**190905514** **MOHAMMAD TOFIK** **BATCH : C3**
**ROLL NO : 62** **SECTION : C** **SEM : FIFTH**

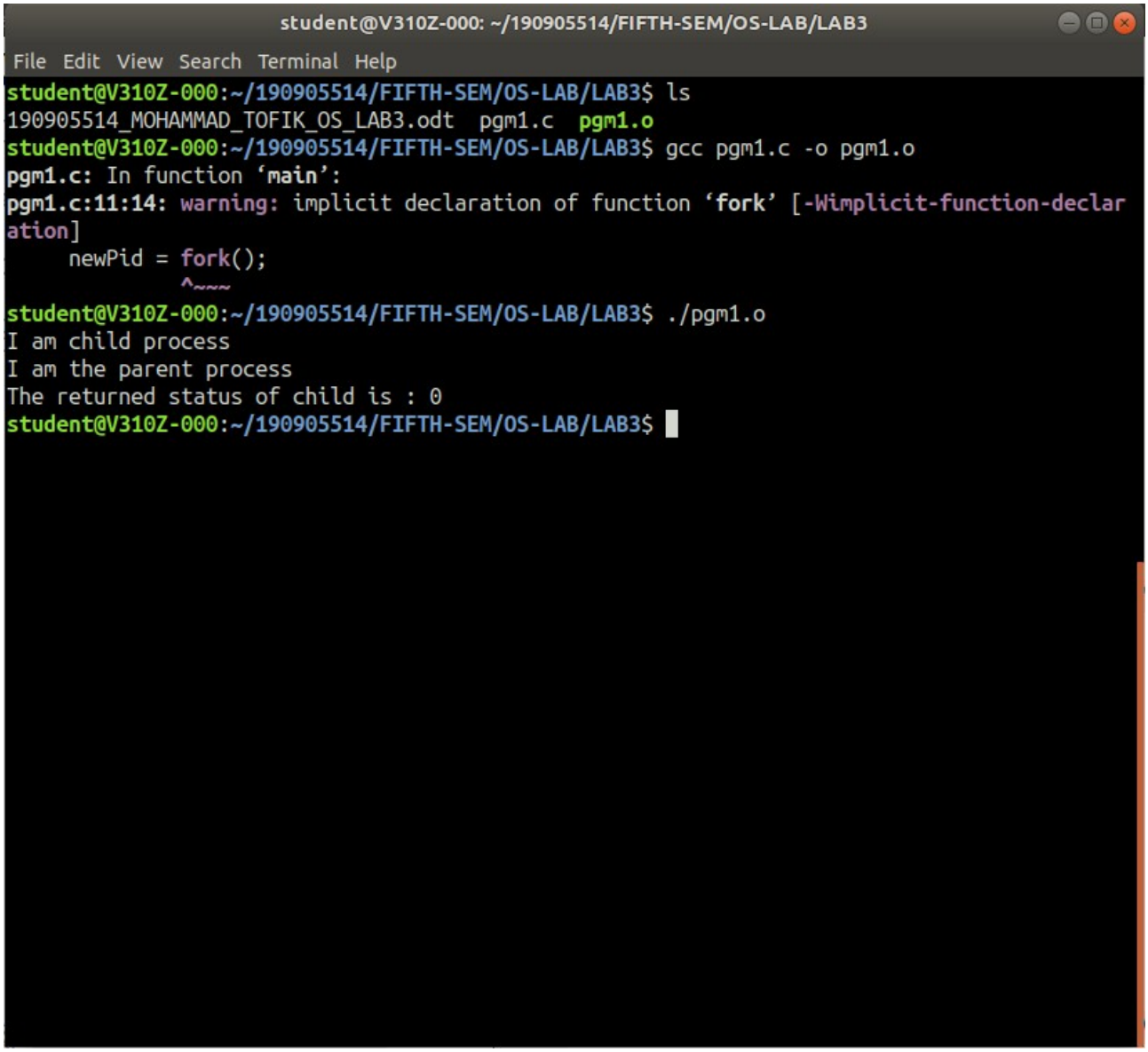
## - : OS-LAB-3 :-

**1.** write a c program to block a parent process untill the child completes using a wait system call.

**pgm1.c**

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

int main()
{
int status;
pid_t newPid;
newPid = fork();
if (newPid == -1)
{
printf("Error while creating child process\n");
}
else if (newPid == 0)
{
printf("I am child process\n");
exit(0);
}
else
{
wait(&status);
printf("I am the parent process\n");
printf("The returned status of child is : %d\n", status);
}
return 0;
}
```

## OUTPUT



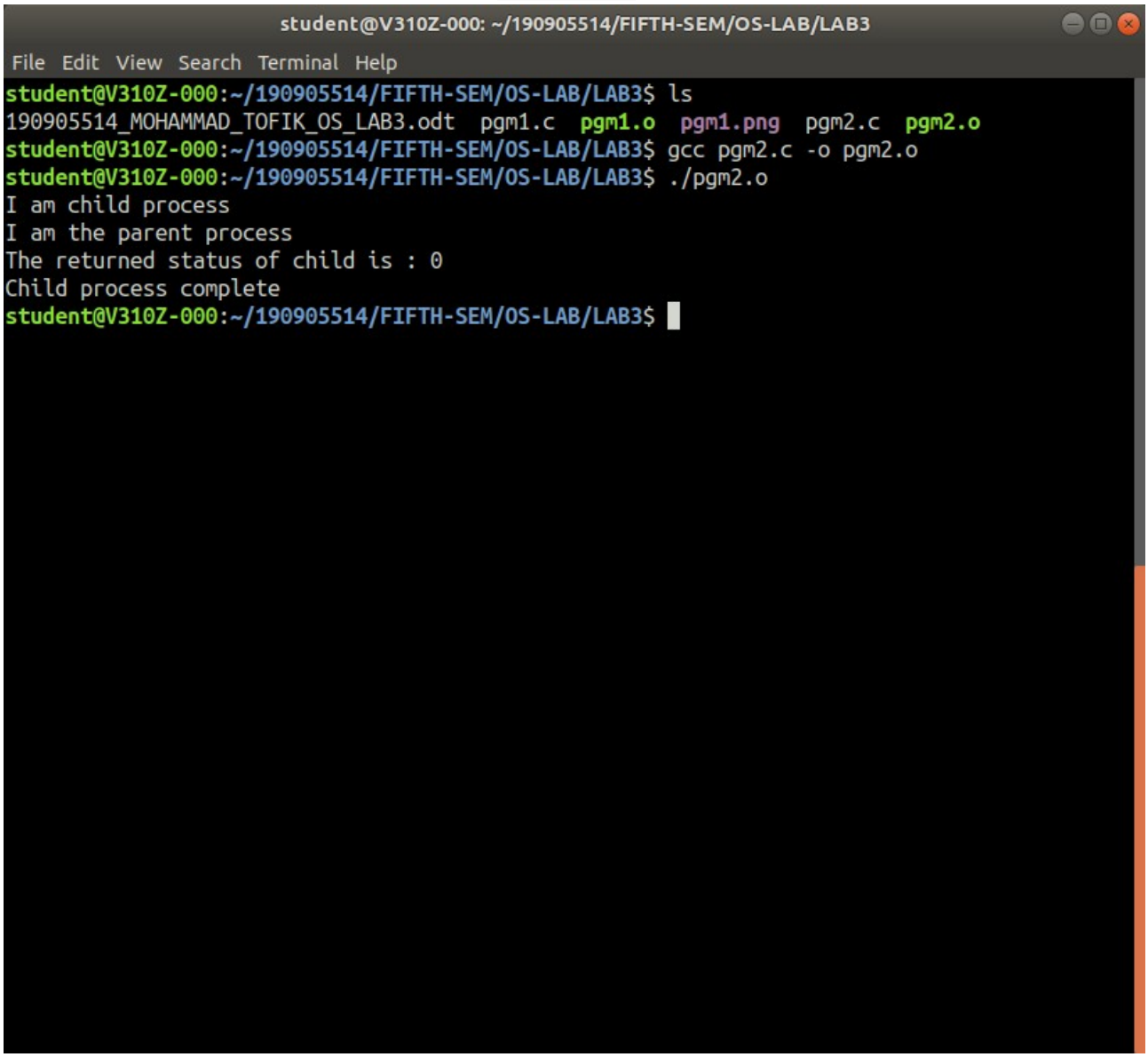

**2.**write a c rpogram to load the binary executable of the previous program in a child process using exec system call.

**pgm2.c**

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

```c
int main()
{
pid_t newPid;
newPid = fork();

if (newPid==-1)
{
fprintf(stderr, "Error while creating child process.\n");
exit(-1);
}

else if (newPid == 0)
{
execlp("./pgm1.o", "pgm1", NULL);
}

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

return 0;
}
```

## OUTPUT



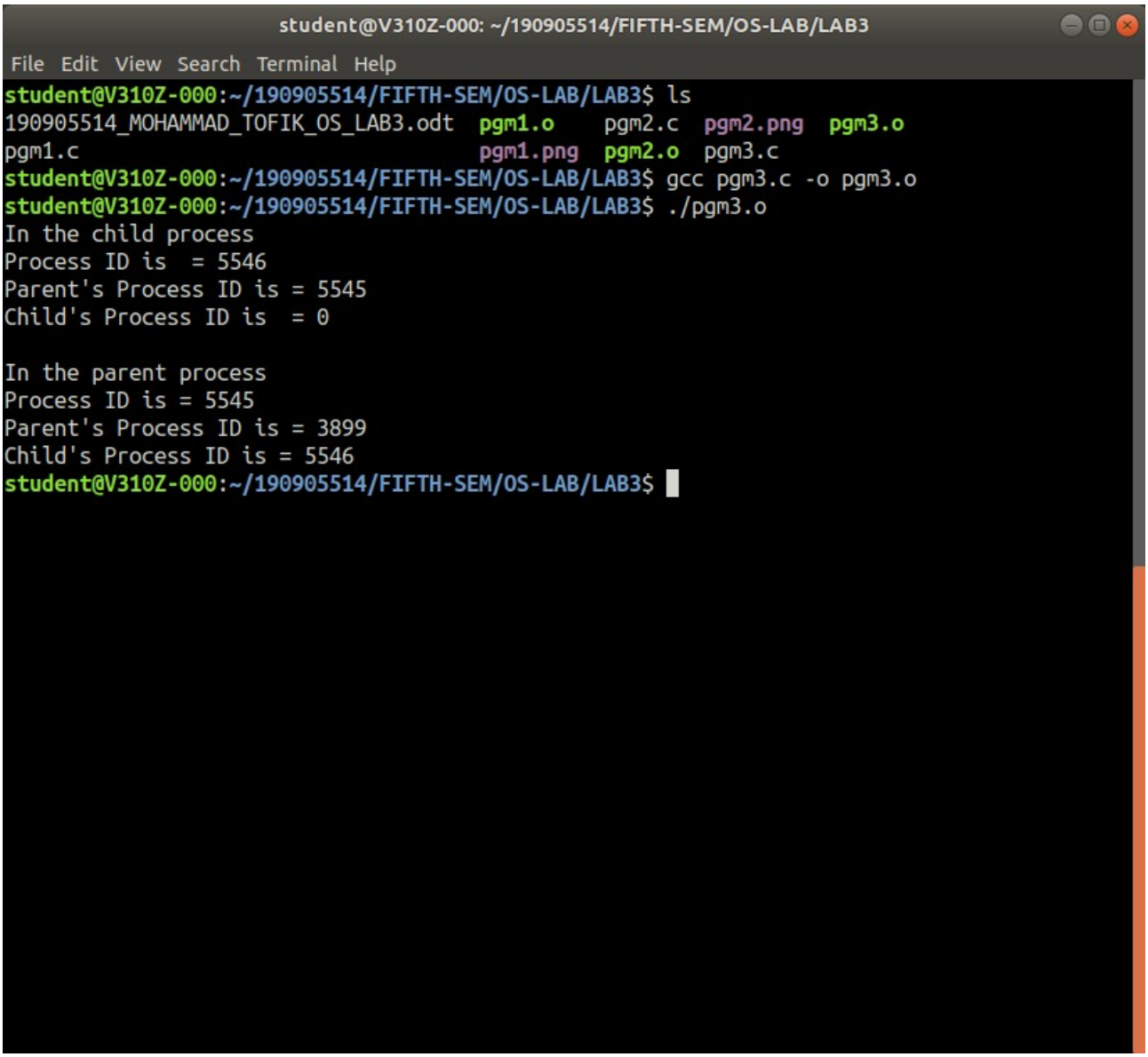

**3.**write a program to create a child process.Display the process Ids of the process,parent and child(if any)in both the parent and child processes.

**pgm3.c**

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

int main()
{
```

```
pid_t newPid;
newPid = fork();

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

else if (newPid == 0)
{
printf("In the child process\n");
printf("Process ID is = %d\n", getpid());
printf("Parent's Process ID is = %d\n", getppid());
printf("Child's Process ID is = %d\n", newPid);
}

else
{
wait(NULL);
printf("\nIn the parent process\n");
printf("Process ID is = %d\n", getpid());
printf("Parent's Process ID is = %d\n", getppid());
printf("Child's Process ID is = %d\n", newPid);
}

return 0;
}
```

## OUTPUT




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

**pgm4.c**

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(int argc, char const *argv[])
{
pid_t pid;
pid = fork();
if(pid == -1)
{
```

```
printf("Error while creating fork\n");
exit(-1);
}
if(pid == 0)
{
printf("child process\n");
exit(0);
}
else
{
sleep(2);
printf("parent process\n");
}
return 0;
}
```

## OUTPUT