PROGRAM NO.1

**Demonstration of FORK() System Call**

#include<stdio.h>

#include<unistd.h>

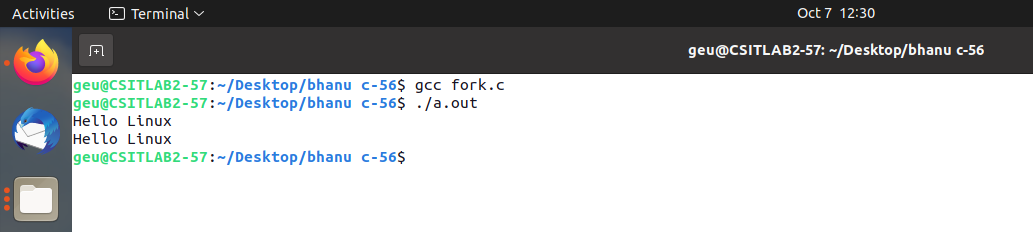
int main(){

fork();

printf("Hello Linux\n");

return 0;

}



**Q2.// MULTIPLE FORK**

#include<stdio.h>

#include<unistd.h>

int main(){

fork();

printf("FIRST\n");

fork();

printf("SECOND\n");

fork();

printf("THIRD\n");

return 0;

}



**3. Write a program using fork() system call, where parent process computes the sum of even numbers and child process computes the sum of odd numbers.**

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#define n 100

int main()

{

int pid;

int arr[n], sumodd = 0, sumeven = 0, size;

printf("Enter the size of array: ");

scanf("%d", &size);

printf("Enter Values in the array: ");

for (int i = 0; i < size; i++){

scanf("%d", &arr[i]);

}

pid = fork();

if (pid == 0){

for (int i = 0; i < size; i++)

if (arr[i] % 2 != 0)

sumodd += arr[i];

printf("Child process\n");

printf("Sum of odd numbers: %d\n", sumodd);

}

else if (pid > 0){

wait(NULL);

for (int i = 0; i < size; i++)

if (arr[i] % 2 == 0)

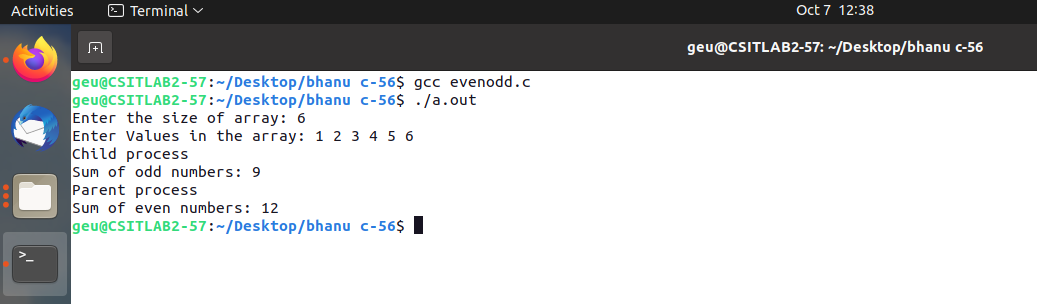
sumeven += arr[i];

printf("Parent process\n");

printf("Sum of even numbers: %d\n", sumeven);

}

}

****

**4. Write a program to implement the orphan process.**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main()

{

pid\_t pid;

pid = fork();

if (pid == 0){

sleep(5);

printf("I am child having PID %d\n", getpid());

printf("My parent PID is %d\n", getppid());

}

else if (pid > 0){

printf("I am parent having PID %d\n", getpid());

printf("My child PID is %d\n", pid);

}

else{

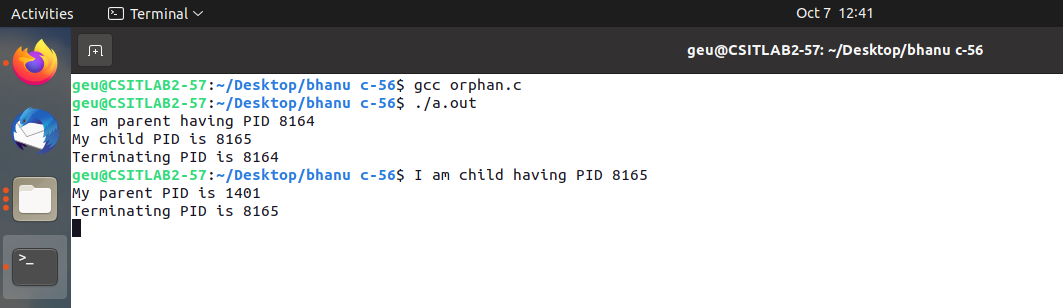
printf("process not created");

}

printf("Terminating PID is %d\n", getpid());

return 0;

}

****

**5. Write a program to implement a Zombie process.**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main()

{

pid\_t pid;

pid = fork();

if (pid == 0){

printf("I am child having PID %d\n", getpid());

printf("My parent PID is %d\n", getppid());

}

else if (pid > 0){

sleep(40);

printf("I am parent having PID %d\n", getpid());

printf("My child PID is %d\n", pid);

}

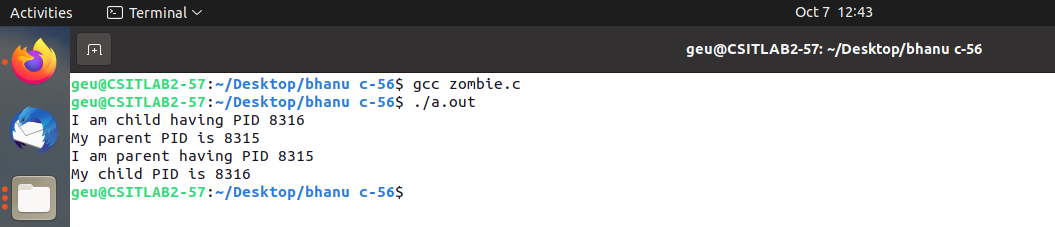
else{

printf("process not created");

}

return 0;

}

****

**6. Write a program to demonstrate the use of wait system call for handling orphan process.**

#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

int main()

{

int status;

pid\_t p;

p=fork();

if(p==0)

{

wait(&status);

printf("I AM A CHILD PROCESS WITH ID : %d\n",getpid());

printf("MY PARENT ID IS : %d \n",getpid());

}

else

{

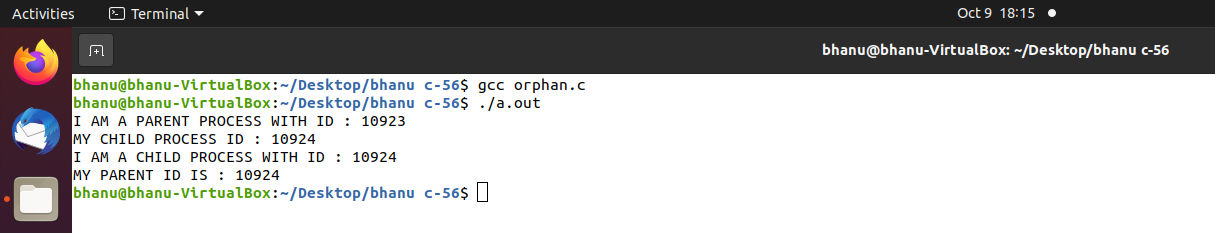
printf("I AM A PARENT PROCESS WITH ID : %d\n",getpid());

printf("MY CHILD PROCESS ID : %d \n",p);

}

return 0;

}

****

**7. Write a program to demonstrate the use of wait system call for handling Zombie process.**

#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

int main()

{

int status;

pid\_t p;

p=fork();

if(p==0)

{

printf("I AM A CHILD PROCESS WITH ID : %d\n",getpid());

printf("MY PARENT ID IS : %d \n",getpid());

}

else

{

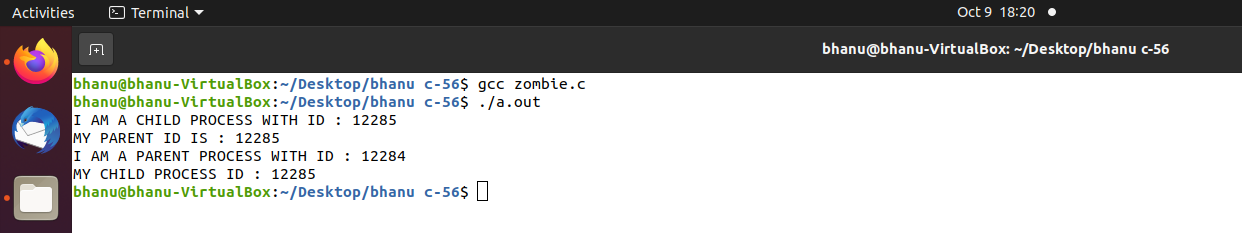
wait(&status);

printf("I AM A PARENT PROCESS WITH ID : %d\n",getpid());

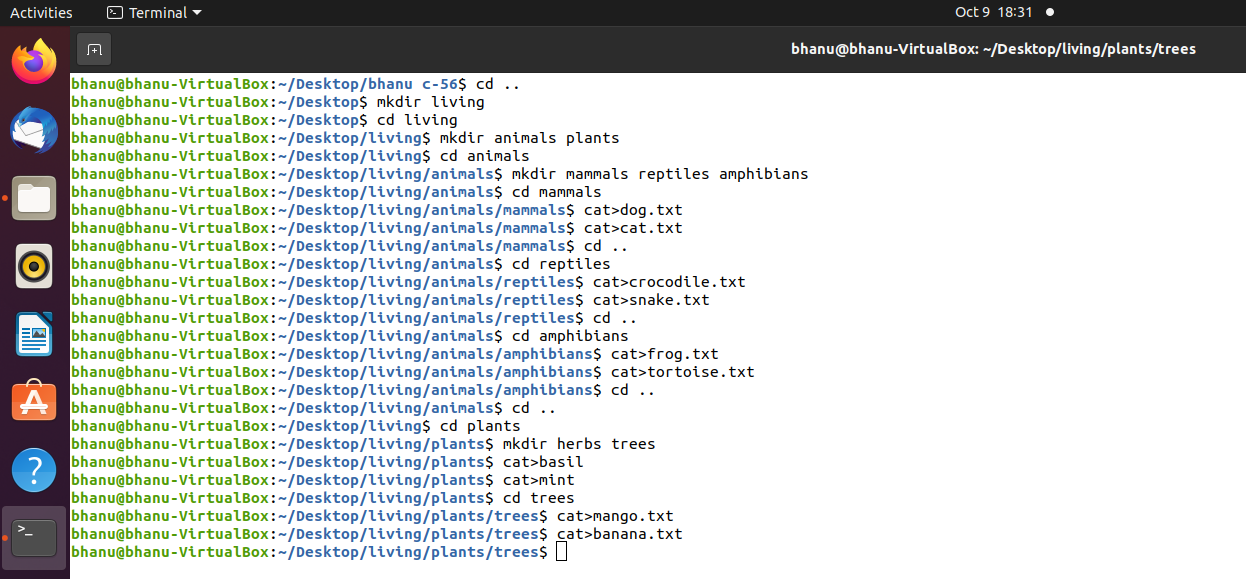
printf("MY CHILD PROCESS ID : %d \n",p);

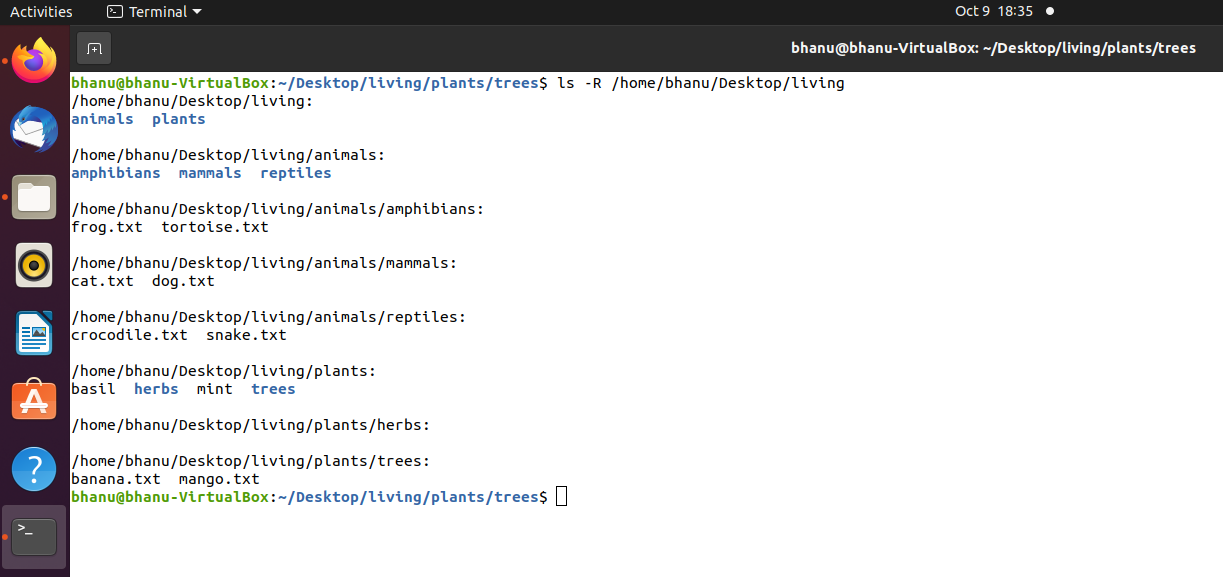
}

return 0;}

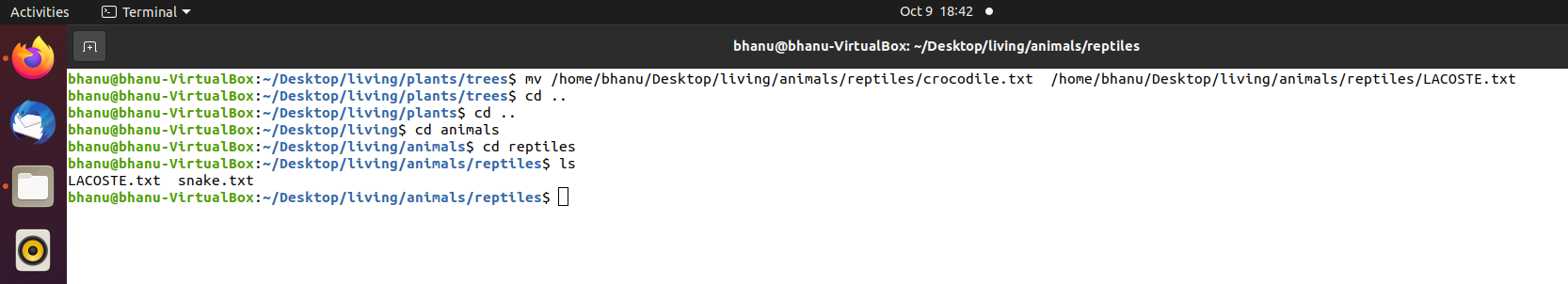
****

**8. Write a sequence of commands to create the following structure:**

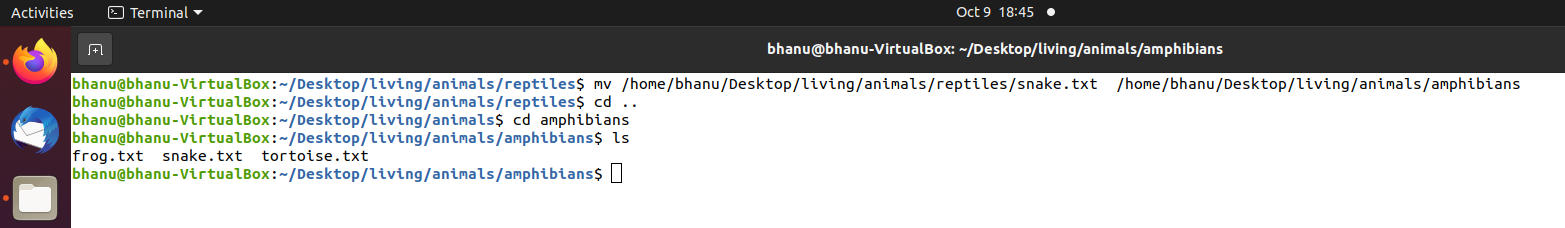
****

1. **Display the complete structure in a single step (suppose you are in Trees Directory)** 
   1. ****

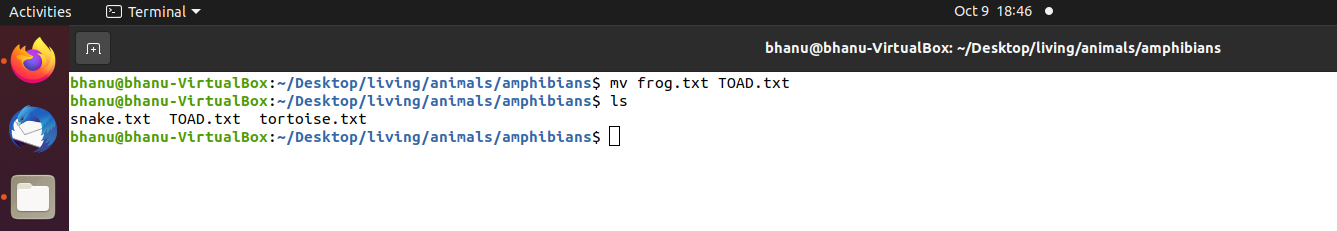
**b. Rename the file crocodile by LACOSTE in a single step (suppose you are in Trees Directory)**

****

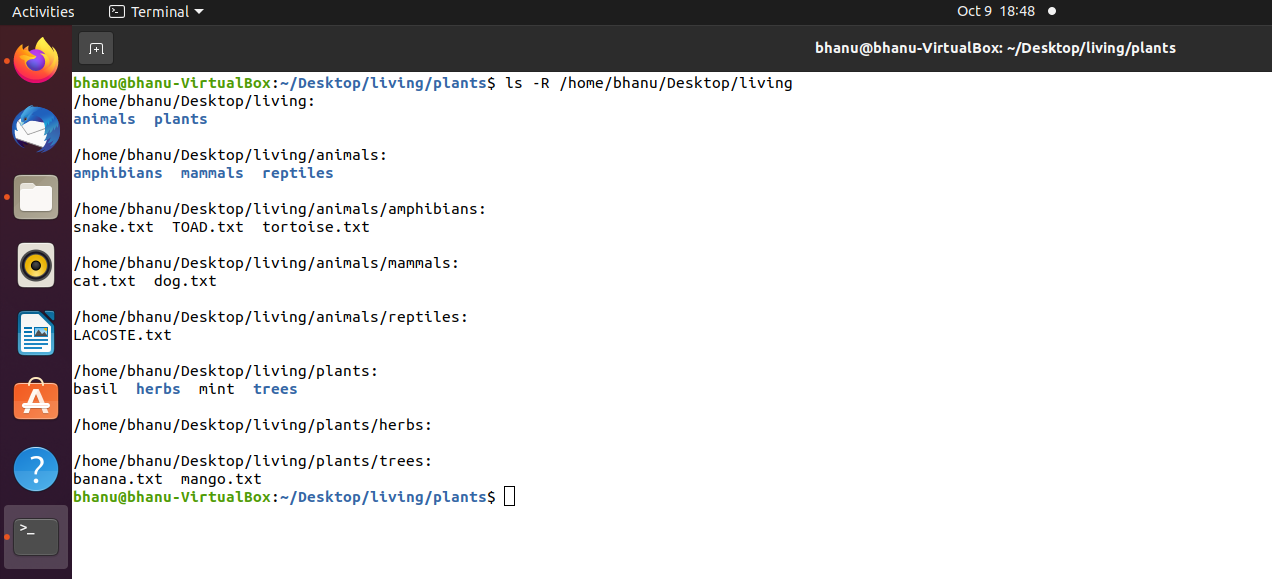
**c. Move the file SNAKE to folder Amphibians in a single step (suppose you are in Trees Directory)**

****

**d. Rename the file FROG to TOAD in a single step (suppose you are in Amphibians Directory)**

****

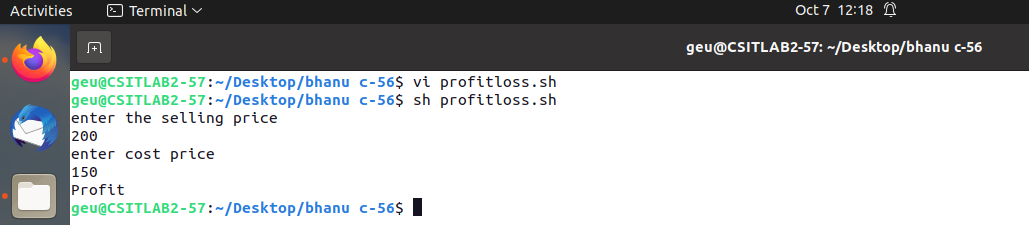
**e. Display the complete structure in a single step (suppose you are in Plants Directory)**

* 1. ****



**9. Write a shell script to check whether you incurred loss or profit after getting CP & SP as inputs from the user.**

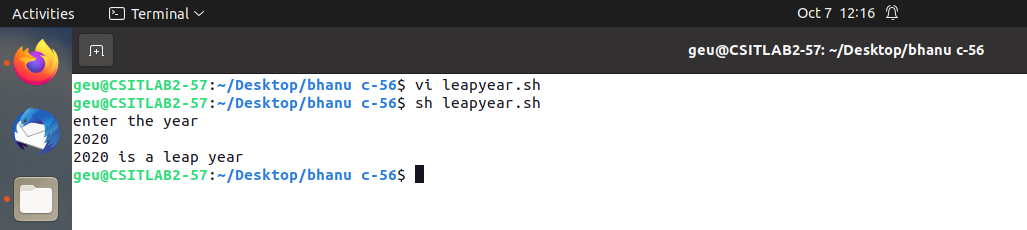
1. echo "enter the selling price"
2. read sp
3. echo "enter cost price"
4. read cp
5. if test `expr $sp - $cp` -gt 0
6. then echo "Profit"
7. else
8. echo "loss"
9. fi

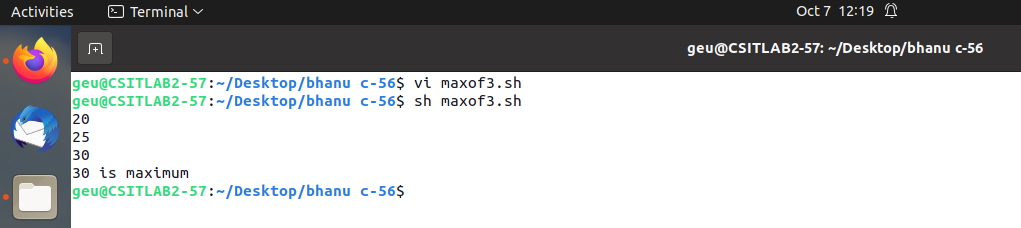




**10. Write a shell script to check whether the given year is a leap year or not.**

* 1. echo "enter the year"
  2. read year
  3. if test `expr $year % 400` -eq 0
  4. then echo "$year is a leap year"
  5. elif test `expr $year % 4` -eq 0
  6. then echo "$year is a leap year"
  7. else
  8. echo "$year is not a leap year"
  9. fi



2. **11. Write a shell script to print maximum of three given numbers.** 
   1. read a
   2. read b
   3. read c
   4. if test $a -gt $b
   5. then
   6. if test $a -gt $c
   7. then
   8. echo "$a is maximum"
   9. else
   10. echo "$c is maximum"
   11. fi
   12. else
   13. if test $b -gt $c
   14. then
   15. echo "$b is maximum"
   16. else
   17. echo "$c is maximum"
   18. fi
   19. fi
   20. 

**12. Write a shell script to check whether the given character is a vowel, or consonant., or a numeral or a special character. [Hint: case esac]**

echo "read ch"

read ch

case $ch in[0-9]) echo "numerical"

;;

[aeiou]) echo "vowel"

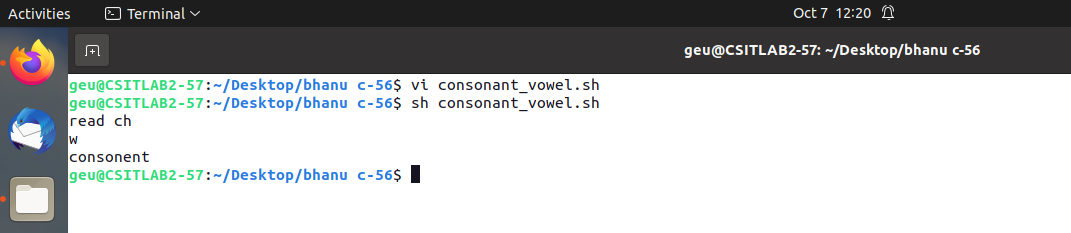
;;

[bcdfghjklmnpqrstvwxyz]) echo "consonent"

;;

\*) echo "special character"

Esac



**13. Write a shell script to generate ‘a’ raised to the power ‘b’.**

echo "Input number"

read no

echo "Input power"

read power

counter=0

ans=1

while [ $power -ne $counter ]

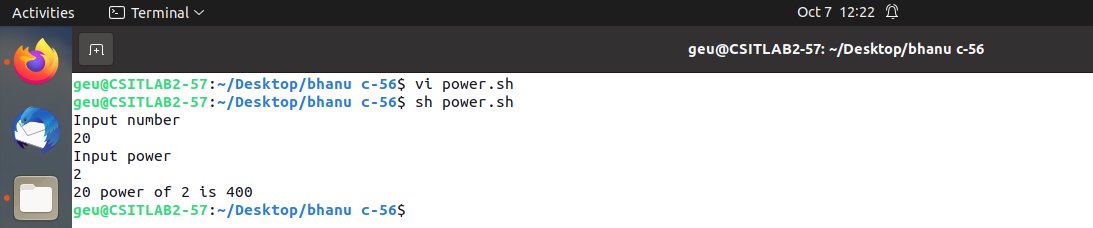
do

ans=`expr $ans \\* $no`

counter=`expr $counter + 1`

done

echo "$no power of $power is $ans"



**14. Write a shell script to print table of a given number.**

echo "Enter a Number"

read n

i=1

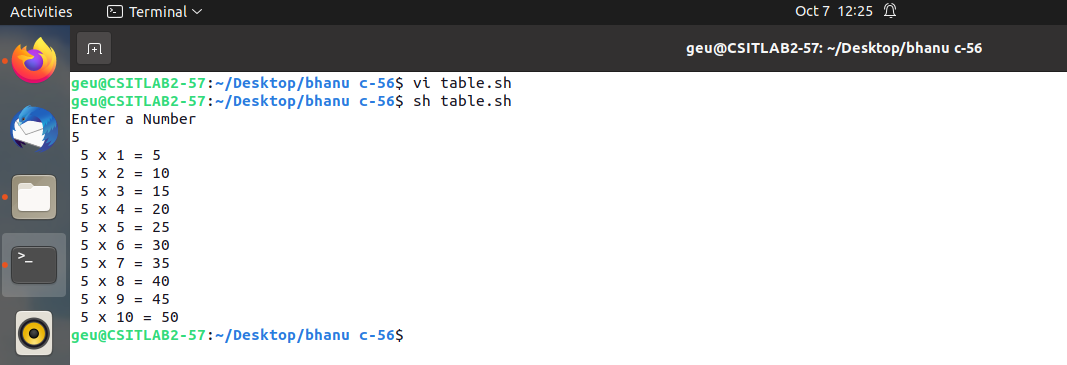
while [ $i -le 10 ]

do

echo " $n x $i = $(( n \* i ))"

i=$(( i + 1 ))

done



**15. Write a shell script to print factorial of a given number entered as command line argument.**

a=$1

f=1

while [ $a -ne 1 ]

do

f=`expr $f \\* $a`

a=`expr $a - 1`

done

echo "factorial = $f"

