**Q1. Demonstration of FORK() System Call**

#include<stdio.h>

#include<unistd.h>

int main(){

fork();

printf("Hello Linux\n");

return 0;

}

Graphical user interface, text, application

Description automatically generated

**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;

}

Graphical user interface, text, application, email

Description automatically generated

**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)

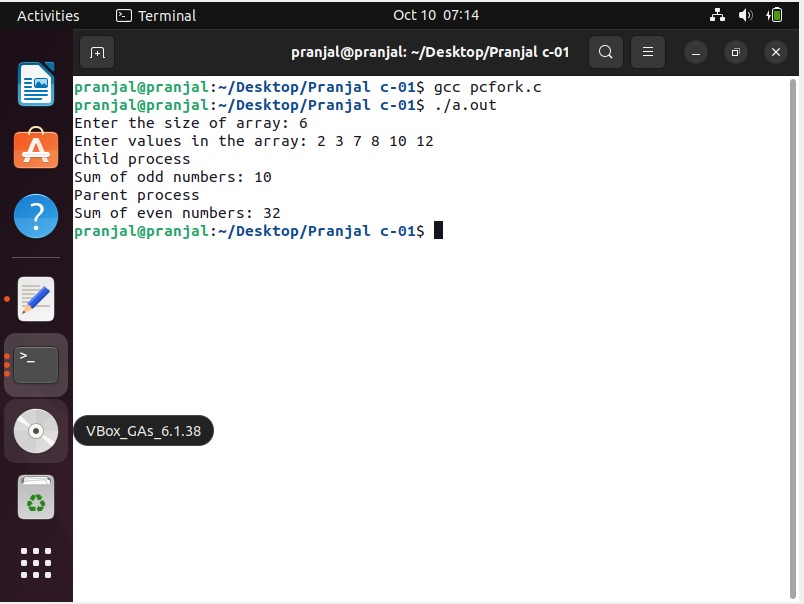
sumeven += arr[i];

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;

}

Graphical user interface, text

Description automatically generated

**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;

}

Graphical user interface, text, application

Description automatically generated

**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;

}

**Text

Description automatically generated**

**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);

}

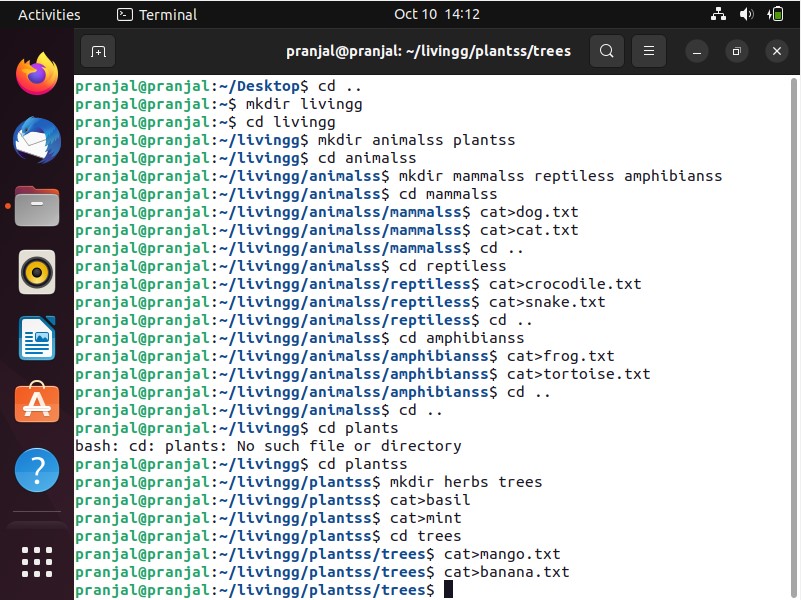
return 0;

}

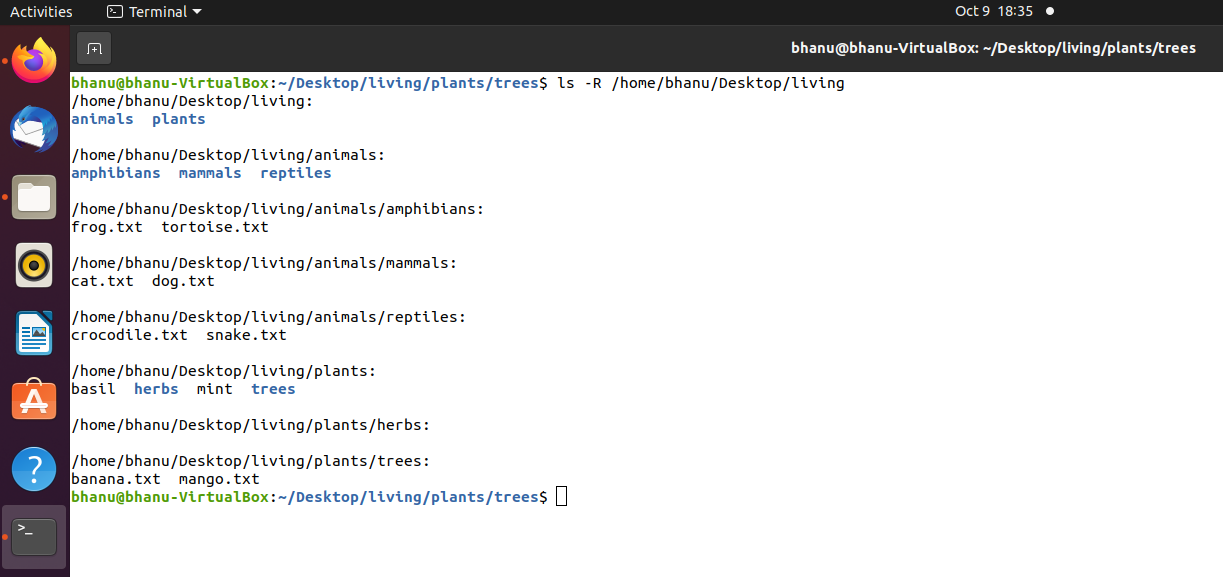
**Graphical user interface, text, application

Description automatically generated**

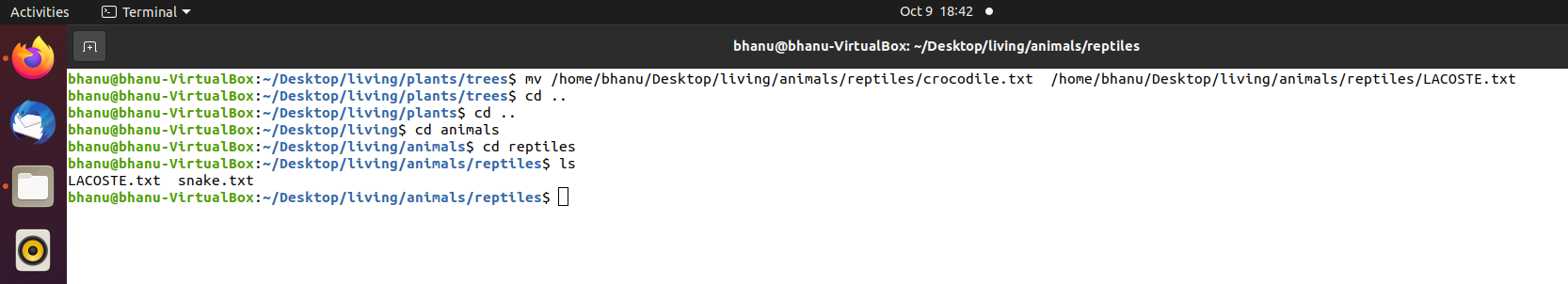
**8. Write a sequence of commands to create the following structure:**

****

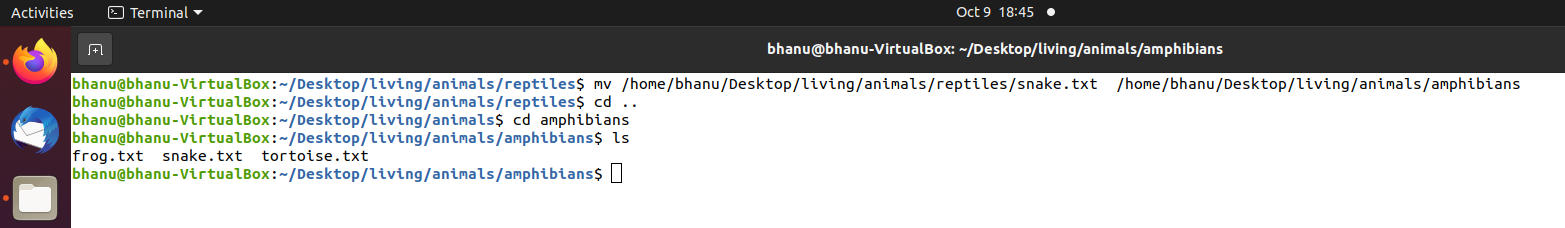
1. **Display the complete structure in a single step (suppose you are in Trees Directory)**

****

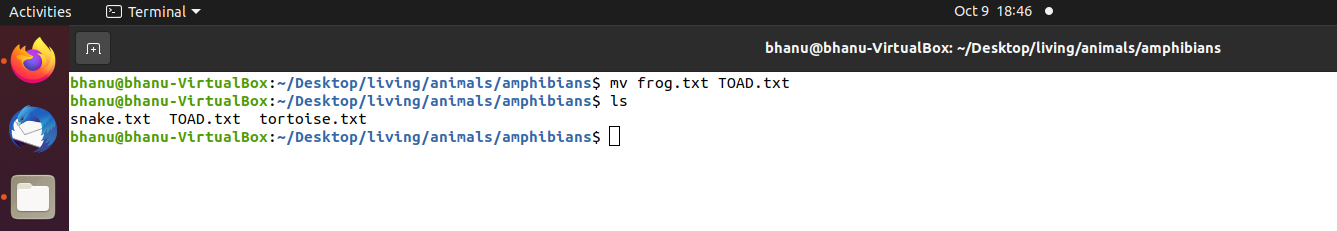
1. **Rename the file crocodile by LACOSTE in a single step (suppose you are in Trees Directory)**

****

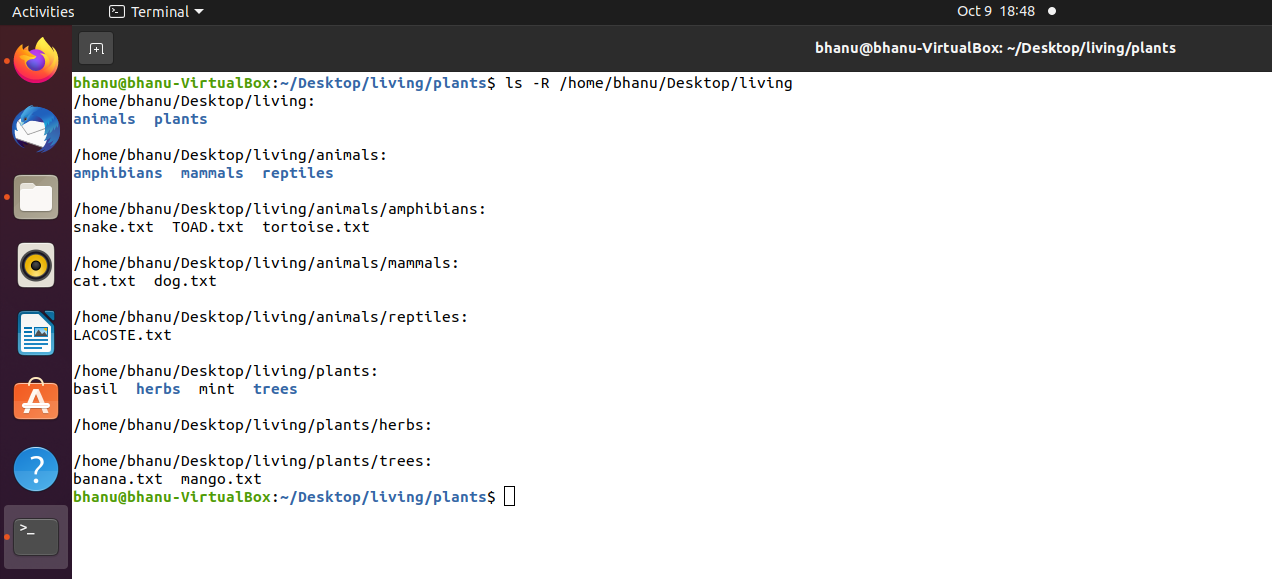
1. **Move the file SNAKE to folder Amphibians in a single step (suppose you are in Trees Directory)**

****

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

****

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

****

**9. Write a shell script to check whether you incurred loss or profit after**

**getting CP & SP as inputs from the user.**

echo "enter the selling price"

read sp

echo "enter cost price"

read cp

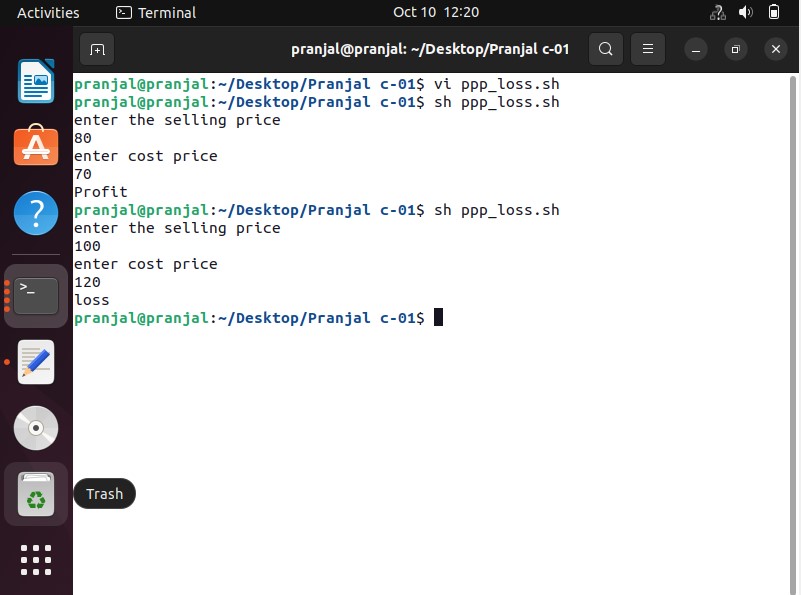
if test `expr $sp - $cp` -gt 0

then echo "Profit"

else

echo "loss"

fi



**10. Write a shell script to check whether the given year is a leap year or**

**not.**

echo "enter the year"

read year

if test `expr $year % 400` -eq 0

then echo "$year is a leap year"

elif test `expr $year % 4` -eq 0

then echo "$year is a leap year"

else

echo "$year is not a leap year"

fi

Graphical user interface, text

Description automatically generated

**11. Write a shell script to print maximum of three given numbers.**

read a

read b

read c

if test $a -gt $b

then

if test $a -gt $c

then

echo "$a is maximum"

else

echo "$c is maximum"

fi

else

if test $b -gt $c

then

echo "$b is maximum"

else

echo "$c is maximum"

fi

fi

Graphical user interface, text, application

Description automatically generated

**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

Graphical user interface, text, application, email

Description automatically generated

**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"

Graphical user interface, text, application

Description automatically generated

**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

Graphical user interface, text

Description automatically generated

**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"

Graphical user interface, text

Description automatically generated