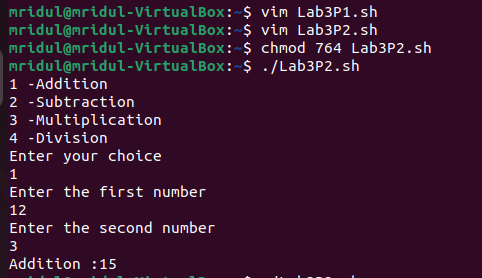


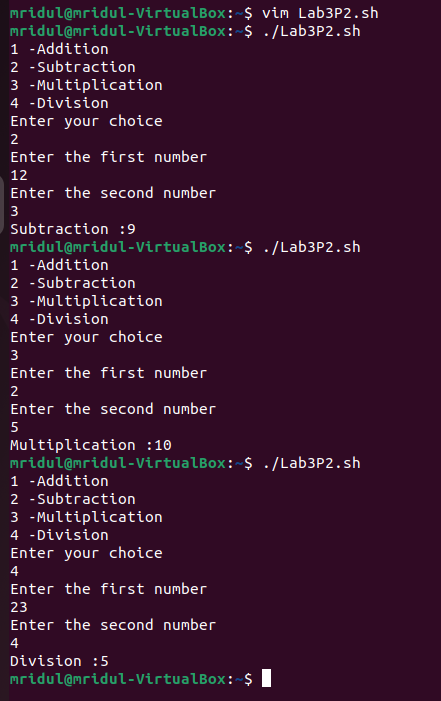
Code:



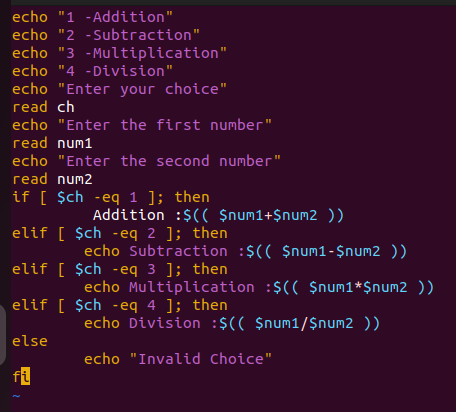
1. Write a shell script to implement a menu driven calculator with basic functionalities of addition, subtraction, division and multiplication. Your script will read two number from users and an operation and display the result

OUTPUT:



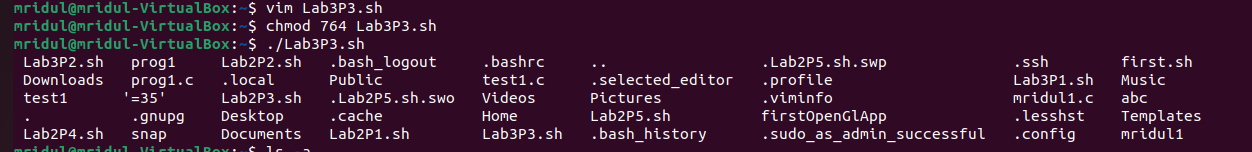


Code:



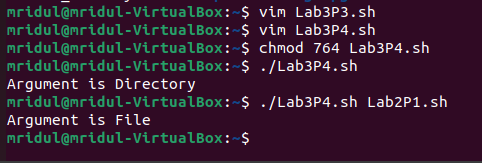
1. Write a shell script that prints all the ordinary files in the current working directory.

OUTPUT:

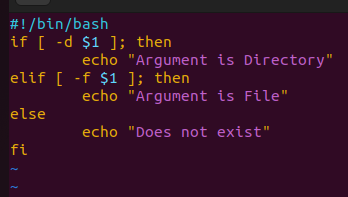


1. Write a shell script that takes a command line argument (the argument entered by the user is the absolute path of any file, directory etc.) and reports on whether it is a directory, a file, or something else.

OUTPUT:

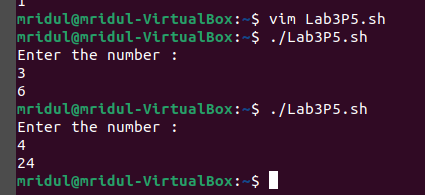


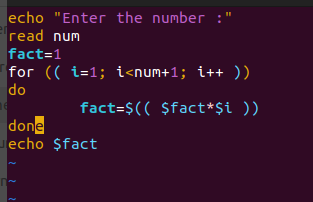
CODE:



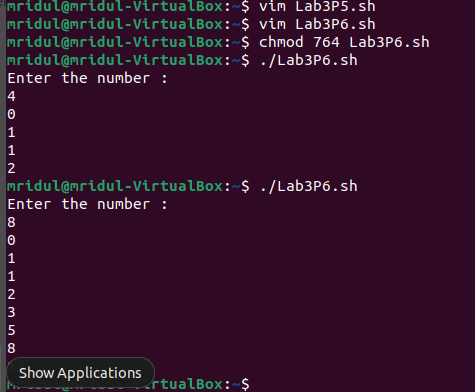
1. Write a shell script to compute factorial of a number entered by the user

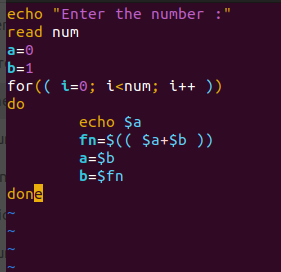
OUTPUT:



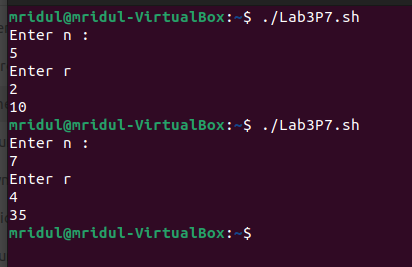


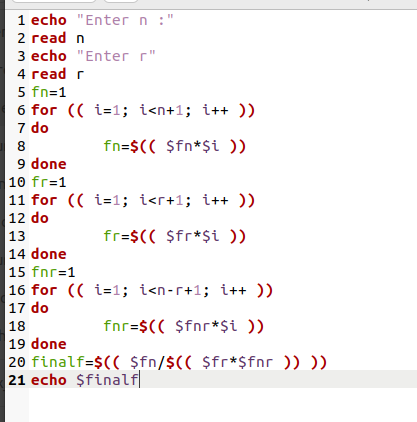
6. Write a shell script to generate Fibonacci series for a positive integer N which is entered by the user.



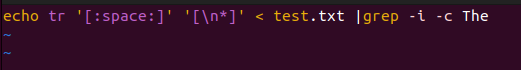


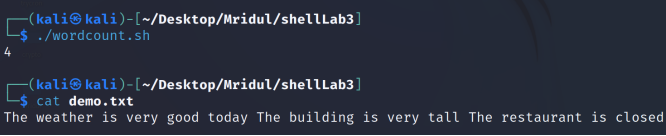
7. Write a shell script to print the value of nCk (read as n choose k), where n and k are entered by the user in the terminal. For example 5C2 is 10 and 10C3 is 120.





8. Write a shell script to find out the total instances of “the” in a given file.



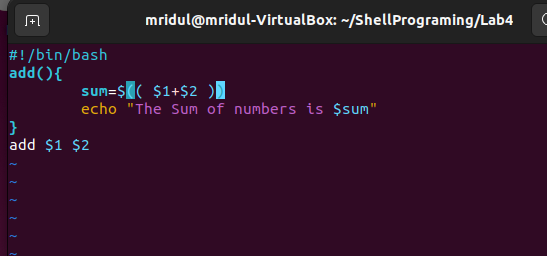


**Experiment No 4**

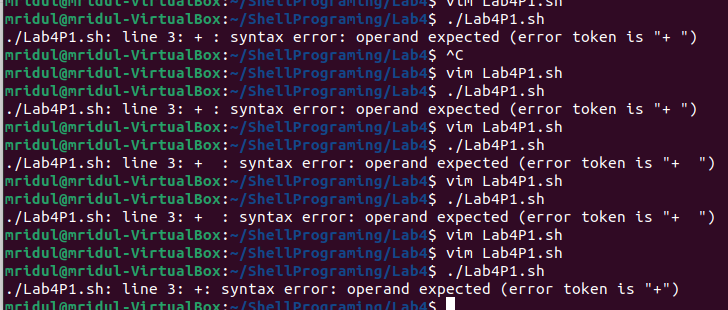
**Functions in Shell Scripting**

1. Write a function to fetch two command line arguments (Integers) and then add them and display the result.

CODE:

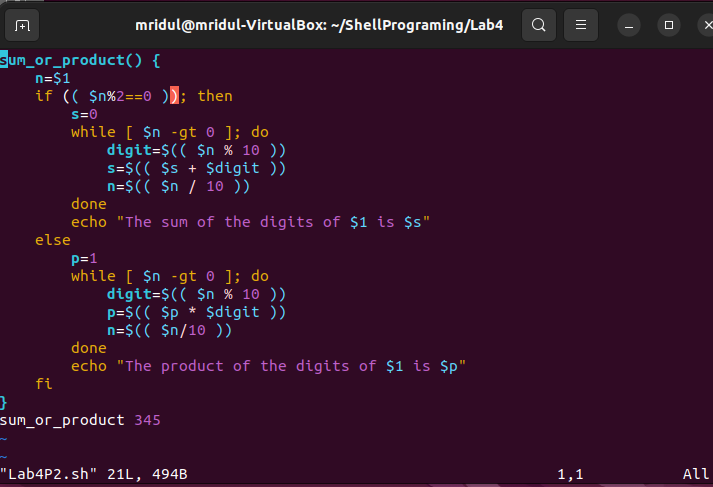


OUTPUT: ((Error is coming))

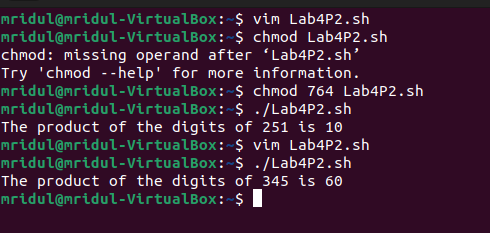


1. Write a function to input a three-digit number from the user and check if the number is even or odd. If the number is even then return the sum of the digits of the number, otherwise return the product of the digits of the given number.

CODE:

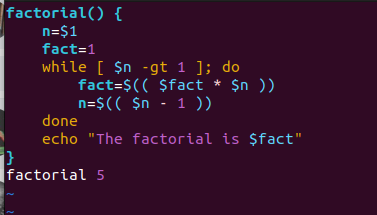


OUTPUT:

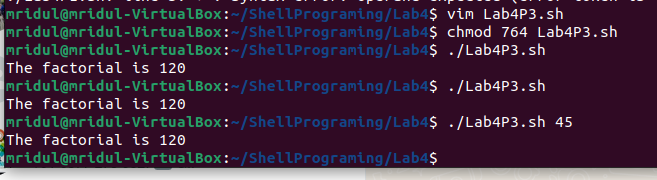


1. Write a function to compute the factorial of a given number N which is input by the user from the terminal

CODE:

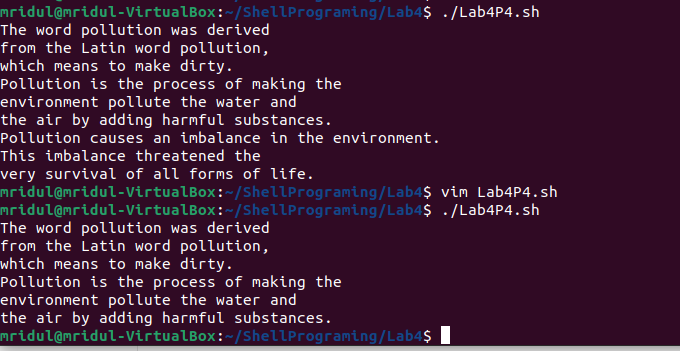


OUTPUT:

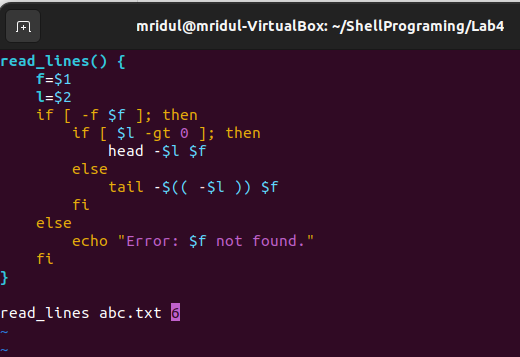


1. Write a function to read the number of lines of a given text file (starting from line 1) as specified by the user. For example, if there is a file demo.txt, and user specifies that he needs first 7 lines of the text then display first 7 lines of demo.txt. Similarly, if the user specifies last 10 lines, then first 10 lines of demo.txt should be printed on the terminal.

OUTPUT:



CODE:



**Experiment No 5**

**Automation Scripts that save Time and Effort**

1. Set the crontab to create text file every minute in a given directory. Note that the name of the files should be different. However, you may choose any name file name.

CODE:

\* \* \* \* \* touch ~/desktop/temp/`date +%Y%m%d%H%M%S`.txt

2. Write a shell script to archive all the created files (in above question 1) in the given directory. Move the archive to a given specified backup directory (different from the given directory). Insert this script as a cronjob using the crontab command, which needs to be run every 5 minutes.

CODE:

#!/bin/bash

$dir=~/desktop/temp

$backup=~/desktop/backup

$files = `ls $dir`

for file in $files

do

tar -cvf $backup/$file.tar $dir/$file

done

1. \* \* \* \* ~/desktop/temp/backup.sh

3. Write a shell script to automatically delete the archive files (in the backup archive) that are older than 30 minutes.

#!/bin/bash

$dir=~/desktop/backup

$files=`ls $dir`

for file in $files

do

if [ `find $dir/$file -mmin +30` ]

then

rm $dir/$file

fi

done

4. Monitor the disk usage of the given disk and alert if it is beyond the given threshold.

#!/bin/bash

df -Ph | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5,$1 }' | while read output;

do

# echo $output

used=$(echo $output | awk '{print $1}' | sed s/%//g)

partition=$(echo $output | awk '{print $2}')

if [ $used -ge 60 ]; then

echo "The partition \"$partition\" has been $used% used"

else

echo "Disk space usage of partition \"$partition\" is in under threshold"

fi

done

5. Write a shell script to install the LAMP stack on Linux Machine (A-Apache, M-Mysql, P-PHP).

CODE:

#!/bin/bash

echo -e "\n\nUpdating Apt Packages and upgrading latest patches\n"

sudo apt-get update -y && sudo apt-get upgrade -y

echo -e "\n\nInstalling Apache2 Web server\n"

sudo apt-get install apache2 apache2-doc apache2-mpm-prefork apache2-utils libexpat1 ssl-cert -y

echo -e "\n\nInstalling PHP & Requirements\n"

sudo apt-get install libapache2-mod-php7.0 php7.0 php7.0-common php7.0-curl php7.0-dev php7.0-gd php-pear php7.0-mcrypt php7.0-mysql -y

echo -e "\n\nInstalling MySQL\n"

sudo apt-get install mysql-server mysql-client -y

echo -e "\n\nPermissions for /var/www\n"

sudo chown -R www-data:www-data /var/www

echo -e "\n\n Permissions have been set\n"

echo -e "\n\nEnabling Modules\n"

sudo a2enmod rewrite

sudo phpenmod mcrypt

echo -e "\n\nRestarting Apache\n"

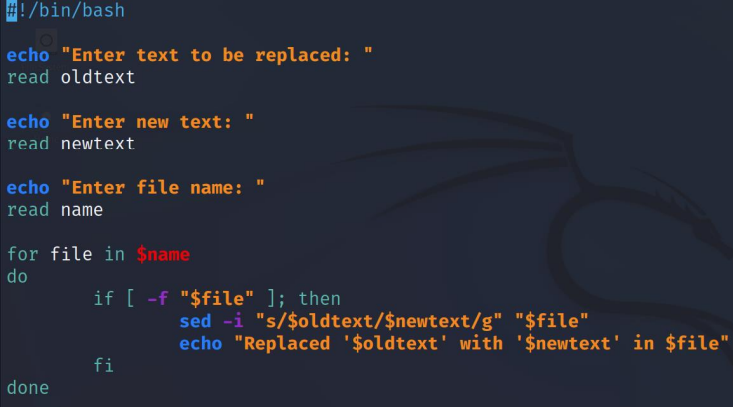
sudo service apache2 restart

echo -e "\n\nLAMP Installation Completed"

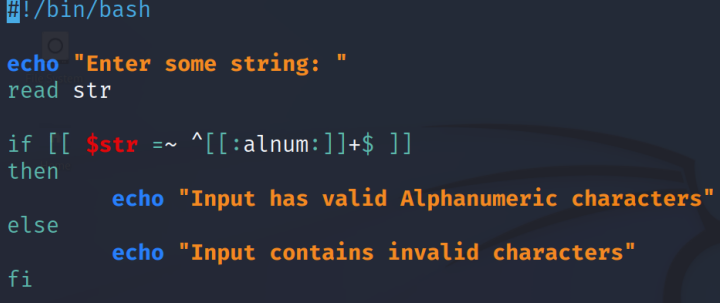
exit 0

**Experiment – 6**

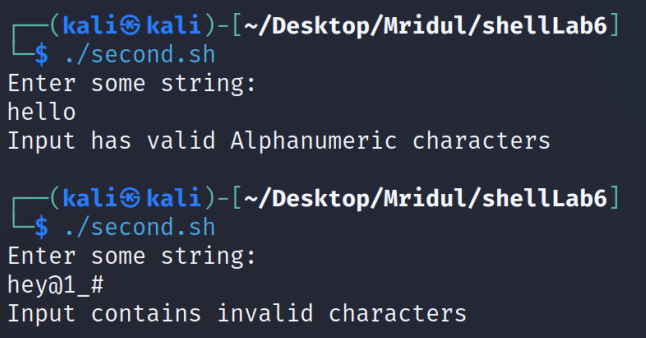
1. Write a shell script to search and replace a given text input by the user with a new text across multiple files.



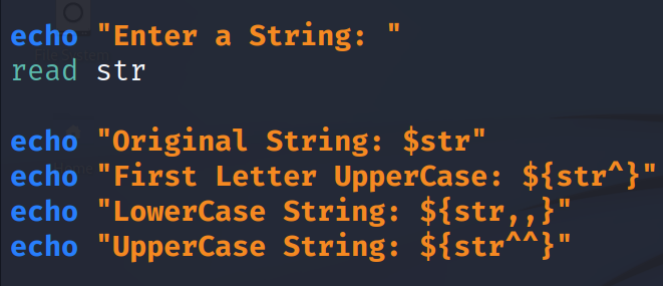
1. Write a shell script to check whether the given input is a valid Alphanumeric character.



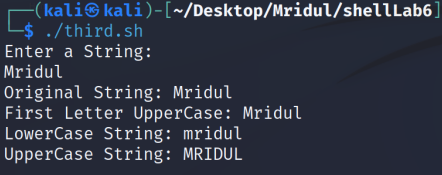
Output



1. Write a shell script to transform the given input text to lowercase, uppercase and first letter uppercase.



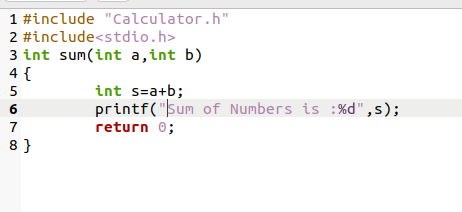
OUTPUT:

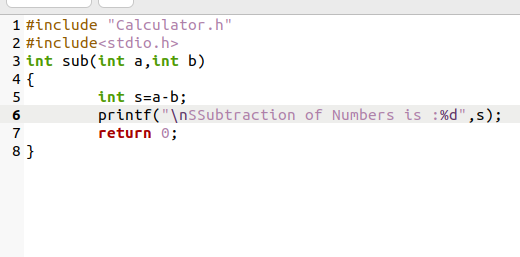


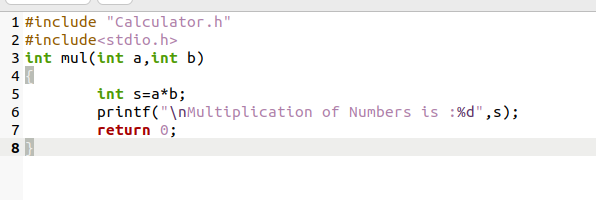
**Experiment No 7**

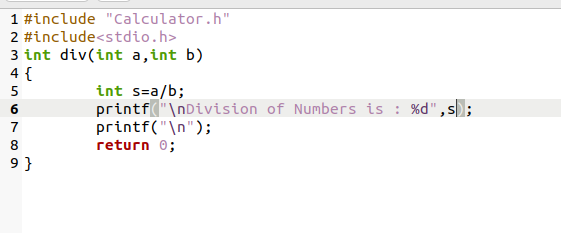
**Working with Make and Make file**

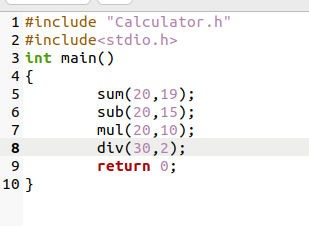
1. Write a C program for calculator with basic functionality subtraction, addition, division and multiplication. Each functionality (addition, subtraction, etc.,) should be written in separate source files along with a single main source file. Use make utility to create an executable file for the calculator.

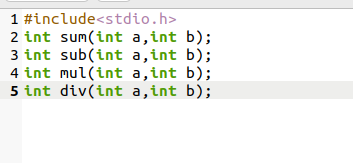


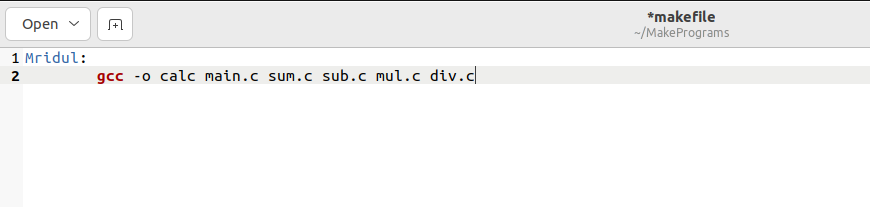


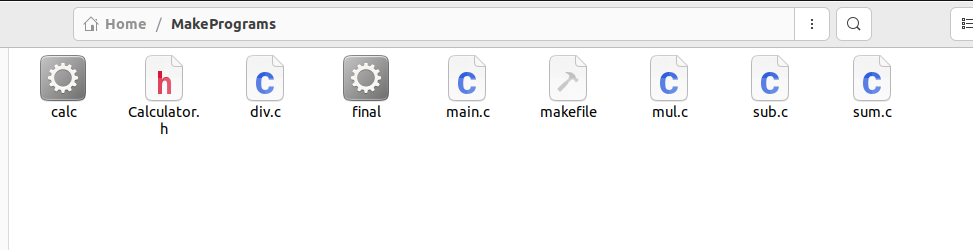




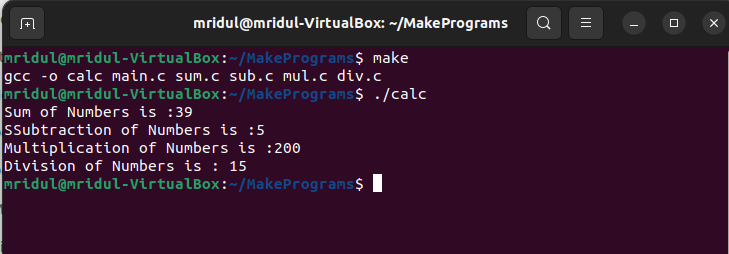




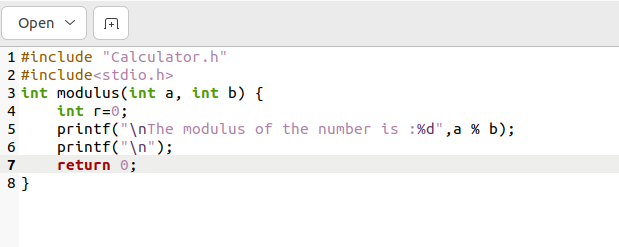


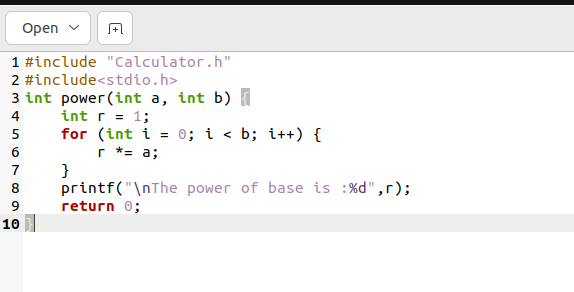


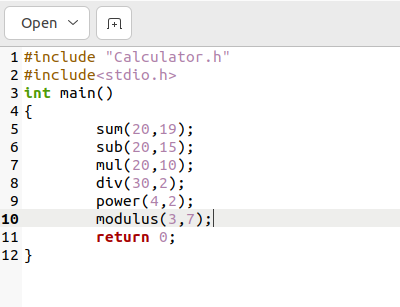
Output:

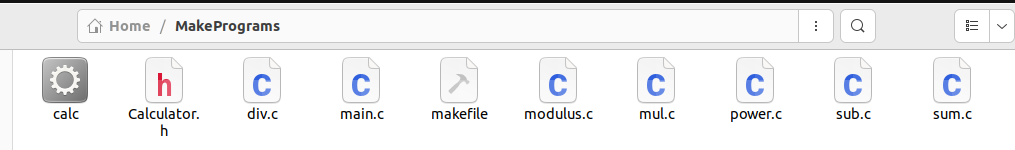


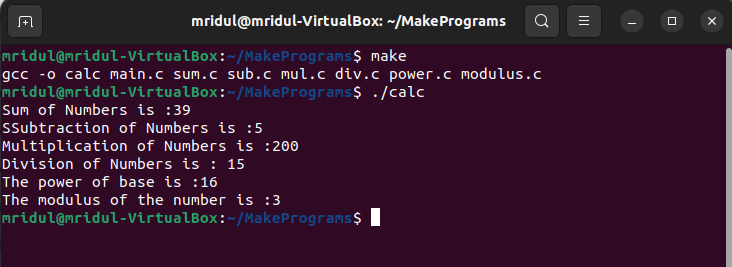
1. In the above program include two additional features for computing power and modulus of given numbers in the calculator program. Use Make utility to automatically build the second version of the calculator with additional features included in it.



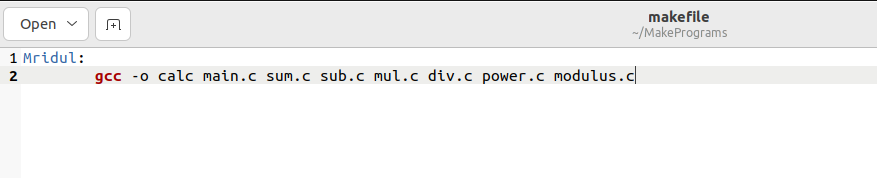




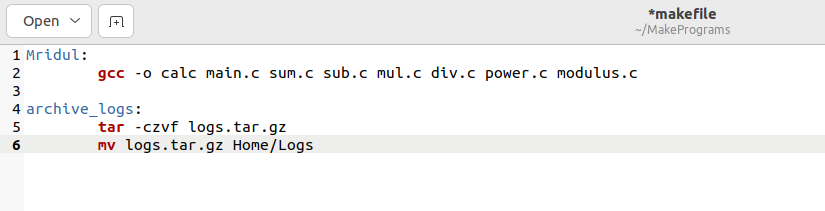




1. Write Make file in above program using the best practices (You may take help of the Lab manual given by Xebia)



1. Use make utility to archive the log files and move it to some other location.



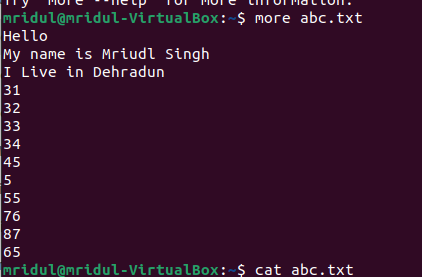
1. Demonstrate the usage of conditional statements in a Make file for a simple C/shell script program.
2. a=10
3. b=20
5. #Check whether they are equal
6. **if** [ $a == $b ]
7. then
8. echo "a is equal to b"
9. fi
11. #Check whether they are not equal
12. **if** [ $a != $b ]
13. then
14. echo "a is not equal to b"
15. fi

**Lab Experiment 8**

**File System Commands and Directory**

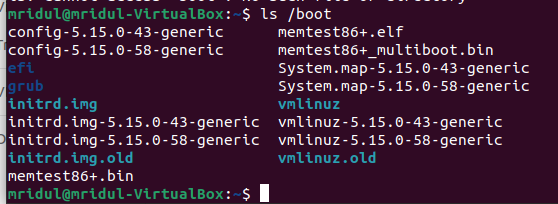
* **More**

As 'cat' command displays the file content. Same way 'more' command also displays the content of a file. Only difference is that, in case of larger files, 'cat' command output will scroll off your screen while 'more' command displays output one screenful at a time.



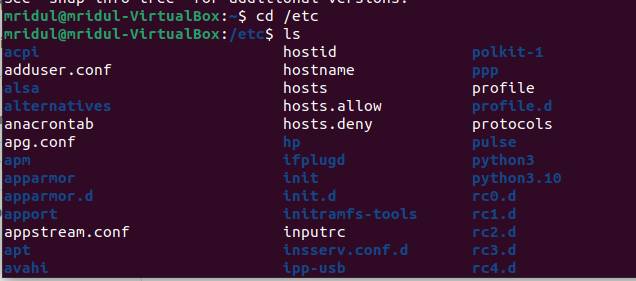
* **/boot directory**

**The boot directory contains everything required for the boot process.**

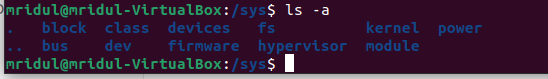
****

* **/etc directory**

**It contains all the system related configuration files.**

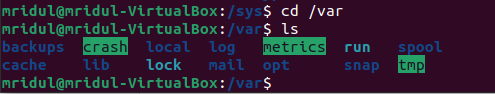
****

* **The /srv, /sys and /tmp directories**

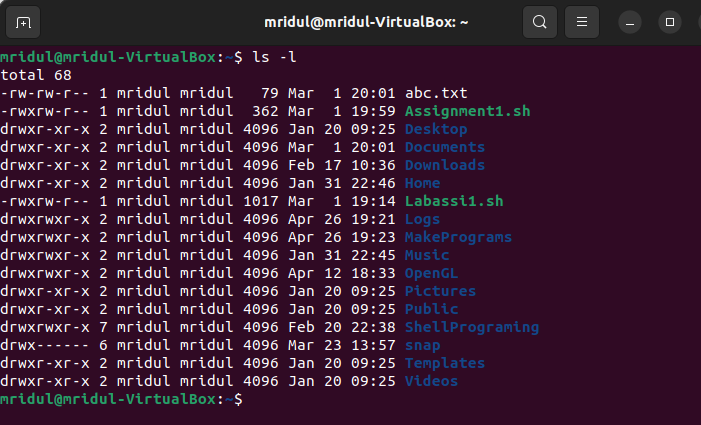
****

* **The /var directory**

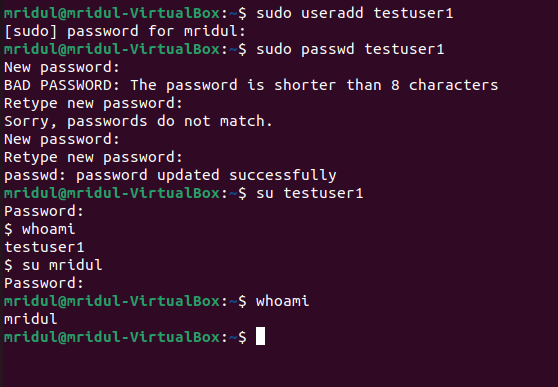
The /var (short for variable data) hierarchy contains files to which the system writes data during the course of its operation.



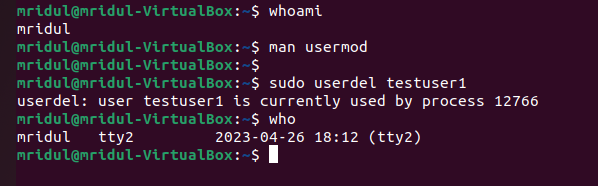
* **Ls -l**

****

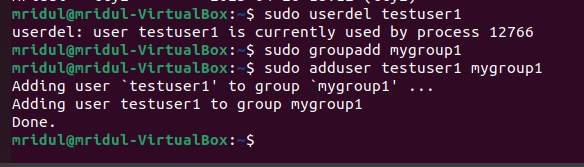
**Add a user**

****

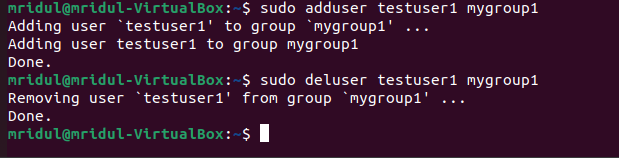
**Delete User**

****

**Add Group**

****

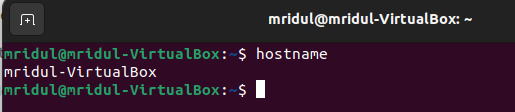
**Delete user from group**



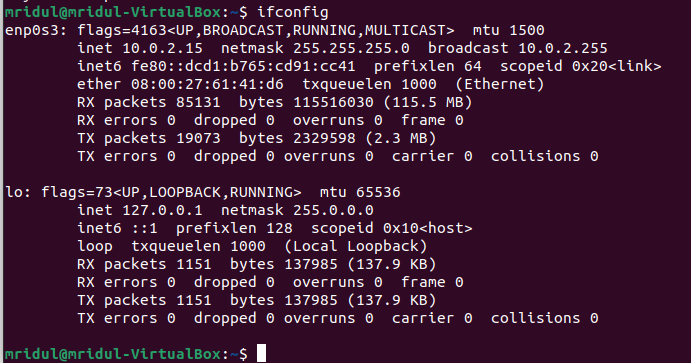
**Experiment 9**

**Networking Commands**

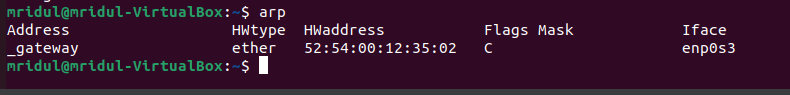
Hostname



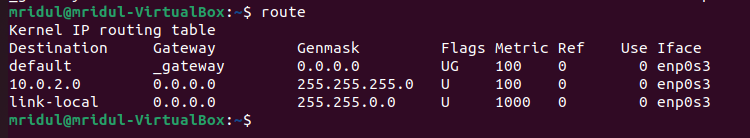
Ifconfig



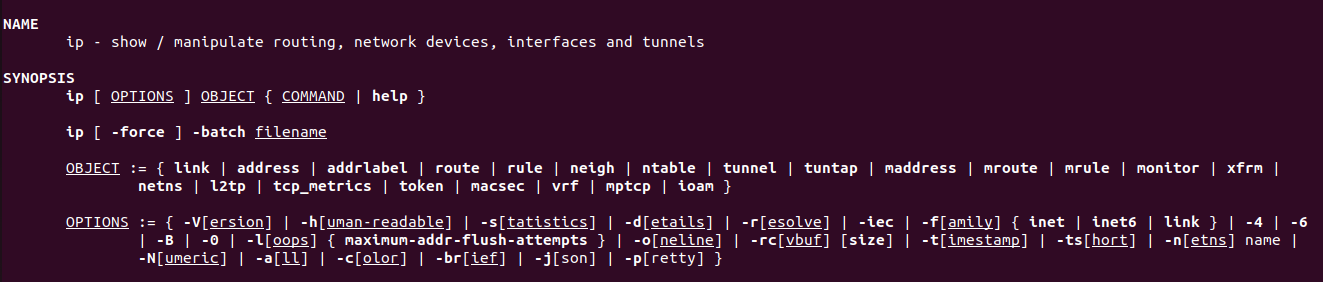
ARP



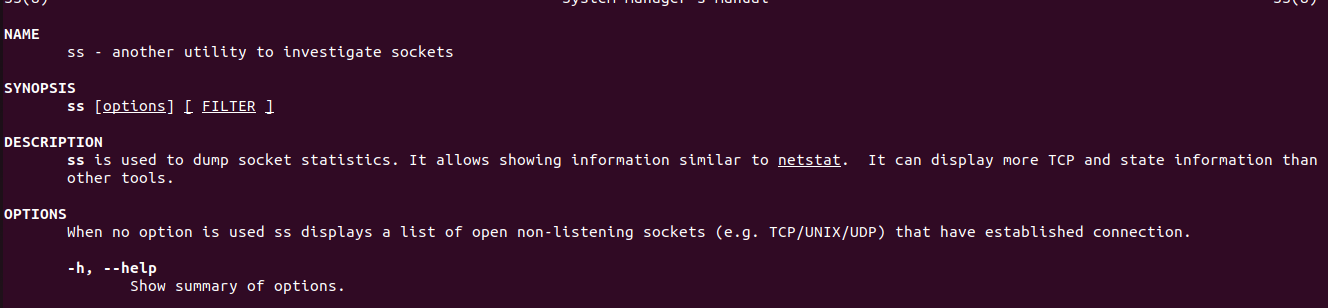
Route

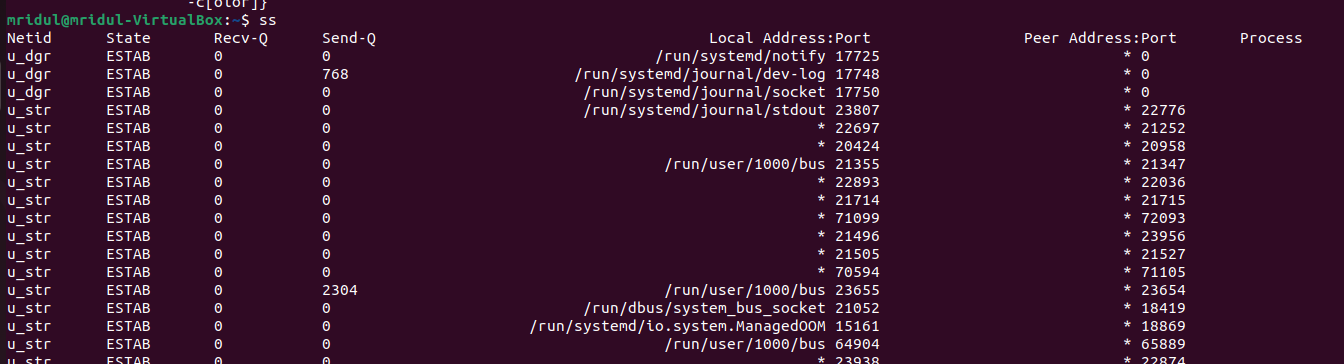


IP

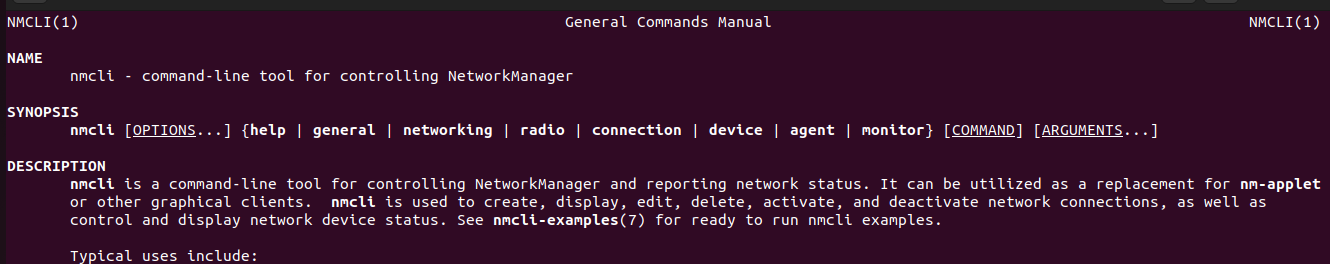


SS





Nmcli





**Thank You**