

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.

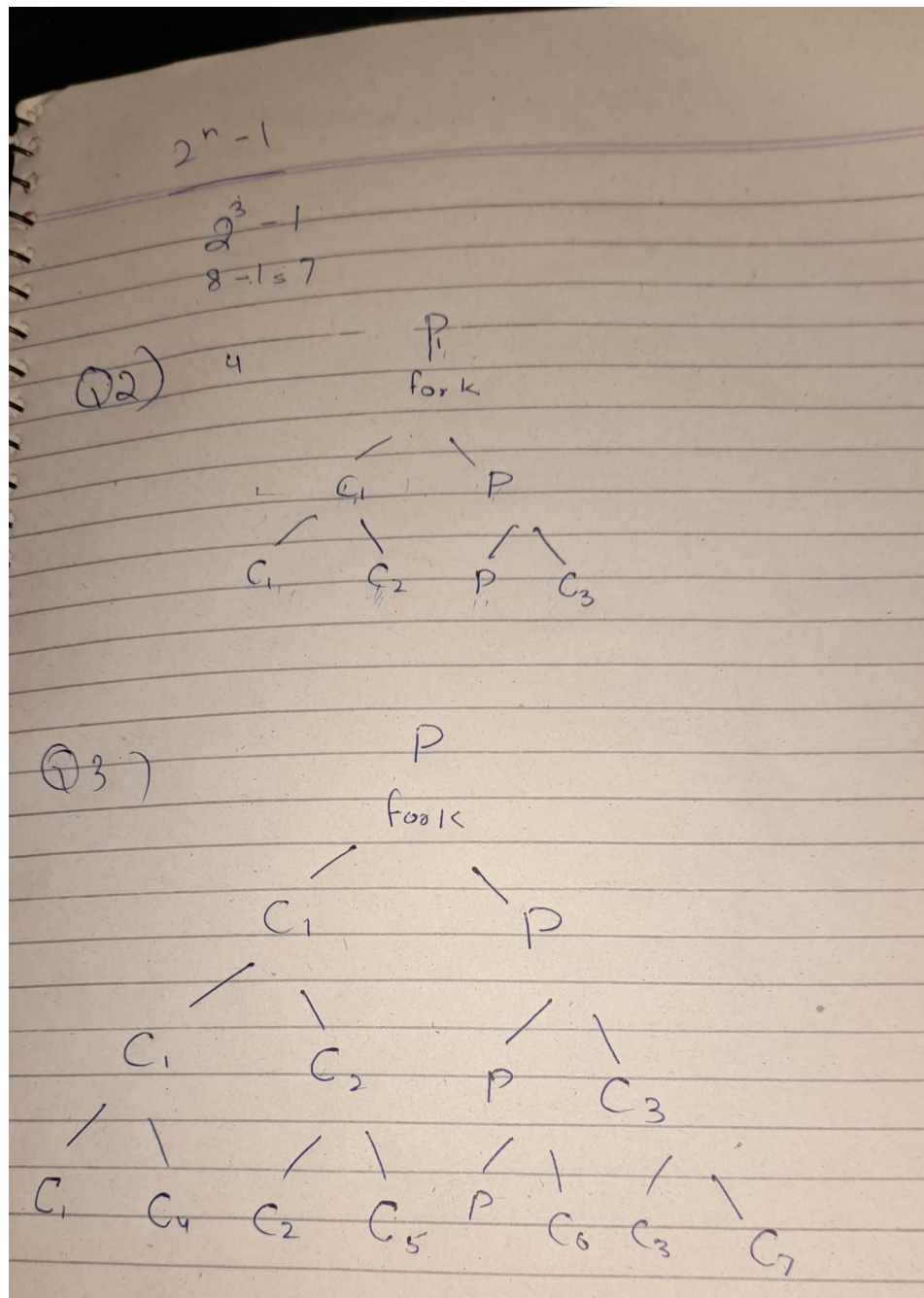
Answer:

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  $2^n - 1$  processes listed on screen?

Answer : 7 processes without main  
8 processes with main

```
22F-9252-Fazmeen-AT102-03-Lab-Task-00/ Q3_3.2
Process PID 7383 PPID 7376
Process PID 7383 PPID 7376
Process PID 7384 PPID 7383
Process PID 7383 PPID 7376
Process PID 7385 PPID 1252
Process PID 7383 PPID 7376
Process PID 7386 PPID 1252
Process PID 7383 PPID 7376
Process PID 7384 PPID 7383
Process PID 7387 PPID 1252
Process PID 7383 PPID 7376
Process PID 7385 PPID 1252
Process PID 7389 PPID 1252
Process PID 7383 PPID 7376
Process PID 7384 PPID 7383
Process PID 7388 PPID 1252
Process PID 7383 PPID 7376
Process PID 7384 PPID 7383
Process PID 7387 PPID 1252
Process PID 7390 PPID 1252
```

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 > G Q4_3.2.c > main(void)
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <unistd.h>
4  #include <stdlib.h>
5
6  int fork_count = 0;
7
8  int main(void)
9  {
10     int i;
11     printf("Process PID %6d \t PPID %6d \n", getpid(), getppid());
12     for (i = 0; i < 100; ++i)
13     {
14         if (fork() == 0)
15         {
16             fork_count++;
17             printf("Process PID %6d \t PPID %6d \n", getpid(), getppid());
18             if (fork_count == 3)
19                 exit(0);
20         }
21         else
22         {
23             fork_count++;
24             if (fork_count == 3)
25             {
26                 break;
27             }
28         }
29     }
30     printf("Total fork() calls: %d\n", fork_count);
31     return 0;
32 }

```

```

[Running] cd "/home/tazmeen/22P-9252-Tazmeen-Afroz-05-Lab-Task-06/3.2.1/" && g++ Q4_3.2.c -o Q4_3.2 && "/home/tazmeen/22P-9252-Tazmeen-Afroz-05-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?**

22P-9252-Tazmeen-Afroz-05-Lab-

**H**e  
H**a**  
**H**o  
He  
Ha  
Ho  
He  
Ha  
Ho  
He  
Ha  
Ho  
He  
Ha  
Ho  
He  
Ha  
Ho  
He  
Ha  
Ho  
He  
Ha  
Ho

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

```
tazmeen@afroz: ~  
FORK(2) Linux Programmer's Manual FORK(2)  
NAME  
    fork - create a child process  
SYNOPSIS  
    #include <sys/types.h>  
    #include <unistd.h>  
  
    pid_t fork(void);  
DESCRIPTION  
    fork() creates a new process by duplicating the calling process. The  
    new process is referred to as the child process. The calling process  
    is referred to as the parent process.  
  
    The child process and the parent process run in separate memory spaces.  
    At the time of fork() both memory spaces have the same content. Memory  
    writes, file mappings (mmap(2)), and unmappings (munmap(2)) performed  
    by one of the processes do not affect the other.  
  
    The child process is an exact duplicate of the parent process except  
    for the following points:  
  
    * The child has its own unique process ID, and this PID does not match  
      the ID of any existing process group (setpgid(2)) or session.  
  
    * The child's parent process ID is the same as the parent's process  
      ID.  
  
    * The child does not inherit its parent's memory locks (mlock(2),  
      mlockall(2)).  
  
    * Process resource utilizations (getrusage(2)) and CPU time counters  
      (times(2)) are reset to zero in the child.  
  
    * The child's set of pending signals is initially empty (sigpend-  
Manual page fork(2) line 1 (press h for help or q to quit)
```

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);
```

```
[Running] cd "/home/tazmeen/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2/" && gcc Q1.c -o Q1 && "/home/tazmeen/22P-9252-Ta
Job Done
Value of P is 5513
Job Done
Value of P is 0

[Done] exited with code=0 in 0.138 seconds
```

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

```
[Running] cd "/home/tazmeen/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2/" && gcc test.c -o t
22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2/"test
Original Process,pid = 6086
Child PID = 6086, PPID = 6079
Parent PID = 6086, Child ID = 6087
Original Process,pid = 6086
Child PID = 6087, PPID = 6086
Parent PID = 6087, Child ID = 0

[Done] exited with code=0 in 0.116 seconds

[Running] cd "/home/tazmeen/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2/" && gcc tempCodeRun
22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2/"tempCodeRunnerFile
Original Process,pid = 6106
Parent PID = 6106, Child ID = 6107
Original Process,pid = 6106
Child PID = 6107, PPID = 6106

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

Answer : If you remove the if/else` 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

```
tazmeen 6550 0.0 0.0 0 10:23 pts/3 00:00:00 grep --color=auto q3
(base) tazmeen@afroz:~/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2$ ps
  PID TTY          TIME CMD
  6506 pts/3    00:00:00 bash
  6591 pts/3    00:00:00 ps
(base) tazmeen@afroz:~/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2$ ps au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
tazmeen   1370  0.0  0.0 162744 6016 tty2      Ssl+  09:22   0:00 /usr/libexec/gdm-wayland-session env GNOME_
tazmeen   1373  0.0  0.1 223396 15616 tty2      Sl+   09:22   0:00 /usr/libexec/gnome-session-binary --session
tazmeen   6008  0.0  0.0  11760  5376 pts/0     Ss   10:10   0:00 bash
tazmeen   6024  0.0  0.0  11536  3968 pts/0     S+   10:10   0:00 man fork
tazmeen   6032  0.0  0.0   9108  2688 pts/0     S+   10:10   0:00 pager
tazmeen   6453  0.0  0.0  11760  5120 pts/1     Ss+  10:23   0:00 bash
tazmeen   6479  0.0  0.0  11760  5120 pts/2     Ss+  10:24   0:00 bash
tazmeen   6506  0.0  0.0  11760  5376 pts/3     Ss   10:24   0:00 bash
tazmeen   6595  0.0  0.0  13024  3328 pts/3     R+   10:27   0:00 ps au
(base) tazmeen@afroz:~/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2$
```

```
└─unattended-upgr(824)───{unattended-upgr}(869)
└─upowerd(1001)───{upowerd}(1003)
                  └─{upowerd}(1004)
└─wpa_supplicant(707)

(base) tazmeen@afroz:~/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2$ ps -ef | grep q3
tazmeen   6503      6479  0 10:24 pts/2    00:00:00 ./q3
tazmeen   6504      6503  0 10:24 pts/2    00:00:00 ./q3
tazmeen   6536      6506  0 10:25 pts/3    00:00:00 grep --color=auto q3
(base) tazmeen@afroz:~/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2$ ps -ef | grep q3
tazmeen   6503      6479  0 10:24 pts/2    00:00:00 ./q3
tazmeen   6504      6503  0 10:24 pts/2    00:00:00 ./q3
tazmeen   6550      6506  0 10:25 pts/3    00:00:00 grep --color=auto q3
(base) tazmeen@afroz:~/22P-9252-Tazmeen-Afroz-OS-Lab-Task-06/3.2.2$
```

Both processes are in a sleeping state (S).



