PROCESS CREATION AND EXECUTION

LAB # 6

Spring 2020

CSE204L Operating Systems Lab

Submitted by: **SHAH RAZA**

Registration No. : **18PWCSE1658**

Class Section: **B**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _______________

Submitted to:

Engr. Mian Ibad Ali Shah

Wednesday, 24 June, 2020

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

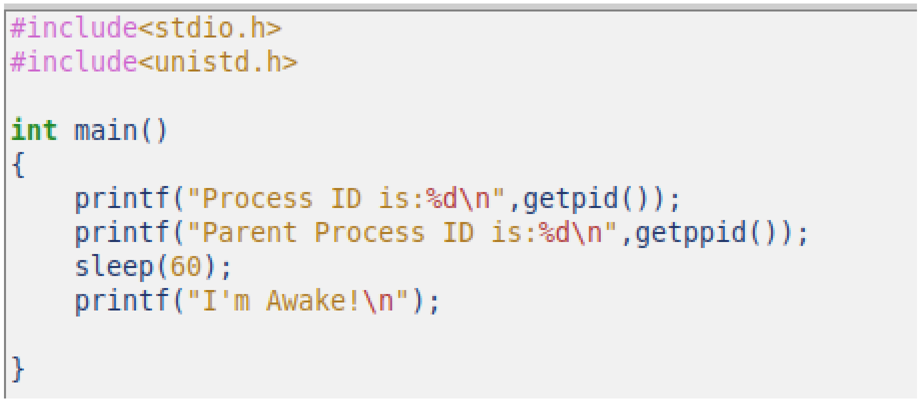
Lab Objective(s):

This lab describes how a program can create, terminate, and control child processes. Actually, there are a few distinct operations involved: **creating a new child process**, and **coordinating the completion of the child process with the original program**.

Task # 01:

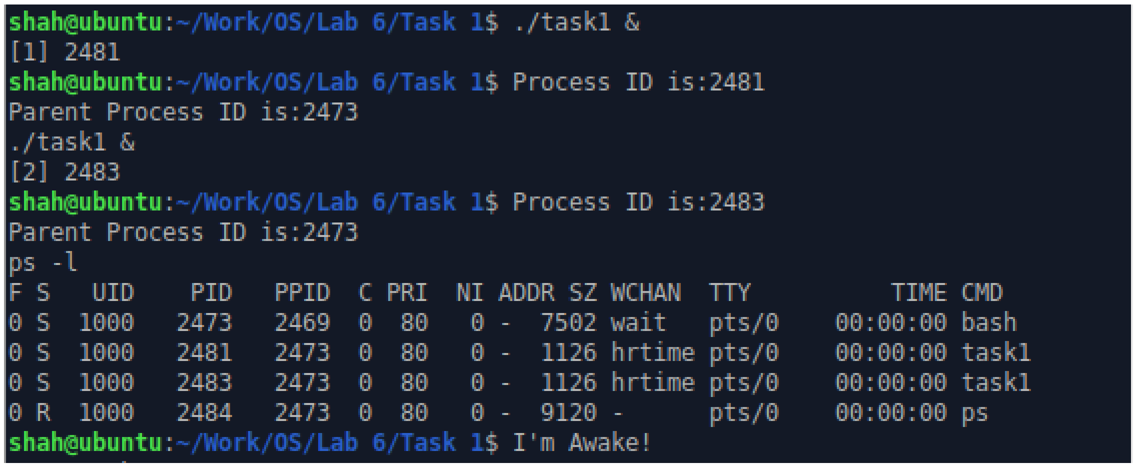
Run the following program twice. Both times as a background process, i.e., suffix it with an ampersand "**&**". Once both processes are running as background processes, view the *process table* using **ps -l** UNIX command. Observe the *process state*, *PID (process ID)* etc. Repeat this experiment to observe the changes, if any. Write your observation about the Process ID and state of the process.

Code:



```
#include<stdio.h>
#include<unistd.h>

int main()
{
    printf("Process ID is:%d\n",getpid());
    printf("Parent Process ID is:%d\n",getppid());
    sleep(60);
    printf("I'm Awake!\n");

}
```

Output :



```
shah@ubuntu:~/Work/OS/Lab 6/Task 1$ ./task1 &
[1] 2481
shah@ubuntu:~/Work/OS/Lab 6/Task 1$ Process ID is:2481
Parent Process ID is:2473
./task1 &
[2] 2483
shah@ubuntu:~/Work/OS/Lab 6/Task 1$ Process ID is:2483
Parent Process ID is:2473
ps -l
F S   UID    PID   PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000   2473   2469  0  80   0 -  7502 wait   pts/0    00:00:00 bash
0 S  1000   2481   2473  0  80   0 -  1126 hrtime pts/0    00:00:00 task1
0 S  1000   2483   2473  0  80   0 -  1126 hrtime pts/0    00:00:00 task1
0 R  1000   2484   2473  0  80   0 -  9120 -      pts/0    00:00:00 ps
shah@ubuntu:~/Work/OS/Lab 6/Task 1$ I'm Awake!
```

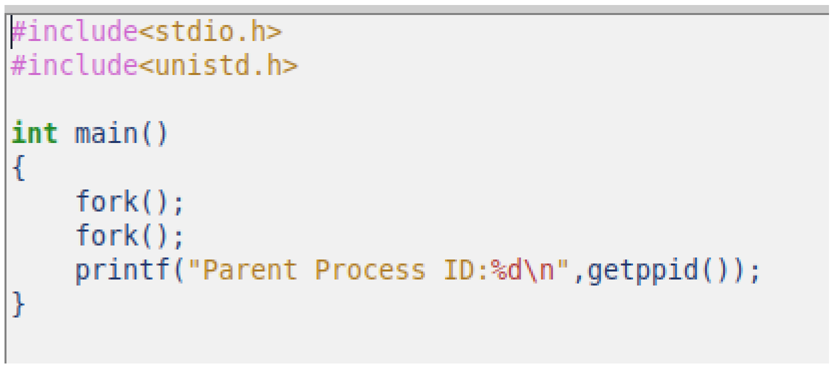
Observations:

By running a process in the background, we can enter other shell commands in the terminal. As asked by the Course Instructor, we run this task twice in the background and after that we see the currently running processes with the help of ps -l command. By observing we see that both the Processes are in Sleep State (S) and we can also see that even though we are running the same task twice but each time it has a different Process ID.

Task # 02:

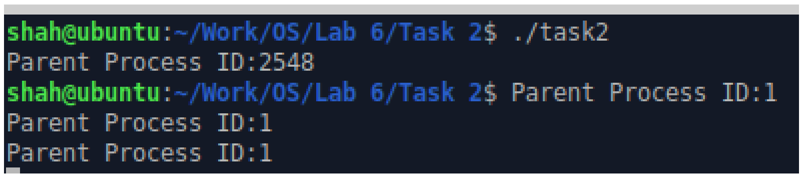
Run the following program and observe the *number of times* and the *order* in which the print statement is executed. The **fork( )** creates a child that is a duplicate of the parent process. The child process begins from the **fork( )**. All the statements after the call to **fork ( )** are executed by the parent process and also by the child process. Draw a family tree of processes and explain the results you observed.

Code:



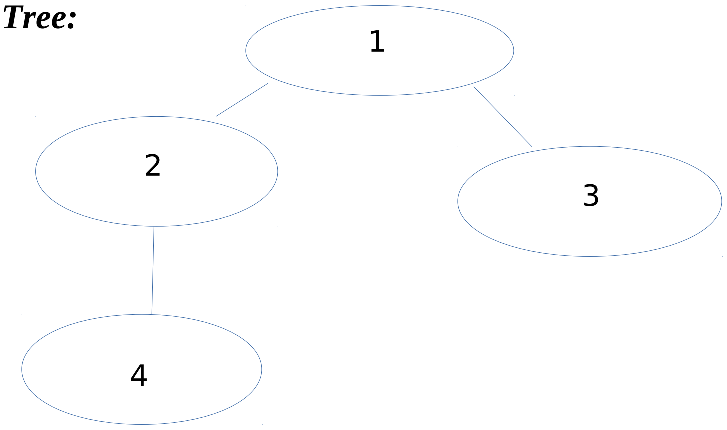
```
#include<stdio.h>
#include<unistd.h>

int main()
{
    fork();
    fork();
    printf("Parent Process ID:%d\n",getppid());
}
```

Output :



```
shah@ubuntu:~/Work/OS/Lab 6/Task 2$ ./task2
Parent Process ID:2548
shah@ubuntu:~/Work/OS/Lab 6/Task 2$ Parent Process ID:1
Parent Process ID:1
Parent Process ID:1
```

Process Tree:



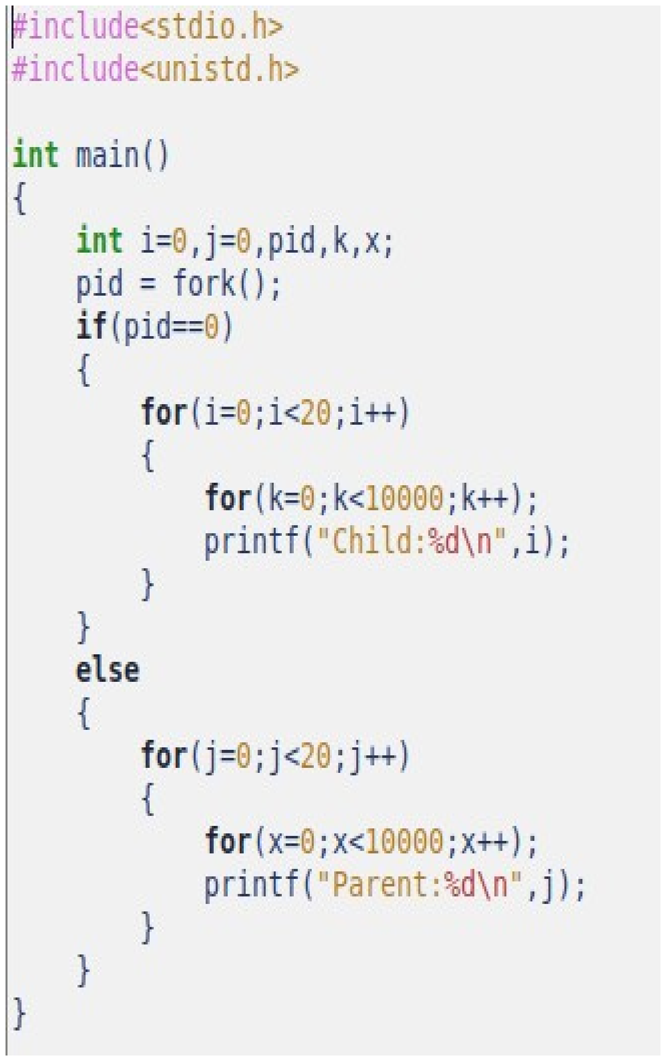
Observations:

By running this task, we observe that the print statement is executed 4 times. One time each by Process 1(Parent), Process 2(Child of P1), Process 3(Child of P1) and Process 4(Child of P2). P1 terminates first so P2 and P3 becomes Orphan Processes after that when P2 terminates, P4 becomes an Orphan Process. All these Orphan Processes are then adopted by the init Process (ID=1).

Task # 03:

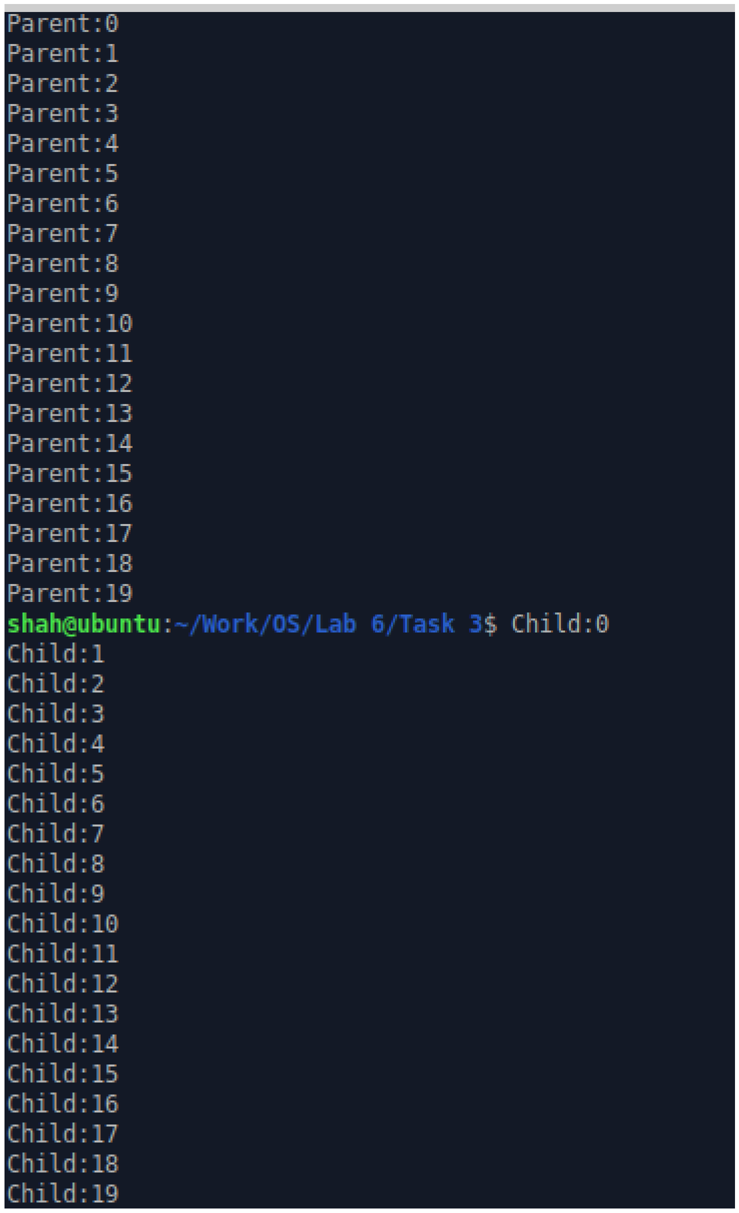
Run the following program and observe the result of **time slicing** used by UNIX.

Code:



```c
#include<stdio.h>
#include<unistd.h>

int main()
{
    int i=0,j=0,pid,k,x;
    pid = fork();
    if(pid==0)
    {
        for(i=0;i<20;i++)
        {
            for(k=0;k<10000;k++);
            printf("Child:%d\n",i);
        }
    }
    else
    {
        for(j=0;j<20;j++)
        {
            for(x=0;x<10000;x++);
            printf("Parent:%d\n",j);
        }
    }
}
```

Output :



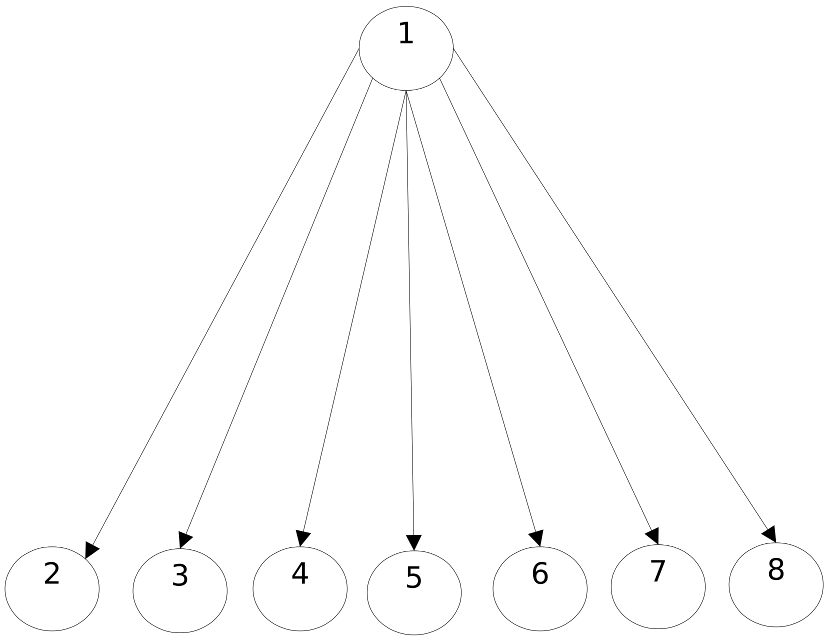
```
Parent:0
Parent:1
Parent:2
Parent:3
Parent:4
Parent:5
Parent:6
Parent:7
Parent:8
Parent:9
Parent:10
Parent:11
Parent:12
Parent:13
Parent:14
Parent:15
Parent:16
Parent:17
Parent:18
Parent:19
shah@ubuntu:~/Work/OS/Lab 6/Task 3$ Child:0
Child:1
Child:2
Child:3
Child:4
Child:5
Child:6
Child:7
Child:8
Child:9
Child:10
Child:11
Child:12
Child:13
Child:14
Child:15
Child:16
Child:17
Child:18
Child:19
```

Observations:

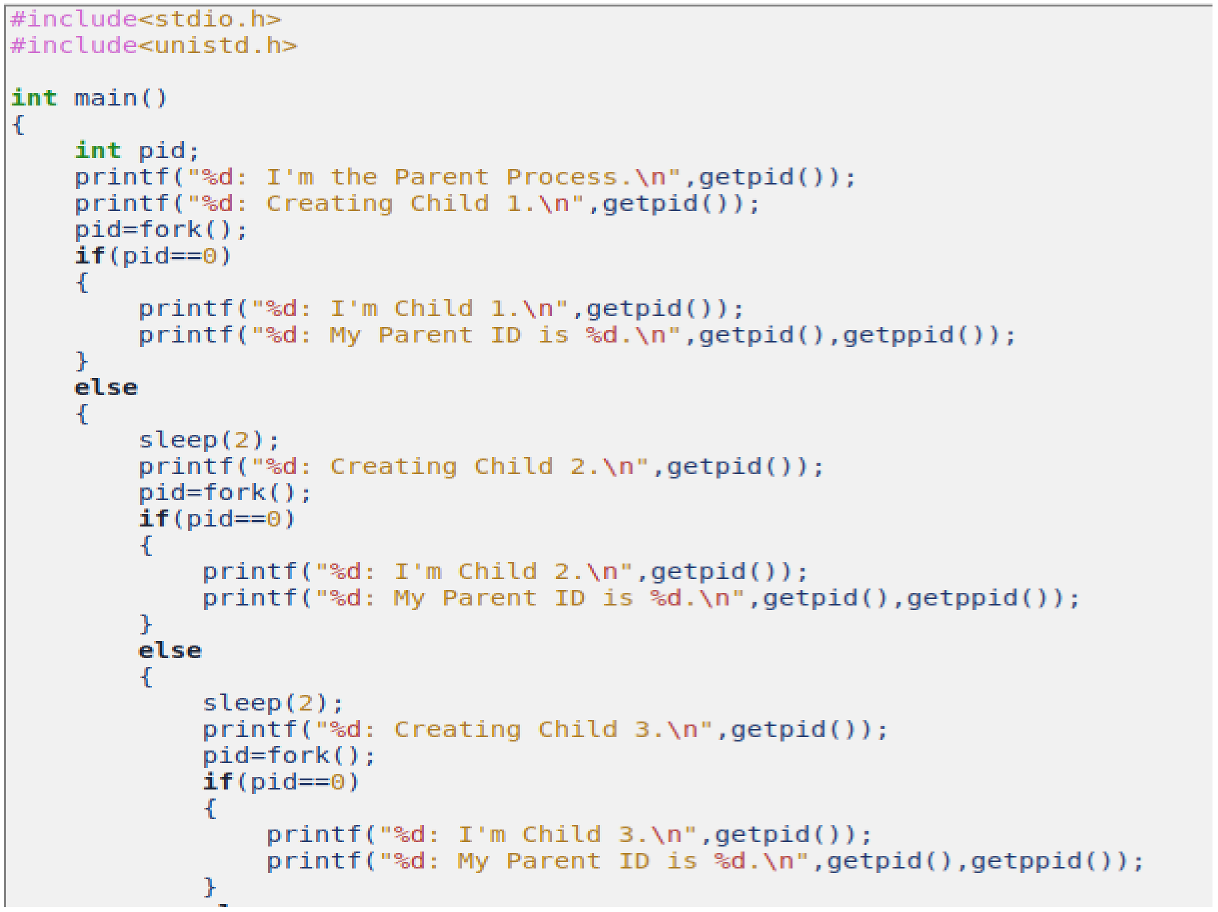
In this task a time slicing technique is used to prevent the termination of Child Process before the Parent Process.

Task # 04:

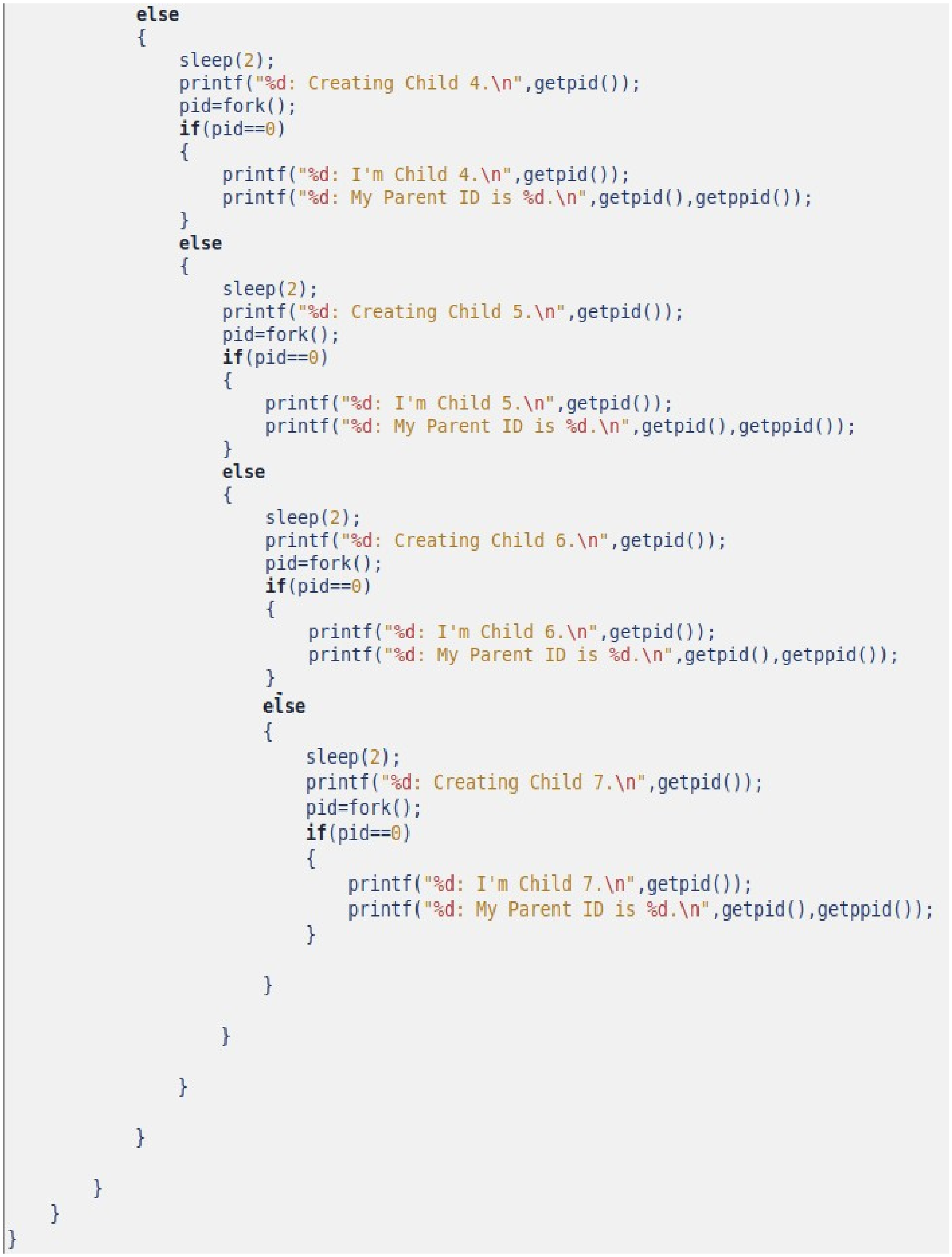
Create process fan as shown in figure 1 (a) and fill the figure 1 (a) with actual IDs.



Code:

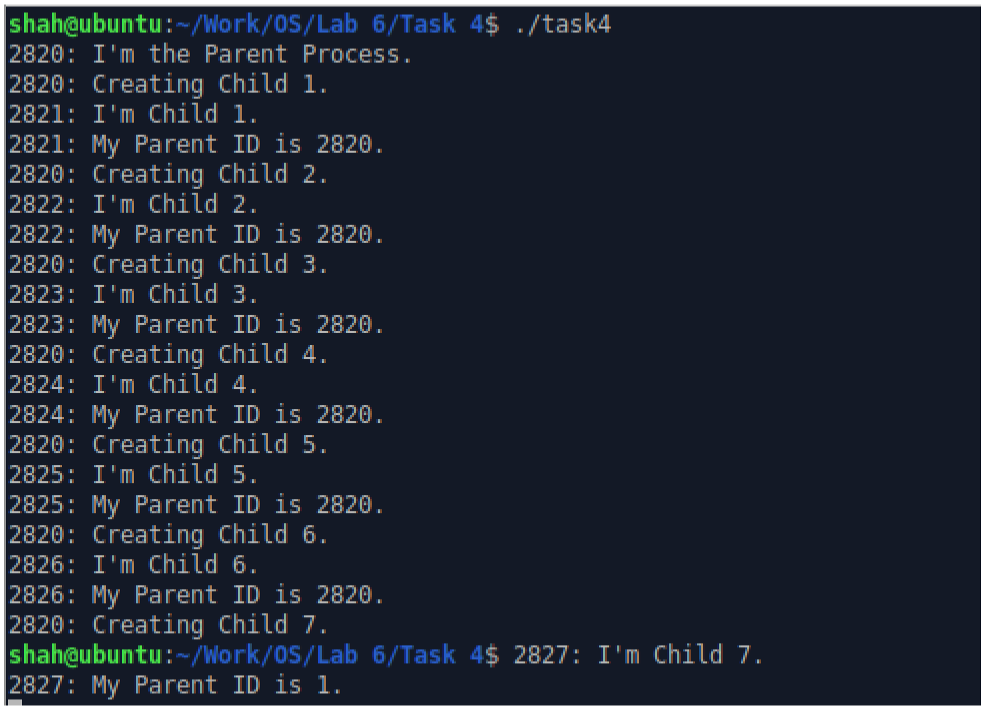


```c
#include<stdio.h>
#include<unistd.h>

int main()
{
    int pid;
    printf("%d: I'm the Parent Process.\n",getpid());
    printf("%d: Creating Child 1.\n",getpid());
    pid=fork();
    if(pid==0)
    {
        printf("%d: I'm Child 1.\n",getpid());
        printf("%d: My Parent ID is %d.\n",getpid(),getppid());
    }
    else
    {
        sleep(2);
        printf("%d: Creating Child 2.\n",getpid());
        pid=fork();
        if(pid==0)
        {
            printf("%d: I'm Child 2.\n",getpid());
            printf("%d: My Parent ID is %d.\n",getpid(),getppid());
        }
        else
        {
            sleep(2);
            printf("%d: Creating Child 3.\n",getpid());
            pid=fork();
            if(pid==0)
            {
                printf("%d: I'm Child 3.\n",getpid());
                printf("%d: My Parent ID is %d.\n",getpid(),getppid());
            }
```



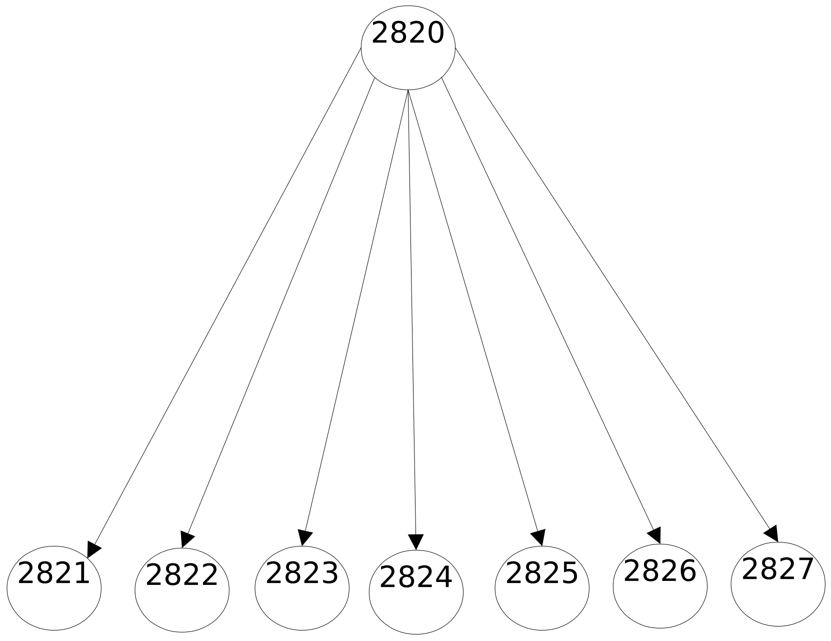
```c
            else
            {
                sleep(2);
                printf("%d: Creating Child 4.\n",getpid());
                pid=fork();
                if(pid==0)
                {
                    printf("%d: I'm Child 4.\n",getpid());
                    printf("%d: My Parent ID is %d.\n",getpid(),getppid());
                }
                else
                {
                    sleep(2);
                    printf("%d: Creating Child 5.\n",getpid());
                    pid=fork();
                    if(pid==0)
                    {
                        printf("%d: I'm Child 5.\n",getpid());
                        printf("%d: My Parent ID is %d.\n",getpid(),getppid());
                    }
                    else
                    {
                        sleep(2);
                        printf("%d: Creating Child 6.\n",getpid());
                        pid=fork();
                        if(pid==0)
                        {
                            printf("%d: I'm Child 6.\n",getpid());
                            printf("%d: My Parent ID is %d.\n",getpid(),getppid());
                        }
                        else
                        {
                            sleep(2);
                            printf("%d: Creating Child 7.\n",getpid());
                            pid=fork();
                            if(pid==0)
                            {
                                printf("%d: I'm Child 7.\n",getpid());
                                printf("%d: My Parent ID is %d.\n",getpid(),getppid());
                            }

                        }

                    }

                }

            }

        }
    }
}
```

Output :



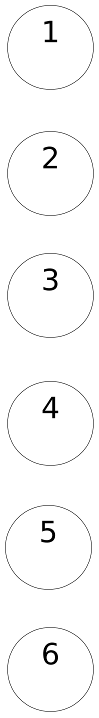
```
shah@ubuntu:~/Work/OS/Lab 6/Task 4$ ./task4
2820: I'm the Parent Process.
2820: Creating Child 1.
2821: I'm Child 1.
2821: My Parent ID is 2820.
2820: Creating Child 2.
2822: I'm Child 2.
2822: My Parent ID is 2820.
2820: Creating Child 3.
2823: I'm Child 3.
2823: My Parent ID is 2820.
2820: Creating Child 4.
2824: I'm Child 4.
2824: My Parent ID is 2820.
2820: Creating Child 5.
2825: I'm Child 5.
2825: My Parent ID is 2820.
2820: Creating Child 6.
2826: I'm Child 6.
2826: My Parent ID is 2820.
2820: Creating Child 7.
shah@ubuntu:~/Work/OS/Lab 6/Task 4$ 2827: I'm Child 7.
2827: My Parent ID is 1.
```

Process Tree:

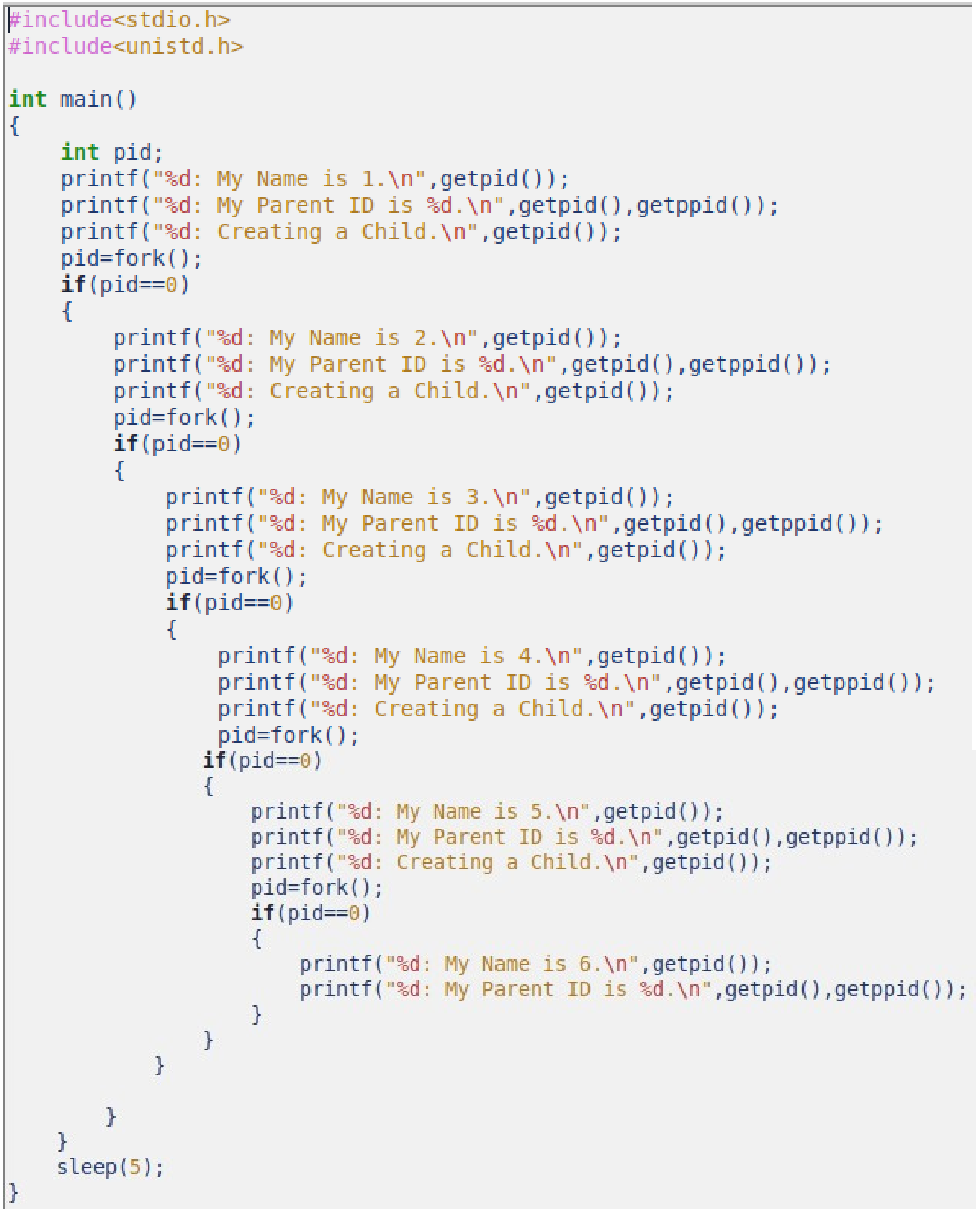


Task # 05:

Create process chain as shown in figure 1(b) and fill the figure 1 (b) with actual IDs.

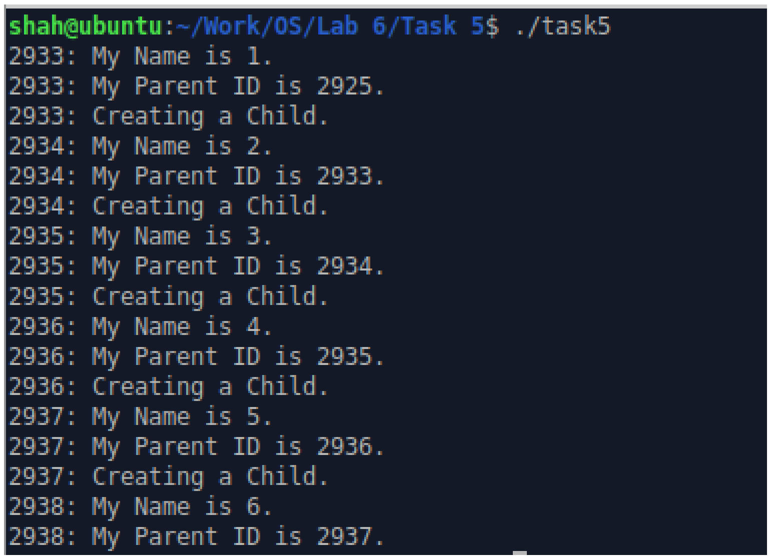


Code:



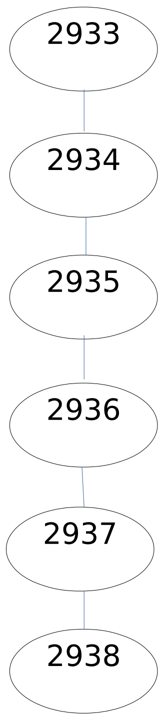
```
#include<stdio.h>
#include<unistd.h>

int main()
{
    int pid;
    printf("%d: My Name is 1.\n",getpid());
    printf("%d: My Parent ID is %d.\n",getpid(),getppid());
    printf("%d: Creating a Child.\n",getpid());
    pid=fork();
    if(pid==0)
    {
        printf("%d: My Name is 2.\n",getpid());
        printf("%d: My Parent ID is %d.\n",getpid(),getppid());
        printf("%d: Creating a Child.\n",getpid());
        pid=fork();
        if(pid==0)
        {
            printf("%d: My Name is 3.\n",getpid());
            printf("%d: My Parent ID is %d.\n",getpid(),getppid());
            printf("%d: Creating a Child.\n",getpid());
            pid=fork();
            if(pid==0)
            {
                printf("%d: My Name is 4.\n",getpid());
                printf("%d: My Parent ID is %d.\n",getpid(),getppid());
                printf("%d: Creating a Child.\n",getpid());
                pid=fork();
               if(pid==0)
               {
                   printf("%d: My Name is 5.\n",getpid());
                   printf("%d: My Parent ID is %d.\n",getpid(),getppid());
                   printf("%d: Creating a Child.\n",getpid());
                   pid=fork();
                   if(pid==0)
                   {
                       printf("%d: My Name is 6.\n",getpid());
                       printf("%d: My Parent ID is %d.\n",getpid(),getppid());
                   }
               }
            }

        }
    }
    sleep(5);
}
```

Output :



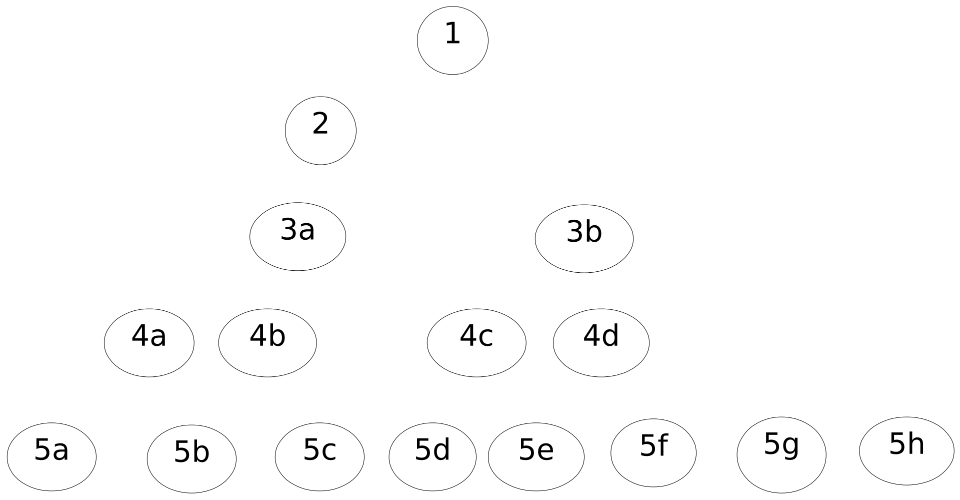
```
shah@ubuntu:~/Work/OS/Lab 6/Task 5$ ./task5
2933: My Name is 1.
2933: My Parent ID is 2925.
2933: Creating a Child.
2934: My Name is 2.
2934: My Parent ID is 2933.
2934: Creating a Child.
2935: My Name is 3.
2935: My Parent ID is 2934.
2935: Creating a Child.
2936: My Name is 4.
2936: My Parent ID is 2935.
2936: Creating a Child.
2937: My Name is 5.
2937: My Parent ID is 2936.
2937: Creating a Child.
2938: My Name is 6.
2938: My Parent ID is 2937.
```

Process Tree:

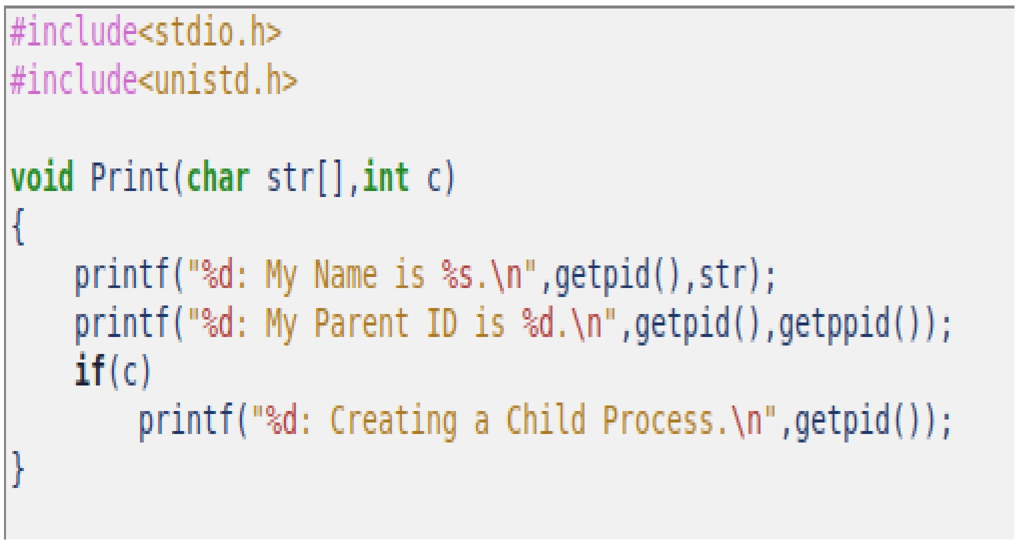


Task # 06:

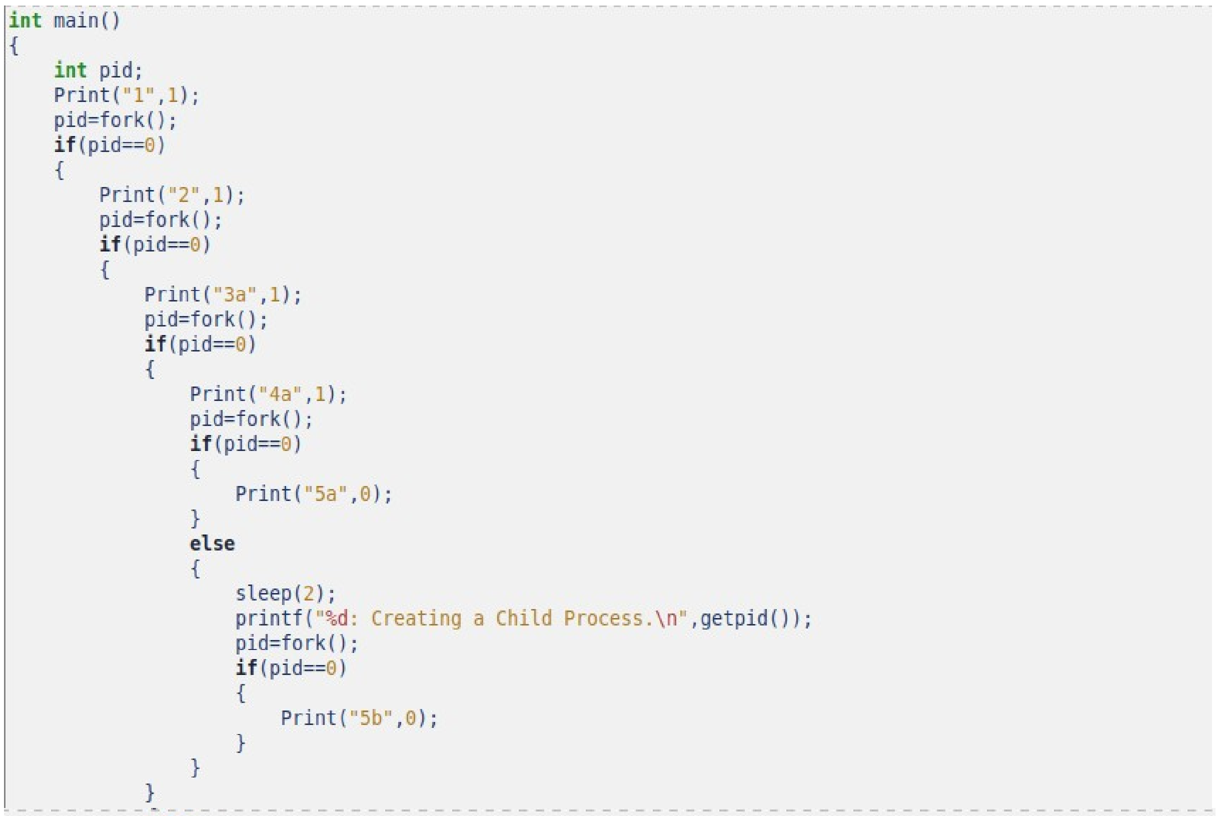
Create process tree as shown in figure 2 and fill the figure 2 with actual IDs.



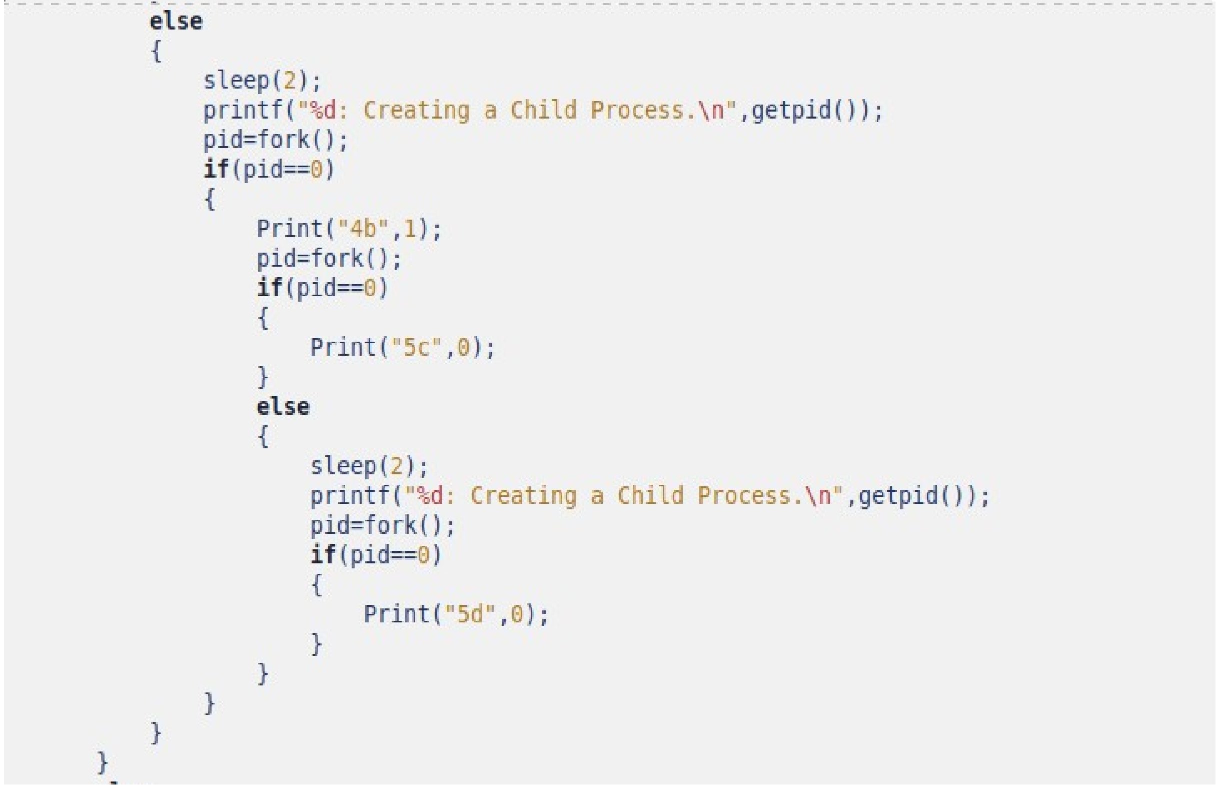
Code:



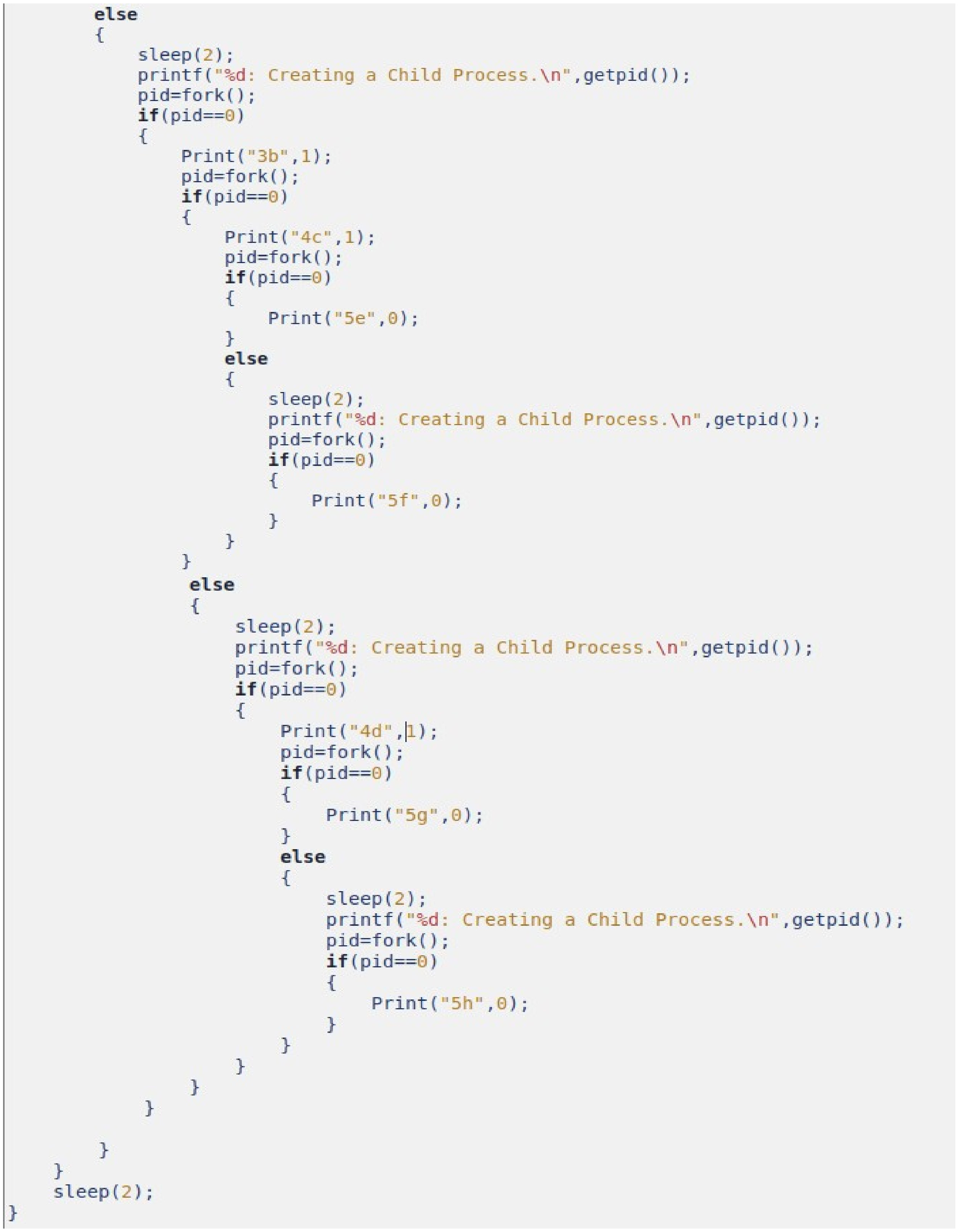
```
#include<stdio.h>
#include<unistd.h>

void Print(char str[],int c)
{
    printf("%d: My Name is %s.\n",getpid(),str);
    printf("%d: My Parent ID is %d.\n",getpid(),getppid());
    if(c)
        printf("%d: Creating a Child Process.\n",getpid());
}
```



```
int main()
{
    int pid;
    Print("1",1);
    pid=fork();
    if(pid==0)
    {
        Print("2",1);
        pid=fork();
        if(pid==0)
        {
            Print("3a",1);
            pid=fork();
            if(pid==0)
            {
                Print("4a",1);
                pid=fork();
                if(pid==0)
                {
                    Print("5a",0);
                }
                else
                {
                    sleep(2);
                    printf("%d: Creating a Child Process.\n",getpid());
                    pid=fork();
                    if(pid==0)
                    {
                        Print("5b",0);
                    }
                }
            }
```

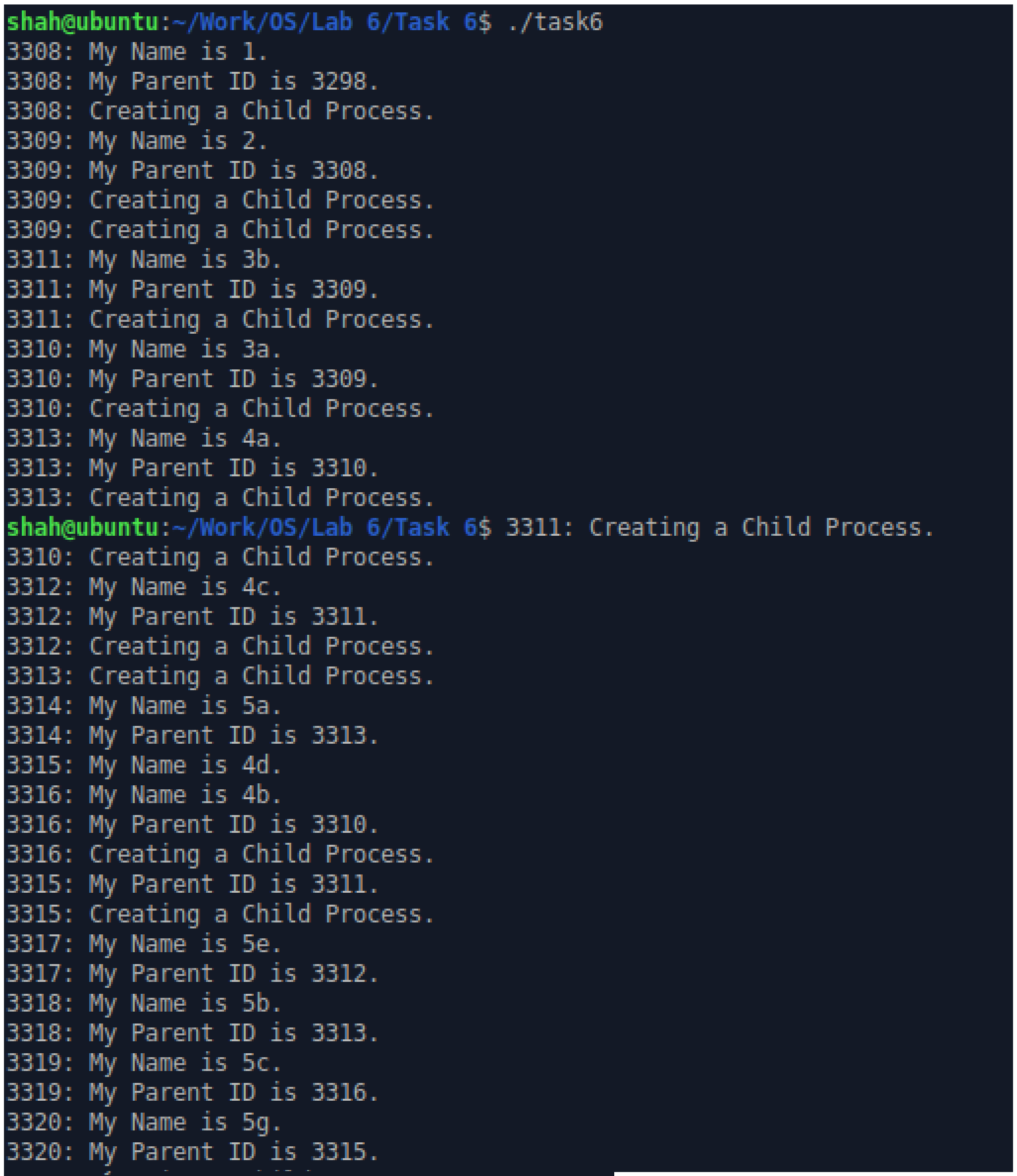


```
            else
            {
                sleep(2);
                printf("%d: Creating a Child Process.\n",getpid());
                pid=fork();
                if(pid==0)
                {
                    Print("4b",1);
                    pid=fork();
                    if(pid==0)
                    {
                        Print("5c",0);
                    }
                    else
                    {
                        sleep(2);
                        printf("%d: Creating a Child Process.\n",getpid());
                        pid=fork();
                        if(pid==0)
                        {
                            Print("5d",0);
                        }
                    }
                }
            }
        }
```

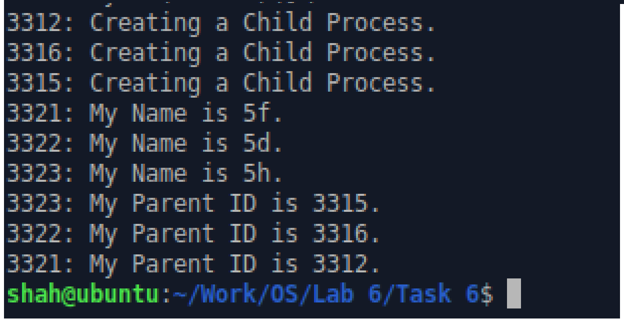


```c
        else
        {
            sleep(2);
            printf("%d: Creating a Child Process.\n",getpid());
            pid=fork();
            if(pid==0)
            {
                Print("3b",1);
                pid=fork();
                if(pid==0)
                {
                    Print("4c",1);
                    pid=fork();
                    if(pid==0)
                    {
                        Print("5e",0);
                    }
                    else
                    {
                        sleep(2);
                        printf("%d: Creating a Child Process.\n",getpid());
                        pid=fork();
                        if(pid==0)
                        {
                            Print("5f",0);
                        }
                    }
                }
                else
                {
                    sleep(2);
                    printf("%d: Creating a Child Process.\n",getpid());
                    pid=fork();
                    if(pid==0)
                    {
                        Print("4d",1);
                        pid=fork();
                        if(pid==0)
                        {
                            Print("5g",0);
                        }
                        else
                        {
                            sleep(2);
                            printf("%d: Creating a Child Process.\n",getpid());
                            pid=fork();
                            if(pid==0)
                            {
                                Print("5h",0);
                            }
                        }
                    }
                }
            }

        }
    }
    sleep(2);
}
```

Output :

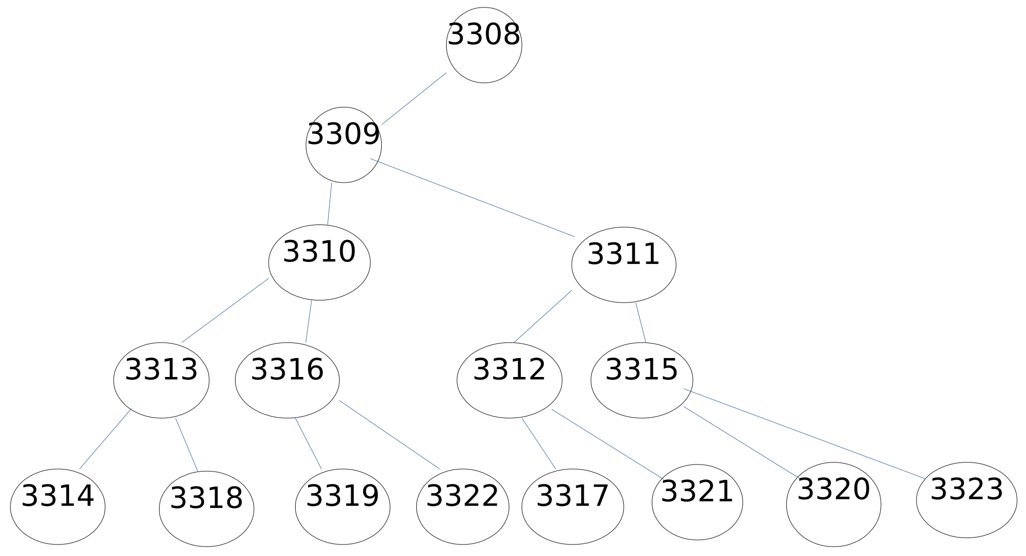


```
shah@ubuntu:~/Work/OS/Lab 6/Task 6$ ./task6
3308: My Name is 1.
3308: My Parent ID is 3298.
3308: Creating a Child Process.
3309: My Name is 2.
3309: My Parent ID is 3308.
3309: Creating a Child Process.
3309: Creating a Child Process.
3311: My Name is 3b.
3311: My Parent ID is 3309.
3311: Creating a Child Process.
3310: My Name is 3a.
3310: My Parent ID is 3309.
3310: Creating a Child Process.
3313: My Name is 4a.
3313: My Parent ID is 3310.
3313: Creating a Child Process.
shah@ubuntu:~/Work/OS/Lab 6/Task 6$ 3311: Creating a Child Process.
3310: Creating a Child Process.
3312: My Name is 4c.
3312: My Parent ID is 3311.
3312: Creating a Child Process.
3313: Creating a Child Process.
3314: My Name is 5a.
3314: My Parent ID is 3313.
3315: My Name is 4d.
3316: My Name is 4b.
3316: My Parent ID is 3310.
3316: Creating a Child Process.
3315: My Parent ID is 3311.
3315: Creating a Child Process.
3317: My Name is 5e.
3317: My Parent ID is 3312.
3318: My Name is 5b.
3318: My Parent ID is 3313.
3319: My Name is 5c.
3319: My Parent ID is 3316.
3320: My Name is 5g.
3320: My Parent ID is 3315.
```



```
3312: Creating a Child Process.
3316: Creating a Child Process.
3315: Creating a Child Process.
3321: My Name is 5f.
3322: My Name is 5d.
3323: My Name is 5h.
3323: My Parent ID is 3315.
3322: My Parent ID is 3316.
3321: My Parent ID is 3312.
shah@ubuntu:~/Work/OS/Lab 6/Task 6$
```

Process Tree:


3308
3309
3310
3311
3313
3316
3312
3315
3314
3318
3319
3322
3317
3321
3320
3323