**PROCESS MANAGEMENT**

**Title : Odd-Even**

Name: Vedant Vichare

Roll No.: 35

Panel : B

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<sys/wait.h>

int main()

{

   int n;

   printf("Enter number n:");

   scanf("%d",&n);

   printf("\n");

   pid\_t p;

   p=fork();

 if(p==0)

     {

          printf("This is child process of pid %d \n",getpid());

          printf("This is child process of parent pid %d \n",getppid());

          printf("This is child process of retun value from p is %d \n",p);

          printf("Child process enter odd numbers:\n");

            for(int i=1;i<n;i++)

                 {

                    if(i%2!=0)

                      {

                          printf("%d\n",i);

                      }

                 }

    }

 else if(p>0)

   {

        wait(NULL);

        printf("parent process of pid %d \n",getpid());

        printf("parent process of parent pid %d \n",getppid());

        printf("parent process of return value from p is %d \n",p);

        printf("Parent process enter even numbers:\n");

           for(int i=1;i<n;i++)

                {

                   if(i%2==0)

                    {

                        printf("%d\n",i);

                    }

               }

  }

 else

  {

    printf("fork process faild or new child process not created\n");

  }

return 0;

}

**OUTPUT:**

(base) computer@computer:~$ gcc fork.c

(base) computer@computer:~$ ./a.out

Enter number n:50

This is child process of pid 5291

This is child process of parent pid 5290

This is child process of retun value from p is 0

Child process enter odd numbers:

1

3

5

7

9

11

13

15

17

19

21

23

25

27

29

31

33

35

37

39

41

43

45

47

49

parent process of pid 5290

parent process of parent pid 5257

parent process of return value from p is 5291

Parent process enter even numbers:

2

4

6

8

10

12

14

16

18

20

22

24

26

28

30

32

34

36

38

40

42

44

46

48

(base) computer@computer:~$

**Title : Zombie process**

Name: Vedant Vichare

Roll No.: 35

Panel : B

Code:

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

   pid\_t p;

   p=fork();

 if(p==0)

     {

          printf("This is child process of pid %d \n",getpid());

          printf("This is child process of parent pid %d \n",getppid());

          printf("This is child process of retun value from p is %d \n",p);

    }

 else if(p>0)

   {

        sleep(5);

        system (" ps -axj | tail ");

        printf("parent process of pid %d \n",getpid());

        printf("parent process of parent pid %d \n",getppid());

        printf("parent process of return value from p is %d \n",p);

  }

 else

  {

    printf("fork process faild or new child process not created\n");

  }

return 0;

}

**OUTPUT:**

(base) computer@computer:~$ gcc zombie.c

(base) computer@computer:~$ ./a.out

This is child process of pid 4344

This is child process of parent pid 4343

This is child process of retun value from p is 0

      2 4265 0 0 ? -1 I 0 0:00 [kworker/u32:1]

      2 4298 0 0 ? -1 I 0 0:00 [kworker/13:2-events]

      2 4299 0 0 ? -1 I 0 0:00 [kworker/4:0-events]

   1087 4305 4305 4305 ? -1 Ssl 1000 0:00 /usr/libexec/tracker-store

      2 4339 0 0 ? -1 I 0 0:00 [kworker/u32:3-events\_unbound]

   4056 4343 4343 4056 pts/0 4343 S+ 1000 0:00 ./a.out

   4343 4344 4343 4056 pts/0 4343 Z+ 1000 0:00 [a.out] <defunct>

   4343 4345 4343 4056 pts/0 4343 S+ 1000 0:00 sh -c ps -axj | tail

   4345 4346 4343 4056 pts/0 4343 R+ 1000 0:00 ps -axj

   4345 4347 4343 4056 pts/0 4343 S+ 1000 0:00 tail

parent process of pid 4343

parent process of parent pid 4056

parent process of return value from p is 4344

**Title : Orphan process**

Name: Vedant Vichare

Roll No.: 35

Panel : B

Code:

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<sys/wait.h>

int main()

{

   pid\_t p;

   p=fork();

 if(p==0)

     {

          printf("This is child process of pid %d \n",getpid());

          printf("This is child process of parent pid %d \n",getppid());

          printf("This is child process of return value from p is %d \n",p);

   sleep(5);

    }

 else if(p>0)

   {

        printf("parent process of pid %d \n",getpid());

        printf("parent process of parent pid %d \n",getppid());

        printf("parent process of return value from p is %d \n",p);

  }

 else

  {

    printf("fork process faild or new child process not created\n");

  }

return 0;

}

**OUTPUT:**

(base) computer@computer:~$ gcc orphan.c

(base) computer@computer:~$ ./a.out

parent process of pid 4781

parent process of parent pid 4706

parent process of return value from p is 4782

This is child process of pid 4782

This is child process of parent pid 4781

This is child process of return value from p is 0