Name : Tazmeen Afroz Roll No: 22P-9252 Section : BAI-5A

Lab Task 06

3.1.1 Exercise

The PID value for my first.c
The PPID value for my first.c

```
[Running] cd "/home/tazmeen/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06
22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/"first
total 24
-rw-rw-r-- 1 tazmeen tazmeen     0 Sep 27 08:16 1.c
-rwxrwxr-x 1 tazmeen tazmeen 16176 Sep 27 08:33 first
-rw-rw-r-- 1 tazmeen tazmeen   819 Sep 27 08:33 first.c
drwxrwxr-x 2 tazmeen tazmeen  4096 Aug 21 12:02 OS_LAB_06

This is C Programming in Ubuntu


Here in this program we will run linux commands

This is Parent process
Parent's PID: 5507
Child's PID: 5510
Parent ID of Parent: 5500

This is C Programming in Ubuntu


Here in this program we will run linux commands

This is Child process
Child's PID: 5510
Parent's PID: 5507
```

3.2.1

Q1 How many processes are created?

2 processes are created (1 main process)

```
[Running] cd "/home/tazmeen/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06
22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/"Q1_3.2
Process PID   6001    PPID   5994
Process PID   6001    PPID   5994
Process PID   6002    PPID   6001

[Done] exited with code=0 in 0.121 seconds
```

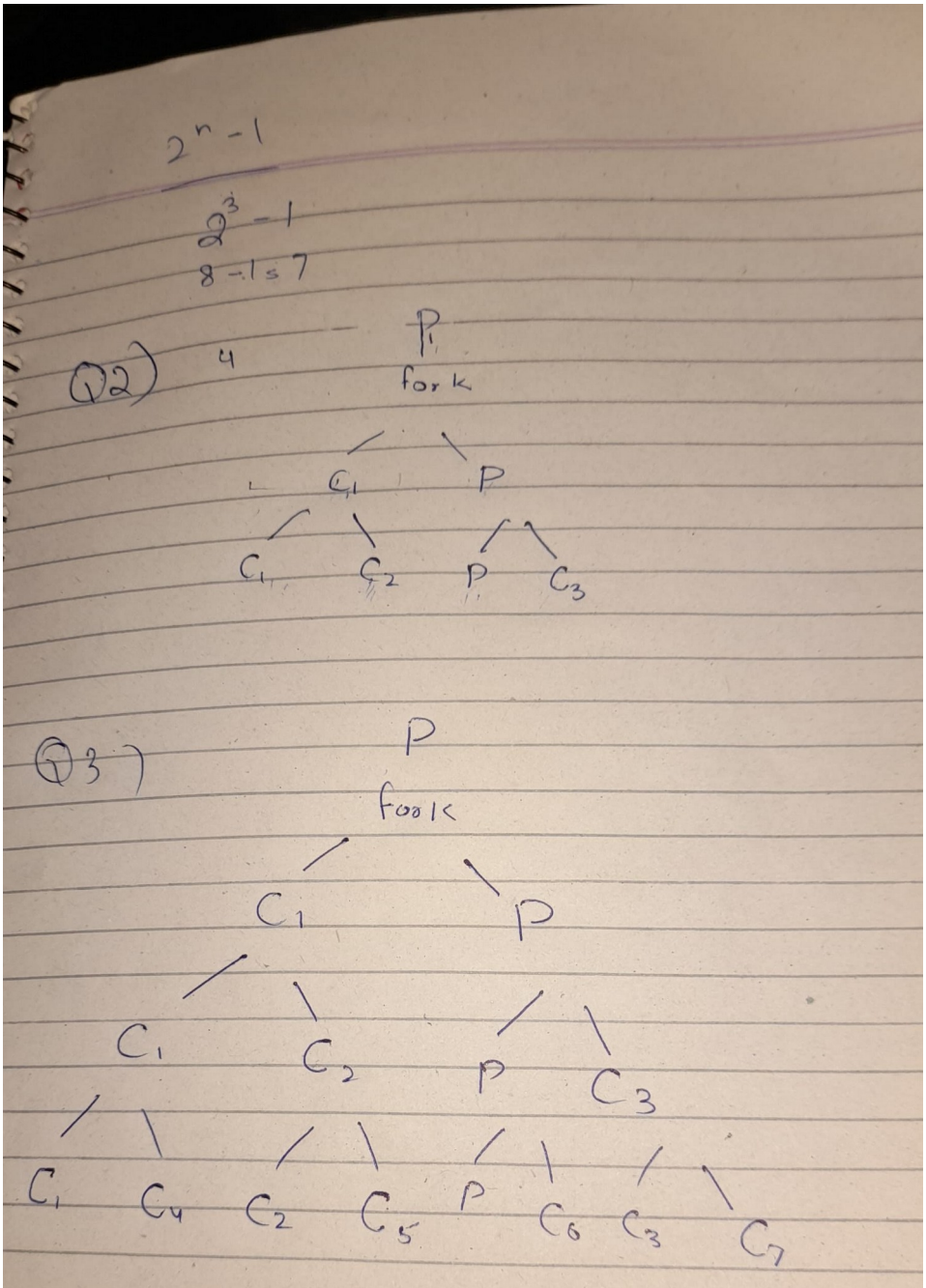
Q2 Increase the value in for loop from i<1 to i<2 (i.e., 2 iterations in the loop). Compile and run your program. How many processes does it show this time? Draw a tree hierarchy of processes that you just created as given in Figure-3.1.

Answer:

4 processes including main

3 processes without main

```
22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/"Q2_3.2
Process PID   6209    PPID   6203
Process PID   6209    PPID   6203
Process PID   6211    PPID   6209
Process PID   6209    PPID   6203
Process PID   6210    PPID   6209
Process PID   6209    PPID   6203
Process PID   6210    PPID   6209
Process PID   6212    PPID   6210
```

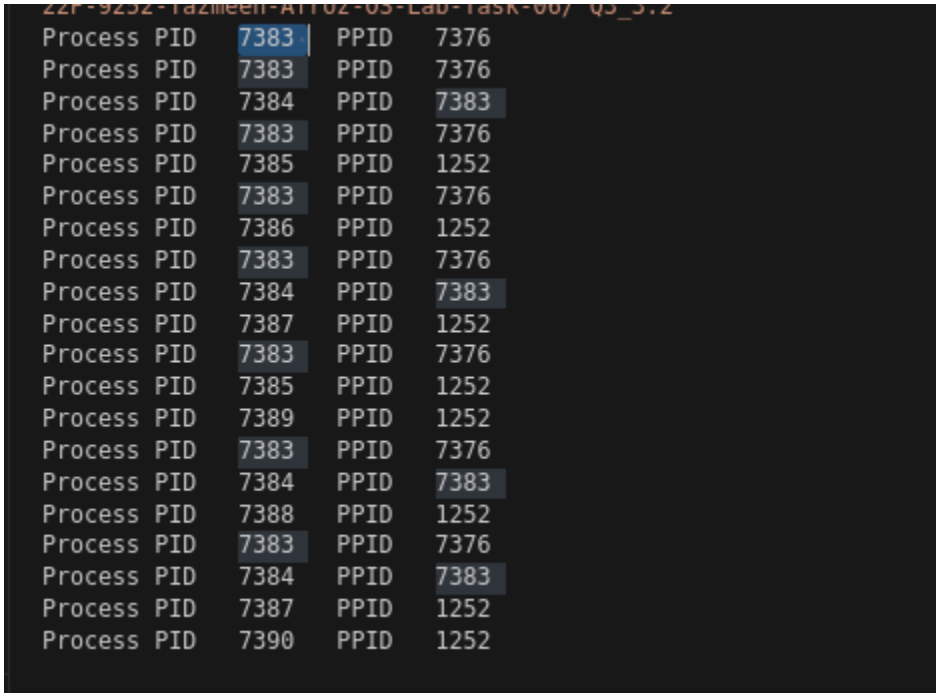


Q3 Increase the value again to i<3 (i.e., 3 iterations). Compile and run your program.

How many processes does it show? Draw a tree again. Why is it that we have called

fork() 3 times in our code, yet we are seeing 2n − 1 processes listed on screen?

Answer : 7 processes without main
8 processes with main



Q4 For fun, increase the value yet again to 100. Compile and run. What is going to happen? Does your OS Crash? Does your Program Crash? Can you modify your code to count the total number of fork()s made?

Answer : Program crash
yes

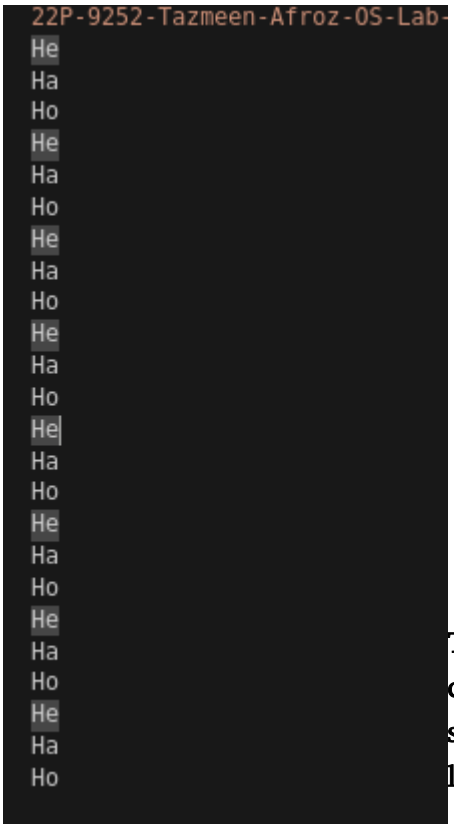
2.1 > Q4_3.2.c > main(void)

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

int fork_count = 0;

int main(void)
{
    int i;
    printf("Process PID %6d \t PPID %6d \n", getpid(), getppid());
    for (i = 0; i < 100; ++i)
    {
        if (fork() == 0)
        {
            fork_count++;
            printf("Process PID %6d \t PPID %6d \n", getpid(), getppid());
            if (fork_count == 3)

            exit(0);

        }
        else
        {
            fork_count++;
            if (fork_count == 3)
            {
                break;
            }
        }
    }
    printf("Total fork() calls: %d\n", fork_count);
    return 0;

}
```

(const char [29])"Process PID %6d \t PPID %6d \n"

```
[Running] cd "/home/tazmeen/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.1/" && g++ Q4_3.2.c -o Q4_3.2 && "/home/tazmeen/
22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.1/"Q4_3.2
Process PID   5134   PPID   5127
Process PID   5135   PPID   5134
Process PID   5134   PPID   5127
Total fork() calls: 3
Process PID   5134   PPID   5127
Process PID   5136   PPID   5134
Process PID   5134   PPID   5127
Process PID   5137   PPID   1299
Total fork() calls: 3

[Done] exited with code=0 in 0.199 seconds
```

Q5) Can a Ho be output before a He? Why?



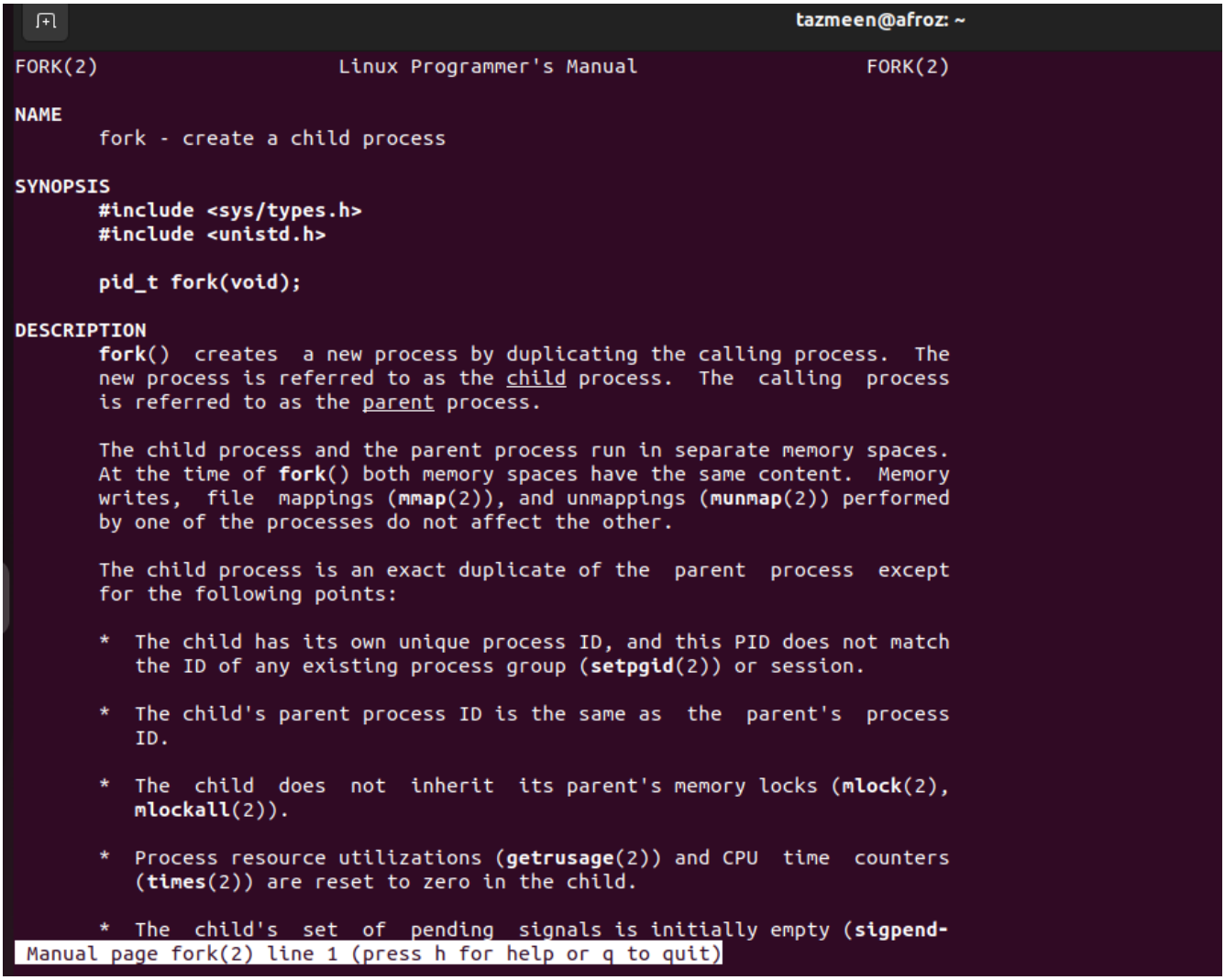
The output order of "He", "Ha", and "Ho" can vary due to the concurrent execution of processes. The operating system's scheduler determines the order in which processes run, which can lead to different sequences of output.

3.2.2

Question1 We have used p = fork(). Why not simply fork()? Check man fork for answer.

Using p = fork() allows the program to capture the return value of the fork() system call, which is crucial for determining whether the current process is the parent or the child.

-man fork command



Question2 Check man page for printf. What library is used for this call?

#include <stdio.h>

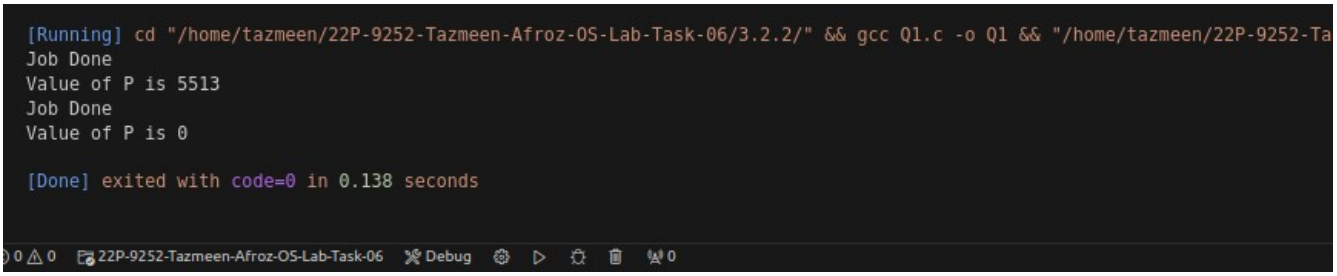
Question3 Run your program. Why is it that printf() is used only once, yet we see the output Job Done displaying twice on our screen.

When you call fork(), it creates a new child process that is a duplicate of the parent process. Since both processes execute the printf("Job Done\n");statement, you see "Job Done" printed twice.

Question4 Add the following statement to the end of your code and run it again.

What output would you see?

printf("Value of P is %d\n", p);



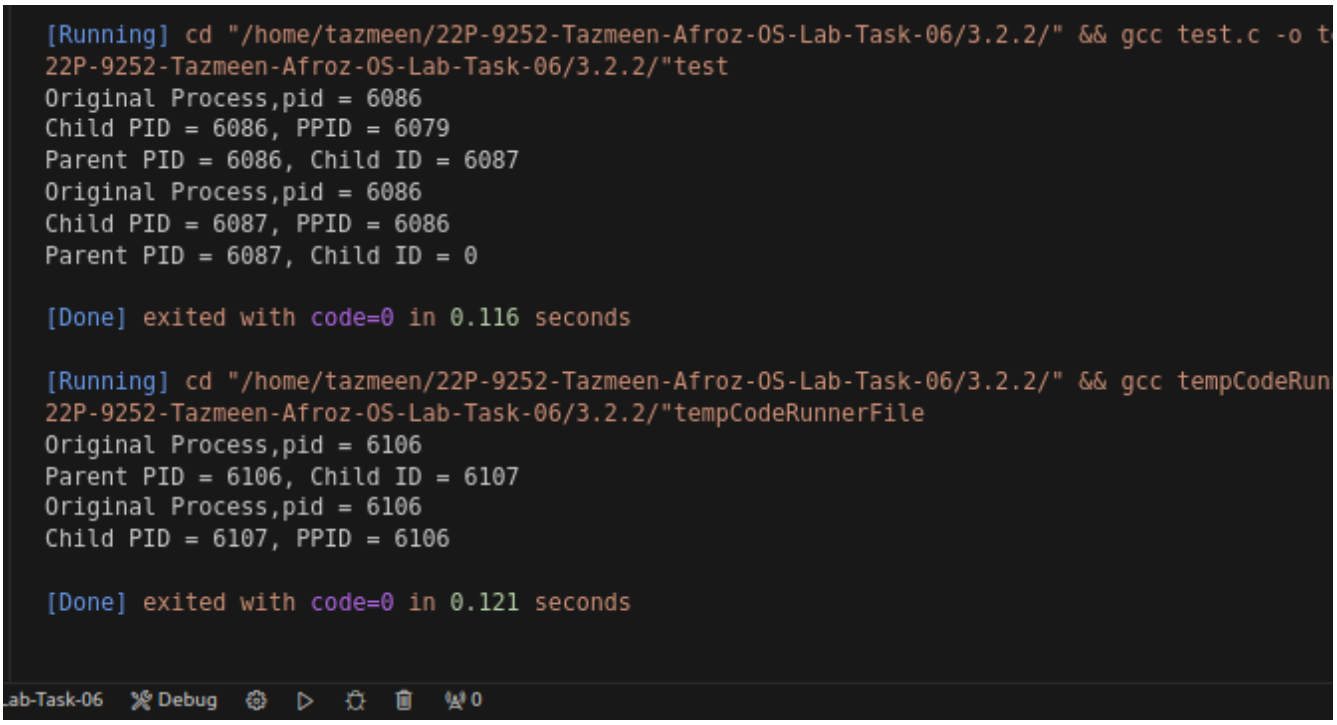
it returns the fork value stored in p

first is the parent process so it returns the child id 5513

second is the child process so it returns 0

What would happen if we don't use the If/Else conditions and immediately write the two printf() statements? Make sure you understand the structure of the Code, as well as what are the ways of knowing the following:

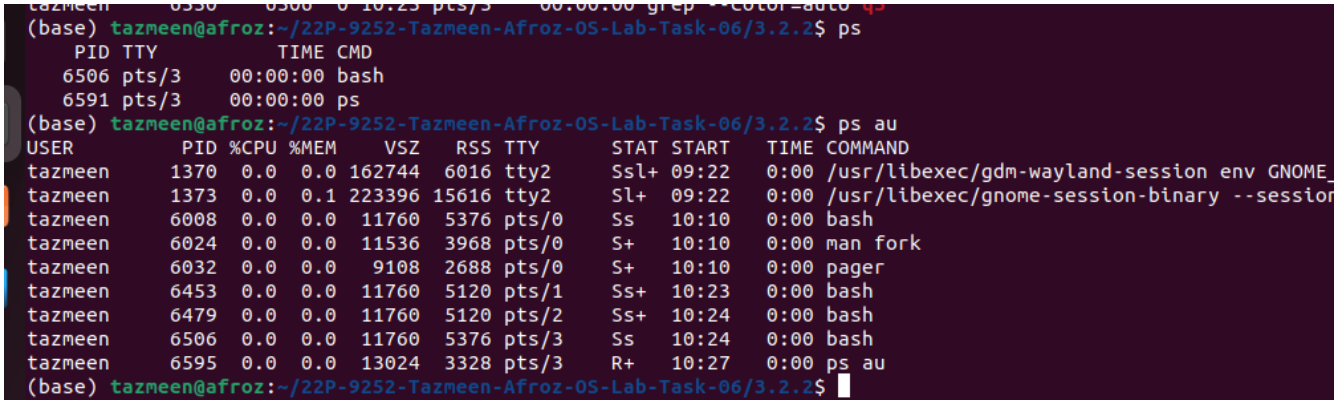
first is without conditions

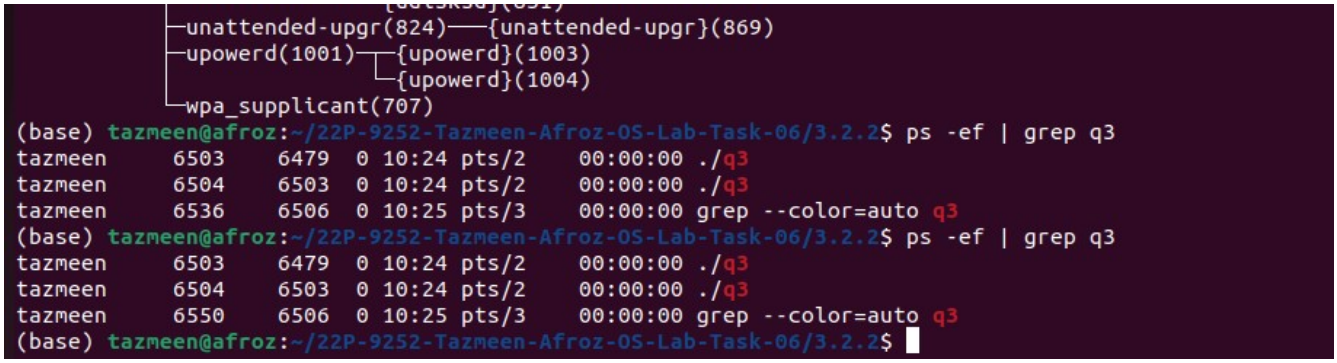


Answer : If you remove the if/els` conditions after a fork(), both the parent and child processes will execute the same code, including all printf() statements. This means both processes will print the same output. Like after one fork call it created the two processes one child one parent,

Exercise 1

ps statement





Both processes are in a sleeping state (S).