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 # 10

(SPRING 2023)

Name:_____ Enroll No: _____

Objective(s) :

To write a program for signal handling in LINUX. To use wait command using c program. To use wait command using c program and creating a separate process using exec( ). To create a Zombie Process.

Lab Tasks :

Task 1 :. Write the output of the program illustrating the Kill Command.

Task 2: Write the output of the program implementing the Wait signal.

Task 3 : Write the output of the program for wait signal using exec( ).

Task 4 : Write the output of the program for a Zombie Process.

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 10: Signal Handling

Objective(s):

- To write a program for signal handling in LINUX,
- To use wait command using c program,
- To use wait command using c program and creating a separate process using exec( )
- To create a Zombie Process.

Tool(s) used:

Ubuntu, VIM Editor

Task 1 Write the output of the program illustrating the Kill Command.

Algorithm

STEP 1:Start the program

STEP 2:Read the value of pid.

STEP 3:Kill the command surely using kill-9 pid.

STEP 4: Stop the program.

Program

```
echo program for performing KILL operations
ps

echo enter the pid
read pid

kill -9 $pid
```

OUTPUT

Task 2 Write the output of the program implementing the Wait signal.

Algorithm

STEP 1: Start the execution

STEP 2: Create process using fork and assign it to a variable

STEP 3: Check for the condition pid is equal to 0

STEP 4: If it is true print the value of i and terminate the child process

STEP 5: If it is not a parent process has to wait until the child terminate

STEP 6: Stop the execution

Program

```
main( ){

   pid_t pid;

     int i=10;

     pid=fork();

     if(pid==0){

            printf("initial value of i %d \n ",i);

            i+=10;

            printf("value of i %d\n ",i);
            printf("child terminated \n");}

      else{

            wait(NULL);

            printf("value of i in parent process %d",i);

      }

    return 0;

}
```

OUTPUT

Task 3 Write the output of the program for a Zombie Process.

Algorithm

STEP 1: Start the execution

STEP 2: Create process using fork and assign it to a variable

STEP 3: Check for the condition pid is equal to 0

STEP 4: If it is true print Child Process

STEP 5: If it is not a parent process has to wait until the child terminates

STEP 6: Parent process should print Parent Process.

STEP 7: Stop the execution

Program

```
int main( ){
    pid_t pid;
    pid=fork( );
    if(pid==0){
            printf("Child Process");}
    else{
            sleep(10);
            wait(NULL);
            printf("Parent Process");}
    return 0; }
```

OUTPUT

Task 4 Write the output of the program for wait signal using exec( )

Algorithm

STEP 1: Start the execution

STEP 2: Create process using fork and assign it to a variable

STEP 3: Check for the condition pid is equal to 0

STEP 4: If it is true print the commands to be executed by ls.

STEP 5: If it is not a parent process has to wait until the child terminates

STEP 6: Parent process should print Child Process completed.

STEP 7: Stop the execution

Program

```
int main( ){
    pid_t pid;
    if(pid<0){
            fprintf(stderr,  "Fork Failed");
            return 1;}
       else if(pid ==0){
            execlp("/bin/ls", "ls",NULL);}
       else{
            wait(NULL);
            printf("Child Completes");}
       return 0;}
```

OUTPUT