Name : Rajvardhan Reddy
Reg No: 180905093
Sec : B
Roll No : 19

OS Lab – 3 : Process and Signal

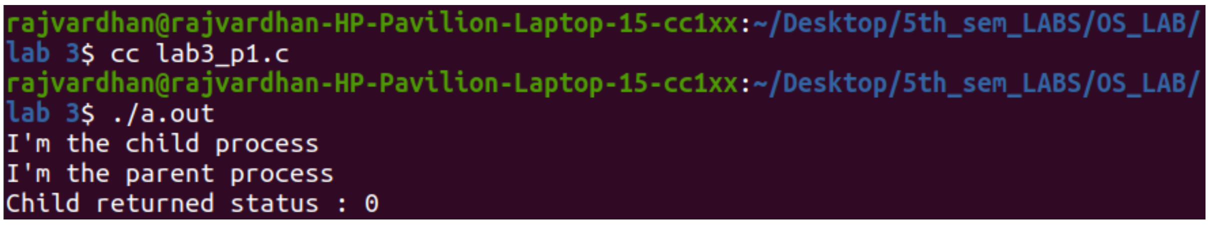
Lab Exercises :

P1) Write a C program to block a parent process until the child completes using a wait system call.

Program :

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char const *argv[])
{
// printf("Q1 program executing\n");
int status;
pid_t pid;
pid = fork();

if(pid == -1)
{
printf("Error creating child process\n");
}

else if(pid == 0)
{
printf("I'm the child process\n");
exit(0);
}

else
{
wait(&status);
printf("I'm the parent process\n");
printf("Child returned status : %d\n",status);
}
return 0;
}
```

Output :



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ cc lab3_p1.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ ./a.out
I'm the child process
I'm the parent process
Child returned status : 0
```

P2) Write a C program to load the binary executable of the previous program in a child process using the exec system call.

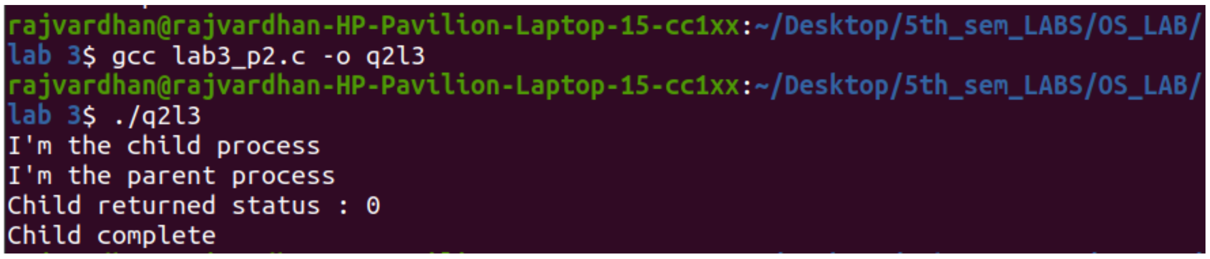
Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<dirent.h>
#include<sys/types.h>
#include<sys/wait.h>
int main(int argc, char const *argv[])
{
pid_t pid;
pid = fork();

if(pid < 0)
{
fprintf(stderr,"Error in creating child process\n");
exit(-1);
}

else if(pid == 0)
{
execlp("./q1l3","lab3_p1",NULL);
}

else
{
wait(NULL);
printf("Child complete\n");
exit(0);
```

}

return 0;
}

Output :



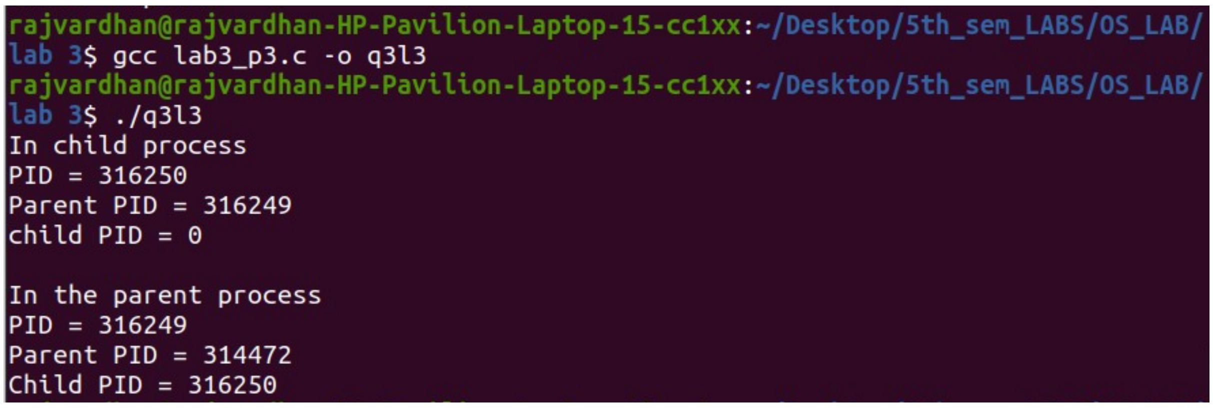
```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ gcc lab3_p2.c -o q2l3
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ ./q2l3
I'm the child process
I'm the parent process
Child returned status : 0
Child complete
```

P3) Write a program to create a child process. Display the process IDs of the process, parent and child (if any) in both the parent and child processes.

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<dirent.h>
#include<sys/types.h>
#include<sys/wait.h>
int main(int argc, char const *argv[])
{
pid_t pid;
pid = fork();if(pid < 0)
{
printf("Error creating child process\n");
exit(-1);
}
else if(pid == 0)
{
printf("In child process\n");
printf("PID = %d\n",getpid());
printf("Parent PID = %d\n",getppid());
printf("child PID = %d\n",pid);
}
else
{
wait(NULL);
printf("\nIn the parent process\n");
printf("PID = %d\n",getpid());
printf("Parent PID = %d\n",getppid());
printf("Child PID = %d\n",pid);
}
return 0;
}
```

Output :



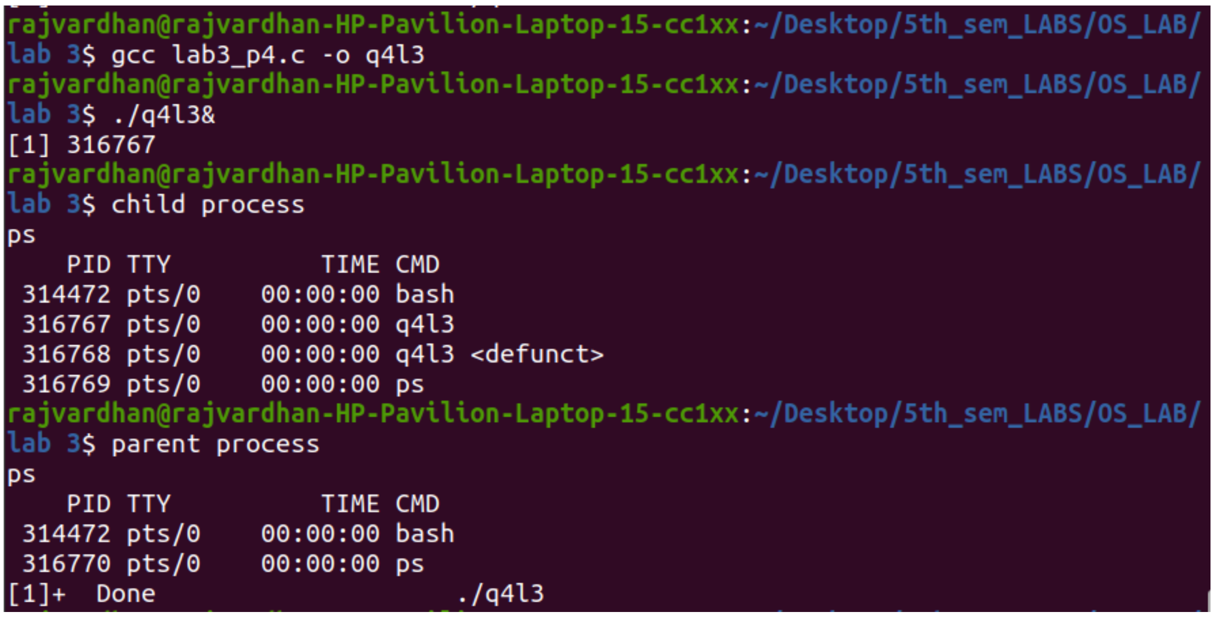
```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ gcc lab3_p3.c -o q3l3
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ ./q3l3
In child process
PID = 316250
Parent PID = 316249
child PID = 0

In the parent process
PID = 316249
Parent PID = 314472
Child PID = 316250
```

P4) Create a zombie (defunct) child process (a child with exit() call, but no corresponding wait() in the sleeping parent) and allow the init process to adopt it (after parent terminates). Run the process as a background process and run the “ps” command.

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(int argc, char const *argv[])
{
pid_t pid;
pid = fork();
if(pid < 0)
{
printf("Error\n");
exit(-1);
}
if(pid == 0)
{ //child process
printf("child process\n");
exit(0);
}
else
{ //parent process
sleep(2);
printf("parent process\n");
}
return 0;
}
```

Output :



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ gcc lab3_p4.c -o q4l3
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ ./q4l3&
[1] 316767
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ child process
ps
    PID TTY          TIME CMD
 314472 pts/0    00:00:00 bash
 316767 pts/0    00:00:00 q4l3
 316768 pts/0    00:00:00 q4l3 <defunct>
 316769 pts/0    00:00:00 ps
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 3$ parent process
ps
    PID TTY          TIME CMD
 314472 pts/0    00:00:00 bash
 316770 pts/0    00:00:00 ps
[1]+  Done                    ./q4l3
```