Monday, March 28, 2022 10:34 AM

Ready Q , Starvation , IR and PC , DMA , Wt , Multitasking, ps command ,
ps and ps -e command , **parent** process ID ppid = the process that created the current process(pid)
kill pid , ISR .

input device --------DMA---------->RAM ----------DMA-------->output device

---

fork ------ child process creation from parent process

parent process is duplicated , child and parent run in RR .
----------fork is creating copy… so both parent and child are running the same code

fork returns child-pid to the parent process
and fork returns 0 to child-process

P1 --- childpid = garbage
1st fork (P2) = childpid = 300
2nd fork (P3)
if is false
Else ---**I am parent my pid = 200 , ppid =bash**
P1 Ends

P2 ---- childpid = 0
2nd fork () (P4)
if(TRUE) ------ I am child my pid =300 , ppid =200
P2 ends

P3 ------- childpid = 300
ELSE ---- I am parent mypid is 350 ,ppid =200
P3 ends

P4 ---- childpid =0
IF I am child my pid is 400 ,ppid =300

---

4 processes

P1 parent
P2 ---------- create P4
P3 ---P3 is the child

To allow child and parent to have different codes --- use the RETURN value of fork() !!!

EX1---------- write a program that creates a single child using fork

The parent will show the maths table of 2 upto 1000

2*1=2

2*2=4

2*3 = 6

….

2*1000 = 2000

The child will show the fibonacci series till 1000

1

1

2

3

5

8

13

…

<1000

```
  if( cchpid == 0)
        {
              fibo
        }
If( chpid > 0)
        {
              Table
        }
```

P1 pid=garbage

1st fork pid = C1

IF TRUE ----

2nd fork pid = C2

If TRUE ---hello

P1 ends

C1 pid =0

ELSE --- hello

C2 pid =0

ELSE hi

FORK	EXEC
Copy the current process to create a child process	Child process TAKES over the parent process
TWO pids are seen ---- 1st Parent process id -- that calls fork 2nd Child process id ---- that is created by fork	ONLY ONE PID

_____

ORPHAN Process ------ CHILD outlives the PARENT !!!

Parent process is creating a child process

Parent process finishes EARLY

Child process continues to RUN ---- this is called as ORPHAN process --- Kernel cannot keep the ORPHAN process without a PPID, kernel will make the systemd or init process to adopt the orphan process .

Now at this point we try to see the PPID of the process --- the parent is no more, so pid of the parent who created is not there , then we see the pid of the init or systemd process that has adopted the ORPHAN process

---

Exec family of system calls -----

EX2 ---------- I want to run the **ls** command from the c program !!!!

EXEC FAMILY comes in many versions ---

1	execl	It accepts a coma separated list of arguments as parameters, it also needs the full **path** of where the command is present
2	execlp	It assumes a path of the command and it accepts a coma separated list of commands and command parameters
3	execv	It accepts an array of arguments as parameters, it also needs the full **path** of where the command is present
4	execvp	It assumes a path of the command and it accepts array of commands and command parameters
5	execle	---------
6	execlep	----------
