OPERATING SYSTEM LABORATORY MANUAL

UNIVERSITY OF THE PUNJAB

FACULTY OF COMPUTING & INFORMATION TECHNOLOGY, LAHORE

DEPARTMENT OF COMPUTER SCIENCE

Course:	**Operating System Lab**	**Date:**
Course Code:	**CC-217-3L**	**Max Marks: 40**
Faculty/Instructor's Name & Email:	**Dr. Ahmad Hassan Butt (ahmad.hassan@pucit.edu.pk)**	

LAB MANUAL # 9

(SPRING 2023)

Name:______________________________________ Enroll No: ____________________________

Objective(s) :

To write a program to create a process in LINUX. To create child with sleep command. To understand getpid( ) and getppid( ).

Lab Tasks :

Task 1 : Write the output of program for process creation using fork command.

Task 2: Write the output of a program for execution of ls command using exec.

Task 3 : Write the output of a program illustrating the sleep command during process creation.

Task 4 : Write the output of the program for getting the pid and ppid while using the sleep command.

Lab Grading Sheet :

Task	Max Marks	Obtained Marks	Comments(*if any*)
1.	10		
2.	10		
3.	10		
4.	10		
Total	**40**		**Signature**

Note : Attempt all tasks and get them checked by your Instructor

Lab 09: Processes

Objective(s):

- To write a program to create a process in LINUX.
- To create child with sleep command.
- To understand getpid( ) and getppid( ).

Tool(s) used:

Ubuntu, VIM Editor

Task 1 Write the output of a program for process creation using fork command.

Algorithm

STEP 1: Start the program.

STEP 2: Declare pid as integer.

STEP 3: Create the process using Fork command.

STEP 4: Check pid is less than 0 then print error else if pid is equal to 0 then execute command else parent process wait for child process.

STEP 5: Stop the program.

Program

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(){
    int id;
    id=fork();
    if(id<0){
            printf ("Cannot Create the file");
            exit(-1);
    }
    if(id==0){
            printf ("Child Process");
```

```
                exit(0);
        }
        else{
                printf ("Parent Process");
                exit(1);
        }
    return 0;
    }
```

Program Execution

```
$gcc pc.c –o pc

$./pc
```

OUTPUT

Task 2 Write the output of a program for execution of ls command using exec.

Algorithm

STEP 1: Start the program.

STEP 2: Execute the command in the shell program using exec ls.

STEP 3: Stop the execution.

Program

```
echo Program for executing LINUX command using Shell Programming
echo Welcome
ps
exec ls
```

OUTPUT

Task 3 Write the output of a program illustrating the sleep command during process creation.

Algorithm

STEP 1: Start the program.

STEP 2: Create process using fork and assign into a variable.

STEP 3: If the value of variable is < zero print not create and > 0 process create and else print child create.

STEP 4: Create child with sleep of 2.

STEP 5: Stop the program.

Program

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
int main( ){
        pid_t id;
        id=fork( );
        if (id==-1){
                printf ("Cannot Create the file");
                exit(1);
        }
        if (id==0){
                sleep(20);
                printf ("This is child Process");
        }
        else{
                printf ("Parent Process");
                exit(1);
        }
return 0;
}
```

OUTPUT

Task 4 Write the output of the program for getting the pid and ppid while using the sleep command.

Algorithm

STEP 1: Start the execution and create a process using fork( ) command.

STEP 2: Make the parent process to sleep for 10 seconds.

STEP 3: In the child process print it pid and it corresponding pid.

STEP 4: Make the child process to sleep for 5 seconds.

STEP 5: Again print it pid and it parent pid.

STEP 6: After making the sleep for the parent process for 10 seconds print it pid.

STEP 7: Stop the execution.

Program

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
        pid_t pid;
        pid=fork();
        if(pid==0){
                printf("\nChild Process");
                printf("\nChild Process ID is %d", getpid());
                printf("\nIts Parent Process ID is %d", getppid());
                sleep(5);
                printf("\nChild Process after sleep=5");
                printf("\nChild Process ID is %d", getpid());
                printf("\nParent Process ID is %d", getppid());
        }
        else{
                printf("\n\nParent Process\n");
                sleep(5);
                printf("\nChild Process ID is %d", getpid());
                printf("\nIts Parent  Process ID is %d", getppid());
                printf("\nParent Terminates\n");
        }
return 0;
}
```

OUTPUT