Lab Exam - 1

Create a process, named Mellow, having two child processes named 'Pillow' and 'Killow'.
All processes show their PIDs & CPU information in an infinite loop.
Mellow also shows its own child processes PIDs. Check two cases:

a) If Pillow and Killow are killed manually before Mellow.
b) If Mellow is killed manually before Pillow and Killow.

Output can be:
$./Mellow
Process-4035, parent of Pillow-4036 and Killow-4037, running at CPU-0.
I am child-4036 running at CPU-7.
I am child-4037 running at CPU-10.
Process-4035, parent of Pillow-4036 and Killow-4037, running at CPU-0.
I am child-4036 running at CPU-7.
Process-4035, parent of Pillow-4036 and Killow-4037, running at CPU-0.
I am child-4037 running at CPU-10.
I am child-4036 running at CPU-7.

```
$ pstree -p
├─gnome-terminal-(3284)─┬─bash(3292)───Mellow(4035)─┬─Killow(4037)
                        │                           └─Pillow(4036)
```

$ kill - 9 <PID>

Lab Exam - 2

1. Create a process, named PipeCreator, which can take 'n' number of names from the terminals and create 'n' named pipes.

2. Create two processes named 'Minu' and 'Binu' which communicate via a named pipe created by PipeCreator.

- Both processes take pipe names via terminals.
- At first Minu will send its PID, its Parent's PID, and a Hello message to Binu via the common named pipe.
- Binu will display all messages received from Minu via the common named pipe in its shell.
- After the introductory message Minu will send a message which the user will type in the terminal.
- Binu will send a message to Minu only once and that will be 'Bye'.
- Both Minu and Binu will terminate when they receive a 'Bye' message from the other side.
- For Minu, the user will type 'Bye' message.

- Binu will send a static 'Bye' message to 'Minu' after getting 'Bye' message from Minu.

Lab Exam - 3

Create a process, named MultiHead, having three threads, named Ittu, Bittu, and Mittu .

- Ittu, Bittu, and Mittu perform addition, subtraction and multiplication of two variables sent by the main thread of the process and keep the output in a global variable, result. All threads print their process ID, thread ID, CPU Info and the global variable, 'result'.
- The main thread waits until each thread finishes its task.
- The main thread reads and sends two integer values to each thread 100 times.
- If any thread finds that 'result' = 100, it immediately sends a signal to the main thread to stop the process.
- Take necessary steps to avoid thread racing.

Lab Exam - 4

Create a process, named Laltu, having three processes, named Ittu, Bittu, and Mittu .

- Laltu reads two variables 'a' and 'b' which all its child processes know somehow.
- Ittu, Bittu, and Mittu perform addition, subtraction and multiplication of two variables read by Laltu and keep the output in a variable, 'Result'.
- Results will be seen and accessed by all processes. That means all processes will see the updated value of 'Result.
- All processes print their process ID, CPU Info, and values of 'a' and 'b' before performing their corresponding operation using separate printf instruction. Then they perform their operations which they are supposed to do. Then each process will print 'Result'.
- Laltu will wait until Ittu, Bittu and Mittu finish their tasks.
- Take necessary steps to avoid inconsistency in 'Result'.

Lab Exam - 5

Create a process named, Galaxy, having two child processes, named Nebula and BlackHole.

- Nebula and BlackHole are in a 100 times loop.
- Main threads of all processes are in a 10 times loop.
- Galaxy waits for Nebula and Blackhole to be finished.
- Both Nebula and BlackHole have two threads, named X and Y.
- All threads of Galaxy, Nebula and BlackHole show their PID, logical CPU info, CPU_Affinity and scheduling algorithm's information.

- Check whether any threads and any processes change their logical processors during their execution.
- Force Blackhole's all threads run in different logical processors whenever they are in the running state.
- Force Nebula to use a scheduling algorithm other than the default scheduling algorithm.