Week : 1

(Fork() System call)

1. **Using a single fork()**

/\*Name : Maulik Gupta

Section : H

Roll no : 36

University Roll no : 2017193 \*/

#include <stdio.h>

#include <unistd.h>

int main()

{

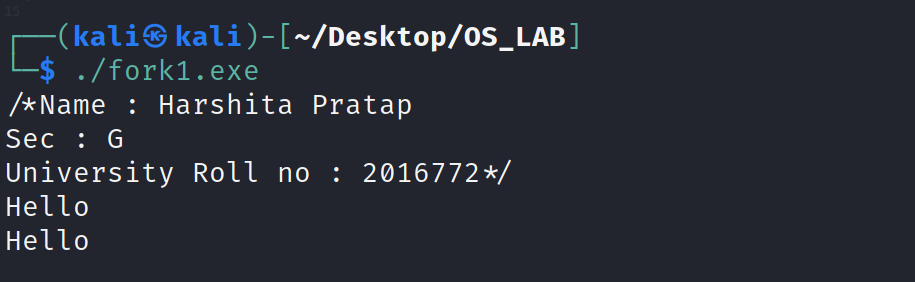
printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

fork();

printf("Hello\n");

return 0;

}



**2.using two fork() system calls**

#include <stdio.h>

#include <unistd.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

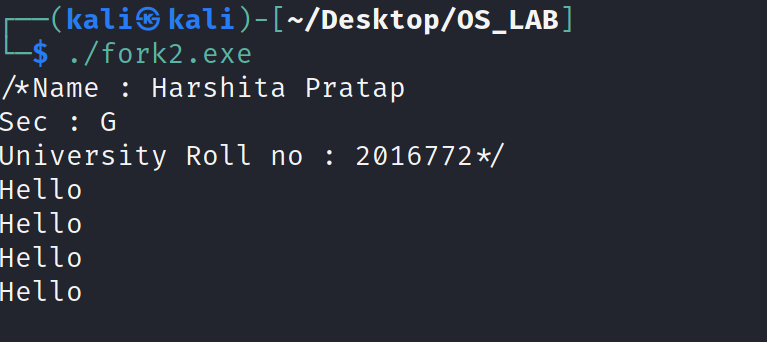
fork();

fork();

printf("Hello\n");

return 0;

}



3.What will be the output of the following code :

#include <stdio.h>

#include <unistd.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

printf("Hello\n");

fork();

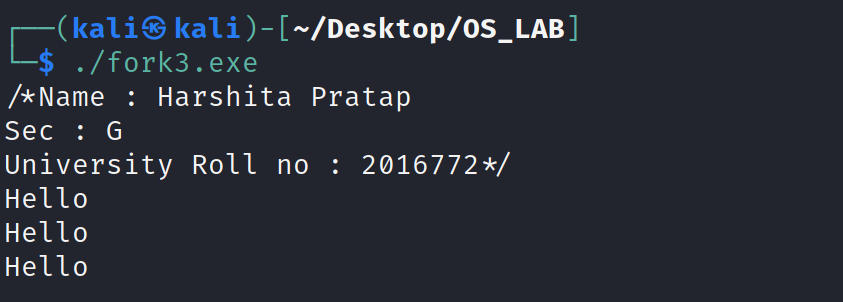
printf("Hello\n");

fork();

fork();

return 0;

}



Week -2

1.**Using of operator && and the fork system call**

#include <stdio.h>

#include <unistd.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

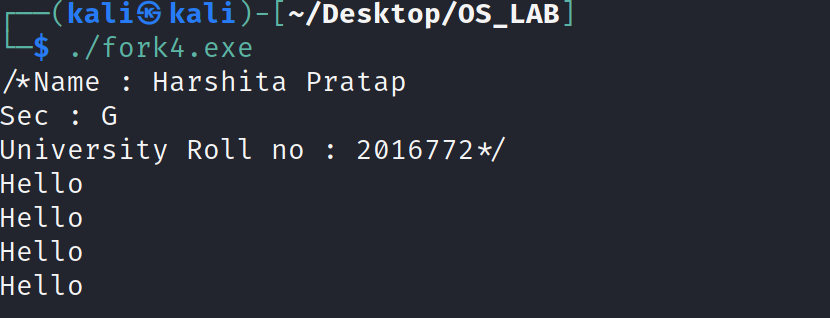
if (fork() && fork())

fork();

printf("Hello\n");

return 0;

}



**2.Using of operator || and fork system call**

#include <stdio.h>

#include <unistd.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

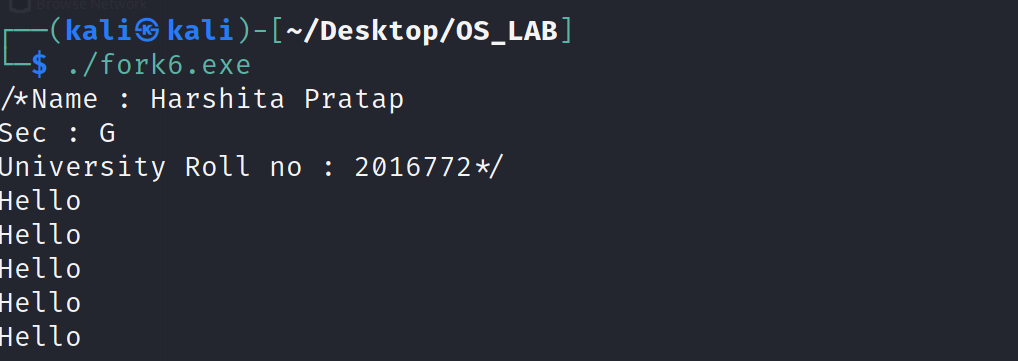
if (fork() || fork())

fork();

printf("Hello\n");

return 0;

}



**3.Program to print even position elements sum if parent and odd element position sum if child**

#include <stdio.h>

#include <unistd.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

int n = 0;

printf("Enter size of array\n");

scanf("%d",&n);

int arr[n];

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

{

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

}

int even = 0 ;

int odd = 0;

int check = fork();

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

{

if (i% 2 == 0)

{

even = even + arr[i];

}

else

odd = odd + arr[i];

}

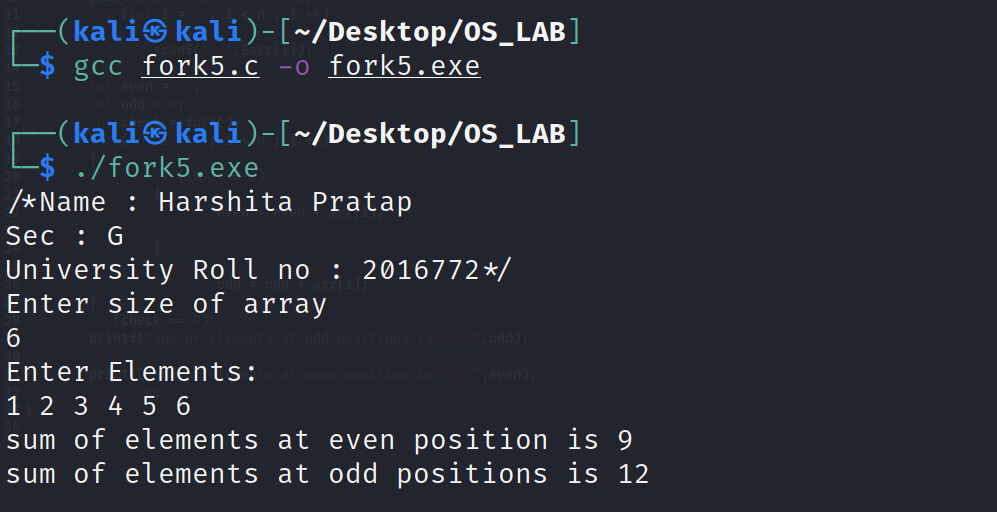
if (check == 0)

printf("Odd no sum is %d",odd);

else

printf("Even no sum is %d",even);

return 0;}



Week : 3

1. **Using PID**

#include <stdio.h>

#include <unistd.h>

int main()

{

int id1 = fork();

int id2 = fork();

if (id1 > 0 && id2 > 0)

{

printf("Hello! I am Parent(Level1)\n");

printf ("\n\nParent : Parent’s PID: %d\n", getpid());

printf ("Parent : Child’s PID1: %d\n", id1);

printf ("Parent : Child’s PID2: %d\n", id2);

}

else if (id1 == 0 && id2 > 0)

{

printf("\nHello! I am child1(Level2)\n");

printf("\n\nPId1 of child1 on level 2 is %d\n",getpid());

printf("child1 : Parent's PID is %d\n",getppid());

}

else if (id1 > 0 && id2 == 0)

{

printf("\n\nHello! I am child2(Level2)\n");

printf("\n\nPId2 of child2 on level 2 is %d\n",getpid());

printf("child2 : Parent's PID is %d\n",getppid());

}

else

{

printf("\n\nHello! I am child3(Level3)\n");

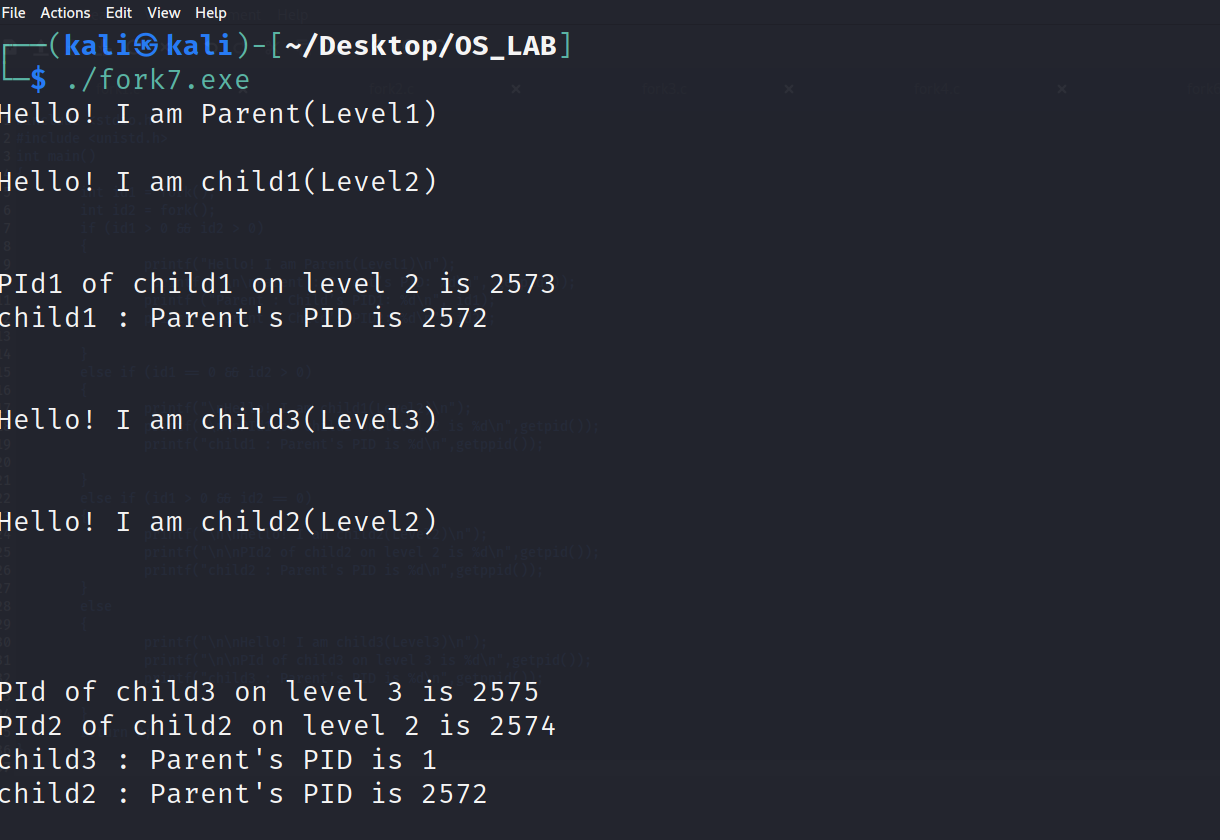
printf("\n\nPId of child3 on level 3 is %d\n",getpid());

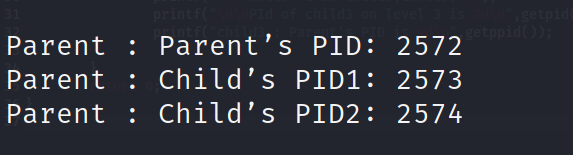
printf("child3 : Parent's PID is %d\n",getppid());

}

return 0;

}





**2.Using PID**

#include <stdio.h>

#include <unistd.h>

int main()

{

int id1 = fork();

int id2 = fork();

int id3 = fork();

if (id1 > 0 && id2 > 0 && id3 > 0)

{

printf("Hello! I am Parent(Level1)\n");

}

else if (id1 == 0 && id2 > 0 && id3 > 0)

{

printf("Hello! I am child1(Level2)\n");

}

else if (id1 > 0 && id2 == 0 && id3 > 0)

{

printf("Hello! I am child2(Level2)\n");

}

else if (id1 > 0 && id2 > 0 && id3 == 0)

{

printf("Hello! I am child3(Level2)\n");

}

else if (id1 == 0 && id2 == 0 && id3 > 0)

{

printf("Hello! I am child4(Level3)\n");

}

else if (id1 == 0 && id2 > 0 && id3 == 0)

{

printf("Hello! I am child5(Level3)\n");

}

else if (id1 > 0 && id2 == 0 && id3 == 0)

{

printf("Hello! I am child6(Level3)\n");

}

else

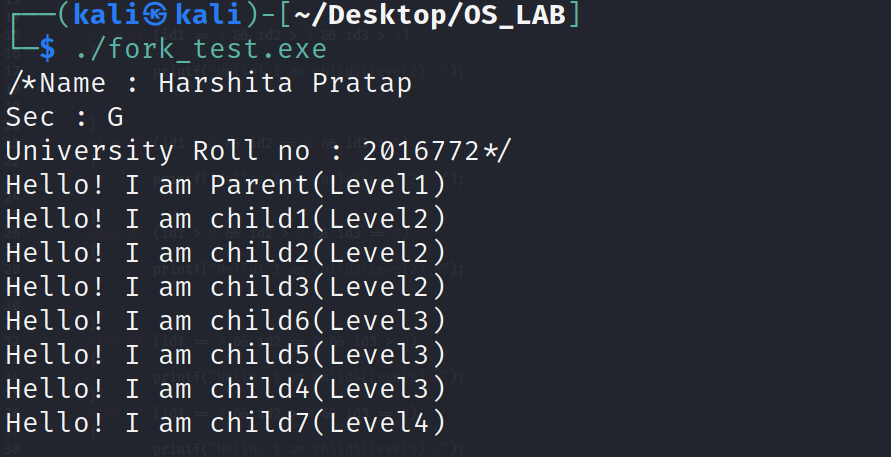
{

printf("Hello! I am child7(Level4)\n");

}

return 0;

}



Week 4

**1. A C program to demonstrate Orphan Process.**

// Parent process finishes execution while the

// child process is running. The child process

// becomes orphan.

#include<stdio.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

// Create a child process

int pid = fork();

if (pid > 0)

{

printf("In parent process");

}

// Note that pid is 0 in child process

// and negative if fork() fails

else if (pid == 0)

{

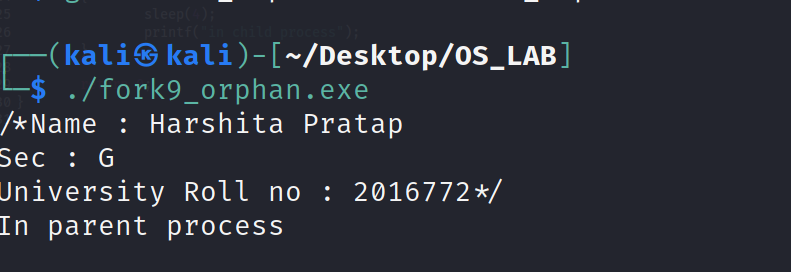
sleep(4);

printf("in child process");

}

return 0;

}



**Solution for orphan process**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

pid\_t p = fork();

if (p > 0)

{

sleep(6);//The parent will go into sleep and till then other process will execute

printf("Hello , I am parent i have waited for child for 6 seconds\n");

}

else

{

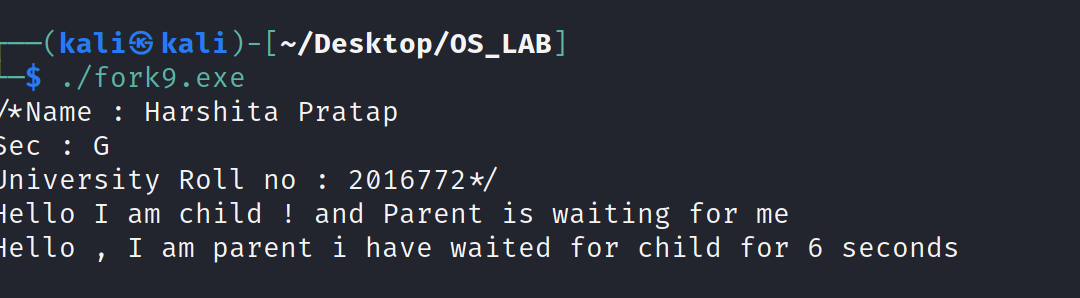
printf("Hello I am child ! and Parent is waiting for me\n");

exit(0);

}

return 0;

}



**2. A C program to demonstrate Zombie Process.**

// Child becomes Zombie as parent is sleeping

// when child process exits.

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

// Fork returns process id

// in parent process

pid\_t child\_pid = fork();

// Parent process

if (child\_pid > 0)

sleep(2);

// Child process

else

{

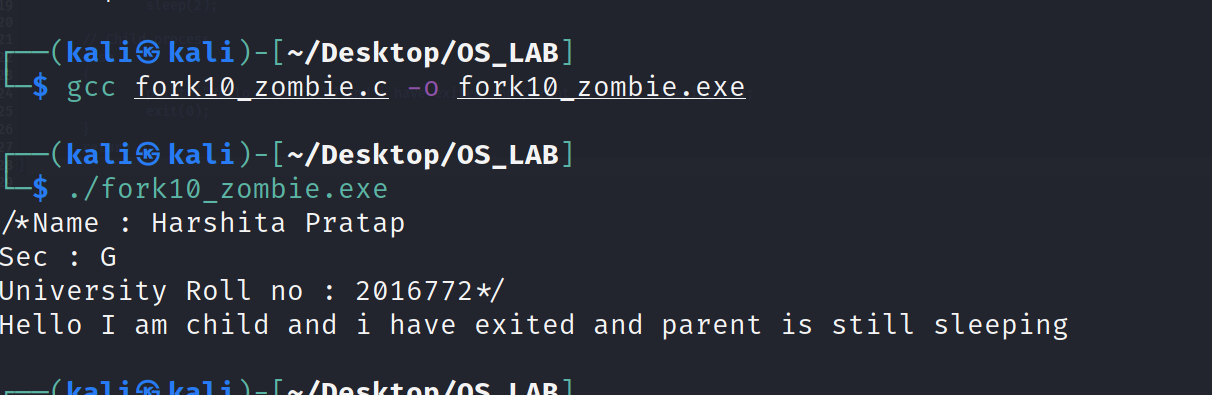
printf("Hello I am child and i have exited and parent is still sleeping\n");

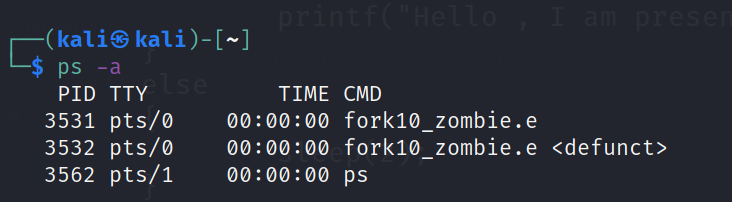
exit(0);

}

return 0;

}





**2.Solution for zombie process**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

int main()

{

printf("/\*Name : Maulik Gupta\nSec : H\nUniversity Roll no : 2017193\*/ \n");

pid\_t p = fork();

if (p > 0)

{

printf("Hello , I am present");

}

else

{

sleep(2);//child is put on sleep

}

return 0;

}

