

OS Lab3

Name: Rhea Adhikari
Reg No: 190905156
Section CSE-D
Roll No: 23

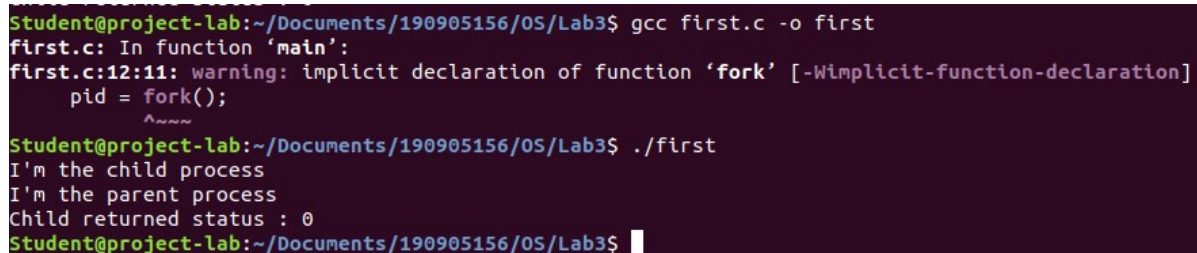
Q1) Write a C program to block a parent process until the child completes using a wait system call.

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

int main(int argc, char const *argv[])
{
    int st;
    pid_t pid;

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



```
Student@project-lab:~/Documents/190905156/OS/Lab3$ gcc first.c -o first
first.c: In function 'main':
first.c:12:11: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
    pid = fork();
            ^~~~~
Student@project-lab:~/Documents/190905156/OS/Lab3$ ./first
I'm the child process
I'm the parent process
Child returned status : 0
Student@project-lab:~/Documents/190905156/OS/Lab3$
```

Q2) Write a C program to load the binary executable of the previous program in a child process using the exec system call.

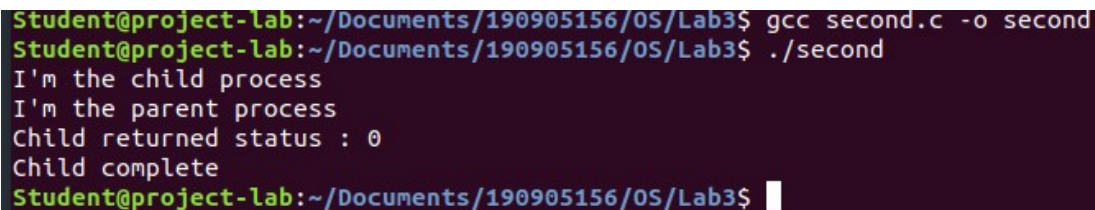
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/wait.h>
```

```

int main(int argc, char const *argv[])
{
    pid_t pid;
    pid = fork();
    if (pid < 0)
    {
        printf("Error in creating child process\n");
        exit(-1);
    }
    else if (pid == 0)
        execlp("./first", "first", NULL);
    else
    {
        wait(NULL);
        printf("Child complete\n");
        exit(0);
    }

    return 0;
}

```



```

Student@project-lab:~/Documents/190905156/05/Lab3$ gcc second.c -o second
Student@project-lab:~/Documents/190905156/05/Lab3$ ./second
I'm the child process
I'm the parent process
Child returned status : 0
Child complete
Student@project-lab:~/Documents/190905156/05/Lab3$

```

Q3)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.

```

#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");
    }
}

```

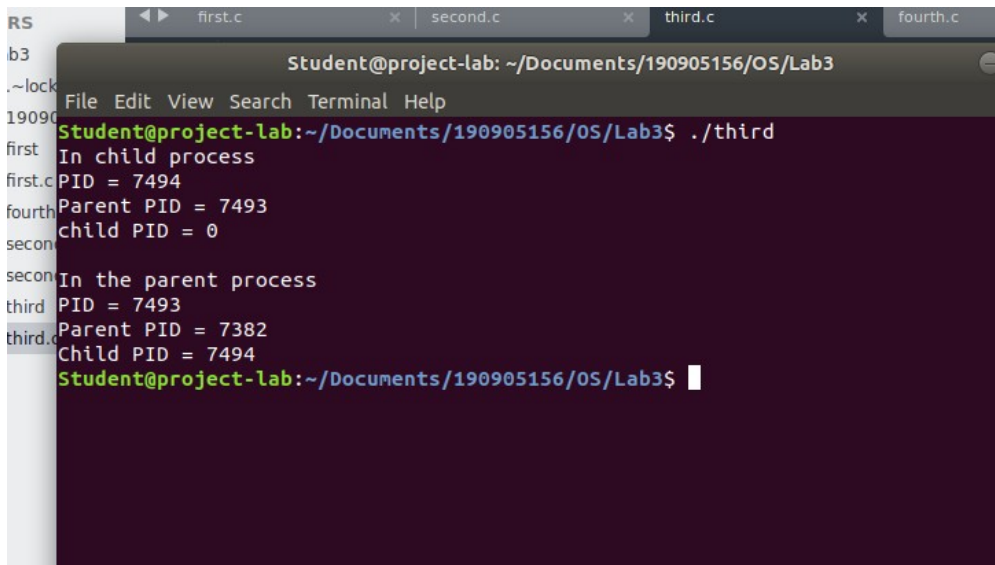
```

    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;
}

```



```

Student@project-lab: ~/Documents/190905156/OS/Lab3
File Edit View Search Terminal Help
Student@project-lab:~/Documents/190905156/OS/Lab3$ ./third
In child process
PID = 7494
Parent PID = 7493
child PID = 0
In the parent process
PID = 7493
Parent PID = 7382
Child PID = 7494
Student@project-lab:~/Documents/190905156/OS/Lab3$

```

Q4) 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.

```

#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)
{
    exit(0);
    printf("Child Process\n");
}
else
{
    sleep(2);
    printf("Parent Process\n");
}
return 0;
}
```

Entered complete

Student@project-lab:~/Documents/190905156/OS/Lab3\$ gcc fourth.c -o fourth

Student@project-lab:~/Documents/190905156/OS/Lab3\$./fourth

Parent Process

Student@project-lab:~/Documents/190905156/OS/Lab3\$