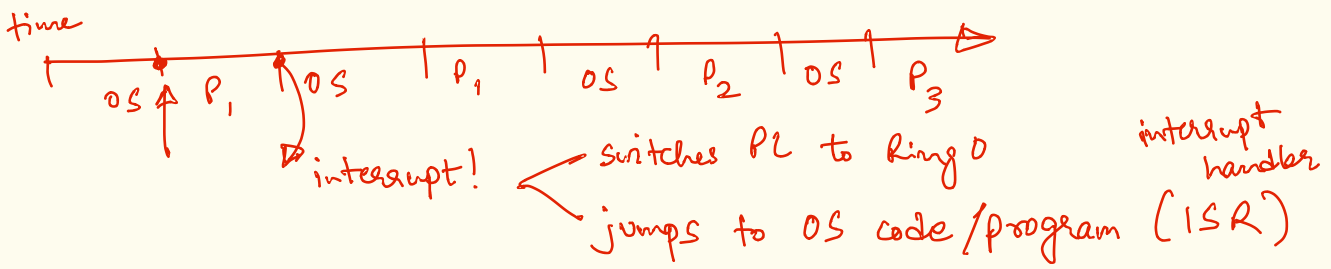
Lecture #4 CS 347 10. 8. 2023

recap:
(i) you are yourself
(ii) the world (?) is non-deterministic
(iii) world peace needs IO

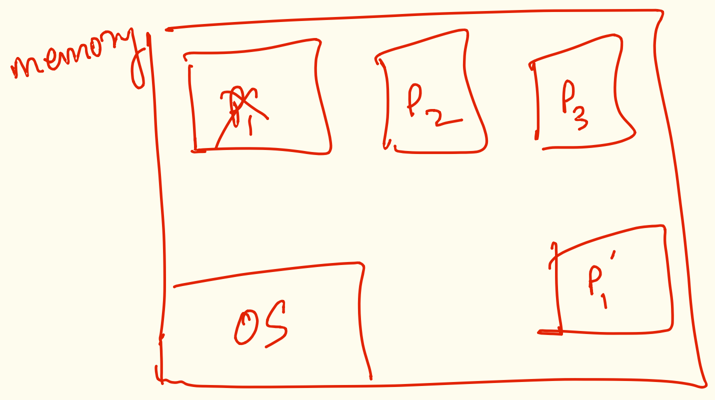
- LDE
- interrupts , process abstraction



(π) program vs process

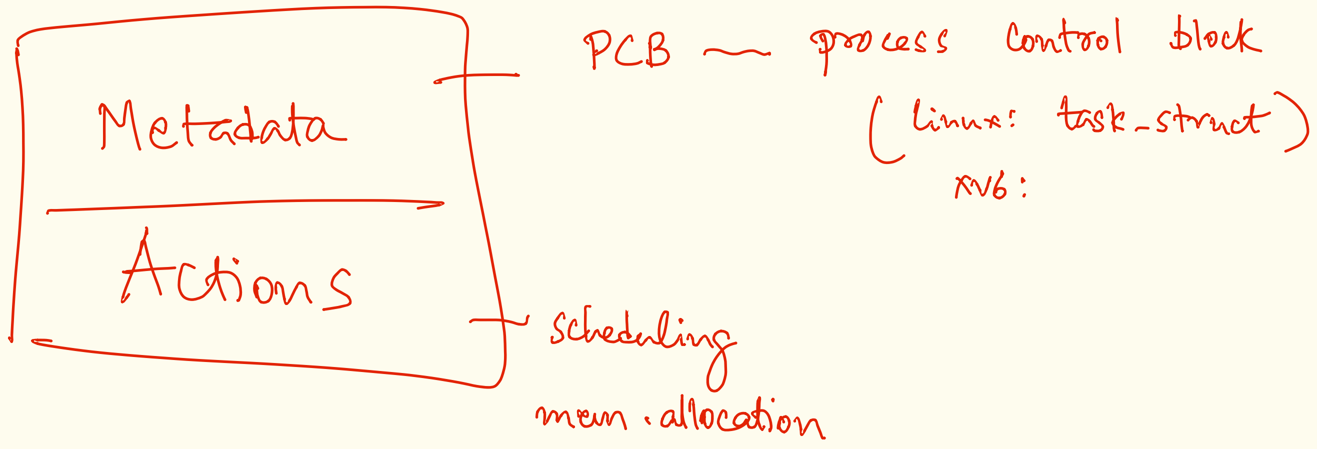
program: paper-weight of instructions

process:
- program in execution
- instance of a program
- entity to associate resources.



- load program in memory
- allocate memory to (program entity // process)
- schedule ⇒ PC points to instructions in memory

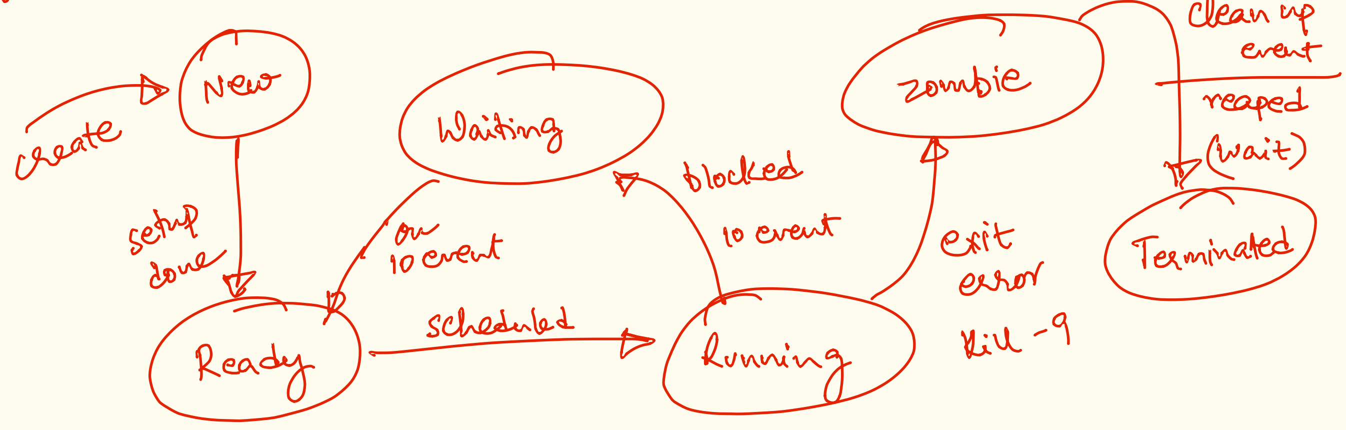
⊛ two-block description of all things OS.



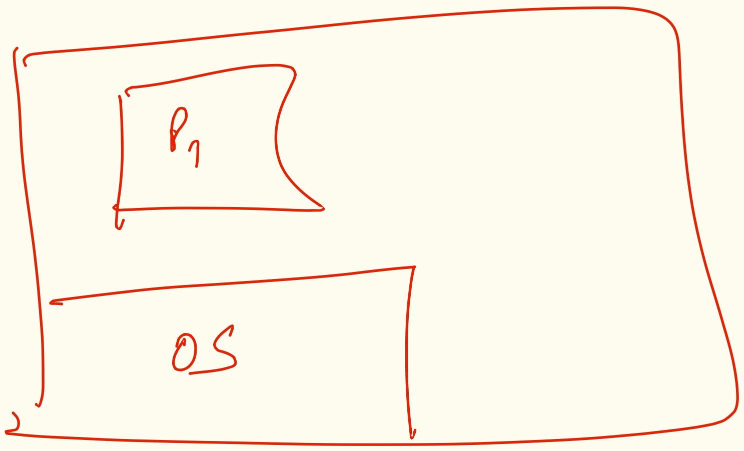
PCB contents:

- pid, ppid, state
- list of files
- memory allocation information
- space of registers
- Kernel stack ptr.
- usage information.

⊛ process state transition diagram.



(*) Memory



- game plan for an OS to have a purpose

step 1: as part of bootup, load itself (OS) in memory.

step 2: handcraft a (user-level) process & jmp to process, instructions (init process pid 1)
start of

step 3: consumer of OS services is in execution.

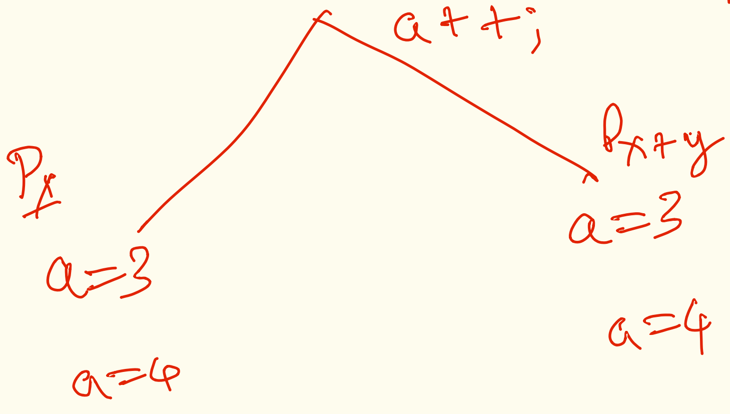
(*)

fork	exec	wait
creates (duplicates) a process - creates a new PCB - ~~&~~ populate PCB entries	- load program into memory & makes it part of a process	- return with a return value / status

```
int a=3;
fork();
a++;
```

in the parent process returns PID of the child

in the child process returns zero.



```
    int a=3;
──▷ if (fork()==0){ // child
            a=a+1;        ⇒ 4 || execv
            print(a);
    }
    else {
            a=a-1;        ⇒ 2
            print(a);
    }
```