

|  |  |  |
| --- | --- | --- |
| **NAME :**  Sriram Vaishnav R | **REG NO :**  21BRS1378 | **MAIL ID :**  sriramvaishnav.r2021@vitstudent.ac.in |
| **PROGRAMME :**  BTech-CSE with spl. in Artificial Intelligence and Robotics | **SCHOOL :**  School of Computer Science and Engineering(SCOPE) | **SEM :**  WS (’22 – ’23) |
| **FACULTY :**  Dr Anandan P | **COURSE :**  BCSE303P | **SLOT :**  (L43 + L44) |

**Assessment-2**

1. **Develop a C program to create four child processes. Print the parent ID in all child processes along with its own ID. Print all Child IDs in parent process along with its own ID.**

Program:

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<stdio.h>

int main()

{

printf("Done by SRIRAM VAISHNAV R 21BRS1378\n\n");

for(int i=0;i<4;i++)

{

if(fork() == 0)

{

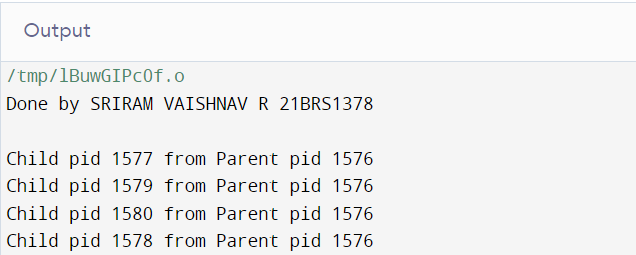
printf("Child pid %d from Parent pid %d\n",getpid(),getppid());

exit(0);

}

}

}



1. **Develop a C program to print even numbers between 0 and 10 within Child process & print odd numbers between 0 and 10 within Parent process.**

Program:

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<stdio.h>

int main()

{

printf("Done by SRIRAM VAISHNAV R 21BRS1378\n\n");

int pid;

pid=fork();

if(pid==0)

{

printf("Child: Process id: %d\n",getpid());

for(int i=0;i<=10;i++)

{

(i%2==0)?printf("%d ",i):printf("");

}

printf("\n");

}

else if(pid>0)

{

printf("Parent: Process id: %d\n",getppid());

for(int i=0;i<=10;i++)

{

(i%2!=0)?printf("%d ",i):printf("");

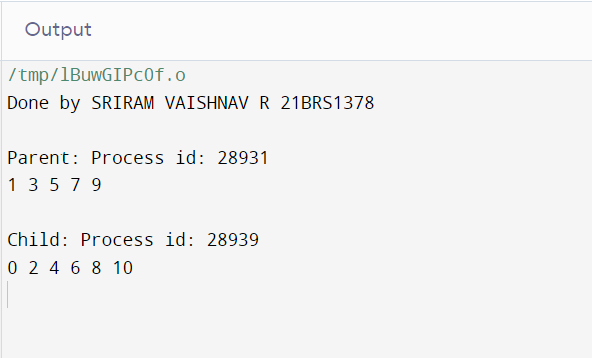
}

printf("\n\n");

}

return 0;

}



1. **Develop a C program to perform matrix addition in child process and determine whether a number is prime or not in parent process.**

Program:

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<stdio.h>

void parent()

{

int flag=1,n;

scanf("%d",&n);

for(int i=1;i<n/2;i++)

{

if(n%i==0)

{

flag=0;

break;

}

}

(flag==1)?printf("The number given is a prime number\n"):printf("The number given is not a prime number\n");

}

void child()

{

int d;

scanf("%d",&d);

int a[d][d],b[d][d],sum[d][d];

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

{

for(int j=0;j<d;j++)

{

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

}

}

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

{

for(int j=0;j<d;j++)

{

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

}

}

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

{

for(int j=0;j<d;j++)

{

sum[i][j]=a[i][j]+b[i][j];

}

}

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

{

for(int j=0;j<d;j++)

{

printf("%d\t",sum[i][j]);

}

printf("\n");

}

}

int main()

{

printf(“Done by SRIRAM VAISHNAV R 21BRS1378\n\n”);

int pid;

pid=fork();

if(pid==0)

{

child();

printf("Child: Process id: %d\n",getpid());

}

if(pid>0)

{

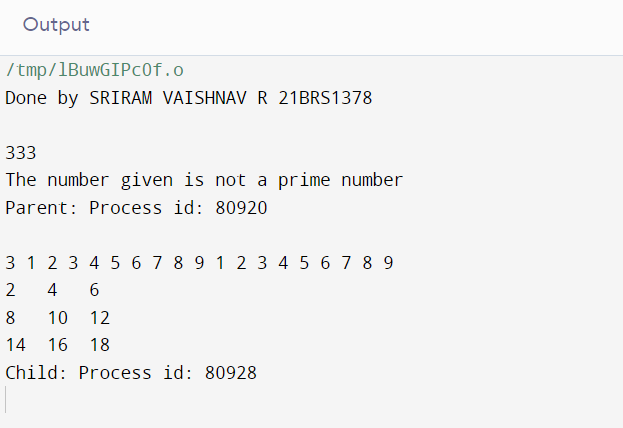
parent();

printf("Parent: Process id: %d\n\n",getppid());

}

return 0;

}



1. **Develop a C program to demonstrate the Orphan and Zombie process.**

Program:

#include<stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

printf("Done by SRIRAM VAISHNAV R 21BRS1378\n\n");

int pid = fork();

if (pid>0)

printf("In parent process\nChild process became orphan and goes to sleep");

else if(pid==0)

{

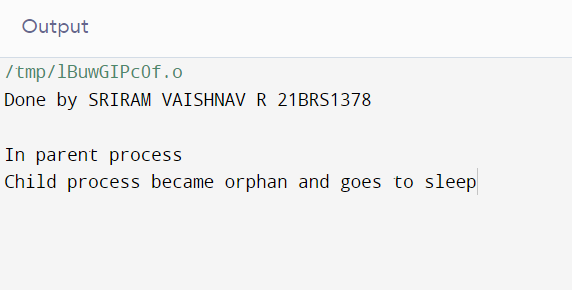
sleep(30);

printf("\nThe End after 30 seconds");

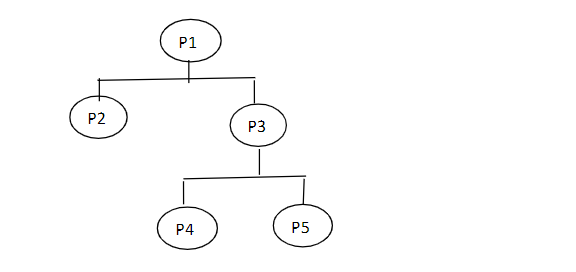
}

return 0;

}



1. **Develop a C program to create a process hierarchy as shown below.**



Program:

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

printf("Done by SRIRAM VAISHNAV R 21BRS1378\n\n");

int P1,P2,P3,P4,P5;

int pid1=fork();

if(pid1==0)

{

printf("P2 of child process id %d under %d\n",getpid(),getppid());

}

else

{

int pid2=fork();

if(pid2==0)

{

int pid3=fork();

if(pid3==0)

{

printf("P4 of child process id %d under %d\n",getpid(),getppid());

}

else

{

int pid4=fork();

if(pid4==0)

{

printf("P5 of child process id %d under %d\n",getpid(),getppid());

}

else

{

printf("P3 of child process id %d under %d\n",getpid(),getppid());

}

}

}

else

{

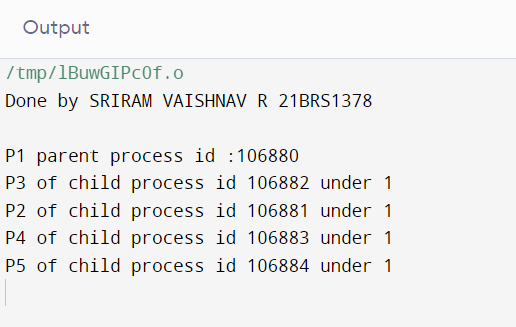
printf("P1 parent process id :%d\n",getpid());

}

}

return 0;

}



**Conclusion:** The given tasks has been performed and thus the intended concepts have been understood.