COPY ON WRITE

When ever a process is in execution mode, a process context is created. All varibales that are in a scope says LOCAL VARIABLE stores in stack segment of process context. When we create a new process by using **fork()** it become the child process of that parent process. Child process share the same process context of parent process. Parent process has its own PID and PPID so as child process a has its own PID and PPID. The PPID of parent process would be the PID of child process. Logical address of both parent and child would be same but the physical address of both parent and child would be different. When we initialize a local variable in parent process and trying to access the same variable in child process the value of that local variable won’t be updated because the physical addresses of both process is different. **Because parent and child process shares the same procces context in read-mode** so when a process want to update the value of local variable OS allots memory for individual variable and allocate different physical address that means we accessing two different variable from different process and from different physical address. The pid of the parent procces becomes ppid of child process and child pid becomes the next number if available.

For example

int num = 10; //For parent process

int num = 12; //For child process

When we create a child process using **fork()** call the value for **num** would be the same for parent process that is ‘10’ and ‘12’ for child process as mentioned above.

There is a 99% chances the return value of function fork() is equal to ‘0’ ie **fork() == 0** and it is happen when child process starts execution.

PARENT

PROCCESS

Ppid = 100

Pid = 101

CHILD

PROCCESS

Ppid = 101

Pid = 102

**Logical address = z**

The physical address of the parent(x) and child procces(y) is different but the logical(z) address of both process is different.

So **num** would be store at different logical address. When ever we try to access or update **num** from both process with different physical address then it won’t overwritten by updated value.

Address for parent process = x + z (num = 10)

Address for child process = y + z (num = 12)

So this is how COPY ON WRITE works