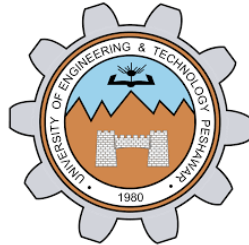


Operating Systems Lab-5

Process Creation and Execution



Submitted By: Awais Saddiqui

Registration# 21pwcse1993

Section: "A"

Submitted to:

Mam Madiha Sher

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar.

CSE 302L: Operating Systems Lab

LAB ASSESSMENT RUBRICS

Marking Criteria	Exceeds expectation (2.5)	Meets expectation (1.5)	Does not meet expectation (0)	Score
1. Correctness	Program compiles (no errors and no warnings). Program always works correctly and meets the specification(s). Completed between 81-100% of the requirements.	Program compiles (no errors and some warnings). Some details of the program specification are violated, program functions incorrectly for some inputs. Completed between 41-80% of the requirements.	Program fails to or compile with lots of warnings. Program only functions correctly in very limited cases or not at all. Completed less than 40% of the requirements.	
2. Delivery	Delivered on time, and in correct format (disk, email, hard copy etc.)	Not delivered on time, or slightly incorrect format.	Not delivered on time or not in correct format.	
3. Coding Standards	Proper indentation, whitespace, line length, wrapping, comments and references.	Missing some of whitespace, line length, wrapping, comments or references.	Poor use of whitespace, line length, wrapping, comments and references.	
4. Presentation of document	Includes name, date, and assignment title. Task titles, objectives, output screenshots included and good formatting and excellently organized.	Includes name, date, and assignment title. Task titles, objectives, output screenshots included and good formatting.	No name, date, or assignment title included. No task titles, no objectives, no output screenshots, poor formatting.	

Instructor:

Name: Engr. Madiha Sher

Signature: _____

Operating Systems Lab

What is a process?:

A process is basically a single running program. It may be a “system” program (e.g. login, update, csh) or program initiated by the user (pico, a.exe or a user written one).

Attributes of Process:

- ❖ some code
- ❖ some data
- ❖ a stack
- ❖ a unique process id number (PID)

Zombie Process:

A process that is waiting for its parent to accept its return code is called a zombie process.

Orphan Process:

If a parent dies before its child, the child (orphan process) is automatically adopted by the original “init” process whose PID is 1.

Objectives:

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 #1:

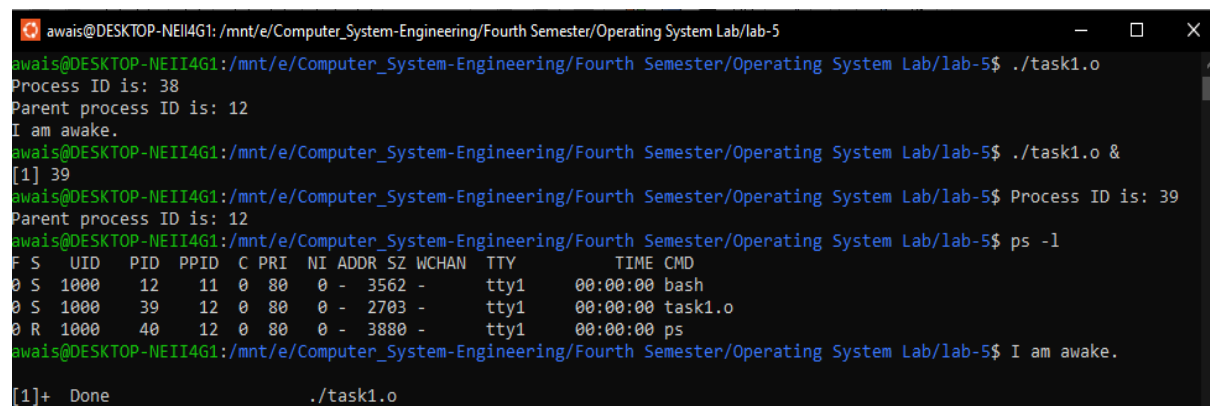
Run the following program twice. Both times as a background process, i.e., suffix it with an ampersand "&". Once both processes are running as background processes, view the process table using `ps -l` UNIX command.

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 am awake. \n");

    return();
}
```

Output:



```
awais@DESKTOP-NEII4G1: /mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5
awais@DESKTOP-NEII4G1:/mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ ./task1.o
Process ID is: 38
Parent process ID is: 12
I am awake.
awais@DESKTOP-NEII4G1:/mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ ./task1.o &
[1] 39
awais@DESKTOP-NEII4G1:/mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ Process ID is: 39
Parent process ID is: 12
awais@DESKTOP-NEII4G1:/mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ  WCHAN  TTY          TIME CMD
0 S  1000   12   11  0  80   0 - 3562 -      tty1      00:00:00 bash
0 S  1000   39   12  0  80   0 - 2703 -      tty1      00:00:00 task1.o
0 R  1000   40   12  0  80   0 - 3880 -      tty1      00:00:00 ps
awais@DESKTOP-NEII4G1:/mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ I am awake.
[1]+  Done                  ./task1.o
```

Explanation:

The functions from the `unistd.h` library `getpid()` and `getppid()` give process ID and parent process ID respectfully. It then waits for 60 seconds using the `sleep()` function before printing "I am awake." Finally, it returns 0 from the `main()` function.

Task #2:

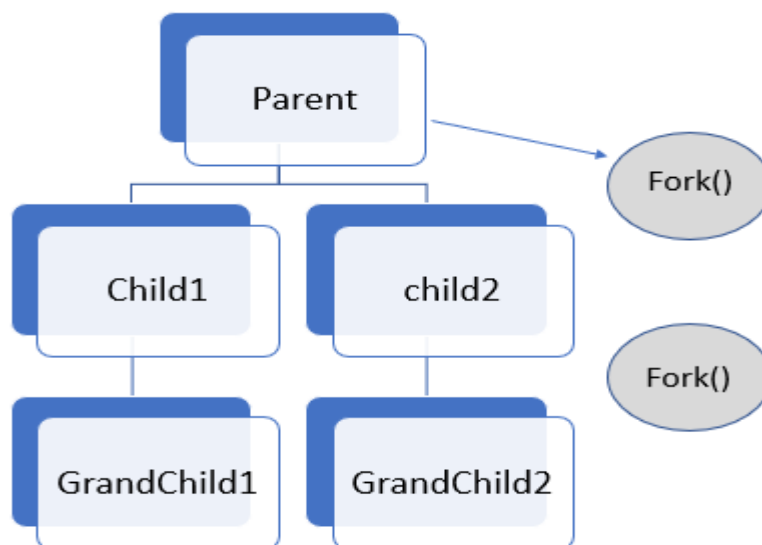
Run the following program and observe the number of times and the order in which the print statement is executed.

Code:

```
#include <stdio.h>
#include <unistd.h>
int main(void) {
    fork( );
    fork( );
    printf("Parent Process ID is: %d\n", getppid( )) ;
}
```

Output:

```
awais@DESKTOP-NEII4G1: /mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5
awais@DESKTOP-NEII4G1: /mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ ./task2.o
Parent Process ID is: 12
Parent Process ID is: 59
Parent Process ID is: 59
Parent Process ID is: 60
awais@DESKTOP-NEII4G1: /mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$
```



Explanation:

This program creates a total of 4 processes through two fork() calls. The first fork() call creates a child process which also calls fork(), creating a grandchild process. The second fork() call creates another child process which also calls fork(), creating another grandchild process. The printf() statement is executed by all four processes. It prints the Parent Process ID using getppid() function.

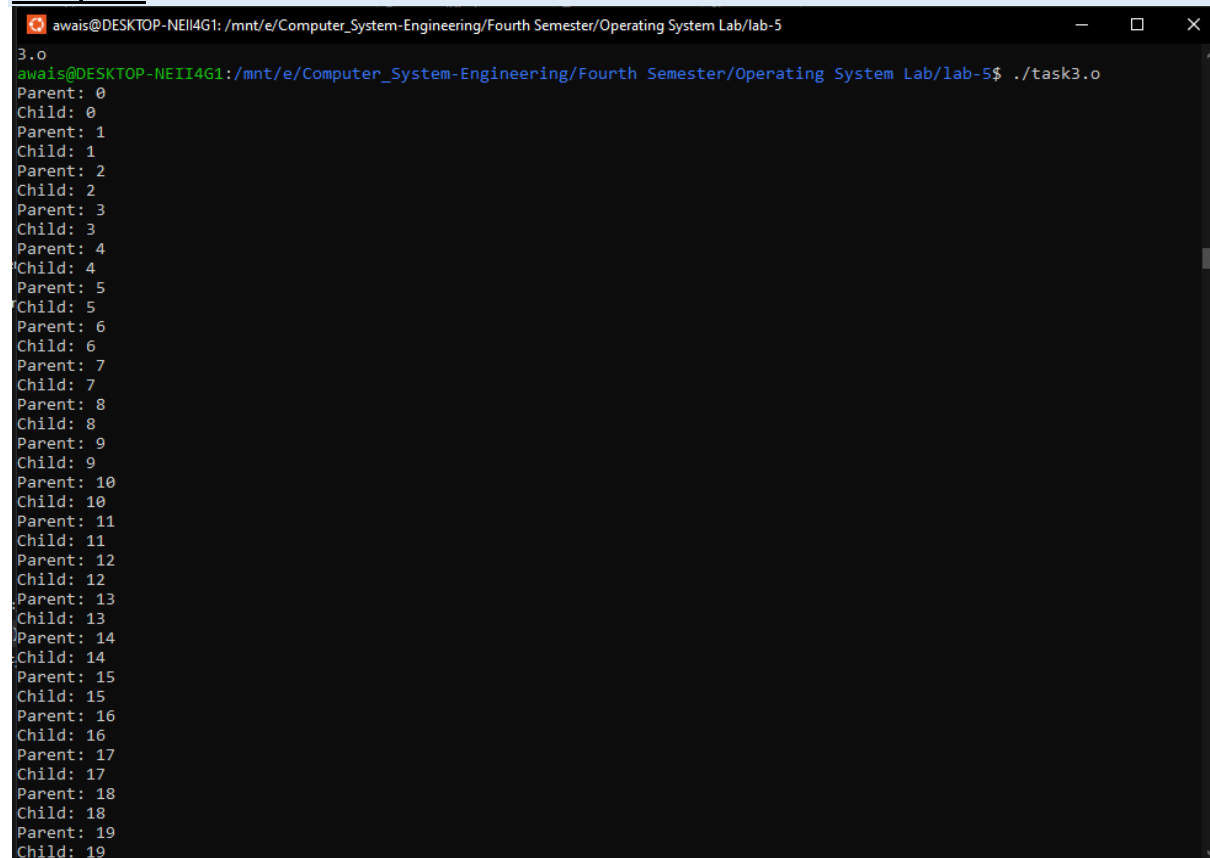
Task #3:

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

Code:

```
#include <stdio.h>
#include <unistd.h>
int main(void){
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:



```
awsais@DESKTOP-NEII4G1: /mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5
3.o
awsais@DESKTOP-NEII4G1: /mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ ./task3.o
Parent: 0
Child: 0
Parent: 1
Child: 1
Parent: 2
Child: 2
Parent: 3
Child: 3
Parent: 4
Child: 4
Parent: 5
Child: 5
Parent: 6
Child: 6
Parent: 7
Child: 7
Parent: 8
Child: 8
Parent: 9
Child: 9
Parent: 10
Child: 10
Parent: 11
Child: 11
Parent: 12
Child: 12
Parent: 13
Child: 13
Parent: 14
Child: 14
Parent: 15
Child: 15
Parent: 16
Child: 16
Parent: 17
Child: 17
Parent: 18
Child: 18
Parent: 19
Child: 19
```

Task #4:

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

Code:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
int main(void){
    int pid;
    printf("Parent id is : %d\n",getpid());
    for(int i=1; i<=8; i++){
        pid = fork();
        if(pid==0){
            break;
        }
    }
    for(int j=0; j<8; j++){
        if(pid>0){
            int ret= wait(NULL);
            printf("Child  %d and parent is %d\n",get, getpid());
        }
    }
}
```

Output:

```
awais@DESKTOP-NEII4G1:/mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ ./task4.o
Parent id is : 80
Child  81 and parent is 80
Child  82 and parent is 80
Child  83 and parent is 80
Child  84 and parent is 80
Child  85 and parent is 80
Child  86 and parent is 80
Child  87 and parent is 80
Child  88 and parent is 80
```

Task #5:

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>
#include <sys/wait.h>
int main(void){
    int pid;
    printf("Parent id is : %d and my parent is: %d\n",getpid(), getppid());
    for(int i=1; i<7; i++){
        pid = fork();
        if(pid>0){
            break;
        }
    }
    for(int j=0; j<7; j++){
        if(pid==0){
            int ret= wait(NULL);
            printf("Child :  %d and parent is : %d\n",
getpid(),getppid());
        }
    }
}
```

Output:

```
awais@DESKTOP-NEI14G1:/mnt/e/Computer_System-Engineering/Fourth Semester/Operating System Lab/lab-5$ Child : 137 and parent is : 136
Child : 137 and parent is : 1
Child : 137 and parent is : 1
Child : 137 and parent is : 1
Child : 137 and parent is : 1
Child : 137 and parent is : 1
Child : 137 and parent is : 1
```
