

IT APPLICATION and data SECURITY

**Subject Code:-BCSE 0601**

**LAB REPORT FILE**

**Submitted to : Submitted by:-**

**Dr. Premnarayan Arya Shikhar Gupta**

**(Asst. Professor) Section :-2AH**

**Department of C.E.A. Roll Number:-2215300033[28]**

**GLA UNIVERSITY-281406**

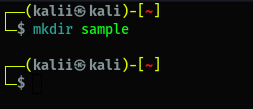
**OS Lab**

**Question 1:**

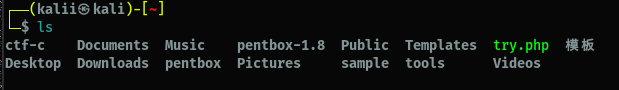
**Aim : Implement the following basic commands used in LINUX UNIX OS ( Also perform all the commands with their switches as assignment).**

**ls, man, pwd, who, whoami, date, cal, mkdir, rm, rmdir, cat, head, tail, more, less, cp, mv, echo.**

* **Make Directory :** 
  + **Command -**  mkdir [Directory name]
  + **Description -**  It creates a new directory(folder) with provided name.



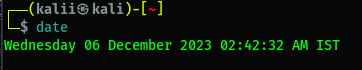
* **List files/directory :** 
  + **Command –** ls
  + **Description –** It list all contents within a directory.



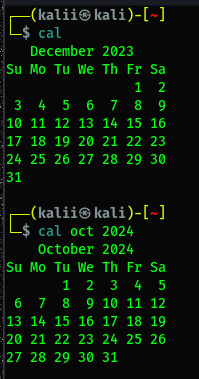
* **Remove Directory :** 
  + **Command –** rmdir [Directory name]
  + **Description –** It removes the directory with provided name.



* **Get current date & time :** 
  + **Command –** date
  + **Description –** It prints current date and time on screen.

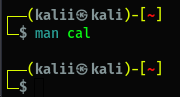


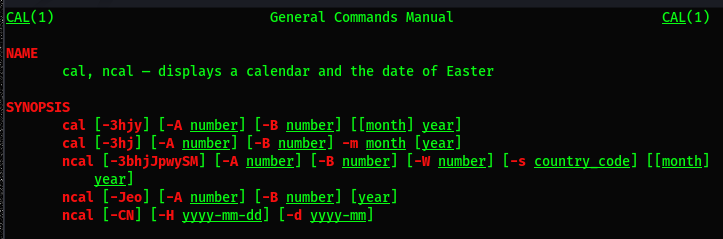
* **Get calendar :** 
  + **Command –** cal <month> <year>
  + **Description –** It prints calendar of current month or given month/year on screen.



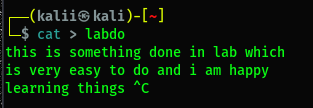


* **Open manual of any command :** 
  + **Command –** man [command]
  + **Description –** It shows the manual of any command which contains all info about that command.

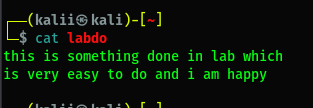




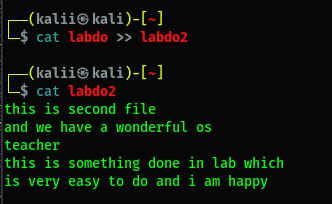
* **Create and write in a file :** 
  + **Command –** cat > [file name]
  + **Description –** It creates a file with a name given and write data in file from std. input till user press ctrl + d.



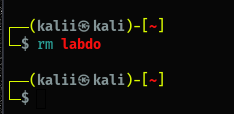
* **Read a file :** 
  + **Command –** cat [file name]
  + **Description –** It prints the file content on terminal.



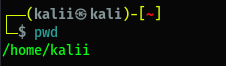
* **Concatinate a file :** 
  + **Command –** cat [file1] >> [file2]
  + **Description –** Cat also refers to concatenate. It concatenate file1 data to file2. ( >> refers to append , > refers to overwrite).



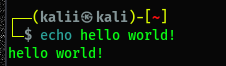
* **Remove a existing file in a directory:** 
  + **Command –** rm [file name]
  + **Description –** It removes the file whose name or path is given.( -r for recursive deletion used to delete a directory).



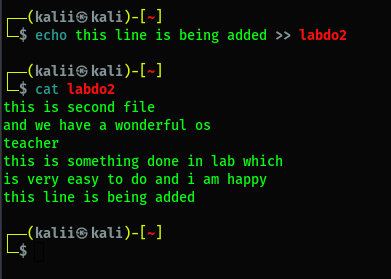
* **Get path of working directory :** 
  + **Command –** pwd
  + **Description –** It gives the path of current working directory.



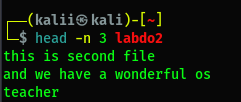
* **Print data :** 
  + **Command –** echo <data>
  + **Description –** It simply print the value of given string or variable on screen.



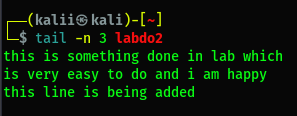
* **Add data :** 
  + **Command –** echo <data> >> [filename]
  + **Description –** It append the given data to the filename given.



* **Get data from starting :** 
  + **Command –** head -n <no. of lines> [filename]
  + **Description –** It prints the data of file from starting upto the given no. of lines (By default 10).



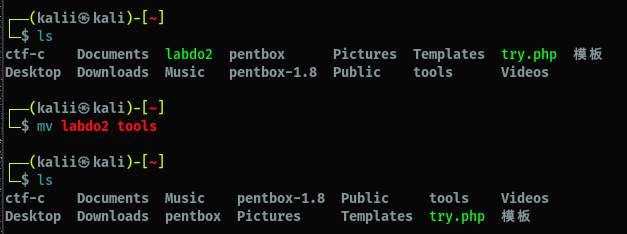
* **Get data from ending :** 
  + **Command –** tail -n <no. of lines> [filename]
  + **Description –** It prints the data of file from last upto the given no. of lines (By default 10).



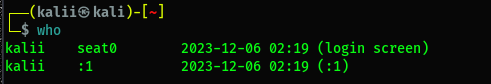
* **Copy :** 
  + **Command –** cp [source path] [destination path]
  + **Description –** It copy the file from source path to desired path.



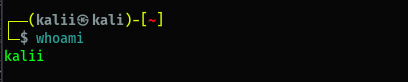
* **Move :** 
  + **Command –** mv [source path] [destination path]
  + **Description –** It move the file from source path to desired path. Also it can be use to rename files.



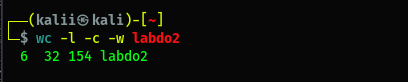
* **List of users :** 
  + **Command –** who
  + **Description –** It display all the users logged in the current session of machine.



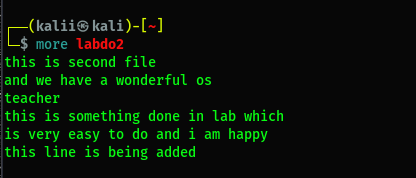
* **Username :** 
  + **Command –** whoami
  + **Description –** It display the name of current user account.



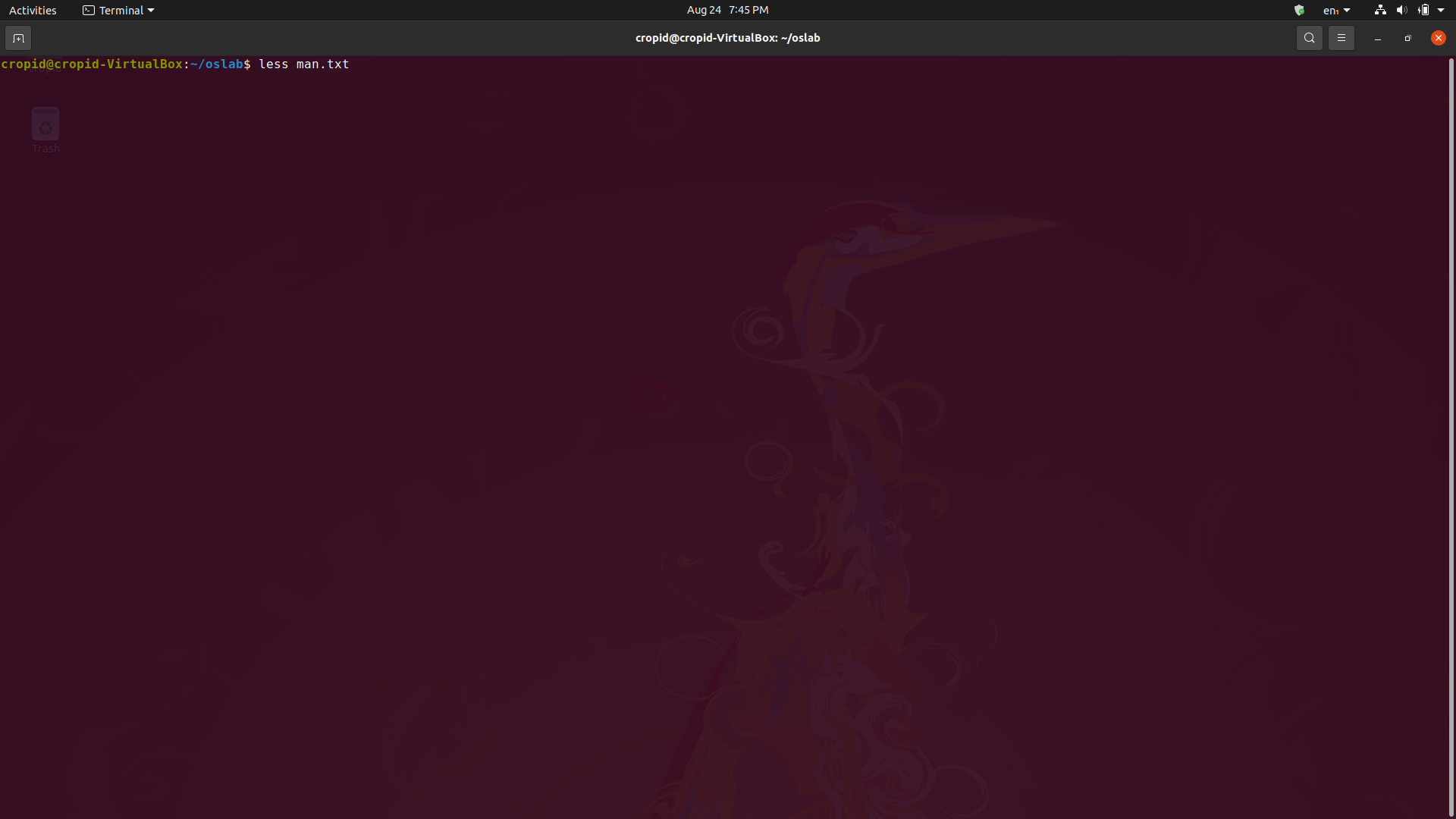
* **Count data :** 
  + **Command –** wc -l -w -c [filename]
  + **Description –** It prints the count of lines, words, and characters in a file or data. (-l for line, -w for words, -c for characters).

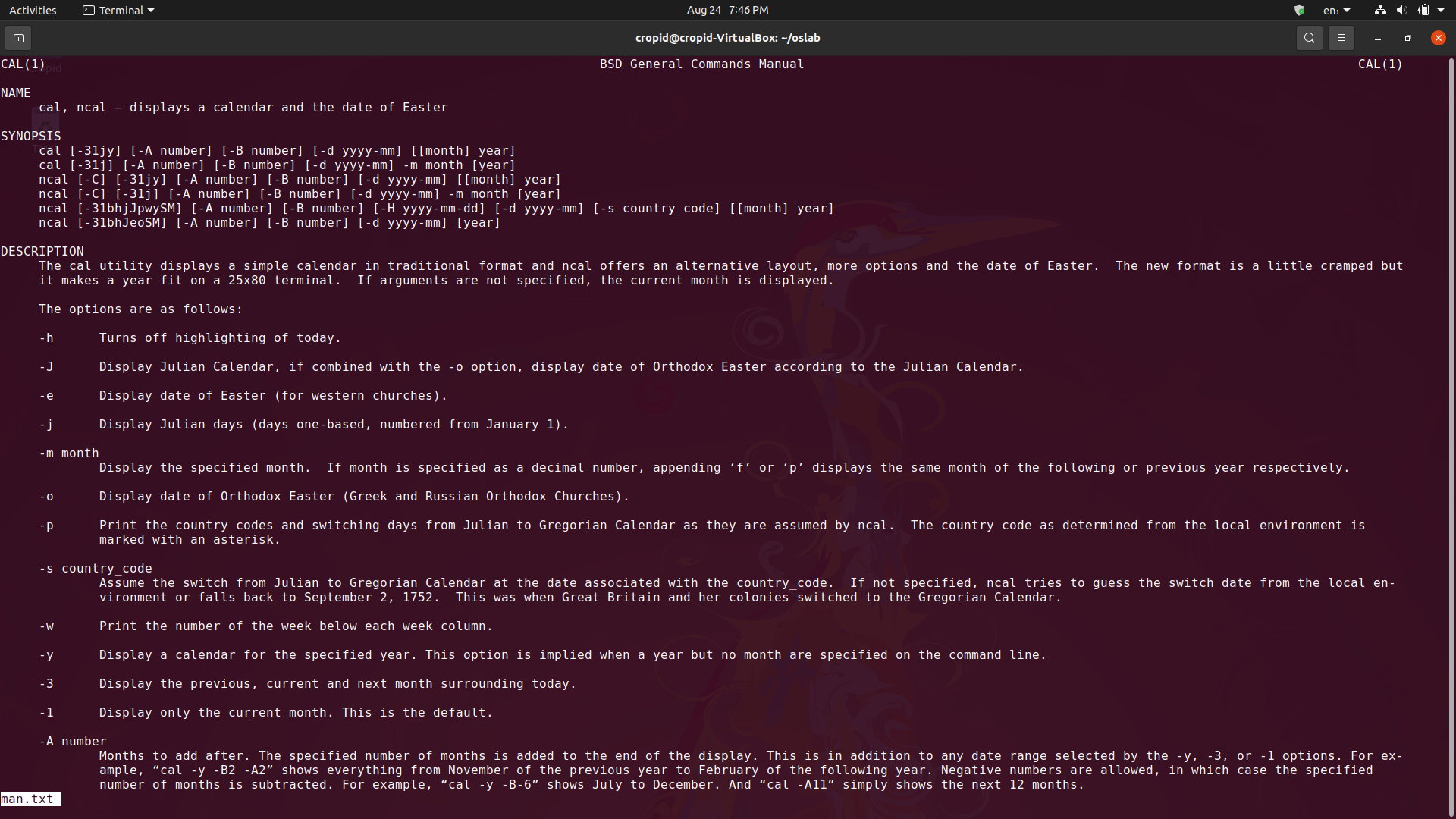


* **Use of more :** 
  + **Command –** more [filename]
  + **Description –** It prints the data till screen gets full and further can be viewed by space key.

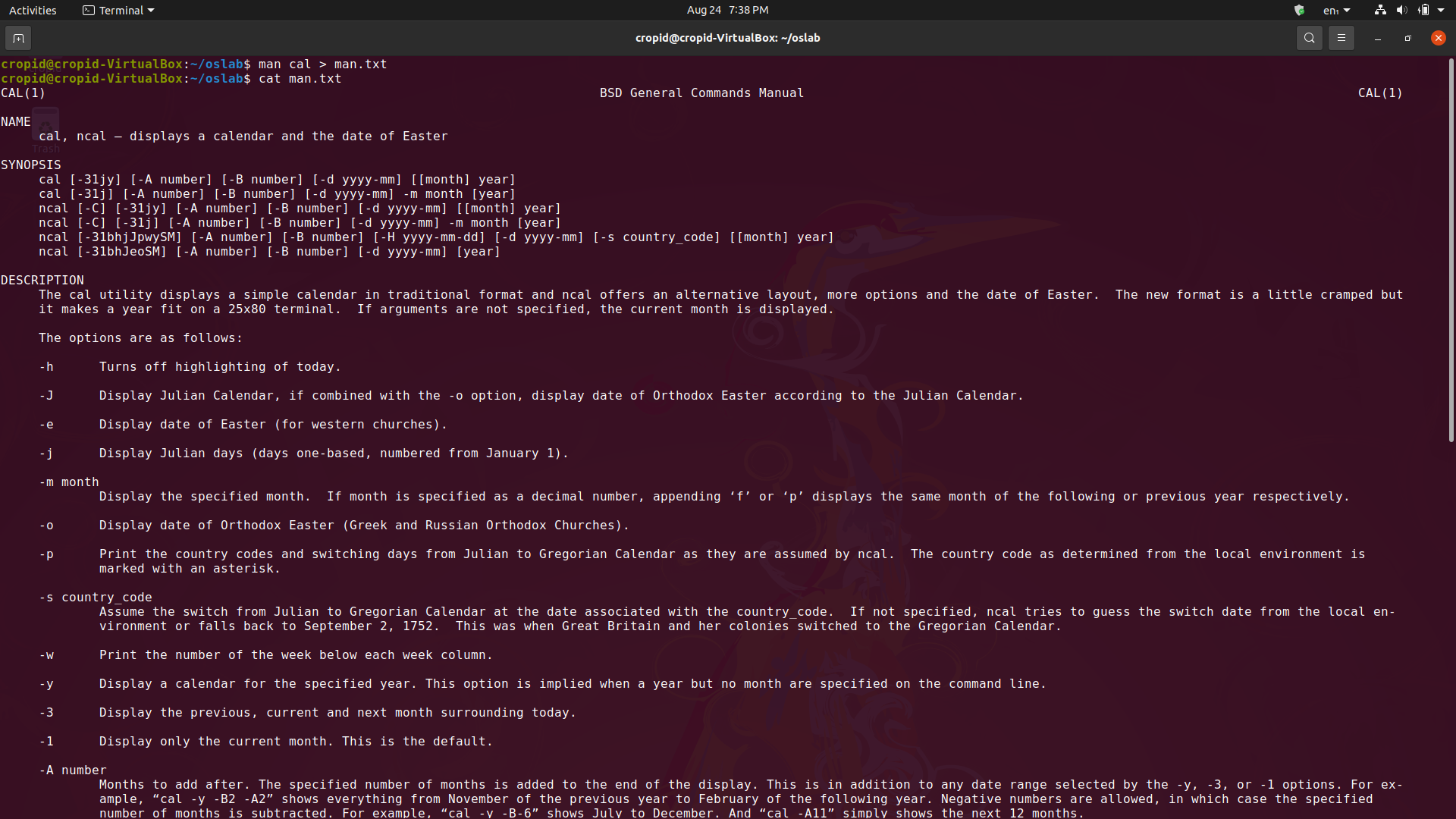


* **Use of less:** 
  + **Command –** less [filename]
  + **Description –** It prints the data till screen gets full and further can be viewed by forward(f key) and backward(b key).





* **Output to a file :** 
  + **Command –** command > [filename]
  + **Description –** It redirect the output of command in the file. (> is used for redirection).



**Questions based on queries**

1. WAC to display the current users. ---

Answer: who

1. WAC to display current login name. ---

Answer: whoami

1. to take backup of a file. ---

Answer: cp filename<destination>

1. to create 3 sub directories into a directory using single line command ---

Answer: mkdir dir1 dir2 dir3

1. how to remove file of .txt extn. ---

Answer: rm\*.txt

1. how to remove file of any extn. ---

Answer: rm\*.\*

1. how to remove file starting with letter 'a' only.

Answer: rm a\*

8. how to remove non empty dirc.

Answer: rm -r your\_directory

**Experiment 2(A) :**

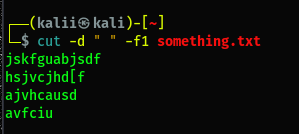
**AIM : Implement the following basic commands used in LINUX UNIX OS ( Also perform all the commands with their switches).**

**Cut, chmod, du, diff, uniq, history, last, sort, sed, grep.**

* **Cut :** 
  + **Command -**  cut -d ”<single character>” -f n [file name]
  + **Description -**  It is used to display data columnwise.

-d “ ” : used to pass delimiter from which data has to be separated into columns passed in “ ” or ‘ ’.

-f n : used to pass field number to access column data.



* **Chmod :** 
  + **Command -**  chmod +rwx [file name] or chmod 777 [file name]
  + **Description -**  It modify the permissions for different users of a file.

+r or 4: used for enabling readable permissions.

+w or 2: used for enabling writeable permissions.

+x or 1: used for enabling executable permissions.

A screenshot of a video game

Description automatically generated

* **Du :** 
  + **Command -**  du [file name]
  + **Description -**  It estimate file disk usage.

-a : List all files and directories size.

A screenshot of a video game

Description automatically generated

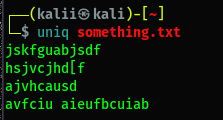
* **Diff :** 
  + **Command -**  diff <options> [file name1] [file name2]
  + **Description -**  Compare files line by line and output different lines.

-a : Take all data of files as text.

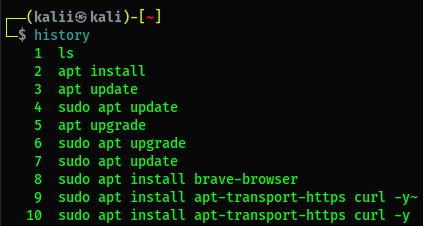
A screenshot of a computer

Description automatically generated with medium confidence

* **Uniq :** 
  + **Command -**  uniq [file name]
  + **Description -**  It filters out the repeated lines in a file.

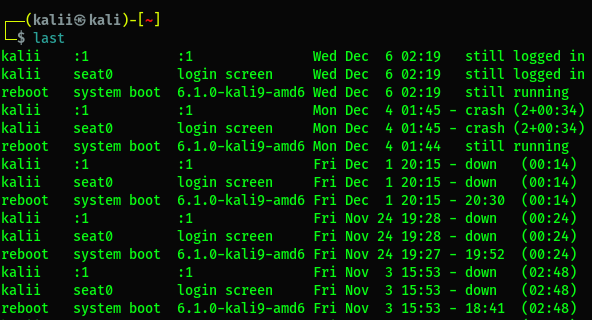


* **History :** 
  + **Command -**  history
  + **Description -**  Used to display previously executed commands.



* **Last :** 
  + **Command -**  last <options>
  + **Description -**  It list all users logged in and out.

-x : It display info of system entries and run level changes.



* **Sort :** 
  + **Command -**  sort <options> [file name]
  + **Description -**  It outputs the data of a file in a sorted manner.

-k n : Used to specify sorting sequence(using which column data is sorted).

-r : Used to sort in descending order.

A screenshot of a computer

Description automatically generated with medium confidence

* **Sed :** 
  + **Command -**  sed <options> [file]
  + **Description -**  Stream editor.
* **Grep :** 
  + **Command -**  grep [search string] [file name]
  + **Description -**  It search the given string in the provided data source and output the which match which passed string.

A screenshot of a computer

Description automatically generated with medium confidence

**Questions based on queries**

1. Total no of users connected to system currently

Answer: user wc -w

1. Display only current local time of system

Answer: date

1. Arranging the files of directory on ascending order of their sizes

Answer:ls -l sort

1. Display 5th character of your name

Answer: echo $name | cut -c 5

echo kartikeyan | cut -c 5

1. Display 7th line of a file

Answer: head -n 7 filename | tail -n 1

1. Sort a file on second column basis

Answer: sort -t',' -k2,2 filename.csv

**Question2(B):**

**AIM : Implement the special shell scripting variables such as $$,$0,$1,$\*,$@,$ etc.**

**ShellScript :**

#! /bin/sh

echo "\$1 = $1"

echo "\$2 = $2"

echo "\$3 = $3"

echo "\$0 = $0"

echo "\$# = $#"

echo "\$\* = $\*"

echo "\$@ = $@"

A screenshot of a computer

Description automatically generated with medium confidence

**Question 3 (a):**

**AIM : Write a shell script to display current date in a particular format, number of users currently login and current month’s calendar.**

**ShellScript :**

#! /bin/sh

echo "Date :"

date | cut -d" " -f1-3

echo "\nNo. of user :"

who | wc -l

echo "\nMonth :\n"

cal

**Question3 (B):**

**AIM : Write a shell script to display the process name and its process id.**

**ShellScript:**

#! /bin/sh

echo "Process Name : $0"

echo "Process Id : $$"

**Question3 (C):**

**AIM : Write a shell script to take name as a input and display a greeting message to the user by checking system clock.**

**ShellScript :**

#! /bin/sh

echo "Enter your Name :"

read n

t= date +%H

echo "Hello $n Good"

if [ $t < 12 ]

then

echo "Morning"

elif [ $t < 16 ]

then

echo "Afternoon"

else

echo "Evening"

fi

**Question 3(D):**

**AIM : Write a shell script to merge the content of 2 files into one file.**

**ShellScript :**

#! /bin/sh

echo "Enter first file name :"

read f1

echo "Enter second file name :"

read f2

echo "Enter third file name :"

read f3

cat < $f1 >> $f3

cat < $f2 >> $f3

echo "Content in $f3 :"

cat $f3

**Question 3(E):**

**AIM : Write a shell script to create a tsv file containing name, roll no. and age of 10 students. Then use that tsv file to display only the names of the students in alphabetical order.**

**ShellScript :**

#! /bin/sh

echo "Enter file name :"

read f1

sort -k1 $f1 | cut -d" " -f1

**AIM : You are given a file of tab-delimited weather data (TSV). There is no header column in this data file. The first five columns of this data are: (a) the name of the city (b) the average monthly temperature in Jan (in Fahrenheit). (c) The average monthly temperature in April (in Fahrenheit). (d) The average monthly temperature in July (in Fahrenheit). (e) the average monthly temperature in October (in Fahrenheit).You need to sort this file on the basis of average monthly temperature in April.**

**ShellScript :**

#! /bin/sh

sort -k3 weatherdata.tsv

A screenshot of a computer

Description automatically generated with medium confidence

**Question4 (A):**

**AIM : Write a shell script that finds whether an entered number is even or odd.**

**ShellScript :**

#! /bin/sh

echo "Enter a no. "

read n

rem=`expr $n % 2`

if [ $rem -eq 0 ]

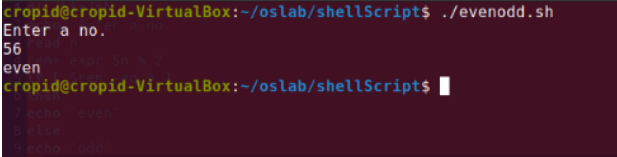
then

echo "even"

else

echo "odd"

fi



**Question4 (B):**

**AIM : Write a shell script to input the name of a file as command line argument and display whether it is a file, a directory or anything else.**

**ShellScript :**

#! /bin/sh

if [ -d $1 ]

then

echo "It is a Directory"

elif [ -f $1 ]

then

echo "It is a File"

else

echo "It is a String"

fi

A screenshot of a computer

Description automatically generated with medium confidence

**Question4 (C):**

**AIM : Write a shell script to input the marks of a student in 3 subjects and find his grade.**

**ShellScript :**

echo "Enter marks of 1st subject :"

read m1

echo "Enter marks of 2nd subject :"

read m2

echo "Enter marks of 3rd subject :"

read m3

sum=`expr $m1 + $m2 + $m3`

if [ $sum -eq 300 ]

then

echo "Grade O" elif [ $sum -ge 200 -a $sum -lt 300 ]

then echo "Grade A"

elif [ $sum -ge 100 -a $sum -lt 200 ]

then

echo "Grade B"

else

echo "Fail"

fi

**Question 4(D):**

**AIM : Write a Shell script to accept a filename as argument and displays the last modification time if the file exists and a suitable message if it doesn't.**

**ShellScript :**

echo -n "Enter file name : "

read f1

if [ -f"$f1" ];then

date -r $f1 | cut -d" " -f5

else

echo "File not exist"

fi

**Question 4(E):**

**AIM : Write the shell script to take file name as input and if the file exists then print the number of lines and also print 10th line of that file.**

**ShellScript :**

echo -n "Enter file name : "

read n

echo -n "No. of lines are : "

cat $n | wc -l

echo -n "10 line : "

head -10 $n | tail -1

**Question5(A):**

**AIM : Write a shell script that print multiplication table of a given no.**

**ShellScript :**

echo -n "Enter a number : "

read n

for ((i=1;i<=10;i++))

do

m=`expr $i'\*'$n`

echo "$n x $i = $m"

done

**Question 5(B):**

**AIM : Write a shell script to implement a timer.**

**ShellScript :**

echo -n "Enter seconds for timer : "

read t

for ((i=t;i>=0;i--))

do

clear

echo "Remainning time : $i second"

sleep 1

done

**Question 5(C):**

**AIM : Write a shell script that print reverse of a given positive number.**

**ShellScript :**

echo -n "Enter a positive integer :"

read n

rev=0

while [ $n -gt 0 ]

do

temp=$(( $n % 10 ))

rev=$(( $rev\*10 ))

rev=$(( $rev + $temp ))

n=$(( $n/10 ))

done

echo "Rev = $rev"

**Question 5(D):**

**AIM : Write a shell script that print factorial of a given number.**

**ShellScript :**

echo -n "Enter positive integer : "

read n

f=1

while [ $n -ne 1 ]

do

f=$(( $f \* $n))

n=$(( $n - 1))

done

echo "Factorial : $f"

**Question 6(A):**

**AIM : Write a shell script to input two strings from the user and determine whether they are same or not.**

**ShellScript :**

echo -n "Enter string1 : "

read s1

echo -n "Enter string2 : "

read s2

if [ "$s1" = "$s2" ]

then

echo "Equal"

else

echo "Not Equal"

fi

**Question 6(B):**

**AIM : Write a shell script to input a string from the user and determine its length.**

**ShellScript :**

echo -n "Enter a string : "

read s

c=$( echo $s | wc -c )

echo `expr $c - 1`

**Question 6(C):**

**AIM : Write a shell script to input two strings from the user and find the occurrences of string2 in**

**string 1.**

**ShellScript :**

echo -n "Enter string1 : "

read s1

echo -n "Enter string2 : "

read s2

echo "$s1" | grep -io "$s2" | wc –l

**Question 6(D):**

**AIM : Write a shell script to input the name of a file as command line argument and display the**

**number of characters, words and lines in the file.**

**ShellScript :**

echo -n "Characters : "

wc -c < $1

echo -n "Words : "

wc -w < $1

echo -n "Lines : "

wc -l < $1

**Question6 (E):**

**AIM : Write a shell script to display a list of directories within the current directory and how much**

**space they consume, sorted from the largest to the smallest.**

**ShellScript :**

echo -n "Enter directory name : "

read d

du $d | sort –nr

**Question6 (F):**

**AIM : Write a short script count txt to count the total number of .txt files in the current directory, and**

**print out this number to screen.**

**ShellScript :**

ls \*.txt | wc –w

**Question7**

**Briefly discuss and Implement following file structure related system calls:**

**Creat(), open(), close() read(),write()**

* **Creat()**

Command:- echo "This is a sample file." | creat example.txt

* **open()**

Command:- echo "This is another sample file." | open example.txt -e

* **close()**

Command:- exec 3<> example.txt

echo "This is a line appended to the file." >&3

exec 3>&-

* **read()**

Command:- fileDescriptor=$(open example.txt -e)

read -n 50 <&$fileDescriptor

echo "Read data: $REPLY"

close $fileDescriptor

* **write()**

Command:- fileDescriptor=$(open example.txt -e)

echo "This line will be written to the file." >&$fileDescriptor

close $fileDescriptor

**Question8**

**Briefly discuss and implement following file structure related system calls:**

**lseek(), dup(), link(), unlink(), stat(), fstat(), access(), chmod(), chown(), umask(), ioctl()**

Answer: - Below is a brief discussion of each file structure-related system call along with an example of how to use them in a Linux environment or with Bash commands:

**1. lseek()**

- lseek() is used to change the file offset of an open file.

**- Linux Example:**

```bash

# Using lseek in a C program

# Example C program: lseek\_example.c

gcc lseek\_example.c -o lseek\_example

./lseek\_example

```

**2. dup ()**

- dup () duplicates an existing file descriptor.

**- Linux Example:**

```bash

# Using dup in a C program

# Example C program: dup\_example.c

gcc dup\_example.c -o dup\_example

./dup\_example

```

**3. link ()**

- link () creates a new link (hard link) to an existing file.

**4. unlink ()**

- unlink () removes a link to a file.

- Linux Example:

# Using unlink in Bash

unlink example.txt

**5. stat ()**

- stat () retrieves file status.

- Linux Example:

# Using stat in Bash

stat example.txt

**6. fstat ()**

- fstat () is similar to `stat () ` but works with a file descriptor.

- Linux Example:

# Using fstat in a C program

# Example C program: fstat\_example.c

gcc fstat\_example.c -o fstat\_example

./fstat\_example

**7. access()**

- access() checks real user's permissions for a file.

- Linux Example:

```bash

# Using access in a C program

# Example C program: access\_example.c

gcc access\_example.c -o access\_example

./access\_example

```

**8. chmod()**

- chmod() changes the permissions of a file.

- Linux Example:

```bash

# Using chmod in Bash

chmod u+rwx file.txt

```

**9. chown ()**

- chown () changes the owner and group of a file.

- Linux Example:

```bash

# Using chown in Bash

chown newowner: newgroup file.txt

```

**10. umask()**

- umask() sets the default permissions for newly created files.

- Linux Example:

```bash

# Using umask in Bash

umask 022

```

**11. ioctl()**

- ioctl() is highly device-specific and may not have a general example. It is often used for controlling device-specific operations.

- Linux Example:

```bash

# Using ioctl in a C program (Note: Device-specific, so example depends on the device)

```