**Q1. Write a Program to create four processes (1 parent and 3 children) where they terminates in a sequence as follows : (a) Parent process terminates at last (b) First child terminates before parent and after second child. (c) Second child terminates after last and before first child. (d) Third child terminates first**

// CPP code to create three child

// process of a parent

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

// Driver code

int main()

{

int pid, pid1, pid2;

// variable pid will store the

// value returned from fork() system call

pid = fork();

// If fork() returns zero then it

// means it is child process.

if (pid == 0) {

// First child needs to be printed

// later hence this process is made

// to sleep for 3 seconds.

sleep(3);

// This is first child process

// getpid() gives the process

// id and getppid() gives the

// parent id of that process.

printf("child[1] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

else {

pid1 = fork();

if (pid1 == 0) {

sleep(2);

printf("child[2] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

else {

pid2 = fork();

if (pid2 == 0) {

// This is third child which is

// needed to be printed first.

printf("child[3] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

// If value returned from fork()

// in not zero and >0 that means

// this is parent process.

else {

// This is asked to be printed at last

// hence made to sleep for 3 seconds.

sleep(5);

printf("parent --> pid = %d\n", getpid());

}

}

}

return 0;

}

Q3. **Write a program to execute consecutive three fork system calls without any condition and then draw the parent child relationship diagram using exact pid of the process.**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include<sys/wait.h>

Int main()

{

system(“clear”);

int ch1,ch2,ch3;

ch1=fork();

ch2=fork();

ch3=fork();

printf(“Parent: %4d, Own: %4d, Child1: %4d, Child2: %4d, Child3: %4d\n”,getppid(),getpid(),ch1,ch2,ch3);

}

Q4**. Write a program to create a child process, which execute an already compiled Fibonacci series program.**

***PRO5.C***

#include<stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include<sys/wait.h>

**int** main()

{

**int** n1=0,n2=1,n3,i,number;

 printf("Enter the number of elements:");

 scanf("%d",&number);

 printf("\n%d %d",n1,n2);//printing 0 and 1

**for**(i=2;i<number;++i)//loop starts from 2 because 0 and 1 are already printed

 {

  n3=n1+n2;

  printf(" %d",n3);

  n1=n2;

  n2=n3;

 }

**return** 0;

 }

***PRO4.C***

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main()

{

pid\_t pid=fork();

if (pid==0)

{

/\* child process \*/

char \*argv[]={"./pro5",NULL};

execv(argv[0],argv);

exit(127);

/\* only if execv fails \*/

}

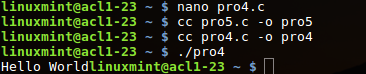
else {

waitpid(pid,0,0);

}

return 0;

}



Q5. **Write a program to demonstrate zombie process and also explain the necessary condition for zombie process.**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include<sys/wait.h>

Int main()

{

system(“clear”);

int ch1,ch2;

ch1=fork();

ch2=fork();

printf(“Parent: %4d, Own: %4d, Child1: %4d, Child2: %4d\n”,getppid(),getpid(),ch1,ch2);

wait(NULL);}

2.**Write a program if the existing child process is being killed or terminated then the parent process is intelligently initiating a new child.**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main(void)

{

pid\_t pid1,status;

pid1 = fork();

if ( pid1 == 0 )

{

printf("child id %u parent id %u",getpid(),getppid());

sleep(2);

exit(0);

}

wait(NULL);

waitpid(pid1, &status, WNOHANG);

if (WIFEXITED(status))

{ printf("\nCreating new child\n");

pid1=fork();

if(pid1==0)

{

printf("New child id %u parent id %u",getpid(),getppid());

}

}

return 0;

}

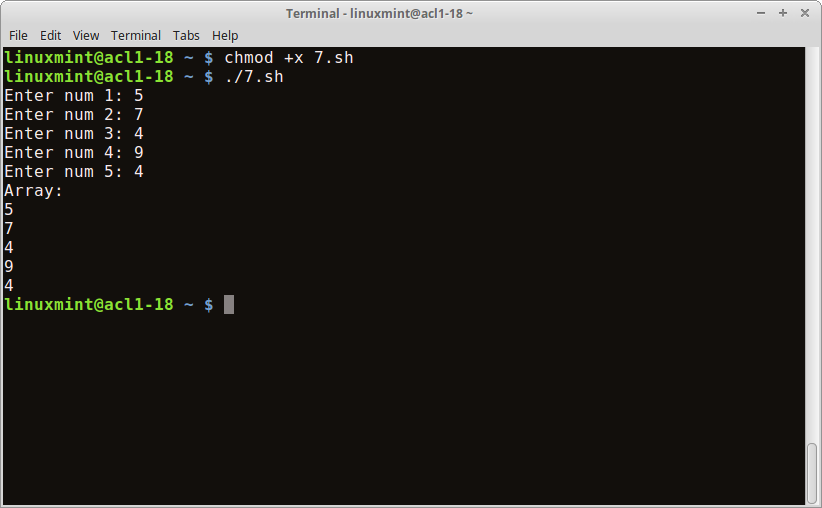
**6. Write a Shell Script to Display numbers Using Array.**

#!/bin/bash

for((i=0;i<5;i++));do read -p "Enter num `expr $i + 1`: " arr[$i];done

echo "Array:"

for((i=0;i<5;i++));do echo ${arr[$i]};done



**7. Write a Shell Script to add two numbers using Function.**

#!/bin/bash

add()

{

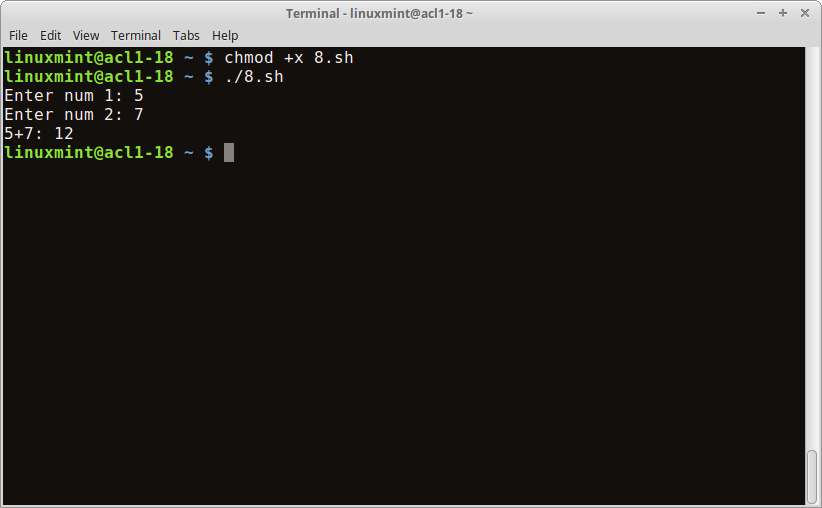
x=$1;y=$2

echo "$1+$2: `expr $x + $y` "

}

read -p "Enter num 1: " first

read -p "Enter num 2: " sec

add $first $sec

**9. Write a Shell Script to implement Bubble Sort.**

#!/bin/bash

read -p "enter maximum number" n

echo "enter Numbers in array:"

for (( i = 0; i < $n; i++ ));do read nos[$i];done

for (( i = 0; i < $n ; i++ ));do

for (( j = $i; j < $n; j++ ));do

if [ ${nos[$i]} -gt ${nos[$j]} ]; then

t=${nos[$i]}

nos[$i]=${nos[$j]}

nos[$j]=$t

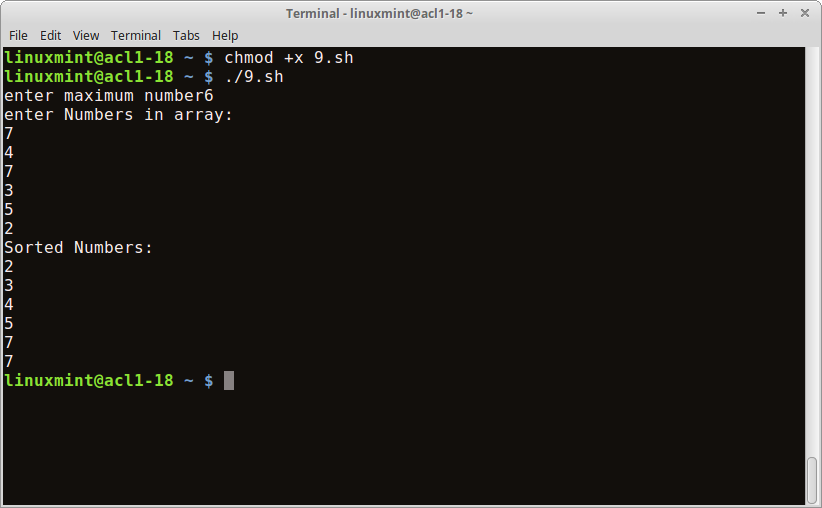
fi

done

done

echo -e "Sorted Numbers: "

for (( i=0; i < $n; i++ ));do echo ${nos[$i]};done



**9. Write a Shell Script to implement Insertion Sort.**

#!/bin/bash

read -p "Enter max: " n

echo "Enter array"

for((i=0;i<n;i++));do read arr[$i];done

for((i=1;i<n;i++));do

j=$i-1

temp=${arr[$i]}

while((j>=0 && arr[j]>temp));do

arr[$j+1]=${arr[$j]}

j=$j-1

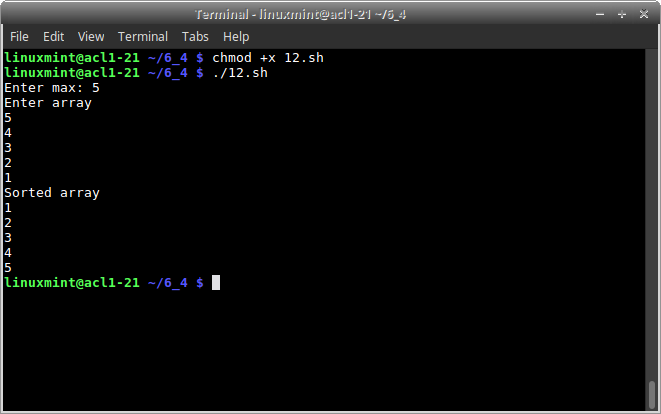
done

arr[j+1]=$temp

done

echo "Sorted array"

for((i=0;i<n;i++));do echo ${arr[$i]};done



10. **. Write shell script using for loop to print the following patterns on screen.**

#! /bin/bash

for (( i=1; i<=9; i++));do

for (( j=1; j<=9-i; j++));do

echo -n " "

done

for (( j=1; j<=i; j++));do

echo -n $i ""

done

echo

done