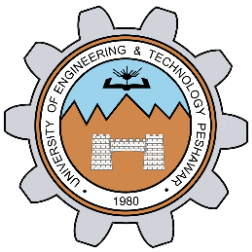
Processes in UNIX

LAB # 03



Fall 2024

CSE-302L

Systems Programming Lab

Submitted by: **AIMAL KHAN.**

Registration No.: **21PWCSE2014**

Class Section: **A**

"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. Abdullah Hamid

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

CSE 302L: SYSTEMS PROGRAMMING LAB

LAB ASSESSMENT RUBRICS

Criteria & Point Assigned	Outstanding 2	Acceptable 1.5	Considerable 1	Below Expectations 0.5	Score
Attendance and Attentiveness in Lab PLO08	Attended in proper Time and attentive in Lab	Attended in proper Time but not attentive in Lab	Attended late but attentive in Lab	Attended late not attentive in Lab	
Capability of writing Program/Algorithm/Drawing Flow Chart PLO1, PLO2, PLO3, PLO5	Right attempt/ no errors and well formatted	Right attempt/ no errors but not well formatted	Right attempt/ minor errors and not well formatted	Wrong attempt	
Result or Output/ Completion of target in Lab PLO9	100% target has been completed and well formatted.	75% target has been completed and well formatted.	50% target has been completed but not well formatted.	None of the outputs are correct.	
Overall, Knowledge PLO10,	Demonstrates excellent knowledge of lab	Demonstrates good knowledge of lab	Has partial idea about the Lab and procedure followed	Has poor idea about the Lab and procedure followed	
Attention to Lab Report PLO4,	Submission of Lab Report in Proper Time i.e., in next day of lab, with proper documentation.	Submission of Lab Report in proper time but not with proper documentation.	Late Submission with proper documentation.	Late Submission very poor documentation	

Instructor:

Name: _____	Signature: _____

Processes in UNIX

Objectives:

In this lab we will learn about the creation of fork, wait function call.

Tasks:

Task 1: Create process chain as shown in figure 3.1(b) and fill the figure 3.1 (b) with actual IDs. The program shall take a single command-line argument that specifies the number of processes to be created. Before exiting, each process shall output its i value (loop variable), its process ID (using getpid()), its parent process ID (getppid()) and the process ID of its child (return value of fork). The parent does not execute wait.

Code:

```
sarwat@DESKTOP-VG976N9:
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(int argc, char *argv[])
{
        int x;
        for (int i=0;i<atoi(argv[1]);i++)
        {
                x= fork();

                if(x>0)
                {
                        printf("Pid: %d,  Parent id: %d,  Child id: %d, i: %d\n",getpid(),getppid(),x,i);
                        break;
                }
        }
        for(int i=0;i<atoi(argv[1]);i++)
                wait(NULL);
        sleep(15);

}
```

Output:

```
sarwat@DESKTOP-VG976N9:~/lab3$ ./t1.o 4
Pid: 15859,  Parent id: 19863,  Child id: 15862, i: 0
Pid: 15862,  Parent id: 15859,  Child id: 15863, i: 1
Pid: 15863,  Parent id: 15862,  Child id: 15865, i: 2
Pid: 15865,  Parent id: 15863,  Child id: 15867, i: 3
^C
```

Task 2: Create process fan as shown in figure 3.1 (a) and fill the figure 3.1 (a) with actual IDs.

Code:

sarwat@DESKTOP-VG976N9:

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(int argc, char *argv[])
{
        int x;
        for (int i=0;i<atoi(argv[1]);i++)
        {
                x= fork();
                if(x==0)
                        break;
        }
        for(int i=0;i<atoi(argv[1]);i++)
                wait(NULL);
        printf("Pid: %d, Parent id: %d\n",getpid(),getppid());
        sleep(15);

}
```

Output:

```
sarwat@DESKTOP-VG976N9:~/lab3$ gcc t2.c -o t2.o
sarwat@DESKTOP-VG976N9:~/lab3$ ./t2.o 7 &
[1] 15057
sarwat@DESKTOP-VG976N9:~/lab3$ Pid: 15064, Parent id: 15057
Pid: 15065, Parent id: 15057
Pid: 15066, Parent id: 15057
Pid: 15067, Parent id: 15057
Pid: 15069, Parent id: 15057
Pid: 15070, Parent id: 15057
Pid: 15072, Parent id: 15057
```

Task3:

Create process tree as shown in figure 3.2 and fill figure 3.2 with actual IDs.

Code:

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(int argc, char *argv[])
{
        int x;
        for (int i=0;i<atoi(argv[1]);i++)
        {
                x= fork();
        }
        for(int i=0;i<atoi(argv[1]);i++)
                wait(NULL);
        printf("Pid: %d, Parent id: %d\n",getpid(),getppid());
        sleep(30);
}
```

Output:

```
sarwat@DESKTOP-VG976N9:~/lab3$ gcc task3.c -o task3.o
sarwat@DESKTOP-VG976N9:~/lab3$ ./task3.o 4
Pid: 52, Parent id: 47
Pid: 56, Parent id: 49
Pid: 55, Parent id: 50
Pid: 58, Parent id: 48
Pid: 59, Parent id: 54
Pid: 60, Parent id: 51
Pid: 61, Parent id: 53
Pid: 62, Parent id: 57
Pid: 54, Parent id: 48
Pid: 50, Parent id: 47
Pid: 53, Parent id: 49
Pid: 57, Parent id: 51
cPid: 49, Parent id: 47
Pid: 51, Parent id: 48
```

Task5:

Write a program that takes N number of integers as argument and displays the factorials of N integers (print 1 only if integers are not less than zero, 0 or 1). Create separate child process for each integer. Make sure no child is orphan/zombie.

Code:

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>

void Factors(int val)
{
        for(int i=1;i<val;i++)
        {
                if(val%i==0)
                        printf(" %d ",i);
        }
        printf("\n");
}

int main(int argc, char *argv[])
{
        int x;
        for (int i=1;i<argc;i++)
        {
                x= fork();
                if(x==0)
                {
                        printf("Pid: %d, Parent id: %d\nFactors of %d: ",getpid(),getppid(),atoi(argv[i]));
                        Factors(atoi(argv[i]));
                        break;
                }

        }
        for(int i=0;i<atoi(argv[1]);i++)
                wait(NULL);
}
```

Code:

```
sarwat@DESKTOP-VG976N9:~/lab3$ gcc t5.c -o t5.o
sarwat@DESKTOP-VG976N9:~/lab3$ ./t5.o 45 3 8
Pid: 8383, Parent id: 8379
Factors of 45:  1  3  5  9  15
Pid: 8384, Parent id: 8379
Factors of 3:  1
Pid: 8385, Parent id: 8379
Factors of 8:  1  2  4
sarwat@DESKTOP-VG976N9:~/lab3$
```

Conclusion:

In conclusion, I have learned about how to creation fork ,and wait function call.
