Assignment 5

Analyse the output of multiple processes
updating/ reading a shared memory simultaneously

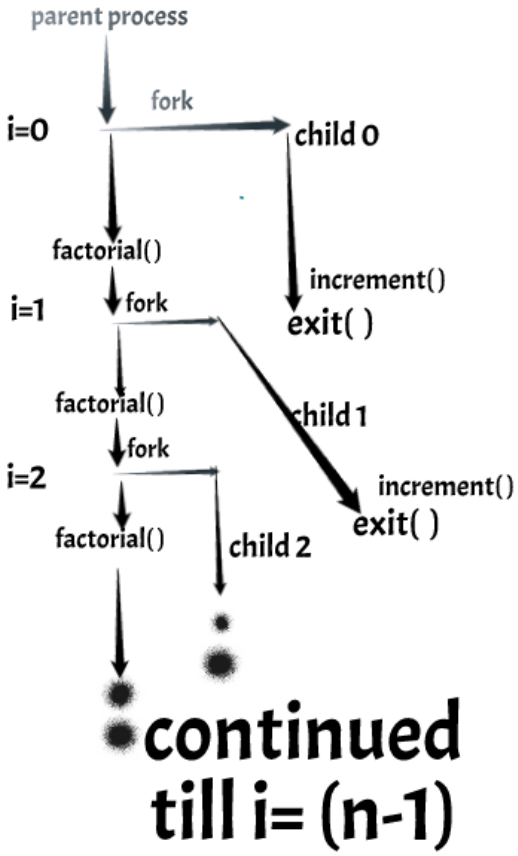
Overview :

- Create a shared memory , shared by (N + 1) processes
- N process : increment the shared variable
- One process : reads the variable, finds the factorial and print it
- Analyse the output

Program Flow :

```
for(i=0;i<n;i++)
    {
        mychild=fork();
        if(mychild==0)
        {
            increment();
            exit(EXIT_SUCCESS);
        }

        else
        {
            printfac(mychild);
        }
    }
```



Observation

- If all processes were to be executed in the process they were created , the flow of program would look like :
    - Child 1 → x=1
    - printfac( ) → 1! → 1
    - Child 2 → x=2
    - printfac( ) → 2! → 2
    - Child 3 → x=3
    - printfac( ) → 3! → 6
    - ....
- However, this is not the case as evident from the output(below); Even though the fork() is called sequentially for the n children, the children may not run sequentially ; the order of execution depends on the OS scheduler
- Also, printfac( ) may not be always executed ***after*** the execution of the previous child process. There may be cases where the factorial is calculated based on the previous value and afterwards the child process is executed.
- There may be interleaving between the child processes too.

P. T.O.

Sample output

```
./a.out 4
```

Part 1.

```
factorial called due to child no. 17420
    current value of x : 0
    factorial computed : 1
return
_____________________

factorial called due to child no. 17421
    current value of x : 0
    factorial computed : 1
return
_____________________

child : 17420
    prev x : 0
    new x : 1
exiting 17420
_____________________

factorial called due to child no. 17422
    current value of x : 1
    factorial computed : 1
return
_____________________
```

Part 2.

```
child : 17421
    prev x : 1
    new x : 2
exiting 17421
_____________________

factorial called due to child no. 17423
    current value of x : 2
    factorial computed : 2
return
_____________________

child : 17422
    prev x : 2
    new x : 3
child : 17423
```

next child starts before the current finishes (context switch)

```
exiting 17422
_____________________

    prev x : 3
    new x : 4
exiting 17423
_____________________
```