Operating Systems Lab Assignment – 1

Name – Mohd Rizwan

Course – B.Tech CSE (Data Science)

Roll No. – 2301420037

Task1- Process Creation Utility

Write a Python program that creates N child processes using os.fork(). Each child prints:
- Its PID
- Its Parent PID
- A custom message

The parent should wait for all children using os.wait().

Sol.-

```
┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# nano process_managment.py
```

```
root@VivobookPro15: /home/
  GNU nano 8.4                                process_managment.py
import os

def task1(n):
        for i in range(n):
                pid = os.fork()
                if pid == 0:
                        print(f'child {i + 1} = PID = {os.getpid()}, Parent PID = {os.getppid()}, Hello From Child')
                        os._exit(0)
        for i in range(n):
                os.wait()

task1(5)
```

```
┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# python process_managment.py
child 1 = PID = 359, Parent PID = 358, Hello From Child
child 2 = PID = 360, Parent PID = 358, Hello From Child
child 3 = PID = 361, Parent PID = 358, Hello From Child
child 4 = PID = 362, Parent PID = 358, Hello From Child
child 5 = PID = 363, Parent PID = 358, Hello From Child
```

Task 2- Command Execution Using exec()

Modify Task 1 so that each child process executes a Linux command (ls, date, ps, etc.) using os.execvp() or subprocess.run().

Sol.-

```python
import os
import time

def task2(commands):
        for cmd in commands:
                pid = os.fork()
                if pid == 0:
                        print(f'child PID = {os.getpid()} executing: {' '.join(cmd)}', flush = True)
                        os.execvp(cmd[0], cmd)
                        os._exit(1)
                else:
                        time.sleep(1)
        for i in commands:
                os.wait()

task2([['ls'], ['date'], ['ps', '-el']])
```

```
┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# python process_managment.py
child PID = 418 executing: ls
process_managment.py
child PID = 419 executing: date
Mon Sep 15 02:55:52 PM IST 2025
child PID = 420 executing: ps -el
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 -  5820 -      ?        00:00:00 systemd
5 S     0       2       1  0  80   0 -   694 x64_sy ?        00:00:00 init-systemd(ka
0 S     0       7       2  0  80   0 -   694 -      ?        00:00:00 init
4 S     0      39       1  0  80   0 - 12713 -      ?        00:00:00 systemd-journal
4 S     0      53       1  0  80   0 -  7892 -      ?        00:00:00 systemd-udevd
4 S     0     173       1  0  80   0 -  1069 hrtime ?        00:00:00 cron
4 S   992     175       1  0  80   0 -  1705 -      ?        00:00:00 dbus-daemon
4 S     0     179       1  0  80   0 -  4657 -      ?        00:00:00 systemd-logind
4 S     0     219       1  0  80   0 -  1303 core_s hvc0     00:00:00 agetty
4 S     0     220       1  0  80   0 -  1292 core_s tty1     00:00:00 agetty
5 S     0     227       2  0  80   0 -   695 -      ?        00:00:00 SessionLeader
5 S     0     228     227  0  80   0 -   695 x64_sy ?        00:00:00 Relay(229)
4 S  1000     229     228  0  80   0 -  2263 do_wai pts/0    00:00:00 bash
4 S     0     230       2  0  80   0 -  1820 do_wai ?        00:00:00 login
4 S  1000     240       1  0  80   0 -  5494 -      ?        00:00:00 systemd
5 S  1000     244     240  0  80   0 -  5583 -      ?        00:00:00 (sd-pam)
4 S  1000     254     230  0  80   0 -  1499 core_s pts/1    00:00:00 bash
4 S     0     264     229  0  80   0 -  4899 x64_sy pts/0    00:00:00 sudo
1 S     0     268     264  0  80   0 -  4899 x64_sy pts/2    00:00:00 sudo
4 S     0     269     268  0  80   0 -  2523 do_wai pts/2    00:00:00 su
4 S     0     273       1  0  80   0 -  5481 -      ?        00:00:00 systemd
5 S     0     275     273  0  80   0 -  5583 -      ?        00:00:00 (sd-pam)
4 S     0     284     269  0  80   0 -  2281 do_wai pts/2    00:00:00 bash
4 S     0     417     284  0  80   0 -  3611 hrtime pts/2    00:00:00 python
4 Z     0     418     417  0  80   0 -     0 -      pts/2    00:00:00 ls
4 Z     0     419     417  0  80   0 -     0 -      pts/2    00:00:00 date
4 R     0     420     417 99  80   0 -  2337 -      pts/2    00:00:00 ps
```

Task 3 - Zombie & Orphan Processes

Zombie: Fork a child and skip wait() in the parent.
Orphan: Parent exits before the child finishes.
Use ps -el | grep defunct to identify zombies.

Sol.-

```python
import os
import time

def zombie():
        pid = os.fork()
        if pid == 0:
                print(f'child (PID = {os.getpid()}) exiting immediately')
                os._exit(0)
        else:
                print(f'Parent (PID = {os.getppid}) not working -> child become zombie')
                time.sleep(5)
                os.wait()
                print("Parent: child reaped, zombie cleared")

zombie()
```

```
┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# python process_managment.py
Parent (PID = <built-in function getppid>) not working -> child become zombie
child (PID = 458) exiting immediately

Parent: child reaped, zombie cleared
```

```python
import os
import time

def orphan():
        pid = os.fork()
        if pid == 0:
                time.sleep(3)
                print(f'child (PID = {os.getpid()}) new Parent PID = {os.getppid()} (adopted by init)')
                os._exit(0)
        else:
                print(f'Parent (PID = {os.getppid}) not working -> child become orphan')
                os._exit(0)

orphan()
```

```
┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# python process_managment.py
Parent (PID = <built-in function getppid>) not working -> child become orphan

┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# child (PID = 500) new Parent PID = 228 (adopted by init)
```

Task 4 - Inspecting Process Info from /proc

Take a PID as input. Read and print:
- Process name, state, memory usage from /proc/[pid]/status
- Executable path from /proc/[pid]/exe
- Open file descriptors from /proc/[pid]/fd

Sol.-

```python
import os

def task4(pid):
        with open(f'/proc/{pid}/status') as f:
                for line in f:
                        if line.startswith(("Name :", "State:", "VmSize:")):
                                print(line.strip())
        print("Executable Path:", os.readlink(f"/proc/{pid}/exe"))
        print("Open FDs:", os.listdir(f"/proc/{pid}/fd"))

task4(os.getpid())
```

```
┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# python process_managment.py
State:  R (running)
VmSize:    14572 kB
Executable Path: /usr/bin/python3.13
Open FDs: ['0', '1', '2', '3']
```

Task 5 - Process Prioritization

Create multiple CPU-intensive child processes. Assign different nice() values. Observe and log execution order to show scheduler impact.

Sol.-

```
import os
import time

def cpu_task():
        x = 0
        for i in range(10**7):
                x += i

def task5():
        for nice_val in [0, 5, 10]:
                pid = os.fork()
                if pid == 0:
                        os.nice(nice_val)
                        print(f'child PID = {os.getpid()} with nice = {nice_val}')
                        cpu_task()
                        print(f'child PID = {os.getpid()} finished')
                        os._exit(0)
        for i in range(3):
                os.wait()

task5()
```

```
┌──(root㉿VivobookPro15)-[/home/rizwan/LabWork/Assignment_1]
└─# python process_managment.py
child PID = 539 with nice = 0
child PID = 540 with nice = 5
child PID = 541 with nice = 10
child PID = 539 finished
child PID = 540 finished
child PID = 541 finished
```